Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину

Виконав: студент 2 курсу, групи 2АКІТ-22м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Богдан САВІН
Ім'я ПРІЗВИЩЕ

Керівник: д.т.н., доцент, т.в.о.зав. кафедри КСУ
степінь, звання, посада

Марія ЮХИМЧУК
Ім'я ПРІЗВИЩЕ

«1» 12 2023 р.

Опонент: к.т.н., доцент кафедри АІІТ
степінь, звання, посада

Марія БАРАБАН
Ім'я ПРІЗВИЩЕ

«5» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
Марія ЮХИМЧУК
« 7 » 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Савіну Богдану Дмитровичу .
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину
керівник роботи Юхимчук Марія Сергіївна
затверджені наказом ВНТУ від "18" вересня 2023 року №247

2. Термін подання студентом роботи "01" грудня 2023 року

3. Вихідні дані до роботи: середовище розробки – VS Code IDE, адаптивний дизайн веб-застосунків, Зв'язок інтернет-магазину з адмін-панеллю реалізований через API, fullstack-застосунок, використання сучасних технологій

4. Зміст текстової частини: вступ, аналіз cms-систем керування інтернет магазином та огляд аналогів, вибір стеку технологій для розробки автоматизованої системи та інтернет магазину, розробка та тестування програмного забезпечення

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) Вступ, Мета, обєкт та предмет дослідження, Огляд та порівняння аналогів, Стек технологій, Послідовність дій при розробці, Діаграма моделей та зв'язків бази даних, Функціональність розробленої cms, Структура інтерфейсу cms-системи, UML-діаграма послідовності дій в інтернет магазині, Структура інтерфейсу, Інтернет-маназину, Зовнішній вигляд розроблених додатків, Висновки.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Бурєннікова Н. В., д.е.н., професор кафедри ЕПВМ.		

2. Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз та методи вирішення проблем електронної комерції	29.09.2023 р.	11.10.2023р.	
2	Огляд аналогів існуючих систем	12.10.2023 р.	24.10.2023р.	
3	Аналіз та вибір технологічного стеку на технологій для розробки автоматизованої системи та інтернет-магазину	25.10.2023 р.	09.11.2023р.	
4	Розробка програмного забебзпечення	10.11.2023 р.	17.11.2023 р.	
5	Тестування програмного забезпечення	18.11.2023 р.	18.11.2023 р.	
6	Оформлення пояснювальної записки	19.11.2023 р.	22.11.2023р.	
7	Розрахунок економічної частини	23.11.2023 р.	23.11.2023р.	
8	Публікації	24.11.2023 р.	25.11.2023р.	
9	Підготовка графічних матеріалів	26.11.2023 р.	26.11.2023р.	

Студент (підпис) Богдан САВІН (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи (підпис) Марія ЮХИМЧУК (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 681.5.004.41

Савін Б. Д. Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2023. 136 с.

На укр. мові. Бібліогр.: 60 назв; рис.: 30; табл. 11.

У магістерській кваліфікаційній роботі розроблено автоматизовану систему налагодження роботи організаційних процесів інтернет магазину. У оглядово-аналітичній частині роботи розглянуто особливості побудови сучасних автоматизованих систем управління, а також розглянуто тенденції у розвитку сучасного бізнесу електронної комерції. У теоретично-методичній частині розглянуто та обрано сучасний стек технологій та обгрунтовано вибір кожної технології. У практичній частині розроблено програмне забезпечення, що складається з двох додатків розроблених з використанням фреймворку NextJS, а саме: автоматизованої системи налагодження роботи організаційних процесів(CMS) та інтернет-магазин, зв’язок яких реалізовано за допомогою розробленої API URL, наведено результати тестування обох додатків. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 25 плакатів із результатами роботи.

Ключові слова: автоматизована система, управління, стек, CMS, інтернет-магазин, NextJS, API, додаток.

ANNOTATION

UDC 681.5.004.41

Savin B. D. Development of an automated system for debugging the organizational processes of an online store. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023. 136 p.

In Ukrainian speech Bibliography: 60 titles; Fig.: 30; table 11.

In the master's qualification work, an automated system for debugging the organizational processes of the online store was developed. In the review and analytical part of the work, the peculiarities of the construction of modern automated management systems are considered, as well as the trends in the development of modern e-commerce business. In the theoretical and methodological part, the modern stack of technologies is considered and selected, and the choice of each technology is justified. In the practical part, a software consisting of two applications has been developed with NextJS framework, namely: an automated system for debugging the work of organizational processes (CMS) and an online store, the connection of which is implemented using the developed API URL, the results of testing both applications are given. In the economic part, the technical level is analyzed and the cost of development implementation is calculated. The illustrative part consists of 25 posters with the results of the work.

Keywords: automated system, management, stack, CMS, online store, NextJS, API, application.

ЗМІСТ

ВСТУП

Актуальність. У сучасному бізнес-середовищі інтернет-магазини стають основними гравцями на ринку електронної комерції. Зростання популярності онлайн-торгівлі спричиняє зростання конкуренції серед підприємств, які намагаються привернути і утримати клієнтів. Окрім того, споживачі стають більш вимогливими і очікують швидкого, зручного та персоналізованого обслуговування.

З ростом конкуренції в електронній комерції, важливо мати ефективний інструмент для управління та вдосконалення процесів в магазині. Наш проект спрямований на створення адаптивної системи, що дозволить власникам інтернет-магазинів ефективно керувати контентом, відстежувати статистику та оптимізувати роботу свого бізнесу.

Ця магістерська робота присвячена розробці та впровадженню автоматизованої системи управління організаційними процесами інтернет-магазину.

Об'єктом дослідження є інтернет-магазин та його організаційні процеси. Розроблена система спрямована на оптимізацію цих процесів, полегшення керування контентом та забезпечення ефективності бізнесу в електронній комерції.

Предметом дослідження є розробка та впровадження автоматизованої системи, що об'єднує адміністративні та управлінські функції інтернет-магазину. Ця система охоплює весь життєвий цикл товарів, від їх додавання до відстеження продажів та аналізу діяльності.

Метою магістерської роботи є розробка автоматизованої системи управління інтернет-магазином, яка охоплює весь цикл від додавання товарів до відстеження оплати та аналізу продажів. Застосування та вивчення сучасних технологій та інструментів робить цей проект актуальним і конкурентоспроможним на ринку електронної комерції.

Аналіз проблем: Визначення основних проблем і викликів, з якими стикаються інтернет-магазини в процесі своєї роботи, включаючи управління

замовленнями, логістикою, інвентаризацією та взаємодію з клієнтами.

Практична цінність: Результати нашої роботи мають велику практичну цінність для бізнес-спільноти. Впровадження автоматизованої системи управління організаційними процесами інтернет-магазину дозволить підприємствам ефективніше ведення бізнесу, підвищити продуктивність та поліпшити обслуговування клієнтів.

Новизна одержаних результатів: однією з ключових особливостей проекту є впровадження сучасних підходів до управління інтернет-магазином. Було врахувано потреби не лише покупців, але й власників бізнесу, надаючи широкий функціонал для адміністрування та аналізу даних. Розроблювана система дозволяє створювати та керувати контентом, відстежувати оплати, а також надає докладну статистику щодо продажів та динаміки розвитку бізнесу.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції Всеукраїнська науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2023)».

Публікації: Савін Б.Д. «Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину», «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ», 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/author/submission/19749

1 АНАЛІЗ АВТОМАТИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ КОНТЕНТОМ ІНТЕРНЕТ МАГАЗИНОМ ТА ОГЛЯД АНАЛОГІВ

1.1 Аналіз проблеми автоматизованих систем управління контентом

Сучасний бізнес, зокрема інтернет-магазини, стикається з великою кількістю контенту, вимог до швидкості та точності обробки замовлень, а також необхідністю залучення та утримання клієнтів. У цьому контексті системи управління контентом (Content Management Systems, CMS) стали незамінним інструментом для ефективного функціонування та успішного розвитку бізнесу.

CMS система[1] – це програмне забезпечення, призначене для створення, редагування, оновлення та управління контентом на веб-сайті без потреби глибоких знань в програмуванні. На професійному жаргоні CMS ще називають "движок сайту". За даними агентства W3Techs, на CMS працює більше половини сайтів в інтернеті. Вона забезпечує легкий доступ до вмісту сайту та надає інтерфейс для його змін.

Важливість CMS для електронної комерції.

CMS системи виявилися критично важливими для інтернет-магазинів з ряду причин:

– Управління контентом: CMS дозволяють ефективно керувати контентом на сайті, що включає товари, описи, зображення та інші елементи. Це робить процес додавання інформації та редагування її більш доступним та продуктивним.

– Легкість оновлення: CMS спрощують процес оновлення веб-сайту, дозволяючи адміністраторам легко додавати нові товари, акції та іншу інформацію без необхідності втручання програмістів.

– Підтримка електронної комерції: CMS надають широкий спектр інструментів для управління електронною комерцією, включаючи можливість обробки замовлень, керування інвентарем, облік оплати та інші аспекти електронної торгівлі.

Під час розробки були враховані всі пункти, щоб система автоматизація була максимально ефективна, та вирішувала всі необхідні бізнес задачі.

Популярні CMS системи:

Існує безліч CMS систем, але деякі з найпопулярніших CMS систем[2], використовуваних в електронній комерції, включають:

- Shopify: Shopify – це веб-платформа для електронної комерції, яка надає широкі можливості для створення та управління інтернет-магазинами.
- WooCommerce: WooCommerce – це розширення для WordPress, яке дозволяє перетворити веб-сайт на потужний інтернет-магазин.
- Magento: Magento – це потужна платформа для електронної комерції, призначена для більших та більш складних проектів.
- OpenCart: OpenCart – це легка та зручна система для створення інтернет-магазинів з відкритим кодом.

При розробці моєї CMS системи для управління інтернет магазином використовувались деякі функції які схожі з функціоналом даних систем управління контентом в інтернет магазині, найпопулярніші розглянемо та проаналізуємо згодом.

CMS системи відіграють критичну роль у сучасному бізнесі, забезпечуючи легкий доступ до контенту, полегшуючи оновлення інформації та надаючи інструменти для управління електронною комерцією. Вони дозволяють інтернет-магазинам ефективно конкурувати на ринку та задовольняти вимоги сучасних споживачів. Тому бізнеси у даній сфері обов’язково повинні звертати увагу на всі плюси CMS систем та схилятись до вибору даних систем управління контентом при розробці інтернет магазинів і не тільки.

Сфера електронної комерції[3] є динамічною та швидкозмінною, і розуміння поточних тенденцій є ключовим для успішного функціонування інтернет-магазину. Ось кілька важливих тенденцій, які варто враховувати:

Персоналізація і рекомендації: Клієнти очікують персоналізованих пропозицій та рекомендацій.

Мобільна торгівля[4]: Зростання кількості користувачів мобільних пристроїв призводить до необхідності мати мобільно-адаптивні веб-сайти та додатки. Тому при розробці самого інтернет магазину це було враховано і дизайн

розроблявся під всі розширення екранів починаючи з великих комп'ютерних моніторів і закінчуючи маленькими екранами смартфонів

Омніканальність: Клієнти очікують можливості покупок через різні канали, включаючи онлайн, фізичний магазин і соціальні медіа.

Аналітика: на стороні адміна було розроблено зрозумілу аналітку аналізу даних, продажів, доступних товарів, аналіз попиту у вигляді графіку, автоматизації обслуговування клієнтів та вдосконалення процесів.

У сфері електронної комерції є інші важливі аспекти[5], які варто враховувати:

- Безпека: Захист особистих даних клієнтів та фінансових операцій є відразу важливою та законною вимогою, для задовілля даного пункту використовувався React фреймворк під назвою NextJS, в якому присутня функція SSR(Server Side Rendering), яка не дає доступу до важливих даних зі сторони клієнту.
- Аналітика: Збір та аналіз даних допомагає розуміти поведінку клієнтів, ефективність маркетингових кампаній та можливості для покращення.
- Маркетинг і реклама: Сучасні методи маркетингу[6], такі як використання соціальних медіа та рекламних платформ, допомагають залучати та утримувати клієнтів.
- Взаємодія з клієнтами: Побудова позитивних відносин з клієнтами через обслуговування.

1.2 Тенденції у розробці інтернет-магазинів та систем управління контентом.

Останні тенденції у розробці інтернет-магазинів[7] та систем управління контентом в магазині відзначаються швидким розвитком технологій. Використання фреймворків, таких як NextJS, спрямоване на покращення продуктивності та швидкості завантаження веб-сайтів. Технологія Progressive Web Apps (PWA) стає стандартом для забезпечення зручного користувацького досвіду на різних пристроях та в умовах обмеженого інтернет-з'єднання.

Трішки детальніше про PWA технологію.

PWA[8] – це цілий простір рішень, що лежить між класичними веб-сайтами та додатками для мобільних пристроїв. Саме це дозволяє їм утримувати на сьогоднішній день лідерство в зручності для користувачів. Це наступний етап розвитку адаптивної верстки веб-сайтів, яка повсюдно впроваджується зі збільшенням мобільного інтернет-трафіку протягом останніх 2-3 років. Однак ключовим словом у визначенні PWA можна вважати прикметник progressive – тобто, прогресивні.

Одні з найбільш широко використовуваних фреймворків для створення PWA-додатків – Vue.js, React.js та NextJS в зв’язці з FireBase[9] або з будь якою іншою ORM та базою даних, хоча фахівці використовують більш широкий спектр сучасних інструментів в роботі над PWA.

В той час як звичайні сайти мають здебільшого інформаційний характер, веб-додатки містять чимало функцій, що полегшують життя користувачам. Наприклад, вони дозволяють вільно обмінюватись повідомленнями, оплачувати товар, працювати з файлами онлайн. Веб-додаток зручний тим, що його не треба встановлювати, його можливості працюють з хмари.

Технічні характеристики прогресивних веб-додатків.

Концепція PWA зосереджена навколо створення кросплатформної програми з мінімальними витратами. Це тому, що PWA базуються на веб-технологіях, які працюють у будь-якій операційній системі.

Технологія PWA вимагає, щоб обмін даними відбувався за протоколом HTTPS[10]. Для цього необхідно встановити сертифікат SSL, який створює безпечне зашифроване з’єднання між внутрішнім і зовнішнім сервером. Дуже важливо, щоб на сайті не було посилань на незахищені ресурси, особливо якщо це інтернет-магазин, де безпека даних клієнтів має вирішальне значення.

Крім того, є ризик, що при посиланні на незахищені сайти деякі браузери не будуть відображати ці дані на екрані. Найчастіше проблема виникає з зображенням, що були взяті з інших ресурсів. Для коректного відтворення їх треба записати у себе або на сервіс, що працює через протокол HTTPS.

Перейдемо до переваг веб-додатків розроблених за принципом PWA:

– Автоматичне оновлення PWA.

Веб-додаток оновлюється автоматично у всіх користувачів, як тільки з'являється нова версія. За бажанням розробник може попереджувати користувачів про зміни завдяки системі кешування файлів веб-додатка.

– Розмір, менший за мобільний додаток.

PWA[11] розв'язує проблему обмеженого місця на смартфоні, оскільки розмір веб-додатка не перевищує 1-3 Мб. Це значно менше, ніж зазвичай має нативний застосунок. Переважну більшість даних PWA зберігає у хмарі, оскільки обмежений розмірами внутрішнього сховища, яке надається браузером для PWA (6% вільного місця для Chrome, 10% для Firefox, 50 Мб для Safari).

– Економія на розробці.

Вигідніше створити веб-додаток, ніж витрачати гроші окремо на сайт, і окремо на створення мобільного застосунку. При цьому варто зазначити, що кожен нативний додаток розрахований на певну операційну систему. Тому може знадобитися або найняти кілька команд розробників, або створювати один кросплатформний застосунок, що працює з різними ОС.

Які проблеми вирішує PWA[11].

Найбільш очевидне: веб-додаток дозволяє привернути увагу клієнта до свого товару, активно рекламувати бренд. Іконка, встановлена на головному екрані, буде завжди у полі зору, й коли знадобиться ваша послуга, клієнт швидше звернеться до вас, ніж до конкурентів.

Прогресивний веб-додаток дозволяє вирішити й інші завдання:

— Спростити для клієнта пошук вашої компанії.
— Просувати бізнес (наприклад, заохотити користувача знову звернутись до ваших послуг).
— Поліпшити торгівлю завдяки push-повідомленням про певний товар чи акції. Одночасно це дозволяє економити на рекламі.
— Швидко взаємодіяти з клієнтом – відправляти повідомлення про відбуття / доставлення товару, реагувати на різні питання, скарги.

— Обійти сайти конкурентів, що знаходяться на ліпших позиціях у пошуковиків.

Отже, ми з'ясували, що таке прогресивний веб-додаток – це різновид веб-застосунку, що може працювати за принципом нативного. PWA можуть ефективно вирішувати завдання бізнесу в найрізноманітніших сферах. Вони прості в інсталяції, зручні, дозволяють використовувати майже увесь нативний функціонал смартфонів, займають на пристрої дуже мало місця, економно використовують мобільний трафік та можуть ефективно працювати офлайн.

Значення прогресивного веб-додатку для бізнесу сьогодні важко переоцінити. Він є відмінною альтернативою складним сайтам і повноцінним мобільним додаткам, і це було враховано при розробці адміністративної панелі та інтернет-магазину.

1.3 Огляд аналогів існуючих автоматизованих систем управління контентом в електронній комерції.

Для більш глибокого розуміння розробки інтернет-магазину важливо вивчити існуючі аналоги. Проведення огляду роботи популярних інтернет-магазинів та систем управління контентом дозволяє визначити оптимальні практики та уникнути можливих труднощів. Аналіз аналогів також виявляє плюси та мінуси, які можуть бути враховані під час розробки системи.

Як ми раніше говорили, в сучасному цифровому світі інтернет-магазини стали ключовим елементом електронної комерції, забезпечуючи покупцям зручність та доступність в придбанні товарів та послуг. Розквіт цього сегмента бізнесу обумовлює необхідність вдосконалення та автоматизації організаційних процесів інтернет-магазинів.

В даному підрозділі буде проведено огляд деяких ключових аналогів CMS-систем, які використовуються для створення та управління інтернет-магазинами. Розгляд обраної трійки - Shopify, Magento та OpenCart - надасть глибший уявлення про їхні можливості та відмінності, допомагаючи визначити оптимальний набір функціоналу без якого не обійтись при розробці такого

програмного забезпечення, як система контролю організаційними процесами інтернет магазину.

Отже приступимо до огляду найпопулярнішої CSM під назвою Shopify:

Shopify[13] — це зручна система керування вмістом, яка розробляється з 2004 року та спеціально розроблена для інтернет-магазинів. За його допомогою численні підприємства продають продукти, в тому числі цифрові, по всьому світу. На даний момент це одне з найпопулярніших рішень для запуску проектів у сфері електронної комерції.

Через веб-сайти, створені на основі Shopify, можна продавати найрізноманітніші категорії товарів: від одягу, побутової техніки та дизайну інтер'єру до консультаційних послуг і вебінарів. Крім того, доступна доставка. Цінова політика дозволяє підібрати план, який підходить як для стартап-проекту на початковій стадії (мінімальна щомісячна плата - $24), так і для магазину з великим товарообігом і великою кількістю товарів.Основні переваги платформи Shopify

Система має велику кількість переваг[14]. Ви зможете в цьому переконатись, почавши з нею працювати. А поки що перерахуємо найістотніші плюси Shopify:

— Швидкий старт проєкту. Бібліотека готових блоків, доступна на платформі, їх злагоджена взаємодія та режим конструктора дозволяють швидко створювати магазини під різні завдання.
— Гнучка тарифна сітка. Наприклад, якщо ваша мета – реалізація товарів через соцмережі, то Вам доступний бюджетний тарифний план Shopify Lite всього за 9 доларів на місяць.
— Максимально широкий функціонал. Перелік додаткових опцій, плагінів та програм включає понад 4 тисячі позицій. Тобто всього кілька кліків можна під'єднати до магазину необхідне стороннє рішення або додати будь-яку функцію.
— Відсутність проблем із пошуком розробника. З платформою Shopify працює така кількість фахівців та агентств, що знайти когось, хто вирішить

складне унікальне завдання, доопрацює систему та надасть послугу техпідтримки, не важко.

— Велика пропозиція цікавих шаблонів. Сервіс пропонує безліч красивих та актуальних тем для вебсайту, кожна з яких обов'язково оптимізована для максимізації обсягу продажу. Ваше завдання – вибрати дизайн, що найбільше відповідає Вашому проєкту.
— Відсутність необхідності реєструвати хостинг. Магазин розміщується на власному хостингу Shopify. Тобто про технічні моменти та проблеми із серверним обладнанням Ви можете сміливо забути.
— Мобільні рішення для iOS та Android. Управління бізнесом доступне зі смартфона або ноутбука у зручному форматі: обробка замовлень, актуалізація даних про товарні позиції, налаштування сайту.
— Відсутність додаткових платежів за проведення транзакцій (за умови підключення Shopify Payments). Якщо магазин приймає оплати за замовлення через внутрішній платіжний шлюз сервісу, комісія за транзакції не стягується. Є можливість під'єднати інші системи (Stripe, PayPal та ін.).
— Масштабованість. При збільшенні трафіку чи кількості товарних позицій система працює так само безвідмовно, як і за малих обсягів продажу.
— Двотижневий безкоштовний тестовий період. Після його завершення всі налаштування пробної версії залишаються. Щоб продовжити працювати в Shopify, достатньо лише вибрати тариф і внести оплату.

Чи є недоліки у Shopify CMS?

Щоб наш огляд був об'єктивним, ми повинні зупинитися не лише на перевагах, але й на мінусах Shopify CMS[13-14]:

– Ви не є власником веб-сайту. Платформа Shopify побудована за хмарним принципом, що означає, що всі дані зберігаються на хостингу сервісу для доступу до деяких файлів, а не до всіх. Це свого роду оренда сайту.
– За кожну транзакцію додатково до абонентської плати сплачується комісія, яка є відсотком від угоди. Залежно від обраного плану цей відсоток може

коливатися від 0,5% до 2%. Виняток становлять проекти, які використовують Shopify Payments, але в цьому випадку все одно стягується комісія за обробку платежу — внутрішня чи зовнішня комісія (наприклад, для Stripe комісія становить 2,9% + 0,3 $).

- Доповнення до платних функцій. Можливості платформи зазвичай розширюються за рахунок використання платних плагінів і програм, що означає, що плата за підписку буде вищою, ніж рекламується. На щастя, базові та необхідні опції надаються безкоштовно.
- Ви не можете редагувати сторінку оформлення замовлення. Щоб мати можливість змінити дизайн форми замовлення, ви повинні витрачати щонайменше 2000 доларів на місяць, оскільки ця опція доступна лише в плані Shopify Plus[15].
- Обмеження швидкості оптимізації сайту. Оскільки власник магазину не має доступу до параметрів хостингу і деяких допоміжних файлів, йому досить складно вплинути на швидкість завантаження сторінки.
- Висока ймовірність блокування магазину. Будь-який проект Shopify ризикує опинитися в пастці, якщо він порушує вимоги сервісу. Справедливості ради відзначимо, що це трапляється нечасто і зазвичай через неправильні дії, пов'язані з підключенням картки.

Як виглядає адміністративна панель Shopify? Вигляд панелі керування каталогом в Shopify зображено на рисунку 1.1.

Вона використовується також і для підключення вбудованого платіжного шлюзу:

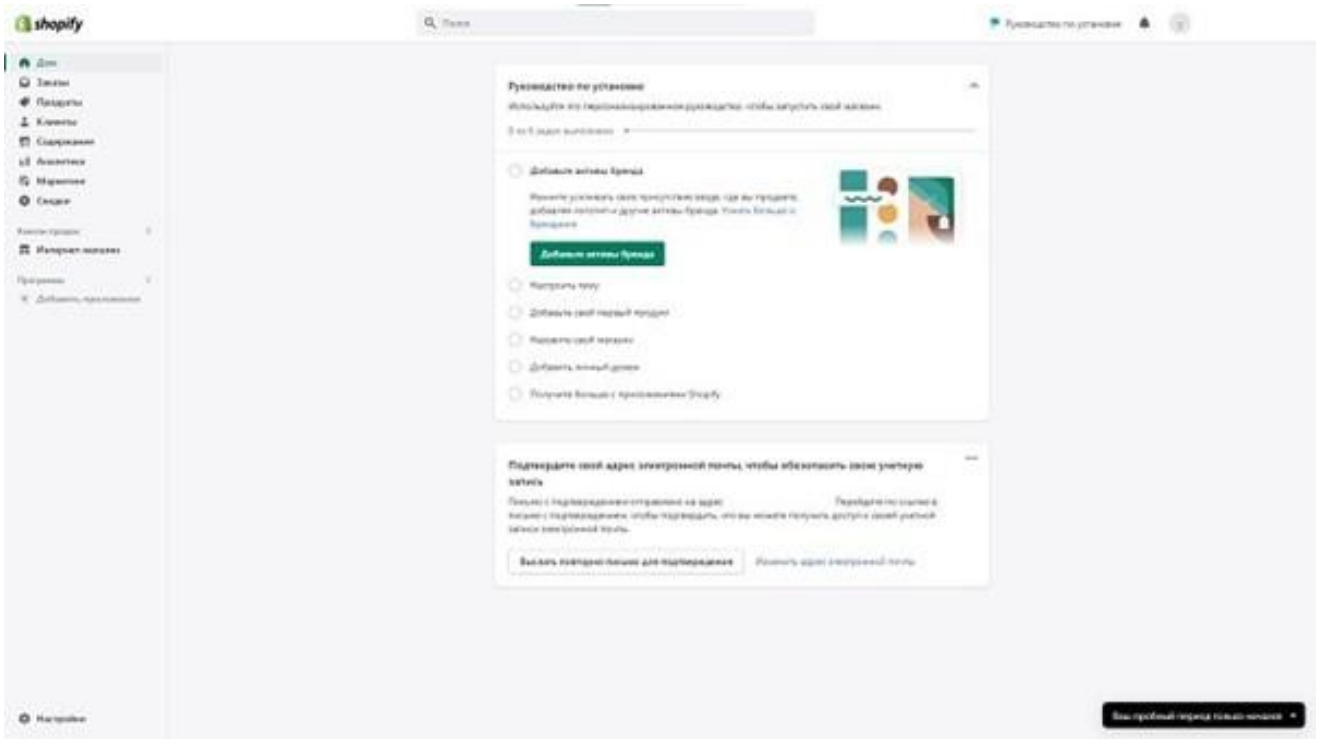


Рисунок 1.1 – Адміністративна панель керування Shopify.

Складові системи керування[17]:

- Home – початкова сторінка, звідки можна перейти в інтерфейс вибору тарифного плану, налаштування товарних позицій, каналів збуту тощо;
- Orders – перелік замовлень, тут же формуються рахунки;
- Products – налаштування категорій та товарних позицій;
- Customers – клієнтська база;
- Analytics – звіти та аналітика;
- Marketing – просування магазину;
- Discount – модуль, у якому можна сформувати купони на знижку;
- Sales channels – налаштування каналів збуту;
- Online store – великий блок з різними параметрами: тема, розміщення постів блогу, редагування сторінок, реєстрація доменного імені, SEO-налаштування тощо;
- Apps – каталог плагінів та додатків;
- Setting – базові налаштування проєкту: платіжний шлюз, обмеження доступів, варіанти доставки та ін.

Завантажувати інформацію про товари можна як окремо для кожної

позиції, так і масивом.

Наступним у нашому списку аналогів буде така CMS-система під назвою – “Magento”,яка являється головним конкурентом Shopify платформи.

Magento[18] — це платформа електронної комерції з відкритим кодом, яка дозволяє компаніям створювати свої онлайн-магазини та керувати ними. Відомий своєю масштабованістю, настроюванням і гнучкістю, він входить до трійки найпопулярніших платформ електронної комерції у світі. Платформа обслуговує магазини будь-якого розміру, від декількох клієнтів до мільйонів, і пропонує три різні версії (включаючи безкоштовну). Це дозволяє підприємствам відкривати свій перший магазин на Magento та розвиватися разом із платформою. Деякі світові бренди, такі як Nike, Coca-Cola та Samsung, створили свої веб-сайти на Magento.

Причина популярності Magento полягає в тому, що практично немає обмежень на те, що ви можете з ним робити. Технологія з відкритим кодом дозволяє контролювати функціональність, зовнішній вигляд і вміст вашого магазину.

Magento CMS[20] має відкритий код, написаний на PHP (з використанням Zend Framework) і використовує MySQL як базу даних. Він також має адаптивний веб-дизайн, тобто магазин на цій платформі зручно переглядати на будь-якому пристрої, включно з мобільним. Це істотно економить ресурси власника інтернет-магазину (як тимчасово, так і матеріально). Magento CMS стала добре відомою своєю адаптивністю та можливістю розширення. Побудований на основі об’єктно-орієнтованого програмування, він пропонує розробникам можливість адаптувати функціональні можливості онлайн-магазину для ефективного задоволення конкретних потреб бізнесу. Крім того, Magento підтримує інтеграцію з широким спектром зовнішніх сервісів і платіжних систем, що підвищує потенціал електронної комерції.

Після встановлення адміністративна панель сайту виглядає так(рис. – 1.2):

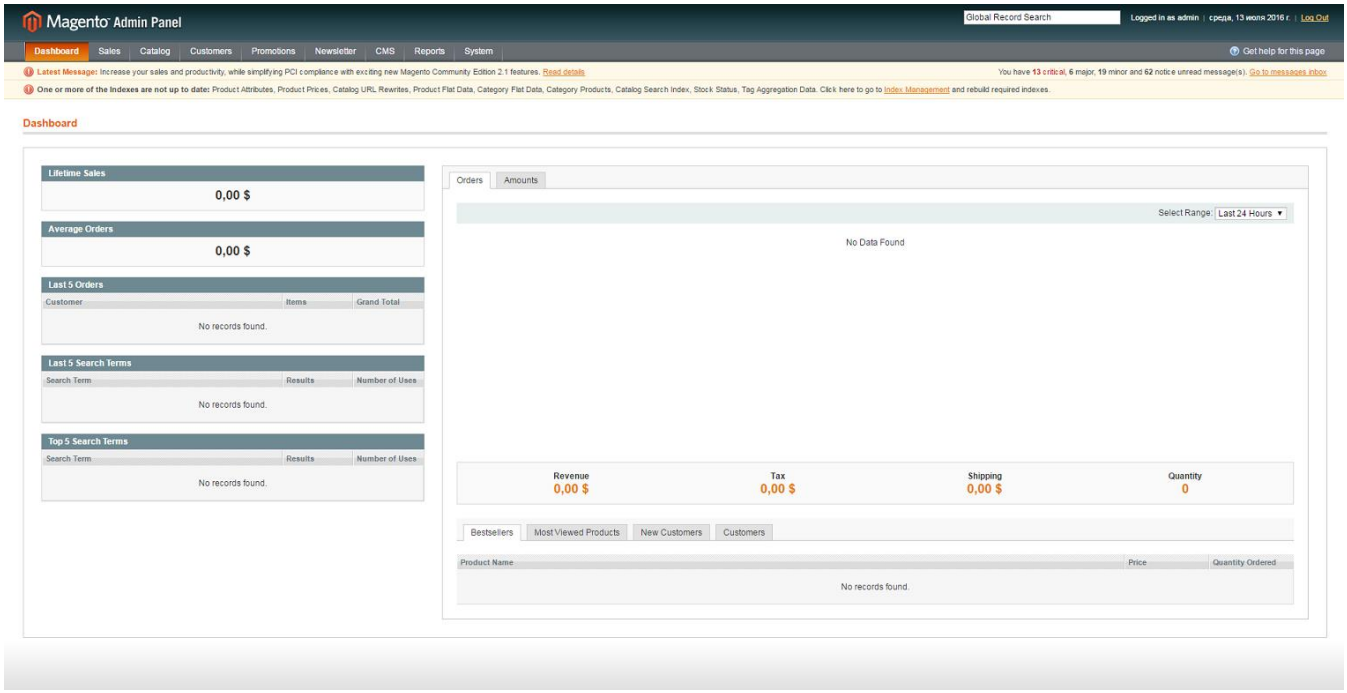


Рисунок 1.2 – Адміністративна панель Magento CMS

Основні переваги:

Magento CMS має потужний набір функцій, включаючи валюту, мови, знижки та купони, звіти та багато іншого. Окрім вбудованих функцій, він пропонує широкий спектр модулів (або розширень) і шаблонів. Це також масштабована та розширювана платформа.

Однією з переваг Magento CMS є можливість створювати кілька інтернет-магазинів і керувати ними з централізованої адміністративної панелі, що є значною перевагою для багатьох власників електронної комерції.

Для пошукової оптимізації можна оптимізувати кожну сторінку продукту, а Magento CMS дозволяє створити XML-карту сайту, яка містить посилання на всі доступні сторінки інтернет-магазину, що необхідно пошуковим системам для коректної індексації ресурсу.

Слід зазначити, що Magento CMS має доступ до коду HTML і PHP, що забезпечує безліч варіантів налаштування.

