МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР ЗА
ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Веб-застосунок інтернет-магазину стікерів за допомогою Node.js та React»

Виконавець: Ольга ШУГАЛІЙ

Керівник: к.пед.н., доцент Юрій СІНЬКО

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)

«_____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Шугалій Ольги Олександрівни
(ПІБ випускника)

1. Тема роботи: «Веб-застосунок інтернет-магазину стікерів за допомогою Node.js та React» затверджена наказом ректора № 623/ст від 01.05.2023р.

2. Термін виконання роботи: з 15 травня 2023 року по 25 червня 2023 року.

3. Вихідні дані до роботи: інтернет-магазин стікерів за допомогою Node.js та React.

4. Зміст пояснювальної записки: 1. Теоретичні засади створення веб-додатку на основі Node.js та React. 2. Розробка веб-застосунку для інтернет-магазину стікерів. 3. Функціонування інтернет-магазину стікерів.

5. Перелік обов'язкового ілюстративного матеріалу: 1. Призначення веб-застосунків. 2. Конкурентні платформи. 3. Node.js – якісний Backend. 4. Робота з базами даних. 5. JWT токен. 6. Frontend з React. 7. Взаємодія з сервером. 8. Функціональні можливості. 9. Демонстрація користувацького інтерфейсу.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Огляд теоретичних засад створення веб-додатку на основі Node.js та React. Написання 1 розділу, представлення керівнику.	15.05.2023-20.05.2023	
2.	Розробка веб-застосунку для інтернет-магазину стікерів. Написання 2 розділу, представлення керівнику.	21.05.2023-27.05.2023	
3.	Протестувати функціонування інтернет-магазину стікерів Написання 3 розділу, представлення керівнику.	28.05.2023-06.06.2023	
4.	Загальне редагування та друк пояснювальної записки.	07.06.2023-09.06.2023	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	09.06.2023-13.06.2023	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	14.06.2023-18.06.2023	

7. Дата видачі завдання _____ 15.05.2023р.

Керівник кваліфікаційної роботи _____ Юрій СІНЬКО
(підпис керівника)

Завдання прийняв до виконання _____ Ольга ШУГАЛІЙ
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Веб-застосунок інтернет-магазину стікерів за допомогою Node.js та React» містить: 53 сторінки, 40 рисунків та 20 інформаційних джерел.

Об’єкт дослідження – веб-застосунок інтернет-магазину стікерів, який розробляється з використанням технологій Node.js та React.

Предмет дослідження – використання Node.js та React у створенні веб-додатку.

Мета кваліфікаційної роботи – розробити веб-застосунок інтернет-магазину стікерів з використанням технологій Node.js та React.

Методи дослідження – аналіз літератури, визначення вимог, моделювання бази даних, розробка та тестування.

Результати кваліфікаційної роботи можуть спонукати до подальших досліджень в галузі веб-розробки використовуючи технології Node.js та React.

Для розробки веб-додатку було проаналізовано та обрано найбільш актуальну та зручну платформу Node.js, що підтримує JavaScript та дозволило створити серверну та клієнтську частину додатку на одній мові.

JAVASCRIPT, NODE.JS, БІБЛІОТЕКА REACT, РОЗРОБКА, BACKEND, FRONTEND, БАЗА ДАНИХ, СЕРВЕР, ВЕБ-ДОДАТОК, ІНТЕРНЕТ-МАГАЗИН, MODEL DEMONSTRATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

HTML	−	HyperText Markup Language
CSS	−	Cascading Style Sheets
URL	−	Uniform Resource Locator
PHP	−	Personal Home Page
API	−	Application Programming Interface
JS	−	JavaScript
npm	−	Node Package Manager
ORM	−	Object-Relational Mapping
SQL	−	Structured Query Language
IDE	−	Integrated Development Environment
id	−	identity document
JWT	−	JSON Web Token
JSON	−	JavaScript Object Notation
БД	−	База даних
СКБД	−	Система керування базами даних

ВСТУП

Розвиток технологій відіграє важливу роль у сучасному суспільстві та стає дедалі важливішим в різних аспектах людського життя.

Технології значно підвищили ефективність і продуктивність майже в кожному секторі. Автоматизація та передові інструменти оптимізували процеси, скоротили ручну працю та збільшили продуктивність. Завдання, які раніше займали години або дні, тепер можна виконати за хвилини або секунди, що веде до підвищення продуктивності та економії коштів.

Інвестування у веб-технології дозволяє компаніям, організаціям і окремим особам створювати присутність в Інтернеті за допомогою веб-сайтів і веб-додатків. Вони надають засоби для демонстрації продуктів, послуг та інформації світовій аудиторії. А також зосереджені на покращенні взаємодії з користувачем шляхом створення інтуїтивно зрозумілих інтерфейсів, адаптивного дизайну та інтерактивних функцій. Компанії спрямовані на те, щоб зробити веб-сайти та веб-додатки візуально привабливими, зручними для користувачів і доступними на різних пристроях і платформах.

Розробка інтернет-магазину на базі Node.js та React відкриває широкі можливості для підприємств в сфері електронної комерції. Цей підхід дозволяє створити потужний та функціональний веб-додаток з інтуїтивно зрозумілим інтерфейсом, який забезпечує зручну навігацію та взаємодію з користувачами.

Актуальність теми кваліфікаційної роботи "Веб-застосунок інтернет-магазину стікерів за допомогою Node.js та React" полягає в тому, що онлайн-торгівля є все більш поширеною та вигідною формою комерційної діяльності. Зростаюча кількість користувачів шукає зручні та доступні способи придбання товарів онлайн. Розробка ефективного та зручного веб-застосунку для інтернет-магазину стікерів може відповісти на цей ринковий запит та задовольнити потреби користувачів.

Використання Node.js та React у розробці веб-застосунків має свої переваги. Node.js є платформою, яка дозволяє виконувати JavaScript на сервері, що забезпечує швидку та ефективну обробку запитів. React, у свою чергу, є потужною бібліотекою

для розробки інтерфейсу користувача, яка забезпечує швидке та зручне відображення даних на клієнтському боці. Використання цих технологій дозволяє створити масштабований, ефективний та зручний веб-застосунок для інтернет-магазину стікерів, тому дослідження можливостей Node.js та React у контексті розробки веб-застосунків для інтернет-магазинів стікерів є актуальним, оскільки дозволяє відповідати на ринкові потреби та забезпечувати зручний та ефективний досвід користувачів.

Об'єктом дослідження кваліфікаційної роботи є веб-застосунок інтернет-магазину стікерів, який розробляється з використанням технологій Node.js та React.

Предмет дослідження – використання Node.js та React у створенні веб-додатку.

Мета кваліфікаційної роботи – розробити веб-застосунок інтернет-магазину стікерів з використанням технологій Node.js та React.

Відповідно до поставленої мети роботи визначено **основні завдання дослідження:**

- визначити призначення веб-застосунків;
- розглянути методи підходу до розробки веб-додатків;
- визначитися з інструментами розробки;
- провести аналіз аналогів платформи Node.js;
- обрати зручну бібліотеку для створення інтерфейсів користувачів;
- розробити серверну частину веб-застосунку;
- розробити клієнтську частину веб-додатку;
- оцінити функціонування інтернет-магазину.

Для досягнення поставленої мети й виконання завдань використано наступні методи: аналіз літератури, визначення вимог, моделювання бази даних, розробка та тестування.

Новизна роботи полягає у створенні застосунку за допомогою технології Node.js, який може вплинути на розвиток бізнесу в Україні.

Практичне значення отриманих результатів. Результати кваліфікаційної роботи можуть спонукати до подальших досліджень в галузі веб-розробки з використанням технологій Node.js та React.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ ВЕБ-ДОДАТКУ НА ОСНОВІ NODE.JS ТА REACT

1.1. Призначення веб-застосунків

Веб-додаток — це програмне забезпечення, до якого доступ і використання здійснюється через веб-браузери через Інтернет. Він призначений для забезпечення інтерактивних функцій, обробки даних і послуг для користувачів. Веб-програми зазвичай розміщуються на веб-серверах і доступ до них можна отримати з різних пристроїв, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони.

Веб-застосунки дотримуються моделі клієнт-сервер, де клієнт (веб-браузер) надсилає запити на віддалений сервер, а сервер обробляє ці запити та надсилає відповіді, що містять необхідні дані або дії.

Доступ до веб-програм здійснюється через такі веб-браузери, як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge. Користувачі взаємодіють із інтерфейсом користувача програми через браузер, який відтворює HTML, CSS і JavaScript для відображення вмісту програми.

Вони не прив'язані до певних операційних систем або типів пристроїв. Можуть працювати на будь-якій платформі з сумісним веб-браузером, що робить їх доступними для користувачів незалежно від їх пристрою чи операційної системи.

Веб-додатки мають широке охоплення, оскільки до них можна отримати доступ через URL-адреси, і вони не обмежуються встановленням із магазину програм. Користувачі можуть отримати доступ до веб-додатків з будь-якого пристрою з веб-браузером і підключенням до Інтернету.

Веб-додаток діє як віртуальна вітрина, що дозволяє компаніям охоплювати глобальну аудиторію 24/7. Він служить центром для клієнтів, щоб дізнатися про продукти, зробити покупки та взаємодіяти з брендом.

Із зростанням онлайн-магазинів веб-сайти надають підприємствам платформу для продажу продуктів і послуг безпосередньо клієнтам. Функції електронної комерції дозволяють здійснювати безпечні транзакції, керувати запасами та персоналізувати покупки.

Розробка веб-застосунків включає оптимізацію їх для мобільних пристроїв, забезпечуючи безперебійну взаємодію з користувачами на різних розмірах екрану. Мобільна сумісність є надзвичайно важливою, оскільки все більше людей мають доступ до Інтернету через смартфони та планшети.

Добре розроблений і професійний веб-застосунок допомагає завоювати довіру та зміцнити довіру потенційних клієнтів. Це дозволяє компаніям продемонструвати свій бренд, цінності та досвід.

Розробка веб-додатків продовжує розвиватися завдяки новим технологіям, структурам і тенденціям дизайну. Чуйний дизайн, інтерактивні елементи, інтеграція відео та інші інноваційні функції покращують залучення користувачів і зберігають веб-застосунки свіжими та привабливими.

Загалом розробка веб-додатків відіграє важливу роль у створенні онлайн-присутності, охопленні цільової аудиторії, стимулюванні зростання бізнесу та підтримці конкурентоспроможності в сучасному цифровому середовищі.

1.2. Методологія створення веб-додатків

Розробка інтернет-магазину передбачає створення веб-додатку, спеціально розробленого для продажу товарів або послуг через Інтернет.

Фронтенд та бекенд є основними компонентами веб-додатку або веб-сайту. Вони працюють разом, щоб створити функціональний та інтерактивний досвід користувача.

Фронтенд-розробка — це створення інтерфейсу користувача на стороні веб-сайту або програми. Це все, що бачить користувач, коли відкриває веб-сторінку, і з чим він взаємодіє: кнопки, банери та анімація. Фронтенд пов'язаний з бізнес-логікою товару (клієнтська частина постійно «спілкується» з серверною), але її розробкою займаються бекенд-програмісти.

Компоненти фронтенд-розробки:

• HTML (мова розмітки гіпертексту): HTML визначає структуру та вміст веб-сторінок. Він використовується для створення базового макета та елементів веб-сайту;

• CSS (каскадні таблиці стилів): CSS використовується для керування візуальним представленням веб-сторінок. Він визначає макет, стиль, кольори, шрифти та швидкість відгуку веб-сайту;

• JavaScript — мова програмування, яка дозволяє використовувати динамічні та інтерактивні елементи на веб-сторінках. Він використовується для обробки взаємодії користувачів, керування DOM, здійснення викликів API та додавання інтерактивності до інтерфейсу.

Розробка бекенда зосереджена на стороні сервера веб-програми. Це передбачає створення базової інфраструктури, логіки та функціональних можливостей, які забезпечують інтерфейс. Розробники бекенда відповідають за серверне програмування та керування даними [1].

Розробку серверної частини можна виконувати за допомогою різних мов програмування, таких як Python, Ruby, Java, PHP або JavaScript (з Node.js). Ці мови дозволяють розробникам обробляти запити, обробляти дані та спілкуватися з базами даних.

Бекенд-розробники працюють із базами даних для зберігання, отримання та керування даними. Зазвичай використовувані бази даних включають MySQL, PostgreSQL, MongoDB або Firebase.

Розробники серверної частини впроваджують такі заходи безпеки, як автентифікація користувачів, шифрування даних і контроль доступу, щоб захистити конфіденційні дані та забезпечити цілісність програми.

Серверні розробники займаються конфігурацією, розгортанням і обслуговуванням сервера. Вони забезпечують безперебійну роботу програми, оптимізують продуктивність і за потреби масштабують серверну інфраструктуру. Також часто інтегрують веб-додатки із зовнішніми службами, такими як платіжні шлюзи, сторонні API, служби електронної пошти або постачальники хмарних

сховищ.

1.3. JavaScript – як мова програмування для веб-розробки

JavaScript (JS) – це мова програмування високого рівня, яка в основному використовується для веб-розробки. Це універсальна мова, яка працює як на стороні клієнта (у веб-браузері), так і на стороні сервера (за допомогою Node.js).

В основному використовується для створення сценаріїв на стороні клієнта, що означає, що він працює у веб-браузері користувача. Це дозволяє розробникам створювати інтерактивні та динамічні веб-сторінки, маніпулюючи об'єктною моделлю документа і реагуючи на події користувача.

Підтримує принципи об'єктно-орієнтованого програмування, дозволяючи розробникам створювати об'єкти та керувати ними. Він надає такі функції, як класи, успадкування, інкапсуляція та поліморфізм [2].

