Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедр Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Гордієнка Данила Дмитровича*
(ПІБ)

академічної групи *121-20-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка інтернет-магазину електронних пристроїв та аксесуарів з використанням React, MobX, Sequelize*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20-1* (група) *Гордієнка Данила Дмитровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтернет-магазину електронних пристроїв та аксесуарів з використанням React, MobX, Sequelize*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2024 р.***

Завдання видав _____ (підпис) *доц. Спірінцев В.В* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Гордієнко Д.Д.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *17.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 76 с., 14 рис., 3 дод., 22 джерел.

Об'єкт розробки: веб-орієнтований додаток з використанням бібліотеки React та ORM Sequelize для інтернет-магазину, що спеціалізується на продажу електронної продукції.

Мета кваліфікаційної роботи: розробка інтернет-магазину електронної техніки з використанням React, MobX, Sequelize.

У вступі аналізується та досліджується поточний стан проблеми, визначається мета кваліфікаційної роботи та сфера її застосування. Також надається обґрунтування важливості обраної теми й формулюється список завдань.

У першому розділі було проведено аналіз предметної галузі, визначено, наскільки актуальне та необхідне це завдання, сформульовано його мету, вказано необхідні вимоги до програмного забезпечення, технологій та інших програмних інструментів.

У другому розділі були проаналізовані існуючі рішення, визначено оптимальну платформу для подальшої розробки, проведено проектування та розробку веб-орієнтованої інформаційної системи. Описано роботу системи, включаючи алгоритм та структуру її функціонування, а також процес виклику та завантаження додатку. Визначено вхідні та вихідні дані, і охарактеризовано склад параметрів технічних засобів, необхідних для правильної роботи системи.

В економічному розділі була встановлена складність розробки інформаційної системи, розраховані витрати на створення додатку та оцінений час, необхідний для його втілення.

Практичне значення полягає у створенні веб-орієнтованого додатку, що забезпечує: можлививість компанії привертати більше клієнтів, збільшувати обсяги продажів і підвищувати свою конкурентоспроможність на ринку. Така система дозволить забезпечити зручність для клієнтів, швидкий доступ до товарів і послуг, а також ефективне управління процесами продажу та складання звітності для компаній.

Актуальність розробки веб-орієнтованого додатку для компанії в сфері електронної комерції залишається високою незалежно від обставин, оскільки він забезпечує ефективну взаємодію з клієнтами та підтримку продажів. У сучасних умовах війни та економічної нестабільності проект набуває ще більшого значення. Онлайн-інструменти для збільшення продажів і залучення нових клієнтів стають ключовими для збереження компанії на ринку у важкі часи.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, JAVA-SCRIPT, REACT, NODEJS, SEQUELIZE.

ABSTRACT

Explanatory note: 76 pp., 14 fig., 3 appendices, 22 sources.

Object of development: A web-oriented application using the React library and ORM Sequelize for an online store specializing in the sale of electronic products.

Purpose of the qualification work: development of an online electronics store using React, MobX, Sequelize.

Introduction: The introduction analyzes and investigates the current state of the problem, defines the goal of the qualification work, and the scope of its application. It also provides a justification for the importance of the chosen topic and formulates the list of tasks.

In the introduction, an analysis of the subject area, determines the relevance and necessity of the task, formulates its goal, and specifies the requirements for the software implementation, technologies, and other software tools.

In the second section, analyzes existing solutions, selects the optimal platform for further development, and carries out the design and development of the web-oriented information system. It describes the system's operation, including its algorithm and structure, as well as the process of invoking and loading the application. It also determines the input and output data and characterizes the parameters of the technical means required for the system's proper functioning.

In the economic section, establishes the complexity of the information system development, calculates the costs of creating the application, and estimates the time required for its implementation.

The practical value lies in the creation of a web-oriented application that allows companies to attract more customers, increase sales volumes, and enhance their competitiveness in the market. This system ensures convenience for customers, quick access to goods and services, and efficient management of sales processes and company reporting.

The relevance of developing a web-oriented application for a company in the field of e-commerce remains high regardless of circumstances, as it ensures effective interaction with clients and supports sales. In the current conditions of war and economic instability, this project gains even greater significance. Online tools for increasing sales and attracting new customers become key to maintaining the company’s presence in the market during challenging times.

List of Keywords: INFORMATION SYSTEM, ONLINE STORE, DATABASE, JAVASCRIPT, REACT, NODEJS, SEQUELIZE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IT - інформаційні технології

КС - комп'ютерна система

React - бібліотека JavaScript для розробки інтерфейсів користувача

MobX - бібліотека для управління станом додатків на JavaScript

Sequelize - ORM (Object-Relational Mapping) для Node.js, який підтримує різні бази даних

API - Application Programming Interface (Інтерфейс програмування додатків)

UI - User Interface (Інтерфейс користувача)

UX - User Experience (Взаємодія користувача)

HTTP - Hypertext Transfer Protocol (Протокол передачі гіпертексту)

URL - Uniform Resource Locator (Уніфікований локатор ресурсів)

JWT - JSON Web Token (JSON-токен)

SQL - Structured Query Language (Мова структурованих запитів)

API - Application Programming Interface (Інтерфейс програмування додатків)

ВСТУП

Із розвитком технологій та зростанням цифрової активності, розробка інтернет-магазинів стає невід'ємною частиною успішного бізнесу с продажом товарів. Сьогодні споживачі все частіше віддають перевагу покупкам онлайн, а якісні та зручні інтернет-магазини відіграють важливу роль у залученні та утриманні цільових клієнтів.