Крім того, ця платформа підходить для тих, хто хоче уважно стежити за діяльністю своїх продуктів і клієнтів, оскільки дозволяє створювати різні звіти, наприклад, що відбувається з продуктами (їх рух і інвентар), що приваблює

відвідувачів сторінки, яка сторінка є останньою перед їхнім відльотом та багато іншого.Але дана CMS має також декілька вагомих мінусів, які описано нижче:

- Складність встановлення та налаштування: Magento відома своєю складністю встановлення та налаштування. Потрібно мати певні технічні навички для успішної реалізації та управління Magento-сайтом.
- Високі вимоги до ресурсів: Magento вимагає потужного сервера та оптимізованого хостингу для ефективної роботи. Низькі ресурси можуть призвести до повільної роботи сайту, що може впливати на користувацький досвід.
- Споживання багато часу та ресурсів: Розробка і підтримка Magento-сайту може вимагати значних зусиль та ресурсів. Це особливо важко для невеликих компаній або тих, хто не має великих бюджетів на розробку та технічну підтримку.
- Великі витрати на розробку: Хоча Magento безкоштовна для використання, розробка та налаштування специфічних функціональностей може вимагати великих витрат. Розробка додаткових модулів або індивідуальних рішень може бути дорогою і складною задачею.
- Споживач ресурсів сервера: Magento може бути великим споживачем ресурсів сервера, що може призводити до повільної завантаження сторінок, особливо при великому обсязі товарів та великої кількості відвідувачів.
- Складна система оновлень: Оновлення Magento може бути складним завданням, особливо для великих та складних інсталяцій. Це може викликати проблеми сумісності з існуючими модулями та розширеннями.
- Великий обсяг коду: Magento має великий обсяг коду, що може бути проблемою для швидкої і ефективної розробки. Це може ускладнити внесення змін та вдосконалень.

Досить вагомі недоліки для однієї з найкращих систем контролю інтернет-магазинами. За домогою даного аналізу ми зможем в деякій мірі створити додаток що буде в деяких моментах обходити навіть такі потужні CMS.

І на останок хочеться описати в декількох словах ще одну CMS-систему, не таку потужну як попередні, але на її базі створено далеко не один бізнес і вона має назву – OpenCart.

Opencart[21] – це безкоштовний конструктор для e-commerce сайтів. Сервіс широко використовується у всьому світі. Opencart входить у ТОП-10 найбільш використовуваних CMS у США та є четвертою за популярністю e-commerce системою у світі.

Функціональні можливості.

CMS Opencart підійде як для запуску невеликого магазину, так і для маркетплейса. Платформа пропонує достатню кількість вбудованих функцій, котрі дозволяють створити e-commerce проєкт з нуля.

Ці функціональні можливості доступні за замовчуванням[21-23]:

— додавання необмеженої кількості товарів, а також їхніх категорій та підкатегорій;
— рейтинги продуктів, огляди, коментарі;
— вбудована система affiliate marketing – метод просування бізнесу в мережі;
— мультимовність (якщо потрібної мови немає в готовій локалізації, її можна додати самостійно).

Адміністратору магазину сподобаються наступні функції:

— зручна адмінпанель, де подані дані із замовленням, продажами та покупцями;
— можливість керувати кількома магазинами з однієї адмінки;
— для кожного товару можна додати різноманітні параметри (кольори, розмір, інші характеристики);
— адмініструвати сайт можуть кілька людей;
— доступні звіти продажів (переглянуті товари та куплені);
— інтеграція з Google Analytics і CRM;
— backup всього магазину (резервну копію можна відновити в адмінці).

Зручний та зрозумілий інтерфейс сайту важливий для покупців так само як і сам товар. Базових функцій Opencart буде достатньо, щоб клієнти могли:

- встановлювати фільтри на товари за запропонованими категоріями;
- платити в зручній валюті;
- отримувати сповіщення на email (статус замовлення, акційні пропозиції, тощо);
- оформити підписку на товари/послуги;

Як і будь-яка інша система, Opencart має свої недоліки[22]:

- Через велику кількість встановлених плагінів движок може працювати повільно. В результаті магазин функціонує гірше.
- Коли справа доходить до кращого функціоналу, не всі плагіни можуть підійти. Часто доведеться спробувати різноманітні комбінації розширень.
- Движок Opencart не передбачає видалення продубльованих сторінок. Вебмайстер повинен усувати цю проблему власноруч або розвантажувати відповідні плагіни.
- Плагіни проходять оцінку якості перед тим, як потрапити на маркетплейс Opencart, але не всі вони повністю безпечні. Якщо ви додаєте розширення від побічних розробників, є ризик витоку даних.
- Користувачі Opencart відзначають, що вивантаження товарів дуже незручне. Часто товари потрапляють у базу MySQL, через що доводиться завантажувати весь асортимент наново. Також користувачам не подобається те, що каталог з більш ніж 30 000 товарів важко оптимізувати
- Навіть оновлена версія має низку проблем з SEO. Щоб магазин відповідав всім параметрам пошукових систем, доведеться завантажити відповідні плагіни.
- Після оновлення движку деякі модулі можуть не працювати. Зазвичай, через несумісність версій.

Підводячи підсумки в даному розділі ми провели аналіз головних проблем теперішнього бізнесу побудованого на електроній комерції, визначили шляхи його вирішення за допомогою інтеграцій з автоматизованими системами управління організаційними процесами в інтернет магазинах під назвою CMS.

Визначили основні тенденції розробки інтернет магазину, а саме Progressive Web App(PWA) підхід, за допомогою якого буде побудована та розроблена наша CMS-система, та безпосередньо інтернет магазин. Також було проаналізовано декілька теперішніх лідерів на ринку CMS з електроної комерції, короткий аналіз було внесено в таблицю, що зображено нижчє:

Таблиця 1.1

Порівняння найпопулярніших CMS-систем контролю контентом інтернет-магазину

Характеристика	**Shopify**	**Magento**	**OpenCart**
Тип CMS	Хмарна служба / SaaS	Самостійна / Open Source	Самостійна / Open Source
Встановлення та налаштування	Просте і швидке	Складне, вимагає технічних навичок	Спрощене і швидке
Ресурси сервера	Менше вимоги	Високі вимоги	Нижчі вимоги
Витрати на розробку	Доступні розширення та теми	Високі витрати, складне розширення	Доступні розширення та теми
Управління товарами	Просте і інтуїтивне	Потужне, але складне	Зручне, але менш потужне
Оновлення	Автоматичні та безкоштовні	Складні, можуть вимагати ручного втручання	Зазвичай прості та безпечні
Кількість розширень	Обмежений маркетплейс	Широкий вибір з Magento Marketplace	Широкий вибір з OpenCart Marketplace
Гнучкість	Обмежена гнучкість	Велика гнучкість, але складніше налаштування	Задовільна гнучкість
Складність розробки	Обмежена можливість налаштування без коду	Велика складність, вимагає технічних навичок	Зазвичай менше складнощів при розробці

Обговорені вже існуючі рішення, технології та аспекти визначать

напрямки подальших досліджень і розробки, включаючи вдосконалення інтеграції з адміністративним інтерфейсом, оптимізацію аналітичних інструментів та підвищення безпеки та масштабованості системи.

2 ВИБІР СТЕКУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ CMS-СИСТЕМИ ТА ІНТЕРНЕТ МАГАЗИНУ.

Вибір стеку технологій є ключовим етапом в розробці будь-якого програмного продукту, і наш проект не є винятком. У цьому розділі ми детально розглянемо вибір технічного стеку для створення CMS-системи та інтернет-магазину. У цьому розділі ми оглянемо широкий спектр технологій та інструментів, обґрунтуємо вибір конкретних фреймворків, мов програмування, систем управління базами даних та інших інструментів, які допоможуть досягти ефективності та високої продуктивності у реалізації функціоналу.

Важливість правильного вибору технічного стеку полягає в забезпеченні оптимальної швидкості розробки, масштабованості, зручності утримання та підтримки, а також забезпеченні високоякісного користувацького досвіду. Наш вибір базується на ретельному аналізі потреб проекту та врахуванні найкращих практик у галузі розробки веб-застосунків.

2.1 NextJS – React фреймворк

Розглядалися різні фреймворки та бібліотеки для розробки як CMS-системи, так і інтернет-магазину. Для забезпечення ефективної розробки та високої продуктивності був обраний NextJS. Вибір свого роду очевидний, так як в наш час React[24] являється найпопулярнішою бібліотекою для веб-розробки, але це всього лише бібліотека, а от NextJS – це уже повноцінний, потужний інструмент, у якого підтримка зі сторони користувачів просто величезна, дуже докладна документація та ще безліч переваг над іншими фреймворками. Далі ми обговоримо ряд основних функцій що надає нам даний фреймворк.

NextJS — це фреймворк React[25] для створення повноцінних веб-додатків. Ви використовуєте React Components для створення користувацьких інтерфейсів і Next.js для додаткових функцій та оптимізації.

Під капотом NextJS[26] також абстрагує та автоматично налаштовує інструменти, необхідні для React, як-от групування, компіляція тощо. Це дозволяє вам зосередитися на створенні програми замість того, щоб витрачати

час на налаштування.

Незалежно від того, чи ви окремий розробник, чи частина великої команди, NextJS може допомогти вам створювати інтерактивні, динамічні та швидкі програми React.

Деякі з основних переваг розробки з використанням фрейворку NextJS включають:

1. Маршрутизація: Маршрутизатор на основі файлової системи, створений на основі серверних компонентів, який підтримує макети, вкладену маршрутизацію, стани завантаження, обробку помилок тощо.

2. Візуалізація: на стороні клієнта та на стороні сервера з компонентами клієнта та сервера. Додатково оптимізовано за допомогою статичного та динамічного рендерингу на сервері з NextJS. Потокове передавання в середовищах виконання Edge і Node.js.

3. Отримання даних: Спрощене отримання даних за допомогою async/await у компонентах сервера та розширений API отримання для запам’ятовування запитів, кешування даних і повторної перевірки.

4. Підтримка стилів: для ваших бажаних методів стилів, включаючи модулі CSS, Tailwind CSS і CSS-in-JS.

5. Оптимізація: Оптимізація зображень, шрифтів і сценаріїв для покращення основних веб-показників вашої програми та взаємодії з користувачем.

6. TypeScript: Покращена підтримка TypeScript із кращою перевіркою типів і ефективнішою компіляцією, а також спеціальним плагіном TypeScript і перевіркою типів.

Поговоримо більш детально про систему маршрутизації, а саме про – App router, який використовувався врі розробці CMS система для адміну.

App router[26] – це відносно новий спосіб маршрутизації який був представлений у NextJS 13, він побудований на компонентах React Server, який підтримує спільні макети, вкладену маршрутизацію, стани завантаження, обробку помилок тощо.

Маршрутизатор програм працює в новому каталозі під назвою app. Каталог програм працює разом із каталогом сторінок, щоб забезпечити поступове впровадження. Це дозволяє вам вибрати нову поведінку деяких маршрутів вашої програми, зберігаючи інші маршрути в каталозі сторінок для попередньої поведінки.

За замовчуванням компоненти всередині додатку є серверними React компонентами. Це оптимізація продуктивності, яка дозволяє легко їх адаптувати, а також можна використовувати клієнтські компоненти.

Ролі папок і файлів[27]:

Next.js використовує маршрутизатор на основі файлової системи, де:

Папки використовуються для визначення маршрутів. Маршрут — це єдиний шлях із вкладених папок, що слідує за ієрархією файлової системи від кореневої папки до кінцевої кінцевої папки, яка містить файл page.js.

Файли використовуються для створення інтерфейсу користувача, який відображається для сегмента маршруту. Приклад даної системи маршрутів показано на рисунку 2.1.

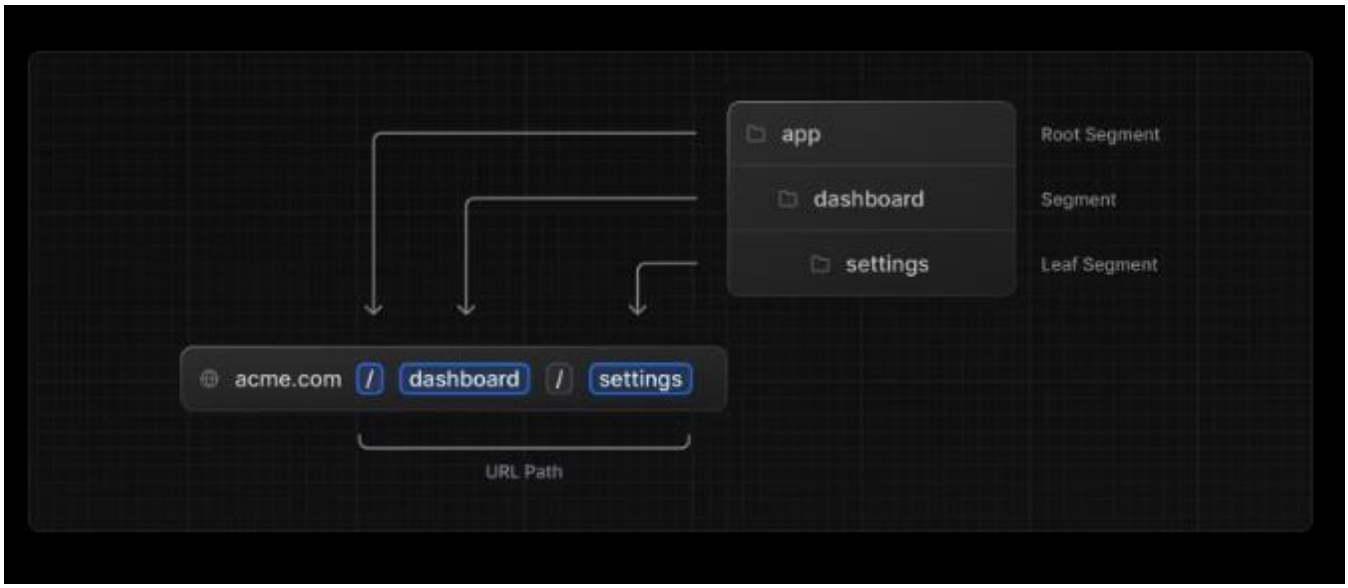


Рисунок 2.1 – Система маршрутизації AppRouter в NextJS

Підсумуємо, кожна папка представляє сегмент маршруту, який відображається на сегмент URL-адреси. Щоб створити вкладений маршрут , ви

можете вкладати папки одна в одну.

Отримання даних є основною частиною будь-якої програми. Існує чотири способи отримання даних в NextJS:

- На сервері, з fetch;
- На сервері, зі сторонніми бібліотеками;
- На клієнті через обробник маршруту;
- На клієнті, зі сторонніми бібліотеками;

Далі вже на прикладі розробленого додатку буде приведено приклад отримання даних декількома способами, що приведені в даному списку.

Що ж таке React Server Component?

React Server Components дозволяють нам змогу створювати користувальницький інтерфейс, який можна візуалізувати та додатково кешувати на сервері. У Next.js роботу рендерингу додатково розділено на сегменти маршруту, щоб увімкнути потокове та часткове рендеринг, і є три різні стратегії рендерингу[28] сервера:

- Статичний рендеринг (Static Rendering),
- Динамічний рендеринг(Dynamic Rendering),
- Потокове передавання (Streaming).

За допомогою статичного рендерингу маршрути рендеряться під час створення або у фоновому режимі після повторної перевірки даних. Результат зберігається в кеш-пам’яті та може бути переданий у мережу доставки вмісту (CDN). Ця оптимізація дозволяє ділитися результатом роботи з рендерингом між користувачами та запитами сервера.

Статичний рендеринг корисний, коли маршрут містить дані, які не персоналізовані для користувача та можуть бути відомі під час створення, наприклад статична публікація в блозі або сторінка продукту.

За допомогою динамічного рендерингу маршрути відображаються для кожного користувача під час запиту.

Динамічний рендеринг[28] корисний, коли маршрут містить дані, персоналізовані для користувача, або інформацію, яка може бути відома лише

під час запиту, наприклад файли cookie або параметри пошуку URL-адреси.

Потокове передавання дає змогу поступово відтворювати інтерфейс користувача з сервера. Робота розбивається на частини та передається клієнту, коли вона стає готовою. Це дозволяє користувачеві побачити частини сторінки відразу, до того, як весь вміст завершить рендеринг.

Потокове передавання вбудовано в Next.js App Router за замовчуванням. Це допомагає покращити продуктивність початкового завантаження сторінки, а також користувальницький інтерфейс, який залежить від повільнішої вибірки даних, що блокує відтворення всього маршруту. Наприклад, відгуки на сторінці товару.

Ви можете розпочати трансляцію сегментів маршруту за допомогою loading.js і компонентів інтерфейсу користувача з React Suspense[29].

Переваги серверного рендерингу:

1. Отримання даних: серверні компоненти дозволяють перемістити вибірку даних на сервер ближче до джерела даних. Це може підвищити продуктивність за рахунок скорочення часу, необхідного для отримання даних, необхідних для візуалізації, і кількості запитів, які клієнт повинен зробити.

2. Безпека. Серверні компоненти дозволяють зберігати на сервері конфіденційні дані та логіку, наприклад маркери та ключі API, без ризику розкриття їх клієнту.

3. Кешування: шляхом відтворення на сервері результат можна кешувати та повторно використовувати для наступних запитів і між користувачами. Це може підвищити продуктивність і зменшити витрати за рахунок зменшення обсягу візуалізації та отримання даних, що виконується для кожного запиту.

4. Розміри пакетів: серверні компоненти дозволяють зберігати великі залежності, які раніше впливали на розмір клієнтського пакета JavaScript на сервері. Це корисно для користувачів із повільнішим Інтернетом або менш потужними пристроями, оскільки клієнту не потрібно завантажувати, аналізувати та виконувати будь-який JavaScript для серверних компонентів.

5. Початкове завантаження сторінки та перше малювання вмісту (FCP): на

сервері ми можемо генерувати HTML, щоб дозволити користувачам переглядати сторінку негайно, не чекаючи, поки клієнт завантажить, розбере та виконає JavaScript, необхідний для відтворення сторінки.

6. Оптимізація пошукової системи та можливість спільного використання в соціальних мережах: візуалізований HTML може використовуватися роботами пошукової системи для індексування ваших сторінок і ботами соціальних мереж для створення попереднього перегляду карток соціальних мереж для ваших сторінок.

7. Потокове передавання: Серверні компоненти дозволяють вам розділити роботу візуалізації на частини та передавати їх клієнту, коли вони будуть готові. Це дозволяє користувачеві побачити частини сторінки раніше, не чекаючи, поки вся сторінка буде відображена на сервері.

Це тільки вершина айсбергу можливостей даного фреймворку, але навіть цеї інформації достатньо аби зрозуміти чому для розробки додатку було обрано саме його. На останок, порівняємо NextJS з іншими популярними фреймворками такими як: Angular та VueJS, порівняння цих фреймворків з обраним нами NextJS ви можете розглянути у таблиці – 2.1.

Таблиця 2.1.

Порівняння найпоппулярніших фреймворків для створення прогресивних веб-додаків сьогодення.

Характеристика	**Next.js**	**Angular**	**Vue.js**
Мова програмування	JavaScript, TypeScript	TypeScript	JavaScript
Роутінг	Вбудований (за допомогою файлів у папці pages)	Вбудований (Angular Router)	Вбудований (Vue Router)

Управління станом	Вбудоване в React (контекст, Redux)	RxJS, Angular services	Vuex (за замовчуванням), можливість використовувати інші бібліотеки для управління станом
Технології	Server-side rendering (SSR), Static Site Generation (SSG)	Dependency Injection, Angular CLI	Vue CLI, Vuex, Vue Router
Встраювання інших бібліотек	Зручна інтеграція з багатьма бібліотеками	Зручна інтеграція з Angular Material, RxJS, інші бібліотеки	Зручна інтеграція з Vuetify, Element, інші бібліотеки
Компонентна архітектура	React components	Angular components	Vue components
Реактивність	Немає вбудованої підтримки реактивності, але можливе використання RxJS	Вбудована підтримка реактивності за допомогою RxJS	Вбудована підтримка реактивності (директива v-model, реактивні дані)
Інтеграція зі сторонніми сервісами	Зручна інтеграція з API, можливість використовувати будь-які бібліотеки	Зручна інтеграція з HTTP сервісами, можливість використовувати Angular HTTP Client	Зручна інтеграція з API, можливість використовувати Axios, інші бібліотеки

Розмір бандлу	Залежить від проекту, можливість оптимізації за допомогою SSG та tree-shaking	Зазвичай більший порівняно з React та Vue	Зазвичай менший порівняно з Angular та React
Ком'юніті та підтримка	Активна ком'юніті, багато плагінів та розширень	Велика активна ком'юніті, офіційна підтримка від Google	Активна ком'юніті, проста документація, менше "офіційних" плагінів, але багато сторонніх

2.2 TypeScript – розширення JavaScript

TypeScript[30] — це безкоштовна високорівнева мова програмування з відкритим вихідним кодом, розроблена корпорацією Майкрософт, яка додає до JavaScript статичний тип із необов’язковими анотаціями типу. Оскільки TypeScript є надмножиною JavaScript, усі програми JavaScript є синтаксично дійсними TypeScript, але вони можуть не перевіряти тип з міркувань безпеки.

TypeScript можна використовувати для розробки програм JavaScript як для виконання на стороні клієнта, так і на стороні сервера (як у випадку з Node.js або Deno). Для транспіляції доступні кілька варіантів. Можна використовувати типовий компілятор TypeScript або можна викликати компілятор Babel[32] для перетворення TypeScript у JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про тип існуючих бібліотек JavaScript, подібно до того, як файли заголовків C++ можуть описувати структуру існуючих об’єктних файлів. Це дозволяє іншим програмам використовувати значення, визначені у файлах, як якщо б вони були статично типізованими сутностями TypeScript. Існують сторонні файли

заголовків для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Також доступні заголовки TypeScript для модулів бібліотеки Node.js, що дозволяє розробляти програми Node.js у TypeScript.

Сам компілятор[33] TypeScript написаний на TypeScript і скомпільований у JavaScript. Він ліцензований за ліцензією Apache 2.0. Андерс Хейлсберг, провідний архітектор C# і творець Delphi і Turbo Pascal, працював над розробкою TypeScript.

Сумісність з JavaScript[34]

TypeScript — це сувора надмножина ECMAScript 2015, яка сама є надмножиною ECMAScript 5, яку зазвичай називають JavaScript.[30] Таким чином, програма JavaScript також є дійсною програмою TypeScript, і програма TypeScript може безперешкодно використовувати JavaScript. За замовчуванням компілятор націлений на ECMAScript 5, поточний переважаючий стандарт, але також може генерувати конструкції, які використовуються в ECMAScript 3 або 2015.

За допомогою TypeScript[35-36] можна використовувати існуючий код JavaScript, включати популярні бібліотеки JavaScript і викликати згенерований TypeScript код з іншого JavaScript. Оголошення типів для цих бібліотек надаються разом із вихідним кодом.

TypeScript забезпечує статичний тип за допомогою анотацій типу, щоб увімкнути перевірку типу під час компіляції. Це необов’язково, і його можна ігнорувати, щоб використовувати звичайний динамічний тип JavaScript.

2.3 ORM Prisma з конектором бази даних MySQL та платформа керування базою даних PlanetScale

Що таке Prisma?

Prisma[37] — це ORM (Object-Relational Mapping) нового покоління з відкритим кодом. Він складається з наступних частин:

– Client Prisma: автоматично згенерований і безпечний конструктор запитів для Node.js і TypeScript

– Prisma Migrate: система міграції

– Prisma Studio: графічний інтерфейс для перегляду та редагування даних у вашій базі даних.

Prisma Studio — це єдина частина Prisma ORM, яка не є відкритим кодом. Ви можете запускати Prisma Studio лише локально. Prisma Studio також інтегрована в нашу комерційну пропозицію Prisma Data Platform під назвою Data Browser. У браузері даних ви можете переглядати та редагувати дані для кожного проекту, а інші члени команди можуть робити те саме після того, як ви надасте їм відповідні дозволи.

Client Prisma можна використовувати в будь-якому серверному додатку Node.js (підтримувані версії) або TypeScript (включаючи безсерверні додатки та мікросервіси). Це може бути REST API, GraphQL API, gRPC API або щось інше, для чого потрібна база даних.

Як працює Prisma?

Кожен проект, який використовує інструмент із набору інструментів Prisma, починається з файлу схеми Prisma. Схема Prisma[38] дозволяє розробникам визначати свої моделі додатків на інтуїтивно зрозумілій мові моделювання даних. Він також містить підключення до бази даних і визначає генератор, приклад представлено у фрагменті коду нижче:

```
// Джерело даних
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

// Генератор
generator client {
  provider = "prisma-client-js"
}

// Модель даних
model Post {
  id        Int     @id @default(autoincrement())
  title     String
  content   String?
  published Boolean @default(false)
  author    User?   @relation(fields: [authorId], references: [id])
  authorId  Int?
```

```
}

// Модель даних
model User {
  id    Int     @id @default(autoincrement())
  email String  @unique
  name  String?
  posts Post[]
}
```

У цій схемі ми налаштовуємо три речі:

1. Джерело даних: визначає підключення до бази даних (через змінну середовища);
2. Генератор: вказує на те, що ви хочете створити клієнт Prisma;
3. Модель даних: визначає моделі ваших програм.

Для розробки системи контролю контентом я обрав базу даних MySQL[39], тобто налаштування моє схеми буде виглядати наступним чином:

```
// Джерело даних
datasource db {
  provider     = "mysql"
  url          = env("DATABASE_URL")
  relationMode = "prisma"


// Генератор
generator client {
  provider = "prisma-client-js"
}

// Далі я буду створювати моделі в яких будуть зберігатись необхідні дані.
```

Чому було обрано саме Prisma?

Головна мета Prisma — зробити розробників додатків більш продуктивними при роботі з базами даних. Ось кілька прикладів того, як Prisma досягає цього:

Використання Prisma, порівняно з чистою MySQL базою даних, має численні переваги, які дозволяють поліпшити розробку та обслуговування додатків. Ось кілька ключових аргументів на користь використання Prisma:

Зручність в роботі з базою даних: Prisma надає абстракцію бази даних, що полегшує взаємодію з нею. Розробники можуть працювати з моделями та

запитами на мові програмування, замість написання чистого SQL коду. Це знижує кількість помилок та полегшує розробку.

Безпека: Prisma дозволяє використовувати параметризовані запити, що захищає від SQL-ін'єкцій та інших безпекових загроз. У чистому MySQL вам доведеться самостійно дбати про це.

Швидкість розробки: Prisma автоматично генерує CRUD-операції для ваших моделей, що значно прискорює розробку. Ви можете швидко створювати, читати, оновлювати та видаляти записи з бази даних без необхідності написання багато коду вручну.

Міграції: Prisma надає засоби для керування міграціями бази даних, що дозволяє легко зберігати та розвивати схему вашої бази даних з плином часу.

Сумісність з різними базами даних: Prisma підтримує різні бази даних, такі як MySQL, PostgreSQL, SQLite та інші. Це дозволяє легко змінювати базу даних, якщо це буде необхідно в майбутньому.

Спільнота та підтримка: Prisma має активну спільноту та регулярні оновлення, що гарантує стабільність та підтримку вашого проекту.

У підсумку, використання Prisma зазвичай робить процес розробки та обслуговування бази даних більш ефективним і безпечним порівняно з чистою MySQL. Ця технологія дозволяє зосередитися на функціональності додатку та робить роботу з базою даних більш інтуїтивною та продуктивною.

Платформа керування базами даних PlanetScale:

PlanetScale[40] – це платформа для керування базами даних, яка надає розподілену систему керування базами даних (DBMS) на основі відкритого програмного забезпечення MySQL. Призначення PlanetScale полягає в полегшенні створення, масштабуванні та керуванні базами даних для сучасних додатків та сервісів. Давайте розглянемо більше деталей про PlanetScale, його використання, переваги та недоліки:

Що таке PlanetScale?

PlanetScale - це розподілена система керування базами даних (DBMS), яка базується на MySQL. Вона пропонує послуги для розміщення, масштабування та

управління базами даних в розподіленому середовищі. PlanetScale розроблено командою, яка раніше працювала над інфраструктурою баз даних у таких компаніях, як YouTube та Dropbox.

Для чого використовують PlanetScale?

Масштабування: PlanetScaleдозволяє легко масштабувати бази даних відповідно до зростання навантаження.

Висока доступність: Вона надає засоби для забезпечення високої доступності та відновлення в разі виникнення проблем.

Розподілені операції: Допомагає в розподілених командах працювати з базами даних та забезпечує синхронізацію даних.

Переваги PlanetScale:

Спрощена розробка: Розробники можуть концентруватися на розробці, не турбуючись про складності масштабування і управління базами даних.

Висока доступність: Платформа надійно забезпечує доступ до даних навіть у випадку відмови окремих вузлів.

Швидка реплікація: PlanetScale забезпечує швидку реплікацію даних між різними регіонами та серверами.

Недоліки PlanetScale:

- Вартість: Використання PlanetScale може бути витратним, особливо для малих команд або проєктів з обмеженим бюджетом.
- Складність налаштування: Для налаштування та оптимізації системи може знадобитися досвід у роботі з розподіленими базами даних.

Усупереч недолікам, PlanetScale є потужним інструментом для організацій та розробників, які мають вимоги до масштабування та високої доступності своїх баз даних.

Вона допомагає забезпечити стабільну та надійну роботу додатків, які вимагають швидкого доступу до даних та їх реплікації в різних частинах світу і повністю задовільняє потреби проекту магістерської роботи, тому дана платформа і ввійшла до стеку використаних технологій.

2.4 Tailwind – CSS фреймворк

CSS фреймворки стали невід'ємною частиною сучасної розробки веб-інтерфейсів. Нижче ми докладно розглянемо один з найбільш популярних і потужних інструментів цього – Tailwind CSS. Давайте розпочнемо з основ та з'ясуємо, чому цей фреймворк став вибором багатьох веб-розробників в тому числі й був обраний для розробки CMS-системи контролю контентом інтернет магазину.

Tailwind CSS[41] - це CSS фреймворк, розроблений з метою спростити процес створення інтерфейсів для веб-сайтів і додатків. Однією з основних ідей цього фреймворку є набір готових класів, які дозволяють вам застосовувати стилі до HTML елементів без необхідності написання власного CSS коду.

Основою Tailwind CSS[42] є великий набір CSS класів, кожен з яких відповідає певному стилю або властивості. Наприклад, клас "bg-blue-500" застосовує синій фон до елемента, а "text-white" робить текст білим кольором. Важливо розуміти, як ці класи працюють і як їх можна комбінувати для створення бажаних стилів.

Основні концепції Tailwind CSS:

У Tailwind CSS основний акцент робиться на використанні CSS класів для визначення стилів. Кожен клас відповідає певному стилі, і вони можуть бути застосовані до HTML елементів. Наприклад, клас "bg-blue-500" встановлює синій фон для елементу. Це дозволяє швидко та зручно застосовувати стилі до елементів і зменшити кількість власного CSS коду.

Модифікації класів.

Один з важливих аспектів Tailwind CSS - це можливість модифікувати класи для визначення більш конкретних стилів. Наприклад, ви можете використовувати "hover:bg-blue-700" для встановлення фонового кольору при наведенні на елемент. Це дозволяє створювати інтерактивні ефекти без написання JavaScript[43] коду.

Tailwind CSS надає можливість створювати адаптивний і респонсивний дизайн за допомогою Utility-передач[43]. Ви можете визначити, як стилі

змінюються на різних розмірах екрану і пристроях, використовуючи такі класи як "sm:", "md:", і "lg:". Це дозволяє забезпечити оптимальний дизайн для різних пристроїв та екранних розмірів.

Ці концепції є основними для розуміння та використання Tailwind CSS у вашому проекті.

Чому ж було обрано даний CSS фреймворк для розробки CMS системи, та безпосередньо інтернет магазину?

1. Швидкість розробки – однією з найбільших переваг Tailwind CSS є швидкість розробки. Завдяки великому набору готових CSS класів, розробники можуть швидко застосовувати стилі до елементів без необхідності великої кількості власного CSS коду. Це спрощує і прискорює процес розробки, зменшуючи час, необхідний для створення інтерфейсу.
2. Масштабованість і обслуговуваність[44-45] – Tailwind CSS сприяє покращенню масштабованості та обслуговуваності проекту. Завдяки чіткій структурі класів та відсутності потреби в написанні власного CSS, код проекту залишається організованим і легко змінюваним. Розробники можуть легко додавати нові стилі або змінювати існуючі без ризику виникнення конфліктів.
3. Спільнота та ресурси – Tailwind CSS має активну та розрослу спільноту розробників. Це означає, що ви можете знайти безліч ресурсів, які допоможуть вам вивчити фреймворк та вирішити проблеми. Існують форуми, документація та навчальні матеріали, які спростять вам роботу з Tailwind CSS.

Ці переваги роблять Tailwind CSS потужним інструментом для створення сучасних веб-інтерфейсів. Також слід додати, що є багато бібліотек і форумів з вже готовими компонентами, які ще більше прискорюють роботу розробника.

В даному проекті також буде використовуватись збірник готових компонентів під назвою Shadcn-UI. Не будемо деталь оглядати даний інструмент, фале декілька слів можна додати – компоненти що представленні в даній бібліотеці повністю підтримують Tailwind та TypeScript, що повністю

задовільняє наші потреби та значно скорочує час розробки дозволяючи нам більше зосередитись на архітектурних особливостях та розробці безпосередньо бізнес логіки в даному додатку.

2.5 Інші інструменти

2.5.1 ClerkJS –.бібліотека керування автентифікацією

JavaScript бібліотека під назвою ClerkJS[47] є нашим основним інструментом для розробки системи управління користувачами та автентифікації. Вона дозволяє нам ефективно реєструвати, входити в систему, перевіряти та керувати користувачами вашого інтернет-магазину через налаштовані потоки.

Зазвичай дану бібліотеку рекомендують використовувати з один із фреймворків для зв'язку (наприклад React або Next.js),що в даному кейсі повністю задовільняє наші потреби.

Використання базового пакету ClerkJS надає більше гнучкості у розгортанні Clerk так, як це вам найбільше підходить у контексті розробки автоматизованої системи керування організаційними процесами нашого інтернет-магазину в рамках магістерської кваліфікаційної роботи.

2.5.2 Stripe – система обробки платежів.

В сучасному світі, де електронна комерція стає все більш і більш важливою галуззю бізнесу, вибір надійної та ефективної системи обробки платежів має ключове значення для успіху інтернет-магазину. У контексті CMS системи контролю організаційних процесів в інтернет-магазині, однією з найперспективніших інтеграцій є використання платіжної системи Stripe.

