МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

КАРАВЕЦЬ Роман Олексійович

Веб-платформа електронної комерції з використанням документо-орієнтованої системи керування базою даних / Web-based e-commerce platform using a document-oriented database management system

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
Р. О. Каравець

_____

Науковий керівник:
І. Р. Кіт

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
КАРАВЕЦЬ Роман Олексійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-платформа електронної комерції з використанням документо-орієнтованої системи керування базою даних / Web-based e-commerce platform using a document-oriented database management system керівник роботи Кіт Іван Романович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. № 753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- провести аналіз предметної області;
- зробити огляд баз даних для електронної комерції;
- дослідження документно-орієнтованих систем керування базою даних;
- розробка архітектури веб-сторінки;
- реалізація веб-платформи;
- тестування та оптимізація.

5. Перелік графічного матеріалу в роботі:
- алгоритм роботи веб-платформи;
- знімки роботи веб-платформи.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент_____ Р.О. Каравець
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І.Р. Кіт
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Клаліфікаційна робота на тему “Веб-платформа електронної комерції з використанням документоорієнтованої системи керування базою даних”
на здобуття освітнього ступення “Бакалавр” зі спеціальності 122 “Комп`ютрні науки” освітньої програми “Комп`ютрні науки”написана обсягом в 50 сторінок і містить 24 ілюстрацій, 2 додатки та 25 використаних джерел.

Мета роботи полягала в розробці веб-інтерфейсу для користувачів, що дозволить їм шукати, переглядати чи придбати товари.

Методи досліджень: Аналіз конкретних випадків, використання платформи в реальних умовах, досліджуючи їхні переваги, недоліки та особливості використання документо-орієнтованих систем керування базами даних.

Результати дослідження: розглянуто наскільки документо-орієнтовані системи керування базою даних є ефективними для зберігання та обробки даних в електронних комерційних платформах. Виявленно переваги цих систем у порівнянні з традиційними реляційними базами даних.

Вони можуть масштабуватися та забезпечувати високу продуктивність при обробці великого обсягу даних, що є важливим для платформ електронної комерції з великою кількістю користувачів та товарів.

Ключові слова: ДОКУМЕНТО-ОРІЄНТОВАНІ БД, ЕЛЕКТРОНА КОМЕРЦІЯ, MERN.

ANNOTATION

This qualification work titled "E-commerce web platform using document-oriented database management system" aimed at obtaining a Bachelor's degree in Computer Science program. It consists of 50 pages, including 24 illustrations, 2 appendices, and references to 25 sources.

The purpose of the work was to develop a web interface for users to search, view, and purchase products.

Research Methods: Analysis of specific cases, utilization of the platform in real-world conditions, examining their advantages, disadvantages, and peculiarities of using document-oriented database management systems.

Research Results: The study investigated the effectiveness of document-oriented database management systems for storing and processing data in electronic commerce platforms.

The advantages of these systems over traditional relational databases were identified. They can scale and provide high performance in processing large volumes of data, which is crucial for e-commerce platforms with numerous users and products.

Keywords: DOCUMENT-ORIENTED DATABASES, E-COMMERCE, MERN.

ЗМІСТ

ВСТУП

Актуальність теми: У сучасному цифровому ландшафті електронна комерція стала не лише необхідністю, а й стратегічним фактором успіху для багатьох бізнесів. Швидкість, зручність та доступність покупок в Інтернеті приваблюють мільйони споживачів щодня. Однак успішна електронна торгівля вимагає надійної та ефективної системи управління базою даних, яка забезпечить не лише зберігання великого обсягу інформації, але й швидкий доступ до неї, а також гнучкість в обробці структурованих та неструктурованих даних.

У цьому контексті використання документо-орієнтованої системи керування базою даних (наприклад, MongoDB) стає важливим аспектом розробки веб-платформ електронної комерції. Документо-орієнтовані бази даних забезпечують гнучкість у роботі з даними, дозволяючи зберігати складні структури інформації у вигляді документів, що робить їх ідеальними для сучасних електронних магазинів з великою кількістю товарів та різноманітними параметрами.

У даній роботі ми дослідимо переваги використання документо-орієнтованих баз даних у сфері електронної комерції, а також розглянемо ключові аспекти розробки веб-платформ, які використовують ці технології. Вивчення цієї теми дозволить нам краще зрозуміти, як створити ефективну та сучасну систему електронної торгівлі, яка відповідає вимогам сьогодення.

Мета даної дипломної роботи полягає у розробці веб-платформи для продажу товарів, яка відповідатиме сучасним вимогам до інтернет-магазинів та забезпечуватиме зручний та ефективний процес покупок для користувачів.

Для досягнення цієї мети необхідно вирішити такі задачі:

- провести аналіз предметної області;
- зробити огляд баз даних для електронної комерції;
- дослідження документно-орієнтованих систем керування базою даних;
- розробка архітектури веб-сторінки;
- реалізація веб-платформи;
- тестування та оптимізація.

Зазначені завдання дозволять успішно реалізувати поставлену мету і створити ефективну та сучасну веб-платформу для продажу смартфонів, яка відповідатиме вимогам електронної комерції.

Об'єкт дослідження є веб-платформа електронної комерції.

Предмет дослідження розробки є веб-сторінки для продажу смартфонів з використанням сучасних технологій веб-розробки та документно-орієнтованої системи керування базами даних.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися: «РОЗРОБКА ВЕБ-ПЛАТФОРМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ ДОКУМЕНТО-ОРІЄНТОВАНОЇ СИСТЕМИ КЕРУВАННЯ БАЗОЮ ДАНИХ» VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

1 АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕНЬ

1.1 Аналіз предметної області

Електронна комерція, відома також як e-commerce, є важливою складовою сучасного бізнесу, особливо в галузі продажу техніки. Цей розділ розглядає ключові аспекти електронної комерції на прикладі інтернет-магазину техніки та пропонує висновки щодо її впливу та перспектив розвитку. Технологічні інновації та зручність для покупців:

Однією з основних переваг електронної комерції для магазинів техніки є можливість використання передових технологій для приваблення покупців. Інтерактивність веб-сайтів, персоналізовані рекомендації, використання віртуальної реальності або розширеної реальності для віртуальної примірки товарів - це лише кілька прикладів того, як технології сприяють зручності та привабливості покупок в інтернеті.

Глобальний доступ та ринкові можливості. Електронна комерція дозволяє магазинам техніки легко отримувати доступ до глобальних ринків. За допомогою інтернету, магазин може привернути покупців з будь-якої точки світу, розширити свою аудиторію та збільшити обсяги продажів. Це відкриває нові можливості для росту бізнесу та міжнародного співробітництва.

Аналіз споживчої поведінки та персоналізація. За допомогою аналітики веб-сайтів та даних про покупців, магазини техніки можуть ретельно аналізувати поведінку клієнтів та надавати персоналізовані пропозиції. Це дозволяє підлаштовувати асортимент товарів та пропозиції до конкретних потреб і вподобань кожного клієнта, що підвищує шанси на успішну покупку.

Забезпечення безпеки та довіри. Однією з найважливіших аспектів електронної комерції є забезпечення безпеки та довіри покупців. Магазини техніки повинні вкладати значні зусилля в захист особистих даних клієнтів, забезпечення безпечних транзакцій та запобігання шахрайству та кібератакам.

Електронна комерція відіграє важливу роль у продажу техніки, надаючи магазинам широкі можливості для приваблення клієнтів, розширення ринків та

покращення обслуговування покупців. Однак успішна електронна комерція вимагає постійного вдосконалення технологій, аналізу ринку та уваги до потреб клієнтів.

1.2 Огляд баз даних для електронної комерції

В розробці веб-сторінки для продажу мобільної техніки, вибір бази даних грає важливу роль у забезпеченні ефективного збереження, організації та доступу до даних. Є багато видів бази даних, таких як

Реляційні бази даних (RDBMS) є одними з найбільш поширених та надійних систем управління базами даних, які використовуються для збереження структурованих даних в таблицях, які пов'язані між собою за допомогою відносин (або "реляцій"). Вони дозволяють ефективно організувати та зберігати дані, а також забезпечують широкі можливості для операцій з цими даними. Основні характеристики реляційних баз даних, які роблять їх привабливими для використання в проектах, включають структурованість даних, реляційні бази даних вимагають, щоб дані були структуровані у вигляді таблиць з рядками і стовпцями. Це дозволяє використовувати чіткі структури даних, що полегшує їх зберігання та обробку.

Відносини між таблицями дозволяють зв'язувати дані між собою і використовувати ці зв'язки для виконання складних запитів та операцій з даними. Реляційні бази даних забезпечують підтримку транзакцій, що дозволяє виконувати групу операцій як єдину одиницю роботи. Це гарантує цілісність та надійність даних.

Більшість реляційних баз даних використовують мову SQL для взаємодії з даними. SQL надає потужні засоби для створення, зміни, видалення та запитування даних в базі даних.

У проекті, де потрібно зберігати структуровані дані про користувачів, продукти, замовлення та інші аспекти, реляційна база даних може бути використана для створення окремих таблиць для кожної сутності (наприклад, таблиці користувачів, продуктів, замовлень тощо) та встановлення відносин між

ними. Наприклад, дані про користувачів можуть бути зв'язані з їх замовленнями за допомогою унікальних ідентифікаторів. MySQL та PostgreSQL є двома найпоширенішими та надійними реляційними базами даних. Обидві системи мають велику спільноту користувачів, широкий набір функцій та хорошу продуктивність. Вибір між ними може залежати від конкретних потреб та вимог проекту, таких як масштабованість, продуктивність, функціональність та інші фактори. Переваги і недоліки SQL та NoSQL.

Таблиця 1.1. Порівняння з SQL та NoSQL:

Особливість	SQL	NoSQL (MongoDB)
Схема даних	Строго визначена схема	Гнучка схема даних
Тип даних	Табличний формат	JSON-подібні документи
Горизонтальне масштабування	Складне	Легке
Продуктивність	Звичайно менш ефективна	Швидка
Використання	Для структурованих даних	Для структурованих даних

NoSQL бази даних стали популярними в останні роки завдяки їхній здатності працювати з великим обсягом даних та забезпечувати гнучкість у моделюванні даних. У порівнянні з реляційними базами даних, які вимагають строго визначеної структури даних у вигляді таблиць з рядками і стовпцями, NoSQL бази даних дозволяють зберігати дані у нереляційній формі, такі як документи, ключ-значення, колонки або графи. Основні характеристики NoSQL баз даних, які роблять їх привабливими для використання в проектах, включають гнучкість у моделюванні даних: NoSQL бази даних не обмежені строгими схемами, що дозволяє зберігати дані з різною структурою та типами. Це особливо корисно в ситуаціях, коли структура даних може змінюватися з часом або коли потрібно працювати з даними різної природи.

Підтримка великого обсягу даних: NoSQL бази даних розроблені з урахуванням роботи з великими обсягами даних та високими навантаженнями. Вони забезпечують ефективне зберігання та опрацювання даних при великому обсязі та високій швидкості запитів.

Багато NoSQL баз даних підтримують горизонтальне масштабування, що дозволяє розширювати систему шляхом додавання нових вузлів до кластера. Це дозволяє підвищити продуктивність та забезпечити високу доступність даних.

NoSQL бази даних підтримують різні моделі даних, такі як документи, ключ-значення, колонки та графи. Це дозволяє вибрати найбільш підходящий тип бази даних для конкретного типу даних та завдань.

MongoDB та CouchDB є двома популярними NoSQL базами даних, які широко використовуються у проектах зі збереженням нереляційних даних. MongoDB використовує модель документа, де дані зберігаються у вигляді JSON-подібних документів, що робить його особливо підходящим для роботи з динамічними або змінюючимися даними. CouchDB, з іншого боку, використовує модель ключ-значення та надає можливості для реплікації та синхронізації даних між різними серверами.