Відмінно справляється з асинхронними операціями, такими як надсилання запитів API або отримання даних із сервера. Він використовує функції зворотного виклику, обіцянки та синтаксис async/await для ефективної обробки асинхронних завдань, запобігаючи блокуванню основного потоку виконання.

JavaScript має велику та активну екосистему бібліотек, фреймворків та інструментів, які роблять розробку простішою та ефективнішою. До популярних фреймворків JavaScript належать React.js, Angular, Vue.js і Express.js. Такі бібліотеки, як jQuery, lodash і moment.js, надають додаткові функції та утиліти.

Підтримується всіма сучасними веб-браузерами, що робить його універсальною мовою для веб-розробки на стороні клієнта. Однак неузгодженість веб-переглядача та варіації в реалізаціях JavaScript все ще можуть створювати проблеми, які розробники повинні враховувати.

JavaScript є важливою частиною веб-розробки, що дозволяє розробникам створювати динамічні та інтерактивні веб-дослідження. Його універсальність, широка підтримка браузера та активна екосистема роблять його потужною мовою для створення різноманітних програм як на стороні клієнта, так і на стороні сервера.

Наведемо приклад простої веб-програми, яка дозволяє користувачеві ввести своє ім’я та отримати персональне привітання (рис. 1.1.).

```
<!DOCTYPE html>
<html>
<head>
    <title>Web Application Example</title>
    <script>
        // JavaScript code for the web application
        function greetUser() {
            var name = document.getElementById('nameInput').value;
            var greeting = 'Hello, ' + name + '!';
            document.getElementById('greetingText').textContent = greeting;
        }
    </script>
</head>
<body>
    <h1>Web Application Example</h1>
    <input type="text" id="nameInput" placeholder="Enter your name">
    <button onclick="greetUser()">Greet</button>
    <p id="greetingText"></p>
</body>
</html>

```

Рис. 1.1. Взаємодія JavaScript з елементами HTML

Тег <script> використовується для вбудовування коду JavaScript у файл HTML.

Коли користувач вводить своє ім’я та натискає кнопку «Привітати», код JavaScript виконується, і персоналізоване привітання відображається на веб-сторінці.

1.4. Середовище виконання Node.js

Node.js — це міжплатформне середовище виконання JavaScript із відкритим кодом, яке дозволяє розробникам створювати серверні та мережеві програми (рис. 1.2.). Воно використовує керовану подіями неблокуючу модель введення-виведення, що робить його ефективним і масштабованим для обробки одночасних з’єднань [3].

Рис. 1.2. Логотип Node.js

Побудовано на механізмі JavaScript V8, який також використовується у веб-переглядачах, таких як Google Chrome. Це дозволяє розробникам писати програми на стороні сервера за допомогою JavaScript, уможливлюючи повну розробку JavaScript.

Особливо популярний Node.js для розробки на стороні сервера. Він забезпечує середовище виконання, яке дозволяє розробникам виконувати код JavaScript на сервері, відповідаючи на запити клієнта та обробляючи та зберігаючи дані [4].

Використовує керовану подіями неблокуючу модель вводу-виводу, що означає, що він може обробляти кілька одночасних з'єднань, не блокуючись тривалими операціями. Ця архітектура робить Node.js високоефективним і добре підходить для додатків з високими вимогами до введення-виведення.

Node.js поставляється з менеджером пакетів вузлів, який є найбільшою екосистемою бібліотек і модулів з відкритим кодом для JavaScript [5]. Менеджер пакетів вузлів забезпечує легкий доступ до широкого спектру готових модулів, які розробники можуть використовувати у своїх проектах, підвищуючи продуктивність розробки.

За допомогою Node.js розробники можуть створювати легкі та ефективні веб-сервери або API. Його керована подіями архітектура дозволяє обробляти велику кількість одночасних з'єднань, що робить його придатним для додатків у реальному часі, таких як додатки чату, ігрові сервери або інструменти для спільної роботи [6].

1.5. Аналіз аналогів Node.js для серверної розробки

Node.js є однією з найпопулярніших платформ для серверної розробки, використовуючи мову програмування JavaScript. Вона набула значної популярності завдяки кільком чинникам. По-перше, використання JavaScript як спільної мови програмування для клієнтського та серверного коду спрощує розробку та забезпечує легший обмін знаннями між розробниками. По-друге, Node.js базується на неблокуючому вводі/виводі, що дозволяє йому ефективно обробляти багатопотокові операції з низькою системною витратою. Це забезпечує високу швидкодію та

масштабованість. Однак, на ринку також існує кілька аналогів та альтернативних платформ для розробки серверних додатків.

1.5.1. Deno

Deno є альтернативною платформою для серверної розробки JavaScript та TypeScript, розробленої творцем Node.js Райаном Далем (рис. 1.3.). Вона пропонує більш безпечне та сучасне середовище виконання, що включає вбудовану підтримку модулів, асинхронні операції, верхній рівень очікування та можливість використання TypeScript без необхідності окремої компіляції. Deno також ставить більший акцент на безпеку та контроль доступу до ресурсів.

Рис. 1.3. Логотип Deno

До переваг Deno можна віднести [7]:

- безпечність за замовчуванням;

Deno має вбудовану безпеку за замовчуванням, що означає, що він захищений від потенційно небезпечних операцій, таких як доступ до файлової системи чи мережі, без явної дозволу. Це допомагає уникнути вразливостей безпеки і забезпечує більшу надійність додатків.

- вбудована підтримка TypeScript;

Deno має вбудовану підтримку мови TypeScript, що робить розробку більш простою та продуктивною. TypeScript дозволяє використовувати сильну типізацію, забезпечує підказки коду та перевірку помилок на етапі компіляції.

- менеджер модулів.

Deno має вбудований менеджер модулів, який дозволяє легко імпортувати

залежності з різних джерел, включаючи веб-URL, локальні файли та пакети з репозиторіїв, таких як npm. Це спрощує управління залежностями та дозволяє легко використовувати сторонні бібліотеки в проектах.

До основних недоліків Deno відноситься:

- екосистема та зрілість;

Deno є новітньою технологією порівняно з Node.js, тому екосистема та доступність бібліотек можуть бути меншими. Вибір бібліотек та інструментів може бути обмеженим порівняно з великим вибором, доступним у Node.js.

- обмежена сумісність з кодом Node.js;

Незважаючи на синтаксичну сумісність з JavaScript, Deno не є повністю сумісним з кодом, написаним для Node.js. Деякі Node.js-специфічні модулі та функції можуть не підтримуватися в Deno, що може вимагати додаткової роботи при перенесенні проектів з Node.js на Deno.

- низька прийнятність у розробників.

Незважаючи на швидкий ріст популярності, Deno все ще має менше розробницьке співтовариство порівняно з Node.js. Це може вплинути на доступність допомоги, плагінів та рішень для певних завдань.

1.5.2. ASP.NET

ASP.NET є однією з основних альтернатив Node.js для серверної розробки (рис. 1.4.). Це фреймворк на основі мови програмування C#, який надає широкий набір інструментів для розробки веб-додатків. ASP.NET має вбудовану підтримку мови C# і забезпечує розширені можливості для створення потужних і масштабованих веб-додатків [8]. Він також має багатофункціональні фреймворки, такі як ASP.NET MVC і ASP.NET Core, які дозволяють створювати додатки в стилі Model-View-Controller.

Рис. 1.4. Логотип ASP.NET

Переваги ASP.NET:

- мова програмування C#;

Використання мови C# дозволяє розробникам мати сильнотипізовану, ефективну та розширювану кодову базу. C# є однією з популярних мов програмування, з великою кількістю ресурсів та підтримкою від Microsoft та спільноти.

- великий набір бібліотек та функціональності;

ASP.NET надає широкий набір готових бібліотек і компонентів, що спрощують розробку. Він має вбудовану підтримку для роботи з базами даних, безпеки, маршрутизації, кешування та багатьох інших функцій.

- висока продуктивність.

ASP.NET (C#) має вбудовану оптимізацію та управління пам'яттю, що дозволяє створювати веб-додатки з високою продуктивністю та швидкодією. Він також має механізми кешування, які допомагають знизити навантаження на сервер та покращити швидкість відповіді.

Недоліки ASP.NET:

- вимоги до хостингу;

Для розгортання веб-додатків ASP.NET (C#) потрібен хостинг, який підтримує ASP.NET і виконується на сервері Windows. Це може бути витратним і не завжди доступним на деяких хостинг-провайдерах.

- великий обсяг коду;

Порівняно з деякими легковаговими фреймворками, такими як Node.js, ASP.NET (C#) може мати більший обсяг коду, що потрібно написати, щоб створити

функціональний веб-додаток. Це може збільшити тривалість розробки.

- витрати на ліцензії.

Для використання деяких продуктів і інструментів в екосистемі ASP.NET (наприклад, Visual Studio Enterprise) можуть бути вимоги до ліцензування, що може збільшити вартість розробки.

1.5.3. Ruby on Rails

Ruby on Rails (рис. 1.5.), також відомий як Rails, є фреймворком на мові програмування Ruby для серверної розробки. Rails пропагує конвенцію над конфігурацією і забезпечує готові рішення для багатьох задач, таких як маршрутизація, ORM (Active Record), шаблонізація та багато іншого. Він має активне та підтримуване спільнотою, а також багато плагінів і гемів, що робить розробку веб-додатків на Ruby зручною та продуктивною.

Рис. 1.5. Логотип Ruby on Rails

Переваги Ruby on Rails [9]:

- продуктивність розробки;

Ruby on Rails має простий та зрозумілий синтаксис, що дозволяє розробникам швидко створювати функціональні веб-додатки. Він пропонує багато вбудованих інструментів та конвенцій, що спрощують розробку та зменшують кількість коду.

- концентрація на зручності розробки;

Ruby on Rails покладає великий акцент на зручність розробки. Він пропонує широкий спектр готових модулів, бібліотек і плагінів, що допомагають розробникам

зосередитися на бізнес-логіці додатку, а не на технічних деталях.

- висока продуктивність.

Ruby on Rails пропонує вбудовану оптимізацію та автоматичне кешування, що дозволяє забезпечити високу продуктивність веб-додатків. Він також має вбудовану підтримку для масштабування та розподіленої роботи, що дозволяє легко розширювати додатки зростанням навантаження.

Недоліки Ruby on Rails:

- швидкодія;

Ruby on Rails не є найшвидшим фреймворком порівняно з іншими мовами та фреймворками. Він може мати обмежену швидкодію при обробці великого обсягу запитів або при великих навантаженнях.

- вимоги до ресурсів;

Ruby on Rails може бути вимогливим до ресурсів сервера, зокрема до пам'яті. Це може потребувати більш потужних серверів для запуску веб-додатків на базі Ruby on Rails.

- обмежена масштабованість;

Хоча Ruby on Rails має підтримку масштабування, він може виявитися обмеженим у порівнянні з іншими фреймворками, такими як Node.js або Go, коли мова йде про горизонтальне масштабування та великі розподілені системи.

- вивчення кривої.

Хоча Ruby має простий синтаксис, вивчення Ruby on Rails може вимагати певного часу та зусиль. Особливо для новачків у веб-розробці, які не мають досвіду з мовою Ruby, може бути потрібно додаткове навчання та адаптація.

1.5.4. Flask

Flask є легковаговим веб-фреймворком мовою Python для серверної розробки. Він забезпечує мінімальний набір інструментів для створення веб-додатків, при цьому залишаючи більшу свободу вибору та гнучкість розробнику. Flask (рис. 1.6.) має просту і інтуїтивно зрозумілу архітектуру і хорошу

документацію, що робить його популярним вибором для розробників-початківців [10].

Рис. 1.6. Логотип Flask

Переваги Flask:

- простота та простота використання;

Flask має простий та інтуїтивно зрозумілий синтаксис, що дозволяє швидко почати розробку веб-додатків. Він не накладає жорстких правил або конвенцій, дозволяючи розробникам більшу свободу в організації свого коду.

- гнучкість;

Flask є гнучким фреймворком, що дозволяє вибирати лише ті компоненти, які вам потрібні для вашого проекту. Ви можете використовувати різні сторонні бібліотеки та інструменти, щоб налаштувати роботу Flask під свої потреби.

- розширюваність;

Flask пропонує широкий вибір розширень та плагінів, які допомагають розширити функціональність вашого додатку. Ви можете легко додавати нові функції, такі як автентифікація, робота з базами даних, роутинг та багато іншого, за допомогою цих розширень.

До недоліку платформи Flask можна віднести:

- відсутність вбудованої функціональності.

Оскільки Flask є легковаговим фреймворком, він не має вбудованої підтримки для багатьох функцій, які можуть бути потрібні.

1.6. Бібліотека React

React — це бібліотека JavaScript, яка використовується для створення інтерфейсів користувача. Вона широко використовується для втілення

інтерактивних і динамічних веб-додатків [11].

Перевагою такої бібліотеки є те, що вона дотримується компонентної архітектури, де користувальницький інтерфейс розділений на повторно використовувані та самодостатні компоненти. Кожен компонент представляє певну частину інтерфейсу користувача та може мати власний стан і поведінку.

React використовує віртуальну модель об'єктів документа для ефективного оновлення та відтворення компонентів. Віртуальна модель об'єктів документа є полегшеною копією фактичної моделі об'єктів документа, і бібліотека використовує її для ефективного оновлення та мінімізації непотрібних повторних візуалізацій.

React дозволяє створювати повторно використовувані компоненти інтерфейсу користувача, які інкапсулюють свою власну логіку та рендеринг. Компоненти можуть мати власний стан і властивості (реквізити) і можуть бути складені разом для створення складних структур інтерфейсу користувача.