Stripe[48] - американська технологічна компанія, що надає платіжні послуги. Заснована компанія була у 2010 році братами з Ірландії Джоном та Патріком Коллінс. Тоді код Stripe складався всього із 7 рядків, а сьогодні сервіс став найдорожчим приватним стартапом вартістю 95 мільярдів доларів.

Про популярність Stripe серед підприємців, а відповідно і серед

користувачів говорять числа. На день сервіс обробляє понад 250 мільйонів запитів – близько 13 тисяч запитів на секунду. 90% дорослих американців купують в інтернеті через Stripe. У 2021 році компанія опрацювала платежі на загальну суму понад 640 мільярдів доларів. Сервіс підтримує понад 135 валют та платіжних методів.

Які проблеми українського бізнесу може вирішити Stripe[49-50]?

Наразі міністр цифрової трансформації з розвитку ІТ Олександр Борняков заявляє, що Міністерству вдалося достукатися до керівництва Stripe. Переговори з компанією тривають близько двох років, але ні пандемія, ні війна в Україні не прискорили їх.

Україна є повноцінним учасником світової економічної системи. Підприємці продають товари на міжнародному ринку, фахівці надають послуги іноземним фірмам, фрілансери працюють і на внутрішньому, і на зовнішньому ринку. Український ІТ-ринок називають найкращим у світі, а значить, і до айтішників з України звертаються найчастіше. Основна проблема українського бізнесу – отримання міжнародних платежів.

Основна послуга Stripe[51] – забезпечення транзакцій. Сервіс допомагає отримувати оплату з сайтів безпосередньо на рахунок продавця. Крім цього, зі Stripe можна створити онлайн-касу для прийому платежів у фізичних магазинах, компанія надає послуги з кредитування та випуску карток, займається захистом від шахрайства.

Отже, Stripe — сервіс, який міг би допомогти українському бізнесу та дозволив би йому отримувати міжнародні платежі набагато простіше. У свою чергу, держава настільки зацікавлена у парафії Stripe, що переговори зараз ведуть не лише сама компанія та Міністерство цифрової трансформації, а й Національний банк України. Приєдналася навіть Visa.

Підведемо підсумки, даний розділ, присвячений вибору технологій для розробки CMS-системи та інтернет-магазину, є стратегічно важливим етапом у процесі створення нашого проекту. На основі ретельного аналізу та обговорення ми здійснили вибір технічного стеку, що найкращим чином відповідає потребам

та вимогам нашого проекту.

Вибір фреймворку. Обраний NextJS визначається сумісністю, високою продуктивністю та активною підтримкою розробників. Цей інструмент надасть нам потужність та гнучкість для швидкої та ефективної розробки функціоналу.

Мова програмування. Вибір мови програмування TypeScript обґрунтовується її ефективністю, сумісністю з обраним фреймворком, можливістю типізації даних, шо дуже прискорює розробку дозволяючи допускати менше помилок, відповідно менше дебажити.

Система управління базою даних. Обрана ORM Prisma відповідає вимогам щодо швидкодії та надійності у зберіганні та обробці великого обсягу даних.

Інструменти для розробки інтерфейсу. Використання Tailwind CSS разом з збіркою готових компонентів ShadcnUI – забезпечить створення інтуїтивно зрозумілого та привабливого користувацького інтерфейсу.

Врахування масштабів та ефективності. Усі вибори технологій ретельно обрані з урахуванням масштабів нашого проекту та прагнення досягти високої ефективності в умовах зростаючого обсягу даних та користувачів.

Загальний висновок полягає в тому, що обраний технічний стек відповідає вимогам та цілям нашого проекту. Він забезпечить швидку розробку, зручне утримання та високу продуктивність системи, сприяючи успішному втіленню наших планів та досягненню бажаних результатів.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Середовище розробки та система контрою версій

Перед початком розробки будь якого проекту необхідно обрати середовище розробки (IDE) в якому безпосередньо буде проводитись майже весь час роботи. Для покриття конкретних цілей для роботи над проектом було обрано Visual Studio Code (рис. – 3.1).

Visual Studio Code[52] (VS Code) – це безкоштовний, легкий та потужний текстовий редактор від Microsoft, призначений для розробки програмного забезпечення. Він є одним з найпопулярніших інтегрованих середовищ розробки (IDE) серед програмістів та розробників. Ось кілька причин, чому VS Code вважається таким популярним та чому його рекомендується використовувати:

1. Безкоштовний та Відкритий Код: VS Code доступний для завантаження безкоштовно, і він є відкритим програмним забезпеченням, що означає, що його вихідний код доступний для громадського перегляду та внесення внесків.
2. Легкий та Швидкий: VS Code має мінімальний обсяг вбудованих функцій, що робить його дуже швидким та легким для використання. Він не навантажує вашу систему надто великою кількістю ресурсів.
3. Розширення та Екосистема: VS Code має не аби який вибір розширень, які ви можете встановлювати, щоб розширити його функціональність. Це дозволяє вам адаптувати середовище розробки під свої потреби, розширення є невід’мною частиною, що значно полегшує роботу розробника. Є підтримка для багатьох мов програмування та технологій.
4. Інтеграція з Git: VS Code має вбудовану підтримку для систем контролю версій Git.
5. Debugging та Profiling: Вбудовані інструменти для налагодження (debugging) та профілювання коду полегшують виявлення та усунення помилок у вашому програмному коді.
6. Розширені функції редактора: VSC має потужні функції редактора, такі як

підсвічування синтаксису, автодоповнення коду, рефакторинг та багато іншого, що полегшує написання та редагування коду.

7. Активна Спільнота та Оновлення: VS Code користується великою та активною спільнотою розробників, що означає регулярні оновлення та надання корисних ресурсів.
8. Розробка Web-додатків: VS Code особливо ефективний для розробки веб-додатків, включаючи підтримку для HTML, CSS, TypeScript, і інших технологій веб-розробки.

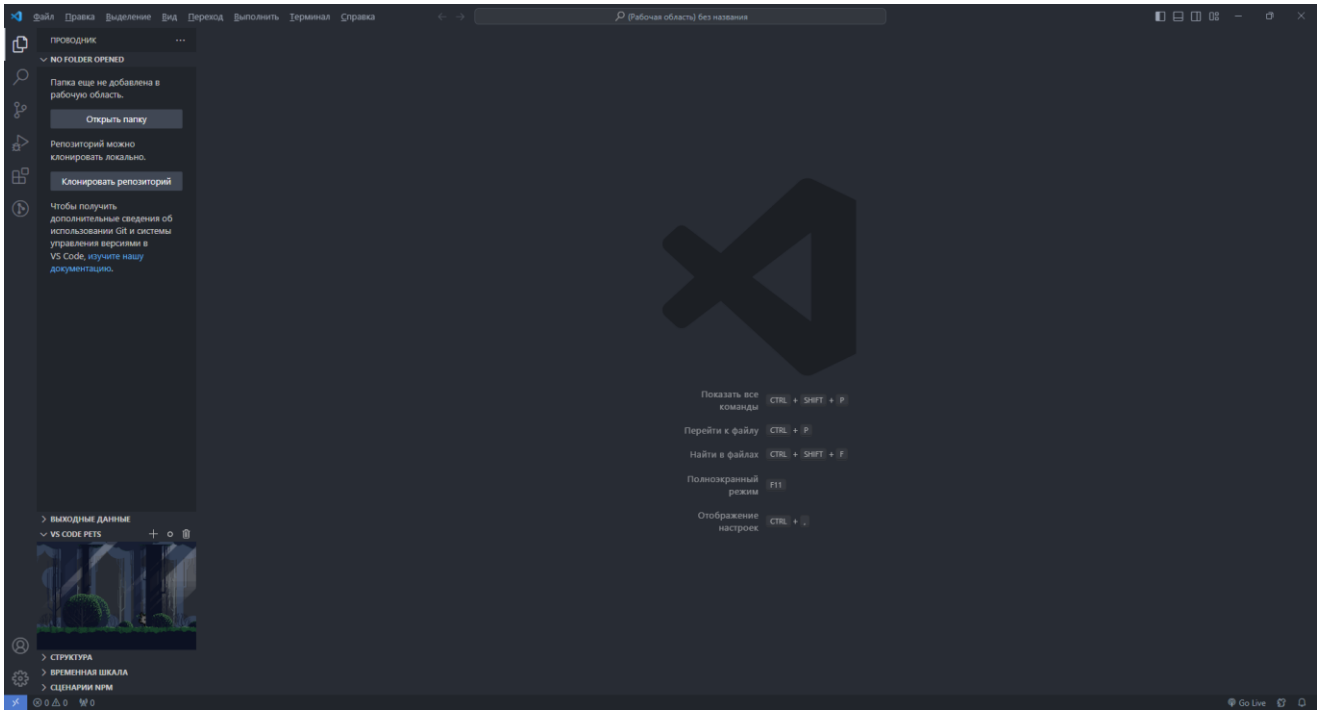


Рисунок 3.1. – Средовище розробки VS Code

Загалом, Visual Studio Code надає зручне та продуктивне середовище для розробки програмного забезпечення, а його легкість використання та розширені можливості роблять його популярним серед розробників усього світу.

Система контролю версій (СКВ) є необхідною для ефективного управління версіями програмного коду.

GIT[53] є однією з найпопулярніших систем контролю версій, і вона дозволяє розробникам спільно працювати над проектом, відстежувати зміни, впроваджувати новий функціонал та виправляти помилки.

Основні переваги використання GIT:

1. Відстеження змін:

За допомогою GIT можна відстежувати всі зміни у коді, включаючи додавання, видалення та зміни файлів. Це дозволяє легко визначити, хто і коли вніс певні зміни.

2. Відгалуження та злиття (Branching and Merging):

GIT дозволяє створювати відгалуження (branches), що дозволяє розробникам працювати паралельно над різними функціями чи виправленнями. Після завершення роботи відгалуження може бути злито (merged) з основною гілкою.

3. Колективна робота:

GIT полегшує колективну роботу над проектом. Кожен учасник може вносити свої зміни, і GIT допомагає узгоджувати ці зміни, зменшуючи конфлікти та непорозуміння.

4. Історія проекту:

GIT зберігає повну історію змін проекту. Це дозволяє повертатись за необхідності до попередніх версій коду, а також аналізувати розвиток проекту з часом та маштабувати код безбечно.

GitHub[54] (рис. – 3.2), з іншого боку, є веб-платформою для спільної роботи над проектами, що використовують GIT. Деякі ключові особливості GitHub:

1. Віддалений доступ:

GitHub надає віддалений доступ до вашого репозиторію, що полегшує спільну роботу команди, яка може бути розподілена по різних місцях.

2. Зручний інтерфейс:

GitHub має інтуїтивно зрозумілий інтерфейс для перегляду коду, внесення змін та ведення дискусій.

3. Проблеми та проекти:

В GitHub можна вести списки завдань, відстежувати проблеми та стежити за проектами, що полегшує організацію роботи при розробці будь якого проекту,

що робить його використання обов’зковим.

4. Інтеграція з іншими сервісами:

GitHub легко інтегрується з іншими сервісами, такими як CI/CD системи, що автоматизує процеси розробки.

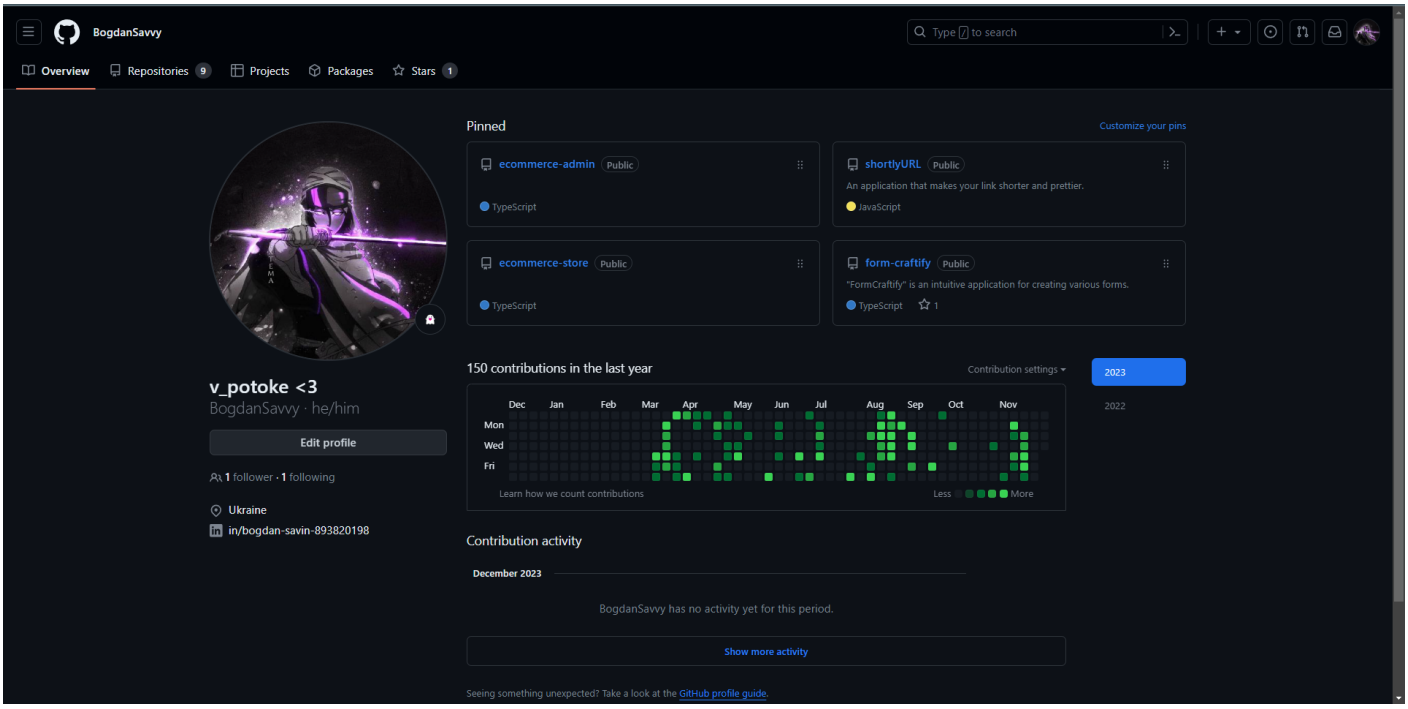


Рисунок 3.2 – Вигляд GitHub профілю

Загалом, використання GIT та платформи, такої як GitHub, сприяє ефективній розробці програмного забезпечення, зменшує ризик конфліктів та допомагає вести та підтримувати проект в організований та структурований спосіб, тому це робить їх необхідністю в будь якому проекті, в тому числі у нашому.

3.2 Ініціалізаця проекту, впровадження авторизації користувача

Початком роботи над проекту на базі будь якого фреймворку є виокристання шаблону для встановлення всіх необхідних інструментів та модулів в середовище розробки. В нашому кейсі ми будемо використовувати стартовий шаблон що надає нам лфіційна документація фреймворку NextJS.

Щоб використовувати даний шаблон ми будемо використовувати powershell консоль що надає на операційна система Windows 11, також

попередньо у нас повиний бути встановлений Node.js, бажано останньої версії. Відкривши powershell нам потрібно перейти в небхідну нам папку використовуючи команду:

– `cd C:\Шлях\До\Вашого\Каталогу`

Перейшовши до небхідного катологу запиуємо наступну команду:

– `npx create-next-app [назва проекту]` – дана команда встановить все необхідне для роботи в вказаний нами каталог.

Наступним кроком ми відкриваємо середовище VS Code та переходимо до катологу з нашим стартовим шаблоном проекту. Структура файлів стартового шаблону NextJS зображено на рисунку 3.3.

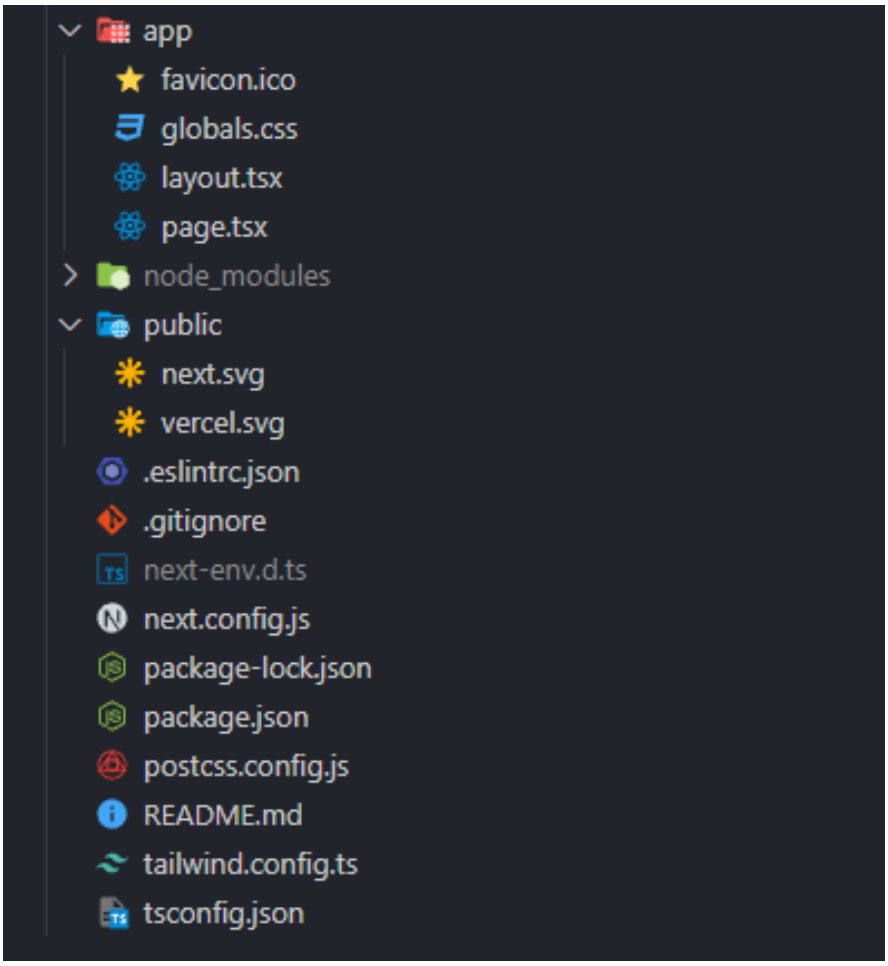


Рисунок 3.3 – стартовий шаблон фреймворку NextJS

Першим кроком розробки CMS-системи контролю організаційними процесами роботи інтернет-магазину бдуе впровадження авторизації, оскільки

доступ до адміністративної панелі буде мати лише авторизований користувач.

Авторизацію в даному додатку буде розроблено за допомогою бібліотеки ClerkJS. Щоб використовувати функціонал даного інструмента просто реєструєм власний кабінет на офіційному сайті Clerk. Створюємо новий проект, після чого отрумуємо всі необхідні API-ключі, які копіюємо та вставляємо в файл “.env”, який створюємо в корені проекту.

“.env” файл (Environment файл) – це текстовий файл, який зазвичай використовується для збереження конфігураційних параметрів для програмного забезпечення. Зазвичай він містить пари "ключ=значення", де ключ - це ім'я параметра, а значення - його відповідне значення.

Цей файл може містити конфігураційні змінні, такі як налаштування бази даних, секретні ключі API, URL-адреси, інші константи та параметри, які використовуються програмою чи середовищем розробки, використовують .env файли, щоб уникнути включення конфіденційної інформації безпосередньо в код програми і легко змінювати конфігурацію, не змінюючи сам код.

Далі слідуючи вказівкам по впровадженю авторизації з офіційної документації Clerk, створюємо так званий проміжний шар. В корені проекту створюємо ще один файл під назвою “middleware.ts” з наступним вмістом:

```
// Імпорт функції authMiddleware з пакету "@clerk/nextjs"
import { authMiddleware } from "@clerk/nextjs";

// Експорт результату виклику функції authMiddleware з об'єктом конфігурації
export default authMiddleware({
  // Визначення публічних маршрутів, які не вимагають автентифікації
  publicRoutes: ["/api/:path*"],
});

// Експорт додаткового об'єкта конфігурації для цього модуля
export const config = {
  // Визначення власного збірника для маршрутів
  matcher: ["/((?!.*\\..*|_next).*)", "/", "/(api|trpc)(.*)"],
};
```

Після чого переходимо до створення сторінок автторизації, щоб користувач мав змогу зареєструватись, якщо немає акаунту, або перейти в існуючий кабінет використовуючи необхідні дані: Google акаунт або ім’я користувача та пароль. ClerkJS надає можливість налаштовувати всі типи

авторизації та реєстрації використовуючі різні платформи, такі як: Google, Facebook, GitHub, Microsoft, Linkedin та багато інших.

Для того щоб все працювало коректно в каталозі «app» ми створюємо наступну структуру, використовуючи можливості App-router. На рисунку 3.4 зображена структура маршрутів для сторінок авторизації та реєстрації.

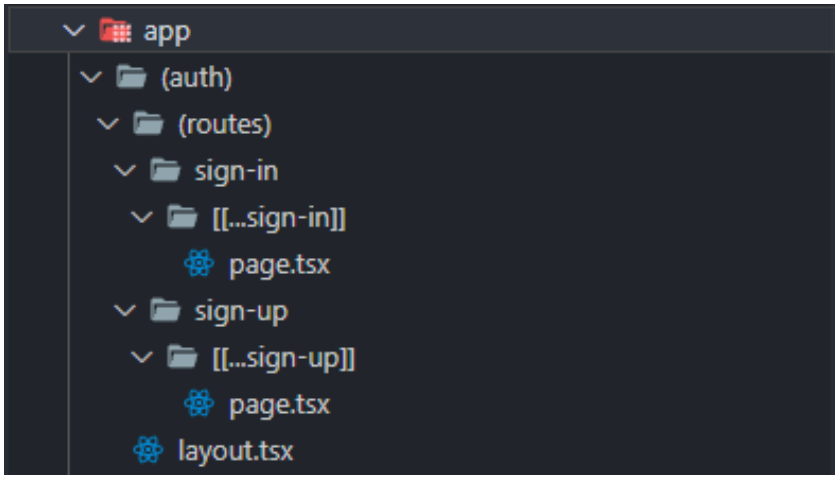


Рисунок 3.4 – Маршрути авторизації

ClerkJS – має обширну адміністративну панель, на якій ми може контролювати майже все, наприклад: відслідковувати кількість зареєстрованих користувачів, додавати, видаляти, редагувати доступ кожного користувача окремо, для тих чи інших цілей, додавати та видаляти різні способи реєстрації, змінювати UI для форм авторизації та реєстрації, сюди ж входить зміна кольорової палітри під потреби конкретного проекту, за необхідності встановлювати ліміт на кількість користувачів, або взагалі заборонити створювати нові кабінети юзерів.

Отже, розглянувши сильні сторони даного інструменту конкретно для даного проекту, повернемось до зовнішнього вигляду форми авторизації, після завершення всіх попередніх кроків та інструкцій ми маємо наступний результат, що зображено на рисунку 3.5.

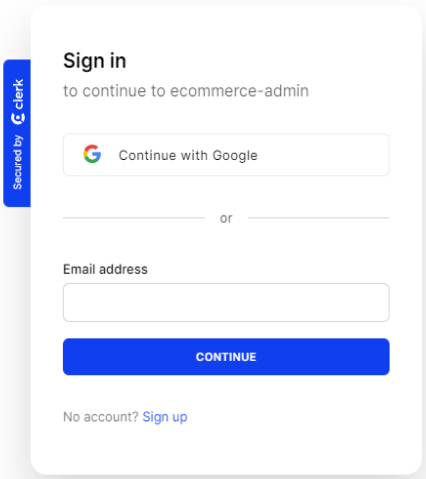


Рисунок 3.5 – Сторінка авторизації користувача

За допомогою даного інструменту ми економимо безліч часу на розробці такого функціоналу, як авторизація користувача. Тепер коли ми закінчили даний етап можна переходити до розробки інших складових автоматизованої системи контролю організаційних процесів інтернет-магазину.

3.3 Архітектура бази даних, розробка та підключення

Одне з найважливіших завдань під час розробки нашої CMS-системи – це звісно ж розробка БД, як було обговорено раніше наша MySQL база даних буде розміщена на платформі Planet Scale, тому першим завданням це буде звісно ж створення власного кабінету на даній платформі. Зробити це дуже просто необхідно слідувати всім вказівкам під час реєстрації, після чого ми потрапляємо в власний кабінет де ми і створимо нашу MySQL базу даних (вона створюється за замовчуванням), натиснувши на кнопку “Create new database”.

Після успішного виконання всіх вказівок по створенню БД, ми

потрапляємо на сторінку налаштувань БД, де ми отримуємо посилання яке необхідно вставити в схему Prisma, що і буде нашим наступним кроком.

Для того щоб працювати з нашою БД за допомогою ORM Prisma спочатку нам потрібно її встановити в наш проект, зробити це можна за допомогою терміналу VS Code.

В папці нашого проекту відкривши термінал нам потрібно вписати в нього наступні команди:

– `npm install prisma` – встановлення необхідних node модулів до нашого проекту

– `npx prisma init` – ініціалізація schema.prisma (файл в якому будуть зберігатись моделі БД)

Після успішного виконання в консолі ми бачимо наступне повідомлення про те що все пройшло успішно (рис – 3.6), а саме:

– наші змінні в файлі .env оновились;

– створилась папка «prisma» в корені проекту, що містить файл «schema.prisma» в якому ми і будемо працювати;

– та наступні необхідні кроки.

```
PS G:\front-end\NEXT_JS\ecommerce-adm> npm i prisma

added 2 packages, and audited 335 packages in 4s

117 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS G:\front-end\NEXT_JS\ecommerce-adm> npx prisma init

✓ Your Prisma schema was created at prisma/schema.prisma
  You can now open it in your favorite editor.

warn You already have a .gitignore file. Don't forget to add `.env` in it to not commit any private information.

Next steps:
1. Set the DATABASE_URL in the .env file to point to your existing database. If your database has no tables yet, read https://pris.ly/d/getting-started
2. Set the provider of the datasource block in schema.prisma to match your database: postgresql, mysql, sqlite, sqlserver, mongodb or cockroachdb.
3. Run prisma db pull to turn your database schema into a Prisma schema.
4. Run prisma generate to generate the Prisma Client. You can then start querying your database.

More information in our documentation:
https://pris.ly/d/getting-started
```

Рисунок 3.6 – Повідомлення в терміналі про успішне встановлення Prisma

Наступним нашим кроком буде саме налаштування файлу shema.prisma, та

створення моделей нашої бази даних. Для початку перейдемо в файл .env та перезапишемо змінну під назвою «DATABASE_URL» вставивши після знака присвоєння посилання на нашу MySQL базу даних, скопіювавши його з PlanetScale.

Далі переходимо в файл shema.prisma та можемо приступати до створення моделей бази даних.

У контексті Prisma, модель — це опис структури даних вашої програми, яка використовується для створення та взаємодії з базою даних. Модель у Prisma описує таблиці бази даних, їхні поля та зв’язки між ними. Він також може містити правила перевірки та інші налаштування. Кожна модель представляє таблицю в базі даних, а поля моделі відповідають стовпцям таблиці. Зв’язки між моделями описують зв’язки між таблицями.

Наша БД буде мати наступні моделі:

1. Store – Зберігає інформацію про магазин, включаючи ідентифікатор, назву, користувача, який його створив, та інші дати. Має зв'язки з більшістю інших моделей, таких як білборди, категорії, розміри, кольори, продукти та замовлення.
2. Billboard – Представляє білборд, пов'язаний з конкретним магазином. Має зображення, категорії та дати створення та оновлення. Забезпечує зв'язок із магазином та може мати індекс для швидкого доступу.
3. Category – Визначає категорію товарів у магазині та пов'язана з конкретним білбордом. Має також зв'язок з продуктами та датами створення та оновлення.
4. Product – Містить інформацію про товар, таку як ім'я, ціну, чи є він улюбленим чи архівованим тощо. Має зв'язки з категорією, розміром, коліром, зображеннями та іншими елементами.
5. Size – Представляє розмір товару та пов'язаний з магазином. Має зв'язок з продуктами та датами створення та оновлення.
6. Color – Визначає колір товару та пов'язаний з магазином. Має зв'язок з продуктами та датами створення та оновлення.

7. Image – Зберігає URL-адресу зображення товару та пов'язаний з конкретним продуктом.
8. Order – Має інформацію про замовлення, таку як статус оплати, телефонний номер та адресу. Містить також зв'язок із магазином та елементами замовлення.
9. OrderItem – Вказує, які товари були включені в конкретне замовлення.

Описана база даних надає засоби для ефективної взаємодії з інформацією про магазини, рекламні щити, категорії, продукти та замовлення. Така організація дає змогу ретельно контролювати товарні запаси, включаючи їх розміри та кольори, обробляти замовлення та пропонувати зручний доступ до зображень товарів через інтегровані моделі.

На рисунку 3.7 ви може більш детально розглянути архітектуру бази даних та зв’язки моделей.

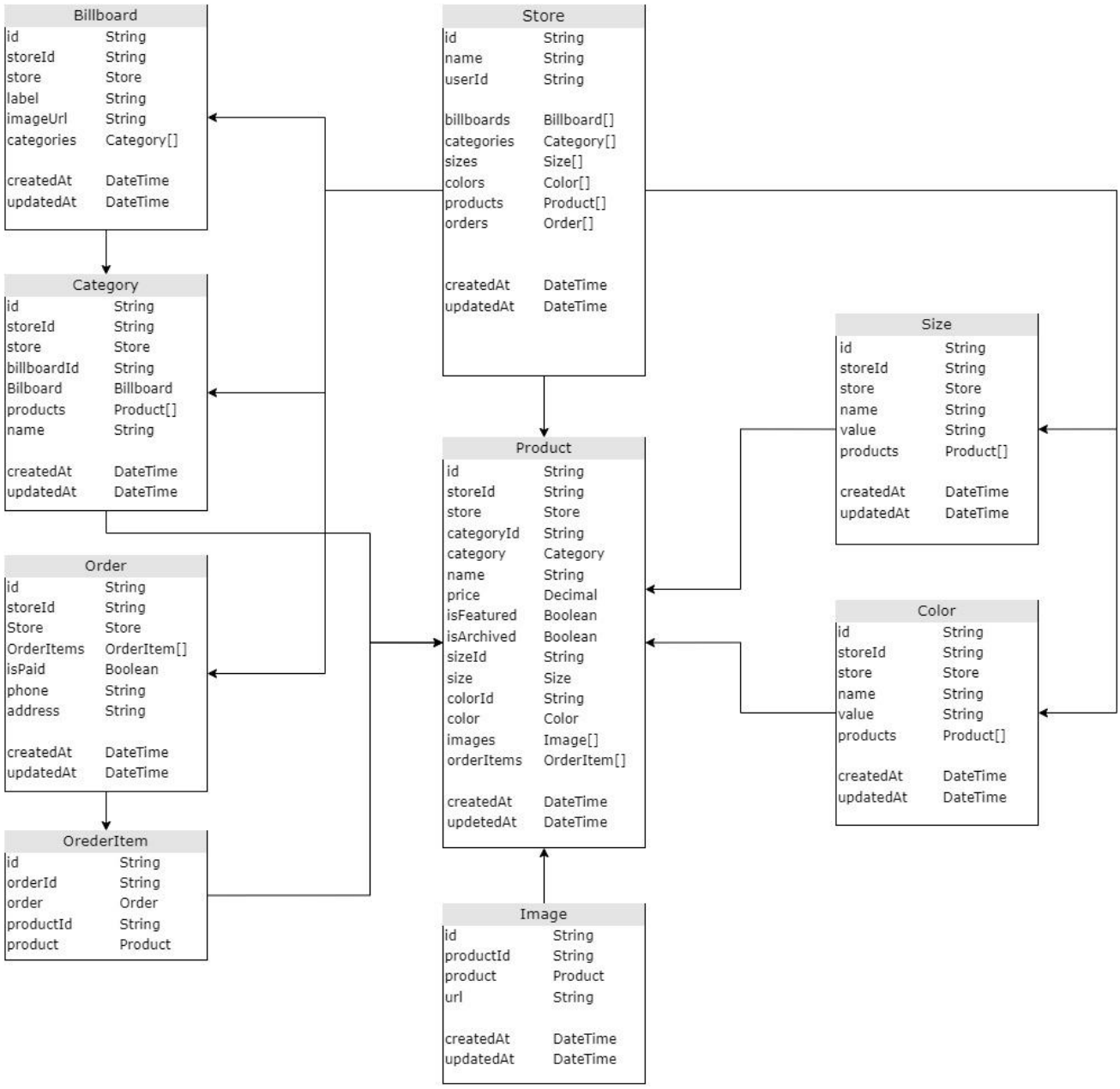


Рисунок 3.7 – Схема бази даних CMS-системи контролю організаційними процесами в інтернет магазині

Після створення усіх необхідних моделей, які будуть зберігати дані, нам необхідно перенести їх у нашу MySQL базу, що знаходиться в PlanetScale, для того щоб мали до них доступ та могли взаємодіяти на майбутніх етапах розробки.

Отже, для цього ми робимо міграцію бази даних, та за допомогою команди «prisma db push» додаємо наші моделі даних на платформу контролю, після цього переходимо до власного кабінету PlanetScale та бачимо

що тепер ми маємо 10 доступних таблиць таблиць, одна таблиця це виконана нами міграція (backup), та 9 моделей про які ми говорили вище, також ми можемо бачити деякі характеристики, такі як розмір таблиці, та розгорнувши її, поля які в ній зберігаються.

На рисунку 3.8 зображений список створених нами таблиць бази даних.