In-memory бази даних є цінним інструментом для додатків, де потрібно швидко отримувати доступ до даних. Вони зберігають дані безпосередньо в оперативній пам'яті комп'ютера, що дозволяє миттєво звертатися до них без затримок, пов'язаних з читанням з диска. Redis - це дуже швидка in-memory база даних, яка підтримує різні структури даних, такі як строки, списки, набори, хеші та сортовані множини. Вона широко використовується для кешування запитів до бази даних, сесій користувачів та збереження тимчасових даних. Redis також має можливості реплікації та підтримує транзакції, що робить його потужним інструментом для розробки швидких та масштабованих додатків. Memcached - це ще одна популярна in-memory система зберігання даних, яка також використовується для кешування та прискорення доступу до даних. Вона працює з ключ-значеннями та забезпечує дуже високу швидкість доступу до даних завдяки зберіганню їх в оперативній пам'яті. Memcached добре підходить для розподіленого

кешування даних та масштабованих додатків, що мають велику кількість запитів до бази даних.

Гібридні рішення використовують комбінацію різних типів баз даних для вирішення різних потреб. Наприклад, реляційну базу даних можна використовувати для збереження основних даних про продукти та користувачів, які мають структурований формат. У той же час, NoSQL базу даних, таку як MongoDB, можна використовувати для збереження великого обсягу нереляційних даних, таких як історія перегляду користувачів або дані з соціальних мереж. Такий підхід дозволяє оптимізувати продуктивність та ефективність додатку, використовуючи найбільш підходящі для кожної конкретної ситуації інструменти зберігання даних.

Саме я вибрав для зберігання даних MongoDB, тому що там гнучка модель даних. MongoDB використовує гнучку нереляційну модель даних, яка дозволяє зберігати дані у форматі JSON-подібних документів. Це дозволяє легко зберігати дані різної структури, що дозволяє ефективно моделювати дані про продукти, користувачів, замовлення та інші аспекти, які можуть мати змінну структуру.

Швидкодія та масштабованість:MongoDB відомий своєю високою швидкодією та можливістю масштабування. Вона використовує денормалізацію та індекси для забезпечення швидкого доступу до даних, що особливо важливо для веб-сторінок з великою кількістю запитів. Крім того, MongoDB дозволяє легко масштабувати базу даних горизонтально за допомогою реплікації та розділення даних на шари.

Висока доступність та надійність:MongoDB забезпечує високу доступність та надійність за рахунок можливості налаштовувати реплікацію даних, автоматичне відновлення та широкий набір інструментів для моніторингу та управління базою даних. Це забезпечує відмінну продуктивність та стабільність роботи системи, навіть при великому навантаженні та випадках відмов.

Спільнота та екосистема:MongoDB має велику та активну спільноту користувачів та розробників, яка постійно розвивається та підтримується. Це забезпечує доступність різноманітних ресурсів, документації, плагінів та інструментів для розробки та управління базою даних.

1.3 Документо-орієнтовані системи керування базою даних

Цей розділ присвячений дослідженню документо-орієнтованих систем керування базами даних (Document-Oriented Database Management Systems, Document-Oriented DBMS) та їхньому значенню в сучасному програмному забезпеченні. У світі, що стрімко розвивається, де постійно зростає обсяг даних та змінюються вимоги, важливо зрозуміти переваги і можливості таких баз даних.

Документо-орієнтовані бази даних - це сучасний підхід до організації та зберігання даних, де інформація представлена у вигляді документів, що забезпечує гнучкість та простоту в роботі з даними. Наприклад, MongoDB, одна з найпопулярніших документо-орієнтованих баз даних, дозволяє зберігати дані у вигляді JSON-подібних документів, що спрощує їхнє моделювання та редагування.

Документо-орієнтовані бази даних відрізняються від традиційних реляційних баз даних тим, що вони не вимагають фіксованої схеми для даних. Це означає, що кожен документ може мати власну унікальну структуру, і ви можете легко додавати або змінювати поля в цих документах без потреби модифікації всієї бази даних.

Однією з переваг такого підходу є здатність працювати з дуже великими обсягами даних та їхньою розношерстністю. Наприклад, якщо у вас є база даних користувачів у форматі документів, ви можете мати користувачів з різними наборами полів (наприклад, деякі користувачі можуть мати ім'я та прізвище, а інші - тільки ім'я).

Крім того, документо-орієнтовані бази даних зазвичай мають дуже швидкий доступ до даних, оскільки вони зберігаються у вигляді документів, що може бути оптимальним для деяких операцій, таких як пошук або оновлення.

Крім MongoDB, існують інші популярні документо-орієнтовані бази даних, такі як Couchbase, CouchDB, Amazon DynamoDB тощо. Кожна з цих систем має свої унікальні особливості та призначення, але всі вони дозволяють працювати з даними у вигляд

Одна з ключових переваг документо-орієнтованих баз даних є їхня гнучкість у моделюванні даних. Вони не потребують строго визначеної схеми, тому можуть легко адаптуватися до змін у вимогах проекту. Крім того, такі бази даних

забезпечують високу продуктивність та масштабованість, що особливо важливо у сучасних великих проектах.

Документо-орієнтовані бази даних широко використовуються у різних галузях. Наприклад, веб-розробники використовують MongoDB для збереження даних користувачів та контенту веб-сайтів, а мобільні додатки можуть використовувати Firebase Firestore для збереження даних користувачів та синхронізації їхнього стану між різними пристроям

Документо-орієнтовані бази даних є важливим інструментом у сучасному програмуванні, який дозволяє розробникам ефективно працювати з даними та швидко реагувати на зміни в вимогах проектів. Розуміння їхніх переваг та можливостей є ключем до успішного використання в розробці програмного забезпечення.

1.4. Постановка задачі

У сучасному світі електронна комерція займає важливе місце в економіці, стаючи ключовим інструментом для підприємств та споживачів. Інтернет-магазини забезпечують можливість зручного та швидкого здійснення покупок, пропонуючи широкий асортимент товарів та послуг. Проте, створення ефективної та надійної веб-платформи для електронної комерції вимагає використання сучасних технологій веб-розробки та баз даних, що забезпечують високий рівень продуктивності та безпеки.

Метою даної дипломної роботи є розробка веб-сторінки для продажу смартфонів, яка відповідатиме сучасним вимогам до інтернет-магазинів та забезпечуватиме зручний та ефективний процес покупок для користувачів. Для досягнення цієї мети буде використано передові технології веб-розробки та документом орієнтовані системи керування базою даних.

Для реалізації поставленої мети необхідно вирішити такі задачі:

– Провести аналіз предметної області.

Вивчення сучасних тенденцій та вимог до веб-платформ електронної комерції, а також визначення ключових функціональних і нефункціональних вимог до інтернет-магазинів.

– Зробити огляд баз даних для електронної комерції.

Проведення огляду різних типів баз даних, які використовуються для веб-платформ електронної комерції, зокрема реляційних та нереляційних баз даних, та визначення їхніх переваг і недоліків.

– Дослідження документно-орієнтованих систем керування базою даних.

Вивчення можливостей та особливостей документно-орієнтованих систем керування базами даних (наприклад, MongoDB), аналіз їхньої відповідності потребам електронної комерції.

– Розробка архітектури веб-сторінки.

Проєктування структури та архітектури веб-сторінки, яка дозволить реалізувати всі необхідні функції для забезпечення ефективного процесу покупок.

– Реалізація веб-платформи

Розробка веб-платформи для продажу смартфонів з використанням сучасних технологій веб-розробки (HTML, CSS, JavaScript, Node.js) та документно-орієнтованої системи керування базами даних.

– Тестування та оптимізація.

Проведення тестування веб-платформи для виявлення та усунення можливих помилок, оптимізація продуктивності та забезпечення високого рівня безпеки.

При реалізації веб-платформи мається виконувати такі функції, а саме. Веб-сторінка повинна мати функції реєстрації та авторизації користувачів. Реєстрація включає форму для введення особистих даних, таких як ім'я, електронна пошта, пароль, а також підтвердження електронної пошти. Авторизація дозволяє користувачам входити на сайт за допомогою електронної пошти та пароля або через соціальні мережі.

Для зручності користувачів, веб-сторінка повинна мати поле для введення запиту, що дозволить швидко знаходити потрібні товари. Фільтрація товарів за брендом, ціною, характеристиками (наприклад, об'єм пам'яті, розмір екрану,

камера тощо) допоможе користувачам швидко знайти товари, що відповідають їхнім потребам.

Каталог продуктів має відображати список доступних смартфонів з основними характеристиками. Зручне сортування за різними параметрами, наприклад, за ціною або популярністю, допоможе користувачам знайти потрібний товар.

Кожен товар повинен мати сторінку з повним описом, технічними характеристиками, відгуками користувачів та фотографіями з можливістю перегляду.

Функція порівняння товарів дозволить користувачам вибрати кілька смартфонів та порівняти їхні характеристики і ціни.

Користувачі повинні мати можливість додавати товари до кошика, переглядати та редагувати замовлення, а також бачити розрахунок загальної суми замовлення.

Оформлення замовлення включає введення адреси доставки, вибір способу оплати та підтвердження замовлення.

Особистий кабінет користувача дозволяє переглядати історію замовлень, зберігати адреси для доставки та змінювати особисті дані.

Служба підтримки повинна включати контактну інформацію для зв'язку з клієнтською підтримкою та форму зворотного зв'язку для запитів та скарг.

Виконання вищезазначених завдань дозволить створити веб-сторінку для продажу смартфонів, яка буде відповідати сучасним вимогам до інтернет-магазинів, забезпечуючи зручний, безпечний та ефективний процес покупок для користувачів. Ця робота сприятиме вдосконаленню електронної комерції шляхом впровадження інноваційних підходів та технологій у розробку веб-платформ.

2. АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-ПЛАТФОРМИ

2.1 Загальна структура проектованої платформи

MERN — це набір технологій з відкритим вихідним кодом, що часто використовується для створення одно сторінкових веб-додатків, відомих як SPA (Single Page Applications) . Стек включає в себе MongoDB, Express, React та Node.js, що дозволяє створювати високоефективну наскрізну структуру для розробки веб-сайтів швидко та зручно.

Особливості MERN:

- Використання бібліотек з відкритим вихідним кодом.
- Односпрямований потік даних.
- Підтримка мобільних додатків.
- Швидкий рендеринг компонентів.
- Доступна та зрозуміла документація.

Для розробки даного додатку було обрано MERN стек, тобто MongoDB, Express, React і Node.js. Головна перевага такого підходу полягає в тому, що вся кодова база написана на JavaScript або, в деяких випадках, на TypeScript. Використання однієї мови програмування в рамках всього проекту усуває потребу в перемиканні між різними мовами, такими як JavaScript і Python, що значно спрощує процес розробки та усуває необхідність у пошуку залежностей для інтеграції різних мов.

Завдяки MERN стеку можна легко створити трирівневу архітектуру, яка включає фронтенд, бекенд та базу даних. Архітектура веб-платформи зображена на рисунку 2.1 нижче.