Ця бібліотека дотримується шаблону односпрямованого потоку даних, де дані протікають в одному напрямку від батьківських компонентів до дочірніх компонентів. Це забезпечує передбачуваність коду, який можна підтримувати.

React дозволяє керувати станом компонента, який представляє дані, які можуть змінюватися з часом. Стан можна оновити за допомогою методу setState(), і коли стан змінюється, бібліотека ефективно оновлює відповідні компоненти в інтерфейсі користувача [12].

React набув величезної популярності завдяки своїй ефективності, можливості багаторазового використання та зручним для розробників функціям. Він широко використовується компаніями та розробниками для створення сучасних, адаптивних і високопродуктивних веб-додатків. Його декларативний синтаксис, віртуальна модель об'єктів документа і компонентна архітектура роблять його потужним вибором для створення інтерактивних інтерфейсів користувача.

ВИСНОВКИ ДО РОЗДІЛУ 1

Розробка веб-додатків є важливим і невід'ємним елементом сучасного цифрового світу. Веб-додатки дозволяють забезпечувати інтерактивні функції, обробку даних та послуги для користувачів через веб-браузери та Інтернет. Вони працюють на різних пристроях, незалежно від операційної системи, що робить їх доступними для широкої аудиторії.

У першому розділі було розглянуто методологію створення веб-додатків, де фронтенд та бекенд є ключовими компонентами для реалізації функціонального та інтерактивного користувацького досвіду. Ці компоненти співпрацюють між собою, щоб забезпечити оптимальну роботу додатку або веб-сайту.

Окрема увага була приділена JavaScript - мові програмування, яка є важливою складовою веб-розробки. Вона дозволяє розробникам створювати динамічні та інтерактивні елементи на веб-сторінках.

Також було досліджено середовище Node.js та його аналоги. Node.js є популярною платформою для серверної розробки, яка використовує мову JavaScript. Це дає змогу розробникам використовувати JS не тільки на клієнтській стороні, але і на сервері.

У розділі було розглянуто бібліотеку React, яка стала популярною завдяки своїм перевагам у продуктивності та яка дозволяє розробникам створювати повторно використовувані компоненти користувацького інтерфейсу. Ці компоненти включають в себе власну логіку та можуть бути ефективно використані для рендерингу веб-сторінок.

В цілому, розробка веб-додатків є важливою для створення сучасного онлайн-середовища, яке надає користувачам функціональність та зручність взаємодії, тому далі слід розглянути етапи створення веб-застосунку для інтернет-магазину стікерів.

РОЗДІЛ 2
РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ СТІКЕРІВ

2.1. Розробка серверної частини – Backend

Розробка інтернет-магазину на базі Node.js та React має значні переваги порівняно з іншими веб-фреймворками (рис. 2.1.). За даними 2022 року, Node.js використовувався більшістю розробників (47,12%), що свідчить про його популярність і визнання веб-суспільством. Його можливості забезпечують потужну серверну розробку, здатну оптимально обробляти запити користувачів та масштабуватися при необхідності.

Рис. 2.1. Найбільш використовувані веб-фреймворки серед розробників у всьому світі, станом на 2022 рік

У той же час, React.js, що займає друге місце серед веб-фреймворків (42,62%), є потужним інструментом для розробки фронтенду. Використання React дозволяє створювати динамічні та інтерактивні інтерфейси, що забезпечує зручний

користувацький досвід та ефективну роботу додатку.

Слід зазначити, що jQuery, який займає третє місце (28,57%), є одним із найпопулярніших JavaScript-фреймворків для розробки веб-додатків. Хоча він не так активно оновлюється, як Node.js та React, jQuery все ще залишається популярним у багатьох проектах та веб-сайтах.

У цілому, використання Node.js та React для розробки інтернет-магазину забезпечує надійну та ефективну архітектуру, яка сприяє швидкому впровадженню та покращенню функціональності. Популярність цих фреймворків серед розробників свідчить про їхню надійність та здатність задовольняти потреби сучасного цифрового світу.

При розробці серверної частини магазину, будуть застосовуватись такі стек технології, як:

- Node.js – платформа, на якій можна створювати бекенд на мові JavaScript;
- зв'язки з Express – фреймворк для написання серверної частини на Node.js;
- Postgre SQL – система управління базами даних;
- Sequelize – в якості ORM для реляційних баз даних.

ORM – це певна технологія, яка дозволяє пов'язувати програмний код з базами даних (рис. 2.2.). За допомогою чого, не прописуючи безпосередньо SQL запити, викликаємо певну функцію, наприклад CREATE і об'єкт додається в БД [13].

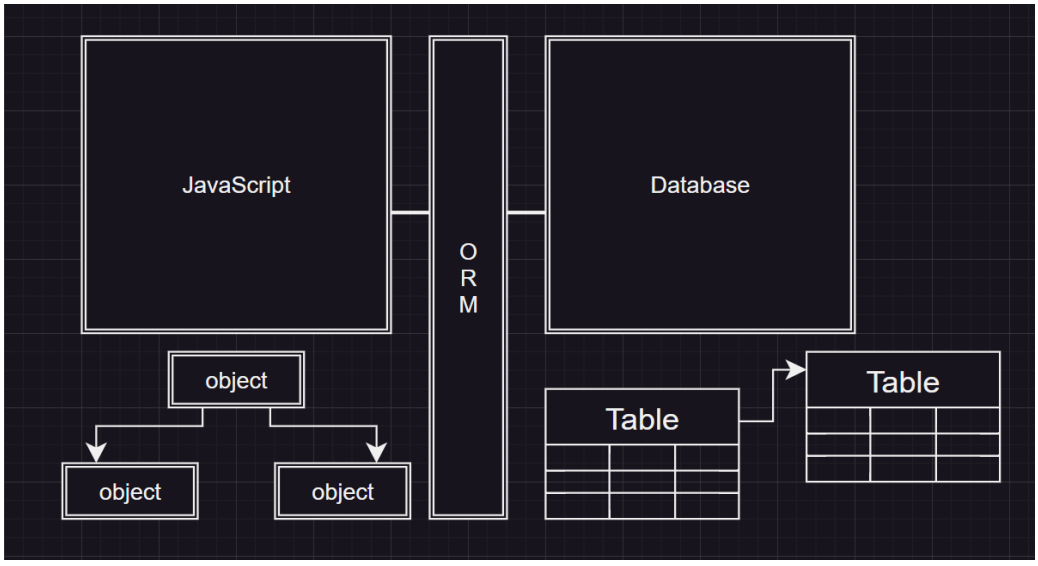


Рис. 2.2. Представлення ORM

Цей рівень відображення забезпечує трансляцію між об’єктно-орієнтованим кодом і реляційною структурою бази даних, полегшуючи взаємодію з базою даних за допомогою знайомих парадигм програмування.

Використовуючи ORM, розробники можуть більше зосередитися на бізнес-логіці своєї програми, а не на деталях взаємодії з базою даних. Структури ORM забезпечують вищий рівень абстракції та допомагають оптимізувати операції з базою даних, покращити читабельність коду та скоротити час, необхідний для завдань, пов’язаних із базою даних.

Однак важливо зазначити, що ORM не завжди є найкращим рішенням для кожного проекту чи випадку використання. У деяких сценаріях написання спеціальних запитів SQL може бути більш ефективним або необхідним для обробки складних операцій бази даних або оптимізації продуктивності.

2.1.1. Редактор коду WebStorm

Процес розробки буде відбуватися із використанням редактору коду WebStorm . Це інтегроване середовище розробки (IDE), спеціально розроблене для веб-розробки, створене міжнародною компанією JetBrains [14]. Воно надає повний набір інструментів і функцій для підвищення продуктивності та оптимізації процесу розробки. Редактор забезпечує перевірку коду, швидкі виправлення та пропозиції щодо покращення якості коду. Допомагає виявити потенційні проблеми та порушення стилю кодування. Його представлення можна побачити на рис. 2.3.

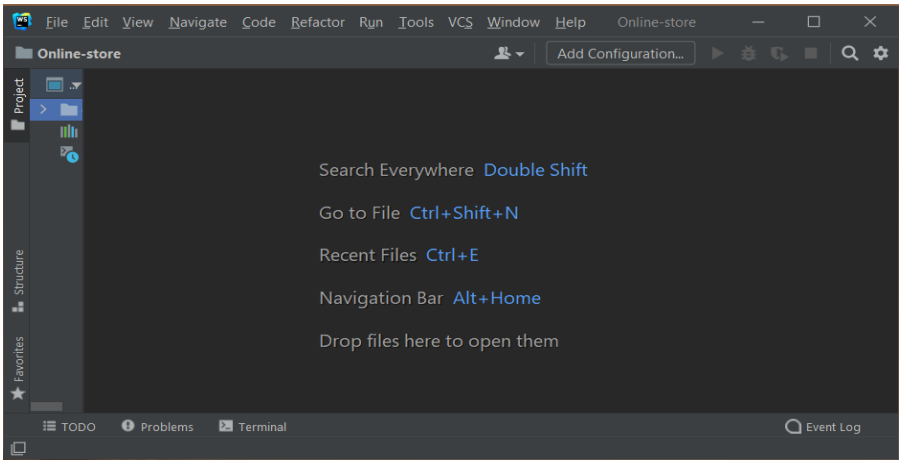


Рис. 2.3. Програмний засіб WebStorm

У новому проекті створюється папка server в корені якої створиться файл “index.js”. З цього файлу буде починатися запуск. В терміналі ініціалізується проект командою “npm init -y”. Після цього в папці має з’явитися файл “package.json”, в ньому відобразиться опис проекту та залежності котрі в ньому використовуються. Ці самі залежності потрібно встановити через “npm install”, таким чином додаються системи управління БД postgrest, фреймворк express, а також sequelize [15]. Після встановлення всі модулі можна побачити в dependencies (рис. 2.4.).

Рис. 2.4. Встановлені залежності

За допомогою require можна імпортувати модулі, таким чином імпортується express в “index.js”.

2.1.2. Підключення до бази даних

Конфігурація підключення до бази даних є одним з важливих етапів розробки, так як БД являє собою основну частину майже будь-якого додатку. До файлу “db.js” імпортуємо sequelize, після чого вже експортуємо на виході об’єкт який створюється з цього класу. В даному випадку потрібна передача користувача, який буде підключатися до бази даних. Для цього необхідне встановлене програмне забезпечення Postgre SQL [16], разом з СКБД автоматично встановлюється pg Admin (рис. 2.5.).

Рис. 2.5. Вікно запуску pg Admin

Pg Admin, дає можливість зручно керувати базами даних. Створюється нова БД, з назвою “online_store” та зберігається (рис. 2.6.). Після цього вона з’являється у списку.

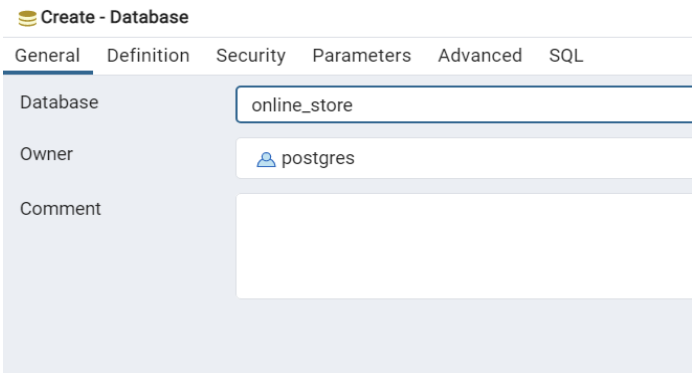


Рис. 2.6. Присвоєння назви для БД

Усі налаштування для підключення до бази даних виносяться в змінне оточення “.env”. Вказується створена назва БД, ім’я користувача під яким буде проходити підключення, а також пароль який задається в момент встановлення pg Admin, хост у режимі розробки вказується localhost, порт за замовчуванням 5432. Після цього потребується передача цих змінних в конструктор.

2.1.3. Побудова діаграми класів

У ході розробки серверної частини будується діаграма класів для бази даних. Для цього використовується draw.io – це популярний веб-інструмент для створення діаграм [17], який дозволяє користувачам створювати широкий спектр діаграм (рис. 2.7.).

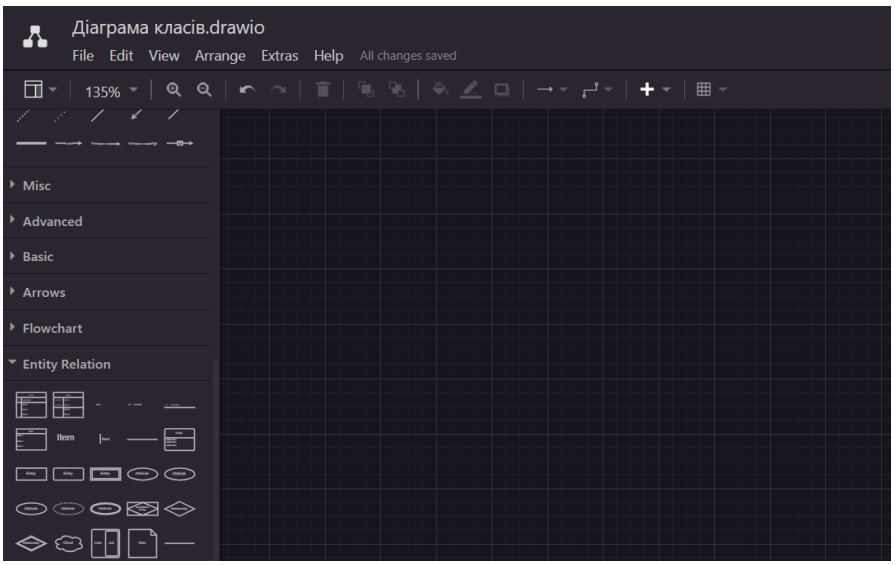


Рис. 2.7. Внутрішній інтерфейс draw.io

Знайшовши на панелі інструментів вкладку Entity Relation, обирається елемент з таблицею. Початок з опису сутності користувача, вказується назва таблиці user. У кожної сутності обов’язково повинен бути унікальний ідентифікатор, який однозначно визначає сутність – це id. Також у користувача буде email та пароль. Наступна сутність кошик користувача, який містить два поля. Це унікальний ідентифікатор та посилання на користувача, щоб розуміти кому цей кошик належить. Зв'язок між цими таблицями буде один до одного. Тобто, одному користувачу може належати тільки один кошик, в той час як один кошик може відноситись тільки до одного користувача. Наступна сутність товар, який має такі поля як id, найменування товару, опис товару, ціну, зображення та категорію як зовнішній ключ. Далі створюється сутність категорій товару, вони мають такі поля як id та зображення. Сутність категорії має зв'язок до сутності товару один до багатьох, одній категорії може належати декілька продуктів.