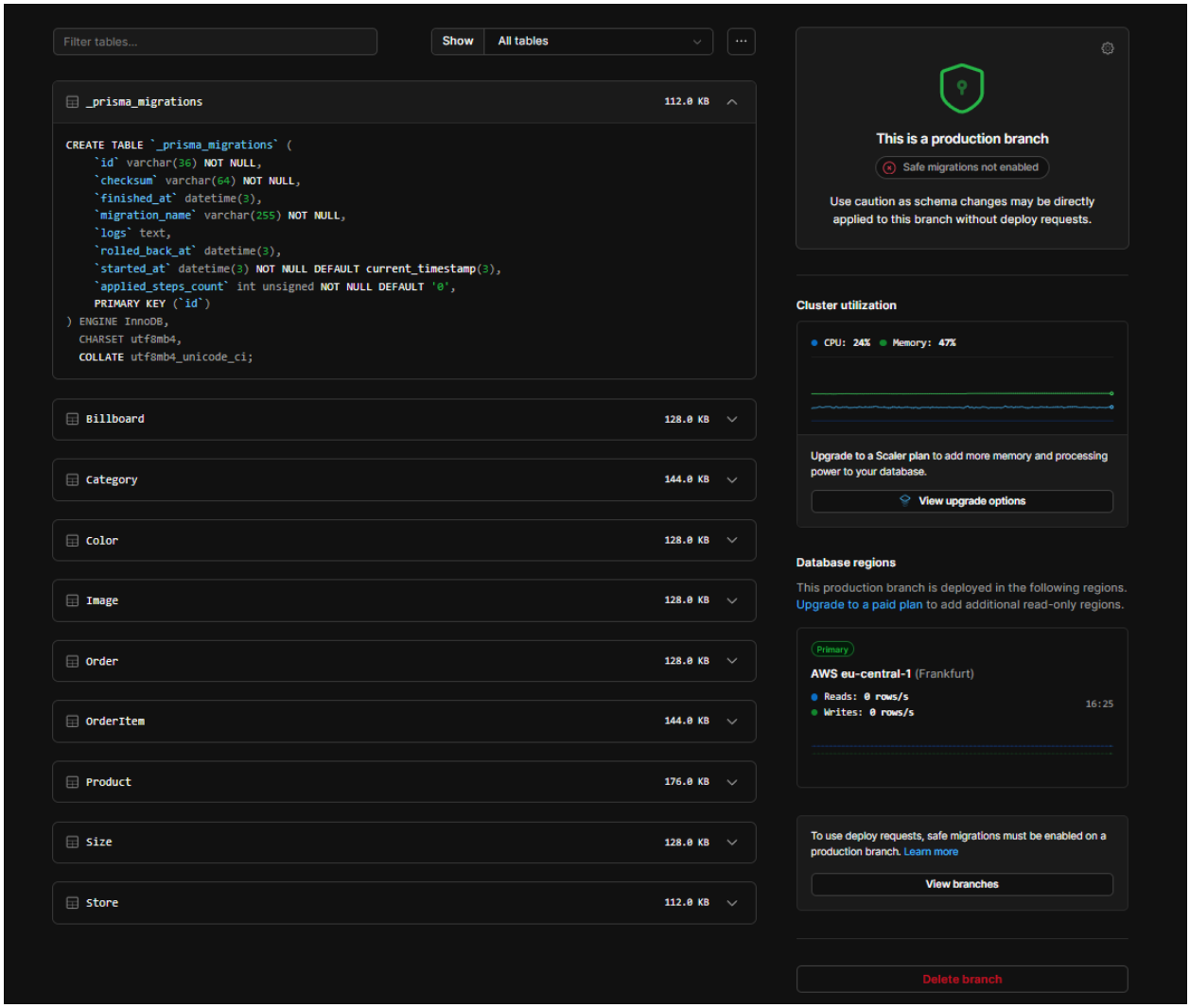


Рисунок 3.8 – Таблиці для збереження даних в нашому кабінеті PlanetScale

Щоб взаємодіяти з базою даних, писати різного роду функції та отримувати дані, які ми повинні показувати користувачу, нам потрібна ще одна деталь, а саме – PrismaClient, що являється свого роду конструктором запитів, який забезпечує безпечний тип доступу до бази даних і зменшує шаблонність.

Створимо новий файл в папці «lib» нашого проекту, під назвою «prismadb.ts», та заповнюємо його наступним кодом, кожна стрічка якого

прокоментована для більшого розуміння:

```
//? Імпортуємо PrismaClient з бібліотеки @prisma/client
import { PrismaClient } from "@prisma/client";

//? Оголошуємо глобальну змінну prisma типу PrismaClient або undefined
declare global {
  var prisma: PrismaClient | undefined;
}

//? Створюємо екземпляр PrismaClient, або використовуємо глобальний, якщо він вже існує
const prismadb = globalThis.prisma || new PrismaClient();

//? Якщо не в режимі продакшн, присвоюємо створений екземпляр глобальній змінній
if (process.env.NODE_ENV !== "production") {
  global.prisma = prismadb;
}

//? Експортуємо створений або використаний екземпляр PrismaClient
export default prismadb;
```

Тепер ми де завгодно зможемо імпортувати екземпляр PrismaClient та взаємодіяти з нашою базою даних. Взаємодія з базой даних буде реалізована за допомогою API-routes.

3.4 Розробка клієнтського інтерфейсу адміністративної панелі управління

Так як NextJS це фреймворк на базі React, а основна ідея React-ту це компонентний підхід, що являє собою створення перевикористовуваних компонентів, тобто умовне розподілення веб-сайту на невеликі частини, які ми потім збираємо у повноцінні веб-сторінки.

Також у NextJS є свої особливості підчас розробки користувацького інтерфейсу, а конкретніше використовуючи App-router, тобто функціонал надає нам такі можливості:

– компонент Layout – визначає загальну структуру сторінок додатку. Він містить елементи, які повторюються на всіх сторінках, такі як header, footer або sidebar. Використовуючи Layout, ми можемо зручно забезпечити

єдність дизайну та навігації.

- компонент Loading – використовується для відображення індікатора завантаження або анімації під час отримання даних або інших асинхронних операцій. Він дозволяє користувачеві знаходитися у курсі процесу завантаження сторінки.
- компонент Error – використовується для відображення інформації про помилку. Він дозволяє контролювати зовнішній вигляд та поведінку сторінки у випадку, коли сталася помилка.
- компонент Page – являється основним компонентом та представляє собою саму сторінку. Кожна сторінка в NextJS повинна мати компонент Page. В Page ми визначаємо контент сторінки та використовуєм дані, отримані з сервера або інших джерел.

Отже, в для кожної сторінки нашого веб-додатку ми створюємо окрему папку в директиві «app» в середовищі розробки, також слід пам’ятати про маршрутизацію App-router про яку ми говорили в попередніх аналітичних розділах.

Окремі невеликі створені нами компоненти, як і компоненти Shadcn-UI, які ми також будемо використовувати під час розробки, ми будемо зберігати окремо в корені проекту в директиві під назвою «components» та за необхідності на тій чи іншій сторінці імпортувати їх в файл “page.tsx”.

Щоб почати використовувати компоненти представлені в Shadcn-ui нам необхідно за допомоги декількох команд ініціалізувати дану бібліотеку в нашому проекті.

Відкривши термінал для почтаку ми пишема команду:

- `npx shadcn-ua@latest init` – виконання даної команди оновлює файл “tailwind.config.ts” слідуючи нашим вказівках під час ініціалізації. В даному файлі зберігаються різні допоміжні класи та змінні кольорів, розмірів для стилізації елементів веб-сторінки.

На цьому все, тепер ми можемо встановлювати потрібні нам компоненти зі списку що надає нам дана збірка компонентів використовуючи метод CLI,

простими словами ми просто використовуємо той самий термінал VS Code, просто пишемо команду:

- `npx shadcn-ui@latest add [назва потрібного компоненту]` – після виконання даної команди необхідний вказаний нами компонент з'являється в папці «» та готовий до використання, такоє ми маємо повний досту до нього, що дає нам можливість змінювати його будьяким чином за необхідності, на мою думку це являється не абияким бонусом для використання даної бібліотеки.

Також підчас розробки користувацького інтеферфейсу ми будемо дотримуватись деяких правил проектування інтерфейсів[59], щоб наш додаток був кросбраузерним, та коректно відображався на всіх типах девайсів починаючи від широкоформатних дисплеїв, до екранів смортфонів. Для досягнення даних цілей підчас стилізації елементу, який того потребує, ми будемо використовувати спеціальні класи, що надає нам TailwindCSS, в яких використовується система брейкпоінтів, тобто при досягненні максимальної ширини цільового екрану стилі будуть динаміно змінюватись.

Прейдемо до структури інтерфейсу додатку. Користувачу буде доступний такий функціонал додатку:

1. Реєстрація та авторизація
2. Створення інтернет-магазину – якщо користувач не має магазину то йому необхідно обовязково його створити.
3. Домашня сторінка магазину – на даній сторінці буде зберігатись статистика конкретного магазину, така як: сума всіх продажів, кількість проданих одиниць товару, кількість товару в наявності та статиска продажів по місяцям у вигляді діаграми.
4. Сторінка перегляду білбордів та їх налаштування – на даній сторінці користувач може переглянути, які білборди використовуються в його магазині, також матиме можливість їх редагувати, видаляти, або створювати нові. Білборди прив’язуються до відповідно обраної категорії і відображається на сторінці категорії інтернет-магазину.

5. Сторінка перегляду категорій товарів та їх налаштування – на даній сторінці користувач так само може робитi всі дії з категоріями свого товару. Категорії товару будуть використовуватись для одного з способів сортування товару в інтернет-магазині
6. Сторінка перегляду розмірів товару та їх налаштування – дана сторінки дає доступ до перегляду, редагування та видалення розмірів товару.
7. Сторінка перегляду кольорів товару та їх налаштування – дана сторінки дає доступ до перегляду, редагування та видалення кольорів товару.
8. Сторінка перегляду продуктів та їх налаштування – дана сторінка дає доступ до створення товарів використовуючи всі попередньо створені дані, такі як: категорія, розмір, колір продукту.
9. Сторінка перегляду покупок – на даній сторінці відображається історія покупок сдійснених покупцями в інтернет магазині.
10. Сторінка налаштування магазину – на даній сторінці адміністратор може змінювати такий параметр магазину, як його назва, або видалити поточний магазин.

На рисунку 3.9 зображена блок схема дерева компонентів для аміністративної панелі CMS-системи.

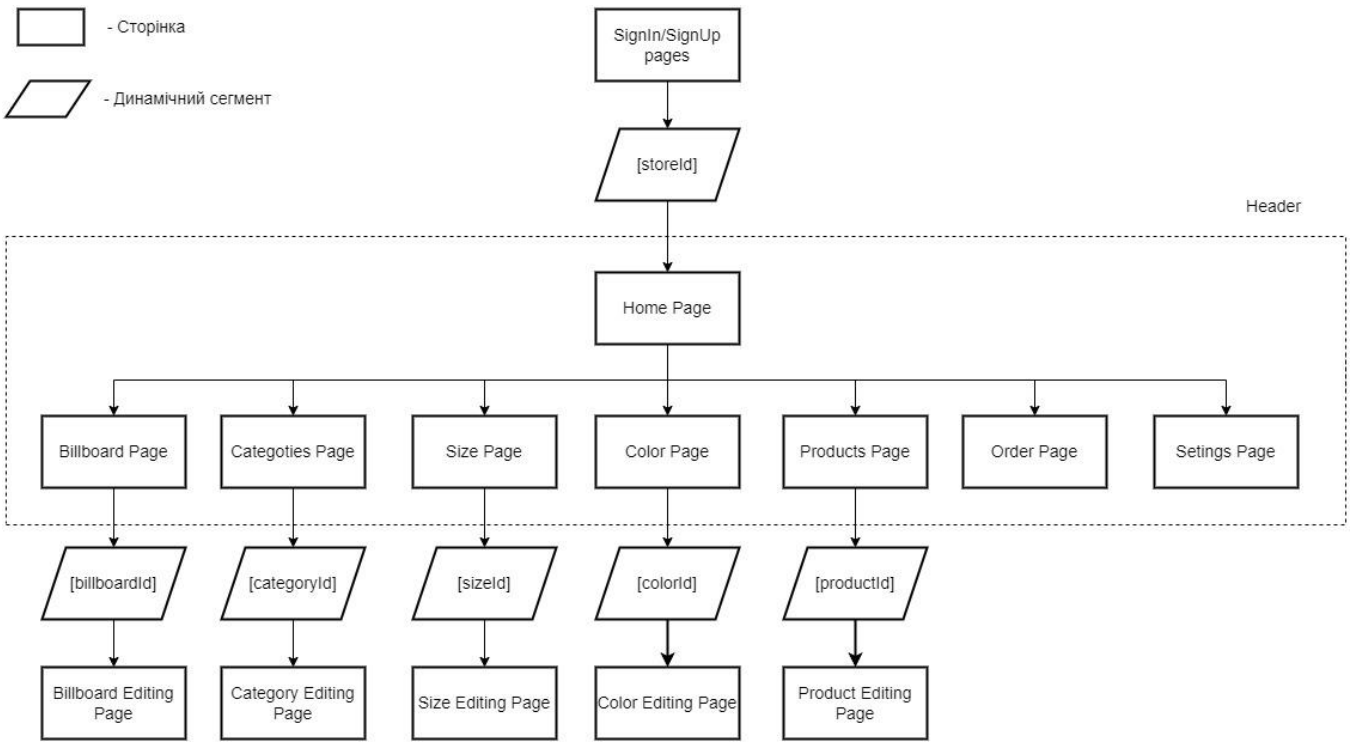


Рисунок 3.9 – Дерево компонентів CMS-системи

Також необхідно приділити особливу увагу розробці зв’язку нашої CMS-системи з інтернет магазином, яка буде реалізована за допомогою API.

Для того щоб реалізувати даний метод на кожній сторінці з налаштуваннями того чи іншого елементу в адмін панелі, ми розроби спеціальні компонети в яких буде зберігатись необхідне посилання, на яке потім буде робитись запит на отримання даних з інтернет магазину.

Для цього ми спочатку створимо кастомний хук який буде отримувати динамічне посилання для отримання даних з адміністративної панелі:

```
// Імпорт хуків useState та useEffect з бібліотеки React
import { useState, useEffect } from "react";

// Експорт функції userOrigin, яка повертає початковий URL користувача
export const userOrigin = () => {
  // Створення стану для відстеження того, чи компонент вже відмонтовано
  const [mounted, setMounted] = useState(false);

  // Визначення початкового URL на основі наявності об'єкта window та його
властивості location.origin
  const origin =
    typeof window !== "undefined" && window.location.origin
      ? window.location.origin
      : "";

  // Використання useEffect для встановлення значення mounted в true при монтуванні
компонента
  useEffect(() => {
    setMounted(true);
  }, []);

  // Якщо компонент ще не відмонтовано, повертаємо порожній рядок
  if (!mounted) {
    return "";
  }

  // Повертаємо початковий URL користувача
  return origin;
};
```

Далі використтовую створюємо компонент в якому й буде відображатись дане посилання, на рисунку 3.10 зображений вигляд даного компоненту на сторінці налаштувань магазину.

Програмно даний компонент матиме наступний вигляд:

```
import { userOrigin } from "@/hooks/use-origin";
```

```
export const ApiUrl = () => {

const origin = userOrigin();

      return(
      <ApiAlert
        title="NEXT PUBLIC API URL"
        description={`${origin}/api/${params.storeId}`}
        variant="public"
      />
      )
}
```

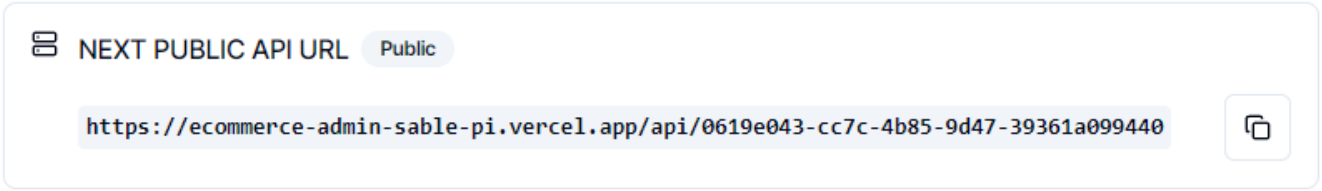


Рисунок 3.10 – Компонент що відображає API адресу для отримання даних з адміністративної панелі в інтернет-магазині

Даний компонет ми будемо використовувати на кожній сторінці адміністративної панелі щоб ми мали доступ до кожного елементу окремо. Тепер використовуючи дане посилання ми зможемо коректно відображати усі дані в нашому інтернет магазині, та покупець зможе взаємодіяти з ними.

Останьою основною задачею буде підключення платіжної системи Stripe. Для того щоб щоб використовувати дану систему обробки платежів реєструємось на офіційному сайті, та створюємо проект. Після чого отримуємо секретний ключ, який ми копіюємо та вставляємо в “.env” файл під назвою «STRIPE_WEBHOOK_SECRET». Наступним кроком юуде створення так званого вебхука.

Вебхуки (webhooks) представляють собою механізм взаємодії між веб-серверами, який дозволяє одному серверу автоматично сповіщати інший про зміни або події, які сталися. Це використовується для отримання миттєвих оновлень, коли відбуваються певні події або зміни в джерелі даних.

Зазвичай вебхуки використовуються в розробці веб-додатків та API. Замість того, щоб періодично опитувати сервер для отримання нової інформації,

веб-додаток може встановити вебхук і отримувати повідомлення в режимі реального часу, коли стаються події.

Процес роботи вебхуків виглядає приблизно так:

1. Додаток, який хоче отримувати повідомлення, реєструє вебхук на сервері, який генерує ці події.
2. Коли відбувається певна подія, сервер, що генерує події, автоматично надсилає HTTP-запит (зазвичай POST-запит) на URL, який був вказаний при реєстрації вебхука.
3. Сервер отримує цей HTTP-запит і обробляє його, надсилаючи необхідну інформацію або сповіщення.

Отже, за допомогою вебхука ми можемо ефективно взаємодіяти з подіями в режимі реального часу, що є корисним в багатьох веб-сценаріях, таких як сповіщення про нові повідомлення, оновлення даних або інші важливі події, в нашому випадку саме оновлення даних, оскільки коли потенційний покупецб зробить оплату товару в інтернет магазині, даний товар повинен перейти з статусу «в найявності» до статусу «немає в наявності».

Перейдемо до розробки вебхуку для Stripe, в директиві «app/api» створюємо нову папку під назвою «webhook» та в ній створюємо файл під назвою “route.ts”, наша ціль написати функцію POST, яка перевіряє валідність вхідних даних та виконує дії в залежності від отриманої події, таких як оновлення інформації про замовлення та архівація продуктів в базі даних.

Дана функція матиме наступний вигляд:

```
// Оголошення асинхронної функції POST, яка обробляє POST-запити для вебхуків
Stripe
export async function POST(req: Request) {
  // Отримання тіла запиту в текстовому форматі
  const body = await req.text();

  // Отримання підпису з хедеру Stripe-Signature
  const signature = headers().get("Stripe-Signature") as string;

  let event: Stripe.Event;

  try {
    // Перевірка валідності події Stripe за допомогою конструктора подій stripe
    event = stripe.webhooks.constructEvent(
      body,
```

```
      signature,
      process.env.STRIPE_WEBHOOK_SECRET!,
    );
  } catch (error: any) {
    // Обробка помилки, якщо перевірка не вдалась
    return new NextResponse(`Webhook Error: ${error.message}`, { status: 400 });
  }

  // Отримання даних з об'єкта Checkout.Session для подальшої обробки
  const session = event.data.object as Stripe.Checkout.Session;
  const address = session?.customer_details?.address;

  // Формування рядка адреси з компонентів адреси
  const addressComponents = [
    address?.line1,
    address?.line2,
    address?.city,
    address?.state,
    address?.postal_code,
    address?.country,
  ];

  const addressString = addressComponents.filter((c) => c !== null).join(", ");

  // Перевірка типу події для подальшої обробки
  if (event.type === "checkout.session.completed") {
    // Оновлення даних замовлення в базі даних
    const order = await prismadb.order.update({
      where: {
        id: session?.metadata?.orderId,
      },
      data: {
        isPaid: true,
        address: addressString,
        phone: session?.customer_details?.phone || "",
      },
      include: {
        orderItems: true,
      },
    });

    // Отримання ідентифікаторів продуктів з замовлення та оновлення їх статусу в
базі даних
    const productIds = order.orderItems.map((orderItem) => orderItem.productId);

    await prismadb.product.updateMany({
      where: {
        id: {
          in: [...productIds],
        },
      },
      data: {
        isArchived: true,
      },
    });   return new NextResponse(null, { status: 200 });
```

Тепер коли покупець буде робити замовлення в інтернет-магазині та підтверджувати його буде викликатись дана функція, та дані, за нобхідності

будуть змінюватись, та коректно відображатись.

На цьому етапі можна вважати закінченою розробку CMS-системи, але додатково для кращого UX ми додамо різного роду попапи, вспливаючі вікна що показують успішне виконання того чи іншого процесу, або помилку, також додамо можливість перемикатись між оформленням додатку(темами) на денний та нічний режим.

3.5 Розробка інтернет-магазину

Почнемо розробку інтернет магазину так само використовуючи стартовий шаблон NextJS.

Для розробки інтернет магазину ми будемо притримуватись тих самих принципів, що і для адміністративної панелі, тобто адаптивність під всі розширення екранів та кроссбраузерність. Основна відмінність полягає в тому, що дані в інтернет магазині будуть отримуватись безпосередньо за допомоги API адрес що були створені нами підчас розроби CMS-системи.

Нашою основною ціллю є створення e-commerce платформи, що не поступується іншим на ринку електронної комерції. Тому нам необхідно щоб розроблена система була розроблена за всіма сучасними стандартами і відповідала усім нормам. Для кращого візуального сприйнятя було розроблено UML діаграму послідовності подій, що відбуватимуться під час взаємодії користувача з інтерет-магазином.

Нижче, на рисунку 3.11, наведено чітку та стислу діаграму діяльності для онлайн-покупок в інтернет-магазині. Клієнт може переглядати або шукати товари, переглядати конкретні позиції, додавати їх у свій кошик для покупок, переглядати й оновлювати свій кошик для покупок і продовжувати оформлення замовлення. Користувач може отримати доступ до свого кошика для покупок у будь-який час, і припускається, що оформлення замовлення передбачає реєстрацію користувача та вхід. Цей приклад не включає розділення, причому передбачається, що більшість дій виконує клієнт.

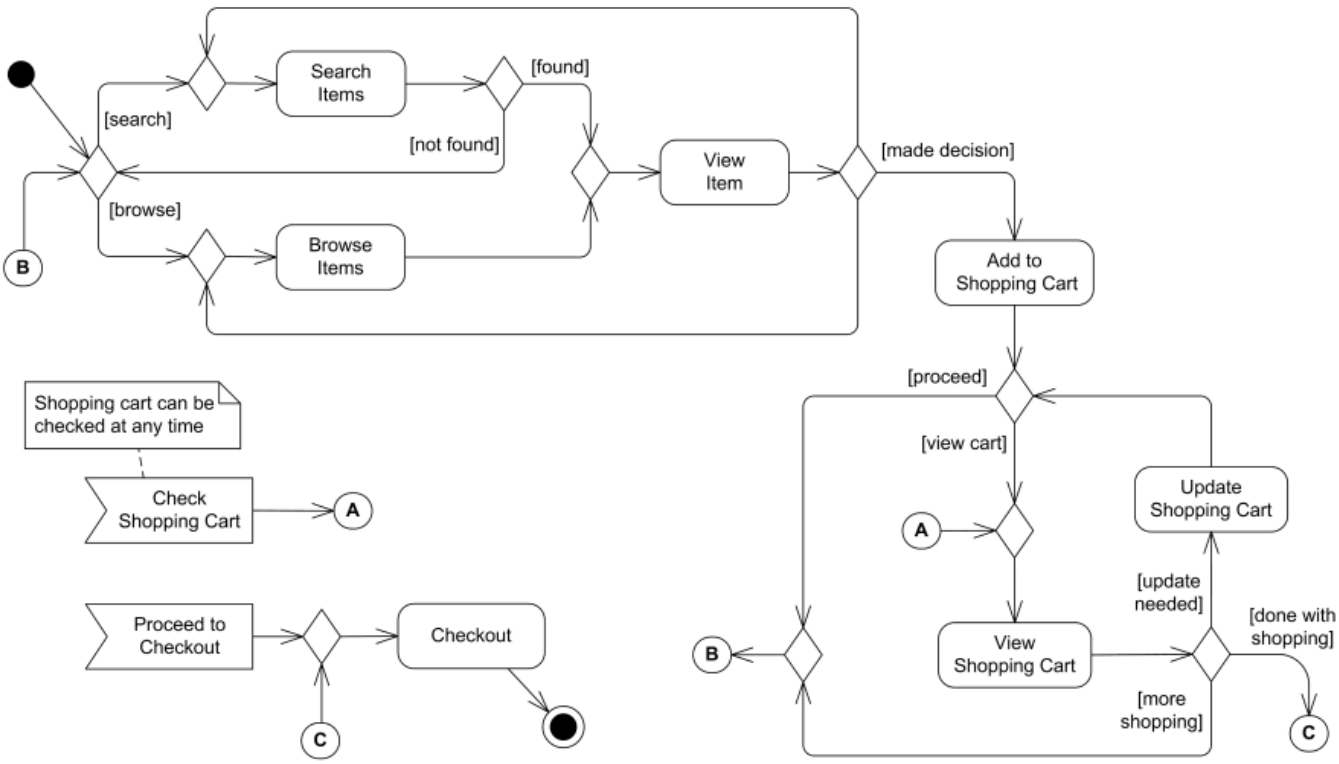


Рисунок 3.11 – Діаграму діяльності онлайн-покупок в інтернет-магазині

Визначившись основні функції які повинен включати в собі інтернет-магазин, перейдемо до розробки клієнтського інтерфейсу, та втілення визначених функцій у життя. Отже, для початку, перейдемо до огляду структури користувацького інтерфейсу в інтернет-магазині:

1. Домашня сторінка – на домашній сторінці відображаються всі товари що доступні в інтернет магазині, та білборд головної сторінки.
2. Сторінка категорії – при переході на сторінку категорії, ми будемо бачити лише ті товари, що належать до обраної категорії.
3. Сторінка продукту – при переході на сторінку обраного продукту користувач може детальніше оглянути його та при бажані додати обраний товар до корзини, заздалегідь обравши необхідні параметри товару, такі як розмір та колір.
4. Корзина – на даній сторінці користувач бачить всі додані ним товари, на цій сторінці він також може оплатити товар.

На рисунку 3.12 зображе блок схема дерева компонентів інтернет магазину.

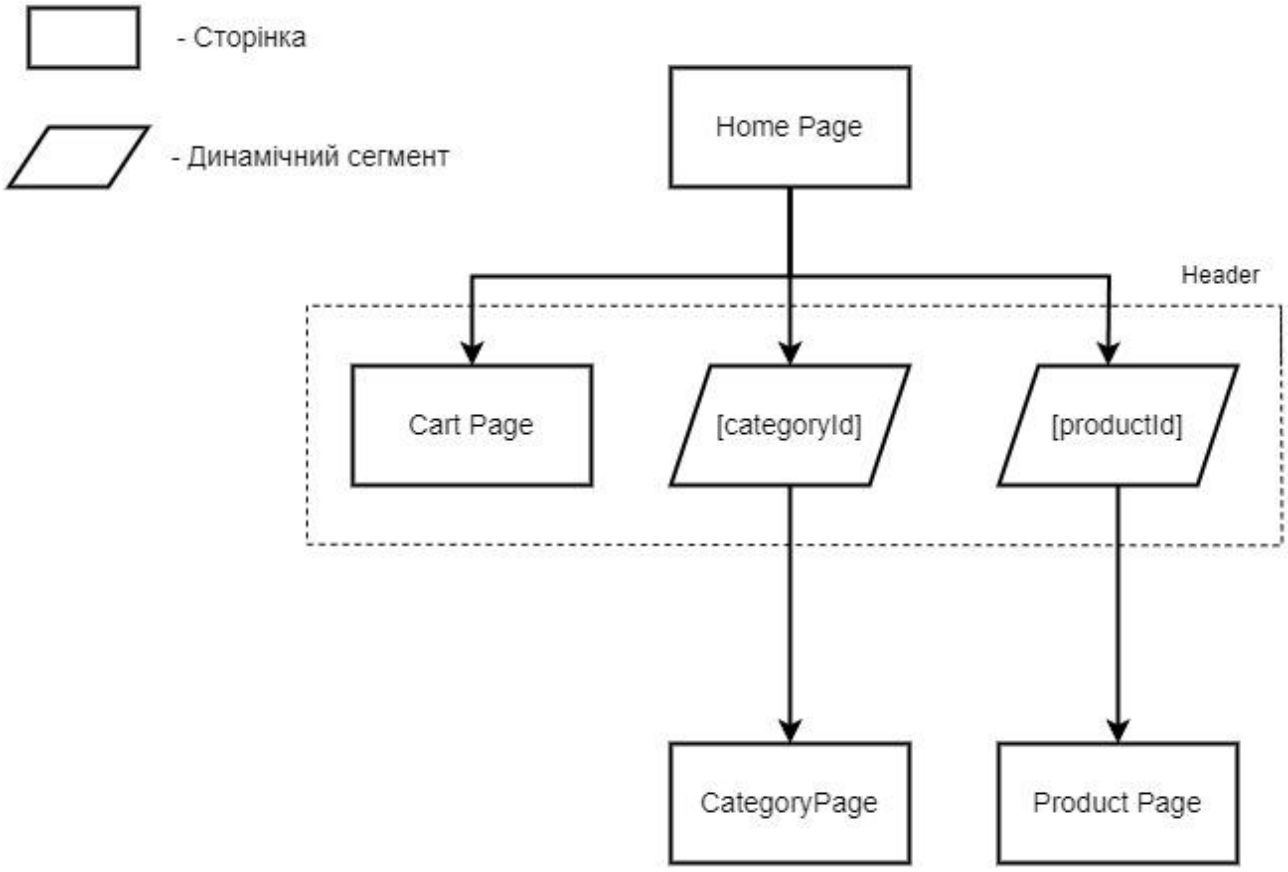


Рисунок 3.12 – Дерево компонентів інтернет магазину

Як вже раніше було сказано отримання даних в інтернет-магазині буде реалізоване за допомогою API, який ми копіюємо з адміністративної панелі, та вставляємо в файл ".env" назвемо змінну – «NEXT_PUBLIC_API_URL».

Для взаємодії з даною адресою нам потрібно описати функцію отримання даних, для кожного елемента окремо, для прикладу візьмемо домашню сторінку.

На домашній сторінці інтернет-магазину нем потрібно показати користувачу всі доступні йому товари, тому функція буде мати наступний вигляд:

```
// Імпорт бібліотеки для обробки рядків запитань у URL
import qs from "query-string";

// Імпорт інтерфейсу Product з файлу типів
import { Product } from "@/types";

// URL до ендпоінту API для отримання продуктів
const URL = `${process.env.NEXT_PUBLIC_API_URL}/products`;

// Оголошення інтерфейсу Query для визначення можливих параметрів запиту
interface Query {
```

```
  categoryId?: string;
  colorId?: string;
  sizeId?: string;
  isFeature?: boolean;
}

// Асинхронна функція getProducts для отримання продуктів з API за допомогою параметрів запиту
const getProducts = async (query: Query): Promise<Product[]> => {
  // Створення URL-рядка з параметрами запиту за допомогою бібліотеки query-string
  const url = qs.stringifyUrl({
    url: URL,
    query: {
      categoryId: query.categoryId,
      colorId: query.colorId,
      sizeId: query.sizeId,
      isFeature: query.isFeature,
    },
  });

  // Виклик API зі сформованим URL та отримання відповіді
  const response = await fetch(url);

  // Парсинг JSON-відповіді та повернення результату
  return response.json();
};

// Експорт функції getProducts для використання в інших частинах програми
export default getProducts;
```

Потім в компоненті в якому відбувається відображення домашньої сторінки ми викликаємо дану функцію, та передаємо дані далі по дереву для відображення, в даному випадку товарів. Таких функцій нам потрібно створити ще декілька, всі функції ми поміщаємо в дерикторію під назвою «actions», та в потрібному місці імпортуємо та викликаємо їх.

Також є сенс додати, що розробка, як CMS-системи, так і інтернет магазину була реалізована притримуючись деяких правил SOLID – принципу програмування, а саме таким частинам: Single Responsibility Principle (Принип Єдиної відповідальність) та Interface Segregation Principle (Принцип разделения интерфейса), за допомогою слідування цим правилам ми розробляємо додатки, які підлягають розширюваності коду, легшій первірці на помилки, та структуруванню файлів середовищі.

На рисунку 3.13 зображена файлова структура завершеного інтернет-магазину.