Інтернет-магазини, побудовані з використанням React, та mobX, мають величезний потенціал зростання на швидкозмінному цифровому ринку, оскільки:

– React дозволяє динамічно оновлювати вміст сторінки без перезавантаження, роблячи процес покупки зручнішим і швидшим для користувача

– MobX відповідає за ефективне управління станом додатку, що важливо для інтерфейсів магазину. Використовуючи сучасні інструменти, він може вивчати поведінку користувачів, оптимізувати процеси продажів і реклами, швидко реагувати на зміни попиту і тенденції .

Впровадження передових технологій не тільки покращує клієнтський досвід, але й підвищує загальну ефективність магазину, роблячи його більш конкурентоспроможним та успішним у довгостроковій перспективі.

Актуальність розробки веб-орієнтованого додатку з продажу електронних товарів для діяльності компанії в сфері електронної комерції полягає в тому, що:

– Веб-додаток дозволяє компанії охопити глобальну аудиторію, залучаючи клієнтів з різних регіонів, що збільшує потенційні продажі та доходи.

– Сучасні споживачі очікують зручного і швидкого доступу до товарів та послуг через інтернет. Веб-додаток забезпечує легкий доступ до каталогу товарів, можливість перегляду характеристик, порівняння цін та швидкого оформлення замовлень.

– Автоматизація багатьох процесів (наприклад, обробка замовлень, управління запасами, оплата та доставка) знижує навантаження на персонал і мінімізує ризик помилок, що покращує ефективність компанії.

– Наявність сучасного веб-додатку є важливим фактором конкурентоспроможності. Компанії, що не використовують цифрові технології, ризикують втратити клієнтів на користь конкурентів, які надають більш зручний і сучасний сервіс.

– Веб-додаток дозволяє компанії швидко реагувати на запити клієнтів, надавати оперативну підтримку та підвищувати загальний рівень обслуговування, що сприяє формуванню лояльності та збільшенню кількості повторних покупок.

– Веб-додаток може бути інтегрований з соціальними мережами, платіжними системами, логістичними сервісами та іншими платформами, що розширює можливості взаємодії з клієнтами та підвищує загальну ефективність бізнесу.

– Веб-додаток дозволяє легко адаптуватися до змін на ринку, швидко оновлювати інформацію про товари, впроваджувати нові функції та розширювати асортимент відповідно до попиту.

Мета кваліфікаційної роботи: створення веб-додатку для інтернет-магазину електронної техніки з використанням технологій React, MobX та Sequelize. Головний фокус робиться на оптимізації роботи магазину шляхом активної участі в електронній комерції.

Для реалізації поставленої мети необхідно вирішити наступні задачі:

– Оцінити потреби та очікування користувачів щодо функціональності та інтерфейсу інтернет-магазину електронної техніки.

– Створити архітектуру бази даних для зберігання інформації про товари, користувачів, замовлення тощо з використанням Sequelize.

– Створити інтуїтивно зрозумілий та привабливий інтерфейс за допомогою бібліотеки React, що дозволить користувачам зручно переглядати товари та здійснювати покупки.

– Використати MobX для ефективного управління станом додатку, забезпечуючи швидку та плавну роботу інтерфейсу та оптимізуючи використання ресурсів.

– Тестування додатку для виявлення та виправлення можливих помилок та недоліків.

– Розгорнути веб-додаток на сервері та забезпечити його стабільну роботу, а також здійснювати підтримку та вдосконалення згідно з потребами користувачів та змінами на ринку електронної комерції.

Розробка цього додатку має конкретне практичне значення як для бізнесу, так і його клієнтів. Для бізнесу це означає можливість ефективно представляти свої товари в онлайн просторі, що розширює аудиторію та збільшує потенційні продажі. Веб-додаток надає зручність у веденні та керуванні асортиментом товарів, обробці замовлень, а також збиранні та аналізі даних про покупців для підвищення ефективності маркетингу та стратегій продажів. Для клієнтів це означає зручний доступ до широкого асортименту електронної техніки, можливість порівняння товарів та їх характеристик, а також зручний процес оформлення замовлення та сплати онлайн. Вони отримують можливість здійснювати покупки безпосередньо з дому чи з мобільного пристрою, що забезпечує високий рівень комфорту та зручності. В результаті, сприяє покращенню як бізнесових, так і користувацьких переваг, роблячи процес покупок електронної техніки більш зручним, ефективним та доступним.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Особливості електронної комерції в України

За останні роки кількість українських покупців в електронній комерції зросла. Це пов'язано з кількома ключовими факторами:

– Зростання доступу до інтернету: з кожним роком населення України має все більше доступу до інтернету. Це зростання супроводжується збільшенням кількості людей, які використовують інтернет для покупок. Зручність онлайн-покупок доступна все більшій кількості людей, незалежно від того, де вони живуть.

