МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ПРОЕКТУВАННЯ І РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ЗАСОБАМИ NEXT.JS»**

Виконав: студент 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
М.Р. Лісняк
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, PhD Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лісняку Максиму Руслановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Проектування і розробка інтернет-магазину засобами Next.js

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області на огляд інструментарію.
2. Огляд структур та схем даних.
3. Реалізація клієнтської частини інтернет-магазину.
4. Демонстрація прикладів роботи вебзастосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.03.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.03.2024	
2.	Збір вихідних даних.	15.04.2024	
3.	Обробка методичних та теоретичних	27.05.2024	
	джерел.		
4.	Розробка першого та другого розділу.	02.09.2024	
5.	Розробка третього та четвертого розділу.	04.11.2024	
6.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	12.12.2024	

Студент _____ М.Р. Лісняк
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Проектування і розробка інтернет-магазину засобами Next.js»: 56 с., 29 рис.,12 джерел.

ІНТЕРНЕТ-МАГАЗИН, BACKEND, FRONTEND, GRAPHQL, MONGODB, NEXT.JS, PAYLOAD CMS, SCSS, STRIPE, TYPESCRIPT.

Об’єкт дослідження – процес розробки інтернет-магазину засобами Next.js.

Мета роботи: показати переваги використання сучасних інструментів у розробці інтернет-магазину.

Методи дослідження – порівняльний аналіз та практичне впровадження.

Новизна роботи: розробка інтернет-магазину з використанням Payload CMS та Next.js в контексті сучасних вимог електронної комерції та технологічних можливостей.

Основні конструкторські характеристики: архітектура клієнтської частини засобами Next.js, схеми даних і взаємодія з сервером через GraphQL.

Технологічні характеристики: Використання Payload CMS для управління контентом і Next.js для реалізації клієнтської частини, інтеграція з MongoDB та Stripe.

Техніко-експлуатаційні характеристики: підтримка і розвиток інтернет-магазину, можливість швидкого впровадження змін і розширення функціоналу.

Значущість у контексті розробки інтернет-магазинів: забезпечення високої якості користувацького досвіду, швидкість розгортання інтерфейсу та ефективне управління контентом.

SUMMARY

Master's qualifying paper "Design and Development of an Online Store Using Next.js": 56 pages, 29 figures, 12 references.

BACKEND, FRONTEND, GRAPHQL, MONGODB, NEXT.JS, ONLINE STORE, PAYLOAD CMS, SCSS, STRIPE, TYPESCRIPT.

The object of the study is the development of an online store using Next.js.

The aim of the study is showing the advantages of using modern tools for the development of an online store.

The methods of research are comparative analysis and practical implementation.

Novelty of work: development of an online store using Payload CMS and Next.js in the context of modern e-commerce requirements and technological capabilities.

Main design features: architecture of the client part using Next.js, data schemas and interaction with the server via GraphQL.

Technological characteristics: use of Payload CMS for content management and Next.js for implementation of the client part, integration with MongoDB and Stripe.

Technical and operational characteristics: maintenance and development of an online store, the ability to quickly implement changes and expand functionality.

Significance in the context of the development of online stores: ensuring a high quality of user experience, speed of deployment of the interface and effective management of content.

ЗМІСТ

Завдання на кваліфікаційну роботу....2
Реферат 4
Summary 5
Вступ.... 8
1 Аналіз предметної області та огляд інструментарію 10
1.1 Сучасний стан проблеми 10
1.2 Функціональні вимоги 11
1.3 Обґрунтування обраного інструментарію 12
1.4 Порівняльний аналіз альтернативних рішень 15
1.4.1 CMS: Payload CMS та Strapi 15
1.4.2 Фреймворки: Next.js та Gatsby 15
1.4.3 Бази даних: MongoDB та SQL 16
1.4.4 API: GraphQL та REST 16
1.5 Місце розробки в системі наявних рішень 17
1.6 Аспекти новизни та актуальності розробки 17
1.7 Висновки до розділу 1 18
2 Структури та схеми даних.... 19
2.1 Огляд колекцій Payload CMS 20
2.2.1 Колекція Orders 21
2.2.2 Колекція Pages.... 22
2.2.3 Колекція Products 22
2.2.4 Колекція Users 24
2.2.5 Колекція Categories 25
2.2.6 Колекція Media 25
2.3 Огляд глобальних налаштувань Payload CMS 26
2.3.1 Глобальне налаштування Footer 26
2.3.2 Глобальне налаштування Header 26

2.3.3 Глобальне налаштування Settings 27
2.4 Висновки до розділу 2 27
3 Проєктування та реалізація клієнтської частини застосунку 28
3.1 Структура клієнтської частини застосунку 28
3.2 Опис сторінок і компонентів 29
3.2.1 Сторінка Products 30
3.2.2 Сторінка Product 30
3.2.3 Сторінка Cart 31
3.2.4 Сторінка Checkout 32
3.3 Інтеграція клієнтської частини на Next.js з серверною частиною 33
3.4 Висновки до розділу 3 38
4 Приклади роботи застосунку 39
4.1 Огляд головної сторінки 39
4.2 Огляд сторінок авторизації та реєстрації 41
4.3 Огляд сторінок адміністратора 42
4.3.1 Огляд сторінок управляння продуктами 42
4.3.2 Огляд сторінок керування категоріями 44
4.3.3 Огляд сторінок керування користувачами 46
4.3.4 Огляд сторінок керування сторінками 47
4.4 Огляд сторінки перегляду товарів 50
4.5 Огляд сторінки перегляду товару 51
4.6 Огляд сторінки кошика 52
4.7 Огляд сторінки оформлення замовлення 53
4.8 Висновки до розділу 4 54
Висновки 55
Перелік посилань 56

ВСТУП

У сучасному світі електронна комерція продовжує швидко розвиватися, що зумовлено зростанням кількості онлайн-покупок. Однією з ключових складників успіху в цій сфері є ефективне створення інтернет-магазинів, що здатні задовольняти потреби споживачів і відповідати вимогам технологічного прогресу. Інтеграція сучасних інструментів та технологій дозволяє створювати гнучкі та масштабовані платформи, які можуть швидко адаптуватися до змін на ринку.

Метою цієї роботи є розгляд функціональних та технічних аспектів розробки інтернет-магазину для продажу техніки, пов’язаної з відеоіграми, з використанням інструментів Payload CMS та Next.js. Такий підхід дозволяє створити ефективну, зручну для користувачів платформу, що поєднує сучасні технології для створення інтуїтивно зрозумілого інтерфейсу, простоти управління контентом і надійної обробки замовлень.

У роботі розглядаються можливості використання технологій, таких як MongoDB для зберігання даних, Stripe для здійснення платежів, GraphQL для створення API, SCSS для стилізації та TypeScript для підвищення ефективності розробки. Окрім цього, важливим аспектом є вибір інструментів, що дозволяють швидко адаптувати та масштабувати функціонал магазину відповідно до потреб користувачів і вимог ринку.

Структурно робота складається з чотирьох розділів. Перший розділ присвячений аналізу сучасних інструментів розробки інтернет-магазинів та порівняльному аналізу можливих рішень, обґрунтуванню вибору Payload CMS та Next.js для створення платформи. У другому розділі описано структуру даних, які використовуються в магазині, а також процес створення схем даних. Третій розділ фокусується на проєктуванні та реалізації клієнтської частини інтернет-магазину з використанням Next.js, а також на інтеграції з серверною частиною через GraphQL. Четвертий розділ надає приклади роботи інтернет-

магазину, зокрема, демонструючи функціонування ключових сторінок та компонентів системи.

Ця робота має на меті продемонструвати ефективність використання CMS систем та сучасних вебфреймворків у розробці інтернет-магазинів та підкреслити переваги таких платформ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІНСТРУМЕНТАРІЮ

1.1 Сучасний стан проблеми

Електронна комерція сьогодні є однією з найдинамічніших галузей економіки, де все більше компаній переходять в онлайн-простір для продажу своїх товарів і послуг. Створення успішного інтернет-магазину вимагає використання сучасних технологій та інструментів розробки. Інтернет-магазин, який розглядається в цій роботі, спеціалізується на продажу техніки, пов’язаної з відеоіграми, що включає комп’ютерну техніку, аксесуари та інші продукти, необхідні геймерам.

Одним із ключових аспектів створення інтернет-магазину є вибір відповідних інструментів. Існує багато різних платформ для створення інтернет-магазинів, таких як Shopify, Magento, WooCommerce та інші. Кожна з них має свої переваги та недоліки, а також різні можливості для налаштування та розширення функціоналу. У цьому випадку, для спеціалізованої тематики магазину, вибір інструментів повинен враховувати потреби користувачів.

У цій роботі розглядається підхід до створення інтернет-магазину на базі Payload CMS та Next.js. Такий підхід дозволяє створити гнучкий і масштабований інтернет-магазин з можливістю легкого налаштування та розширення завдяки модульній архітектурі CMS та можливостям фреймворку Next.js для серверного рендерингу та оптимізації продуктивності. Також використовуються інструменти, такі як MongoDB для зберігання даних, Stripe для виконання оплати замовлень, GraphQL для створення API, SCSS для стилізації та TypeScript для написання коду, що підвищує надійність та ефективність розробки.

1.2 Функціональні вимоги

Функціональні вимоги інтернет-магазину комп’ютерної техніки включають:

- авторизація та реєстрація користувачів: користувачі повинні мати можливість реєструвати акаунти, входити в систему для доступу до функцій, таких як історія покупок, збереження замовлень;
- каталог товарів: платформа повинна містити каталог з товарами, включаючи комп’ютерну техніку та аксесуари для відеоігор, з можливістю перегляду інформації про товар та фото товару;
- фільтрація та сортування товарів: користувачі повинні мати можливість фільтрувати товари за категоріями та сортувати їх за різними критеріями, такими як ціна, новизна;
- перегляд сторінки товару: кожен товар повинен мати окрему сторінку з детальною інформацією, включаючи фото, опис, ціну і наявність;
- кошик: користувачі повинні мати можливість додавати товари до кошика, переглядати їх перед оформленням замовлення та змінювати кількість товарів або видаляти їх;
- замовлення: після вибору товарів користувачі повинні мати можливість оформити замовлення, вибираючи способи оплати; процес має включати підтвердження даних;
- CMS панель для адміністраторів: адміністратори повинні мати доступ до панелі керування для управління товарами, замовленнями, користувачами, а також налаштування різних функцій платформи.

Ці вимоги забезпечують зручний процес покупки для користувачів та ефективне адміністрування магазину з боку адміністрації.

Нижче на рисунку 1.1 представлена діаграма прецедентів.

Загалом, інтернет-магазин на базі Payload CMS та Next.js має функціонал для ефективного обслуговування клієнтів. Ця архітектура дозволяє легко розширювати функціональність та підтримувати високий рівень продуктивності та надійності.

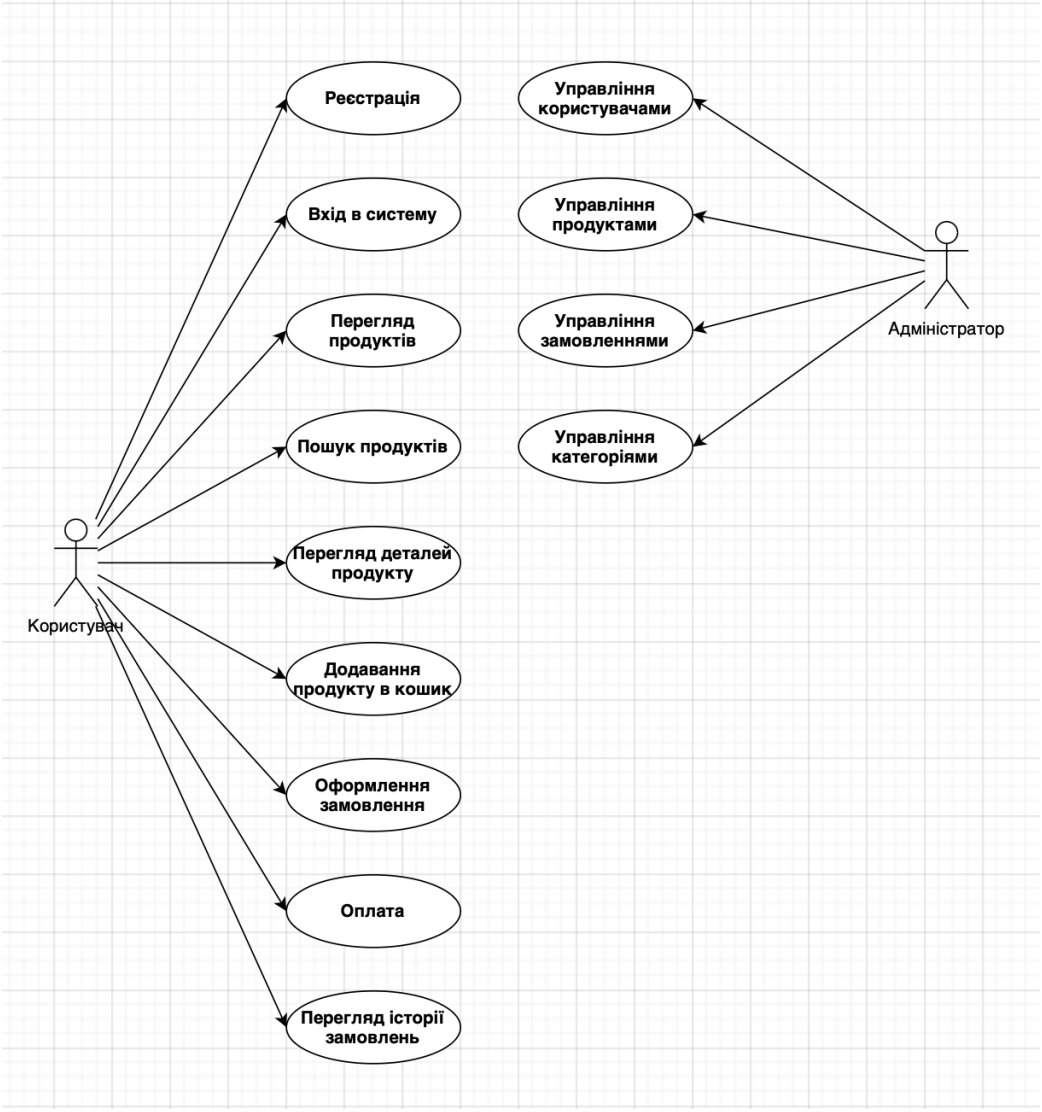


Рисунок 1.1 – Діаграма прецедентів

1.3 Обґрунтування обраного інструментарію

Для розробки інтернет-магазину обрано наступні інструменти та технології.