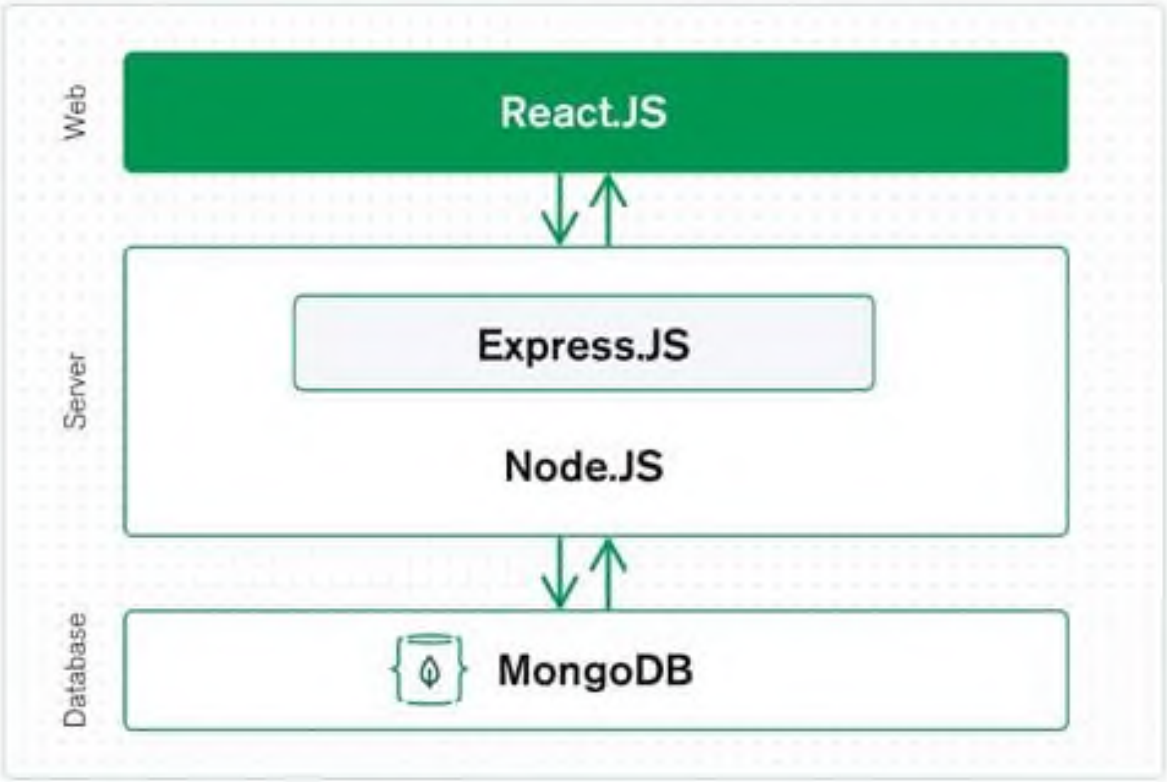


Рисунок 2.1 – Трирівнева архітектура веб-платформи

Варто зазначити, що ще однією суттєвою перевагою використання MERN є гнучкість у розробці архітектури. Незалежно від обраного підходу, ви можете спочатку створити серверну частину, підключити її до бази даних, а потім інтегрувати з клієнтською частиною. Інший варіант — розробити клієнтську частину з генерацією псевдо даних, які відповідатимуть структурі даних, що надходитимуть від сервера, а потім перейти до реалізації серверної частини та її підключення до бази даних. Це дозволяє налаштувати процес розробки найбільш зручним для команди способом та підвищує ефективність роботи.

MongoDB - це кросплатформенна база даних NoSQL з відкритим вихідним кодом, яка в основному використовується для масштабованих додатків з великими обсягами даних і завдань, які погано функціонують в реляційних базах даних. Вона використовує формат зберігання документів, відомий як BSON. Дизайн MongoDB базується на колекціях і документах, як показано на рисунку 2.2 нижче, які замінюють використання таблиць і рядків у звичайних реляційних базах даних.

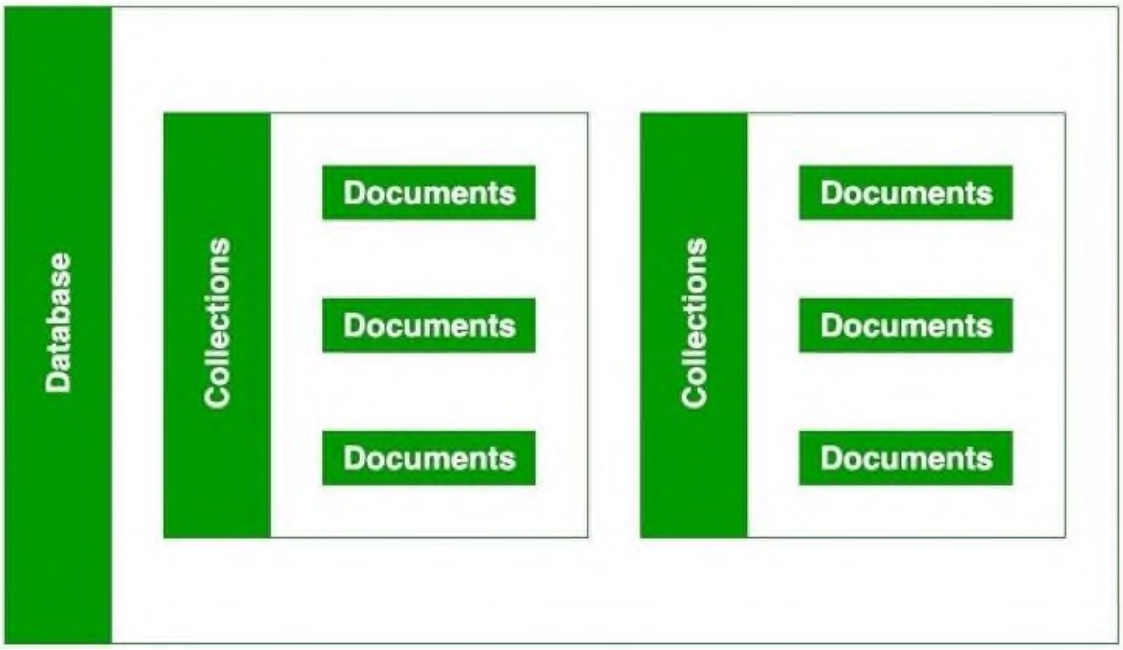


Рисунок 2.2. Дизайн архітектури MongoDB

MongoDB також підтримує різноманітні операції з документами, включаючи додавання, запити, оновлення та видалення. MongoDB підходить для різних сценаріїв використання завдяки різноманітності значень полів і потужним мовам запитів. Крім того, її здатність масштабуватися для розміщення більших обсягів даних по горизонталі сприяла її зростаючому успіху як найпопулярнішої у світі NoSQL-бази даних.

Express, що представляє літеру «E» у стеку MERN, - це легкий та універсальний фреймворк для веб-додатків, побудований на основі NodeJS. Завдяки великій спільноті підтримки, він включає в себе багатий набір функціональних можливостей для розробки веб- та мобільних додатків. Навіть незважаючи на велику кількість пакетів підтримки разом з функціоналом для кращого створення програмного забезпечення, Express не впливає на продуктивність NodeJS.

React, що представляє літеру «R» у стеку MERN, фокусується на створенні шару перегляду (View Layer), який відповідає за всі видимі частини сторінки додатку. React - це багатоцільова бібліотека JavaScript з відкритим вихідним кодом, яка використовується для створення користувацьких інтерфейсів на основі компонентів інтерфейсу користувача

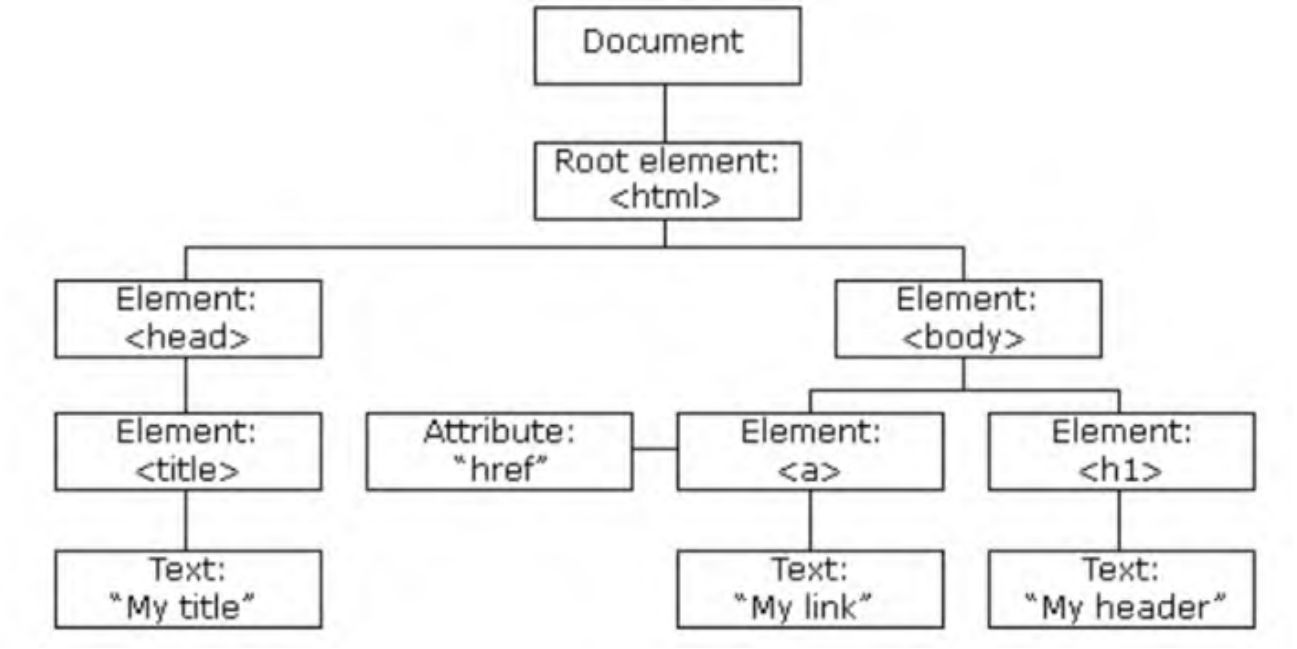


Рисунок 2.3. Приклад деревоподібної структури об'єктної моделі документа

Віртуальний DOM (або VDOM) - це абстрактне представлення DOM (Document Object Model), і його рішення будуються на основі звичайного DOM. DOM представляє інтерфейс користувача програми, модель якого зображує документ як сукупність різних вузлів і об'єктів, що взаємодіють зі структурою, макетом і вмістом веб-сайту за допомогою мов програмування (MDN Web Docs. 2022b). На рисунку 2.3 вижче показано приклад структури DOM

NodeJS - це кросплатформенне середовище виконання JavaScript з відкритим вихідним кодом, призначене для створення масштабованих додатків. NodeJS створено на основі движка виконання V8 для Google Chrome, який добре відомий своєю ефективною роботою поза браузером.

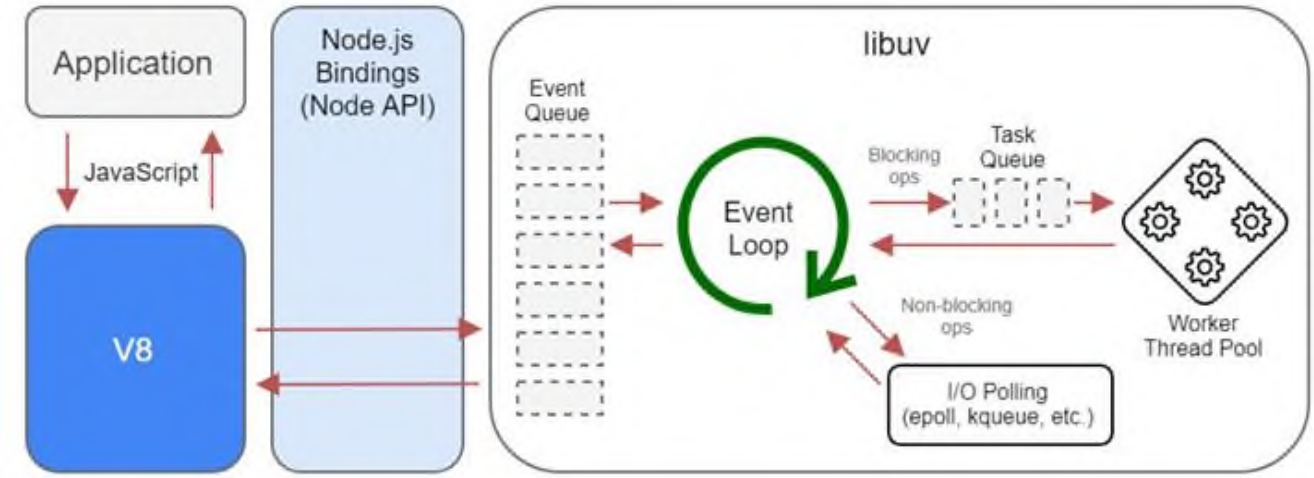


Рисунок 2.4 Цикл обробки подій NodeJS

Завдяки використанню керованого подіями дизайну та однопотокового циклу обробки подій, NodeJS забезпечує асинхронну та неблокуючу оптимізацію вводу/виводу для підвищення продуктивності та масштабованості веб-додатків, як показано на рисунку 2.4 нижче. Таким чином, це альтернативний підхід для розробників, який дозволяє чекати і виконувати запити на розробку легких додатків, що працюють в реальному часі.