Діаграму класів (рис. 2.8.) потрібно перенести у проект та реалізувати схему того, як ці дані будуть дійсно зберігатись у базі даних. Створюється папка models, в якій створюється файл “models.js”. Імпортується об’єкт sequelize, в якому знаходиться клас DataTypes за допомогою якого описуються типи полів: string, integer і так далі.

Викликається функція define, передається об’єкт вказуючи назву моделі як перший параметр. Всередині об’єкта описуються поля які належать до моделі (рис. 2.9.). Для користувача перше поле id має тип integer, це буде первинний ключ і він

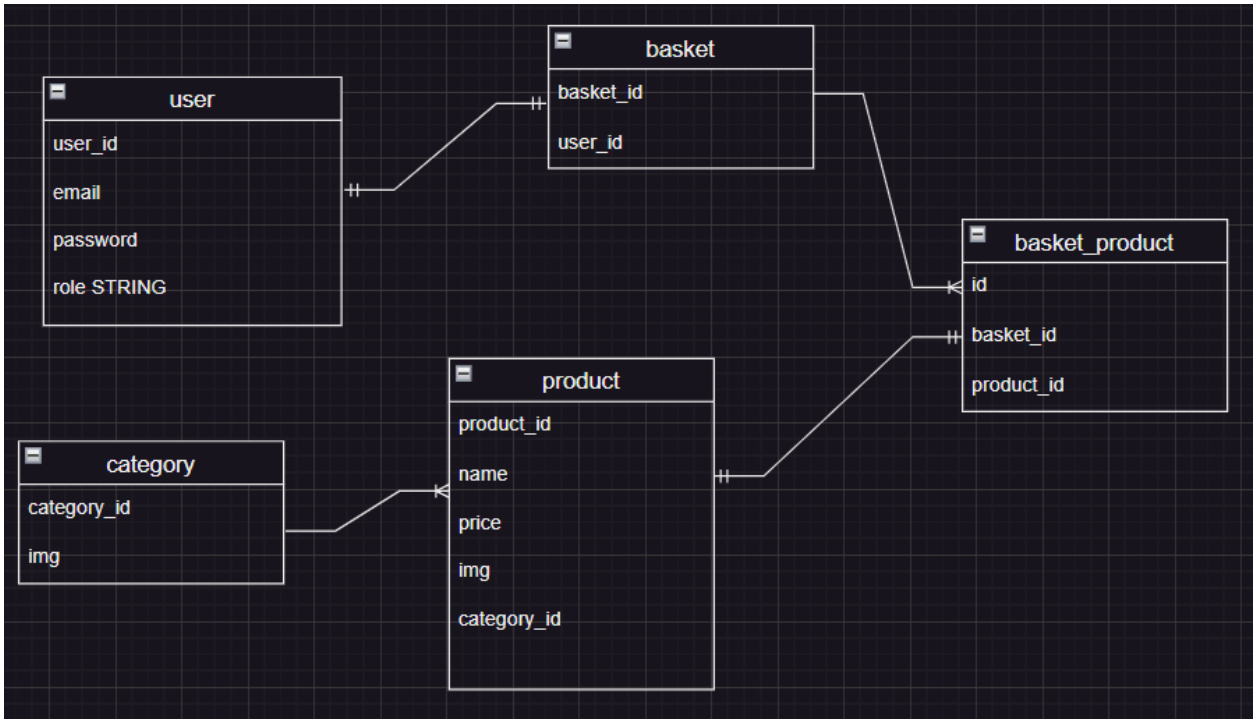


Рис. 2.8. Діаграма класів

буде автоматично інкрементувати, тобто при створенні кожного нового об’єкту id буде збільшуватись, наступне поле email має тип string, він має бути унікальним оскільки двох користувачів з однаковою поштою в системі не може бути, третє поле пароль типу string, він не має бути унікальним та може повторюватися у різних користувачів, останнє поле роль типу string та за замовчуванням це буде звичайний користувач.

```
models.js
1  const sequelize = require('../db')
2  const {DataTypes} = require('sequelize')
3
4  const User = sequelize.define( modelName: 'user', attributes: {
5      id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
6      email: {type: DataTypes.STRING, unique: true,},
7      password: {type: DataTypes.STRING},
8      role: {type: DataTypes.STRING, defaultValue: "USER"},
9  })
```

Рис. 2.9. Поля які належать до моделі користувача

Для моделі товару id має тип integer, найменування товару типу string, ціна типу integer та зображення буде типу string, звичайний рядок у якому буде зберігатися назва файлу з розширенням.

2.1.4. JWT токени

Повноцінна реєстрація та авторизація реалізуються по JWT токену (рис. 2.10.).

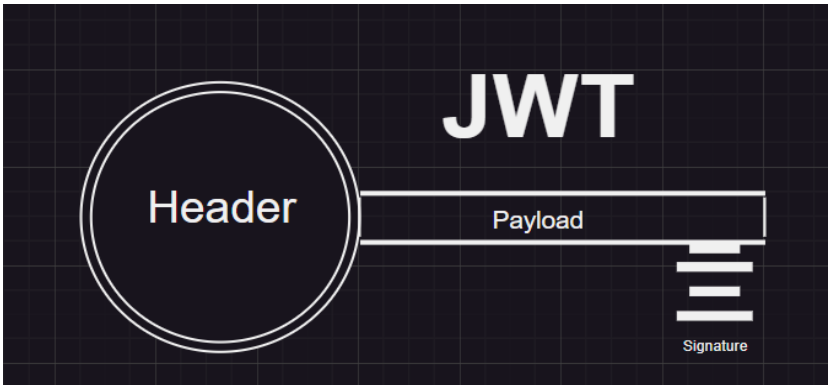


Рис. 2.10. Представлення JWT

JWT токен або токен доступу це простий рядок розділений точками на три частини, здебільшого необхідна центральна частина це Payload, туди будуть ховатись дані користувача, такі як email, id та роль [18]. Після того як користувач зареєструється введе email і пароль, для нього згенерується подібний токен, всі його дані вшиються в середину. Ці дані ніяк не шифруються, вони в принципі ніякої секретної інформації не несуть і на клієнті можна спокійно їх розшифровувати, але перевірити на те валідний токен чи ні, можливо тільки за допомогою секретного ключа, який оголошується на сервері та його за наміром вже ніхто не повинен знати. Загалом користувач вводить email, пароль. В першу чергу перевіряється, чи існує користувач з таким email у системі. Якщо так, то порівнюється пароль, який знаходиться в базі даних, з паролем який написав користувач. Якщо ці дані збігаються, генерується JWT токен і відправляється на клієнт. Потім токен прикріплюється до запитів, де необхідна авторизація.

2.2. Розробка клієнтської частини – Frontend

Розробка фронтенду веб-застосунку включає створення і розміщення всіх клієнтських компонентів, які відповідають за візуальну частину додатку і взаємодію з користувачем. У розробці фронтенду можуть використовуватись різні технології та інструменти, але одним з найпопулярніших варіантів є використання бібліотеки React.

Необхідні інструменти для розробки фронтенду з React включають: Редактор коду, наприклад Visual Studio Code або WebStorm. Node.js та пакетний менеджер npm (або Yarn) для установки залежностей та розробки сервера розробки. Система контролю версій, така як Git, для збереження та керування кодом. Інші утиліти та пакети, такі як Create React App або Next.js, для спрощення створення проекту та розробки функціональності.

При розробці клієнтської частини, яка відображатиметься вже у браузері, буде застосований наступний стек технологій:

- React JS – бібліотека JavaScript;
- React bootstrap – надає попередньо розроблені компоненти та стилі, які можна легко використовувати. Вони дотримуються принципів React, що забезпечує легку інтеграцію, налаштування та динамічний рендеринг на основі стану програми;
- Axios – бібліотека для запитів на сервер;
- React-router-dom – використовується для навігації сторінками;
- Mobx – використовується в якості стейт менеджеру [19].

Створюється нова папка client. Та за допомогою утиліти “npx create-react-app” розвертається React застосунок. Тепер встановлюються необхідні залежності “npm i axios react-router-dom mobx mobx-react-lite”, axios знадобиться для того, щоб надсилати необхідні запити на сервер, react-router-dom, mobx та mobx react lite, для того щоб зв’язати mobx з функціональними компонентами React. Також буде використовуватись React bootstrap, застосуємо скрипт для встановлення “npm install react-bootstrap bootstrap”.

Файл “App.js” буде основним компонентом додатку, напишемо working і командою “npm start” запустимо React додаток (рис. 2.11.).

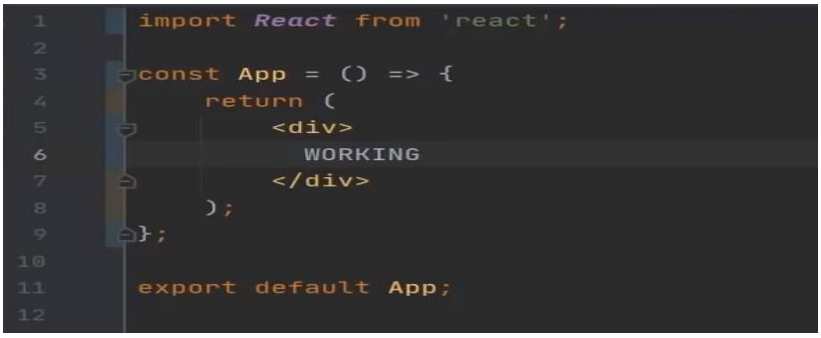


Рис. 2.11. Відображення файлу “App.js”

Відкривається браузер, бачимо напис working значить додаток працює (рис. 2.12.). Після задаємо структуру програми, створюємо всі папки які надалі знадобляться. Перша папка store, в якій будемо взаємодіяти з Mobx та зберігати дані. Друга папка pages, в якій будуть розташовані корневі компоненти, які будуть сторінками. Також папка components з компонентами, де будуть розташовані усілякі навігаційні панелі та інше.

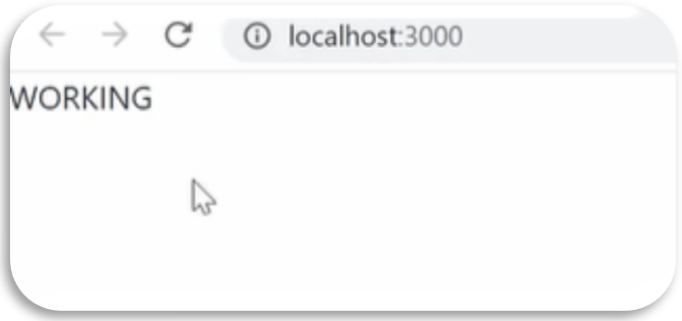


Рис. 2.12. Перевірка праці додатку

Відразу ж створюються сторінки у папці pages. У файлі “Auth.js” створюється сторінка авторизації. Прописується “rsc” та натискається кнопка Tab, при цьому відбувається розгортання компоненту.

Створення сторінки адмін панелі, з якої адміністратор буде розташовувати товари.

Створюється “Shop.js”, це основна сторінка в якій будуть картки з товарами та посторінковий вивід (рис. 2.13.).

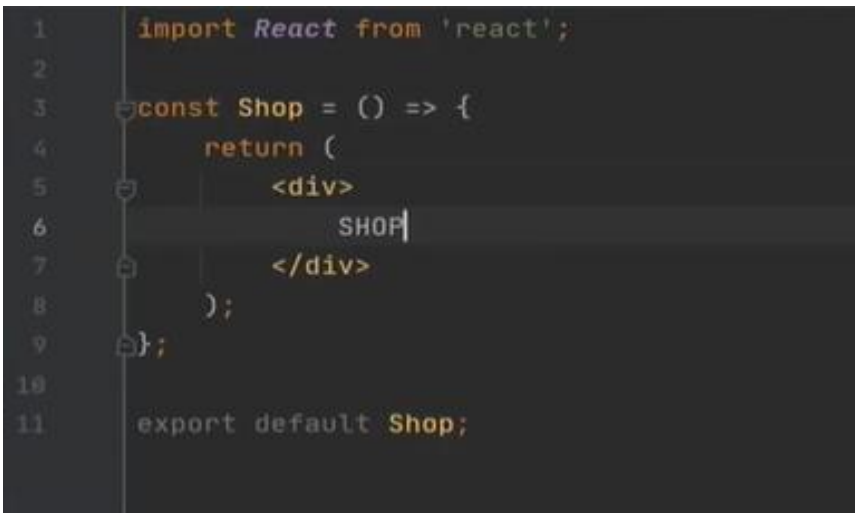


Рис. 2.13. Відображення файлу “Shop.js”

Реалізується навігацію по сторінкам. При цьому створюється компонент “AppRouter”, в ньому буде описана логіка навігації по сторінкам. На деякі сторінки зможе зайти будь-який користувач, на деякі – тільки авторизовані. З пакету react-router-dom, який вже встановлювали, можна використовувати декілька компонентів для роботи з маршрутизацією в React: switch, route, redirect.