Next.js є обраним фреймворком для фронтенду інтернет-магазину з огляду на його технічні особливості та переваги. Використання цього інструменту обґрунтоване його гнучкою системою маршрутизації, яка дозволяє керувати навігацією сторінок, таких як продукти та кошики покупок.

Друга ключова причина полягає у підтримці TypeScript, що забезпечує високу надійність та стабільність коду.

Крім того, фреймворк надає можливості для оптимізації вебсторінок, включаючи зображення, шрифти та скрипти, що сприяє поліпшенню користувацького досвіду та відповідності Core Web Vitals, згідно з [1].

Ці технічні аспекти роблять Next.js оптимальним вибором для розробки інтернет-магазину, забезпечуючи зручний інструментарій для розробників і надійність вебдодатку для користувачів.

Для управління контентом в інтернет-магазині використовується Payload CMS, яке є системою керування вмістом. Payload надає REST та GraphQL API для збереження та маніпулювання даними, а також гнучкий контроль доступу до них, що ідеально підходить для складних застосунків, таких як інтернет-магазини.

Згідно з [2], система відзначається інтуїтивно зрозумілим інтерфейсом для розробників та користувачів, що спрощує процес розробки та підтримки інтернет-магазину. Payload також забезпечує можливість розгортання на сервері, що робить його ідеальним вибором для проєктів, які вимагають контролю над інфраструктурою та хостингом без залучення сторонніх платформ.

Ця CMS відповідає вимогам сучасного інтернет-магазину, забезпечуючи надійність, гнучкість і простоту у використанні, що робить її гарним рішенням для успішної реалізації проєктів цього типу.

Для обробки платежів в інтернет-магазині інтегровано Stripe, що забезпечує безпечну обробку платежів банківськими картками через Stripe.js. Це дозволяє зменшити ризики безпеки й забезпечити зручний процес оплати для користувачів.

Згідно з [3], Payload CMS має вбудовану інтеграцію зі Stripe, що дозволяє легко та швидко під’єднати систему оплати. Крім того, Payload CMS надає зручний та інтуїтивно зрозумілий інтерфейс для керування платежами та замовленнями в інтернет-магазині.

MongoDB є вибраною базою даних для збереження даних, що підтримується Payload CMS, згідно з [4]. Однією з переваг використання MongoDB в цій роботі є те, що вона є гнучкою та масштабованою системою.

Можна самостійно змінювати структуру бази даних засобами Payload CMS, що є зручним для швидкого внесення змін та адаптації до змінних вимог бізнесу.

MongoDB є також зручною для зберігання даних, таких як колекції та глобальні налаштування. Завдяки використанню документів, MongoDB дозволяє зберігати складні структури даних в одному місці, що є корисним для зберігання даних, які мають багато вкладених полів.

GraphQL використовується разом з Payload CMS для створення надійного та гнучкого інтерфейсу для роботи з даними інтернет-магазину. GraphQL повністю підтримується Payload CSM, згідно з [5]. Його використання дозволяє запитувати лише потрібні дані та легко створювати складні запити.

Крім того, GraphQL дозволяє легко створювати складні запити, які можуть містити дані з різних джерел. Це корисно для інтернет-магазинів, які можуть мати багато різних типів даних, таких як товари, категорії, колекції, відгуки тощо.

В цій роботі для написання коду використовується мова TypeScript. Однією з головних переваг TypeScript є те, що він є сумісним з JavaScript. Це означає, що його можна впроваджувати у проєкти, не турбуючись про те, що це може зламати наявний код. Також ця мова повністю підтримується Payload CMS, згідно з [6]. Завдяки використанню TypeScript, можна створити надійний та масштабований код, що є важливим аспектом розробки. Крім того, TypeScript дозволяє використовувати нові інструменти та бібліотеки, що дозволяє створювати більш ефективні та продуктивні проєкти.

Також для створення стилів інтернет-магазину використовується SCSS. Завдяки SCSS, можна створити більш ефективний та зручний код.

SCSS є розширенням мови CSS, яка дозволяє використовувати додаткові інструменти та можливості, що покращують продуктивність та зручність роботи зі стилями. Він дозволяє використовувати такі інструменти, як вкладені селектори, міксіни, функції та оператори.

Однією з головних переваг SCSS є те, що він є повністю сумісним з CSS. Крім того, SCSS має широку підтримку з боку інструментів розробки, таких як Visual Studio Code. Особливості використання SCSS з Payload CMS наведено в [7].

1.4 Порівняльний аналіз альтернативних рішень

Нище наведено порівняння основних інструментів, що було використано при розробці інтернет-магазину з їх аналогами.

1.4.1 CMS: Payload CMS та Strapi

Payload CMS є потужним інструментом для створення API, але на відміну від Strapi, він має більш гнучку та кастомізовану архітектуру, що робить його гарним вибором для розробки інтернет-магазинів. Payload CMS дозволяє легко налаштовувати структуру даних та забезпечує високий рівень контролю над процесом розробки. Адміністративний інтерфейс Payload CMS є простим, інтуїтивно зрозумілим та зручним, що значно спрощує управління контентом та налаштуваннями магазину. Завдяки гнучкості та потужності цієї системи, можна швидко адаптувати платформу під специфічні вимоги інтернет-магазину, а також інтегрувати додаткові функції, необхідні для забезпечення ефективної роботи бізнесу. Більше інформації про Strapi можна знайти у [8].

1.4.2 Фреймворки: Next.js та Gatsby

При розробці інтернет-магазину на базі React, вибір між Next.js та Gatsby є ключовим. Next.js пропонує кращі можливості для серверного рендерингу та

оптимізації продуктивності, що є важливими для інтернет-магазину. На відміну від Gatsby, який є статичним генератором сайтів, Next.js дозволяє як статичне, так і динамічне генерування сторінок, що важливо для магазинів з великою кількістю товарів та постійно оновлюваним контентом.

Next.js також має гнучку архітектуру та підтримує розширення завдяки використанню middleware та плагінів, що полегшує інтеграцію з іншими сервісами та технологіями. У порівнянні з Gatsby, Next.js надає більш гнучкі можливості для роботи з даними за допомогою getStaticProps та getServerSideProps, що робить цей підхід зручнішим та ефективнішим. Більше про Gatsby за [9].

1.4.3 Бази даних: MongoDB та SQL

Основним конкурентом для MongoDB є SQL-системи управління базами даних, такі як MySQL або PostgreSQL. Однак, на відміну від SQL-систем, MongoDB пропонує гнучку схему даних, що дозволяє зберігати дані у форматі JSON-подібних документів. Це полегшує зміну структури даних і дозволяє масштабувати базу даних відповідно до змінюваних вимог інтернет-магазину. Гнучкість MongoDB робить її ідеальним вибором для проєктів, які потребують швидкої адаптації до змінних бізнес-вимог та великої кількості непередбачених змін у даних. Це особливо важливо для інтернет-магазинів, де структура товарів, категорій та замовлень може змінюватися з часом.

1.4.4 API: GraphQL та REST

Основним конкурентом GraphQL є REST API. Основна різниця між ними полягає в тому, що GraphQL дозволяє клієнту запитувати лише ті дані, які йому необхідні, і отримувати їх в одному запиті, що підвищує ефективність та зменшує навантаження на сервер. В порівнянні з REST, де клієнт може

отримувати надмірні або непотрібні дані з кількох запитів, GraphQL забезпечує більш точне й оптимізоване використання ресурсів. Це робить GraphQL ідеальним вибором для проєктів, де важлива швидкість роботи та ефективність при взаємодії з великими обсягами даних.

1.5 Місце розробки в системі наявних рішень

Запропонований інтернет-магазин використовує сучасні технології, такі як Payload CMS та Next.js, для створення гнучкого та масштабованого рішення з можливістю легкого налаштування та розширення функціоналу. Для зберігання даних використовується MongoDB, а API створено за допомогою GraphQL. SCSS використовується для стилізації, а TypeScript підвищує надійність та ефективність розробки.

1.6 Аспекти новизни та актуальності розробки

Розробка інтернет-магазину є новою та актуальною, оскільки використовує передові технології та інструменти, такі як Payload CMS, Next.js, MongoDB, GraphQL, SCSS та TypeScript. Такий підхід дозволяє створити гнучку та масштабовану архітектуру, що забезпечує швидке налаштування, адаптацію до змін і можливість розширення функціоналу. Використання сучасних технологій забезпечує ефективну інтеграцію між різними частинами системи та підвищує надійність платформи.

Подальші напрямки розвитку включають впровадження персоналізації контенту для покращення користувацького досвіду, використання штучного інтелекту для аналізу поведінки користувачів і надання рекомендацій, а також інтеграцію з іншими популярними сервісами для розширення можливостей платформи. Крім того, важливою є постійна оптимізація продуктивності та безпеки.

1.7 Висновки до розділу 1

У цьому розділі було детально описано інструменти та технології, які використовувалися для розробки інтернет-магазину, включаючи Payload CMS, Next.js, MongoDB, GraphQL, SCSS та TypeScript. Було проведено порівняльний аналіз альтернативних рішень, що дозволило обґрунтувати вибір цих технологій для забезпечення гнучкості, масштабованості та ефективності платформи. Такий підхід дозволяє створити надійне рішення, яке відповідає вимогам сучасного ринку електронної комерції.

2 СТРУКТУРИ ТА СХЕМИ ДАНИХ

Для зберігання даних інтернет-магазину використовується база даних MongoDB. Нижче наведено опис колекцій MongoDB, які були створені при виконанні цієї роботи:

- _pages_versions: містить історію версій сторінок, створених в системі керування вмістом (CMS) Payload;
- _products_versions: містить історію версій товарів, створених в CMS Payload;
- categories: містить інформацію про категорії товарів, включаючи назву, опис та зображення;
- globals: містить глобальні налаштування для інтернет-магазину, включаючи назву, опис, логотип та контактну інформацію;
- media: містить інформацію про медіафайли, завантажені в CMS Payload;
- orders: містить інформацію про замовлення, створені в інтернет-магазині, включаючи інформацію про товари та суму замовлення;
- pages: містить інформацію про сторінки, створені в CMS Payload, включаючи назву, опис, контент та SEO-налаштування;
- payload-migrations: містить інформацію про міграції, виконані в CMS Payload;
- payload-preferences: містить інформацію про налаштування користувачів CMS Payload;
- products: містить інформацію про товари, включаючи назву, опис, ціну, зображення та іншу додаткову інформацію;
- redirects: містить інформацію про перенаправлення сторінок, створені в CMS Payload;
- users: містить інформацію про користувачів інтернет-магазину, включаючи інформацію про обліковий запис, адресу доставлення та історію замовлень.

Нижче на рисунку 2.1 представлена діаграма колекцій MongoDB.

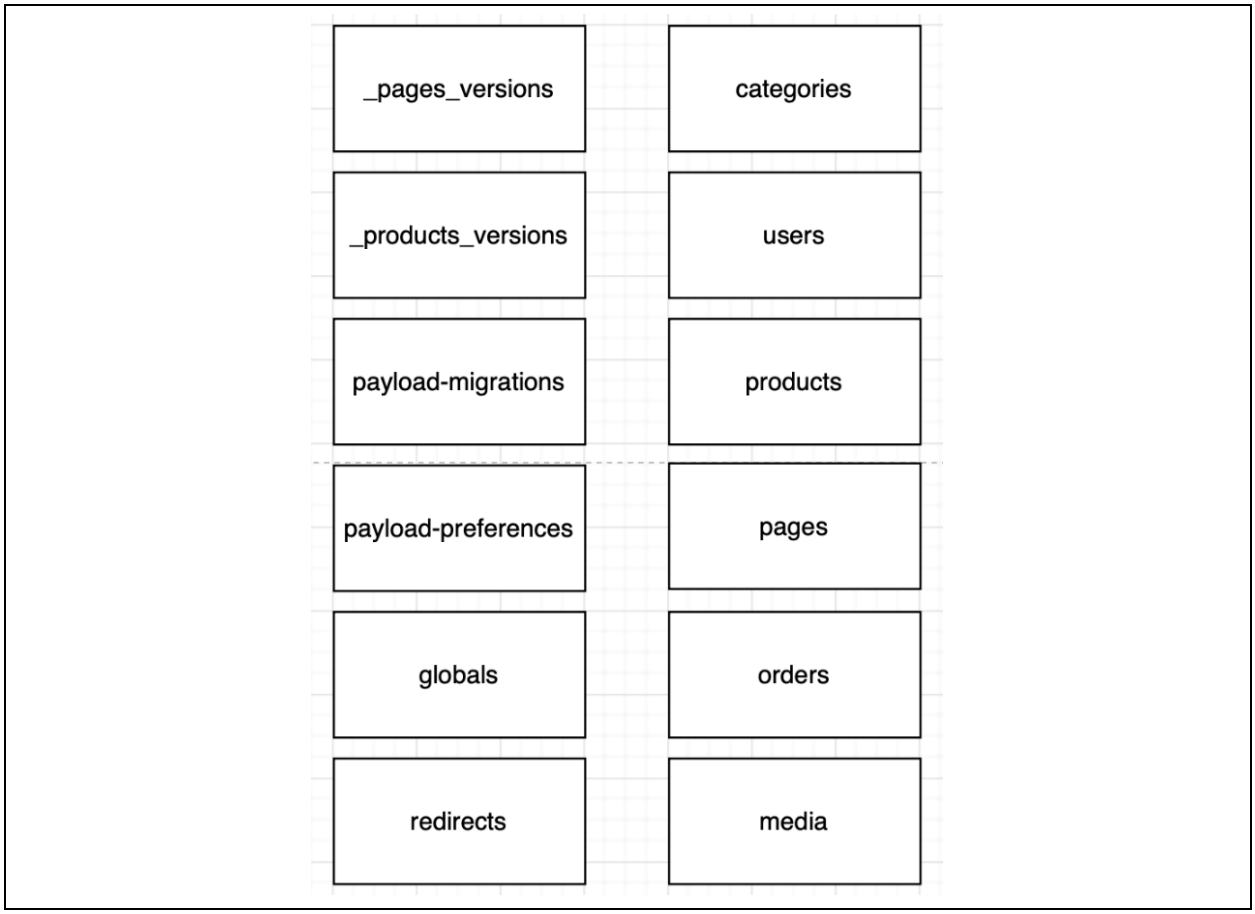


Рисунок 2.1 – Діаграма колекцій MongoDB

В цій роботі, колекції та глобальні налаштування були використані для зберігання даних інтернет-магазину, включаючи інформацію про товари, категорії, сторінки, замовлення та користувачів. Завдяки використанню колекцій та глобальних налаштувань, можна створювати та керувати контентом інтернет-магазину.