– Розвиток мобільних технологій: поширення смартфонів і планшетів дозволяє споживачам здійснювати покупки на ходу і в будь-який час доби. Мобільна електронна комерція стає все більш популярною серед українських користувачів

– Розвиток систем онлайн-платежів: поява нових платіжних систем та вдосконалення існуючих сприятиме зручності та безпеці онлайн-покупок. Споживачі довіряють цим системам і готові платити онлайн.

– Широкий асортимент товарів: інтернет-магазини пропонують широкий спектр товарів і послуг, від електроніки та моди до продуктів харчування та послуг доставки. Таким чином, споживачі можуть знайти все, що їм потрібно, не виходячи з дому.

– Маркетинг і реклама: рекламні кампанії та онлайн-маркетингові заходи допомагають привернути увагу споживачів до товарів і послуг, що пропонуються в електронній комерції. Ефективна реклама допомагає збільшити попит на електронну комерцію.

Однак воєнні дії суттєво вплинули на електронну комерцію українських компаній. З одного боку, зростаючий попит на онлайн-торгівлю стимулює розвиток бізнесу. З іншого боку, економічна нестабільність та високі ціни на бізнес-ресурси ускладнюють ведення бізнесу. На цих ринках підприємцям важливо бути гнучкими та швидко реагувати на зміни. Забезпечення високого рівня обслуговування є ключовим викликом, оскільки конкуренція є жорсткою, а споживачі продовжують фокусуватися на якості та комфорті. Окрім адаптації до нових умов, важливо також знаходити та впроваджувати інноваційні підходи, що відповідають потребам аудиторії. Крім того, в умовах війни етика та відповідальність бізнесу набувають ще більшого значення. Бізнес повинен діяти соціально відповідально та підтримувати своїх працівників і клієнтів у важкі часи.

1.1.2. Аналіз існуючих рішень

Для аналізу було обрано такі сайти як Touch [1] та Foxtrot [2], які являються лідерами ринку з продажу електронної техніки.

Головна сторінки сайтів (рис. 1.1 - 1.2) приваблюють своєю простотою та офіційним стилем, спрямованим на ефективну комунікацію з користувачем. Чистий та лаконічний дизайн сторінки підкреслює його функціональність і зручність використання.

На сайтах головна сторінка включає в себе:

– Шапка сайту: містить логотип компанії, назву, основне меню навігації (зазвичай з посиланнями на головні розділи сайту), поля для пошуку та вхід в обліковий запис.

– Карусель або слайдер: банери , та зображення, які презентують акції, нові товари чи основні пропозиції компанії. Це важливий елемент для привертання уваги відвідувачів і переходу до конкретних розділів або товарів.

– Каталог товарів: популярні або рекомендовані товари, які можна придбати на сайті. Це дає можливість користувачам швидко знайти продукти, які їх цікавлять.

– Акції та промоції: банери, які пропонують знижки, спеціальні умови або акції. Це може бути один з головних механізмів привертання нових клієнтів і стимулювання продажів.

– Інформація про компанію: короткий опис діяльності компанії, її цінності, історії або основних досягнень.

– Контактна інформація: контактні дані компанії, такі як адреса, номер телефону або форма зворотного зв'язку, що дозволяє клієнтам швидко зв'язатися з представниками компанії.