Компонент Switch використовується для вибору першого співпадіння маршруту відповідно до шляху URL. Він дозволяє відображати тільки один компонент Route або Redirect одночасно.

Компонент Route визначає шляхи URL та пов'язані з ними компоненти, які мають бути відображені при збігу шляху. Він приймає два головні атрибути: path (шлях URL) та component (компонент, який має бути відображений при збігу шляху).

Компонент Redirect використовується для перенаправлення користувача на інший шлях URL. Він може бути використаний для реалізації автоматичного перенаправлення або управління перенаправленням у вашому додатку.

Для того щоб в принципі навігація по сторінці була можлива, весь наш додаток потрібно обернути в BrowserRouter, який також імпортується з react-router-dom (рис. 2.14.).

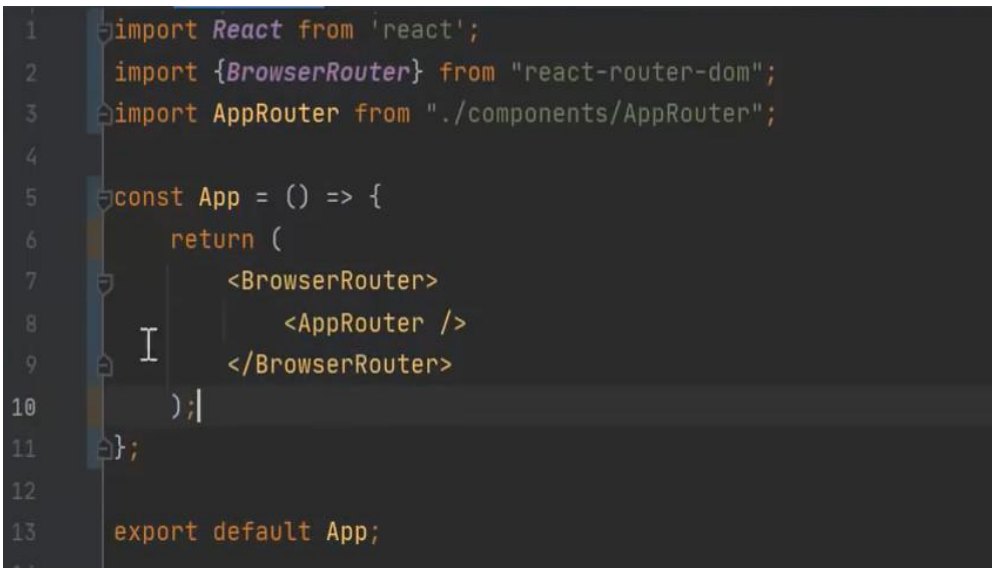


Рис. 2.14. Обернення додатку в BrowserRouter

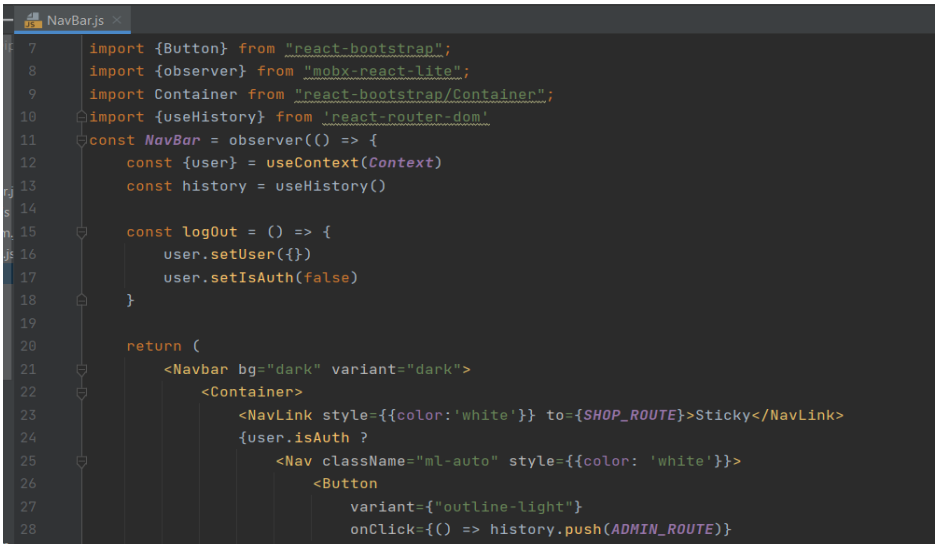
BrowserRouter - це компонент, який використовується для навігації та управління маршрутами в React-додатку за допомогою браузерного API. Він надає контекст для дочірніх компонентів, щоб вони могли отримати доступ до поточного URL та виконувати дії, пов'язані з маршрутизацією. Основна роль BrowserRouter полягає в тому, щоб взаємодіяти з історією браузера та оновлювати URL згідно визначених маршрутів [20].

Створення файла “routes.js”, в ньому будуть описані усі маршрути по сторінкам які є в нашому застосунку. Прописуються два масиви: authRoutes, який буде мати список тих сторінок до котрих має доступ авторизований користувач та publicRoutes на маршрути якого може перейти будь-який користувач. Обидва масиви експортуємо та добавляємо об’єкт. У кожного об’єкта буде шлях, посилання за якою та чи інша сторінка буде відпрацьовувати, а також компонент, безпосередньо сама сторінка. По URL /admin, буде викликано компонент адміністратора (рис. 2.15.).

```
import Admin from "./pages/Admin";

export const authRoutes = [
    {
        path: '/admin',
        Component: Admin
    }
]

export const publicRoutes = [

]
```

Рис. 2.15. Виклик компонента адміністратор

Створення файла з назвою “NavBar.js”, в який імпортуються компоненти з bootstrap для графічного оформлення додатку (рис. 2.16.). Bootstrap має велику колекцію готових компонентів, таких як кнопки, форми, навігація, картки, модальні вікна та інші. Він дозволяє налаштовувати стилі компонентів шляхом використання Sass-змінних або CSS-класів. Ви можете легко змінювати кольори, шрифти, відступи та інші параметри відповідно до ваших потреб.



NavBar.js

```
import {Button} from "react-bootstrap";
import {observer} from "mobx-react-lite";
import Container from "react-bootstrap/Container";
import {useHistory} from 'react-router-dom'
const NavBar = observer(() => {
    const {user} = useContext(Context)
    const history = useHistory()

    const logOut = () => {
        user.setUser({})
        user.setIsAuth(false)
    }

    return (
        <Navbar bg="dark" variant="dark">
            <Container>
                <NavLink style={{color:'white'}} to={SHOP_ROUTE}>Sticky</NavLink>
                {user.isAuth ?
                    <Nav className="ml-auto" style={{color: 'white'}}>
                        <Button
                            variant={"outline-light"}
                            onClick={() => history.push(ADMIN_ROUTE)}
```

Рис. 2.16. Імпорт компонентів з bootstrap

Bootstrap є популярним вибором для розробників, оскільки він допомагає прискорити розробку фронтенду, забезпечує консистентний дизайн та широкі

можливості налаштування. Він також має велику активну спільноту, що підтримує його розвиток та надає документацію та приклади використання.

Для поєднання серверної та клієнтської частини створюється папка http всередині якої створюється файл “index.js” та налаштується axios. Імпортуємо його та створюємо два інстанси: перший для звичайний запитів які не потребують авторизації, даємо йому назву “host”, у другому інстансі для кожного запита автоматично буде підставлятись Header авторизація та додаватися токен.

Після того, як користувач відкрив додаток у браузері або натиснув на будь-яку кнопку має відбутися певна дія та повинні отримати дані з сервера. Сервер розташований за якоюсь адресою, наприклад http://server.com. Надсилаючи на цю адресу запит, сервер взаємодіє з базою даних, обробляє дані, відбуваються певні маніпуляції, після чого сервер повертає ці дані на клієнт та користувач вже отримує їх у вигляді інформації, картки якогось товару, список користувачів тощо.

На сервері визначено список ендпоінтів, за допомогою яких можемо з цим сервером взаємодіяти. Найчастіше взаємодія відбувається за протоколом http. Http протокол має 4 основні методи, це GET для отримання даних, POST для створення даних, PUT для оновлення даних і DELETE для того, щоб їх видалити.

Взаємодія з сервером за допомогою HTTP-протоколу та використання різних методів запитів веб-додатком дозволяє користувачам здійснювати різноманітні дії і отримувати необхідні дані.

Метод GET використовується для отримання даних з сервера. Цей метод найчастіше використовується для запиту інформації про продукти, категорії, профілі користувачів тощо.

Метод POST використовується для створення нових даних на сервері. Наприклад, користувач може використовувати цей метод для додавання товару до свого кошика або для створення нового замовлення.

Метод PUT використовується для оновлення існуючих даних на сервері. Користувач може використовувати цей метод для оновлення свого профілю, зміни налаштувань або внесення змін до замовлення.

Метод DELETE використовується для видалення даних з сервера. Наприклад, користувач може використовувати цей метод для видалення товару зі свого кошика або для скасування замовлення.

Кожен ендпоінт сервера може мати свою власну функціональність та обробляти різні типи запитів для різних дій. Це дозволяє веб-додатку реалізувати широкі можливості і надати користувачам зручний та функціональний інтерфейс для взаємодії з додатком та отримання необхідних даних.

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділі була проведена повноцінна фул стек розробка, що означає розробку як бекенду, так і фронтенду веб-додатку. Бекенд відповідає за обробку запитів і взаємодію з базою даних, тоді як фронтенд відповідає за візуальне представлення і взаємодію з користувачем.

У процесі розробки бекенду була використана технологія Node.js, яка дозволяє виконувати JavaScript код на серверній стороні. За допомогою Node.js була побудована серверна архітектура, реалізовані API для обробки запитів від клієнтів і взаємодії з базою даних. Для комунікації між клієнтом і сервером використовувалися HTTP-запити.

Одним з ключових елементів бекенду була база даних, в даному випадку використана Postgre SQL. Вона забезпечує збереження та організацію даних про стікери, категорії, користувачів тощо.

Для фронтенду була використана бібліотека React, яка дозволяє створювати динамічні та інтерактивні користувацькі інтерфейси. Були розроблені компоненти, які відповідають за відображення інформації про стікери, категорії, кошик покупок та оформлення замовлення. React також забезпечує ефективне оновлення візуальної частини додатку при зміні даних.

Під час розробки були використані сучасні технології та інструменти, такі як React Router для навігації між сторінками, Axios для взаємодії з сервером через HTTP-запити, а також бібліотека Bootstrap для швидкої та зручної стилізації з готовими стилями та компонентами.

Завдяки фул стек розробці було досягнуто повноцінної функціональності веб-додатку і забезпечено зручний інтерфейс для користувачів.

РОЗДІЛ 3
ФУНКЦІОНУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ СТІКЕРІВ

3.1. Основні етапи функціонування інтернет-магазину

Функціонування інтернет-магазину зазвичай базується на наступних основних етапах:

1. Реєстрація та авторизація користувачів. Користувачі можуть створювати облікові записи в інтернет-магазині, вводити особисті дані та обирають метод авторизації. Це може бути звичайний логін та пароль, соціальна авторизація (наприклад, через Facebook або Google) або інші методи.

2. Пошук та перегляд товарів. Користувачі можуть шукати товари в інтернет-магазині за допомогою пошукової функції або шляхом навігації по категоріях. Вони можуть переглядати детальну інформацію про товари, таку як опис, зображення, ціна, відгуки попередніх покупців тощо.

3. Додавання товарів до кошика. Користувачі можуть вибирати товари, які вони бажають придбати, і додавати їх до свого кошика. Вони можуть вказувати кількість товарів, вибирати варіанти (наприклад, розмір, колір тощо) та переглядати загальну суму замовлення.

4. Оформлення замовлення. Користувачі можуть перевіряти та редагувати зміст свого кошика перед оформленням замовлення. Вони повинні надати необхідну інформацію для доставки, вибрати метод оплати та підтвердити своє замовлення перед його остаточним оформленням.

5. Оплата та обробка замовлення. Після оформлення замовлення користувачі мають здійснити оплату за допомогою вибраного методу. Після успішної оплати інтернет-магазин обробляє замовлення, генерує підтвердження замовлення та надсилає його користувачеві.

6. Доставка та відстеження замовлення. Інтернет-магазин забезпечує доставку замовлених товарів до вказаної адреси. Користувачі можуть відстежувати статус свого замовлення, отримувати сповіщення про стан доставки та відслідковувати його відправлення.

7. Обробка повернень та обмінів. Якщо користувачі не задоволені отриманими товарами, вони можуть звернутися до інтернет-магазину з проханням про повернення або обмін. Інтернет-магазин надає процедуру повернення та обміну товарів та забезпечує обробку таких запитів.

8. Забезпечення безпеки та конфіденційності. Інтернет-магазин повинен забезпечити безпеку особистої інформації користувачів, такої як дані платіжних карток та адреси доставки. Він повинен використовувати шифрування, захист від шахрайства та інші заходи безпеки для запобігання несанкціонованому доступу до даних користувачів.