2.1 Огляд колекцій Payload CMS

Однією з особливостей CMS Payload є те, що вона дозволяє створювати колекції та глобальні налаштування. Колекції представляють тип вмісту, який зберігається в системі та може містити багато документів, згідно з [10].

Колекції визначають структуру даних та функціональність, прив’язану до цих даних. Вони можуть представляти все, що можна зберігати в базі даних, наприклад, сторінки, пости, користувачів, людей, замовлення, категорії, події, клієнтів, транзакції та все інше, що потрібно вашому додатку.

Далі надано опис налаштувань колекцій Payload CMS, використаних у цій роботі.

2.2.1 Колекція Orders

Колекція Orders містить кілька полів, які дозволяють зберігати дані про замовлення. Поле orderedBy є відношенням до колекції users і містить інформацію про користувача, який зробив замовлення. Перед зміною цього поля викликається хук beforeChange, що виконує функцію populateOrderedBy, яка автоматично заповнює інформацію про користувача, який створив замовлення.

Поле stripePaymentIntentID є текстовим і використовується для зберігання ідентифікатора платіжного наміру Stripe, що пов’язаний з конкретним замовленням. У адміністративному інтерфейсі це поле відображається в бічній панелі, і містить компонент LinkToPaymentIntent, який дозволяє перейти за посиланням до відповідного платіжного наміру в Stripe.

Поле total є числовим і обов’язковим для заповнення. Воно містить загальну суму замовлення, з мінімальним значенням 0.

Поле items є масивом, що містить інформацію про всі товари в замовленні. Кожен елемент масиву включає відношення до колекції products, яке вказує на замовлений товар, а також поля для ціни та кількості товару, які зберігаються як числові значення з мінімальними обмеженнями на значення.

Ця конфігурація дозволяє керувати замовленнями в інтернет-магазині, забезпечуючи структуроване зберігання даних та зручний доступ до них через адміністративний інтерфейс.

2.2.2 Колекція Pages

Колекція Pages містить кілька полів, які відповідають за зберігання та управління сторінками в інтернет-магазині.

Поле title є текстовим і є обов’язковим для заповнення. Воно містить заголовок сторінки.

Поле publishedOn має тип date і відображається в адміністративному інтерфейсі в бічній панелі. Воно дозволяє вибрати дату та час публікації сторінки. Хук beforeChange перевіряє, чи була сторінка раніше опублікована (статус published). Якщо статус сторінки вже опублікований і нова дата не вказана, то автоматично встановлюється поточна дата і час.

Поле tabs дає змогу розділяти контент сторінки на вкладки. Одна з вкладок – Hero – містить поля, визначені в компоненті hero. Інша вкладка – Content – має обов’язкове поле layout типу blocks, яке включає різні блоки контенту, такі як CallToAction, Content, MediaBlock та Archive. Ці блоки дозволяють налаштовувати різні елементи на сторінці, які можуть бути заповнені для створення сторінки з відповідним контентом.

Останнє поле – slugField – генерує ідентифікатор сторінки для використання в URL, зазвичай це короткий текст, що є унікальним для кожної сторінки.

Ця конфігурація дозволяє керувати сторінками вебсайту, забезпечуючи структуроване зберігання даних та зручний доступ до них через адміністративний інтерфейс.

2.2.3 Колекція Products

Колекція Products містить кілька полів, що визначають різні аспекти продуктів в інтернет-магазині.

Поле title є текстовим і обов’язковим для заповнення. Воно містить назву продукту.

Поле stock має тип number і є обов’язковим для заповнення. Воно визначає кількість одиниць продукту, доступних на складі. Якщо значення кількості товару є меншим за нуль, спрацьовує хук beforeChange, який викидає помилку, що кількість товару не може бути від’ємною. Значення за замовчуванням – 1.

Поле publishedOn має тип date і відображається в адміністративному інтерфейсі. Воно дозволяє вказати дату та час публікації продукту. Хук beforeChange перевіряє, чи був продукт уже опублікований (статус published). Якщо так, і дата публікації не була вказана, автоматично встановлюється поточна дата.

Поле tabs дозволяє розділяти налаштування продукту на вкладки. Вкладка Content містить поле layout типу blocks, яке включає блоки контенту, такі як CallToAction, Content, MediaBlock та Archive, для налаштування відображення продукту. Вкладка Product Details містить кілька специфічних полів:

- stripeProductID для зберігання ідентифікатора продукту Stripe;
- priceJSON для збереження JSON даних про ціну;
- enablePaywall, який дозволяє увімкнути платіжну стіну для продукту;
- paywall для налаштування платіжної стіни, доступ до якої обмежено на основі покупок користувача.

Поле categories є відношенням до колекції categories і дає змогу призначити продукту кілька категорій.

Поле relatedProducts є відношенням до колекції products і дозволяє вказати інші продукти, що можуть бути пов’язані з поточним продуктом. Використовує фільтрацію для уникнення вибору самого продукту.

Поле slugField генерує унікальний ідентифікатор для використання в URL.

Поле skipSync є чекбоксом, яке позначає, чи потрібно пропустити синхронізацію цього продукту. Воно є тільки для адміністратора і приховане від інших користувачів.

Ця конфігурація дозволяє керувати товарами на вебсайті, забезпечуючи гнучкість налаштувань і доступу до даних через адміністративний інтерфейс.

2.2.4 Колекція Users

Колекція Users містить кілька полів, які визначають інформацію про користувачів і їхню взаємодію з платформою.

Поле name зберігає ім’я користувача.

Поле roles є селектором із кількома значеннями, що дозволяє призначати ролі користувачеві. За замовчуванням кожен новий користувач отримує роль customer. Доступні ролі: admin і customer. Хук beforeChange викликає функцію ensureFirstUserIsAdmin, яка гарантує, що перший користувач в системі буде адміністратором. Доступ до читання, створення та оновлення цього поля мають лише адміністратори.

Поле purchases є відношенням до колекції products і зберігає продукти, які були придбані користувачем. Хук beforeChange викликає функцію resolveDuplicatePurchases, щоб уникнути дублювання придбаних продуктів.

Поле stripeCustomerID зберігає ідентифікатор користувача в системі Stripe. Доступ до цього поля обмежений лише для адміністраторів. В адміністративному інтерфейсі поле розташоване у бічній панелі й має спеціальний компонент CustomerSelect для зручності вибору.

Поле cart є групою, що містить інформацію про товари в кошику користувача. Воно включає поле items, яке є масивом з відношенням до продуктів і кількістю кожного продукту. Кількість товару задається через поле quantity, яке має мінімальне значення 0 та збільшується на заданий крок.

Поле skipSync є чекбоксом, який позначає, чи потрібно пропустити синхронізацію даних користувача. Це поле приховане й доступне лише адміністраторам.

Ця конфігурація дозволяє управляти користувачами системи,

забезпечуючи гнучкість налаштувань доступу та функціональність для роботи з основними даними користувачів.

2.2.5 Колекція Categories

Колекція Categories містить два основних поля для визначення категорій у системі.

Поле title є текстовим і зберігає назву категорії. Це поле є обов’язковим для заповнення.

Поле media є відношенням до колекції media та використовується для збереження зображення або іншого файлу, що пов’язаний із категорією. Це поле дозволяє завантажувати й асоціювати медіафайли з конкретною категорією.

Ця конфігурація дозволяє створювати й управляти категоріями зручним і простим способом, забезпечуючи доступ для всіх користувачів до перегляду інформації про категорії.

2.2.6 Колекція Media

Колекція Media призначена для зберігання інформації про завантажені медіафайли.

Поле alt є текстовим і обов’язковим для заповнення. Воно зберігає альтернативний текст для опису медіафайлу, що покращує доступність і SEO.

Поле caption є полем із багатим текстом, яке дозволяє додавати підпис до медіафайлу. Воно використовує редактор slateEditor, який підтримує додавання посилань у підписі, що дає змогу створювати більш інформативні та інтерактивні підписи.

Ця конфігурація дозволяє зручно управляти медіафайлами, забезпечуючи простий доступ до них для всіх користувачів системи.

2.3 Огляд глобальних налаштувань Payload CMS

Глобальні налаштування є «одноразовим» фрагментом вмісту, який підходить для зберігання навігаційних структур, тем, верхньо-рівневих метаданих та іншої інформації, яка використовується на всіх сторінках інтернет-магазину, згідно з [11]. Глобальні налаштування в багатьох аспектах схожі на колекції, але існує лише один екземпляр глобальних налаштувань, тоді як колекції можуть містити багато документів.

2.3.1 Глобальне налаштування Footer

Глобальне налаштування Footer визначає вміст футера на всіх сторінках сайту.

Поле copyright є текстовим, обов’язковим для заповнення і зберігає текст, що відображається як повідомлення про авторські права.

Поле navItems є масивом, який дозволяє додати до шести елементів навігації. Кожен елемент використовує функцію link для створення навігаційних посилань, але без налаштування зовнішнього вигляду посилань, забезпечуючи гнучке налаштування структури футера.

Це налаштування дозволяє легко змінювати й оновлювати інформацію, що відображається у підвалі вебсайту, забезпечуючи простоту управління важливими даними для користувачів.

2.3.2 Глобальне налаштування Header

Глобальне налаштування Header визначає структуру навігаційного меню в шапці сайту.

Поле navItems є масивом, що дозволяє додати до шести елементів навігації. Кожен елемент створюється за допомогою функції link, яка

відповідає за створення навігаційних посилань без специфічних налаштувань зовнішнього вигляду. Це забезпечує можливість легко налаштовувати основне меню сайту.

Це налаштування дозволяє легко змінювати й оновлювати навігаційні елементи у верхній частині вебсайту, забезпечуючи користувачам зручний спосіб навігації по сторінках.

2.3.3 Глобальне налаштування Settings

Глобальне налаштування Settings містить поле productsPage, яке є відношенням до колекції pages. Це поле дозволяє вибрати сторінку, яка буде використовуватися як сторінка з переліком товарів на сайті.

Це налаштування дозволяє зручно управляти основними параметрами вебсайту через CMS, зокрема встановлювати зв’язок зі сторінкою, де розміщені товари, що спрощує настройку та управління контентом.

2.4 Висновки до розділу 2

У цьому розділі було представлено опис структур даних для основних колекцій, що використовуються в інтернет-магазині. Кожна колекція визначає набір полів, що відповідають за збереження та обробку інформації про користувачів, продукти, замовлення, категорії, медіаресурси та налаштування сайту. Використання полів, таких як текстові значення, числа, відносини між колекціями, а також блоки для зберігання більш складних даних, дозволяє забезпечити ефективне управління контентом та налаштуваннями. Крім того, для деяких полів додано бізнес-логіку через хуки, що забезпечує додаткову валідацію та автоматизацію процесів. Завдяки такій організації даних інтернет-магазин може гнучко налаштовуватися під різні вимоги та потреби.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКУ

3.1 Структура клієнтської частини застосунку

Для реалізації інтернет-магазину було використано шаблон Payload E-Commerce Template, який надає інструменти для побудови платформ електронної комерції. Публічний репозиторій шаблону доступний за [12]. Цей шаблон надає необхідні інструменти для побудови інтернет-магазину, однак для адаптації під конкретні вимоги проєкту було виконано налаштування та інтеграції, а також внесені зміни в дизайн та функціональність, щоб забезпечити зручний і ефективний інтерфейс для кінцевих користувачів.

Нижче наведена структура директорій, що використовується у проєкті:

– /app: основна директорія клієнтської частини проеєту, що містить всі компоненти для побудови інтерфейсу користувача, управління даними та інтеграції з бекендом;
– /_api: містить файли для роботи з API-запитами, наприклад, для завантаження документів (fetchDoc.ts, fetchDocs.ts), отримання глобальних налаштувань (fetchGlobals.ts) та управління автентифікацією (token.ts, getMe.ts);
– /_blocks: містить блоки контенту для повторного використання на різних сторінках, такі, як – RelatedProducts – для відображення схожих товарів, CallToAction для створення закликів до дії, ArchiveBlock для динамічного відображення архіву товарів за допомогою компонента CollectionArchive;
– /_components: містить компоненти сайту, такі як картки товарів, форми, кнопки та елементи навігації;
– /_css: містить стилі проєкту, де app.scss – глобальні стилі, colors.scss – кольорова палітра, common.scss – базові стилі, queries.scss – медіазапити, theme.scss – тема, type.scss – типографіка;

- /_graphql: містить GraphQL-запити й мутації для взаємодії з бекендом, наприклад, для роботи з товарами (products.ts), категоріями (categories.ts), замовленнями (orders.ts) та кошиком (cart.ts);
- /_heros: компоненти для створення hero-блоків, де наприклад, CustomHero використовується для індивідуальних вітальних банерів, HighImpact, MediumImpact, LowImpact – для візуально привабливих елементів різного рівня впливу;
- /_providers: провайдери для управління глобальним станом додатка, де наприклад, Auth і Cart для автентифікації й кошика, Filter для фільтрації товарів, Theme для вибору теми;
- /_utilities: допоміжні утиліти, де наприклад, canUseDOM.ts перевіряє доступність DOM, formatDateTime.ts форматування дат, generateMeta.ts генерує SEO метадані;
- /(pages): директорія зі сторінками проєкту, такими як /account (профіль користувача), /cart (кошик), /checkout (оформлення замовлення), not-found.tsx (сторінка 404);
- /api: містить API-обробники для попереднього перегляду контенту та оновлення кешу (exit-preview.ts, preview.ts, revalidate.ts);
- /constants: файли з постійними значеннями, які використовуються в проєкті, такі як налаштування API, мови або валюти;
- cssVariables.js: оголошення змінних для стилів, що використовуються глобально (кольори, шрифти, відступи);
- layout.tsx: основний макет для всіх сторінок проєкту, який містить шапку, футер і контейнер для основного контенту;
- page.tsx: файл для рендерингу контенту окремих сторінок проєкту.

3.2 Опис сторінок і компонентів

Нижче наведено опис та структуру сторінок та компонентів, які вони включають.

3.2.1 Сторінка Products

Сторінка Products відповідає за відображення списку товарів з можливістю фільтрації та сортування. Вона складається з двох основних компонентів: Filters та Blocks.

Filters: компонент, що дозволяє користувачам фільтрувати товари за категоріями та сортувати їх. Він включає два підкомпоненти: Checkbox для вибору категорій товарів та RadioButton для вибору способу сортування. Компонент фільтрів працює з провайдером useFilter, який зберігає вибрані категорії та критерії сортування. Користувач може вибирати або скидати фільтри, що відображаються на сторінці.

Blocks: компонент, який містить різні блоки контенту для відображення на сторінці. У випадку сторінки Products використовується блок CallToActionBlock, який включає компоненти RichText для відображення тексту та CMSLink для додавання посилань. Блоки складаються з таких елементів, як кнопки заклику до дії, текстові блоки та посилання. CallToActionBlock реалізує відображення тексту та кнопок, що закликають користувача до конкретної дії, наприклад, перегляду товарів чи переходу до інших розділів магазину.