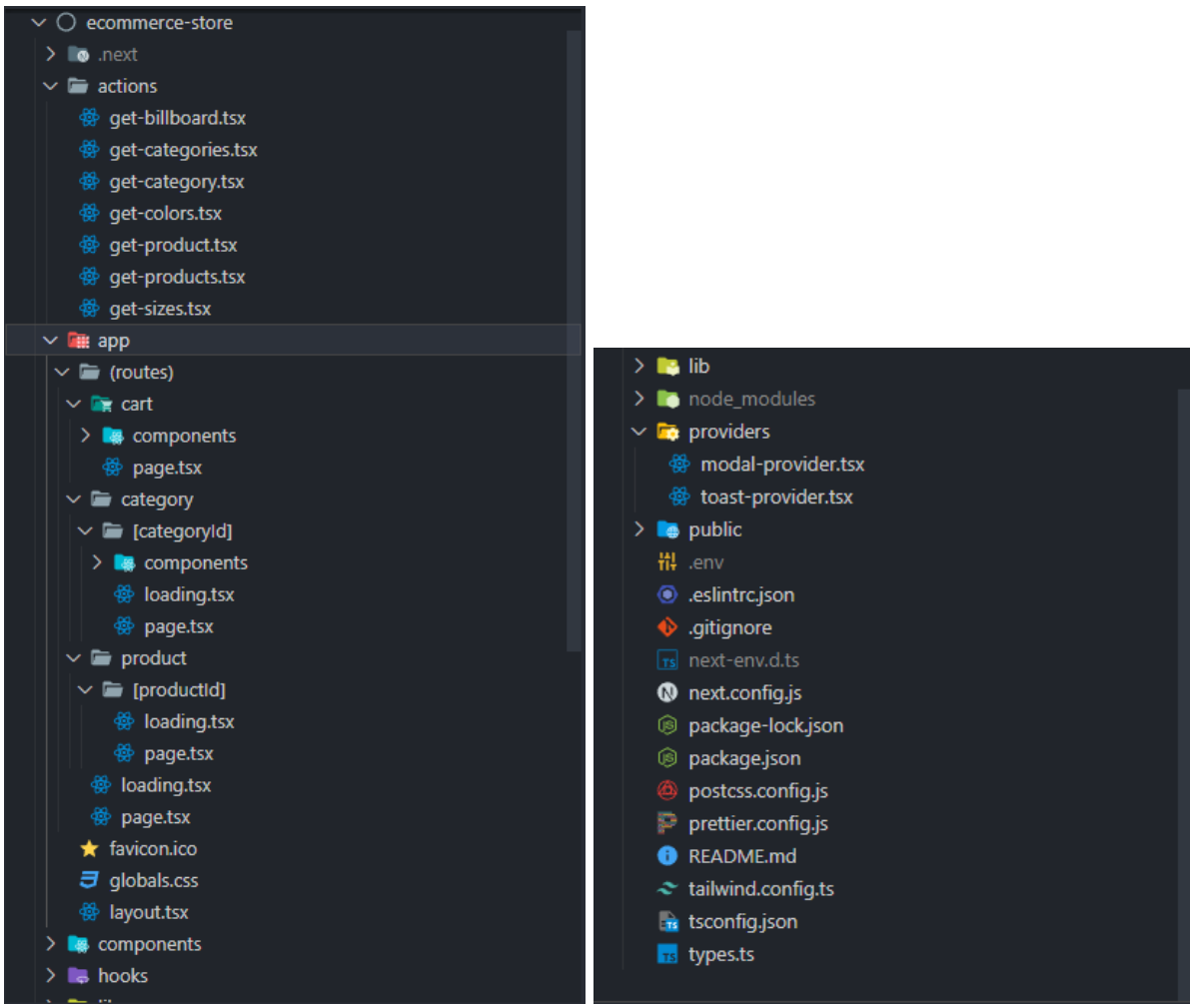


Рисунок 3.13 – файлова структура додатку – інтернет-магазин

3.6 Огляд та тестування розроблених додатків

Після завершення розробки наступним не менш важливим етапом йде тестування. В даному підрозділі ми оглянемо та протестуємо обидва розроблені нами додатки. Тестування адаптивності дизайну буде проводитись за допомогою Chrome DevTools[60].

Тестування буде поділятись на декілька етапів:

– Оглянемо та перевіримо користувацький інтерфейс на наявність багів
– Перевіримо чи коректно працює вся функціональність додатку CMS-системи та інтернет-магазину
– Спробуємо скористатись оплатою товару

Поперше, почнемо з зовнішнього вигляду, для кращого візуального представлення заздалегідь були додані деякі тестові дані до адміністративної панелі інтернет магазину.

На рисунках 3.14 – 3.18 зображено користувацький інтерфейс CMS-системи.

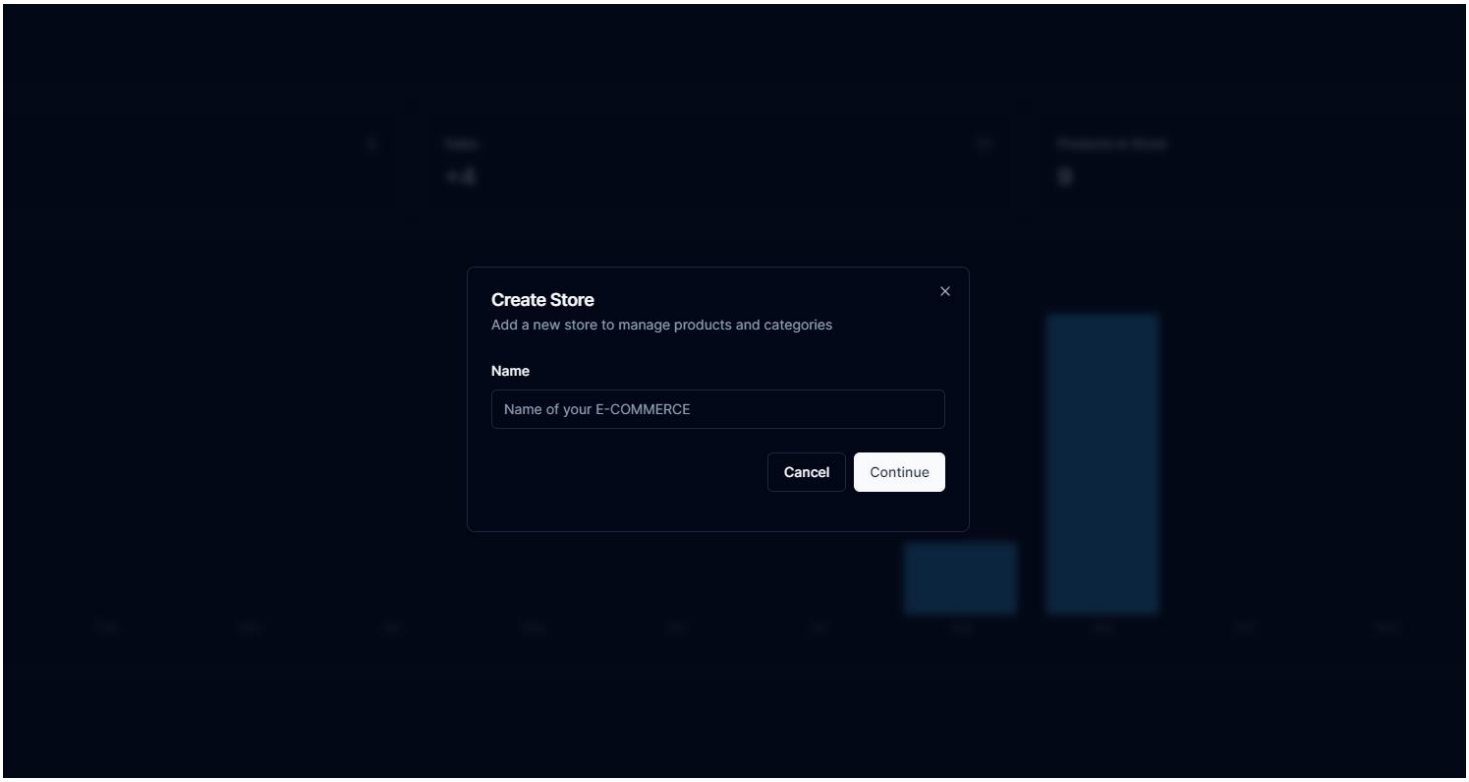


Рисунок 3.14 – Модальне вікно створення нового магазину

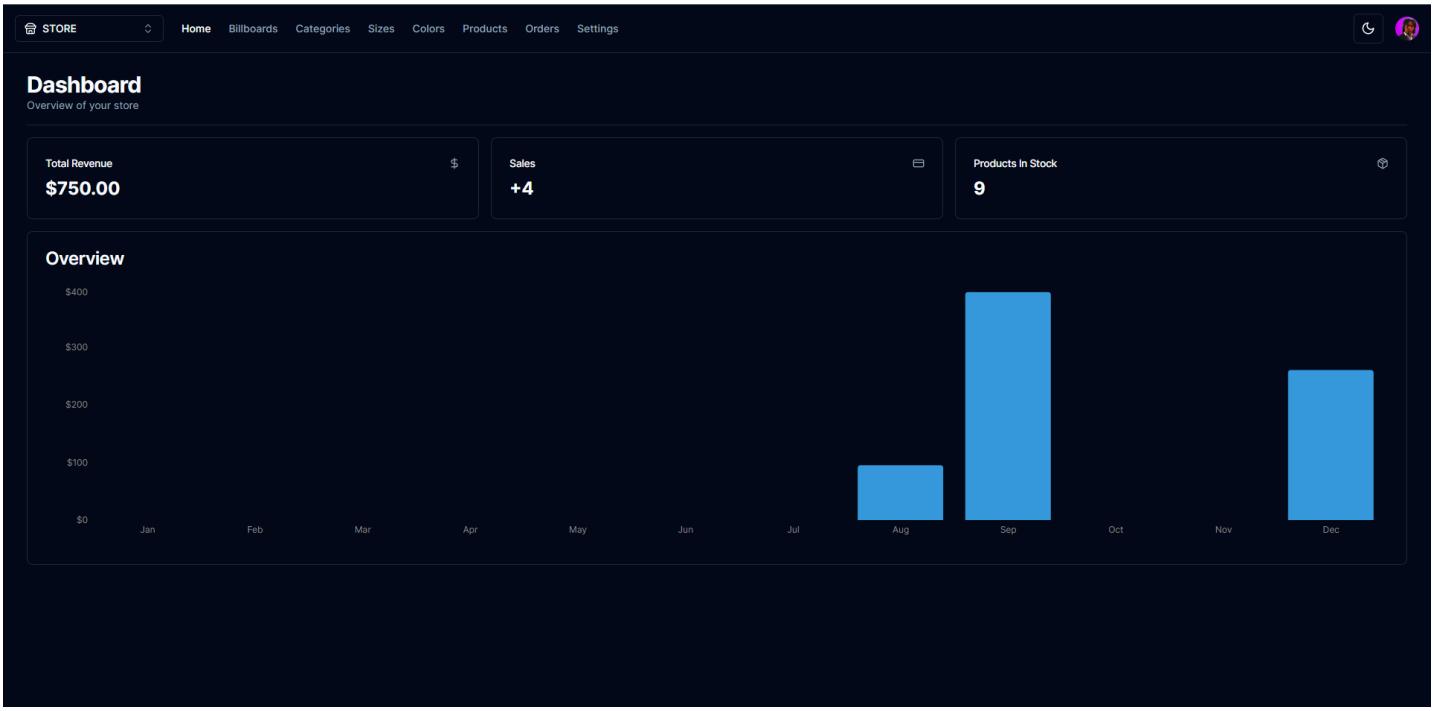


Рисунок 3.15 – Домашня сторінка CMS-системи

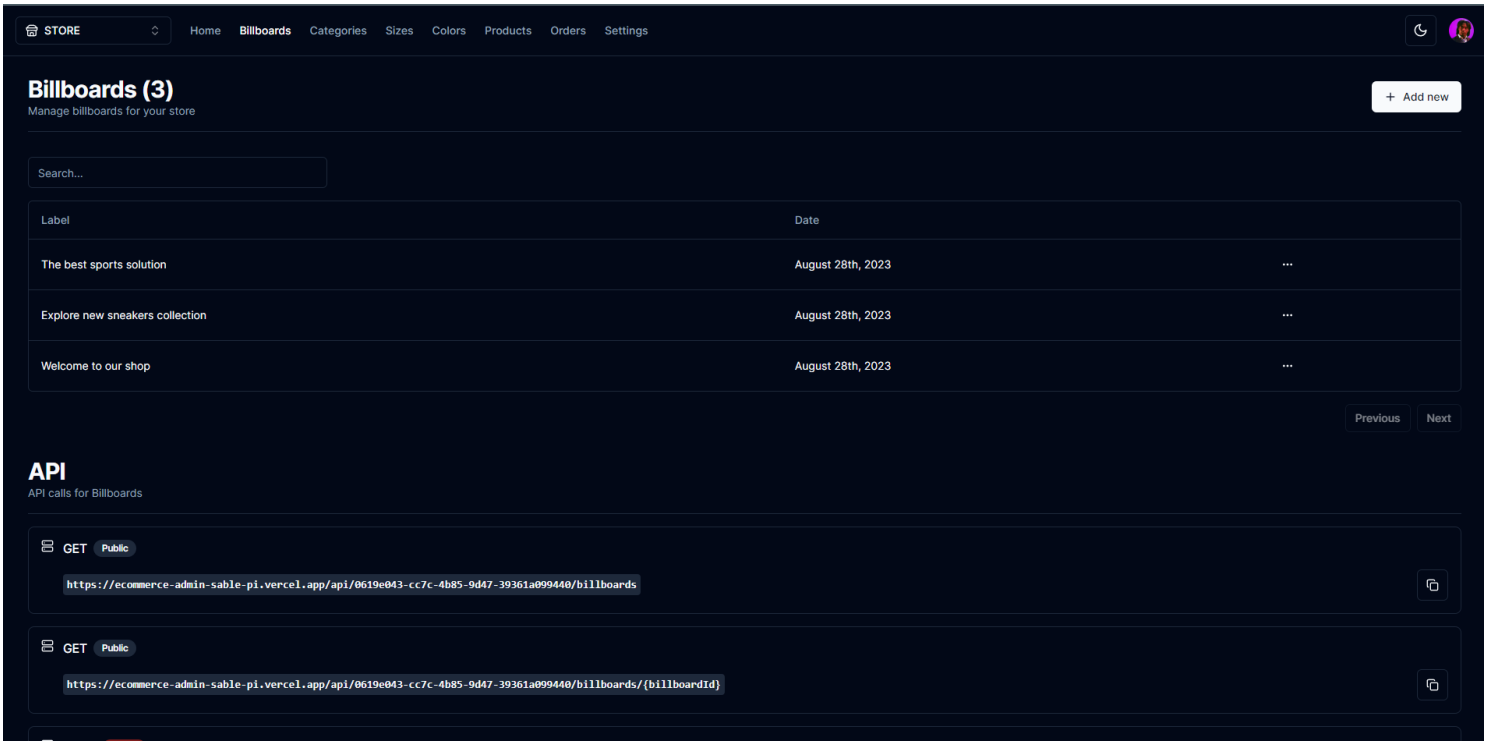


Рисунок 3.16 – Сторінка перегляду білбордів

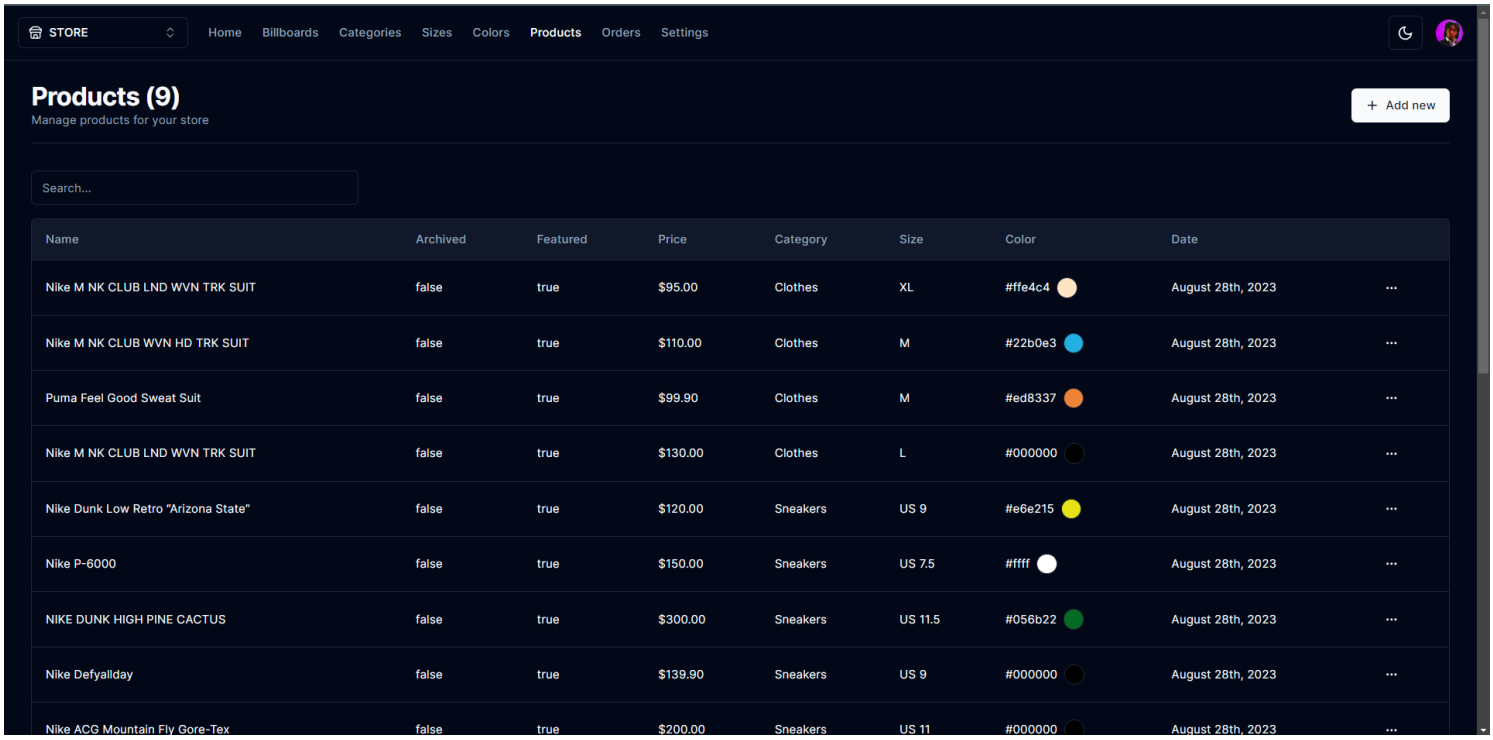


Рисунок 3.17 – Сторінка перегляду товарів

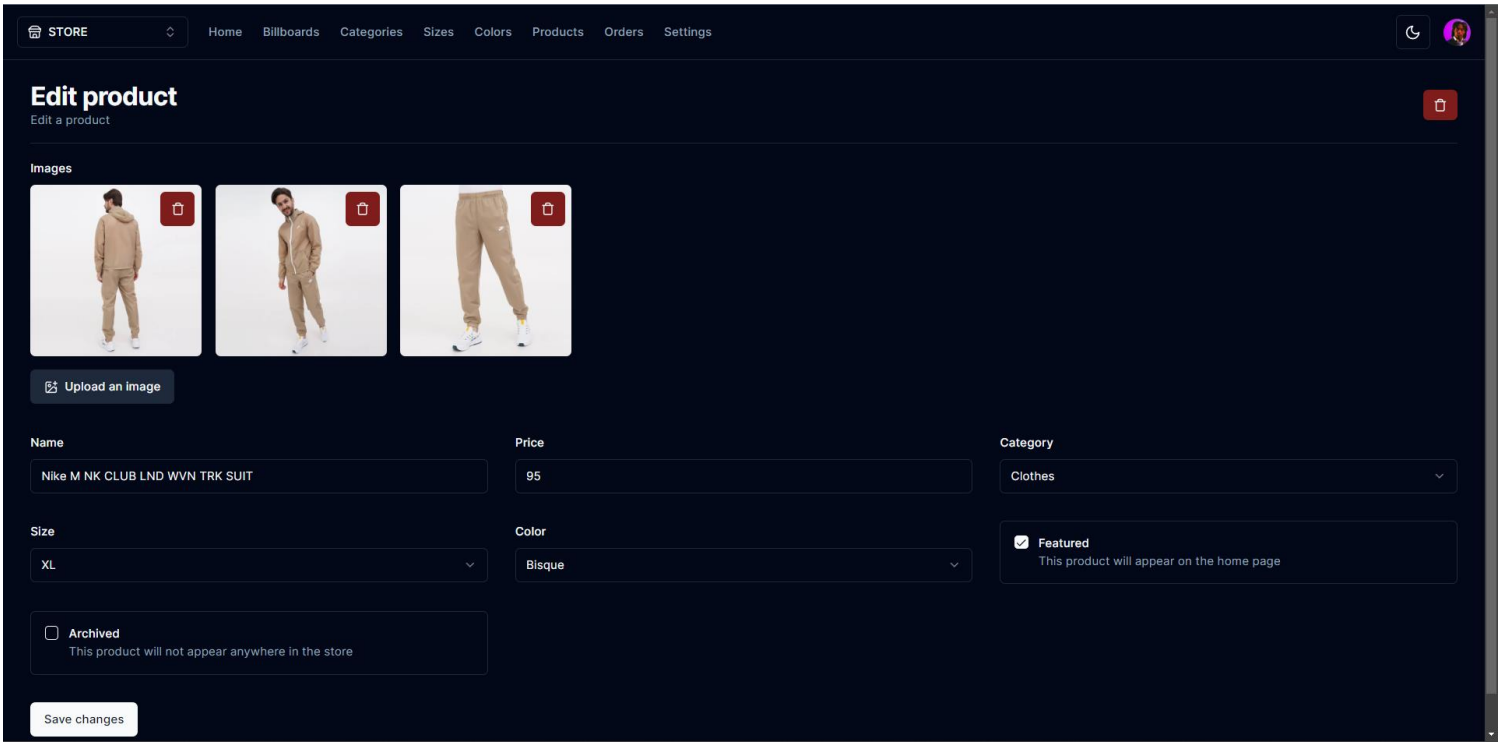


Рисунок 3.18 – Сторінка редагування товару

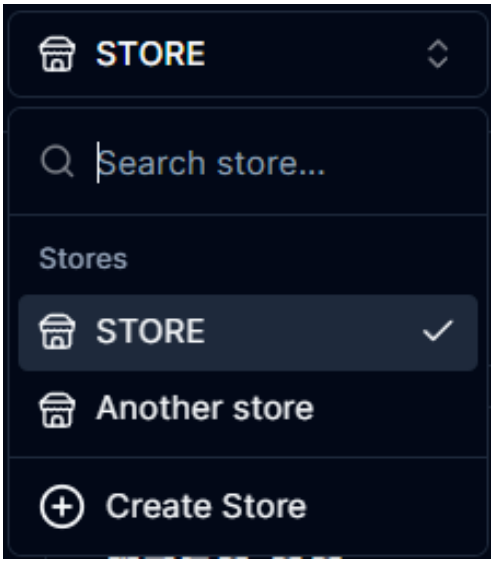


Рисунок 3.19 – Компонент перемикання між різними магазинами

Під час проведення огляду користувацького інтерфейсу було також перевірено наступні функціональні можливості, такі як: редагування, додавання, видалення, усіх можливих елементів в адміністративній панелі.

Перейдемо до тестування інтернет-магазину, будемо тестувати за тим

самим принципом, що і CMS-систему. На рисунках 3.20 – 3.24 зображено користувацький інтерфейс розробленого інтернет-магазину.