3.2. Перевірка функціонування веб-додатку через адміністратора

Для того, щоб перейти на веб-сторінку інтернет-магазину, потрібно в консолі кореневої папки проекту ввести “npm run dev” (рис. 3.1.).

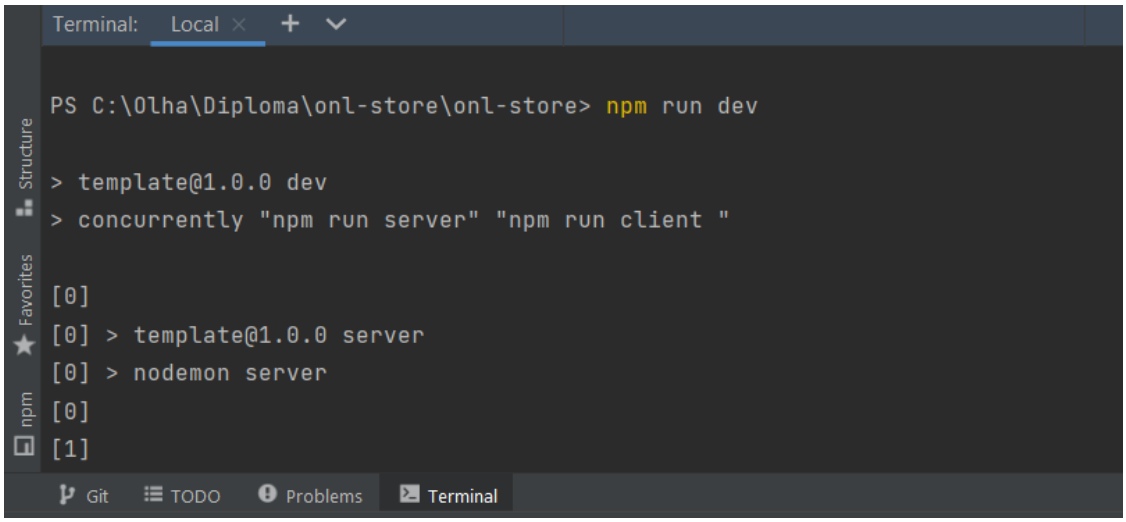


Рис. 3.1. Введення команди “npm run dev” у консоль

Це потрібно для того, щоб запустити сервер та клієнт. Чекаємо на відповідь від терміналу, чи вдалося зробити запуск.

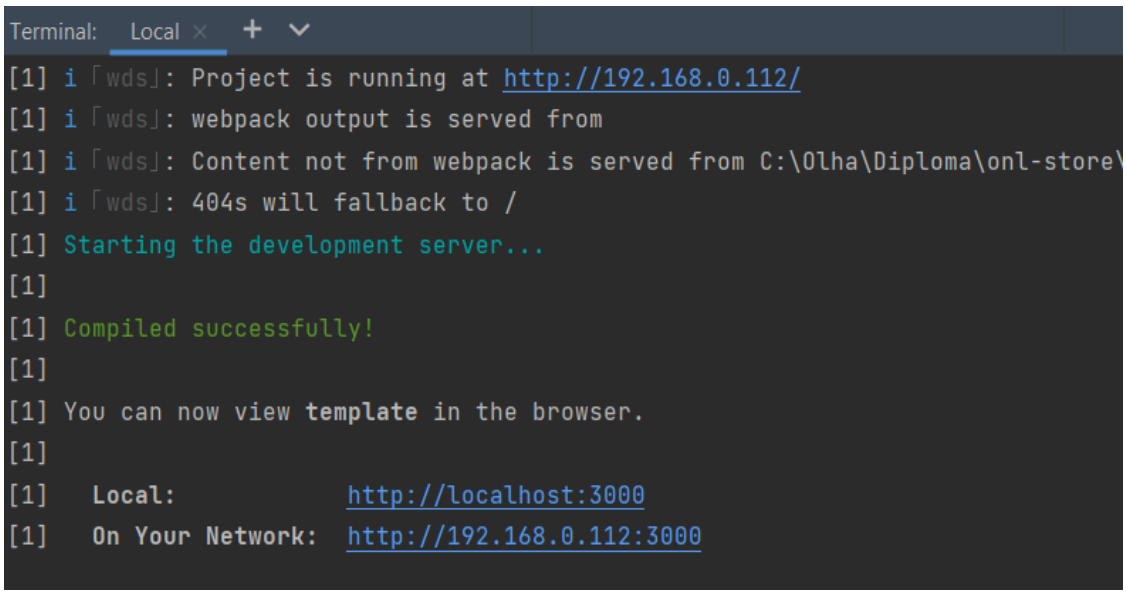


Рис. 3.2. Відклик терміналу на запит

Таке повідомлення свідчить про те, що запуск є успішним (рис. 3.2.). Можна переходити у браузері за посиланням http://localhost:3000, та інтернет-магазин має працювати (рис. 3.3.).

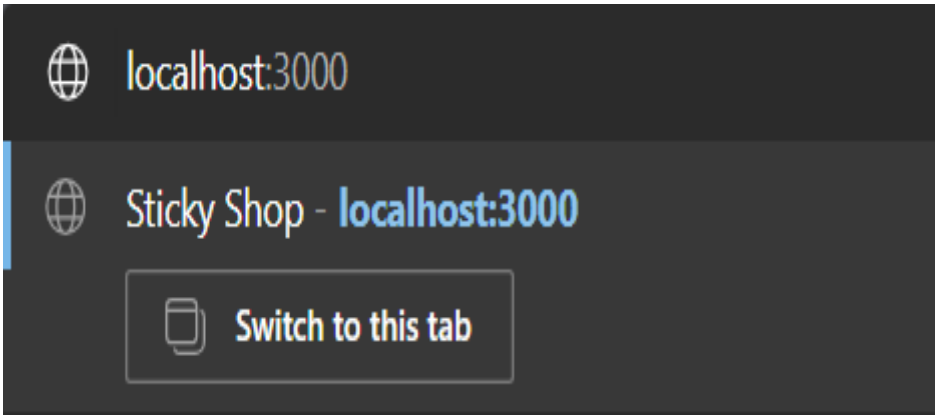


Рис. 3.3. URL посилання

В ході користування веб-застосунком, для того щоб клієнти мали змогу замовити товар, адміністратор повинен їх добавити в інтернет-магазин.

Таким чином, на початку розглянемо використання додатку з точки зору адміністратора.

Щоб виконати вхід у режимі адміністратора, потрібно у URL посиланні вказати це: http://localhost:3000/admin. Адміністратор вже зареєстрований в системі, тож пройдемо авторизацію (рис. 3.4.).

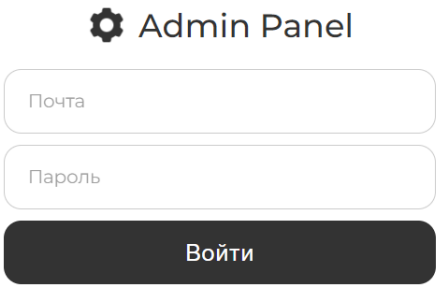


Рис. 3.4. Панель авторизації

Перш ніж увійти в систему, навмисно припустимо помилку в написанні електронної адреси. Додаток зазвичай перевіряє введені дані на коректність. У випадку помилки, коли електронна адреса не відповідає очікуваному формату система може повідомити користувача про некоректний формат електронної адреси.

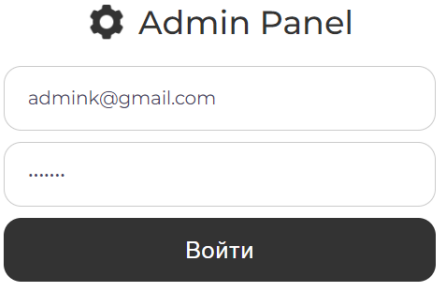


Рис. 3.5. Введення некоректних даних для перевірки обробки

Було додано один зайвий символ задля того, щоб перевірити процес обробки помилки (рис. 3.5.).

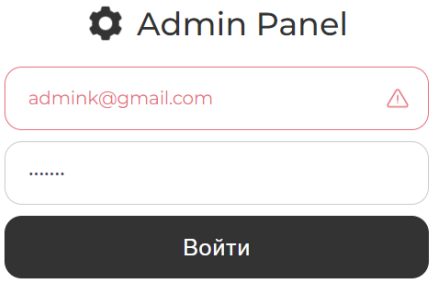


Рис. 3.6. Розпізнавання помилки системою

Під час роботи додаток успішно функціонує та виявляє помилку у введеному електронному адресі користувача, оскільки не знайдено відповідного запису в системі з такою поштовою скринькою. Це може означати, що користувач не зареєстрований в системі або ввів неправильну інформацію (рис. 3.6.).

Після правильного введення даних (рис. 3.7.), система успішно розпізнає користувача і дозволяє йому увійти до середовища інтернет-магазину.

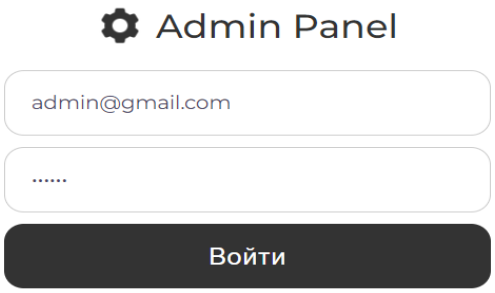


Рис. 3.7. Введення коректних даних

Увійшовши в систему, користувач отримує доступ до розширеного функціоналу та особистих налаштувань (рис. 3.8.).

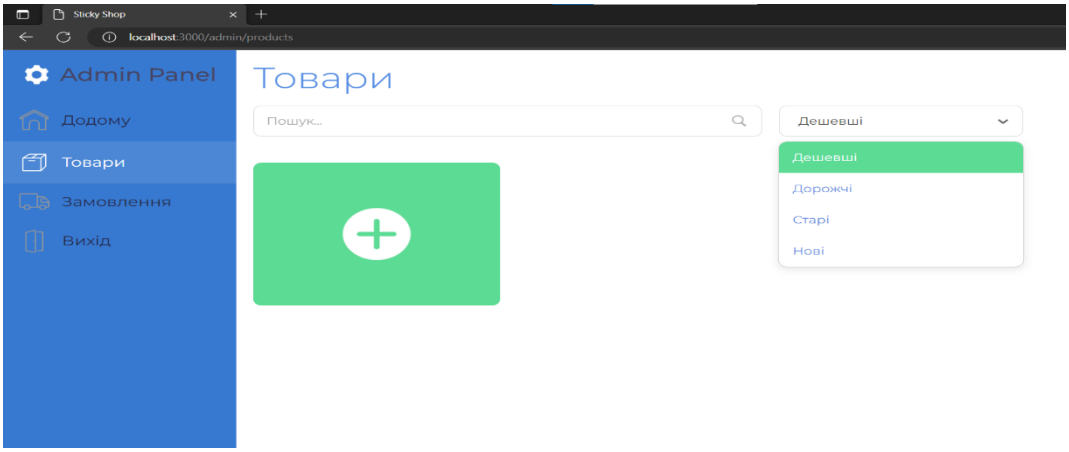


Рис. 3.8. Зовнішній вигляд панелі адміністратора

Функціонування інтернет-магазину під керуванням адміністратора може включати наступні аспекти в залежності від потреб та специфіки конкретного магазину:

1. Управління товарами. Адміністратор може додавати нові товари до магазину, встановлювати ціни, описи, зображення та інші характеристики товарів. Він також може редагувати і видаляти існуючі товари, оновлювати їх інформацію та керувати наявністю товарів на складі.

2. Керування категоріями та тегами. Адміністратор може створювати та редагувати категорії товарів для організації асортименту. Він також може встановлювати теги для товарів, що полегшує пошук та навігацію для користувачів.

3. Управління замовленнями. Адміністратор отримує доступ до замовлень, зроблених користувачами, і може переглядати їх деталі, такі як вміст кошика, адреса доставки, статус оплати та статус доставки. Він також може оновлювати статус замовлення та надсилати повідомлення користувачам про стан їх замовлення.

4. Керування користувачами. Адміністратор має можливість керувати користувачами, такими як блокування акаунтів, скасування замовлень, видалення акаунтів тощо. Він також може переглядати і редагувати особисті дані користувачів за їх згодою.

5. Статистика та звіти. Адміністратор може отримати доступ до аналітичних даних, таких як кількість замовлень, обсяг продажів, найпопулярніші товари тощо. Це допомагає адміністратору аналізувати продажі та впливати на стратегію розвитку магазину.

6. Безпека та автентифікація. Адміністратор має спеціальні привілеї та доступ до адміністративних функцій, тому необхідно забезпечити безпеку та захист доступу до адміністративної панелі. Це може включати автентифікацію двофакторним підтвердженням, обмеження доступу до адміністративних функцій лише для авторизованих адміністраторів та інші заходи безпеки.

В інтернет-магазині стікерів, з панелі адміністратора можна додати нові товари, а також переглянути існуючі замовлення (рис. 3.8.). Перш за все слід розпочати з додавання товарів у інтернет-магазин (рис. 3.9.).