3.2.2 Сторінка Product

Сторінка Product відповідає за відображення сторінки конкретного продукту. Вона завантажує дані продукту за допомогою fetchDoc і передає їх іншим компонентам для відображення. Якщо продукт не знайдений, викликається notFound. Вона також включає компонент ProductHero для відображення основної інформації про продукт, та перевіряє, чи потрібно відображати PaywallBlocks, якщо продукт має paywall. Для відображення пов’язаних товарів використовує компонент Blocks, що отримує дані про пов’язані продукти.

ProductHero: компонент, що відповідає за відображення основної інформації про продукт, включаючи зображення, заголовок, категорії, опис і ціну. Він також включає компонент Media для відображення зображення продукту, компонент Price для відображення ціни, а також кнопку AddToCartButton, щоб додати продукт до кошика.

PaywallBlocks: компонент, що відповідає за відображення контенту, який захищений за допомогою платіжної стіни (paywall). Якщо користувач не авторизований, він побачить повідомлення з посиланням для входу. Якщо користувач авторизований, компонент завантажує додаткові блоки контенту, пов’язані з платіжною стіною, і відображає їх. В разі завантаження або помилки відображається відповідний інтерфейс, зокрема LoadingShimmer або Message.

Blocks: компонент, який відповідає за відображення різних блоків контенту на сторінці. У цьому випадку він використовується для відображення списку пов’язаних продуктів. Блоки можуть містити різні типи контенту, такі як текст, зображення або списки.

3.2.3 Сторінка Cart

Сторінка Cart відповідає за відображення кошика покупок користувача. Вона складається з наступних компонентів:

CartPage: компонент, що відповідає за відображення кошика з товарами, їх кількістю та ціною. Він відображає повідомлення, якщо кошик порожній, пропонуючи перейти до каталогу товарів або увійти в обліковий запис для збережених товарів. Якщо кошик не порожній, CartPage показує список товарів, їхню кількість, підсумкову суму та кнопку для оформлення замовлення або для входу, якщо користувач не авторизований. В цьому компоненті використовуються підкомпоненти LoadingShimmer (для відображення анімації завантаження), Button (для кнопки оформлення

замовлення або переходу до входу), а також Link (для переходу до інших сторінок, таких як каталог товарів або сторінка входу).

CartItem: компонент, що відповідає за відображення інформації про один товар у кошику. Він включає зображення товару (якщо воно є), його назву, кількість та ціну. Користувач може змінювати кількість товару або видаляти його з кошика через відповідні кнопки. Для зміни кількості використовується інтерфейс з кнопками для збільшення та зменшення кількості, а також поле введення для ручного введення кількості. Компонент також використовує підкомпоненти Price (для відображення ціни товару), Media (для відображення зображення товару) і RemoveFromCartButton (для видалення товару з кошика).

Blocks: компонент, що відповідає за відображення різних блоків контенту на сторінці. У контексті сторінки кошика, Blocks можуть включати блоки з підсумковою інформацією про кошик, такими як загальна сума, вартість доставлення та фінальна сума. Вони можуть включати компоненти для відображення ціни та іншої важливої інформації, а також кнопки для переходу до оформлення замовлення або інших дій.

3.2.4 Сторінка Checkout

Сторінка Checkout відповідає за відображення процесу оформлення покупки. Вона завантажує дані користувача та кошика за допомогою getMeUser та useCart, передаючи ці дані іншим компонентам для відображення та обробки. Якщо користувач не авторизований, викликається компонент для автентифікації. Для відображення деталей товару на сторінці оформлення використовуються компоненти, як CheckoutItem, що показує окремі елементи кошика, та CheckoutForm, який містить форму для введення платіжних даних та завершення покупки. В разі завантаження або помилки на сторінці відображається інтерфейс за допомогою LoadingShimmer або Message.

CheckoutItem: компонент, що відповідає за відображення кожного елемента кошика з деталями товару. Він показує зображення товару за допомогою компонента Media, назву, кількість та ціну товару, використовуючи компонент Price. Також може бути можливість редагування кількості або видалення товару.

CheckoutForm: компонент, що відповідає за відображення форми для введення платіжних даних. Включає компонент PaymentElement для обробки платежів, а також кнопку Button для підтвердження покупки. Якщо форма не заповнена або виникає помилка, відображається компонент Message.

3.3 Інтеграція клієнтської частини на Next.js з серверною частиною

Для інтеграції клієнтської частини з сервером, який реалізовано за допомогою Payload CMS, використовуються запити GraphQL. Взаємодія забезпечується через утиліти, одні з яких є fetchDoc, fetchDocs, fetchGlobals які дозволяють отримувати окремий документ або список документів, а також глобальні налаштування сайту з CMS. Ці утиліти побудовані на основі GraphQL-запитів, що охоплюють різні колекції, такі як продукти, сторінки, замовлення тощо.

Функція fetchDoc відповідає за отримання одного документа з колекції Payload CMS. Вона приймає параметри collection (назва колекції), slug (унікальний ідентифікатор документа), а також необов’язкові id і draft. Ця функція використовує відповідний запит GraphQL із заздалегідь визначеної мапи queryMap для вибраної колекції.

Приклад використання наведено на рисунку 3.1.

```
const product = await fetchDoc<Product>({
  collection: 'products',
  slug: 'example-product',
})
```

Рисунок 3.1 – Приклад використання функції fetchDoc

Функція fetchDocs забезпечує отримання списку документів із вказаної колекції. Вона також підтримує параметр draft для роботи з чернетками. Ця функція викликає відповідний запит GraphQL із мапи queryMap, після чого обробляє і повертає масив документів.

Приклад використання наведено на рисунку 3.2.

```
const products = await fetchDocs<Product[]>('products')
```

Рисунок 3.2 – Приклад використання функції fetchDocs

Далі наведено приклади використаних запитів GraphQL, а саме запити на отримання товарів.

Запит PRODUCTS повертає список товарів, приклад наведено на рисунку 3.3.

```
query Products {
  Products(limit: 300) {
    docs {
      slug
    }
  }
}
```

Рисунок 3.3 – Приклад використання запита GraphQL для отримання товарів

Запит PRODUCT використовується для отримання детальної інформації про конкретний товар за його slug. Він також підтримує параметр draft, що дозволяє отримувати чернетки. Приклад запиту наведено на рисунку 3.4.

Запит дозволяє отримувати пов’язані товари, ціну, інформацію для платіжної стіни та метадані.

Утиліта fetchGlobals реалізована для отримання глобальних даних сайту: налаштувань, хедера та футера. Вона забезпечує паралельне виконання трьох окремих запитів за допомогою функцій fetchSettings, fetchHeader і fetchFooter.

```
query Product($slug: String, $draft: Boolean) {
  Products(where: { slug: { equals: $slug}}, limit: 1, draft: $draft) {
    docs {
      id
      title
      stock
      stripeProductID
      layout {
        // Вкладені блоки контенту
      }
      priceJSON
      enablePaywall
      relatedProducts {
        id
        slug
        title
      }
    }
  }
}
```

Рисунок 3.4 – Приклад використання запита GraphQL для отримання конкретного товару

Функція fetchSettings відповідає за отримання глобальних налаштувань сайту. Вона виконує запит до GraphQL API, використовуючи запит SETTINGS_QUERY, і повертає об’єкт Settings. У разі помилки на будь-якому етапі (відповідь сервера або обробка даних) функція кидає відповідне повідомлення про помилку. Цей запит відображено на рисунку 3.5.

```
export const SETTINGS_QUERY = `
  query Settings {
    Settings {
      productsPage {
        slug
      }
    }
  }
`
```

Рисунок 3.5 – Приклад запита GraphQL для отримання глобальних налаштувань

Функція fetchHeader виконує GraphQL-запит HEADER_QUERY для отримання даних хедера. Вона повертає об’єкт Header, що містить контент, необхідний для навігації сайту або іншої інформації, що відображається у верхній частині сторінки. Приклад запита наведено на рисунку 3.6.

```
export const HEADER_QUERY = `
  query Header {
    Header {
      navItems {
        link {
          label
          type
          newTab
          url
          icon {
            url
          }
          reference {
            relationTo
            value {
              ...on Page {
                slug
              }
            }
          }
        }
      }
    }}`
```

Рисунок 3.6 – Приклад запита GraphQL для отримання хедера сайта

Аналогічно до попередніх функцій, fetchFooter виконує запит FOOTER_QUERY, щоб отримати дані футера. Вона повертає об’єкт Footer, що містить інформацію для відображення в нижній частині сайту, наприклад, контактні дані або посилання. Приклад наведено нижче на рисунку 3.7.

```
export const FOOTER_QUERY = `
  query Footer {
    Footer {
      copyright
      navItems {
        link {
          label
          type
          newTab
          url
          icon {
            url
          }
          reference {
            relationTo
            value {
              ...on Page {
                slug
              }
            }
          }
        }
      }
    }
  }
`
```

Рисунок 3.7 – Приклад запита GraphQL для отримання футера сайта

Функція fetchGlobals відповідає за паралельне виконання всіх трьох функцій. Вона використовує Promise.all, щоб ініціювати виконання запитів одночасно і чекати їх завершення. Це дозволяє оптимізувати час завантаження, забезпечуючи швидке отримання всіх необхідних глобальних даних. Результат функції – це об’єкт, що містить усі глобальні дані, необхідні для відображення інтерфейсу користувача. Приклад реалізації можна побачити нижче на рисунку 3.8.

```
export const fetchGlobals = async (): Promise<{
  settings: Settings
  header: Header
  footer: Footer
}> => {
  const settingsData = fetchSettings()
  const headerData = fetchHeader()
  const footerData = fetchFooter()
  const [settings, header, footer]: [Settings, Header, Footer] = await Promise.all([
    await settingsData,
    await headerData,
    await footerData,
  ])
  return {
    settings,
    header,
    footer,
  }
}
```

Рисунок 3.8 – Приклад запита GraphQL для отримання футера сайта

3.4 Висновки до розділу 3

У цьому розділі було детально описано клієнтську частину застосунку, розроблену за допомогою Next.js, а також її інтеграцію із серверною частиною на основі Payload CMS. Було розглянуто структуру застосунку, функціональні особливості основних сторінок, таких як Products, Product, Cart та Checkout, і їхні ключові компоненти.

Окрема увага приділена механізмам інтеграції з сервером, зокрема, через GraphQL-запити для отримання та обробки даних. Описані утиліти для роботи з глобальними даними, такими як заголовок і футер, продемонстрували ефективний підхід до обробки даних, що забезпечує динамічне наповнення інтерфейсу.

Завдяки розглянутій архітектурі клієнтської частини та її інтеграції із серверною, застосунок здатний забезпечити ефективну роботу з даними, гнучкість і зручність для користувача, а також підтримку сучасних стандартів розробки вебзастосунків.

4 ПРИКЛАДИ РОБОТИ ЗАСТОСУНКУ

4.1 Огляд головної сторінки

Головна сторінка інтернет-магазину створює перше враження та забезпечує зручну навігацію для користувачів. Її структура включає кілька ключових компонентів:

– хедер (Header): містить навігаційне меню з основними розділами сайту, такими як каталог, кошик, сторінка входу або реєстрації; відображає логотип і забезпечує швидкий доступ до основних функцій;
– футер (Footer): додаткові посилання, а саме соціальні іконки для переходу до сторінок магазину у відповідних соціальних мережах;
– вітальний банер: розташовується в центрі головної сторінки та привертає увагу відвідувачів; містить повідомлення, що надає користувачам загальну інформацію про магазин та підкреслює його спеціалізацію на ігровій техніці; під банером містить кнопку «каталог», яка веде до основного асортименту товарів;
– список категорій: представляє основні категорії товарів у вигляді плиток із зображеннями та посиланнями, щоб користувачі могли швидко перейти до відповідного розділу каталогу для перегляду товарів з обраної категорії.

Ці елементи забезпечують інтуїтивно зрозумілу навігацію та привабливий візуальний вигляд, покращуючи користувацький досвід і спонукаючи відвідувачів досліджувати магазин далі. Демонстрація сторінки наведена на рисунку 4.1.

Кнопки на хедері змінюються залежно від того, чи авторизований користувач. Коли користувач не авторизований, на хедері відображаються кнопки для основних функцій: Головна, Каталог та Увійти в обліковий запис. Кнопка «Головна» веде на основну сторінку інтернет-магазину, кнопка

«Каталог» відкриває доступ до списку товарів, а «Увійти в обліковий запис» дозволяє користувачеві здійснити вхід для доступу до персональних налаштувань.