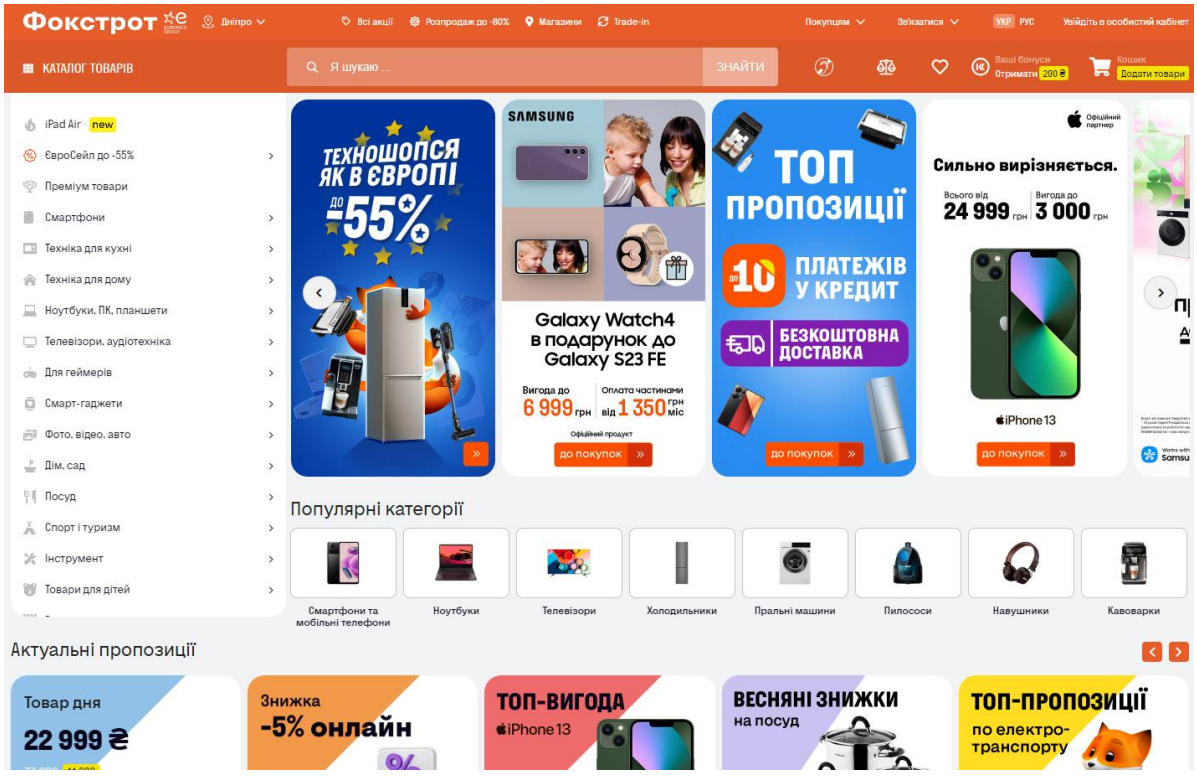

Рис. 1.1. Головна сторінка сайту Foxtrot

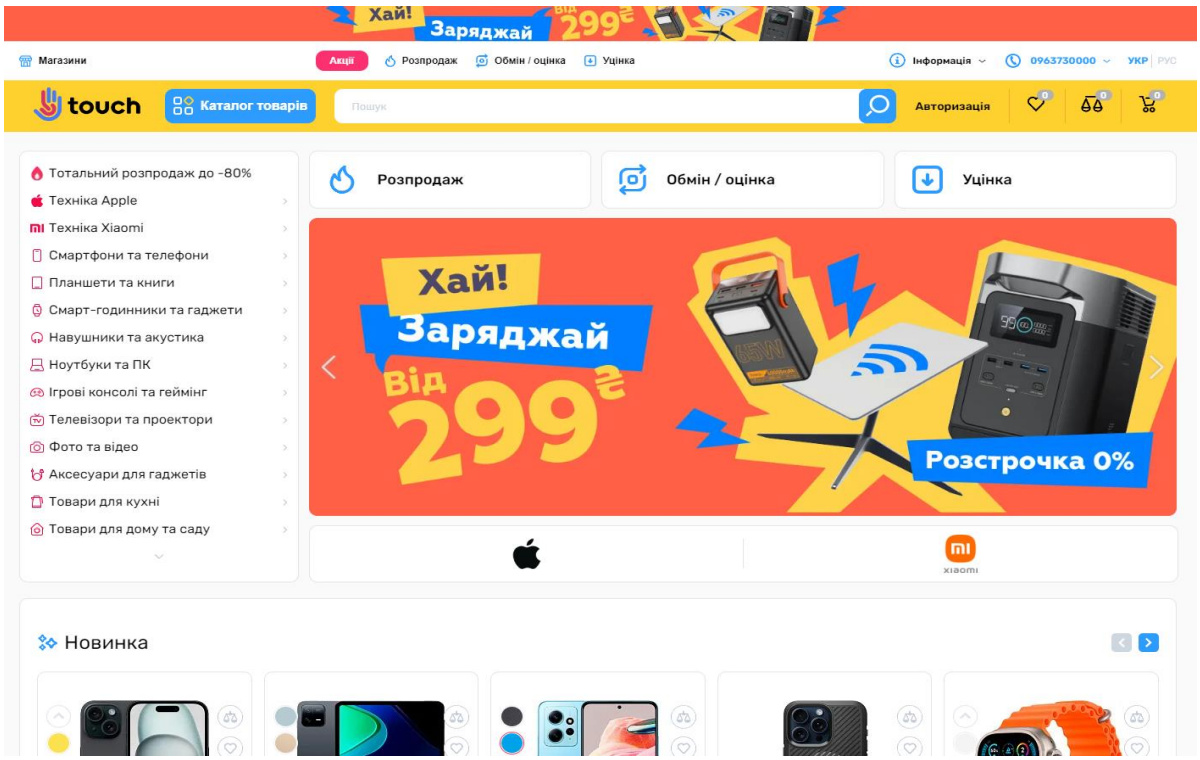

Рис. 1.2. Головна сторінка сайту Touch

Тож було зроблено висновок, що компоненти є ключовими для залучення користувачів та забезпечення комфортного перебування на сайті. Шапка сайту з логотипом, головним меню та полем пошуку підкреслює бренд і створює першу знайому точку входу, де користувачі можуть швидко знайти потрібну інформацію. Банери та слайдери з акціями та найпопулярнішими пропозиціями привертають увагу користувачів та стимулюють їхню активність. Каталоги товарів полегшують навігацію та допомагають користувачам швидко знайти потрібний продукт. Інформація про компанію та колонтитули надають додаткову інформацію та можливості комунікації, які покращують взаємодію з відвідувачами. Загалом, наявність цих компонентів створює гармонійне та привабливе середовище для успішної електронної комерції.