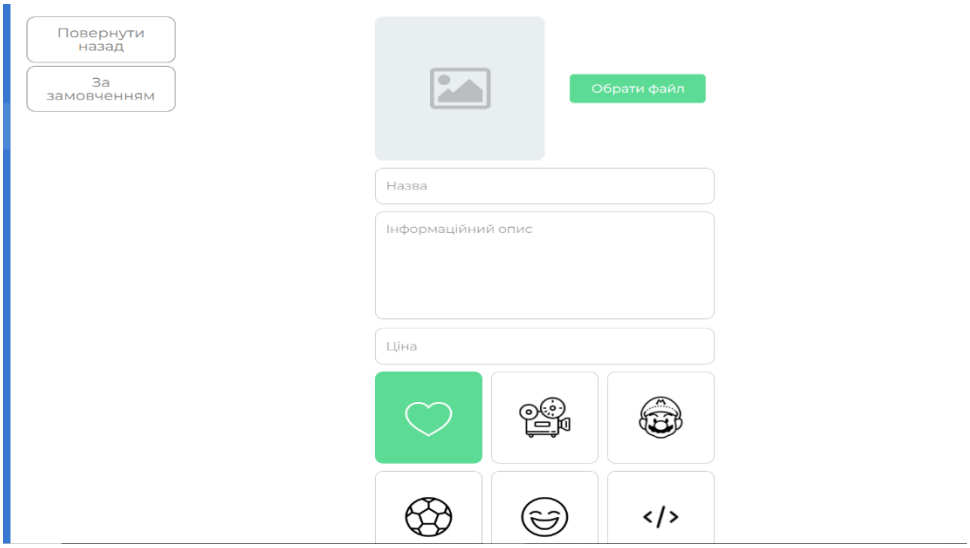


Рис. 3.9. Пуста форма додавання товару

Розглянувши форму для додавання нового асортименту стікерів, можна побачити, що вона містить наступні поля для заповнення:

1. Назва стікера. Це поле призначене для введення назви нового стікера.

2. Інформація про стікер. Тут можна ввести детальну інформацію про стікер, наприклад, його характеристики, особливості, матеріал тощо.

3. Вказується ціна стікера.

4. Можна вибрати категорію, до якої відноситься стікер.

Крім того, у формі є кнопка "Обрати файл" (рис. 3.10), яка дозволяє завантажити зображення товару. Це дає можливість додати візуальне представлення стікера для покупців.

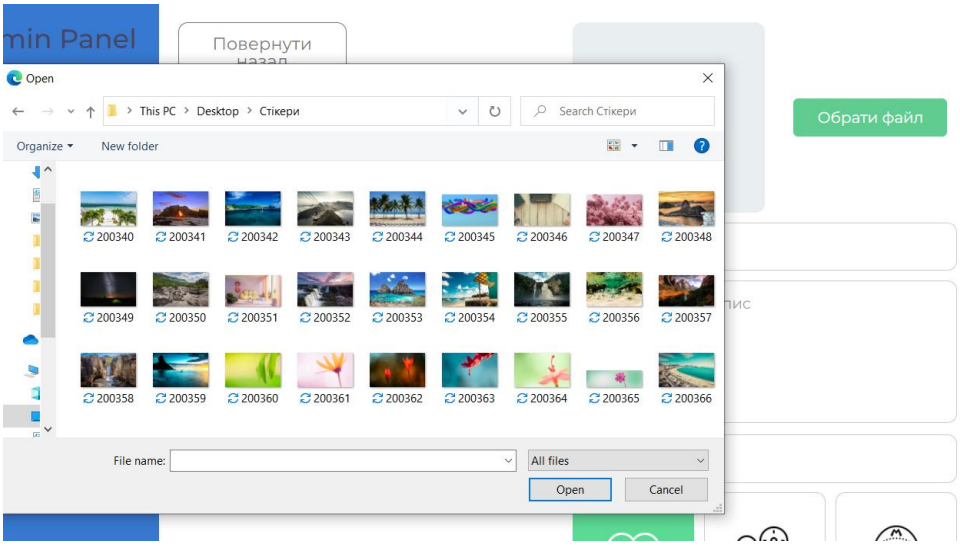


Рис. 3.10. Вибір зображення для завантаження

Після того, як обрали зображення, переходимо до заповнення інших полів з інформацією (рис. 3.11.).

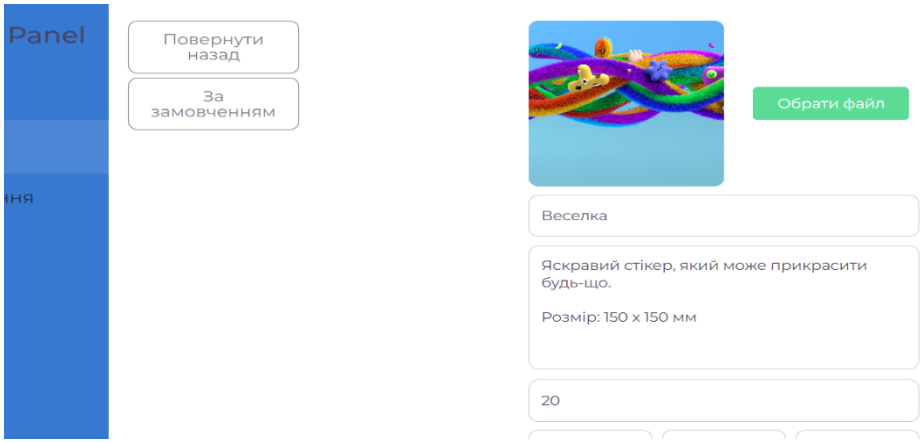


Рис. 3.11. Заповнена форма товару

У нашому додатку створено шість різних категорій (рис. 3.12.), серед яких потрібно обрати ту, до якої відноситься даний стікер. Це дозволяє логічно структурувати асортимент товарів і зробити їх пошук та навігацію зручними для користувачів.

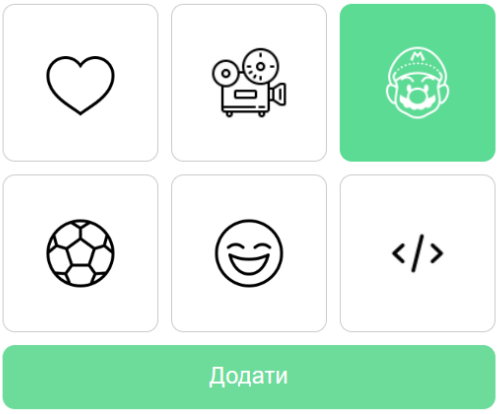


Рис. 3.12. Категорії стікерів

Після заповнення усіх необхідних полів та завантаження зображення, користувач може натиснути кнопку "Додати", щоб зберегти новий асортимент стікерів в системі.

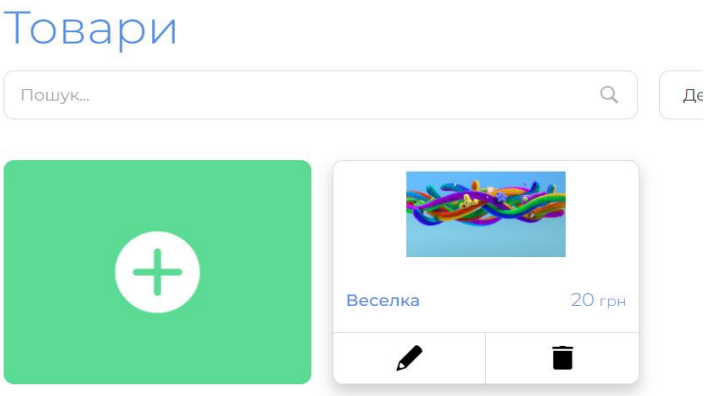


Рис. 3.13. Можливі операції з товаром

Після успішного додавання товару до системи, маємо можливість відредагувати його або видалити з бази даних, якщо потрібно. Це дає нам гнучкість і контроль над асортиментом товарів в інтернет-магазині. Можемо змінити назву, інформацію, ціну, категорію або навіть замінити зображення товару (рис. 3.13.). Якщо товар втратив актуальність або вже не доступний, можемо видалити його з бази даних, щоб підтримувати актуальний та оновлений каталог товарів для наших користувачів.

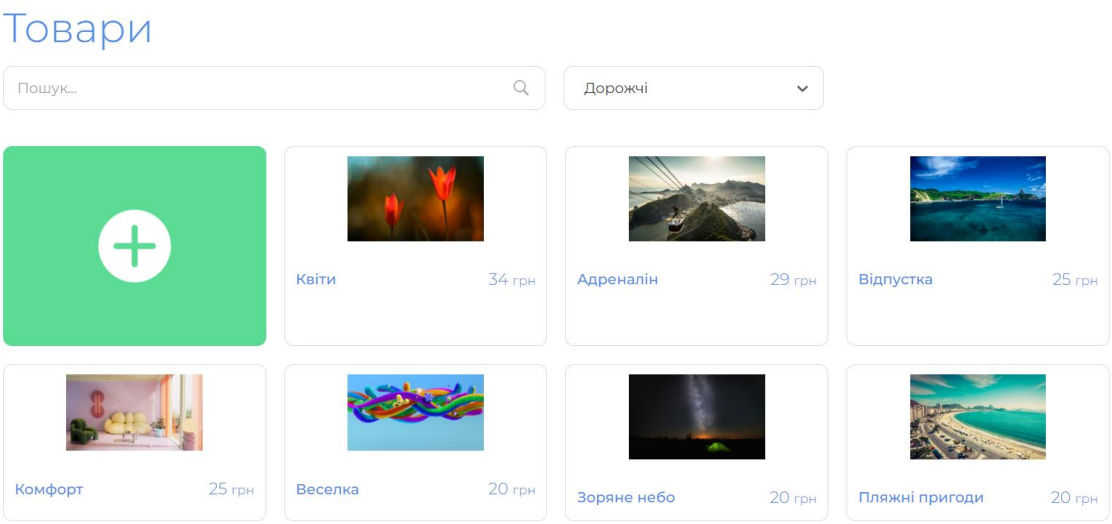


Рис. 3.14. Перелік нового товару

Завершивши процес додавання товару (рис. 3.14.), адміністратор буде відслідковувати вхідні замовлення. Кожне замовлення, що надійде в систему, буде відображатися у списку замовлень адміністратора. Для кожного замовлення будуть доступні такі дані, як ім'я клієнта, контактна інформація, вибрані товари та їх кількість, адреса доставки та будь-які додаткові вказівки.

3.3. Використання додатку звичайним користувачем

Перейшовши за посиланням, користувач має зареєструватись (рис. 3.15.).

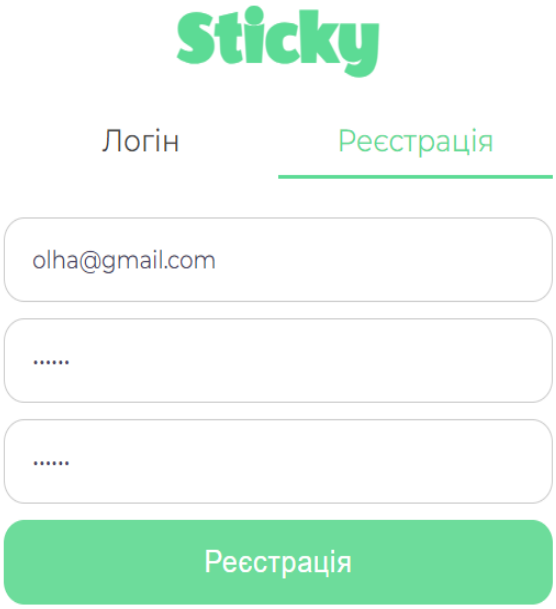


Рис. 3.15. Вікно реєстрації

Після успішної реєстрації, користувач автоматично перенаправляється на головну сторінку інтернет-магазину (рис. 3.16.). Тут він зустрічає різноманітні товари і може здійснювати пошук, переглядати категорії або просто прогортати сторінку, щоб оглянути всі доступні товари. Кожен товар супроводжується зображенням, назвою, описом і ціною.

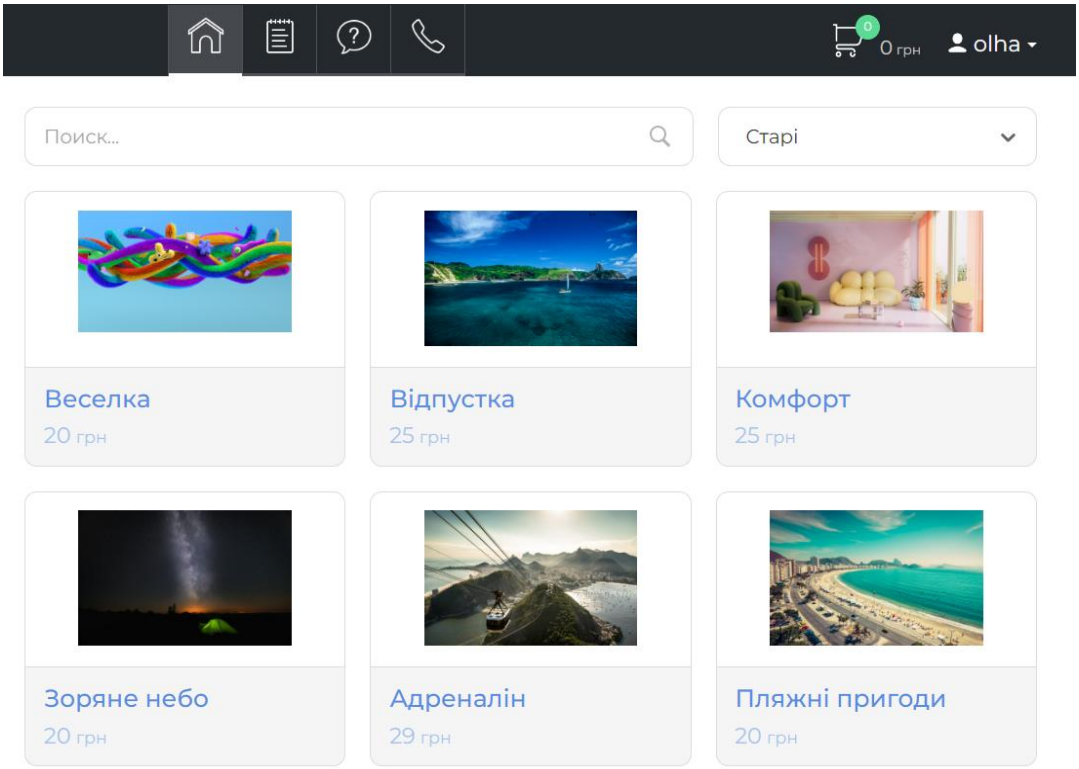


Рис. 3.16. Головна сторінка інтернет-магазину стікерів

Користувач може скористатися фільтрами для звуження вибору товарів за категоріями, ціновим діапазоном або іншими параметрами. При виборі конкретного товару, користувач може переглянути його докладну інформацію, включаючи властивості, опис та відгуки попередніх покупців.