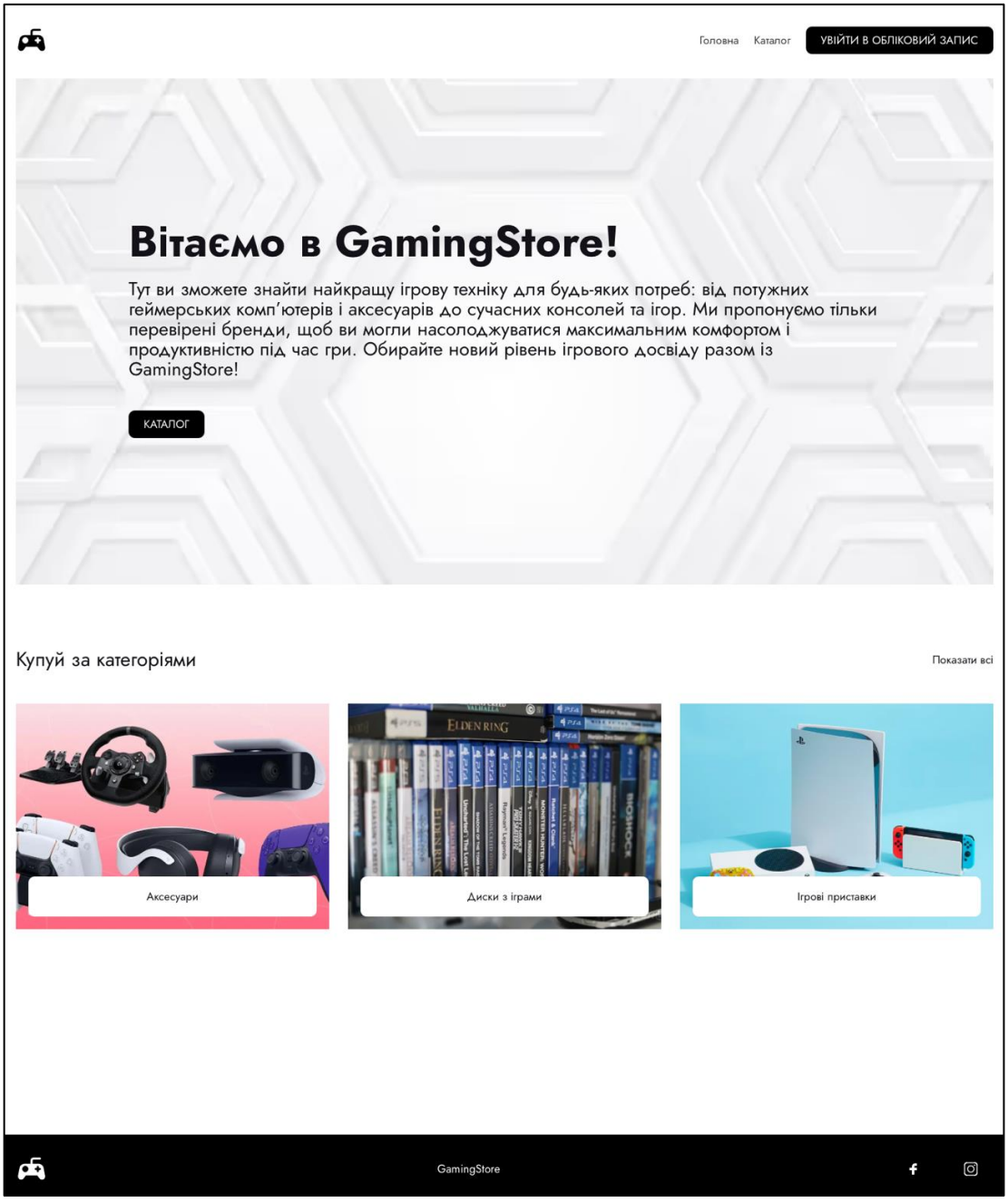


Рисунок 4.1 – Демонстрація головної сторінки інтернет-магазину

Коли користувач авторизований, на хедері з'являються додаткові кнопки, які відображають доступ до персональних функцій: Головна, Каталог, Обліковий запис та Кошик. Кнопка «Обліковий запис» дає доступ до особистого кабінету, де користувач може переглянути свою інформацію та

виконанні замовлення, а «Кошик» відкриває доступ до вибраних товарів перед оформленням замовлення. Ця адаптивна навігація полегшує користування сайтом, надаючи користувачу зручний доступ до потрібних функцій залежно від його статусу. Демонстрація цієї логіки наведена на рисунку 4.2.

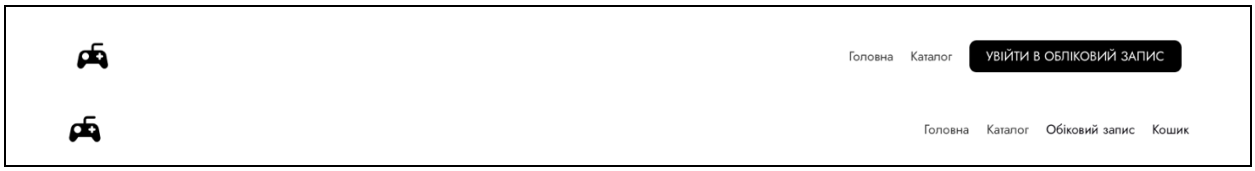


Рисунок 4.2 – Демонстрація зміни елементів навігації хедера

4.2 Огляд сторінок авторизації та реєстрації

Сторінки авторизації та реєстрації забезпечують доступ до особистого кабінету користувача.

Сторінка авторизації дозволяє користувачам увійти до свого облікового запису, ввівши електронну пошту та пароль. У разі неправильного введення даних система відображає повідомлення про помилку, пропонуючи спробувати знову або скористатися функцією відновлення пароля. Демонстрація цього процесу наведена на рисунку 4.3.

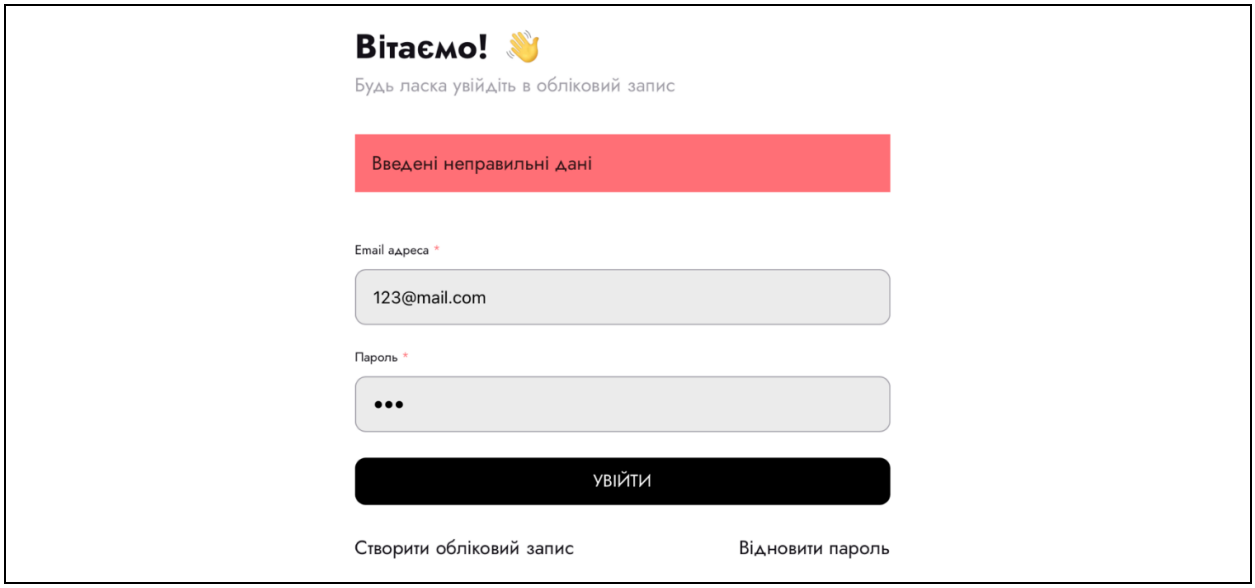


Рисунок 4.3 – Демонстрація повідомлення про невдалий вхід

Сторінка реєстрації дозволяє створити новий обліковий запис. Користувач вводить своє ім’я, електронну пошту та пароль та підтвердження пароля. Після успішної реєстрації користувач входить до системи. Приклад сторінки показано на рисунку 4.4.

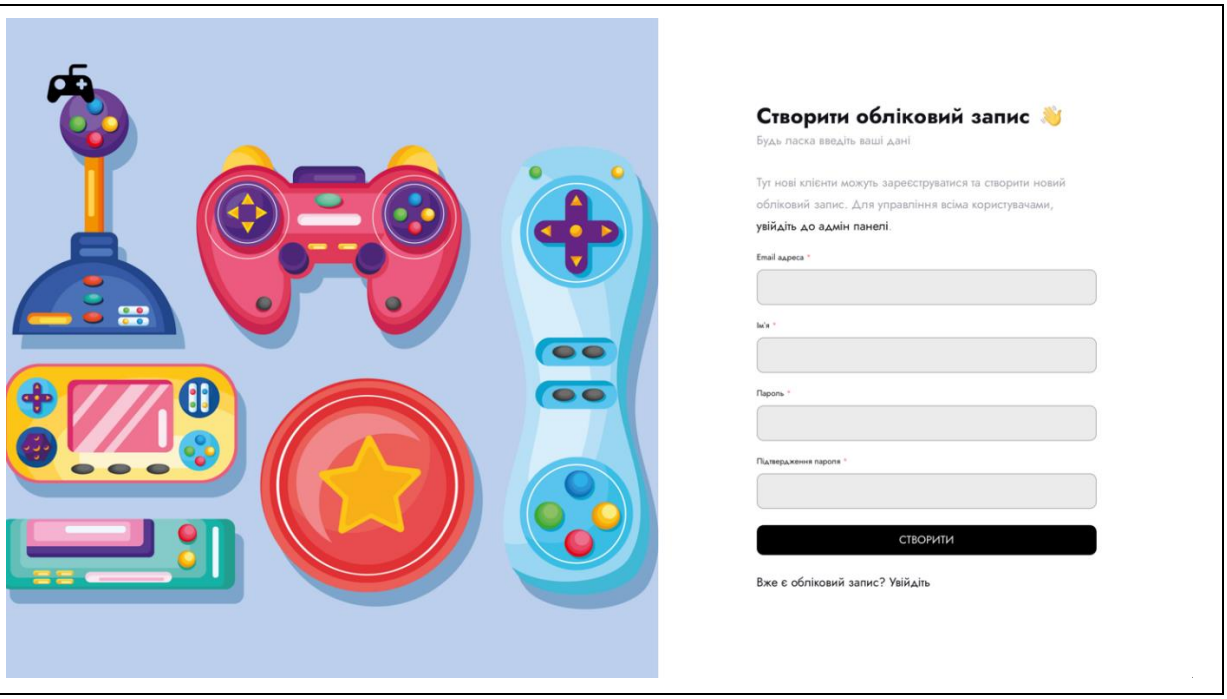


Рисунок 4.4 – Демонстрація сторінки реєстрації

4.3 Огляд сторінок адміністратора

Далі наведено опис функціоналу сторінок адміністратора.

4.3.1 Огляд сторінок управляння продуктами

Сторінка з продуктами надає адміністраторам можливість переглядати список усіх створених продуктів. Вона забезпечує функції фільтрації продуктів за різними критеріями, що дозволяє швидко знаходити потрібні товари. Крім того, на сторінці є можливість створювати нові продукти, та видаляти наявні. Демонстрація сторінок наведена на рисунках 4.5 – 4.6.

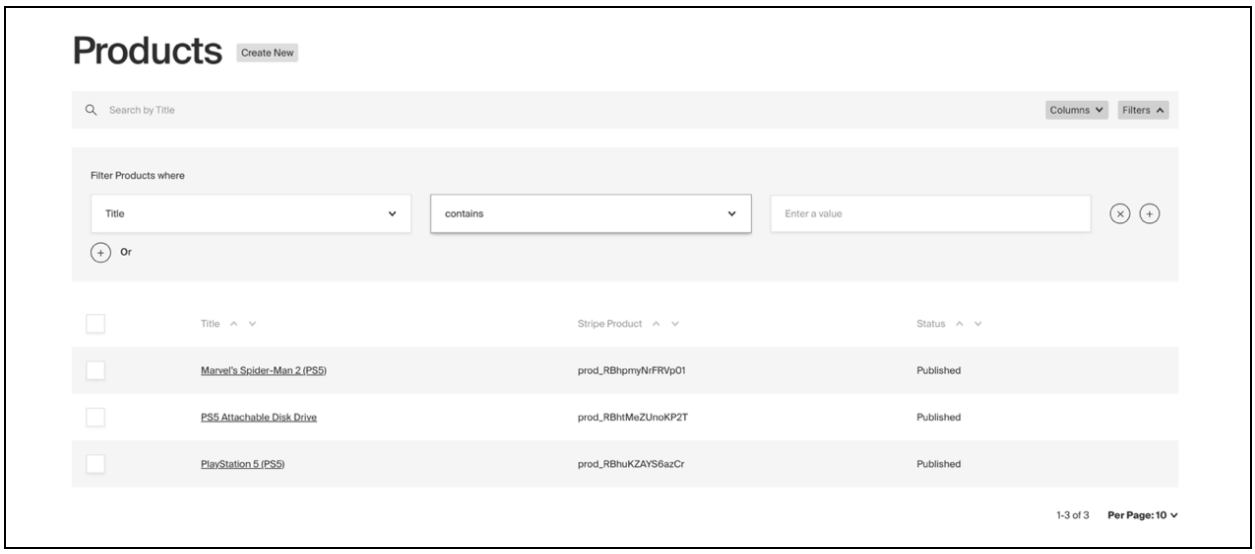


Рисунок 4.5 – Демонстрація сторінки з продуктами

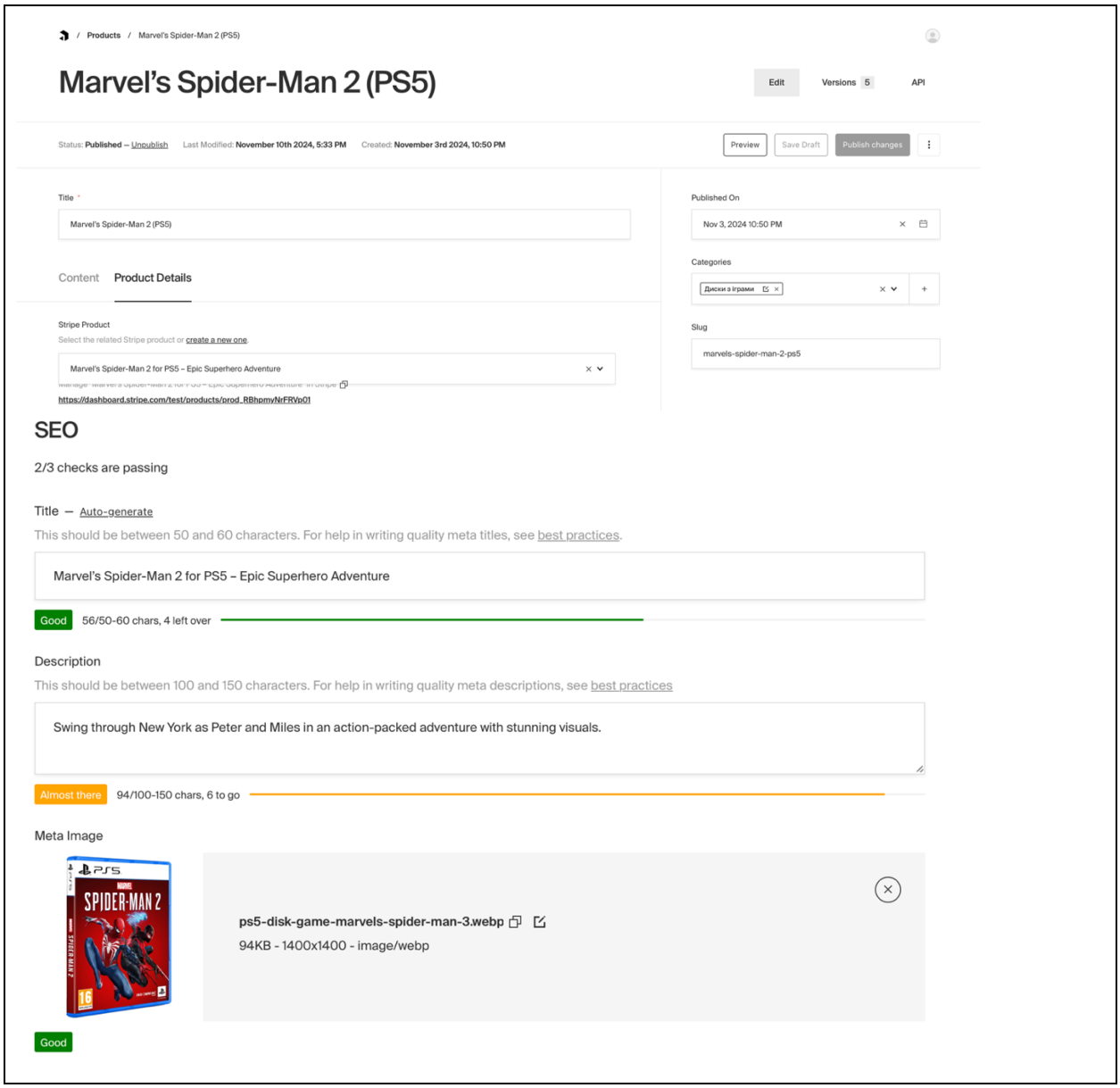


Рисунок 4.6 – Демонстрація сторінки створення або редагування продукту

Сторінка створення/редагування продукту дає адміністраторам змогу вводити основну інформацію про продукт. Це охоплює введення назви товару, доєднання продукту до Stripe для подальшої інтеграції з платіжною системою, а також додавання SEO-даних, таких як назва та опис для покращення видимості в пошукових системах. На цій сторінці можна завантажити зображення продукту, що є важливим для його візуального представлення на сайті.