1.2. Призначення розробки та галузь застосування

У сучасному світі електронна комерція стала невід'ємною частиною повсякденного життя. Все більше клієнтів віддають перевагу зручності та

доступності онлайн-покупок. Тому виникає потреба в розробці інноваційних інтернет-магазинів, які не тільки пропонують зручне замовлення товарів, але й мають привабливий дизайн, що змушує клієнтів відвідувати їх знову і знову. У цьому розділі проаналізовано цілі та потреби для розробки інтернет-магазину Dom1store. Також пояснюється сфера застосування системи.

Призначення розробки:

– Зручне та швидке замовлення товарів. Основна мета інтернет-магазину Dom1store - створити сайт, на якому покупці зможуть швидко і легко придбати вподобану електроніку та аксесуари За допомогою Dom1store покупці можуть ефективно переглядати асортимент, шукати потрібний товар і додавати його в кошик Потім вони можуть знайти потрібний товар і додати його до кошика.

– Підвищення конкурентоспроможності компанії Dom1store може підвищити конкурентоспроможність компанії за допомогою електронної комерції. Інтернет залучає нових клієнтів, збільшує продажі та розширює бізнес компанії.

– Покращений користувальницький досвід. Одна з головних цілей розробки Dom1store - створення інтуїтивно зрозумілого, привабливого та зручного для клієнтів інтерфейсу Сучасні технології реактивного програмування, такі як React та MobX, гарантують простоту використання та створення швидких, адаптивних веб-інтерфейсів.

– Забезпечння безпеки та надійності. Безпека та захист персональних даних є пріоритетом у розвитку Dom1store, а такі технології, як Sequelize, забезпечують аутентифікацію та авторизацію, а також захист від потенційних кібератак.

Галузь застосування розробленного продукту спеціалізується на продажу широкого спектру електронних пристроїв, включаючи смартфони, ноутбуки, планшети, комп'ютери та інші гаджети. Кожен клієнт може відвідати сайт за допомогою комп'ютера або телефону і знайти широкий асортимент різних марок і моделей комп'ютерів і смартфонів за доступними цінами.

1.3. Підстави для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

– ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 469-с від 23.05.2024 р;

– завдання на кваліфікаційну роботу на тему «Розробка інтернет-магазину електронних пристроїв та аксесуарів з використанням React, MobX, Sequelize».

1.4. Постановка завдання

Техніко-економічна суть проблеми і причини необхідності її вирішення: створення інтернет-магазину у відповідь на зростаючий попит сучасного ринку обумовлено тим, що все більше користувачів віддають перевагу онлайн платформам для покупок. Основоню є те , що кожен клієнт може легко і швидко отримати той товар , що йому потрібно через інтернет. З точки зору його технічної та економічної сутності, це повинно бути використання новітніх технологій, які забезпечують ефективність, безпеку і надійність системи.

Основні вимоги до інтернет-магазину:

– Сайт повинен мати зрозумілу і привабливу структуру, легкий у навігації та зручний для використання.

– Сайт повинен бути оптимізований для перегляда на мобільних пристроях, оскільки все більше людей користуються смартфонами та планшетами для покупок.

– Забезпечення захисту конфіденційної інформації користувачів, такої як дані платіжних карток, є надзвичайно важливим.

– Сайт повинен швидко завантажуватися, оскільки користувачі не люблять чекати.

– Можливість додавати, оновлювати та видаляти товари та інші матеріали з легкістю.

– Ефективна система пошуку дозволяє користувачам швидко знаходити необхідні товари.

– Різноманітні методи оплати та доставки, щоб задовольнити різні потреби користувачів.

– Можливість відстежувати та аналізувати різноманітну інформацію про відвідування та покупки для удосконалення бізнесу.

Етапи стоверення додатку:

– Аналіз ринку та визначення цільової аудиторії.

– Розробка бізнес-плану.

– Вибір платформи для інтернет-магазину.

– Розробка дизайну та функціональності.

– Налаштування та інтеграція.

– Тестування.

– Запуск.

– Підтримка та оптимізація.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для успішного розвитку інтернет-магазину електроніки потрібно впровадити ряд функцій:

– Реєстрація та авторизація: щоб увійти в систему, користувачі повинні створити обліковий запис, ввівши особисті дані, такі як адреса електронної пошти, пароль, ім’я та фамілія. Крім того, система повинна дозволяти користувачам, які вже мають обліковий запис, входити в систему. По-перше, користувачі повинні мати можливість додавати товари до свого кошика, видаляти товари та змінювати їх кількість. Що стосується особистих кабінетів,

то користувачі повинні мати можливість мати власний обліковий запис, переглядати дані та інформацію, наприклад, історію замовлень, а також мати можливість змінювати персональні дані..