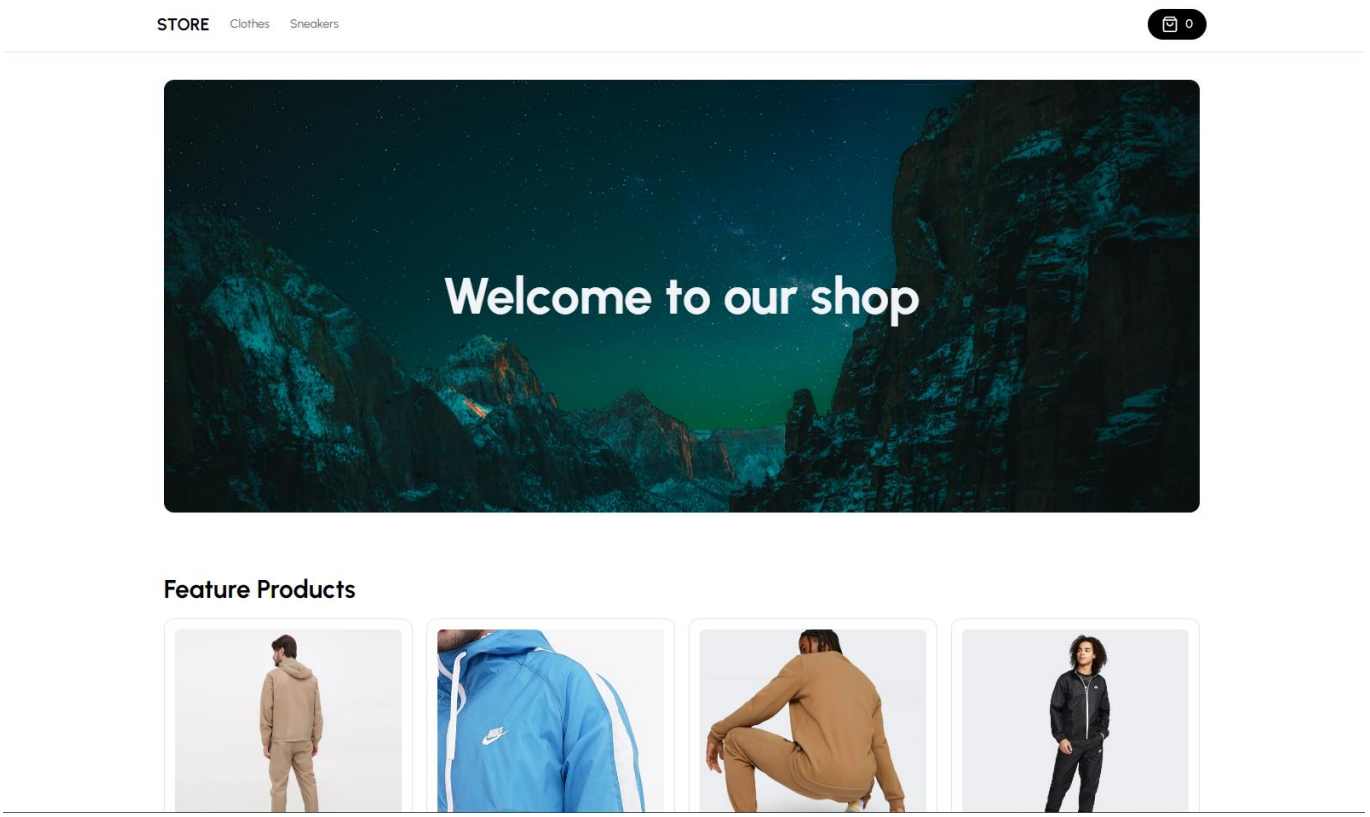


Рисунок 3.20 – Головна сторінка інтернет-магазину

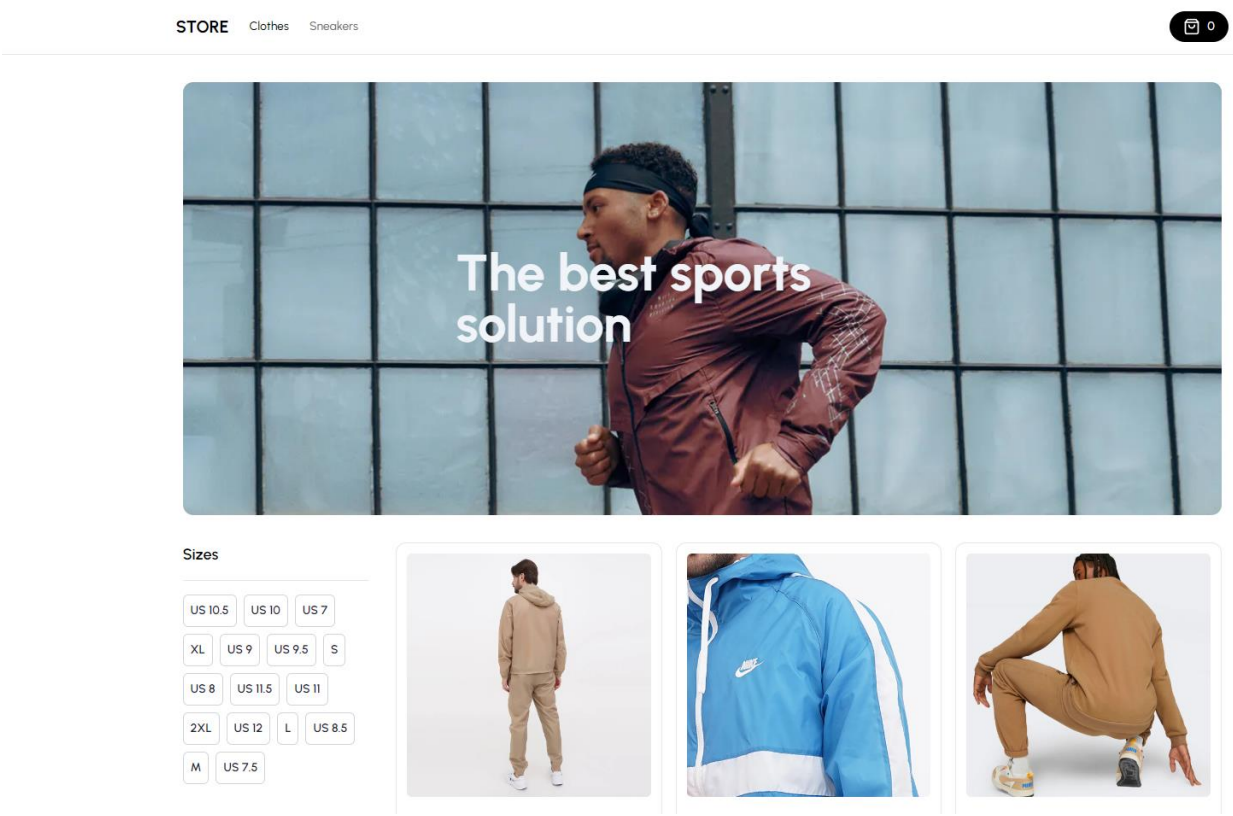


Рисунок 3.21 – Сторінка по категорії інтернет-магазину

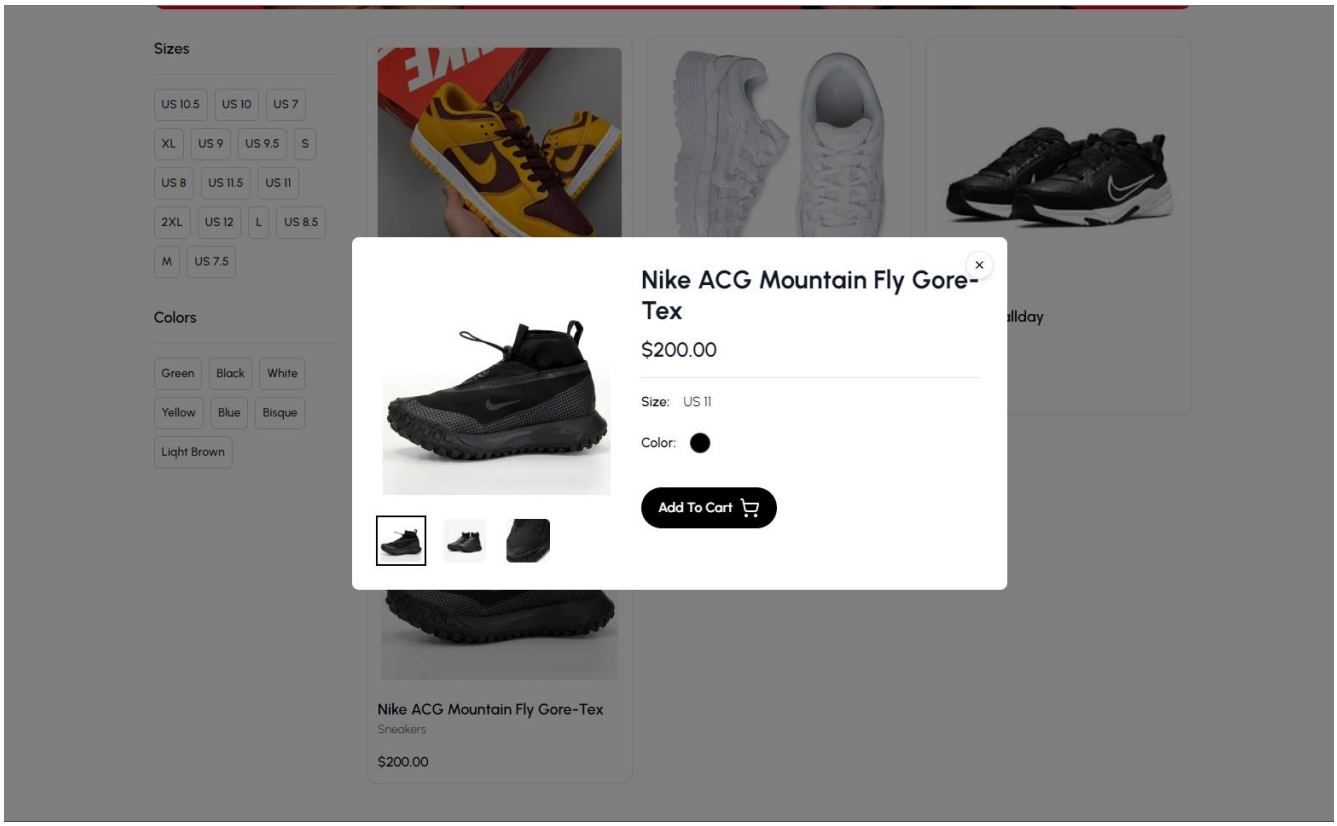


Рисунок 3.22 – Модальне вікно швидкого огляду товару

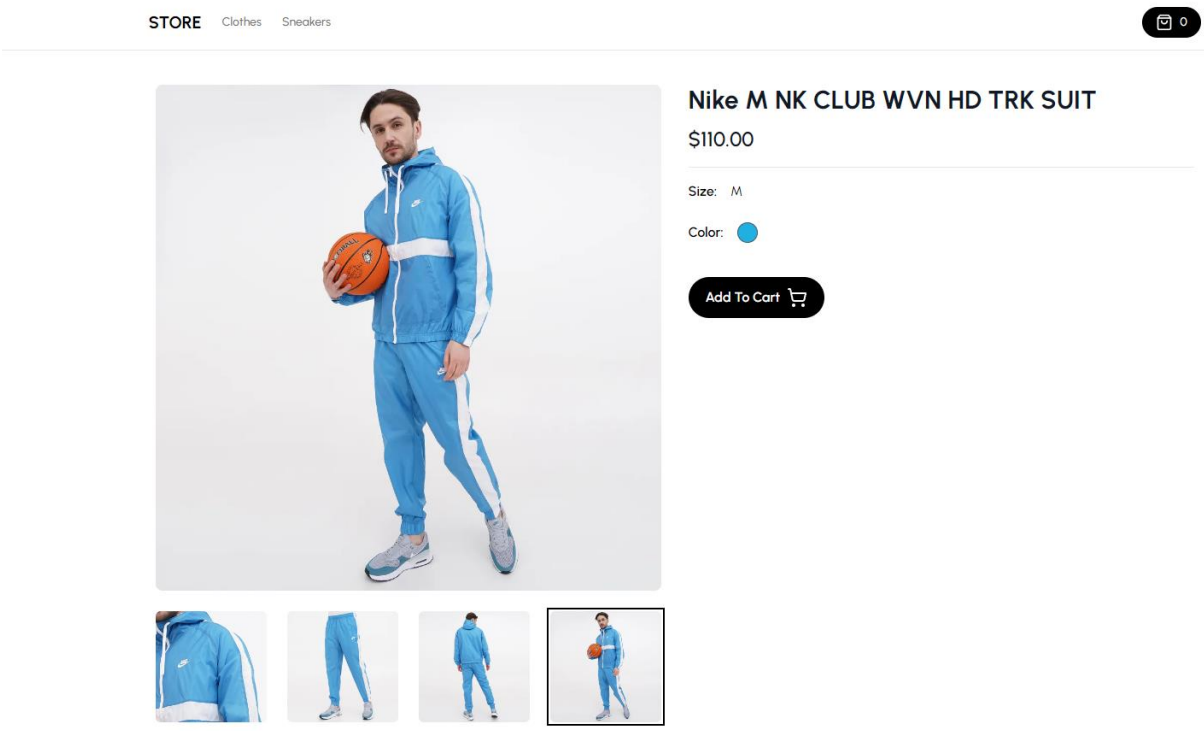


Рисунок 3.23 – Сторінка конкретного товару

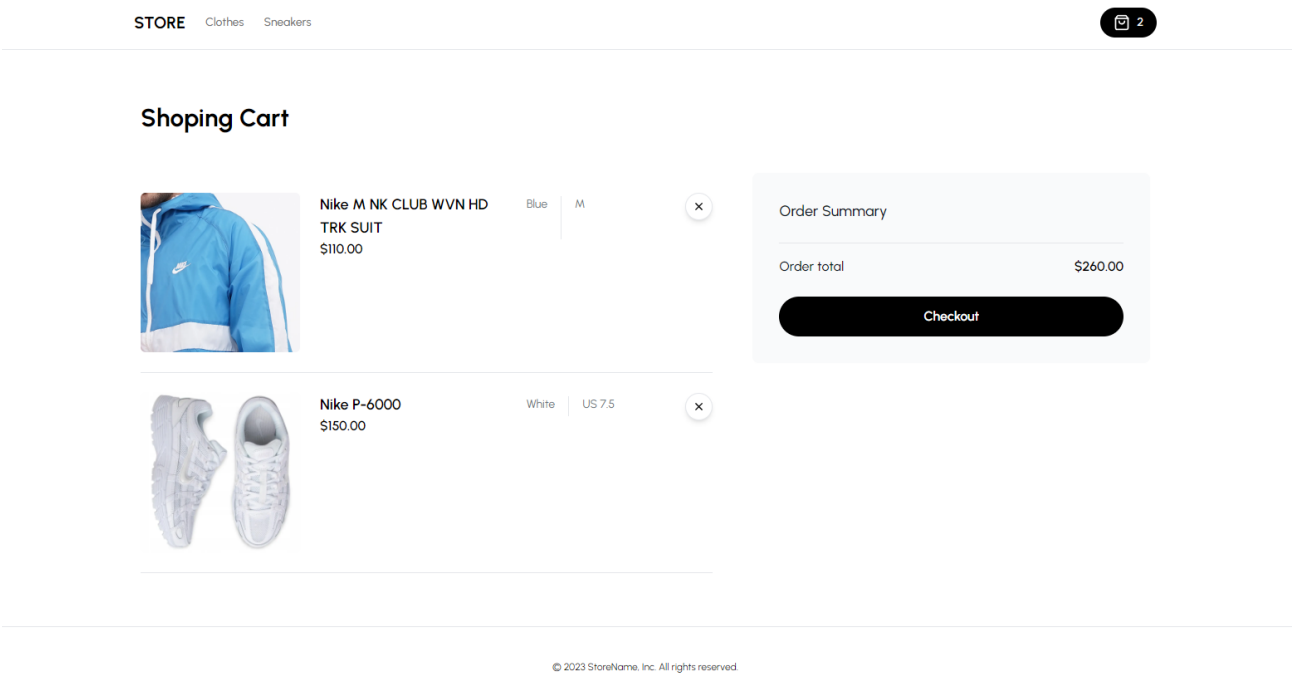


Рисунок 3.24 – Сторінка корзини

Наступним кроком тестування буде перевірка оплати товарів, потрапити

на яку ми можемо натиснувши кнопку «Checkout» на сторінці корзини. На рисунку 3.25 зображена сторінка оплати товарів що були додані у корзину.

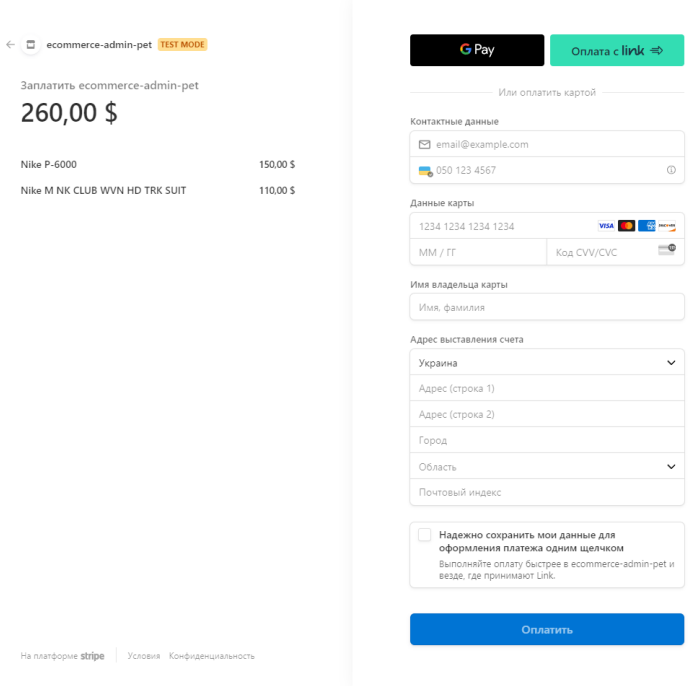


Рисунок 3.25 – Сторніка оплати товару за домопогою платформи Stripe

Заповнимо усі поля тестовими даними для перевіруи роботи, та оплатимо нашу покупку. Після успішного завершення покупки ми повертаємось до нашої корзини та бачимо, що вона пуста, це означає що ми все виконали правильно.

Перейдемо до нашої панелі адмістратору, та бачимо оновлену інформацію на домашній сторінці, а саме: наш загальний прибуток збільшився, кількість продажів збільшилась, зменшилась кількість доступних одиниць товару(оскільки ми купили 2 позиції), також графік продажів теж змінився, це також означає, що написаний нами вебхук для Stripe теж відпрацював.

На рисунку 3.26 зображена оновлена домашня сторінка панелі адміністратора.

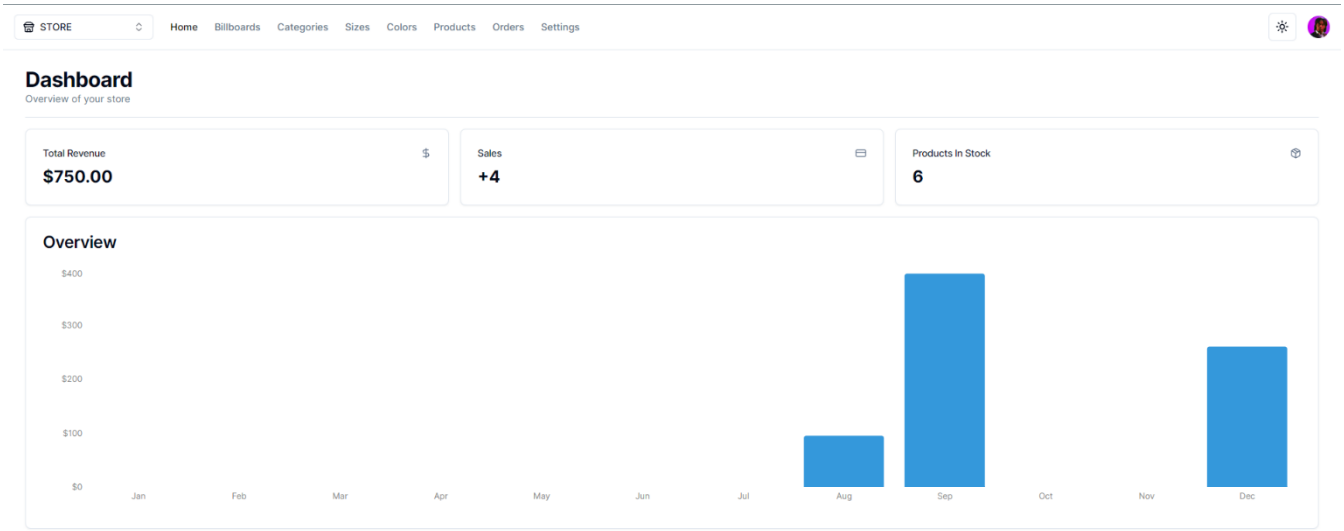


Рисунок 3.26 – Оновлена статистика магазину на домашній сторінці CMS-системи

Також оновився список продажів, це ми можемо побачити зрівнявши знімки інтефейсів на початку тестування та після здійснення покупки в інтернет магазині, з’явилося нове поле про успішно оплату товарів на першій позиції, оскільки підчас розробкипередбачалось сортування за принципом – «Спочатку нові», результат зображено на наступному рисунку – 3.27.

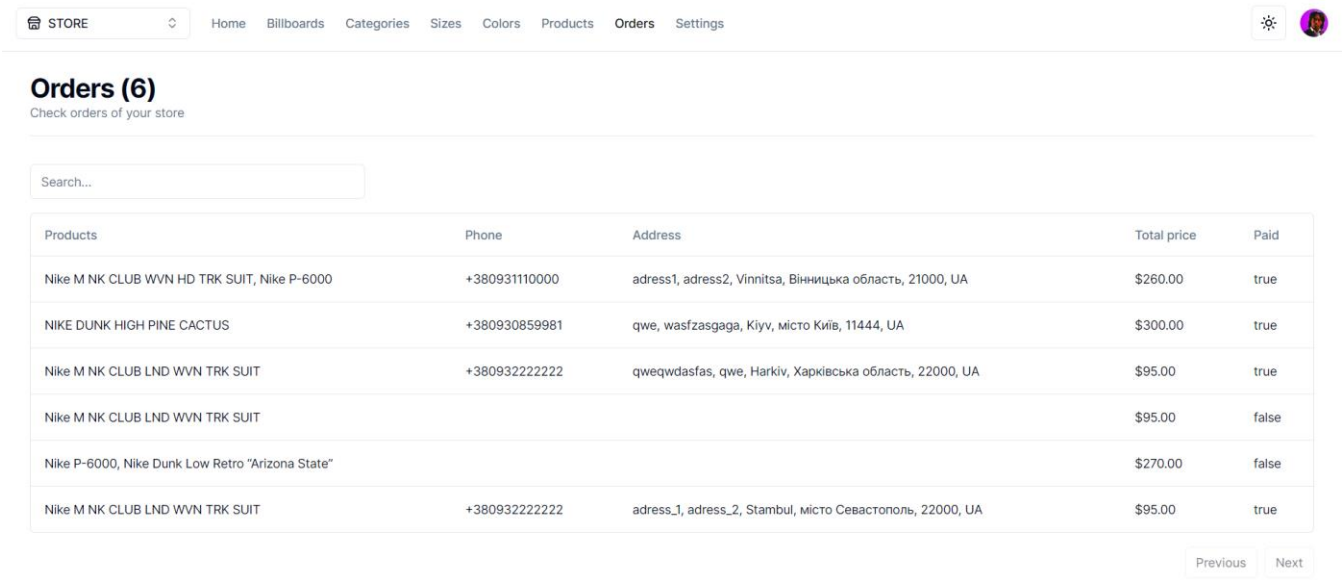


Рисунок 3.27 – Список продажів

На даному етапі тестування наших додатків можна завершити. Завдяки тестуванню ми змогли виявити деяки невелики помилки в користувацькому інтерфейсі та виправити їх, також дало нам змогу перевірити весь не маленький функціонал в розроблених нами дадатках. Тестування успішно завершено.

Отже, підбиваючи підсумки, даний розділ відображає фазу практичної реалізації нашого проекту, включаючи налаштування середовища розробки, створення бази даних, розробку клієнтського інтерфейсу та функціоналу для системи управління контентом (CMS) та інтернет-магазину, а також огляд і тестування розроблених додатків.

Середовище розробки та система контролю версій. Використання VS Code сприяло зручності роботи команди та ефективному управлінню проектом через систему контролю версій Git з віддаленим репозиторієм на Github.

Ініціалізація проекту, впровадження авторизації користувача. Процес ініціалізації проекту був успішно виконаний, а впровадження системи авторизації забезпечило безпеку та дозволило ефективно контролювати доступ до функціоналу.

Архітектура бази даних, розробка та підключення. Розроблена архітектура бази даних відповідає вимогам проекту, забезпечуючи надійне

зберігання та ефективний доступ до даних. Підключення бази даних інтегровано у програмний продукт без проблем.

Розробка клієнтського інтерфейсу CMS-системи. Клієнтський інтерфейс CMS-системи розроблено з урахуванням зручності користувача та його потреб у зборі, редагуванні та управлінні контентом.

Розробка інтернет-магазину. Розробка інтернет-магазину включає в себе повноцінний функціонал, дозволяючи додавати та керувати товарами, вибирати категорії та ефективно обробляти оплату, забезпечуючи користувачам зручний та безпечний досвід покупок.

Результати цього розділу свідчать про успішну реалізацію програмного забезпечення, враховуючи всі аспекти від ініціалізації проекту до тестування та оптимізації розроблених додатків.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення науково-технологічного аудиту розробки

Метою цього розділу є проведення технологічного аудиту для розробки автоматизованої системи налагодження роботи організаційних процесів у контексті інтернет-магазину. У даному випадку система спрямована на поліпшення управління контентом та оптимізацію рутинних операцій у магазині.

Аналогами розробленої CMS системи можуть бути наступні можуть бути такі CMS e-commerce системи як: Shopify, Magento та Opencart. Ціни аналогів:

– Shopify – поділяється на декілька планів:
  - Basic (for individuals & small businesses) plan: 32$/month – 1185грн/міс.
  - Shopify (for small businesses): 92$/month – 3404 грн/міс.
  - Advanced (for medium to large businesses): 399$/month – 14763/міс.
– Magento:
  - Magento Open Source: безкоштовно
  - Magento Commerce: 1988$/month – 73556грн/міс.
– OpenCart – платформа сама по собі безкоштовна, але додаткова оплата йде за обрані інструменти, або за додаткове сховище, ціни за інстурменти можуть варіюватись від 5$ – 100$

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за п’ятибальною шкалою)					
	0	1	2	3	4
1	*2*	*3*	*4*	*5*	*6*
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорів- нює цінам анало- гів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивос- ті продукту на рівні аналогів	Технічні та споживчі влас- тивості продук- ту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатацій ні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною ди- намікою	Великий стабі- льний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незна- чне навчання фахівців та збільшення їх штату	Необхідне незначне навчан- ня фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

1	*2*	*3*	*4*	*5*	*6*
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовують ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реаліза- ції ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5- ти років. Термін окупності інвес- тицій більший 5- ти років	Термін реаліза- ції ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років
12	Необхідна розробка регламентних докумен- тів та отриман- ня великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отри- мання дозвільних документів для виробництва та реалізації проду- кту потребує незначних коштів та часу	Необхідно тіль- ки повідомлення відповідним органам про виробництво та реалізацію про- дукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію про- дукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	4	4
2. Ринкові переваги (наявність аналогів)	4	3	3
3. Ринкові переваги (ціна продукту)	3	4	3

Продовження таблиці 4.2

4. Ринкові переваги (технічні властивості)	3	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	4
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	4	5	3
8. Практична здійсненність (наявність фахівців)	3	3	4
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	2	3	3
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	4
Сума балів	40	43	42
Середньоарифметична сума балів $СБ_c$	(40+43+42)/3 = 42		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового застосунку є високою, що досягається за рахунок того, що дана розробка не потребує великих затрат для її розробки та в подальшому підтримки. Також являється досить гнучким, це видно з огляду функціональності додатку, а саме в можливості управління одразу декількома магазинами, що значно збільшить майбутній дохід, в порівнянні з його аналогами, навіть в деяких функціях обганяючи їх, також в економії часу на розробку завдяки використанню нових технологій та великому ком'юніті, що в свою чергу дає велику кількості блогів, статей.

4.2 Розрахунок витрат на здійснення розробки

4.2.1 Основна заробітна плата дослідників, яка розраховується за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} * t_i}{T_p} \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p=*21…23* дні. Проведені розрахунки занесено до таблиці – 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	20000	870	28	23350
Програміст	25000	1080	28	30430
Всього				53780

4.2.2 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_р = \sum_{i=1}^{n} C_i * t_i \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_М * K_i * K_c}{T_р * t_{зм}} \quad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімаль

ної місячної заробітної плати (залежно від діючого законодавства), – 6700 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б);

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мініма- ної заробітної плати. (табл. Б.1, додаток Б)

$Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р = 21…23$ *дні*;

$t_{зм}$ – тривалість зміни, год.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
CMS	168	6	2,0	120	20160
E-commerce platform	56	3	1,35	82	4592
Всього					24752

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.3 Додаткова заробітна плата дослідників та робітників

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$$З_{дод} = (53780 + 22752) * \frac{10\%}{100\%} = 7653{,}2(грн.)$$

4.2.4 Відрахування та нарахування на заробітну плату дослідників та робітників

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = \left(З_о + З_р + З_{дод}\right) * \frac{Н_{зп}}{100\%} \quad (4.5)$$

$$З_н = (53780 + 22752 + 7653) * \frac{22\%}{100\%} = 18522 \text{ (грн.)}$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

4.2.5 Розрахунок витрат на матеріали та комплектуючі

Оскільки для розроблювального програмного засобу не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.6 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, апаратів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До

балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_i * С_{пр.i} * K_i \quad (4.6)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$С_{пр.i}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (Кі = 1,10…1,12);

k – кількість найменувань устаткування. Отримані результати занесено до таблиці – 4.6.

Таблиця 4.6 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Монітор Samsung G5	3	8500	28050
Клавіатура DarkProject	2	2000	4400
Комп'ютерна миша Logitech g105	2	1050	2310
Всього			34760

4.2.7 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} * С_{прг.і} * К_i \quad (4.7)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{прг.і}$. – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10…1,12);

k – кількість найменувань програмних засобів.

Отримані результати занесено до таблиці – 4.7.

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Система контролю БД – PlanetScale	1	1656	1821,6
TailwindCSS	1	4520	4972
Всього			6793,6

4.2.8 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Розрахуємо, для прикладу, амортизаційні витрати на комп’ютер балансова

вартість якого становить 35000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 2 міс.

$$А_{обл} = \frac{35000}{3} * \frac{2}{12} = 1936 \text{ (грн)}$$

Аналогічно визначаємо амортизаційні витрати на додаткове програмне забезпечення, такі як: ліцензійний ключ Windows 11, хмарні сервіси, сервіси обсулоговування, також розраховуєио інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.8.

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп`ютер	35000	3	2	1936
Додаткове програмне забезпечення	20000	2	2	1660
Офісне обладнення	20000	5	2	664
Приміщення	850000	15	2	9406,6
Всього				13666,6

4.2.9 Витрати на електро енергію

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень.

Витрати на силову електроенергію (Ве) розраховують за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} * t_i * Ц_е * K_{впі}}{\eta_і} \quad (4.9)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії

визначається за даними енергопостачальної компанії);

$K_{\text{впі}}$ – коефіцієнт, що враховує використання потужності, $K_{\text{впі}} < 1$;

$\eta_{\text{і}}$ – коефіцієнт корисної дії обладнання, ηі < 1;

Проведені розрахунки занесено до таблиці.

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер	0,45	214	762,3
Офісні прилади	0,20	80	26,6
Модем маршрутизатор TP-LINK TL-WR840N	0,05	120	47,5
Всього			836,4

4.2.10 Службові відрядження

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%} \quad (4.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = (53780 + 24752) * \frac{25\%}{100\%} = 19633 \text{ (грн)}$$

4.2.11 Витрати на роботи, які виконують сторонні підприємства, установи

і організації

До статті «Витрати на роботи, які виконують сторонні підприємства, установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{нсп}}{100\%} \quad (4.11)$$

де $Н_{нсп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = (53780 + 24752) * \frac{40\%}{100\%} = 31412{,}8 \text{ (грн)}$$

4.2.12 Інші виробничі витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.12)$$

де $Н_{ів}$– норма нарахування за статтею «Інші витрати».

$$I_в = (53780 + 24572) * \frac{50\%}{100\%} = 39176 \text{ (грн)}$$

4.2.13 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (53780 + 24752) * \frac{100\%}{100\%} = 78535 \text{ (грн)}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_є + В_{сп} + I_в + В_{нзв} \quad (4.14)$$

$$В_{заг} = 53780 + 24752 + 7653,2 + 18522 + 10290 + 13666,6 + 836,6 + 31412,8 + 39176 + 78535 = 278624,2 \text{ (грн)}$$

Загальні витрати на завершення науково-дослідної (науково-технічної)

роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.15)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9.

Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка.

$$ЗВ = \frac{278624,2}{0,5} = 557248,4 \text{ (грн)}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інфестором.

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати

науковотехнічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науковотехнічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту; г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_{\text{і}} = (\pm \Delta Ц_{\text{о}} * N * Ц_{\text{о}} * \Delta N)_i * \lambda * \rho * (1 - \frac{\vartheta}{100}) \qquad (4.16)$$

де $\pm \Delta Ц_{\text{о}}$– зміна вартості програмного продукту (зростання чи зниження)

від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$; $Ц_б$

$Ц_б$– вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість.

Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні підриски на наш розроблений інтернет ресурс, а саме початковий план у 500 грн. за місяць користування, термін збільшення прибутку складе 3 роки. Після завершення розробки і вдосконалення системи, можна буде додати додаткові плани, піднявши ціну на 200грн. Протягом першого року кількість користувачів буде – на 10000, протягом другого року – на 30000, протягом третього року 50000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0 * 200 + (500 + 200) * 10000) * 0{,}8333 * 0{,}5 * (1 - 0.18) = 2391571 \text{ (грн)}$$

$$\Delta П_2 = (0 * 200 + (500 + 200) * (10000 + 30000)) * 0{,}8333 * 0{,}5 * (1 - 0.18) = 9566284 \text{ (грн)}$$

$$\Delta П_3 = (0 * 200 + (500 + 200) * (10000 + 30000 + 50000)) * 0{,}8333 * 0{,}5 * (1 - 0.18) = 21524139 \text{ (грн)}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 33481994 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^i} \quad (4.17)$$

де ΔПі – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науковотехнічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Збільшення прибутку ми отримаємо починаючи з першого року:

$$ПП = \left(\frac{2391571}{(1+0{,}1)^1}\right) + \left(\frac{9566284}{(1+0{,}1)^2}\right) + \left(\frac{21524139}{(1+0{,}1)^3}\right) = 2174155{,}45 + 9378709{,}8 + 20897222{,}33 = 32450087{,}6 \text{ (грн)}$$

Далі розраховують величину початкових інвестицій PV, які потенційний

інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} * \text{ЗВ} \quad (4.18)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 557248{,}4 = 1114496{,}8 \text{ (грн)}$$

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV \quad (4.19)$$

$$E_{\text{абс}} = 32450087{,}6 - 1114496{,}8 = 31335590{,}8 \text{ (грн)}$$

Оскільки $\text{E}_{\text{абс}} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи є доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність $\text{E}_{\text{в}}$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науковотехнічної розробки, розраховується за формулою:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 \quad (4.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

Тж – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[з]{1 + \frac{31335590,8}{1114496,8}} - 1 = 3,07$$

Далі визначають бар'єрну ставку дисконтування мін τ_{min}, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науковотехнічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій мін τ визначається за формулою:

$$\tau_{min} = d + f \quad (4.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = 0,9...0,12;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

$$\tau_{min} = 0,12 + 0,05 = 0,17$$

Так як Ев > τmin, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховуємо період окупності інвестицій Ток (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \quad (4.22)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{3{,}07} = 0{,}33 \text{ (р.)}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,33 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу

Висновки до економічного розділу магістерської кваліфікаційної роботи генерують важливий контекст для оцінки фінансової стійкості та ефективності розробленого програмного продукту. Зокрема:

Витрати на розробку: Розраховані витрати на розробку нового програмного продукту становлять 557248,4 гривень. Це включає прогнозовані витрати за кожною зі статей витрат, що є ключовою інформацією для управління фінансами та планування подальших етапів роботи.

Чистий прибуток: Розраховано чистий прибуток, який виробник може отримати від реалізації нового програмного продукту. Це дозволяє визначити фінансовий виграш, який можна очікувати внаслідок впровадження даного технічного рішення.

Період окупності: Обчислений період окупності становить близько 0,33 роки, що свідчить про те, що інвестиції в розробку нового продукту повернуться

вкрай швидко. Це важливий показник ефективності проекту та привабливості для потенційних інвесторів.

Економічний ефект: Розраховано абсолютний економічний ефект при використанні розробленого продукту, що становить 31335590,8 грн. Цей показник дозволяє оцінити практичну користь продукту та його вплив на фінансовий результат.

Зазначені висновки надають повний образ економічного аспекту розробленого програмного продукту, враховуючи витрати, прибуток, окупність і загальний економічний виграш. Це важливий внесок у розуміння фінансової стійкості та ефективності проекту, що може бути корисним для різних стейкхолдерів, включаючи інвесторів, розробників та управлінський персонал.

ВИСНОВКИ

Магістерська робота "Розробка автоматизованої системи організації бізнес-процесів Інтернет-магазину. Розробка CMS-системи є результатом глибоких досліджень, розробки та впровадження програмного забезпечення, спрямованого на підвищення ефективності та управління Інтернет-магазином.

Процес розробки включав вибір технологічного стеку, ініціацію проекту, розробку бази даних, створення клієнтського інтерфейсу для системи CMS та інтернет-магазину, а також огляд і тестування розроблених додатків.

Серед основних результатів – успішне впровадження системи авторизації, впровадження функціоналу для управління товарами та контентом, а також створення ефективної системи обробки платежів. Тестування додатків виявило деякі недоліки, але загальний стан системи відповідає вимогам і високим стандартам якості.

Магістерська робота вирішує актуальні задачі автоматизації та управління процесами в інтернет-магазині, пропонуючи ефективне програмне рішення. Отримані результати є важливим кроком на шляху розвитку сучасних технологій електронної комерції.

Майбутні перспективи включають подальші дослідження та вдосконалення системи з урахуванням зростаючих потреб ринку. Можливості розширення функціональності, оптимізації продуктивності та впровадження нових технологій роблять цей проект основою для подальших інновацій у сфері електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CityHost – “CMS – системи, повна характеристика” 2023 [Електронний ресурс] – Режим доступу: https://cityhost.ua/uk/blog/chto-takoe-cms-i-kakie-vidy-cms-dlya-saytov-byvayut.html
2. W3tech – Статистика популярності усіх відомих CMS [Електронний ресурс] – Режим доступу: https://w3techs.com/technologies/overview/content_management
3. LiveAbout – ”Defining the Different Types of E-Commerce Businesses” [Електронний ресурс] – Режим доступу: https://www.liveabout.com/ecommerce-businesses-understanding-types-1141595
4. UA Reatail – Стаття «10 основних тенденцій електронної комерції на 2021 рік. Інфографіка» [Електронний ресурс] – Режим доступу: https://ua-retail.com/2021/06/10-osnovnix-tendencij-elektronno%D1%97-komerci%D1%97-na-2021-rik-infografika/
5. В. БРИЖКО – «Електронна комерція – головні аспекти електронної торгівлі» [Електронний ресурс] – Режим доступу: https://ippi.org.ua/sites/default/files/08bvusv.pdf
6. Roland Li – «2nd most valuable U.S. startup to leave SF as city loses another headquarters» [Електронний ресурс] – Режим доступу: https://www.sfchronicle.com/business/article/2nd-most-valuable-U-S-startup-to-leave-SF-as-14558067.php
7. Максим Дімура – “Створення інтернет-магазину за всіма правилами e-commerce у 2023 році” [Електронний ресурс] – Режим доступу: https://www.site2b.ua/ua/web-blog-ua/stvorennya-internet-magazinu-za-vsima-pravilami-e-commerce-u-2023-roci.html
8. Jhon Bristowe – “What is a Hybrid Mobile App?” [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20161204050511/http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/
9. Ketie Terrell Hanna – “What is Google Firebase” [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchmobilecomputing/definition/Google-

Firebase

10. E. Rescorla. «HTTPS characteristics» [Електронний ресурс] – Режим доступу: https://www.rfc-editor.org/rfc/rfc2818.html
11. Wezom.com «Які проблеми вирішує технологія створення прогресивниз веб-додатків» [Електронний ресурс] – Режим доступу: https://wezom.com.ua/ua/blog/pwa-prilozheniya-web-progressive-app
12. Walls, Colin – «Embedded software» [Електронний ресурс] – Режим доступу: https://books.google.com.ua/books?id=FLvsis4_QhEC&pg=PA344&redir_esc=y#v=onepage&q&f=false
13. ADWServices – «Огляд Shopify CMS: плюси та мінуси платформи» [Електронний ресурс] – Режим доступу: https://adwservice.com.ua/uk/ogljad-shopify-cms
14. ADWServices – «Налаштування середовища взаємодії з магазином на Shopify» [Електронний ресурс] – Режим доступу: https://adwservice.com.ua/uk/nalashtuvannja-kontekstnoji-reklamy-na-shopify
15. WebNaunts – «Shopify — що краще для SEO та електронної комерції» [Електронний ресурс] – Режим доступу: https://webnauts.pro/uk/blog/wordpress-vs-shopify-scho-krasche/
16. HostIQ – «Що означає CMS в теперішніх реаліях» [Електронний ресурс] – Режим доступу: https://hostiq.ua/wiki/ukr/cms/
17. Shopify.dev – official documentation [Електронний ресурс] – Режим доступу: https://shopify.dev/docs
18. Platon.ua – “Magento Commerce VS. Magento Open Source” [Електронний ресурс] – Режим доступу: https://platon.ua/news/magento-commerce-vs-magento-open-source.html
19. Mari Skula – “Інструкція по створенню та підтримки інтернет магазину “ [Електронний ресурс] – Режим доступу: https://magefan.com/ua/blog/stvorennia-onlajn-mahazynu-na-magento
20. Adobe – “Adobe Commerce Developer Documentation” [Електронний ресурс] – Режим доступу: https://developer.adobe.com/commerce/docs/

21. Interkassa – “CMS Opencart: плюси та мінуси, огляд функціоналу” [Електронний ресурс] – Режим доступу: https://interkassa.com/blog/cms-opencart-plyusi-ta-minusi-oglyad-funkcionalu
22. OceanAgency – “Аналітичний огляд CMS Opencart” [Електронний ресурс] – Режим доступу: https://ocean-agency.com.ua/ua/vse-pro-cms-opencart
23. OpenCart official documentation [Електронний ресурс] – Режим доступу: https://docs.opencart.com/en-gb/introduction/
24. “React Up and Running” авторства Stoyan Stefanov та Joe Lencioni.
25. React.dev – “React Official Documentation” [Електронний ресурс] – Режим доступу: https://react.dev/
26. Vercel – NextJS Documentation [Електронний ресурс] – Режим доступу: https://nextjs.org/docs
27. Vercel – Посібник по вивченню фреймворку NextJS для початківців [Електронний ресурс] – Режим доступу: https://nextjs.org/learn/foundations/about-nextjs?utm_source=next-site&utm_medium=homepage-cta&utm_campaign=home
28. DPU – “CSR, SSR, SSG: типи рендерингу та який з них краще використовувати” [Електронний ресурс] – Режим доступу: https://dou.ua/forums/topic/31720/
29. Dere, Mohan «How to integrate create-react-app with all the libraries you need to make a great app» (2017-12-21) [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/how-to-make-create-react-app-work-with-a-node-backend-api-7c5c48acb1b0
30. Bright, Peter (3 October 2012). "Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem?" Ars Technica. Condé Nast. Retrieved 26 April 2015 [Електронний ресурс] – Режим доступу: https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/

31. Foley, Mary Jo (1 October 2012). [Електронний ресурс] – Режим доступу: "Microsoft takes the wraps off TypeScript, a superset of JavaScript". ZDNet. CBS Interactive. Retrieved 26 April 2015.
32. Jackson, Joab (1 October 2012). "Microsoft Augments Javascript for Large-scale Development". CIO. IDG Enterprise. Archived from the original on 17 December 2013. Retrieved 26 April 2015. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20131217223751/http:/www.cio.com/article/717679/Microsoft_Augments_Javascript_for_Large_scale_Development
33. Bright, Peter (22 September 2016). "TypeScript, Microsoft's JavaScript for big applications, reaches version 2.0". Ars Technica. Condé Nast. [Електронний ресурс] – Режим доступу: https://arstechnica.com/information-technology/2016/09/typescript-microsofts-javascript-for-big-applications-reaches-version-2-0/
34. TypeScriptLang – "Електронний посібник – TypeScriptLang" [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html
35. FreeCodeCamp – "Learn TypeScript – The Ultimate Beginners Guide" [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/learn-typescript-beginners-guide/
36. What is a type in TypeScript [Електронний ресурс] – Режим доступуhttps://www.typescripttutorial.net/typescript-tutorial/typescript-types/
37. Prisma – "Prisma official documentation" [Електронний ресурс] – Режим доступу: https://www.prisma.io/docs/concepts
38. Prisma github [Електронний ресурс] – Режим доступу: https://github.com/prisma/prisma
39. MySQL – "Офіційна документація MySQL" [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
40. Документація по управлінню базами даних в PlanetScale [Електронний ресурс] – Режим доступу: https://planetscale.com/docs

41. ChangeLog github “TailwindCSS ver.” 13 July 2023. Retrieved 26 July 2023. [Електронний ресурс] – Режим доступу: https://github.com/tailwindlabs/tailwindcss/releases/tag/v3.3.3
42. [Електронний ресурс] – Режим доступу: "Github: tailwindlabs/tailwindcss, LICENSE"
43. Gerchev, Ivaylo (2022). “Talwind CSS is simple”. Sebastopol: O'Reilly [Електронний ресурс] – Режим доступу: https://search.worldcat.org/title/1314257318
44. TaiwindCSS – "Utility-First - Tailwind CSS". [Електронний ресурс] – Режим доступу: https://tailwindcss.com/docs/utility-first
45. TaiwindCSS – "Responsive Design - Tailwind CSS". Retrieved 2021-11-13. [Електронний ресурс] – Режим доступу: https://tailwindcss.com/docs/responsive-design
46. "Hover, Focus, & Other States - Tailwind CSS". tailwindcss.com. Retrieved 2021-11-13. [Електронний ресурс] – Режим доступу: https://tailwindcss.com/docs/hover-focus-and-other-states
47. Official Clerk Documentation [Електронний ресурс] – Режим доступу: https://clerk.com/docs
48. Official Stripe Documentation [Електронний ресурс] – Режим доступу: https://stripe.com/docs
49. Stripe company – “DashBoard” [Електронний ресурс] – Режим доступу: Stripe: Press resources stripe.com.
50. Li, Roland 2nd most valuable U.S. startup to leave SF as city loses another headquarters San Francisco Chronicle (24 октября 2019). [Електронний ресурс] – Режим доступу: https://www.sfchronicle.com/business/article/2nd-most-valuable-U-S-startup-to-leave-SF-as-14558067.php
51. Rao, Leena Payments Company Stripe Makes First Acquisition, Buys Team Task Management And Collaboration App Kickoff. TechCrunch [Електронний ресурс] – Режим доступу: https://techcrunch.com/2013/03/11/payments-company-stripe-acquires-team-task-management-and-collaboration-app-kickoff/

52. Microsoft – Official Documentation Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/docs
53. GIT – “Official GIT Documentation” [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
54. GitHub – “Official GitHub Documentation” [Електронний ресурс] – Режим доступу: https://docs.github.com/en
55. Mozilla.org – “JavaScript shells” [Електронний ресурс] – Режим доступу: https://www-archive.mozilla.org/rhino/shell
56. NPMJS – “Npm typescript” [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/typescript?activeTab=versions
57. "Eloquent JavaScript" авторства Marijn Haverbeke.
58. "You Don't Know JS" серія книг від Kyle Simpson.
59. NNGgroup – “Статті та рекомендації щодо дизайну і взаємодії користувача” [Електронний ресурс] – Режим доступу: https://www.nngroup.com/
60. Google – Chrome developer tools Documentation [Електронний ресурс] – Режим доступу: https://developer.chrome.com/docs/devtools/command-menu?hl=en

ДОДАТКИ

Додаток А

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **« Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину»**

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93,2% Схожість 6,8%

Аналіз звіту подібності (відмітити потрібне)

ᵛ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Богдан САВІН
(підпис) (прізвище, ініціали)

Керівник роботи _____ Марія ЮХИМЧУК
(підпис) (прізвище, ініціали)

Додаток Б

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о.Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
" 06 " жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину

(тема)

08-33.МКР.17.00.000 ТЗ
номер

Студент групи 2АКІТ-22м

_____ Богдан САВІН
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник д.т.н., доцент, т.в.о.зав. кафедри КСУ

_____ Марія ЮХИМЧУК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – автоматизованої системи налагодження роботи організаційних процесів інтернет магазину.