Крім того, адміністратор може додавати продукт до однієї або кількох категорій, що спрощує його пошук на сайті. Також є можливість редагувати slug продукту, що дозволяє налаштувати URL-адресу для кращого сприйняття користувачами та пошуковими системами. Ці функції забезпечують гнучкість і зручність у керуванні асортиментом товарів магазину, допомагаючи адміністраторам швидко оновлювати інформацію про продукти.

4.3.2 Огляд сторінок керування категоріями

Сторінка управління категоріями надає адміністраторам інструменти для ефективного керування категоріями товарів у магазині.

Сторінка категорій відображає список усіх створених категорій, що дозволяє переглядати доступні категорії, фільтрувати їх за різними критеріями та знаходити потрібні. Користувач має можливість видаляти непотрібні категорії, а також створювати нові, що дозволяє організовувати асортимент товарів залежно від потреб бізнесу.

Сторінка створення/редагування категорії дає адміністраторам можливість вводити основні дані для нової або редагованої категорії. Це охоплює введення назви категорії, що допомагає чітко ідентифікувати її серед інших. Також є можливість додати зображення для категорії, що робить її більш привабливою для користувачів і покращує візуальне представлення на сторінках сайту.

Крім того, адміністратор може прив’язати категорію до батьківської категорії, що дозволяє створювати ієрархічну структуру для зручнішої навігації по магазину. Ще однією важливою функцією є можливість редагувати breadcrumbs для категорії, що забезпечує покращену навігацію по сайту, дозволяючи користувачам легше орієнтуватися в структурі інтернет-магазину. Демонстрація сторінок наведена на рисунках 4.7 – 4.8.

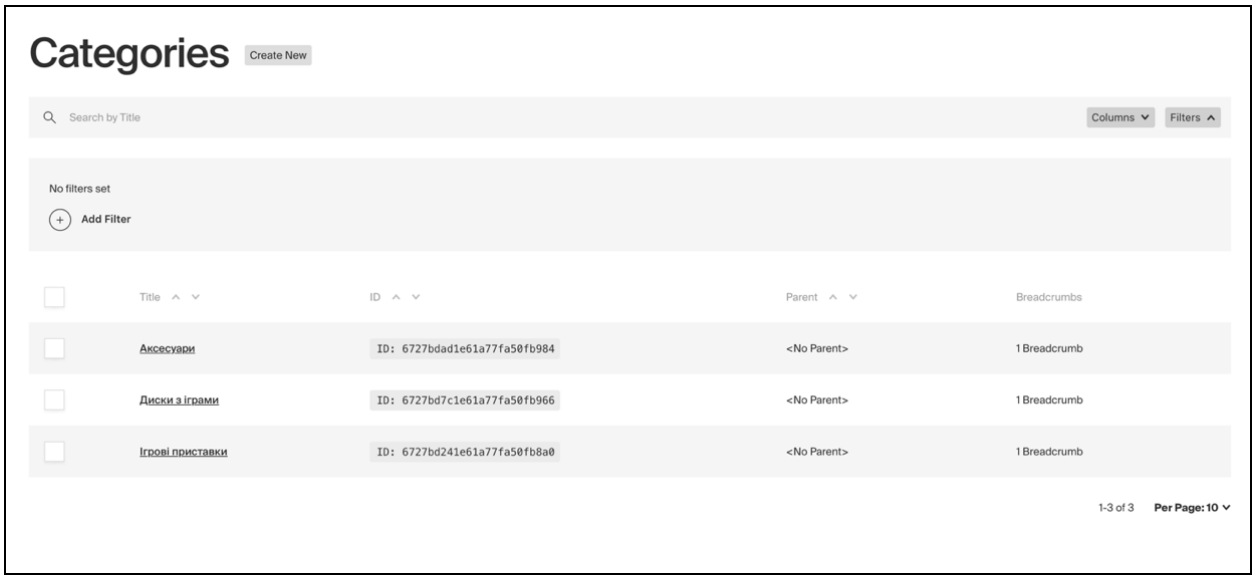


Рисунок 4.7 – Демонстрація сторінки з категоріями

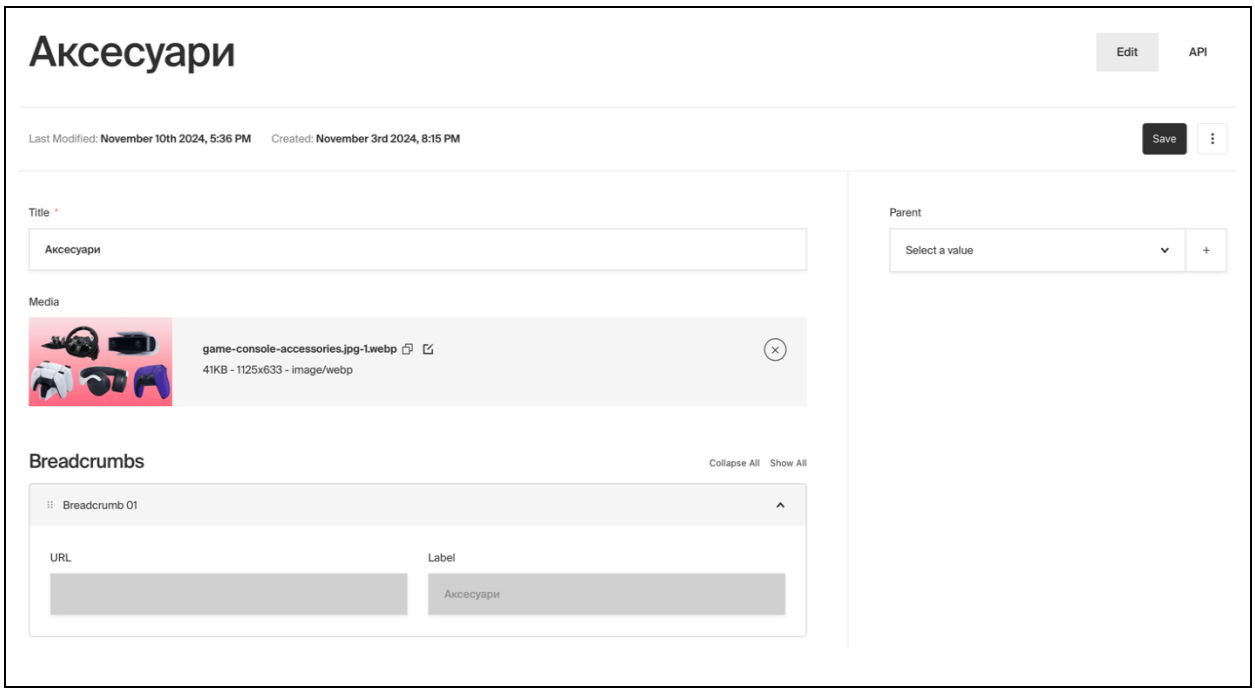


Рисунок 4.8 – Демонстрація сторінки створення або редагування категорії

4.3.3 Огляд сторінок керування користувачами

Сторінка управління користувачами дозволяє адміністраторам управляти обліковими записами користувачів інтернет-магазину. Вона містить список усіх зареєстрованих користувачів, що дає можливість переглядати їхні дані, фільтрувати за різними критеріями, видаляти непотрібні акаунти, а також створювати нові облікові записи. Демонстрація сторінок наведена на рисунках 4.9 – 4.10.



Рисунок 4.9 – Демонстрація сторінки з користувачами

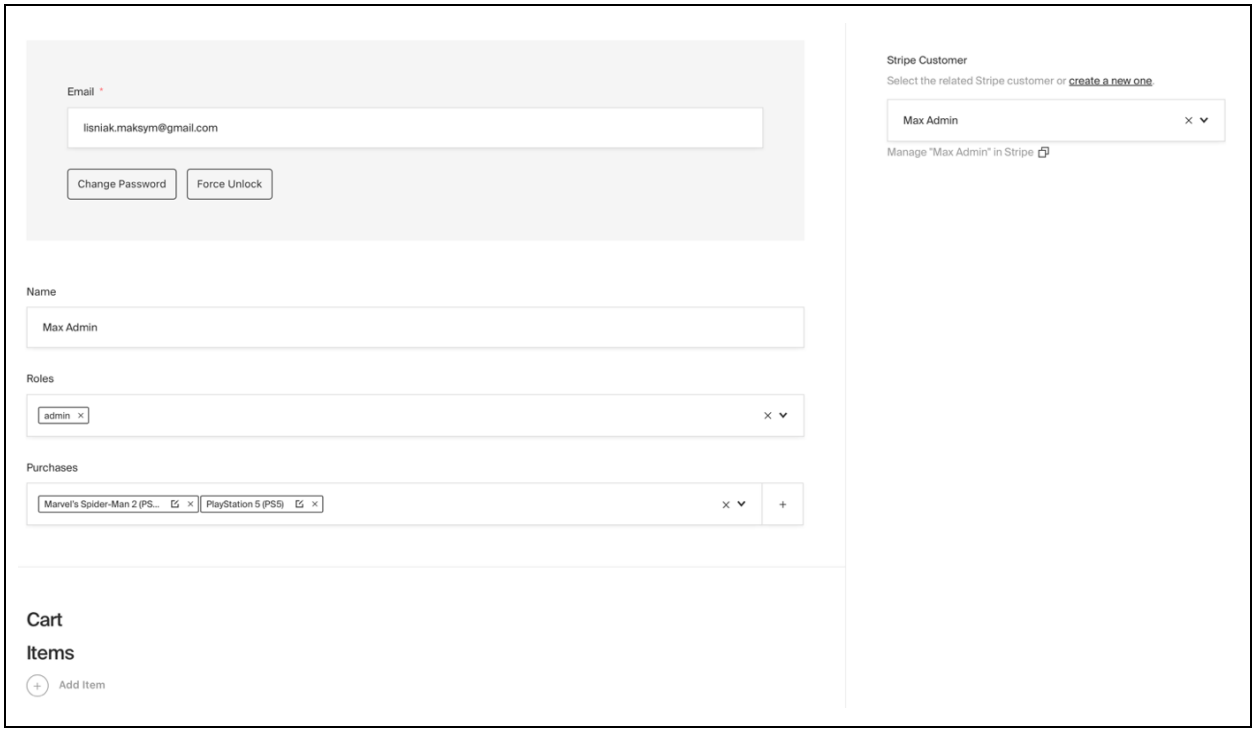


Рисунок 4.10 – Демонстрація сторінки створення або редагування користувачів

Сторінка створення/редагування користувача дозволяє адміністраторам редагувати дані користувачів або додавати нових. Тут можна ввести основні відомості, такі як ім’я, email, а також змінити пароль для користувача. Крім цього, адміністратор має можливість вказати роль користувача, що дозволяє регулювати його доступ до різних функцій системи, визначаючи, чи буде він мати обмежений доступ або адміністраторські повноваження.

Також на сторінці редагування користувача є можливість переглянути та редагувати інформацію про покупки користувача, товари в його кошику, що дозволяє оперативно реагувати на запити або проблеми з оформленням замовлень. Важливою функцією є можливість прив’язати **Stripe-користувача** для інтеграції з платіжною системою, що забезпечує зручний процес обробки платіжних даних та управління фінансовими операціями через Stripe.

4.3.4 Огляд сторінок керування сторінками

Сторінка управління сторінками інтернет-магазину надає адміністраторам можливість переглядати, фільтрувати, видаляти та створювати нові сторінки магазину. Серед створених сторінок є домашня сторінка, сторінка продуктів, та кошика. Демонстрація сторінки наведена на рисунку 4.11.

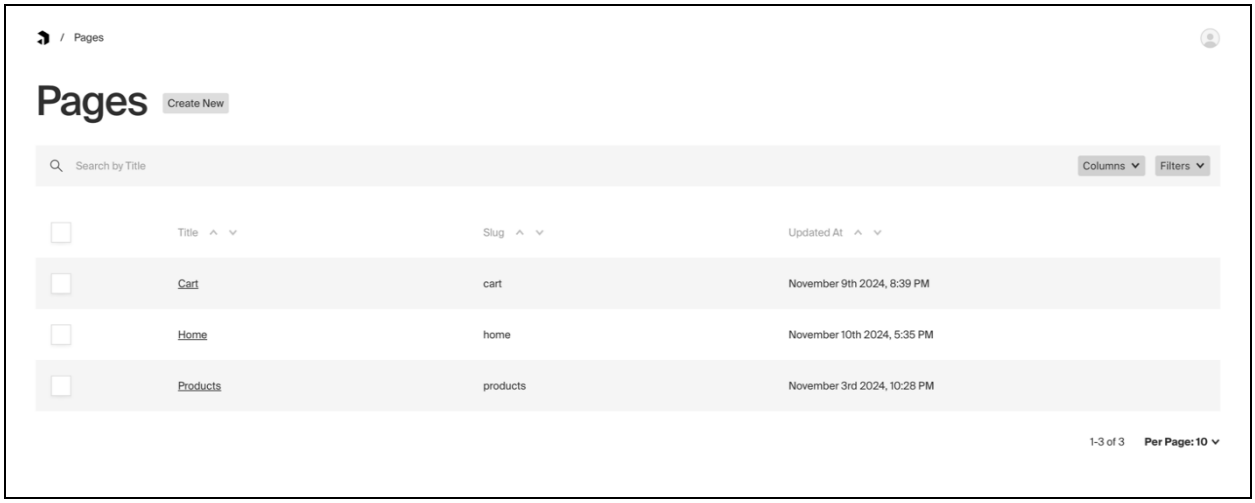


Рисунок 4.11 – Демонстрація сторінки з сторінками

Далі наведено приклад налаштувань створеної сторінки Cart (див. рис. 4.12).