– Адміністративні функції. В управлінні продуктами адміністратор має право додавати, оновлювати, видаляти і контролювати класифікацію продуктів. В управлінні замовленнями адміністратор переглядає і контролює замовлення, розміщені користувачем, включаючи підтвердження, скасування або відправку замовлень.

– Часові характеристики. Продуктивність веб-браузера користувача залежить від швидкості, з якою обробляються ці запити; тобто сервер повинен обробляти запит в найкоротші терміни і відповідати протягом однієї-двох секунд, щоб забезпечити безперебійну і безпроблемну роботу. досвід. Швидкість завантаження сторінки: для забезпечення приємного користувацького досвіду веб-сайт повинен завантажуватися швидко і без помітних затримок.

– Організація та зберігання даних. Використовуючи Sequelize для ORM, дані про продукт, користувача та замовлення знаходяться в реляційній базі даних.

Існує кілька критеріїв для вибору формату представлення даних. Дані вводяться у форми кінцевим користувачем за допомогою веб-інтерфейсу, текстових полів, випадаючих списків, перемикачів тощо. Система надає зручні для читання вихідні дані, такі як брошури, таблиці введення даних, звіти та діаграми.

1.5.2. Вимоги до інформаційної безпеки

Інформаційна безпека забезпечує конфіденційність, цілісність та доступність інформації шляхом впровадження відповідних технічних, організаційних та правових заходів. Це включає заходи для захисту інформації від несанкціонованого доступу, зміни та знищення, а також для забезпечення доступу до інформації авторизованих користувачів у потрібний час. Основними

цілями інформаційної безпеки є запобігання втраті, крадіжці або пошкодженню інформації, а також забезпечення захисту від кіберзлочинності та інших загроз.

Реалізовані заходи безпеки системи включають:

– Валідація даних: здійснюється перевірка вхідних даних для запобігання вразливостям та недолікам у системі.

– Повідомлення про помилки: використовуються спеціальні повідомлення, щоб оперативно реагувати на проблеми та помилки у роботі системи.

– Шифрування паролів користувачів: застосовується надійний алгоритм шифрування BCrypt для захисту паролів користувачів від несанкціонованого доступу.

– Розподіл користувачів на ролі: користувачі системи класифікуються за ролями, щоб ефективно керувати їх доступом до різних ресурсів.

– Обмеження передачі даних: дані, які передаються з сервера, обмежуються лише до необхідної інформації, щоб забезпечити конфіденційність та захист даних.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для забезпечення належного функціонування веб-додатків необхідно враховувати вимоги до конфігурації та параметрів технічних заходів. Ключові моменти включають:

Клієнтська частина:

– Браузер: Microsoft Internet Explorer, Mozilla Firefox, Opera, Google Chrome.

– Доступ до Інтернету.

– Пристрої введення: клавіатура, комп'ютерна миша, або тач-пад.

Серверна частина:

– Операційна система: Windows XP, Windows 7 або вище.

– Оперативна пам'ять: мінімум 8 Гб.

– Вільне місце на диску: 10 Гб.

– Node.js.

– Швидкість Інтернет-підключення: 512 Кбіт/с і вище. Сервер бази даних:

– Процесор: Intel Core або Xeon 3ГГц (або Dual Core 2ГГц) чи подібний AMD процесор.

– Графічні прискорювачі: nVidia або ATI з підтримкою OpenGL 1.5 або вище.

Ці технічні вимоги сприяють безперебійному та надійному функціонуванню додатку, забезпечують швидкість обробки даних і високий рівень безпеки, що важливо для клієнтів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Інтернет-магазини потребують як програмної, так і інформаційної сумісності для забезпечення ефективної роботи та інтеграції з іншими системами. Розглянемо докладніше, що включає в себе кожен з цих аспектів і в чому полягають їхні відмінності.

Сумісність програмного забезпечення - це коректне функціонування інтернет-магазину на різних апаратних і програмних платформах. Вона охоплює всі технічні аспекти програми, включаючи сумісність з операційними системами, браузерами, мобільними пристроями та іншими програмними компонентами.

Основні аспекти програмної сумісності:

– Підтримка основних браузерів (Chrome, Firefox, Safari, Edge). Кросбраузерне тестування для забезпечення коректного відображення і функціонування сайту у всіх підтримуваних браузерах.

– Адаптивний дизайн, який забезпечує коректне відображення на різних розмірах екранів.

– Можливість розробки мобільних додатків (наприклад, з використанням React Native).

– Підтримка основних операційних систем (Windows, macOS, Linux) для кінцевих користувачів.

– Сумісність із серверними операційними системами для розгортання (наприклад, Ubuntu Server, CentOS).

– Використання React для створення динамічного інтерфейсу користувача.

– Використання Git для відстеження змін у коді.

– Налаштування CI/CD для автоматизації процесів тестування та розгортання.

– Оптимізація продуктивності на рівні бази даних, серверної логіки та фронтенду.

Інформаційна сумісність - це здатність інтернет-магазину обмінюватися даними з іншими системами та забезпечувати їхнє правильне зберігання й обробку. Це включає стандартизацію даних, інтеграцію з іншими системами та забезпечення точності й узгодженості даних.