2.2 Алгоритмічне забезпечення

Проект базувався на веб-додатку для електронної комерції B2C, який містить два типи користувачів: адміністратор і користувач. Адміністратори відповідають за різноманітні управлінські завдання, включаючи створення, зміну та видалення продуктів з баз даних. З іншого боку, користувач може здійснювати навігацію та вивчати інформацію про продукт, а також купувати продукт, додаючи його до кошика та завершуючи транзакцію для цього продукту. Деякі певні сторінки та веб-маршрути є видимими для громадськості, в той час як інші обмежені для адміністраторів та зареєстрованих користувачів.

Поведінка користувачів показана в Таблиці 2.1 нижче, яка ілюструє деякі з основних особливостей веб-застосунків для електронної комерції.

Таблиця 2.1. Різні моделі поведінки користувачів.

Клієнти	Адміністратори
Створити новий обліковий запис	Недійсний
Сортуйте список товарів за ціною, датою додавання, кількістю проданих товарів та категорією	Сортувати список товарів за ціною, датою додавання, кількістю проданих товарів та категорією
Переглянути список товарів	Переглянути список товарів
Переглядати інформацію про товар із зображеннями	Переглядати інформацію про товар із зображеннями
Додавання товарів у кошик	Не вірно
Змінювати кількість товарів у кошику	Не вірно
Оплатити додані товари в кошику	Не вірно
Не вірно	Додати нові товари в базу даних
Не вірно	Видалити товари з бази даних
Не вірно	Оновити товари в базі даних
Не вірно	Додати категорію в базу даних
Переглянути власні замовлення	Переглянути всі замовлення користувача

Після визначення поведінки різних користувачів додатку необхідно визначити їхні залежності, щоб закрити прогалини в початковій розбивці результатів. Таким чином, блок-схема залежностей додатку електронної комерції, побудована на основі таблиці 2.1, показана на рисунку 2.5 нижче

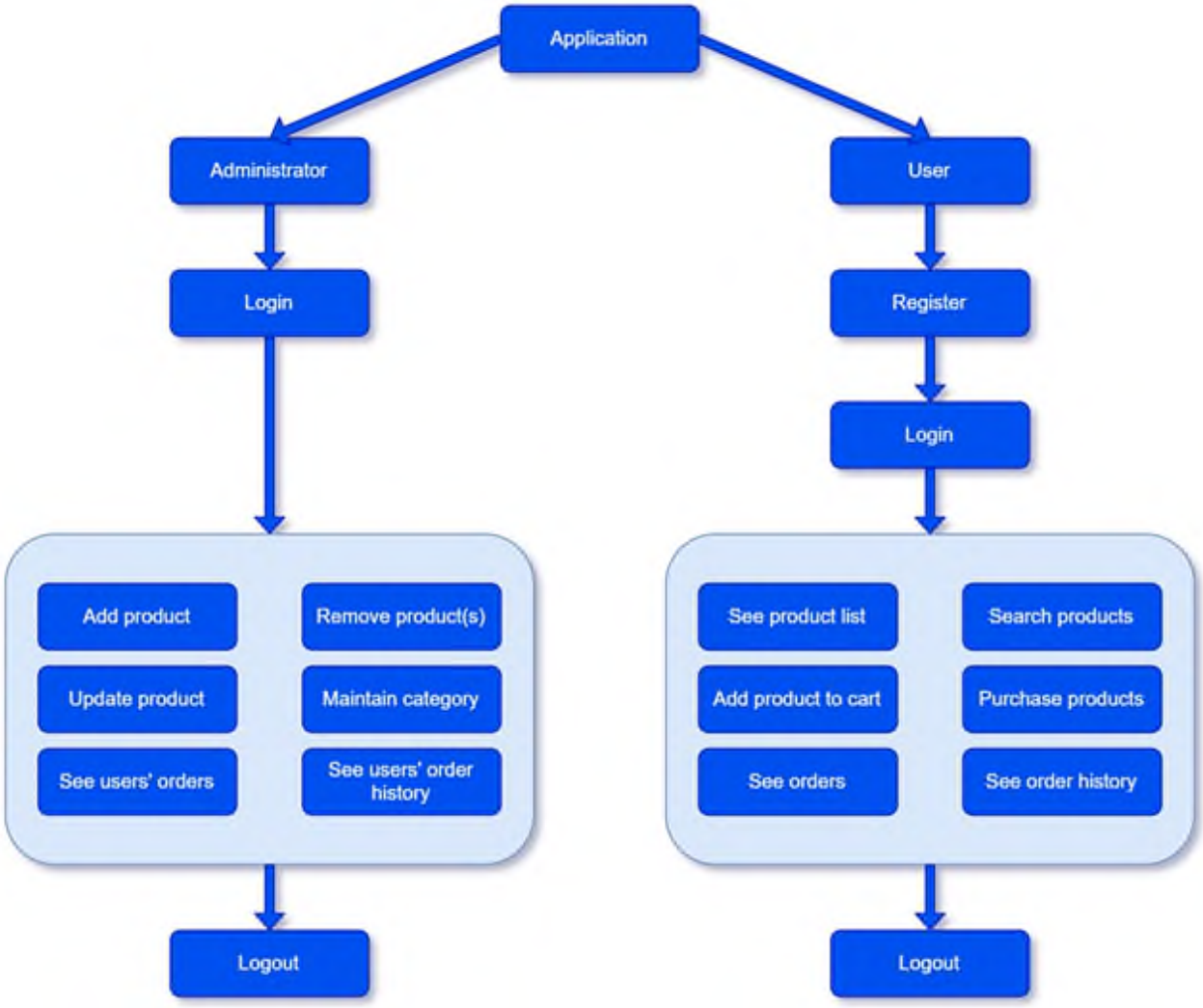


Рисунок 2.5 Блок-схема залежностей.

Веб-платформа електронної комерції, розроблена на основі стеку технологій MERN (MongoDB, Express.js, React.js, Node.js), пропонує інтуїтивно зрозумілий та зручний інтерфейс для користувачів з метою здійснення покупок в Інтернеті. Процес роботи платформи Додаток А можна розглядати наступним чином:

Крок 1. Користувач ініціює взаємодію з платформою, переглядаючи доступний асортимент товарів. Для цього веб-платформа надає можливість швидкого та зручного пошуку товарів за різними критеріями.

Крок 2. Після знаходження цікавого товару користувач додає його до кошика за допомогою відповідної кнопки або опції на сторінці товару.

Крок 3. Система отримує дані про вибраний товар та додає їх до кошика користувача, оновлюючи відповідні дані.

Крок 4. Користувач має можливість переглянути вміст свого кошика, де відображаються всі додані ним товари.

Крок 5. Після перегляду кошика користувач переходить до оформлення замовлення, де він може обрати зручний спосіб доставки та сплати за товари.

Крок 6. Система перевіряє наявність товарів на складі. У разі, якщо вони є в наявності, оновлює дані про наявність. У протилежному випадку, користувач отримує повідомлення про те, що деякі товари відсутні.

Крок 7. Якщо перевірка наявності товарів пройшла успішно, система переходить до обробки оплати. Вона виконує обробку оплати згідно обраного користувачем способу оплати.

Крок 8. Після успішної оплати система генерує замовлення на основі даних з кошика користувача.

Крок 9. Згенеровані дані замовлення зберігаються в базі даних для подальшої обробки та аналізу.

Крок 10. Надсилається підтвердження електронною поштою, яке містить інформацію про замовлення, деталі оплати та інші важливі дані. Після відправлення підтвердження електронною поштою процес роботи веб-платформи завершується.

2.3 Інформаційне забезпечення платформи

Mongoose - це бібліотека об'єктно-документного відображення (ODM), яка використовується для полегшення розробки Node і MongoDB. Вона відповідає за управління зв'язками даних, виконує перевірку схем та слугує посередником між об'єктами в коді та представленнями об'єктів в MongoDB. Крім того, Mongoose пропонує безліч методів і функцій, які ефективно полегшують взаємодію між NodeJS і MongoDB. Рисунок 2.6 нижче ілюструє взаємозв'язок між Mongoose, NodeJS та MongoDB.

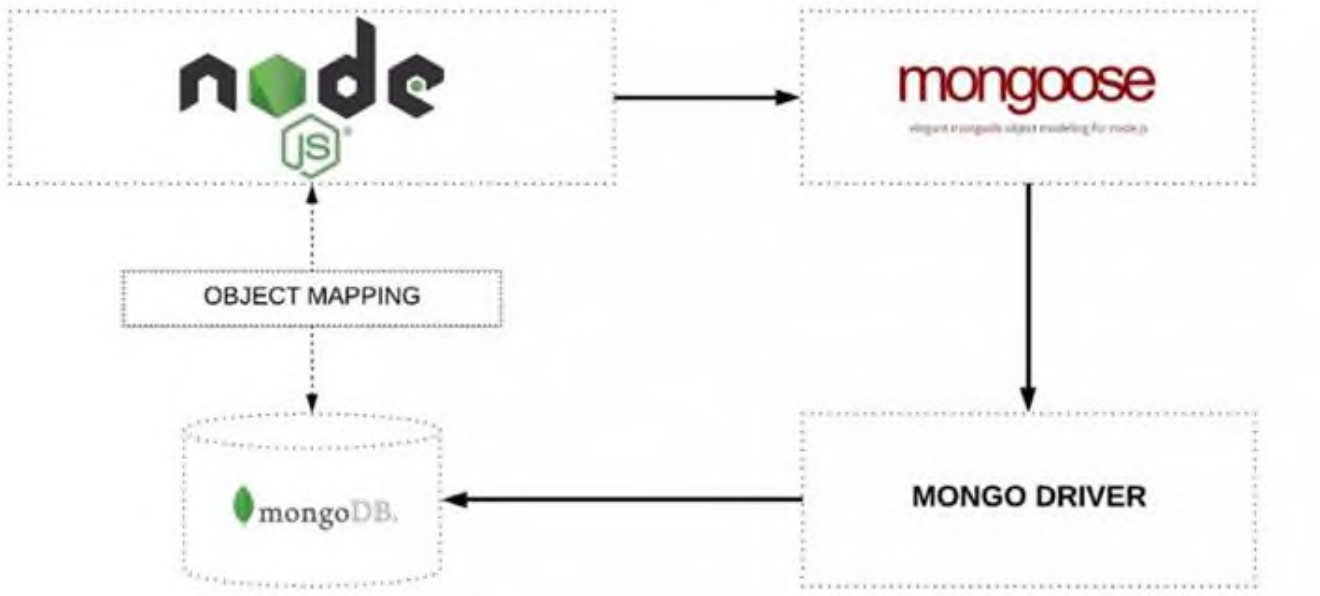


Рисунок 2.6. Взаємозв'язок між Mongoose, NodeJS та MongoDB

Як показано на Рисунку 2.6, Mongoose використовується для створення взаємодії між Node та MongoDB через мапування об'єктів. Після цього Mongoose формує з'єднання з MongoDB за допомогою Mongo Driver. Таким чином, взаємозв'язок між Mongoose, NodeJS та MongoDB забезпечує можливість роботи з даними. Першим кроком у роботі з Mongoose, як і з іншими ODM бібліотеками, є створення схеми. Як описано на сторінці документації Mongoose, схема визначає структуру даних і приведення властивостей, а також наступні методи: методи екземплярів, складені індекси, статичні методи моделі та проміжне програмне забезпечення (Mongoose, n.d. (b)). Після завершення першого етапу розроблені схеми будуть використані для зіставлення з колекціями MongoDB та формування документів даних, включених до кожної колекції. Другий етап, необхідний програмістам, полягає у створенні моделі Mongoose. Моделі складаються з конструкторів схем, основним завданням яких є створення та сканування документів у базі даних Mongo. Запити, видалення та оновлення документів у базі даних є додатковими можливостями моделей, про які варто згадати.