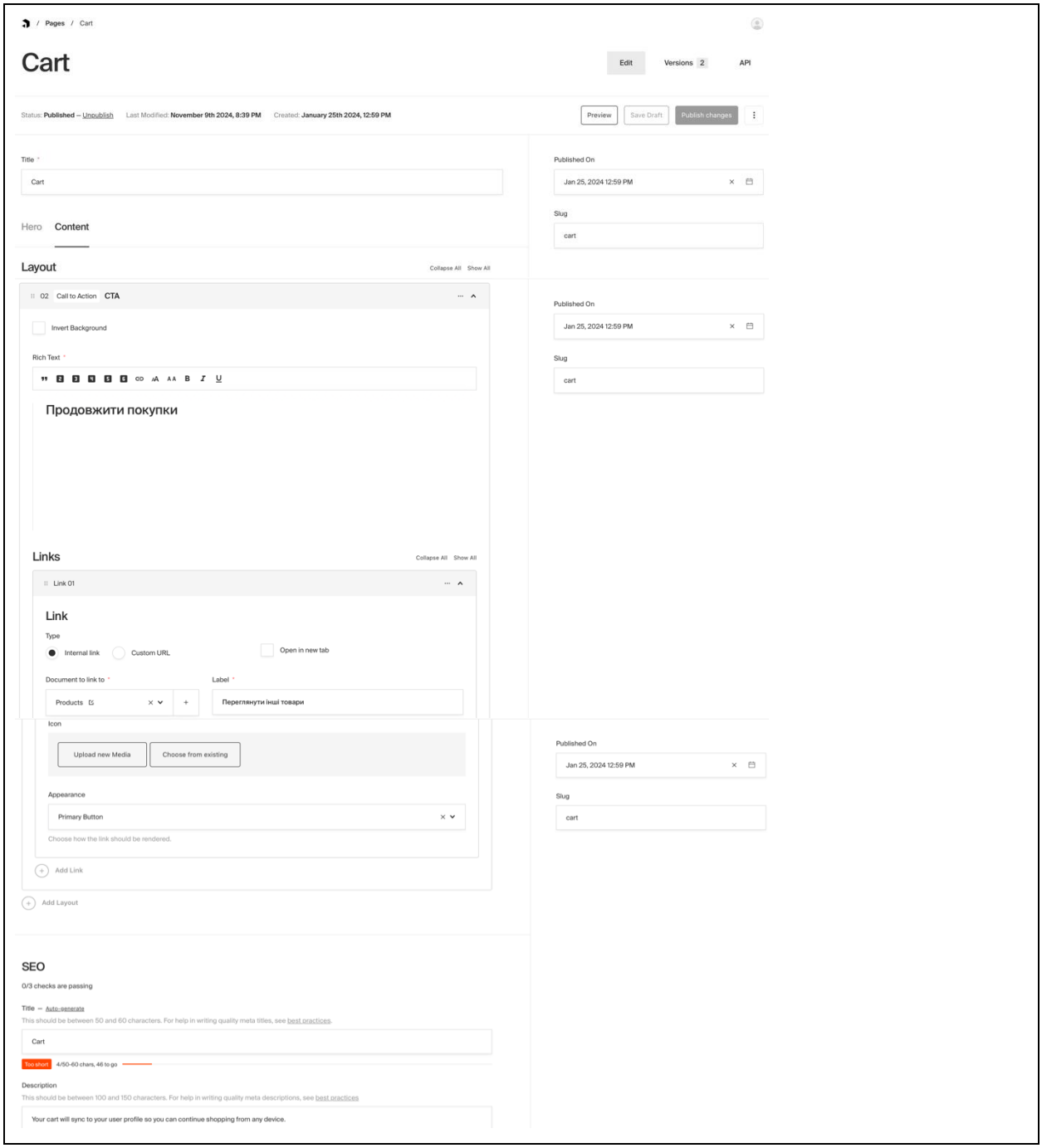


Рисунок 4.12 – Демонстрація сторінки створення або редагування сторінки – Cart

Панель адміністратора дає адміністраторам гнучкість у налаштуванні та управлінні контентом сторінки кошика. Тут можна змінювати назву, контент і SEO-інформацію сторінки, включаючи метатеги (назва, опис, зображення).

Окрім цього, є можливість редагувати slug сторінки для коректної адресації в URL.

Розділ layout дозволяє змінювати вигляд сторінки. Можна змінювати зміст блоків, їх порядок, та елементи, що вони включають. Додатково, додана секція Call to Action дозволила додати текст з кнопкою та посиланням, що пропонує користувачу продовжити покупки.

Сторінка Products також включає розділ layout, що є невіддільним елементом її структури. У цьому розділі була додана секція Archive Block, яка відповідає за відображення списку продуктів. Її конфігурація передбачає налаштування таких параметрів, як тип наповнення, вибір колекції, фільтрація за категоріями та кількість продуктів, що відображаються на одній сторінці. Демонстрація сторінки Products наведена нижче на рисунку 4.13.

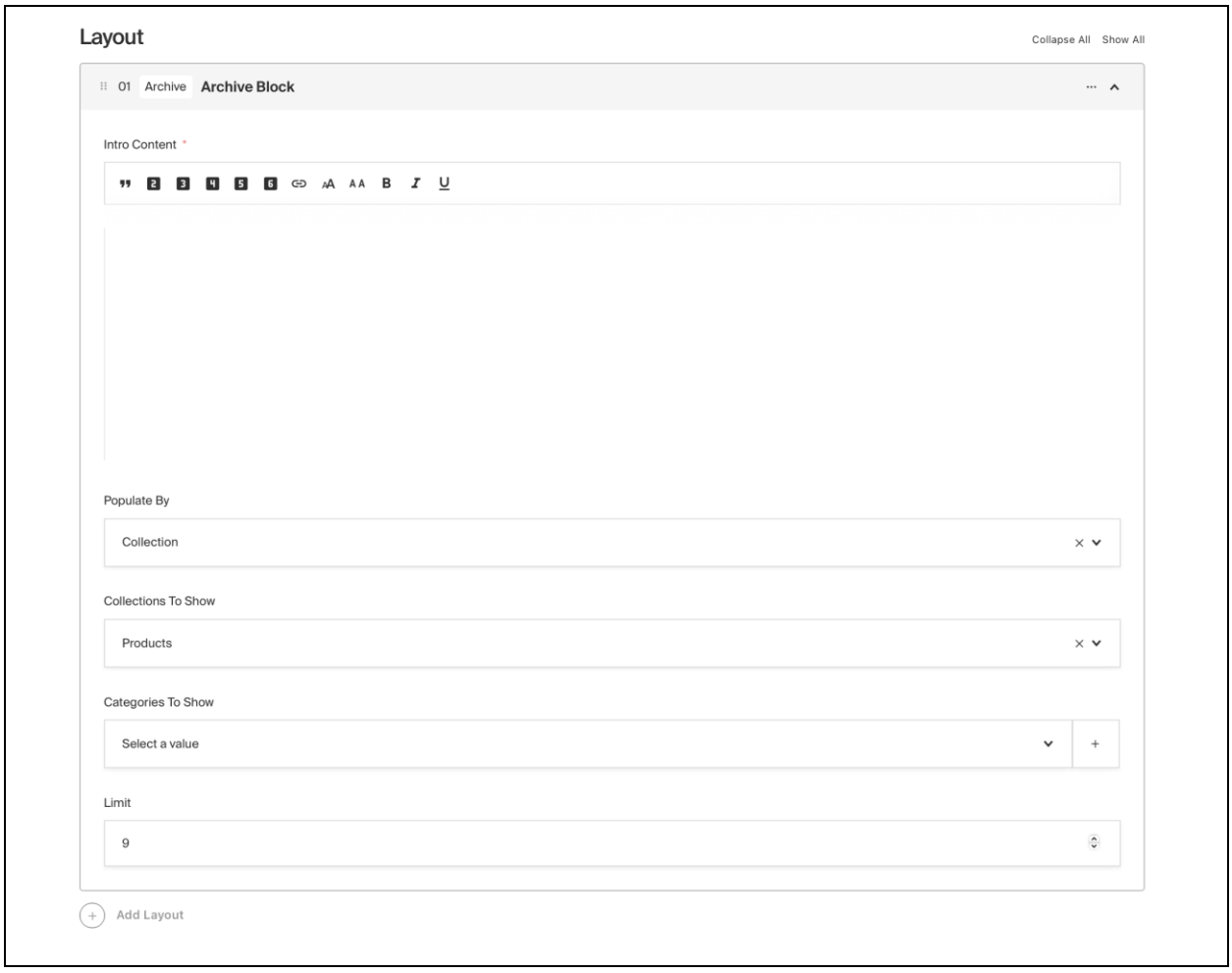


Рисунок 4.13 – Демонстрація сторінки створення або редагування сторінки – Products

Код клієнтського застосунку забезпечує інтеграцію цих компонентів і відповідає за їхню стилізацію та функціональність. З іншого боку, CMS дозволяє зручно управляти контентом сторінок, змінювати їхню структуру та наповнення без необхідності втручання в код. Такий підхід дозволяє розділити відповідальності: розробники фокусуються на створенні функціонального інтерфейсу, а контент-менеджери – на наповненні сторінок.

Компоненти для відображення сторінки Cart розташовані у директорії /app/(pages)/cart, а для Products у /app/(pages)/products, Директорія /(pages), своєю чергою, містить компоненти для відображення інших сторінок інтернет-магазину, таких як Cart, Checkout, Account тощо. Додатково, ці сторінки, та іншими, можна створити та налаштувати в панелі адміністратора CMS. Це забезпечує ще більшу гнучкість і зручність у керуванні контентом.

Адміністратори можуть редагувати вміст, змінювати структуру сторінок, додавати нові секції або налаштовувати SEO-інформацію без необхідності змінювати код. Це значно прискорює процес оновлення сторінок і дозволяє оперативно реагувати на зміну бізнес-вимог чи побажань клієнтів.

4.4 Огляд сторінки перегляду товарів

Сторінка перегляду товарів виконує роль каталогу, на якому представлений список доступних продуктів. Вона забезпечує зручну навігацію та пошук за допомогою інтегрованих фільтрів.

Фільтри дозволяють користувачам сортувати товари за категоріями, вибираючи лише ті, що належать до конкретної групи. Крім того, можна налаштувати сортування за датою додавання, що дає змогу швидко знайти найновіші товари.

Демонстрація сторінки, та функціоналу фільтрації за категоріями наведена на рисунках 4.14 – 4.15.

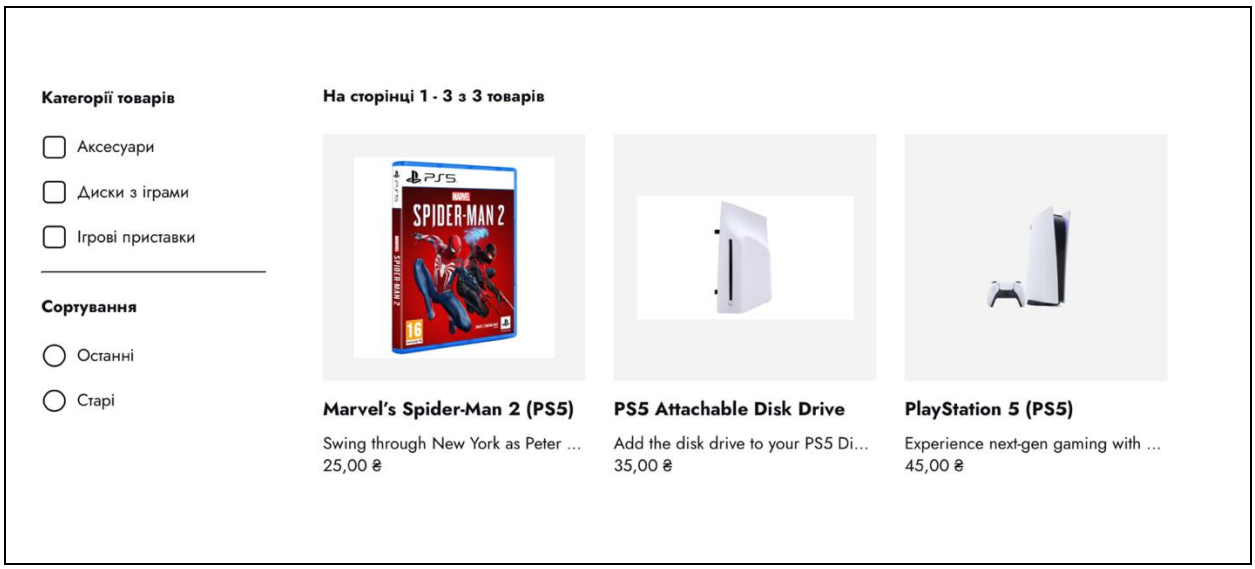


Рисунок 4.14 – Демонстрація сторінки перегляду товарів



Рисунок 4.15 – Демонстрація функціоналу фільтрації товарів за категорією

4.5 Огляд сторінки перегляду товару

Сторінка перегляду товару зосереджена на відображенні інформації про продукт: назви, опису, зображення, ціни, категорії та статусу наявності. Інтерактивна частина сторінки включає кнопку для додавання товару до кошика, а також секцію з пов'язаними товарами, що забезпечує користувача додатковими варіантами вибору. Демонстрація сторінки наведена на рисунку 4.16.