1.2. Галузь застосування – електронна комерція.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від “18” вересня 2023 року №247

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є розробка автоматизованої системи управління інтернет-магазином, яка охоплює весь цикл від додавання товарів до відстеження оплати та аналізу продажів.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Офіційна документація по NextJS [Електронний ресурс] – Режим доступу: https://nextjs.org/docs

2. Learn TypeScript – The Ultimate Beginners Guide [Електронний ресурс] – Режим доступу: https://www.freecodecamp.org/news/learn-typescript-beginners-guide/

3. Prisma github [Електронний ресурс] – Режим доступу: https://github.com/prisma/prisma

4. Документація по управлінню базами даних в PlanetScale [Електронний ресурс] – Режим доступу: https://planetscale.com/docs

5. "Responsive Design - Tailwind CSS". tailwindcss.com. Retrieved 2021-11-13. [Електронний ресурс] – Режим доступу: https://tailwindcss.com/docs/responsive-design

6. Stripe docs [Електронний ресурс] – Режим доступу: https://stripe.com/docs

5. Вимоги до розробки.

5.1. Перелік головних функцій:

– Додавання, видалення, зміна контенту інтернет магазину за допомогою розробленої CMS

– Здійснення керування одразу декількома магазинами

– Можливість повної взаємодії користувача з товаром представленим в інтернет магазині, а саме: фільтрація товару, додавання в корзину, покупка товару

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10/11;
- Visual Studio Code.
- Доступ до мережі інтернет;
- Акаунт Gmail, або будь якої іншої електронної пошти.

5.2.2. Умови експлуатації системи:

- робота на мобільних пристроях та веб-браузерах;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «29» 09 2023 р.
2. Аналіз та методи вирішення проблем електронної комерції «01» 10 2023 р.
3. Огляд аналогів існуючих автоматизованих систем управління контентом «12» 10 2023 р.
4. Аналіз та вибір технологічного стеку на технологій для розробки автоматизованої та інтернет-магазину «25» 10 2023 р.

5. Розробка програмного забезпечення «10» 11 2023 р.

6.2 Графічні матеріали:

1. Розробка моделі бази даних системи «26» 11 2023 р.
2. Розробка UML-діаграм системи «27» 11 2023 р.
3. Тестування програмного забезпечення «28» 11 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «11» грудня 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «11» грудня 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «20» грудня 2023 р.

Додаток В

Лістинг файлу “schema.prisma”

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider     = "mysql"
  url          = env("DATABASE_URL")
  relationMode = "prisma"
}

model Store {
  id         String @id @default(uuid())
  name       String
  userId     String
  billboards Billboard[] @relation("StoreToBillboard")
  categories Category[] @relation("StoreToCategory")
  sizes      Size[] @relation("StoreToSize")
  colors     Color[] @relation("StoreToColor")
  products   Product[] @relation("StoreToProduct")
  oreders    Order[] @relation("StoreToOrder")
  createdAt  DateTime @default(now())
  updatedAt  DateTime @updatedAt
}

model Billboard {
  id         String @id @default(uuid())
  storeId    String
  store      Store @relation("StoreToBillboard", fields:[storeId], references:
[id])
  label      String
  imageUrl   String
  categories Category[]
  createdAt  DateTime @default(now())
  updatedAt  DateTime @updatedAt

  @@index([storeId])
}

model Category {
  id            String @id @default(uuid())
  storeId       String
  store         Store @relation("StoreToCategory", fields: [storeId], references:
[id])
  billboardId   String
  billboard     Billboard @relation(fields: [billboardId], references: [id])
  products      Product[] @relation("CategoryToProduct")
  name          String
  createdAt     DateTime @default(now())
```

```
  updatedAt     DateTime @updatedAt

  @@index([storeId])
  @@index([billboardId])
}

model Size {
  id            String @id @default(uuid())
  storeId       String
  store         Store @relation("StoreToSize", fields: [storeId], references: [id])
  name          String
  value         String
  products      Product[]
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  @@index([storeId])
}

model Color {
  id            String @id @default(uuid())
  storeId       String
  store         Store @relation("StoreToColor", fields: [storeId], references:
[id])
  name          String
  value         String
  products      Product[]
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  @@index([storeId])
}

model Product {
  id            String @id @default(uuid())
  storeId       String
  store         Store @relation("StoreToProduct", fields: [storeId], references:
[id])
  categoryId    String
  category      Category @relation("CategoryToProduct", fields: [categoryId],
references: [id])
  name          String
  price         Decimal
  isFeatured    Boolean @default(false)
  isArchived    Boolean @default(false)
  sizeId        String
  size          Size @relation(fields: [sizeId], references: [id])
  colorId       String
  color         Color @relation(fields: [colorId], references: [id])
  images        Image[]
  orderItems    OrderItem[]
```

```
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  @@index([storeId])
  @@index([categoryId])
  @@index([colorId])
  @@index([sizeId])
}

model Image {
  id            String @id @default(uuid())
  productId     String
  product       Product @relation(fields: [productId], references: [id], onDelete:
Cascade)
  url           String
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  @@index([productId])
}

model Order {
  id            String @id @default(uuid())
  storeId       String
  Store         Store @relation("StoreToOrder",fields: [storeId], references: [id])
  orderItems    OrderItem[]
  isPaid        Boolean @default(false)
  phone         String @default("")
  address       String @default("")
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt

  @@index([storeId])
}

model OrderItem {
  id            String @id @default(uuid())
  orderId       String
  order         Order @relation(fields: [orderId], references: [id])
  productId     String
  product       Product @relation(fields: [productId], references: [id])

  @@index([orderId])
  @@index([productId])
}
```

Додаток В

Лістинг файлу “setings-form.tsx”

```
"use client";

import { Store } from "@prisma/client";
import { Trash } from "lucide-react";
import * as z from "zod";
import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
import { useState } from "react";
import axios from "axios";
import { useParams, useRouter } from "next/navigation";
import { toast } from "react-hot-toast";

import { Heading } from "@/components/ui/heading";
import { Button } from "@/components/ui/button";
import { Separator } from "@/components/ui/separator";
import { Input } from "@/components/ui/input";
import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "@/components/ui/form";
import { AlertModal } from "@/components/modals/alert-modal";
import { ApiAlert } from "@/components/ui/api-alert";
import { userOrigin } from "@/hooks/use-origin";

interface SettingsFormProps {
  initialData: Store;
}

const formSchema = z.object({
  name: z.string().min(1),
});

type SettingsFormValues = z.infer<typeof formSchema>;

export const SettingsForm: React.FC<SettingsFormProps> = ({ initialData }) => {
  const [open, setOpen] = useState(false);
  const [loading, setLoading] = useState(false);

  const params = useParams();
  const router = useRouter();

  const origin = userOrigin();

  const form = useForm<SettingsFormValues>({
```

```
  resolver: zodResolver(formSchema),
  defaultValues: initialData,
});

const onSubmit = async (data: SettingsFormValues) => {
  try {
    setLoading(true);

    await axios.patch(`/api/stores/${params.storeId}`, data);

    router.refresh();

    toast.success("Store updated");
  } catch (error) {
    toast.error("Something went wrong");
  } finally {
    setLoading(false);
  }
};

const onDelete = async () => {
  try {
    await axios.delete(`/api/stores/${params.storeId}`);

    router.refresh();
    router.push("/");

    toast.success("Store is deleted success");
  } catch (error) {
    toast.error("Make sure you removed all products and categories first.");
  } finally {
    setLoading(false);
    setOpen(false);
  }
};

return (
  <>
    <AlertModal
      isOpen={open}
      onClose={() => setOpen(false)}
      onConfirm={onDelete}
      loading={loading}
    />
    <div className="flex items-center justify-between">
      <Heading
        title="Store Settings"
        description="Manage store preferences"
      />
      <Button
        disabled={loading}
```

```
          variant="destructive"
          size="sm"
          onClick={() => setOpen(true)}
        >
          <Trash className="h-4 w-4" />
        </Button>
      </div>
      <Separator />
      <Form {...form}>
        <form
          onSubmit={form.handleSubmit(onSubmit)}
          className="w-full space-y-8"
        >
          <div className="grid grid-cols-3 gap-8">
            <FormField
              control={form.control}
              name="name"
              render={({ field }) => (
                <FormItem>
                  <FormLabel>Name</FormLabel>
                  <FormControl>
                    <Input
                      disabled={loading}
                      placeholder="Store name"
                      {...field}
                    />
                  </FormControl>
                  <FormMessage></FormMessage>
                </FormItem>
              )}
            />
          </div>
          <Button disabled={loading} className="ml-auto" type="submit">
            Save Changes
          </Button>
        </form>
      </Form>
      <Separator />
      <ApiAlert
        title="NEXT PUBLIC API URL"
        description={`${origin}/api/${params.storeId}`}
        variant="public"
      />
    </>
  );
};
```

Додаток В

Лістинг файлу “get-graph-revenue.ts”

```
import prismadb from "@/lib/prismadb";

interface graphData {
  name: string;
  total: number;
}

export const getGraphRevenue = async (storeId: string) => {
  const paidOrders = await prismadb.order.findMany({
    where: {
      storeId,
      isPaid: true,
    },
    include: {
      orderItems: {
        include: {
          product: true,
        },
      },
    },
  });

  const monthlyRevenue: { [key: number]: number } = {};

  for (const order of paidOrders) {
    const month = order.createdAt.getMonth();
    let revenueForOrder = 0;

    for (const item of order.orderItems) {
      revenueForOrder += item.product.price.toNumber();
    }

    monthlyRevenue[month] = (monthlyRevenue[month] || 0) + revenueForOrder;
  }

  const graphData: graphData[] = [
    { name: "Jan", total: 0 },
    { name: "Feb", total: 0 },
    { name: "Mar", total: 0 },
    { name: "Apr", total: 0 },
    { name: "May", total: 0 },
    { name: "Jun", total: 0 },
    { name: "Jul", total: 0 },
    { name: "Aug", total: 0 },
    { name: "Sep", total: 0 },
    { name: "Oct", total: 0 },
    { name: "Nov", total: 0 },
    { name: "Dec", total: 0 },
```

```
  ];

  for (const month in monthlyRevenue) {
    graphData[parseInt(month)].total = monthlyRevenue[parseInt(month)];
  }