Після встановлення MongoDB можна підключитися до бази даних з вашого додатку. Mongoose надає зручний інтерфейс для взаємодії з MongoDB.

```
const mongoose = require('mongoose');
const mongodbUrl = 'mongodb://localhost:27017/mydatabase';
const options = {
  useNewUrlParser: true,
  useUnifiedTopology: true};
mongoose.connect(mongodbUrl, options)
  .then(() => {
    console.log('Connected to MongoDB');
  .catch((error) => {
    console.error('Error connecting to MongoDB:', error);
```

Рисунок 2.7 Код взаємодії

У MongoDB дані зберігаються у вигляді колекцій, які можна уявити як еквівалент таблиць у реляційних базах даних. Для створення нової колекції використовується метод createCollection():

```
javascript
Copy code
const { Schema } = mongoose;

const UserSchema = new Schema({
    name: String,
    email: String,
    age: Number
});

const User = mongoose.model('User', UserSchema);

// Створення нової колекції користувачів
User.createCollection();
```

Рисунок 2.8. Створення нової колекції

Після створення колекції можна виконувати різні операції з даними, такі як додавання нових записів, оновлення, видалення тощо. MongoDB використовує мову запитів, яка нагадує JavaScript та надає зручний спосіб взаємодії з даними. Ось деякі приклади запитів на рисунку 2.9:

```javascript
// Знайти всіх користувачів з іменем "John"
User.find({ name: 'John' }, (err, users) => {
    if (err) throw err;
    console.log(users);
});

// Оновити вік користувача з іменем "John" на 30
User.updateOne({ name: 'John' }, { $set: { age: 30 }}, (err, result) => {
    if (err) throw err;
    console.log(result);
});

// Видалити всіх користувачів з віком менше 25
User.deleteMany({ age: { $lt: 25 }}, (err, result) => {
    if (err) throw err;
    console.log(result);
});
```

Рисунок 2.9. Приклади запитів

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація програмного забезпечення

Перше, що потрібно зробити на серверній стороні, це створити файл package.json для підтримки необхідних залежностей і devDependencies, а також для виконання різних завдань за допомогою запуску команд. Перед створенням команди на рисунку 3.1 нижче було встановлено NodeJS.

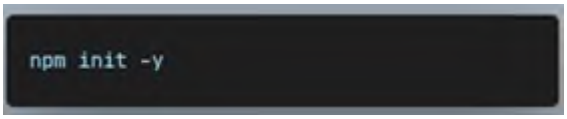


Рисунок 3.1. Команда NPM для ініціалізації сервера.

Запустивши команду на рисунку 3.1 вище з каталогу сервера, package.json буде автоматично згенеровано з налаштуваннями за замовчуванням без проходження інтерактивного процесу. Файли index.js та app.js додаються до кореневого каталогу серверних додатків, включаючи встановлення пакунків «dotenv» та «express» як залежностей. Файли logger.js та config.js також містяться в папці utils. Конфігурація змінних оточення зберігається у файлі utils/config.js, як показано на рисунку 3.2. нижче:

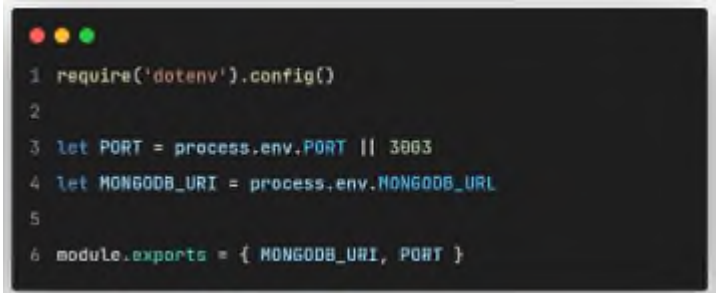


Рисунок 3.2. Вміст файлу config.js.

У рядку 1 імпортується пакет 'dotenv' для завантаження змінних оточення, що зберігаються у файлі .env, а потім для того, щоб не втручатися в історію git'a,

він включається у файл «.gitignore». Таким чином, змінні оточення можуть бути відокремлені від кодової бази і виключені під час запуску виробничого додатку.

Файл app.js створюється як власне додаток, який включає в себе всі завдання, а також деякі пакети вузлів, щоб змусити сервер працювати належним чином. На рисунку 3.3 нижче показано вміст початкового файлу app.js, створеного на початку конфігурації бекенда.

```
1  const config = require('./utils/config')
2  const express = require('express')
3  const logger = require('./utils/logger')
4  const mongoose = require('mongoose')
5  const cors = require('cors')
6  const cookieParser = require('cookie-parser')
7
8  const app = express()
9  app.use(express.json())
10 app.use(cookieParser())
11 app.use(cors())
12
13 // db connection
14 mongoose
15   .connect(config.MONGODB_URI, {
16     useCreateIndex: true,
17     useFindAndModify: false,
18     useNewUrlParser: true,
19     useUnifiedTopology: true,
20   })
21   .then(() => {
22     logger.info('Connected to MongoDB')
23   })
24   .catch((error) => {
25     logger.error('error connection to MongoDB:', error.message)
26   })
27
28 app.get('/', (req, res) => {
29   return res.json({ message: 'Running...' })
30 })
31
32 module.exports = app
```

Рисунок 3.3 Вміст файлу app.js.

Як показано на Рисунку 3.3, деякі пакети вузлів та проміжного програмного забезпечення, що входять до складу app.js, можна продемонструвати нижче: - mongoose стимулює з'єднання між базою даних MongoDB і сервером, як показано з рядка 14 по рядок 12.

cookie-parser, у рядку 6, заповнює всі файли cookie у властивості request (або req) об'єктом, ключовим для якого є імена файлів cookie після декодування заголовка Cookie.

Що стосується встановлення зв'язку між базою даних MongoDB і сервером, то MONGO_URI є незамінним. Після виконання необхідних дій на веб-сайті MongoDB Atlas створюється кластер, налаштований на хмарного провайдера та місцезнаходження. Потім надається рядок з'єднання, який додається до файлу .env як MONGO_URL. Нарешті, MONGO_URL вказується у файлі config.js як MONGO_URI, що показано на рисунку 3.3 вище. У рядку 22 термінал записує повідомлення «Connect to MongoDB», коли з'єднання буде успішним. Крім того, буде скомпільовано повідомлення про помилку з'єднання, як показано у рядку 24 на Рисунку 3.3 вище. Файл index.js просто імпортує власне додаток з файлу app.js і пізніше запускає додаток, як показано на рисунку 3.4 нижче.

```
1  const app = require('./app')
2  const http = require('http')
3  const config = require('./utils/config')
4  const logger = require('./utils/logger')
5
6  const server = http.createServer(app)
7
8  server.listen(config.PORT, () => {
9    logger.info('Server is running on port', config.PORT)
10 })
```

Рисунок 3.4. Вміст файлу index.js.

У рядку 4 відокремлено логгер, щоб усіма повідомленнями логу можна було керувати в одному місці для запису логів у файл або надсилання їх зовнішнім сервісам керування логами, таким як Graylog та Papertrail. Для виводу на консоль використовується інформація про функції модуля логера, яка вказує на те, що в даний момент додаток виконується на порту 3003 з config.PORT config-модуля. Відповідь «Running...» від додатку прийшла з кореневої URL-адреси, завантаженої з app.js. Після успішної симуляції функціонального сервера до уваги береться повна версія сервера для додатку. В результаті всі файли створюються і

організовуються в окремі папки для різних цілей, як показано на Рисунку 3.5 нижче.

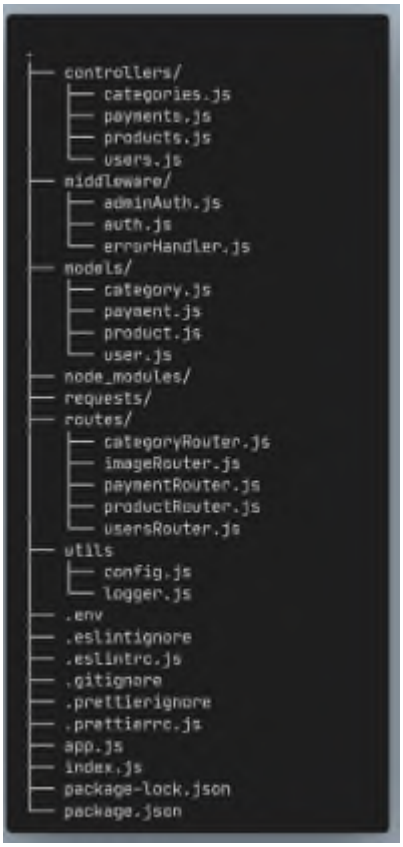


Рисунок 3.5 Оновлена структура папок на сервері.

Як показано на рисунку 3.5, папка server містить шість підпапок, основними з яких є контролери, проміжне програмне забезпечення, моделі та маршрути. Тека models містить усі необхідні mongoose-схеми для програми. Тим часом, у папці «контролери» зберігаються всі логічні обробники подій, які виконують вхідні запити, що відповідають відповідним маршрутам API, визначеним у папці routes.