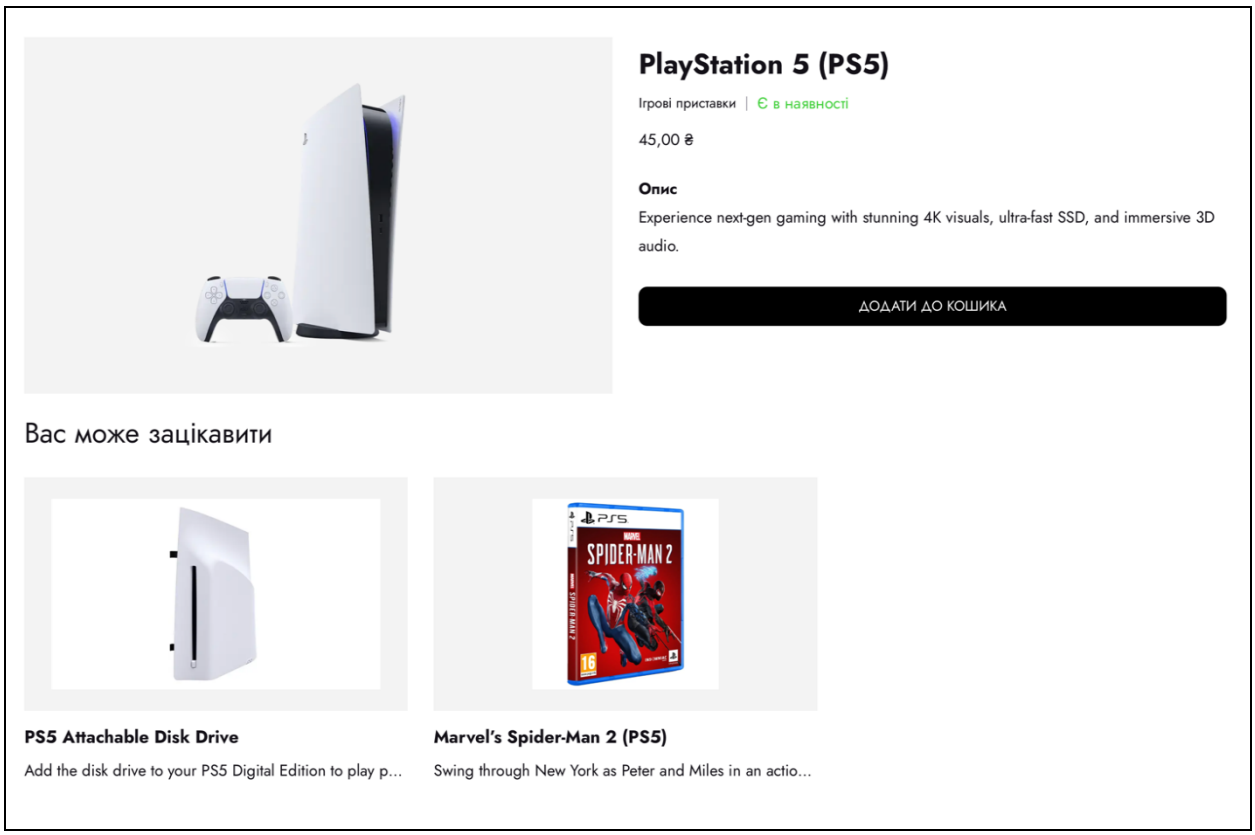


Рисунок 4.16 – Демонстрація сторінки перегляду товару

4.6 Огляд сторінки кошика

Сторінка кошика надає користувачеві зручний інтерфейс для керування товарами, доданими до кошика (див. рис. 4.17).

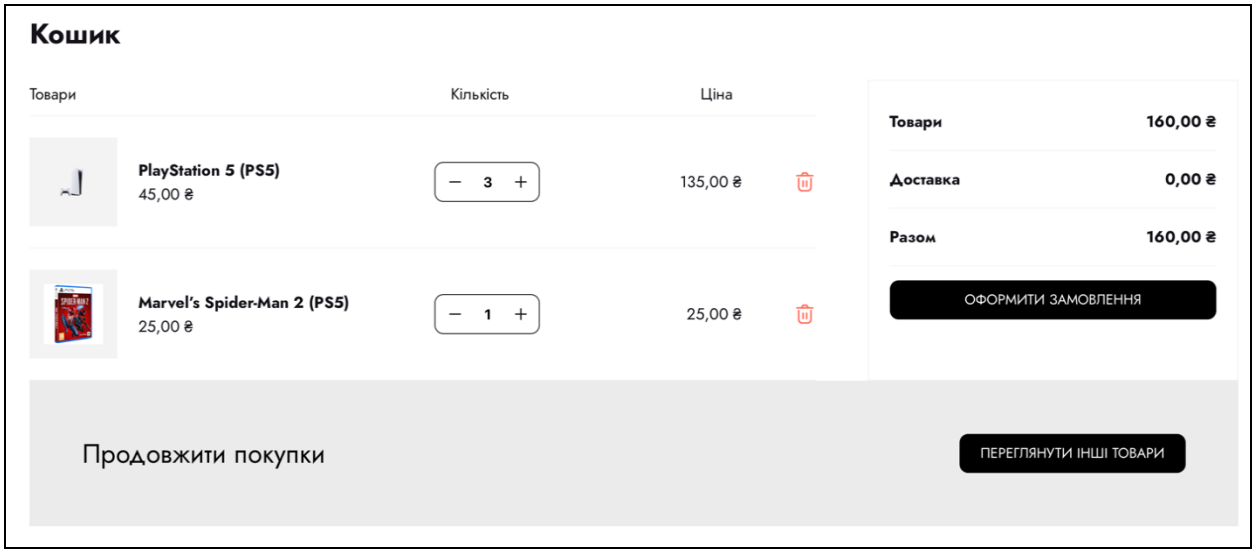


Рисунок 4.17 – Демонстрація сторінки кошика

Вона містить список обраних товарів із можливістю змінювати їх кількість або видаляти окремі позиції. Біля списку відображається остаточна сума замовлення. Для подальших дій передбачені кнопки: одна для переходу до оформлення замовлення, інша – для повернення до каталогу товарів, що заохочує користувача продовжити покупки.

4.7 Огляд сторінки оформлення замовлення

Сторінка оформлення замовлення завершує процес покупки (див. рис. 4.18). Вона дає можливість користувачеві переглянути перелік обраних товарів із фінальною вартістю. У відповідних полях користувач вводить платіжну інформацію та контактні дані.

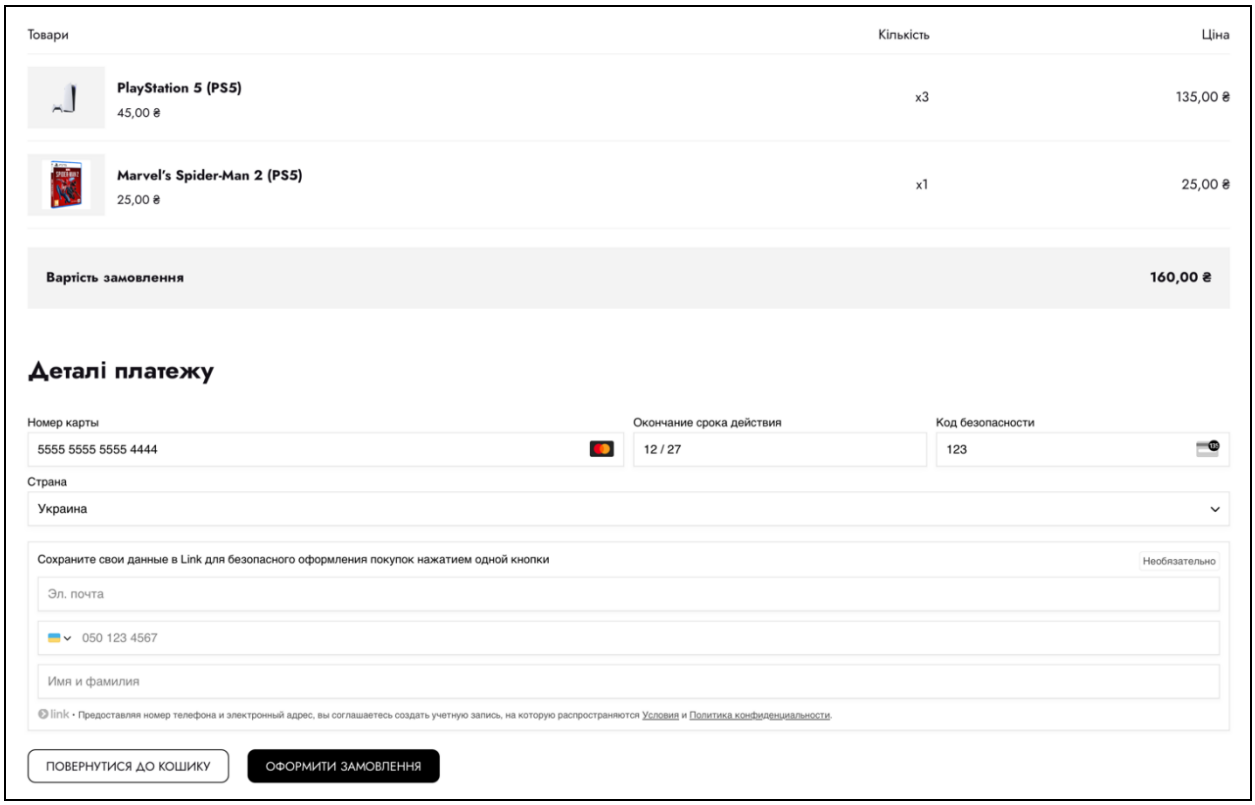


Рисунок 4.18 – Демонстрація сторінки оформлення замовлення

Після натискання кнопки «Оформити замовлення» система обробляє запит, і у разі успішної транзакції, користувач перенаправляється на сторінку

з повідомленням про успішне оформлення замовлення (див. рис. 4.19). Ця сторінка містить підтвердження замовлення, деталі транзакції та подяку за покупку.

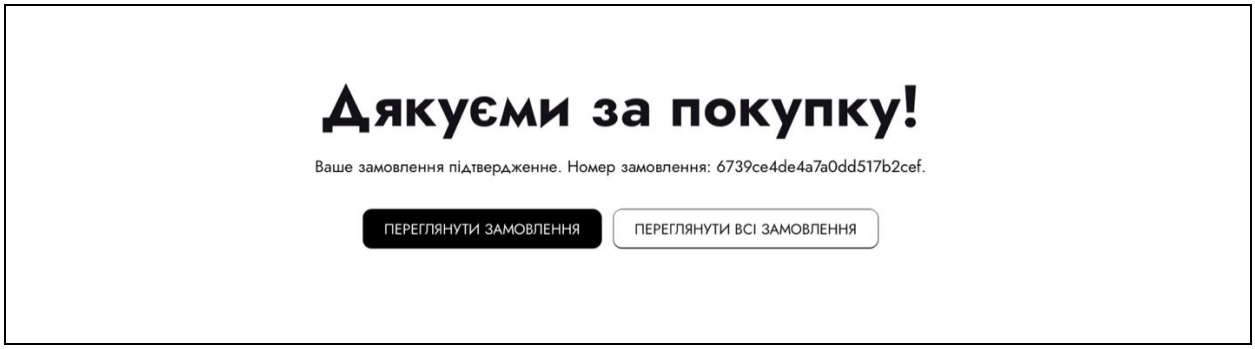


Рисунок 4.19 – Демонстрація сторінки повідомлення про успішне замовлення

4.8 Висновки до розділу 4

У цьому розділі було детально розглянуто структуру та функціонал ключових сторінок інтернет-магазину, розробленого за допомогою Next.js та Payload CMS. Проведено огляд головної сторінки, яка містить основні елементи навігації, сторінок авторизації та реєстрації, які забезпечують можливість входу та створення облікового запису для користувачів. Також описано сторінки адміністративної панелі, які дозволяють керувати продуктами, категоріями, користувачами та контентом окремих сторінок інтернет-магазину.

Окрему увагу приділено сторінкам перегляду товарів і конкретного товару, які забезпечують зручний пошук і детальне ознайомлення з продукцією. Описано функціонал сторінок кошика та оформлення замовлення, що дають змогу користувачам легко переглядати обрані товари, змінювати їх кількість і завершувати покупки.

Таким чином, розділ демонструє функціонал розробленого інтернет-магазину, висвітлюючи ключові можливості як для користувачів, так і для адміністраторів.

ВИСНОВКИ

У ході виконання роботи був успішно розроблений інтернет-магазин, що використовує сучасні технології, такі як Payload CMS, Next.js, MongoDB, GraphQL, SCSS та TypeScript. Проєкт має гнучку та масштабовану архітектуру, яка дозволяє керувати контентом, легко розширювати функціональність та забезпечувати зручний досвід для користувачів.

Основна увага була зосереджена на адаптації серверної частини під специфічні функціональні вимоги проєкту, зокрема через налаштування Payload CMS для оброблення та зберігання даних. Було визначено структури даних для основних колекцій, таких як користувачі, продукти, замовлення, категорії та медіаресурси. Для деяких полів додано бізнес-логіку через хуки, що автоматизує процеси та забезпечує валідацію даних, створюючи таким чином гнучку архітектуру, здатну адаптуватися до змін і потреб інтернет-магазину.

Клієнтська частина проєкту була розроблена за допомогою Next.js, що дозволило інтегрувати її з серверною частиною через GraphQL. Це сприяло оптимізації обміну даними між клієнтом і сервером та динамічному оновленню інтерфейсу. Ключові сторінки магазину, такі як авторизація, кошик, замовлення та адміністративна панель, були налаштовані з фокусом на зручність для користувачів та адміністраторів.

Окремо було налаштовано інтеграцію з платіжною системою Stripe для безпечних і зручних платежів, що є важливим аспектом для онлайн-торгівлі. Після налаштування, система успішно проходила тестування, що підтвердило правильність роботи як платіжних операцій, так і загальної функціональності магазину.

Таким чином, було створено масштабоване й ефективне рішення, яке відповідає вимогам сучасного інтернет-магазину і здатне легко адаптуватися до змін у функціональних вимогах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Next.js by Vercel – The React Framework. URL: https://nextjs.org (дата звернення: 25.03.2024).

2. Payload: The fastest growing open-source headless CMS. URL: https://payloadcms.com (дата звернення: 26.03.2024).

3. Stripe Plugin | Documentaion | Payload. URL: https://payloadcms.com/docs/plugins/stripe (дата звернення: 27.03.2024).

4. MongoDB | Documentaion | Payload. URL: https://payloadcms.com/docs/database/mongodb (дата звернення: 28.03.2024).

5. GraphQL Overview | Documentation | Payload. URL: https://payloadcms.com/docs/graphql/overview (дата звернення: 29.03.2024).

6. TypeScript Overview | Documentaion | Payload. URL: https://payloadcms.com/docs/typescript/overview (дата звернення: 01.04.2024).

7. Customizing CSS & SCSS | Documentation | Payload. URL: https://payloadcms.com/docs/admin/customizing-css (дата звернення: 02.04.2024).

8. Open Source Node.js Headless CMS. URL: https://strapi.io (дата звернення: 03.04.2024).

9. The Best React-Based Framework | Gatsby. URL: https://www.gatsbyjs.com (дата звернення: 04.04.2024).

10. Collection Config | Documentation | Payload. URL: https://payloadcms.com/docs/configuration/collections (дата звернення: 08.04.2024).

11. Global Config | Documentation | Payload. URL: https://payloadcms.com/docs/configuration/globals (дата звернення: 09.04.2024).

12. Payloadcms/template-ecommerce. URL: https://github.com/payloadcms/template-ecommerce?tab=readme-ov-file (дата звернення: 15.03.2024).