Основні аспекти інформаційної сумісності:

– Інтеграція з базами даних:

– Підтримка популярних систем управління базами даних (наприклад, MySQL, PostgreSQL, MongoDB).

– Використання ORM інструментів (наприклад, Sequelize) для взаємодії з базами даних.

– Система повинна дозволяти автоматизовану обробку замовлень і платежів.

– Можливість адміністратору змінювати каталог продукції та відстежувати ціни та наявність на складі.

– Захист даних користувачів від несанкціонованого доступу.

– Автоматичне вирішення завдань може бути припинено у випадку технічних збоїв.

Сумісність програмного забезпечення - це забезпечення належного функціонування інтернет-магазину на різних платформах і пристроях з використанням новітніх технологій і процедур. Інформаційна сумісність - це правильний обмін, зберігання та обробка даних між різними системами для забезпечення точності, стандартизації та безпеки даних. Обидва аспекти важливі для успіху інтернет-магазинів у сучасному цифровому середовищі.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Веб-додаток призначений для підвищення ефективності електронної комерції. Основні функції та завдання програмного забезпечення наведені нижче.

Управління товарами та каталогами:

– Забезпечення можливості додавання, видалення та редагування товарів у каталозі.

– Систематизація товарів за різними критеріями, такими як ціна, бренд, технічні характеристики.

– Надання докладної інформації про кожен товар, включаючи фото, описи та технічні параметри.

Процес оформлення замовлення:

– Функція додавання товарів у кошик покупця.

– Організація процесу оформлення замовлення.

Користувацькі системи:

– Надання користувачам можливості відстежувати свої замовлення.

– Інформування користувачів про статус замовлень через повідомлення.

– Реєстрація та аутентифікація користувачів.

– Надання можливості перегляду та редагування особистих даних.

Управління замовленнями та запасами:

– Використання інструментів для моніторингу рівня запасів на складах.

– Надсилання повідомлень про низький рівень запасів для своєчасного поповнення.

– Відстеження кількості товарів на складі.

– Оновлення даних про запаси після кожного розміщеного замовлення.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці інтернет-магазину електронних пристроїв та аксесуарів математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Додаток побудований на основі трирівневої клієнт-серверної архітектури, що дозволяє нам ефективно розділити логіку між клієнтом і сервером.

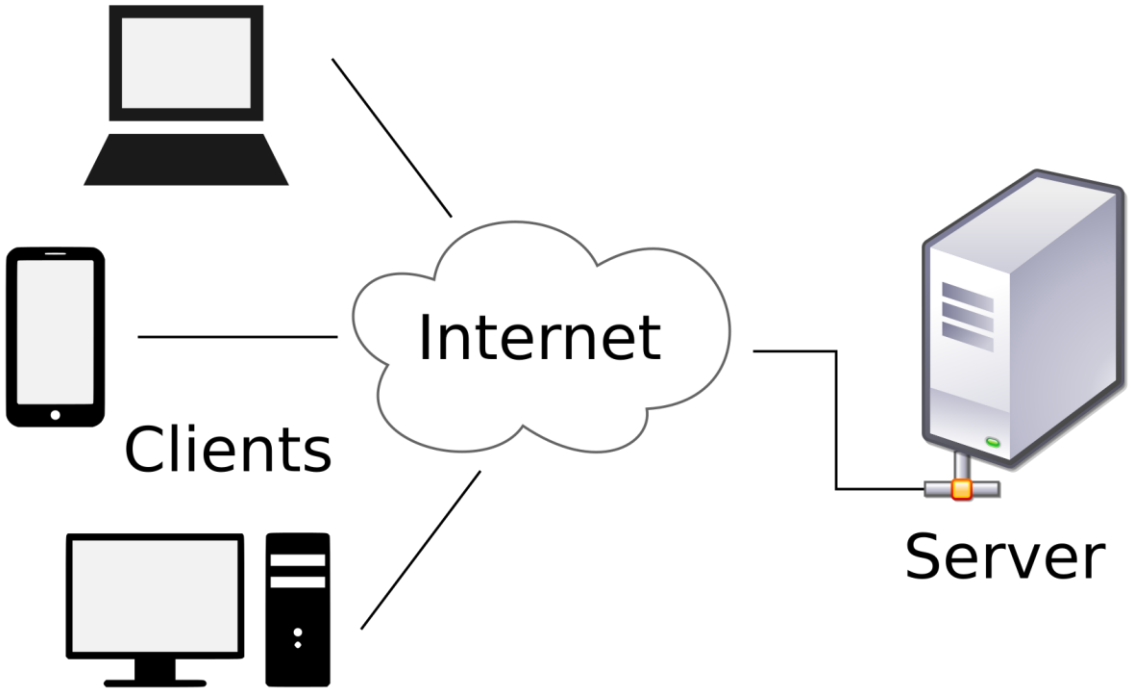

Рис. 2.1. Модель клієнт-серверної архітектури

Клієнт-серверна архітектура складається з двох основних елементів: клієнта і сервера. Клієнт - це комп'ютер користувача, який потребує інформації з сервера або надсилає інструкції серверу.