Для реалізації серверної частини на Node.js з використанням бібліотеки Express та бази даних MongoDB з бібліотекою Mongoose була описана така логіка:

Рисунок 3.6 Підключення до Mongo DB

У цьому коді ми спочатку підключаємо необхідні модулі Express та Mongoose. Потім встановлюємо з'єднання з базою даних MongoDB за допомогою mongoose.connect()

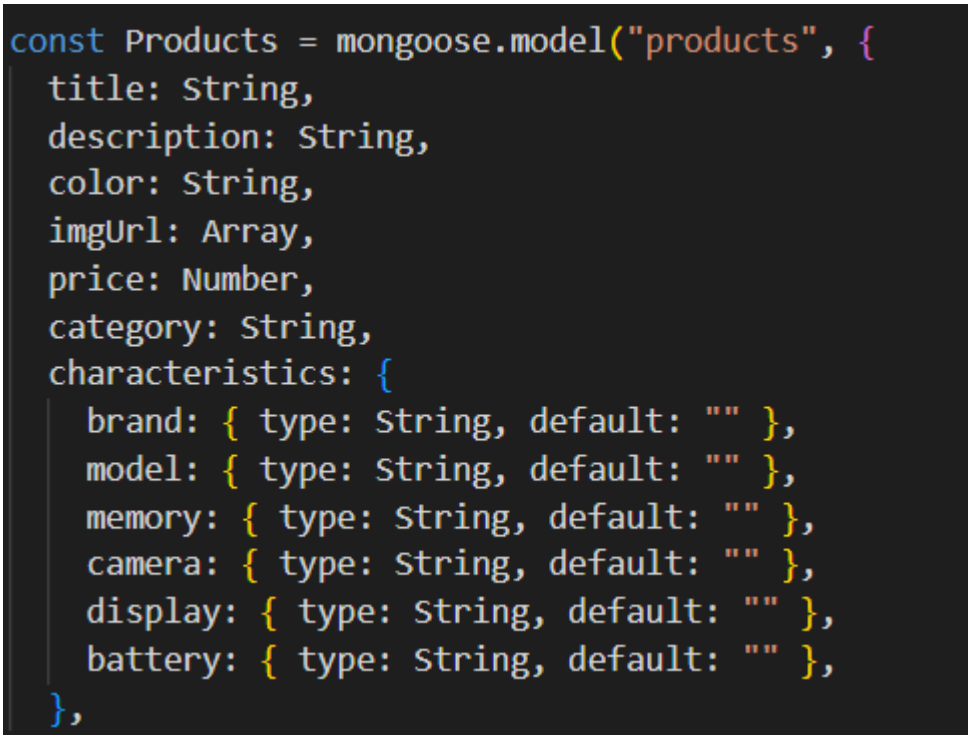


Рисунок 3.7- Структурний опис товарів

Ця модель документу MongoDB для колекції "products" містить поля заголовка, опису, кольору, масиву URL-адрес зображень, ціни, категорії та характеристик продукту, таких як бренд, модель, пам'ять, камера, дисплей та батарея, з можливістю задання значень за замовчуванням.

```
app.get("/products", async (req, res) => {
  try {
    const { category, search } = req.query;

    let query = {};

    if (category) {
      query.category = category;
    }

    if (search) {
      query.title = { $regex: new RegExp(search, "i") };
    }

    const products = await Products.find(query);

    res.send({
      products,
    });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Internal Server Error" });
  }
});
```

Рисунок 3.8 - Категорії товарів

Цей маршрут /products обробляє запит GET для отримання списку продуктів з бази даних MongoDB з урахуванням можливості фільтрації за категорією та пошуком за заголовком продукту, а потім відправляє цей список у відповідь на запит, або повертає статус 500 з повідомленням про внутрішню помилку у разі її виникнення.

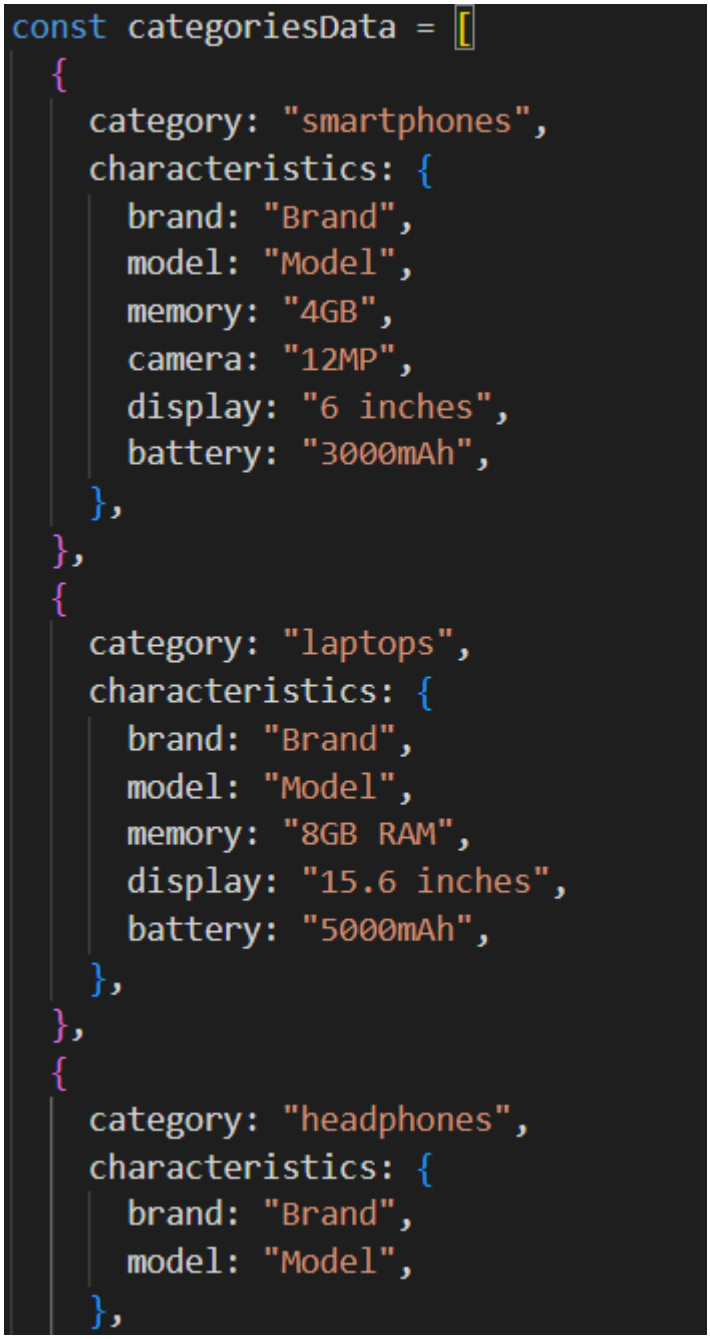


Рисунок 3.9 - Опис структурних даних для кожної з категорій продуктів

Ця частина коду відповідає за виклик функції для підключення до бази даних перед тим, як сервер почне приймати запити.

Рисунок 3.10 Запуск сервера на порті

Ця частина коду відповідає за виклик функції для підключення до бази даних перед тим, як сервер почне приймати запити.

3.2 Графічний інтерфейс

Графічний інтерфейс – це візуальний спосіб взаємодії користувачів з сайтом через елементи, такі як кнопки, текстові поля, зображення, меню та інші інтерактивні компоненти. Це дозволяє користувачам легко навігувати по сайту і взаємодіяти з його функціональністю без необхідності знати, як він програмно реалізований.

Я провів глибокий аналіз компонентів, які сприяють створенню ефективного та привабливого інтерфейсу. Моя ціль полягала у розробці інтерфейсу, який би не лише був візуально привабливим і зручним для користувачів, але й оптимізованим під потреби ринку мобільних технологій. На рисунку 3.11 зображено головна сторінка сайту:

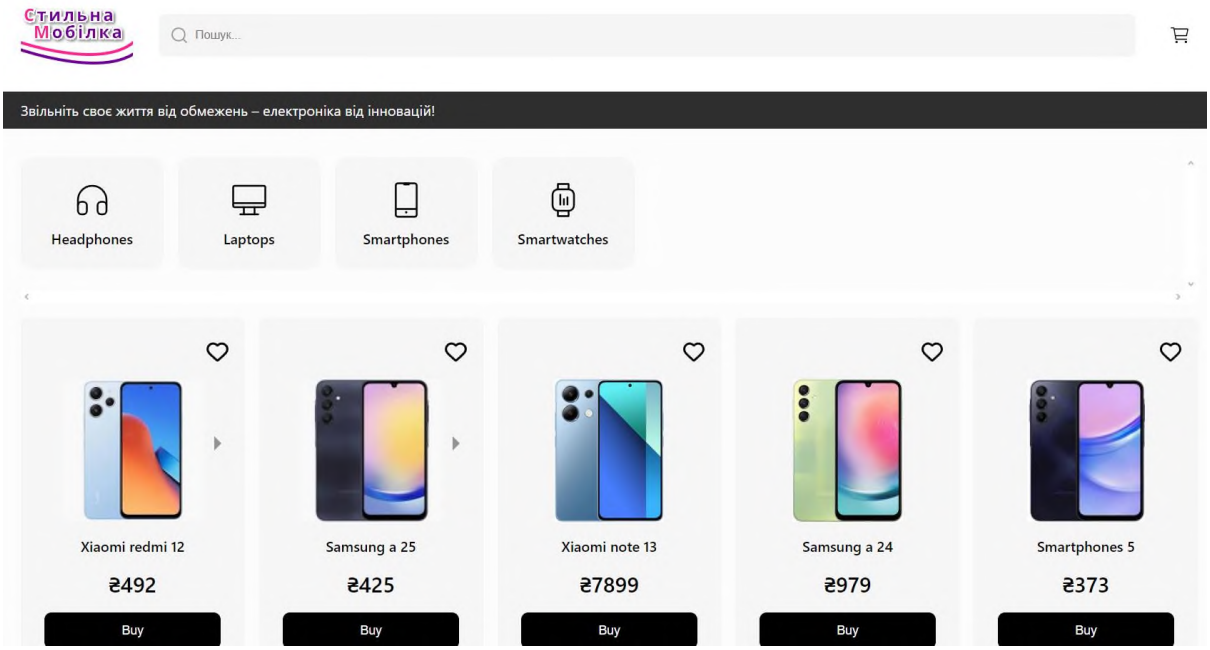


Рисунок 3.11 - Головна сторінка сайту.

Перш за все, важливо було зосередитися на якості зображень продуктів. Ясні, високорезолюційні фотографії з різних перспектив допомагають користувачам краще зрозуміти, що вони купують. Тому я інтегрував додаткові фотографії, які дозволяють користувачам роздивитись зображення та бачити деталі продукту без втрати якості.

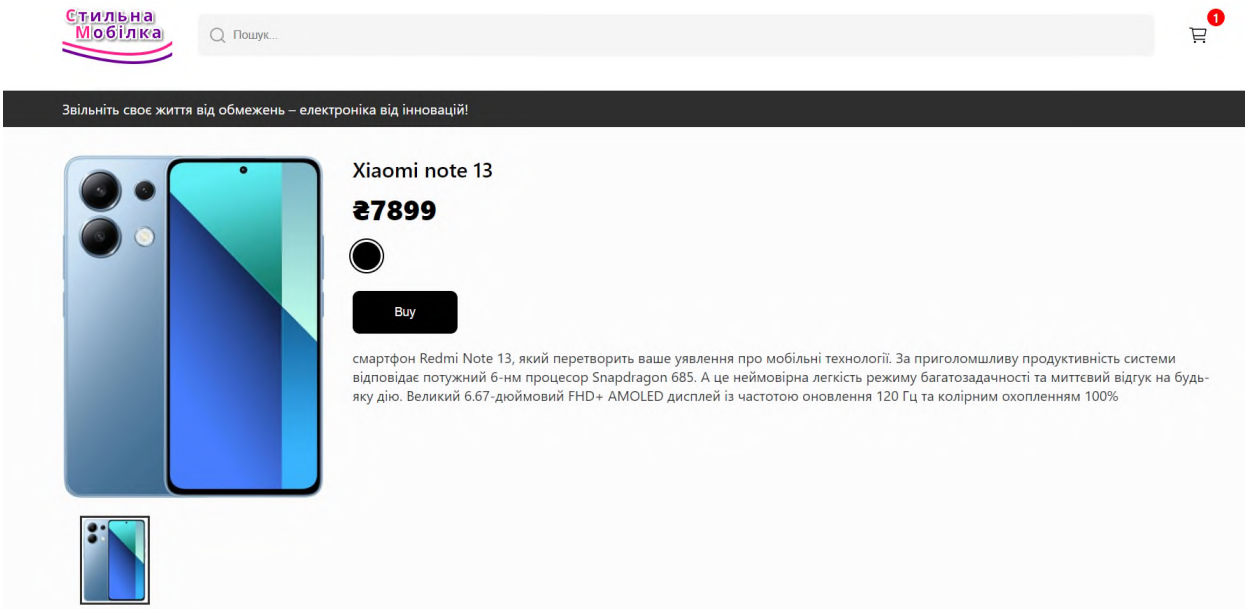


Рисунок. 3.12 - Сторінка товару

Особливу увагу було приділено функціональності пошуку та фільтрації, що дозволяє користувачам легко навігувати між категоріями та знаходити продукти за ключовими параметрами, такими як ціна, бренд, рейтинг та інші специфікації. Це важливо, оскільки спрощує процес вибору і зберігає час користувачів.