Користувач має можливість додати бажаний товар до свого кошика, вказати кількість одиниць і вирішити, продовжувати покупки або перейти до оформлення замовлення (рис. 3.17.).

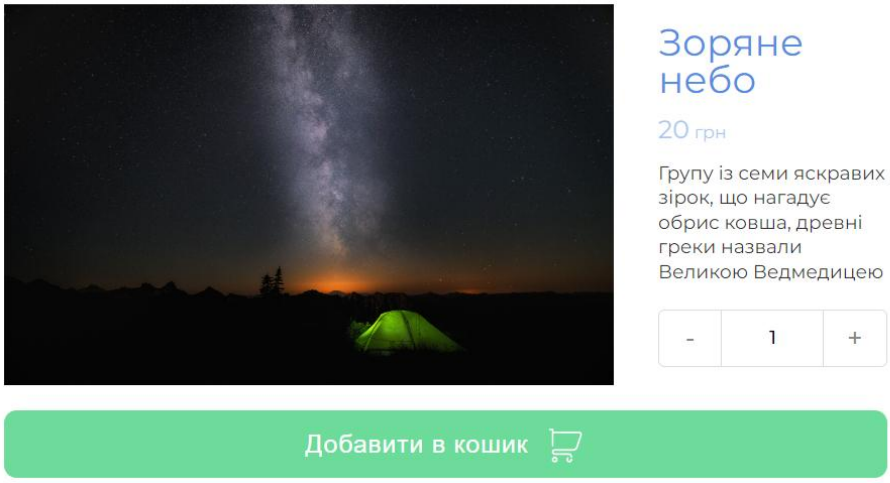


Рис. 3.17. Докладна інформація товару

На сторінці кошика користувач може переглянути всі додані товари перед оформленням замовлення (рис. 3.18.). Кожен товар буде відображатися з назвою, зображенням, ціною та кількістю, яку користувач вибрав. Користувач матиме можливість змінити кількість товару, оновивши значення поля з кількістю або видаливши товар повністю з кошика.

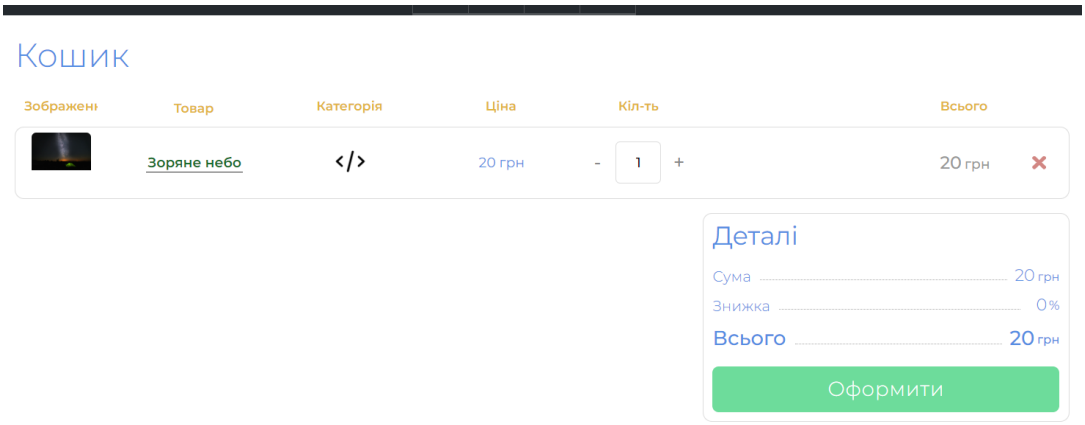


Рис. 3.18. Кошик користувача

Якщо користувач вже визначився зі своїм замовленням, він може перейти до процесу оформлення замовлення, натиснувши відповідну кнопку. Після цього він буде перенаправлений на сторінку оформлення замовлення, де буде потрібно вказати необхідні дані для доставки, обрати спосіб оплати та підтвердити замовлення.

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі було розглянуто основні аспекти роботи та функціональні можливості інтернет-магазину стікерів, розробленого за допомогою Node.js та React. Користувачі мають можливість створити свій обліковий запис в магазині, вказавши необхідні особисті дані та електронну пошту. Після успішної реєстрації користувачі можуть використовувати свої облікові дані для входу в систему.

Один з основних функціональних елементів інтернет-магазину - це можливість користувачів переглядати доступний асортимент стікерів. Кожен товар включає назву, опис, ціну, зображення та належить до певної категорії. Після обрання необхідних товарів, користувачі мають можливість додавати їх до свого кошика. Вони можуть вказувати бажану кількість товару, а також змінювати або видаляти товари з кошика.

Коли користувачі готові зробити покупку, вони можуть оформити замовлення. Вони можуть переглянути загальну суму замовлення, яка автоматично розраховується на основі цін та кількостей товарів у кошику. Крім того, користувачі можуть вказати свої контактні дані та підтвердити замовлення.

Окремий розділ системи призначений для адміністратора, який має розширені можливості. Він може додавати нові товари до магазину, редагувати існуючі товари та відслідковувати вхідні замовлення, що надходять в систему.

Загалом, розроблений інтернет-магазин стікерів на базі Node.js та React надає користувачам зручну та функціональну платформу для перегляду та покупки стікерів. Завдяки реєстрації, кошику, можливості оформлення замовлень та адміністраторському доступу, магазин забезпечує ефективну роботу та задоволення потреб користувачів.

ВИСНОВКИ

У кваліфікаційній роботі було проведено розробку веб-додатку інтернет-магазину для стікерів з використанням сучасних технологій веб-розробки. Проект демонструє значимість веб-додатків у сучасному цифровому світі, які забезпечують інтерактивні функції та послуги користувачам через веб-браузери та Інтернет.

Розробка веб-додатків включає два ключові компоненти: фронтенд і бекенд. Фронтенд використовує бібліотеку React для створення динамічного та інтерактивного користувацького інтерфейсу. Бекенд заснований на технології Node.js, яка дозволяє виконувати JavaScript код на серверній стороні. Це дозволяє побудувати серверну архітектуру, реалізувати API для обробки запитів від клієнтів та взаємодію з базою даних.

Node.js має велику та активну екосистему бібліотек та модулів, які спрощують розробку та розширення додатків. Існує велика кількість сторонніх модулів, доступних через менеджер пакетів npm, що дозволяє розробникам ефективно використовувати готові рішення та функціональність. Одна з головних переваг Node.js полягає у його здатності обробляти багатопотоковість за допомогою асинхронних операцій та подій. Це дозволяє побудувати ефективні та швидкодіючі додатки, які можуть обробляти багато запитів одночасно без блокування виконання коду.

У цьому проекті для комунікації між клієнтом і сервером використовуються HTTP-запити, а база даних Postgre SQL забезпечує збереження та організацію даних про стікери, категорії, користувачів та інше. Застосування сучасних технологій та інструментів, таких як React Router, Axios і Bootstrap, сприяє ефективній навігації між сторінками, взаємодії з сервером та швидкій та зручній стилізації інтерфейсу.

Веб-додаток інтернет-магазину для стікерів має функціонал, що дозволяє користувачам переглядати асортимент товарів, додавати їх до кошика, змінювати кількість товару та оформлювати замовлення з вказанням контактних даних. Цей функціонал забезпечує зручний та ефективний процес покупок для користувачів і відображає основні можливості інтернет-магазинів.

Під час розробки даної кваліфікаційної роботи були набуті практичні навички в розробці веб-додатків з використанням сучасних технологій та інструментів. Розуміння процесу веб-розробки та взаємодії між фронтендом та бекендом стало фундаментом для подальшого розвитку та вдосконалення проектів у цій сфері.

Зробивши розробку веб-додатку інтернет-магазину для стікерів, я набула важливого досвіду в різних аспектах веб-розробки. Розуміння методології розробки веб-додатків та використання сучасних інструментів та технологій було вирішальним для успішного завершення проекту.

У процесі розробки бекенду використовувався Node.js, що дозволило ефективно обробляти запити від клієнтів та взаємодіяти з базою даних. Це дало мені можливість освоїти серверну розробку за допомогою JavaScript, а також ознайомитися з різноманітними пакетами та модулями, що розширюють можливості розробки.

Також, використання бібліотеки React для фронтенду сприяло створенню динамічного та інтерактивного користувацького інтерфейсу. Ознайомлення з React Router дозволило налаштувати навігацію між сторінками додатку, а використання Axios спростило взаємодію з сервером через HTTP-запити. Bootstrap забезпечив швидку та зручну стилізацію додатку за допомогою готових стилів та компонентів.

Процес розробки веб-додатку інтернет-магазину для стікерів вимагав від мене не лише технічних навичок, але й уміння аналізувати та враховувати потреби та вимоги користувачів. Важливим етапом було проектування та розробка бази даних, яка мала забезпечити ефективне збереження та організацію даних про стікери, категорії та користувачів. Використання Postgre SQL дозволило мені створити структуровану базу даних та виконувати потрібні запити для отримання необхідної інформації.

Розробка веб-додатку інтернет-магазину для стікерів дала мені значний досвід у веб-розробці, ознайомила з ключовими компонентами та інструментами, що використовуються для створення функціональних та інтерактивних веб-додатків.

Крім того, цей проект показав мені важливість розуміння потреб користувачів та створення зручного та ефективного інтерфейсу для них.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How to Build a Web App in 12 Simple Steps. 2023. URL: https://kissflow.com/application-development/how-to-create-a-web-application/ (Дата звернення 15.05.2023)
2. JavaScript language overview. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_overview (Дата звернення 17.05.2023)
3. Griggs B. Node Cookbook: Discover solutions, techniques, and best practices for server-side web development with Node.js 14, 4th Edition. London: Packt Publishing. 2020. 512 с.
4. Stone S. M. Automating with Node.js. 2018. 196 с.
5. Khan S. How to Create a Winning Ecommerce Web App With Node.js. JavaScript in Plain English. 2021. URL: https://javascript.plainenglish.io/make-your-ecommerce-app-10x-faster-by-creating-it-with-node-js-c96cfff79a4 (Дата звернення 15.05.2023)
6. LIM G. Beginning Node.js, Express & MongoDB Development. 2019. 155 с.
7. Dillion M. Building Your First Application With Deno. Stream. 2022. URL: https://getstream.io/blog/build-deno-app/ (Дата звернення 16.05.2023)
8. Truman N. D. The Ultimate Guide to Node.js, MongoDB, and Express: Learn to Build Modern and Scalable Web Applications from Scratch (Web Development Crash Course 4th book). 2023. 185 с.
9. Korolov S. Should You Consider Ruby on Rails for E-Commerce Development? Railsware. 2023. URL: https://railsware.com/blog/ruby-on-rails-ecommerce/ (Дата звернення 16.05.2023)
10. Creating Web Applications with Flask. PyCharm. 2023. URL: https://www.jetbrains.com/help/pycharm/creating-web-application-with-flask.html (Дата звернення 16.05.2023)
11. React. The library for web and native user interfaces. URL: https://react.dev/ (Дата звернення 15.05.2023)

12. How to create an e-commerce website with React, Cloudinary, and Xata. Hackmamba. 2022. URL: https://dev.to/hackmamba/how-to-create-an-e-commerce-website-with-react-cloudinary-and-xata-1875 (Дата звернення 17.05.2023)

13. Wexler J. Get Programming with Node.js. First edition. New York: Manning. 2019. 480 с.

14. Hunter Thomas II. Distributed Systems with Node.js: Building Enterprise-Ready Backend Services. 1st Edition. California: O'Reilly Media. 2020. 377 с.

15. What Does a Back-End Developer Do? Coursera. 2023. URL: https://www.coursera.org/articles/back-end-developer (Дата звернення 21.05.2023)

16. Harbuzava K. Ways to Make an Ecommerce Store on Node.js in 2023 |Guide for Beginners. Flatlogic Platform. 2022. URL: https://flatlogic.com/blog/ways-to-make-an-ecommerce-store-on-node-js-in-2021-guide-for-beginners/ (Дата звернення 15.05.2023)

17. Shiotsu Y. A Beginner's Guide to Back-End Development. 2021. URL: https://www.upwork.com/resources/beginners-guide-back-end-development (Дата звернення 21.05.2023)

18. Owolabi B. How I built an E-commerce API with NodeJs, Express and MongoDB (Part 1). Geek Culture. 2021. URL: https://medium.com/geekculture/how-i-built-an-e-commerce-api-with-nodejs-express-and-mongodb-7b42b5253ffb (Дата звернення 16.05.2023)

19. Ismail Kaya. What Is Frontend as a Service? 2023. URL: https://instantcommerce.io/blog/frontend-as-a-service (Дата звернення 21.05.2023)

20. Front-End Development: The Complete Guide. Cloudinary. URL: https://cloudinary.com/guides/front-end-development/front-end-development-the-complete-guide (Дата звернення 21.05.2023)