  return graphData;
};
```

Додаток Г
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину

(тема)

1. Вступ
2. Мета, обєкт та предмет дослідження
3. Огляд та порівняння аналогів
4. Стек технологій
5. Послідовність дій при розробці
6. Діаграма моделей та зв'язків бази даних
7. Функціональність розробленої CMS
8. Структура інтерфейсу CMS-системи
9. UML-діаграма послідовності дій в інтернет-магазині
10. Структура інтерфейсу інтернет-магазину
11. Зовнішній вигляд розроблених додатків
12. Висновки

Студент групи 2АКІТ-22м

Підпис Богдан САВІН
Ім'я ПРІЗВИЩЕ

Керівник д.т.н., доцент, т.в.о.зав. кафедри КСУ

Підпис Марія ЮХИМЧУК
Ім'я ПРІЗВИЩЕ

“Розробка автоматизованої системи налагодження роботи організаційних процесів інтернет магазину”

Магістерська кваліфікаційна робота
Студента групи 2АКІТ-22м: Савіна Богдана Дмитровича

Керівник роботи: Юхимчук Марія Сергіївна
д.т.н., доцент, т.в.о.зав. кафедри КСУ

ВСТУП

2

Декілька слів про актуальність: у сучасному бізнес-середовищі інтернет-магазини стають основними гравцями на ринку електронної комерції. Зростання популярності онлайн-торгівлі спричиняє зростання конкуренції серед підприємств, які намагаються привернути і утримати клієнтів. Окрім того, споживачі стають більш вимогливими і очікують швидкого, зручного та персоналізованого обслуговування.

З ростом конкуренції в електронній комерції, важливо мати ефективний інструмент для управління та вдосконалення процесів в магазині. Проект спрямований на створення адаптивної системи, що дозволить власникам інтернет-магазинів ефективно керувати контентом, відстежувати статистику та оптимізувати роботу свого бізнесу.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

3

- МЕТА ДОСЛІДЖЕННЯ

Розробка автоматизованої системи управління інтернет-магазином, яка охоплює весь цикл від додавання товарів до відстеження оплати та аналізу продажів. Застосування сучасних технологій та інструментів робить цей проект актуальним і конкурентоспроможним на ринку електронної комерції.

- ОБ'ЄКТ ДОСЛІДЖЕННЯ

Інтернет-магазин та його організаційні процеси. Розроблена система спрямована на оптимізацію цих процесів, полегшення керування контентом та забезпечення ефективності бізнесу в електронній комерції.

- ПРЕДМЕТ ДОСЛІДЖЕННЯ

Розробка та впровадження автоматизованої системи, що об'єднує адміністративні та управлінські функції інтернет-магазину. Ця система охоплює весь життєвий цикл товарів, від їх додавання до відстеження продажів та аналізу діяльності.

ОГЛЯД ТА ПОРІВНЯННЯ АНАЛОГІВ

4

Для більш глибокого розуміння розробки інтернет-магазину важливо вивчити існуючі аналоги. Проведення огляду роботи популярних інтернет-магазинів та систем управління контентом дозволяє визначити оптимальні практики та уникнути можливих труднощів. Аналіз аналогів також виявляє плюси та мінуси, які можуть бути враховані під час розробки системи.

Характеристика	Shopify	Magento	OpenCart
Тип CMS	Хмарна служба / SaaS	Самостійна / Open Source	Самостійна / Open Source
Встановлення та налаштування	Просте і швидке	Складне, вимагає технічних навичок	Спрощене і швидке
Ресурси сервера	Менше вимоги	Високі вимоги	Нижчі вимоги
Витрати на розробку	Доступні розширення та теми	Високі витрати, складне розширення	Доступні розширення та теми
Управління товарами	Просте і інтуїтивне	Потужне, але складне	Зручне, але менш потужне
Оновлення	Автоматичні та безкоштовні	Складні, можуть вимагати ручного втручання	Зазвичай прості та безпечні
Кількість розширень	Обмежений маркетплейс	Широкий вибір з Magento Marketplace	Широкий вибір з OpenCart Marketplace
Гнучкість	Обмежена гнучкість	Велика гнучкість, але складніше налаштування	Задовільна гнучкість
Складність розробки	Обмежена можливість налаштування без коду	Велика складність, вимагає технічних навичок	Зазвичай менше складнощів при розробці

ВИБІР СТЕКУ ТЕХНОЛОГІЙ

5

- Вибір фреймворку. Обраний NextJS визначається сумісністю, високою продуктивністю та активною підтримкою розробників. Цей інструмент надасть нам потужність та гнучкість для швидкої та ефективної розробки функціоналу.
- Мова програмування. Вибір мови програмування TypeScript обґрунтовується її ефективністю, сумісністю з обраним фреймворком, можливістю типізації даних, що дуже прискорює розробку дозволяючи допускати менше помилок, відповідно менше дебажити.
- Система управління базою даних. Обрана ORM Prisma, в доповненні з системою керування базами даних (DBMS) – PlanetScale: відповідають вимогам щодо швидкодії та надійності у зберіганні та обробці великого обсягу даних
- Інструменти для розробки інтерфейсу. Використання Tailwind CSS разом з збіркою готових компонентів ShadcnUI – забезпечить створення інтуїтивно зрозумілого та привабливого користувацького інтерфейсу.
- Додаткові інструменти. Бібліотека для впровадження авторизації користувача – ClerkJS, система обробки платежів – Stripe.
- Врахування масштабів та ефективності. Усі вибори технологій ретельно обрані з урахуванням масштабів нашого проекту та прагнення досягти високої ефективності в умовах зростаючого обсягу даних та користувачів.
- Загалом обраний технічний стек відповідає вимогам та цілям нашого проекту. Він забезпечить швидку розробку, зручне утримання та високу продуктивність системи, сприяючи успішному втіленню наших планів та досягненню бажаних результатів.

ПОСЛІДОВНІСТЬ РОЗРОБКИ

6

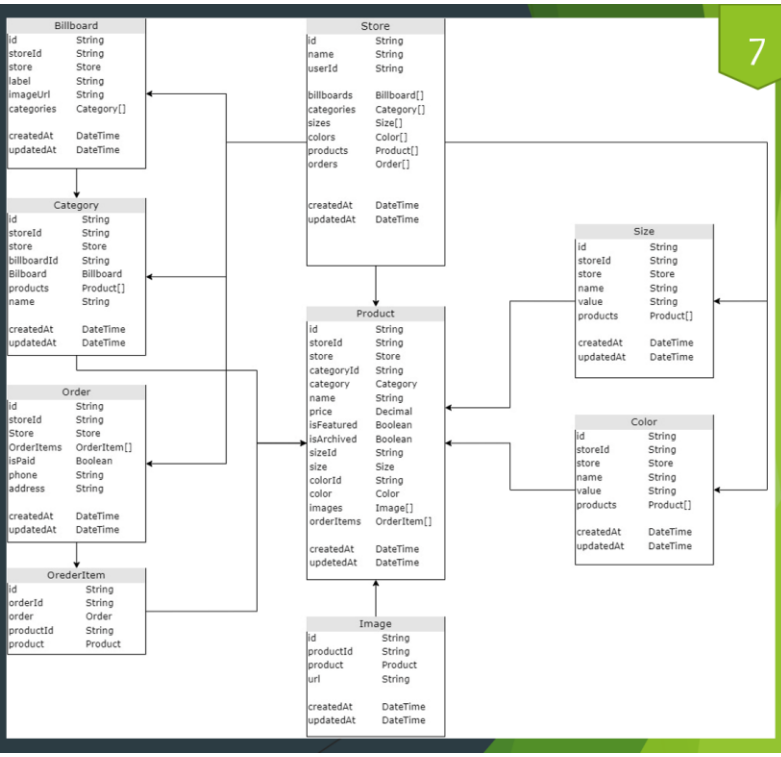
1. Налаштування середовища розробки та системи контролю версій
2. Ініціалізація проекту за допомогою стартового шаблону NextJS
3. Розробка системи авторизації користувача
4. Проектування та розробка БД використовуючи ORM Prisma
5. Розробка клієнтського інтерфейсу CMS
6. Створення API для зв’язку CMS з інтернет-магазином
7. Розробка інтернет-магазину (об’єднання з CMS та розробка клієнтського інтерфейсу)
8. Тестування розроблених додатків

Діаграма моделей та зв'язків в БД

7

- **Store** – Зберігає інформацію про магазин, включаючи ідентифікатор, назву, користувача, який його створив, та інші дати. Має зв'язки з більшістю інших моделей, таких як білборди, категорії, розміри, кольори, продукти та замовлення.
- **Billboard** – Представляє білборд, пов'язаний з конкретним магазином. Має зображення, категорії та дати створення та оновлення. Забезпечує зв'язок із магазином та може мати індекс для швидкого доступу.
- **Category** – Визначає категорію товарів у магазині та пов'язана з конкретним білбордом. Має також зв'язок з продуктами та датами створення та оновлення.
- Product – Містить інформацію про товар, таку як ім'я, ціну, чи є він улюбленим чи архівованим тощо. Має зв'язки з категорією, розміром, кольором, зображеннями та іншими елементами.
- **Size** – Представляє розмір товару та пов'язаний з магазином. Має зв'язок з продуктами та датами створення та оновлення.
- **Color** – Визначає колір товару та пов'язаний з магазином. Має зв'язок з продуктами та датами створення та оновлення.
- **Image** – Зберігає URL-адресу зображення товару та пов'язаний з конкретним продуктом.
- **Order** – Має інформацію про замовлення, таку як статус оплати, телефонний номер та адресу. Містить також зв'язок із магазином та елементами замовлення.
- **OrderItem** – Вказує, які товари були включені в конкретне замовлення.

Описана база даних надає засоби для ефективної взаємодії з інформацією



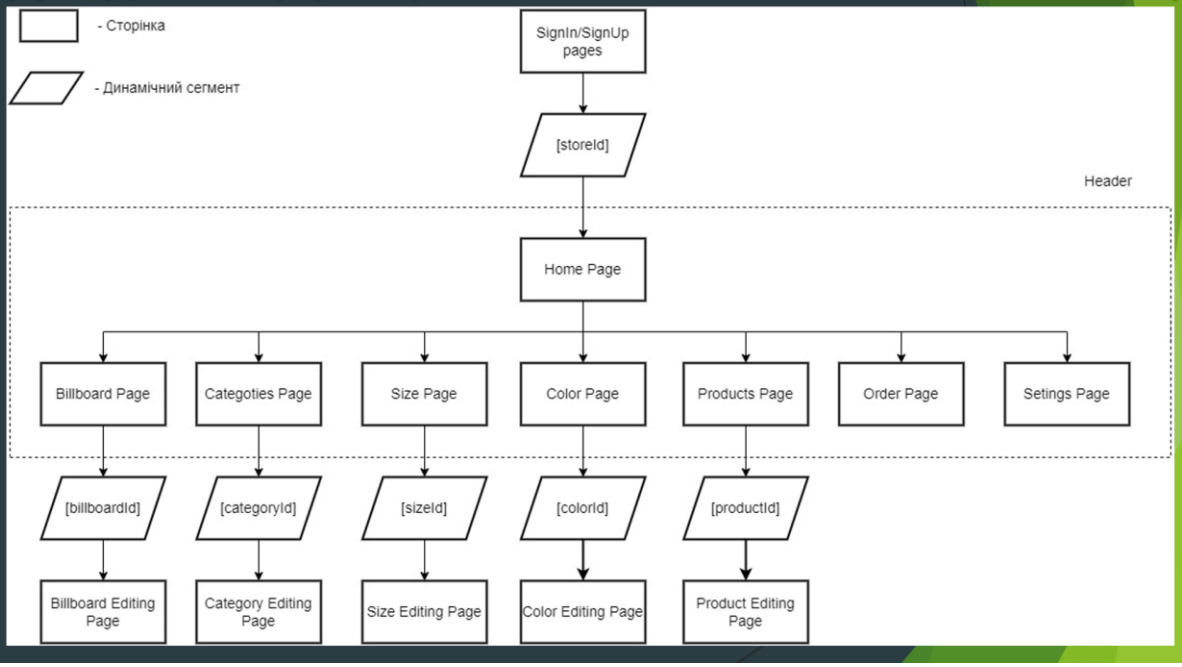
ОГЛЯД ФУНКЦІОНАЛЬНОСТІ РОЗРОБЛЕНОЇ CMS

8

- Реєстрація та авторизація
- Створення інтернет-магазину – якщо користувач не має магазину то йому необхідно обов'язково його створити.
- Домашня сторінка магазину – на даній сторінці буде зберігатись статистика конкретного магазину, така як: сума всіх продажів, кількість проданих одиниць товару, кількість товару в наявності та статистика продажів по місяцям у вигляді діаграми.
- Сторінка перегляду білбордів та їх налаштування – на даній сторінці користувач може переглянути, які білборди використовуються в його магазині, також матиме можливість їх редагувати, видаляти, або створювати нові. Білборди прив'язуються до відповідно обраної категорії і відображається на сторінці категорії інтернет-магазину.
- Сторінка перегляду категорій товарів та їх налаштування – на даній сторінці користувач так само може робити всі дії з категоріями свого товару. Категорії товару будуть використовуватись для одного з способів сортування товару в інтернет-магазині
- Сторінка перегляду розмірів товару та їх налаштування – дана сторінки дає доступ до перегляду, редагування та видалення розмірів товару.
- Сторінка перегляду кольорів товару та їх налаштування – дана сторінки дає доступ до перегляду, редагування та видалення кольорів товару.
- Сторінка перегляду продуктів та їх налаштування – дана сторінка дає доступ до створення товарів використовуючи всі попередньо створені дані, такі як: категорія, розмір, колір продукту.
- Сторінка перегляду покупок – на даній сторінці відображається історія покупок здійснених покупцями в інтернет магазині.
- Сторінка налаштування магазину – на даній сторінці адміністратор може змінювати такий параметр магазину, як його назва, або видалити поточний магазин.

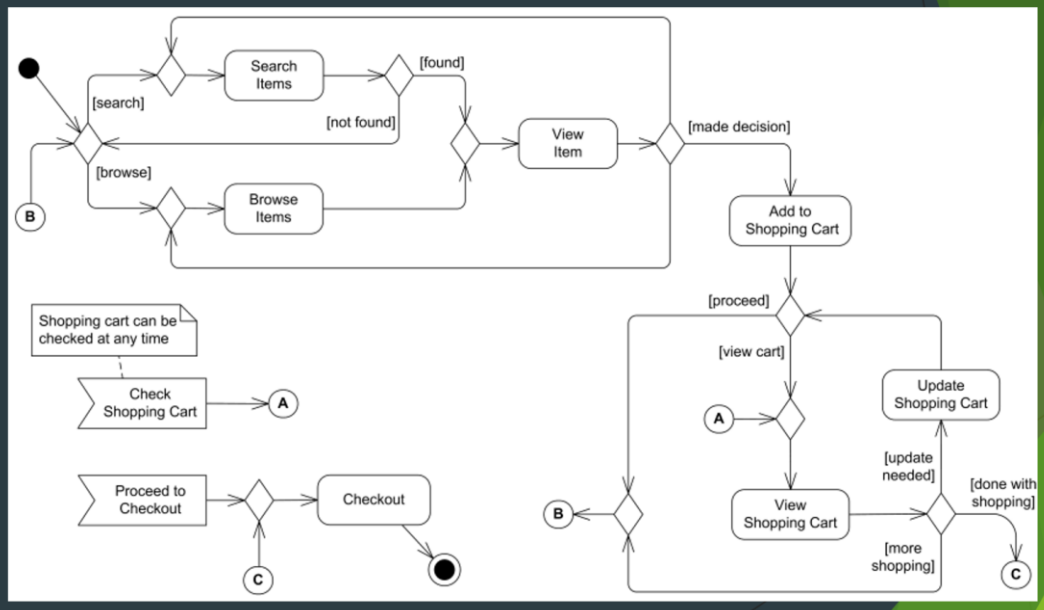
Структура інтерфейсу CMS-системи

9



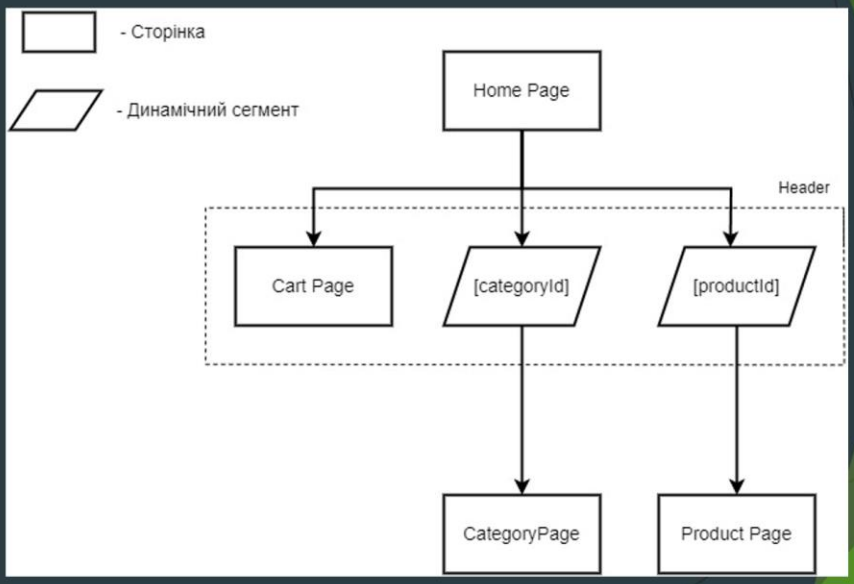
UML діаграма послідовності дій в інтернет магазині

10



Структура інтерфейсу інтернет магазину

11

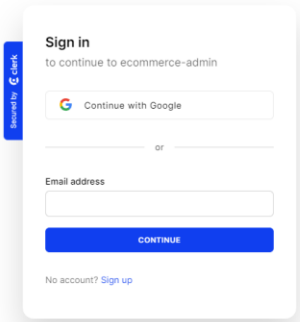


12

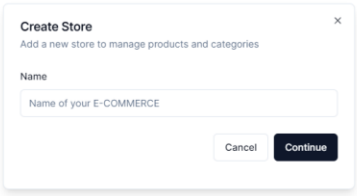
ЗОВНІШНІЙ ВИГЛЯД РОЗРОБЛЕНИХ ДОДАТКІВ

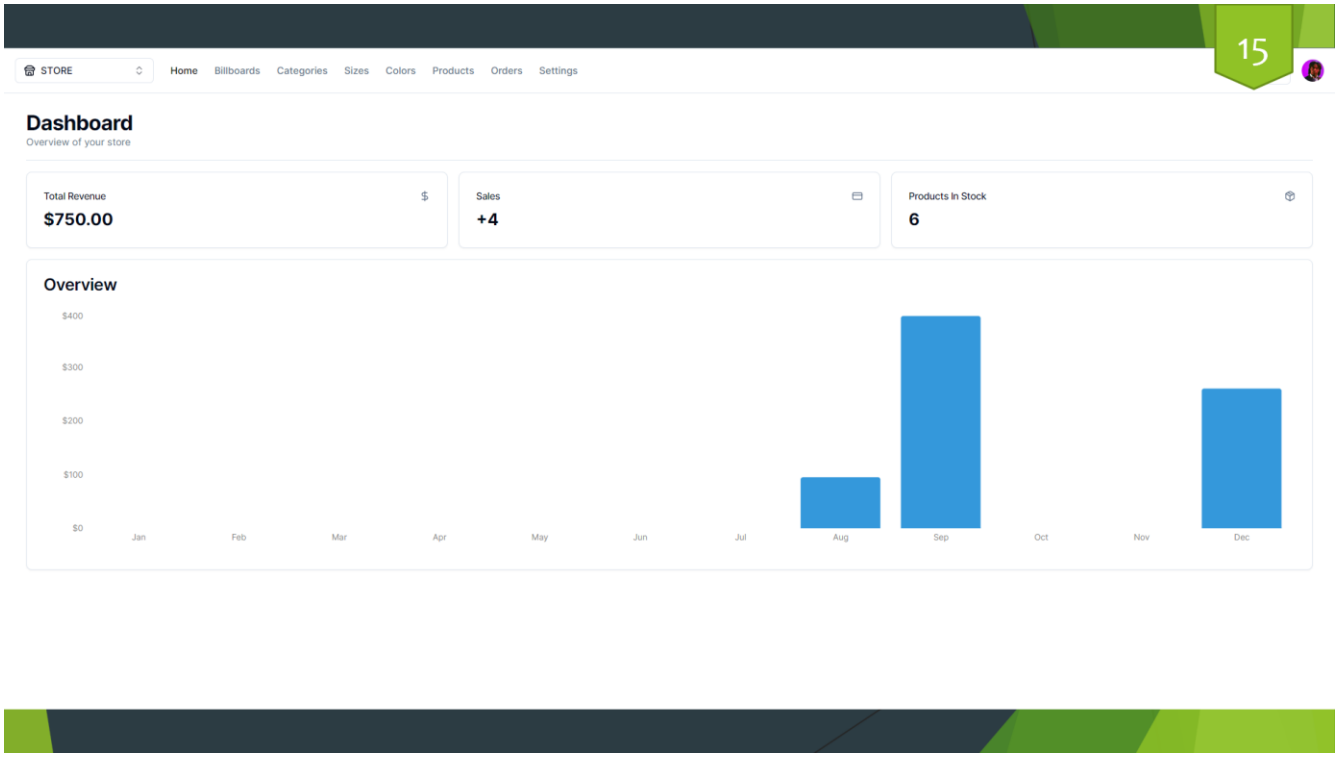
CMS - система

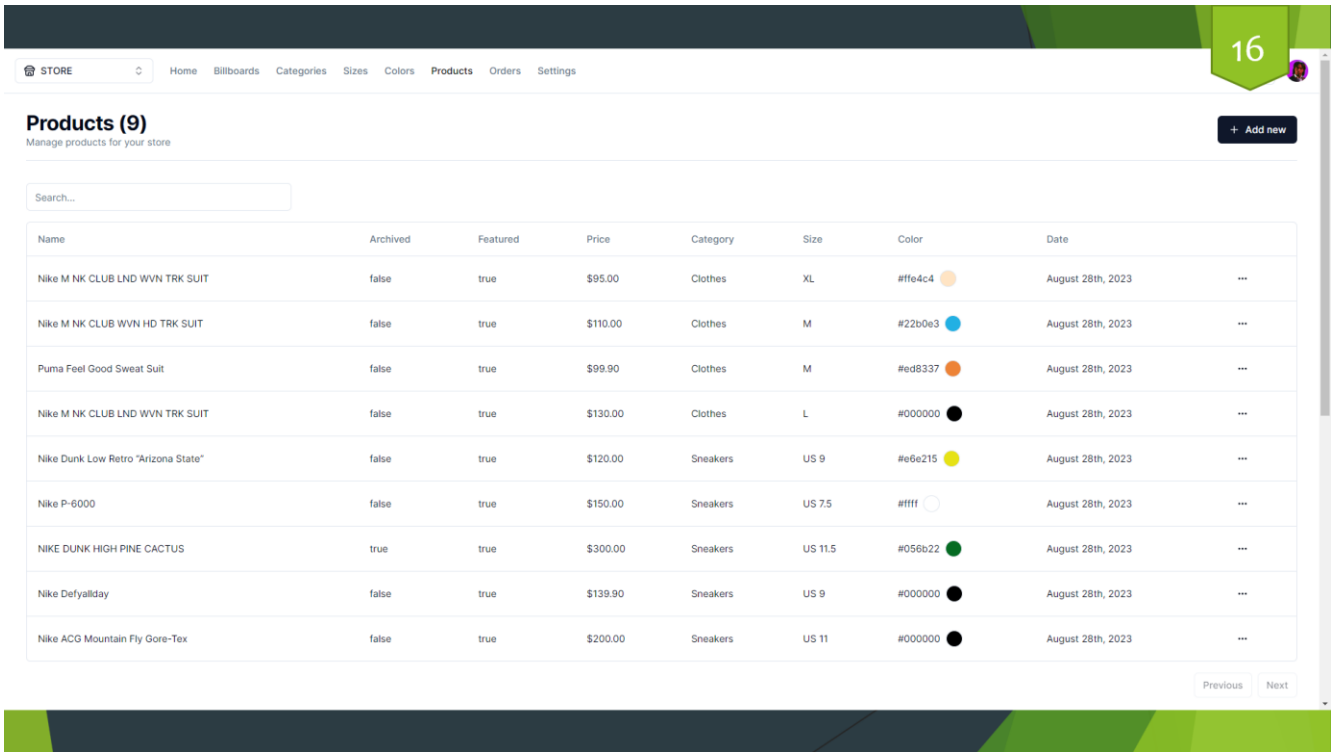
13

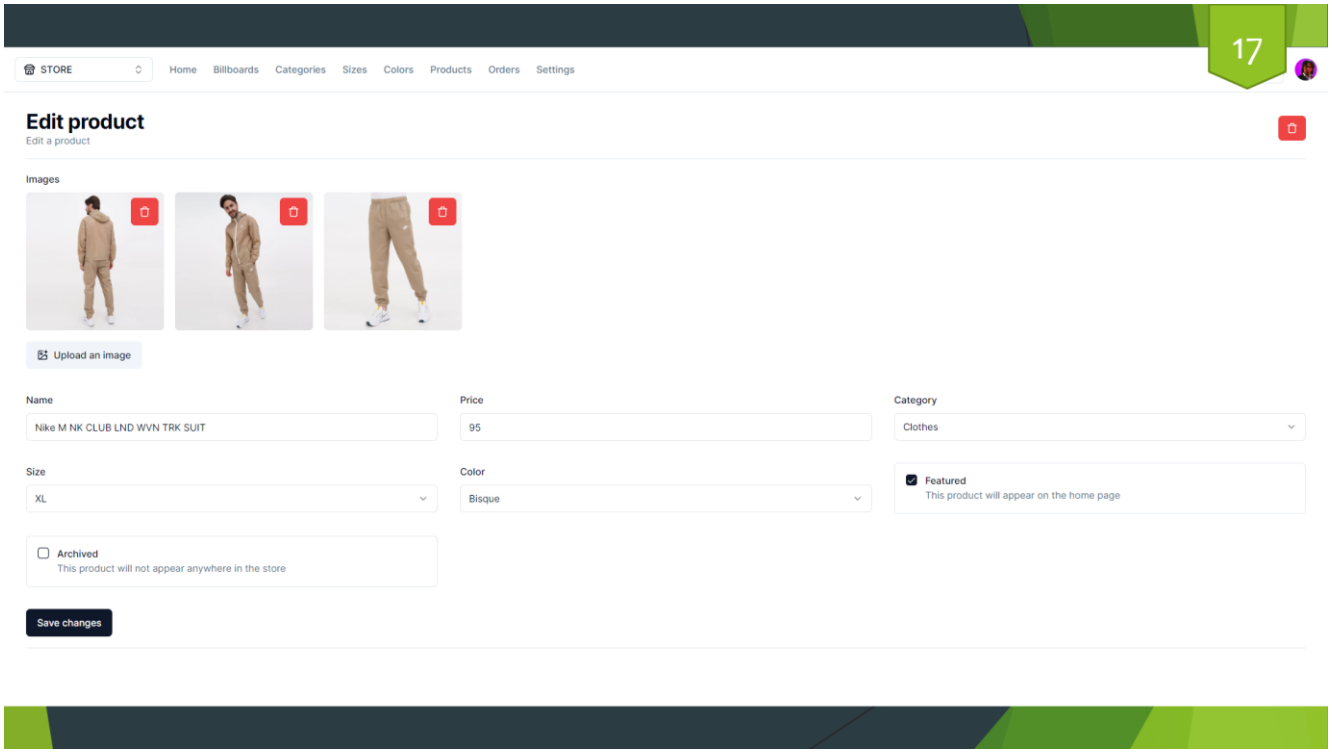


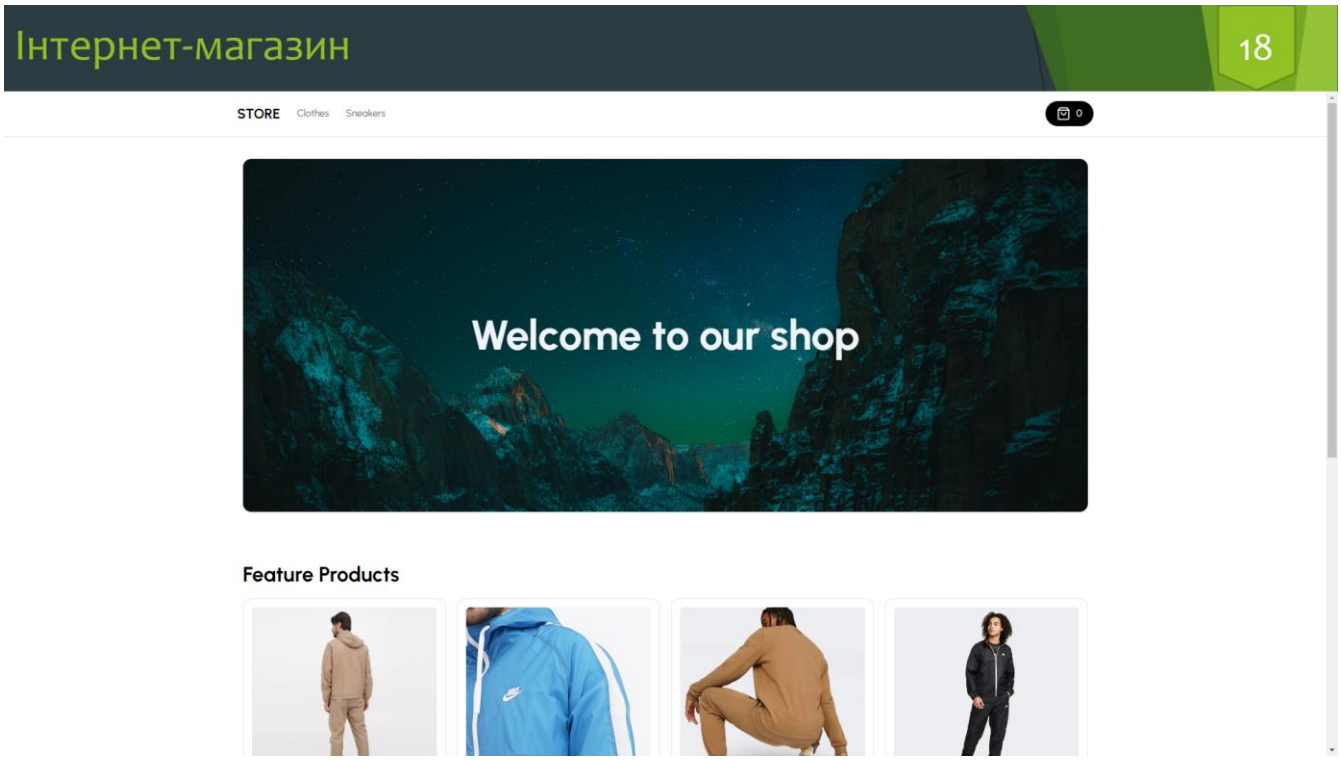
14

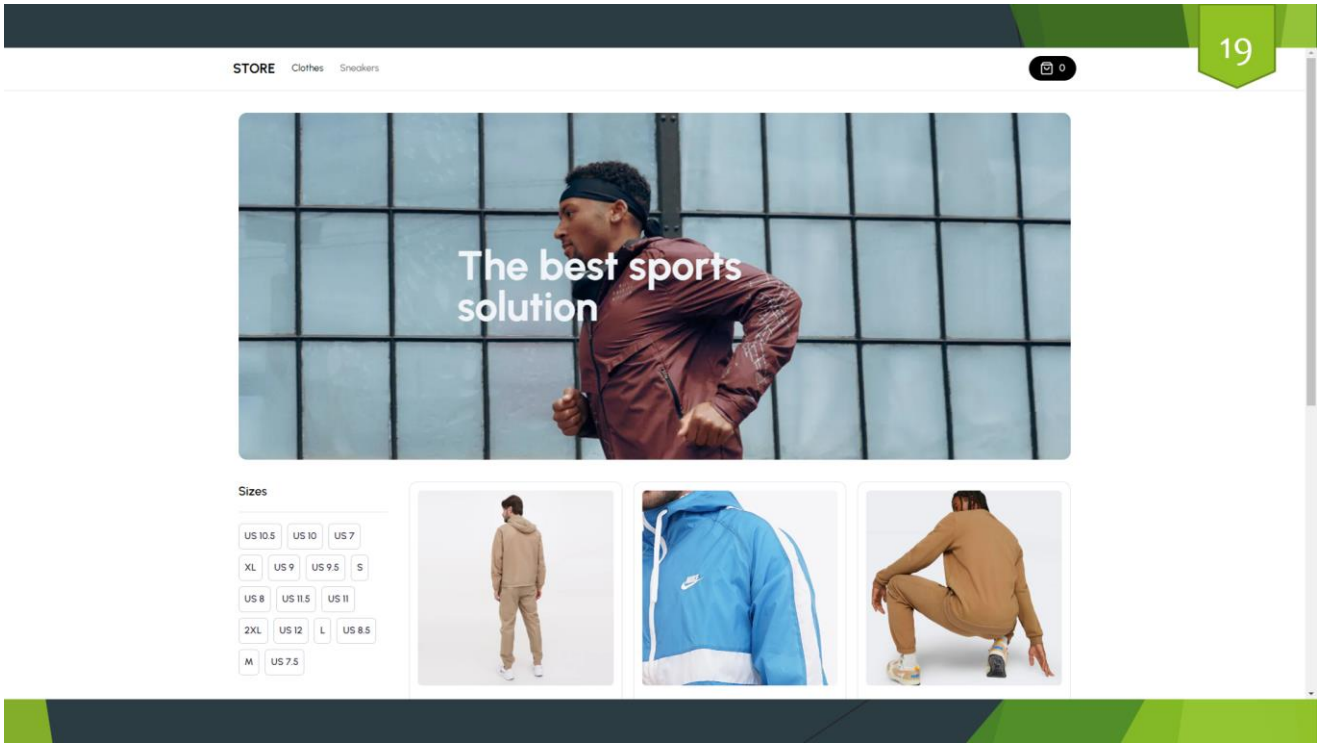


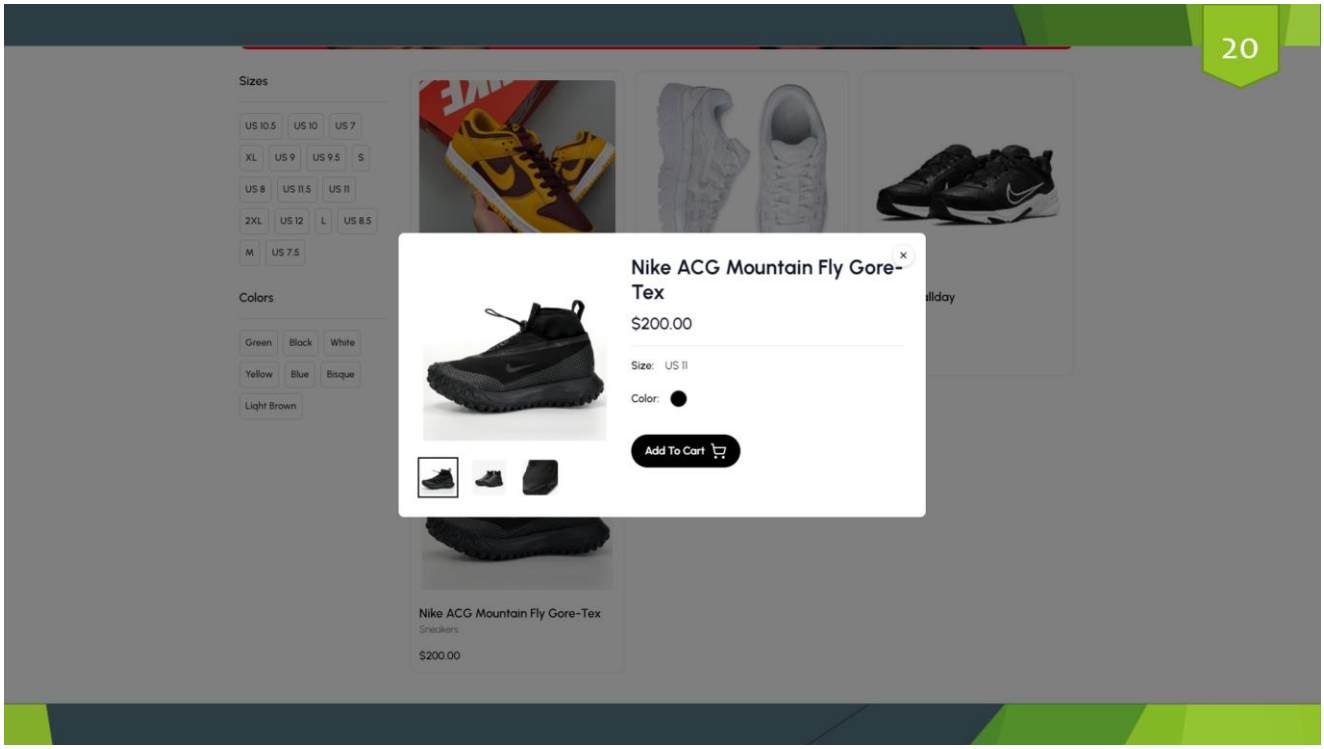

15
STORE
Home
Billboards
Categories
Sizes
Colors
Products
Orders
Settings
Dashboard
Overview of your store
Total Revenue
$750.00
Sales
+4
Products In Stock
6
Overview
$400
$300
$200
$100
$0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

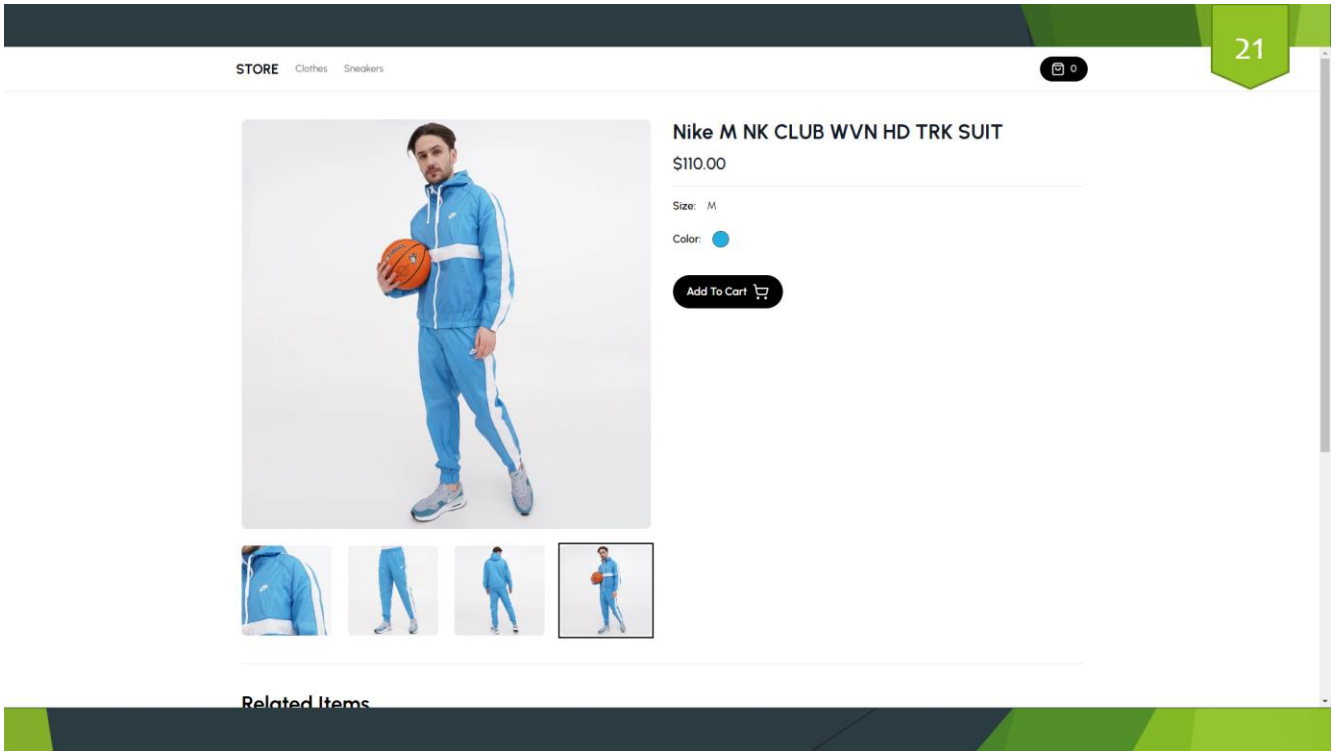

16
STORE
Home
Billboards
Categories
Sizes
Colors
Products
Orders
Settings
Products (9)
Manage products for your store
+ Add new
Search...
Name
Archived
Featured
Price
Category
Size
Color
Date
Nike M NK CLUB LND WVN TRK SUIT
false
true
$95.00
Clothes
XL
#ffe4c4
August 28th, 2023
Nike M NK CLUB WVN HD TRK SUIT
false
true
$110.00
Clothes
M
#22b0e3
August 28th, 2023
Puma Feel Good Sweat Suit
false
true
$99.90
Clothes
M
#ed8337
August 28th, 2023
Nike M NK CLUB LND WVN TRK SUIT
false
true
$130.00
Clothes
L
#000000
August 28th, 2023
Nike Dunk Low Retro "Arizona State"
false
true
$120.00
Sneakers
US 9
#e6e215
August 28th, 2023
Nike P-6000
false
true
$150.00
Sneakers
US 7.5
#ffff
August 28th, 2023
NIKE DUNK HIGH PINE CACTUS
true
true
$300.00
Sneakers
US 11.5
#056b22
August 28th, 2023
Nike Defyallday
false
true
$139.90
Sneakers
US 9
#000000
August 28th, 2023
Nike ACG Mountain Fly Gore-Tex
false
true
$200.00
Sneakers
US 11
#000000
August 28th, 2023
Previous
Next

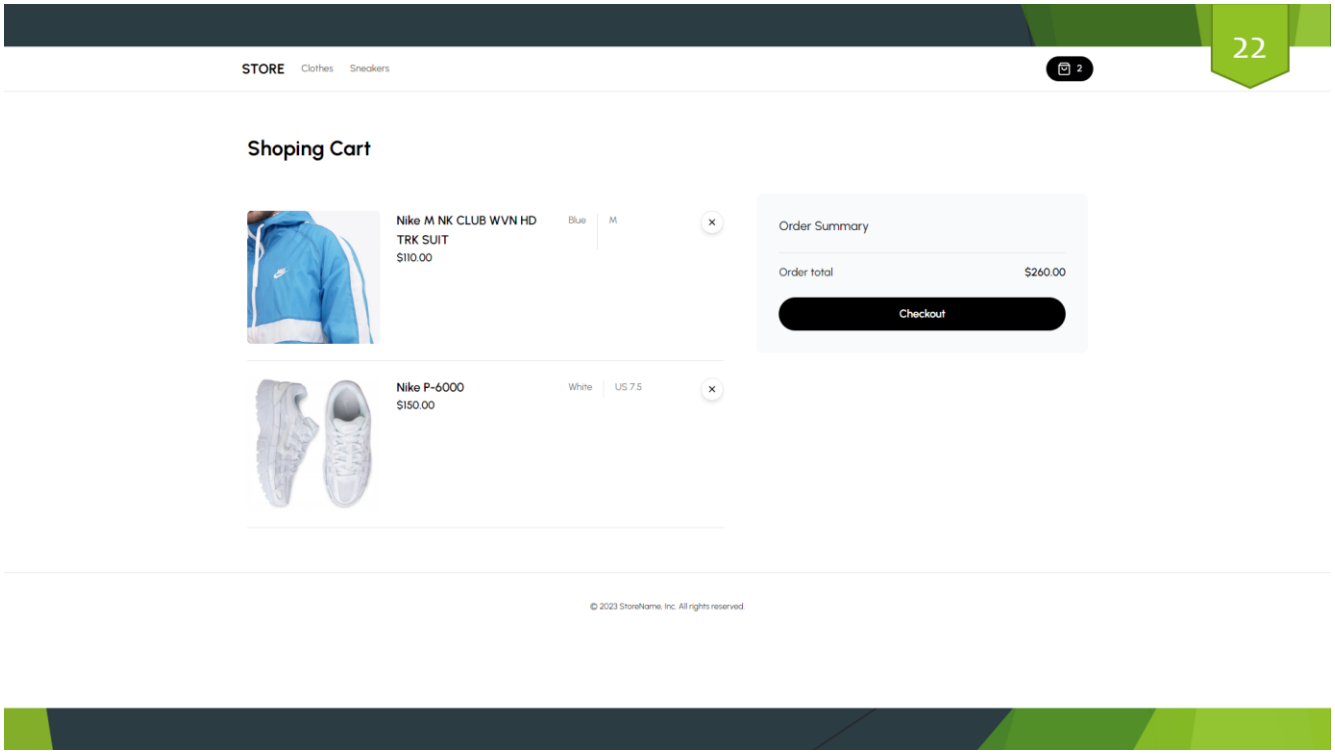

17
STORE
Home
Billboards
Categories
Sizes
Colors
Products
Orders
Settings
Edit product
Edit a product
Images
Upload an image
Name
Nike M NK CLUB LND WVN TRK SUIT
Price
95
Category
Clothes
Size
XL
Color
Bisque
Featured
This product will appear on the home page
Archived
This product will not appear anywhere in the store
Save changes


Інтернет-магазин
18
STORE
Clothes
Sneakers
0
Welcome to our shop
Feature Products

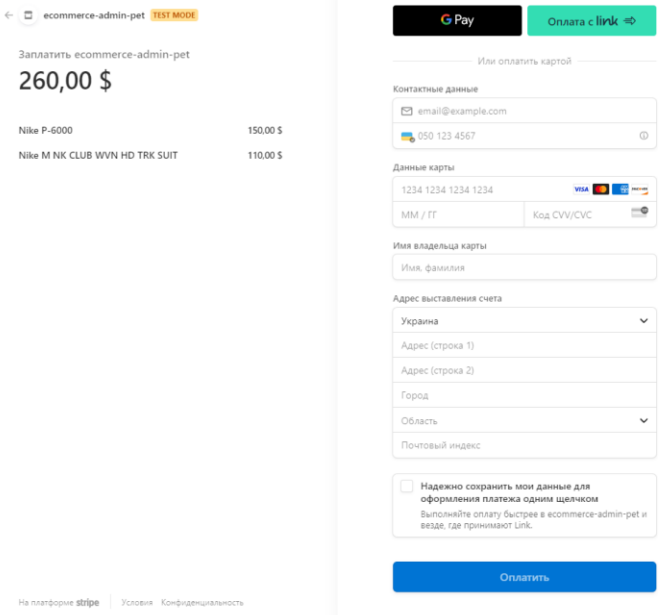

19
STORE
Clothes
Sneakers
0
The best sports solution
Sizes
US 10.5
US 10
US 7
XL
US 9
US 9.5
S
US 8
US 11.5
US 11
2XL
US 12
L
US 8.5
M
US 7.5


20
Sizes
US 10.5
US 10
US 7
XL
US 9
US 9.5
S
US 8
US 11.5
US 11
2XL
US 12
L
US 8.5
M
US 7.5
Colors
Green
Black
White
Yellow
Blue
Bisque
Light Brown
Nike ACG Mountain Fly Gore-Tex
$200.00
Size: US 11
Color:
Add To Cart
Nike ACG Mountain Fly Gore-Tex
Sneakers
$200.00


21
STORE
Clothes
Sneakers
0
Nike M NK CLUB WVN HD TRK SUIT
$110.00
Size: M
Color:
Add To Cart
Related Items


22
STORE
Clothes
Sneakers
2
Shoping Cart
Nike M NK CLUB WVN HD TRK SUIT
$110.00
Blue
M
Nike P-6000
$150.00
White
US 7.5
Order Summary
Order total
$260.00
Checkout
© 2023 StoreName, Inc. All rights reserved.

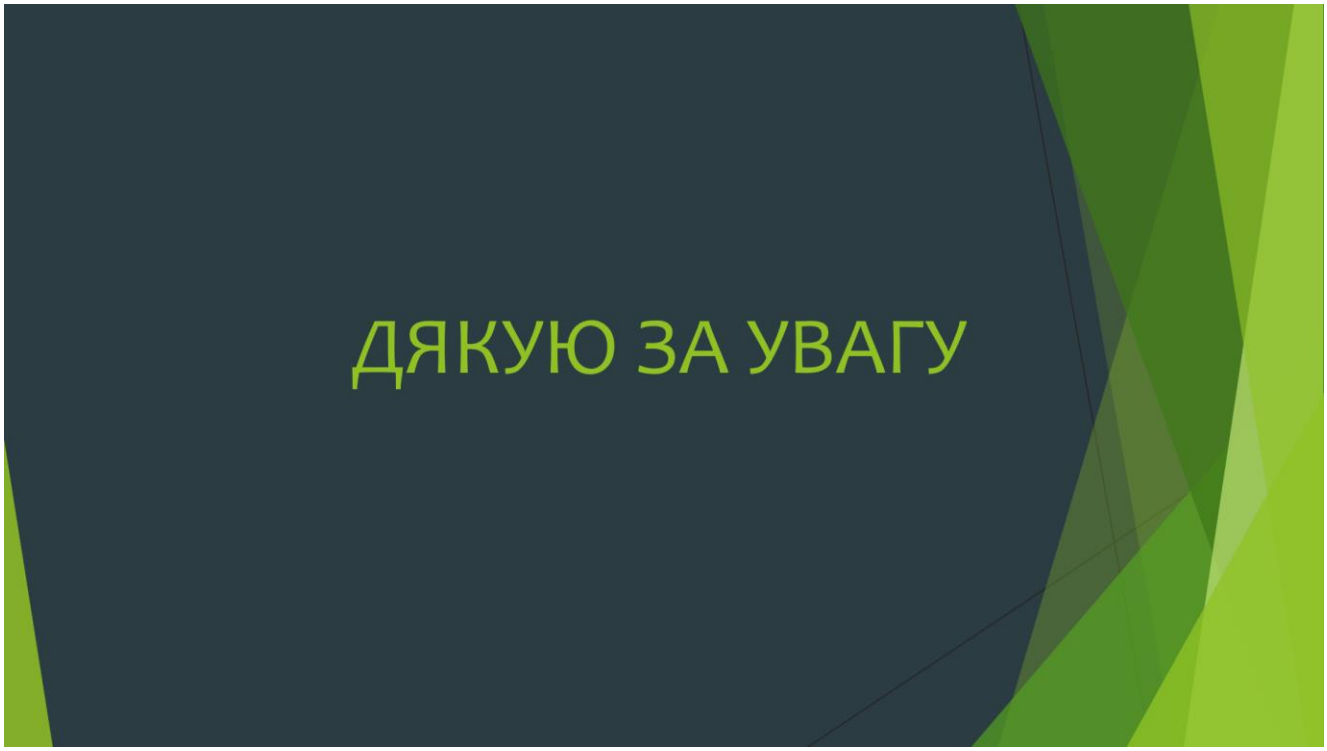
23



24

ВИСНОВКИ

- Магістерська робота "Розробка автоматизованої системи організації бізнес-процесів інтернет-магазину. Розробка CMS-системи є результатом глибоких досліджень, розробки та впровадження програмного забезпечення, спрямованого на підвищення ефективності та управління інтернет-магазином.
- Процес розробки включав вибір технологічного стеку, ініціацію проекту, розробку бази даних, створення клієнтського інтерфейсу для системи CMS та інтернет-магазину, а також огляд і тестування розроблених додатків.
- Серед основних результатів – успішне впровадження системи авторизації, впровадження функціоналу для управління товарами та контентом, а також створення ефективної системи обробки платежів. Тестування додатків виявило деякі недоліки, але загальний стан системи відповідає вимогам і високим стандартам якості.
- Магістерська робота вирішує актуальні задачі автоматизації та управління процесами в інтернет-магазині, пропонуючи ефективне програмне рішення. Отримані результати є важливим кроком на шляху розвитку сучасних технологій електронної комерції.
- Майбутні перспективи включають подальші дослідження та вдосконалення системи з урахуванням зростаючих потреб ринку. Можливості розширення функціональності, оптимізації продуктивності та впровадження нових технологій роблять цей проект основою для подальших інновацій у сфері електронної комерції.


ДЯКУЮ ЗА УВАГУ