Для оптимізації користувацького досвіду, кнопки покупки та додавання в кошик були зроблені великими і кольоровими, щоб вони були легко видимими та доступними на будь-якій сторінці. Також я забезпечив, що сайт містить розділ відгуків з детальною інформацією про досвід інших користувачів, що є цінною інформацією при прийнятті рішення про купівлю.

У процесі розробки велику увагу було приділено мобільній версії сайту. Враховуючи, що більшість користувачів відвідують веб-сайти через мобільні пристрої, було критично важливо адаптувати дизайн так, щоб він залишався функціональним і легким для навігації навіть на маленьких екранах.

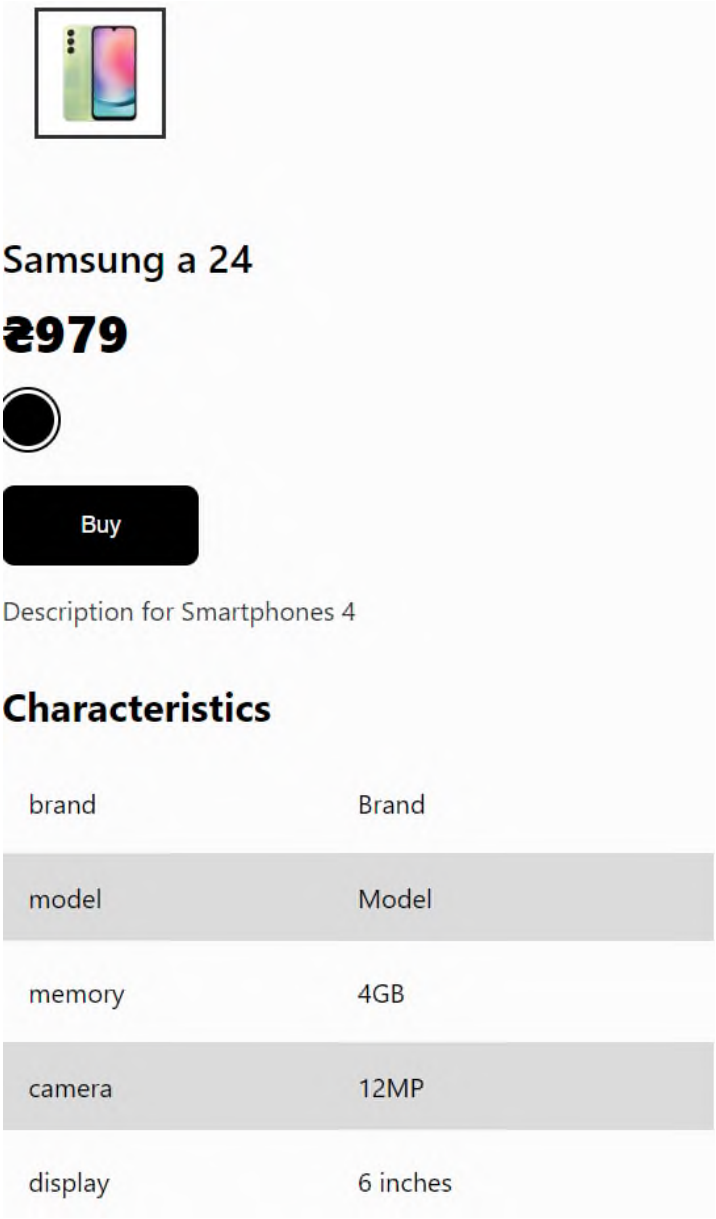


Рисунок. 3.13- Характеристики товару

Також впровадилось розширені можливості для інтерактивності, такі як анімаційні ефекти для загрузки сторінок та переходів, що робить взаємодію більш приємною і динамічною. Все це допомагає залучити та утримати увагу користувача, підвищуючи загальну ефективність сайту.

Закінчуючи впровадження цих елементів створює солідну основу для інтерфейсу, який не тільки виглядає професійно, але й забезпечує високий рівень користувацького досвіду. Використання сучасного дизайну, оптимізованих функцій пошуку та навігації, а також міцний фокус на мобільну адаптацію, дозволяють цьому інтерфейсу виступати як ефективний інструмент у світі електронної комерції.

3.3 Тестування

Під час розробки сервера також береться до уваги тестування, щоб переконатися, що логіка обробки запитів працює належним чином. Тому сторонній інструмент або розширення є кращим варіантом, ніж написання тестового коду в папці сервера.

Тому Postman є популярним вибором для цієї ситуації. Однак у нього є чимало недоліків. По-перше, це окремий додаток, а отже, його потрібно завантажити та встановити. Тому він вимагає перемикання контексту, оскільки відокремлюється від IDE, що використовується для розробки, що може спричинити споживання оперативної пам'яті та сповільнення роботи комп'ютера.

По-друге, контроль версій не дуже добре підтримується у безкоштовній версії, і він сильно залежить від графічного інтерфейсу Postman, який відрізняється від Git'a для контролю версій у проекті. REST Client - розширення VSCode для надсилання HTTP-запитів і перегляду відповіді безпосередньо у VSCode - розглядається як засіб вирішення цих проблем. Створивши після встановлення плагіна в корені серверної програми каталог з назвою requests, можна зберігати всі запити REST-клієнта в цьому каталозі у вигляді файлів із суфіксом «.rest». На рисунках 3.14 і 3.15 показано тестування видалення продукту за допомогою REST-клієнта.

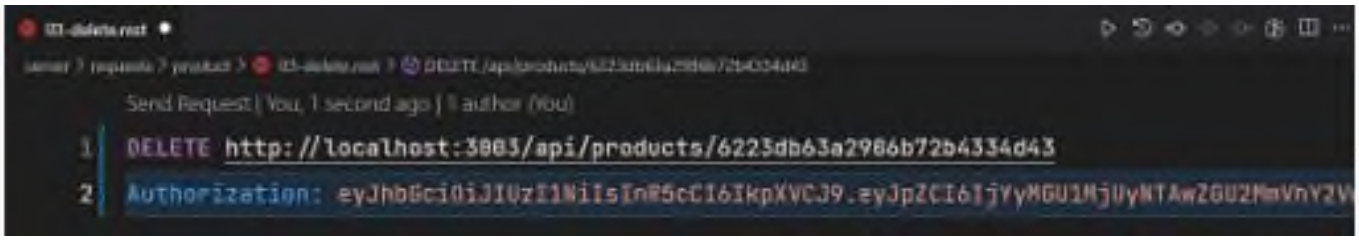


Рисунок 3.14. Скріншот тестування видалення продукту за допомогою REST-клієнта.

На Рисунку 3.14 HTTP-запит DELETE витягує продукт з відповідним ідентифікатором, визначеним у лівому вікні. Заголовок авторизації також містить токен, що підтверджує користувача на основі його ролі. Після натискання на текст

«Відправити запит» HTTP-запит виконується, і відповідь сервера відкривається в новій вкладці, як показано на малюнку 3.15. В результаті буде отримано статус «204 No Content», який вказує на те, що продукту більше немає в базі даних і видалення пройшло успішно.

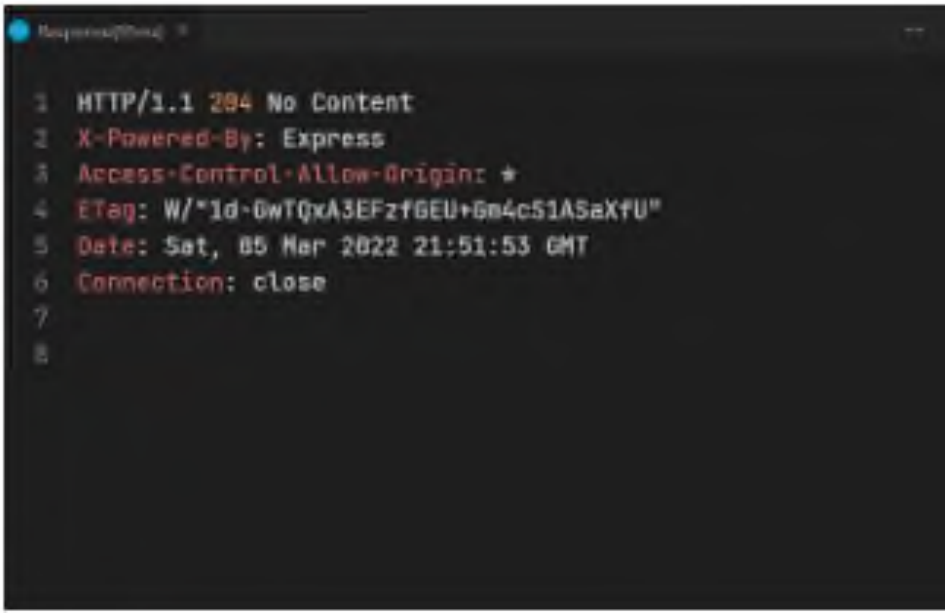


Рисунок 3.15. Результат тестування видалення товару за допомогою REST-клієнта.

ВИСНОВОКИ

У сучасному цифровому світі розвиток електронної комерції відіграє ключову роль у формуванні та трансформації бізнесу. Росте значення інтернет-магазинів як засобу забезпечення ефективної торгівлі та задоволення потреб споживачів. Створення ефективної та надійної веб-платформи для електронної комерції стає важливим завданням для багатьох підприємств та розробників.

У даній кваліфікаційній роботі було досліджено процес розробки веб-сторінки для продажу смартфонів з використанням передових технологій веб-розробки та документно-орієнтованої системи керування базами даних. Проведено аналіз предметної області, огляд різних типів баз даних для електронної комерції та дослідження можливостей документно-орієнтованих систем керування базами даних.

Результатом цього дослідження стало створення ефективної та сучасної веб-платформи для продажу смартфонів, яка відповідає сучасним вимогам електронної комерції. Розроблена система дозволяє користувачам здійснювати зручні та безпечні покупки, отримуючи доступ до широкого асортименту товарів та послуг.

У цій роботі були виконані всі поставлені завдання, а саме проведений аналіз, розроблено архітектуру, реалізовано веб-платформу, проведено тестування та оптимізацію. Отримані результати підтверджують досягнення поставленої мети та вказують на успішне вирішення поставлених завдань.

Ця кваліфікаційна робота є важливим внеском у вивчення та розвиток сучасних технологій електронної комерції та веб-розробки. Результати дослідження можуть бути використані для подальших наукових досліджень та розвитку практичних застосувань у галузі електронної торгівлі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aggarwal, S., 2018. Сучасна веб-розробка за допомогою ReactJS. Міжнародний журнал останніх дослідницьких аспектів, 5(1), с. 133.

2. Baron, D., 2021. Альтернатива VS Code для Postman. [online] Daniela Baron. Доступно на: https://danielabaron.me/blog/postman-alternative-vscode/ [Дата доступу: 9 березня 2024].

3. DeBill, E., 2022. Module Counts. [online] Доступно на: http://www.modulecounts.com/ [Дата доступу: 30 березня 2024].

4. Facebook Inc., 2022a. Компоненти та Props. [online] Доступно на: https://reactjs.org/docs/components-and-props.html [Дата доступу: 9 квітня 2024].

5. Facebook Inc., 2022b. Представлення Hooks – React. [online] Доступно на: https://reactjs.org/docs/hooks-intro.html [Дата доступу: 5 травня 2024].

6. Facebook Inc., 2022c. Представлення JSX. [online] Доступно на: https://reactjs.org/docs/introducing-jsx.html [Дата доступу: 9 квітня 2024].

7. Facebook Inc., 2022d. React – JavaScript бібліотека для побудови інтерфейсів користувача. [online] Доступно на: https://reactjs.org/ [Дата доступу: 24 квітня 2024]

8. Facebook Inc., 2022e. Узгодження. [online] Доступно на: https://reactjs.org/docs/reconciliation.html [Дата доступу: 24 квітня 2024].

9. Facebook Inc., 2022f. Використання ефекту Hook – React. [online] Доступно на: https://reactjs.org/docs/hooks-effect.html [Дата доступу: 12 травня 2024].

10. Facebook Inc., 2022g. Використання стану Hook – React. [online] Доступно на: https://reactjs.org/docs/hooks-state.html [Дата доступу: 12 травня 2024].

11. Facebook Inc., 2022h. Віртуальний DOM та внутрішності. [online] Доступно на: https://reactjs.org/docs/faq-internals.html [Дата доступу: 9 квітня 2024].

12. FullStackOpen, 2022. Повноцінна стек, частина 4. [online] FullStackOpen. Доступно на: https://fullstackopen.com/en/part4 [Дата доступу: 12 березня 2024].

13. GeeksforGeeks, 2021. Що таке MongoDB - робота та особливості. [online] GeeksforGeeks. Доступно на: https://www.geeksforgeeks.org/what-is-mongodbworking-and-features/ [Дата доступу: 10 квітня 2024].

14. Gupta, L., 2022. Що таке REST. [онлайн] Посібник з REST API. Доступно на: https://restfulapi.net/ [Дата доступу: 24 квітня 2024].

15. Hamedani, M., 2018. React Virtual DOM пояснений простою англійською - Програмування з Mosh. [онлайн] Програмування з Mosh. Доступно на: https://programmingwithmosh.com/react/react-virtual-dom-explained/ [Дата доступу: 24 квітня 2024].

16. Heroku. н.р. Облаштування хмарної платформи | Heroku. [онлайн] Доступно на: https://www.heroku.com/ [Дата доступу: 24 квітня 2024].

17. Ho, C., 2016. Розуміння шаблону Middleware в Express.js. [онлайн] DZone. Доступно на: https://dzone.com/articles/understanding-middlewarepattern-in-expressjs [Дата доступу: 29 квітня 2024].

18. Karnik, N., 2018. Вступ до Mongoose для MongoDB | Codementor. [онлайн] Codementor. Доступно на: https://www.codementor.io/@theoutlander/introduction-to-mongoose-formongodb-gw9xw34el [Дата доступу: 24 квітня 2024].

19. Keshishyan, A., 2019. Архітектура подійного циклу Node.js. [онлайн] Medium. Доступно на: https://medium.com/preezma/node-js-event-loop-architecture-godeeper-node-core-c96b4cec7aa4 [Дата доступу: 29 квітня 2024].

20. MDN Web Docs. 2022a. Спільне використання ресурсів між різними доменами (CORS) - HTTP | MDN. [онлайн] Доступно на: https://developer.mozilla.org/enUS/docs/Web/HTTP/CORS [Дата доступу: 10 березня 2024].

21. MDN Web Docs. 2022b. Вступ до DOM - Веб-інтерфейси | MDN. [онлайн] Доступно на: https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model/Introduction [Дата доступу: 24 квітня 2024].

22. MDN Web Docs. 2022c. Використання HTTP cookies - HTTP | MDN. [онлайн] Доступно на: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies [Дата доступу: 09 лютого 2024].

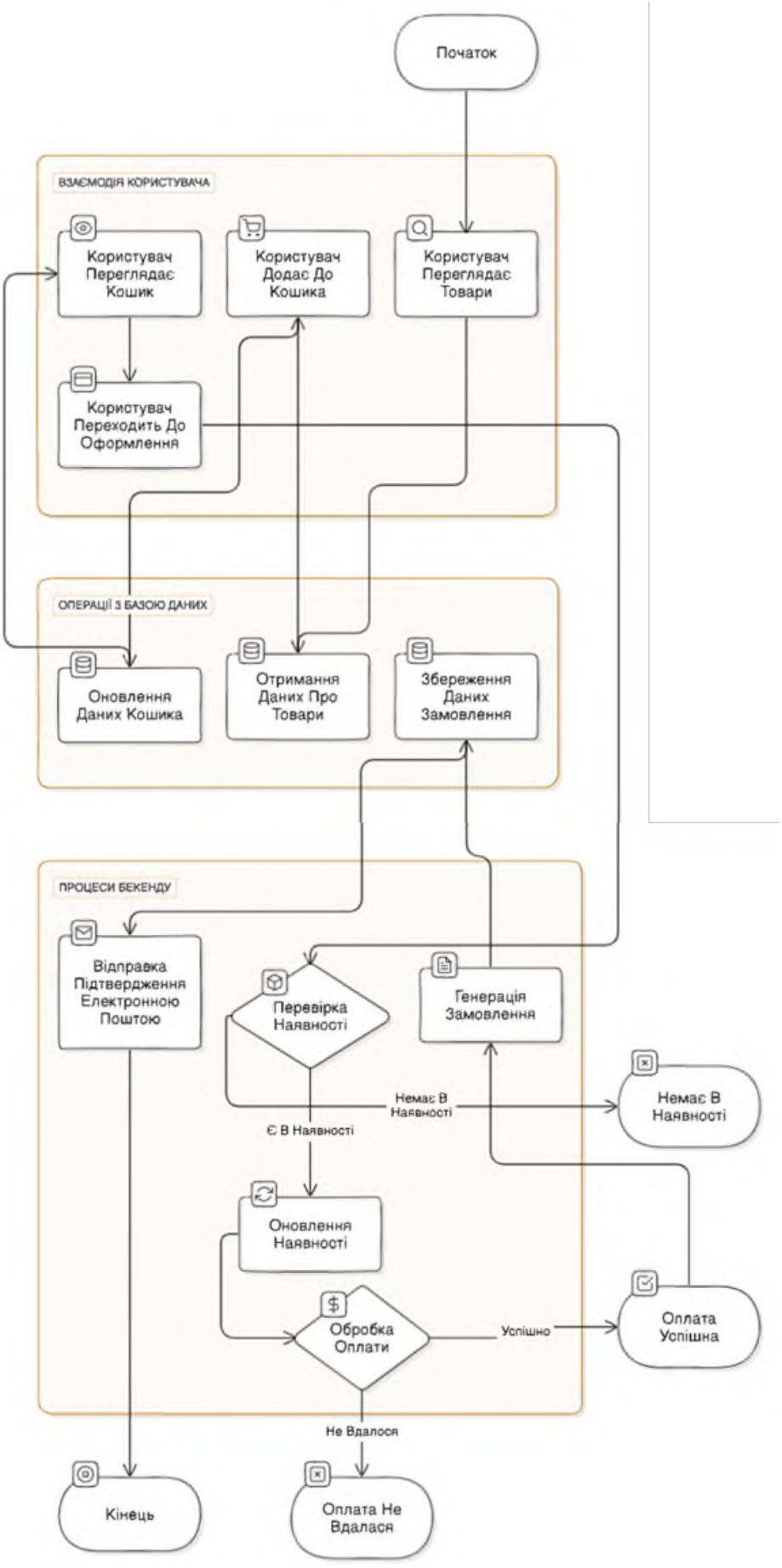
23. MongoDB Inc., 2013. Стек MEAN: MongoDB, ExpressJS, AngularJS та Node.js | Блог MongoDB. [онлайн] MongoDB. Доступно на: https://www.mongodb.com/blog/post/the-mean-stack-mongodb-expressjsangularjs-and [Дата доступу: 29 квітня 2024].

24. Каравець Р. О. «РОЗРОБКА ВЕБ-ПЛАТФОРМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ ДОКУМЕНТО-ОРІЄНТОВАНОЇ СИСТЕМИ КЕРУВАННЯ БАЗОЮ ДАНИХ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

25. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

ДОДАТОК А

Алгоритм роботи веб-платформи



ДОДАТОК Б

Апробація отриманих результатів

РОЗРОБКА ВЕБ-ПЛАТФОРМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ ДОКУМЕНТО-ОРІЄНТОВАНОЇ СИСТЕМИ КЕРУВАННЯ БАЗОЮ ДАНИХ

Каравець Роман Олексійович
здобувач вищої освіти факультету
комп'ютерних інформаційних технологій
Західноукраїнський національний університет, Україна
Науковий керівник: Кіт Іван Романович
викидач кафедри інформаційно обчислювальних систем і управління
Західноукраїнський національний університет, Україна

У сучасному цифровому ландшафті електронна комерція стала важливим аспектом розвитку бізнесу. Зручність та доступність покупок в Інтернеті залучають мільйони споживачів щодня. Для забезпечення успішної діяльності інтернет-магазинів необхідна надійна та ефективна система управління базами даних, яка може обробляти великий обсяг інформації і забезпечувати швидкий доступ до неї.

Розробка веб-платформи для електронної комерції, яка використовує документо-орієнтовану систему керування базами даних для забезпечення гнучкості, ефективності та надійності роботи з даними.

Завдання дослідження:

1. Провести аналіз предметної області електронної комерції.
2. Зробити огляд баз даних, які використовуються для електронної комерції.
3. Дослідити документо-орієнтовані системи керування базами даних.
4. Розробити архітектуру веб-платформи.
5. Реалізувати веб-платформу.
6. Провести тестування та оптимізацію платформи.

Веб-платформа електронної комерції.

Використання документо-орієнтованої системи керування базами даних у веб-платформах електронної комерції.

Аналіз, синтез, моделювання, розробка програмного забезпечення, тестування та оптимізація.

Запропоновано використання документо-орієнтованих систем керування базами даних для підвищення ефективності та гнучкості роботи веб-платформ електронної комерції.

Розроблена веб-платформа може бути використана для створення сучасних інтернет-магазинів, що забезпечують зручний та ефективний процес покупок для користувачів.

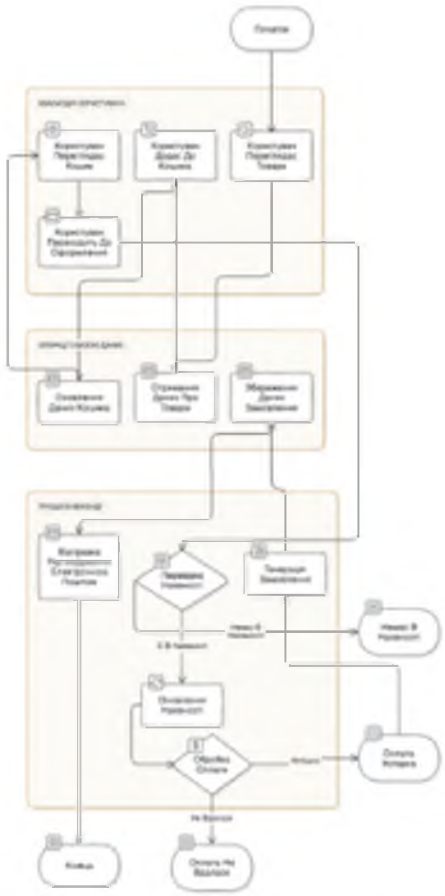


Рис.1. **Алгоритм роботи веб-платформи**

У сучасному цифровому світі розвиток електронної комерції відіграє ключову роль у формуванні та трансформації бізнесу. Інтернет-магазини стають ефективним засобом забезпечення торгівлі та задоволення потреб споживачів. Створення надійної та ефективної веб-платформи для електронної комерції є важливим завданням для багатьох підприємств.

Список використаних джерел

1. Aggarwal S. (2018). "Сучасна веб-розробка за допомогою ReactJS". Міжнародний журнал останніх дослідницьких аспектів, 5(1), с. 133.
2. Facebook Inc. (2022). "React – JavaScript бібліотека для побудови інтерфейсів користувача".
3. GeeksforGeeks (2021). "Що таке MongoDB - робота та особливості".
4. Ho C. (2016). "Розуміння шаблону Middleware в Express.js". DZone.