Сервер - це комп'ютер або пристрій, який є потужнішим за звичайний і може виконувати певні дії, такі як запуск програмних додатків, виконання

сервіс-орієнтованих завдань за запитом клієнта, полегшення доступу користувачів до певних ресурсів і зберігання інформації та баз даних. Сервер. Модель системи полягає в тому, що клієнт надсилає запит на сервер, сервер обробляє запит і надсилає кінцевий результат клієнту. Сервер може обслуговувати декілька клієнтів одночасно. При одночасному надходженні декількох запитів сервер ставить їх в чергу і обробляє по черзі.

Запити можуть мати пріоритети, і може існувати класифікація, за якою запит з найвищим пріоритетом виконується в першу чергу. Модель такої системи полягає в тому, що клієнт надсилає запит на сервер, де він обробляється і готовий результат надсилається клієнту. Сервер може обслуговувати декілька клієнтів одночасно. При одночасному надходженні декількох запитів вони ставляться в чергу і обробляються сервером один за одним. У деяких випадках запити можуть мати пріоритет, і запити з вищим пріоритетом повинні бути виконані раніше.

Функції, які реалізуються на сервері:

– зберігання, доступ, захист і резервне копіювання даних;
– обробка клієнтського запиту;
– відправлення результату (відповіді) клієнту;

Функції, які реалізуються на стороні клієнта:

– надання користувальницького інтерфейсу;
– формулювання запиту до сервера і його відправка;
– отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних) .

2.4. Опис використаних технологій та мов програмування

Серед різних мов програмування, доступних в індустрії веб-розробки, JavaScript є дуже відомою і популярною мовою. Вона широко використовується для створення динамічних веб-сайтів і веб-додатків. Розглянемо деякі переваги використання JavaScript для розробки інтернет-магазину. Клієнтська та серверна розробка: JavaScript можна використовувати як на стороні клієнта, так і на

стороні сервера. Щоб використовувати JavaScript на стороні клієнта, ви створюєте інтерактивний користувальницький інтерфейс і спілкуєтеся з сервером через AJAX, Node.js дозволяє створювати швидкі і гнучкі серверні додатки з використанням JavaScript. Асинхронний: однією з найбільш корисних особливостей JavaScript є можливість виконувати операції вводу/виводу без переривання основного потоку, тобто в асинхронному режимі [3].

Node. js - це серверне середовище виконання JavaScript, яке працює на механізму V8 Google Chrome. Деякі з переваг, доступних при розробці інтернет-магазину, включають: node.js використовує неблокуючу модель вводу/виводу, що відповідає за його швидкість і ефективність. Під час виконання серверні додатки можуть бути спроектовані швидкими та ресурсоефективними. Він забезпечує високу пропускну здатність і не страждає від великої кількості паралельних з'єднань. Вона також надає широкий спектр бібліотек і фреймворків, які полегшують розробку серверних додатків. Наприклад, Express.js є найкращим фреймворком для розробки веб-серверів на основі Node.js. Як єдину мову, JavaScript можна використовувати як мову сценаріїв на стороні сервера, так і на стороні клієнта, що дозволяє розробникам створювати інтегровані додатки, використовуючи єдину мову програмування. Для нашого проекту ми обрали JavaScript та Node.js. Тому що обидві мови мають широкий спектр можливостей, дуже ефективні та швидкі у розробці веб-додатків, дуже популярні та мають активну спільноту розробників. Вони надають чудовий ресурс, який дозволяє створити сучасний інтернет-магазин, який добре працює, гарно виглядає і водночас є зручним для користувача [4].

Технології на стороні клієнта.

React.js - популярна JavaScript-бібліотека для створення користувацьких інтерфейсів, відома своєю компонентною архітектурою, що сприяє повторному використанню коду та його підтримці React спрощує процес створення інтерактивних інтерфейсів, розбиваючи додатки на менші, придатні для багаторазового використання компоненти [5].

Redux: контейнер для прогнозування стану застосунку на JavaScript, що використовується з React для управління станом застосунку Redux надає єдине джерело істини про стан застосунку, що полегшує управління та усунення несправностей у складних потоках даних Полегшує. [6].

React Router DOM: бібліотека маршрутизації для React-додатків, яка підтримує декларативну маршрутизацію. Це дозволяє розробникам визначати динамічні маршрути у своїх додатках та керувати навігацією, що полегшує створення односторінкових додатків з декількома відображеннями.

Axios: HTTP-бібліотека для асинхронних запитів у JavaScript-додатках. Axios спрощує надсилання HTTP-запитів та обробку відповідей завдяки таким функціям, як захоплення запитів і відповідей, автоматичний розбір JSON та обробка помилок [7].

Formik: бібліотека для створення форм у React-додатках, що надає простий та інтуїтивно зрозумілий API для керування статусом, валідацією та відправкою форм Formik обробляє логіку, пов'язану з формами, та надає інструменти для валідації та обробки помилок, таким чином спрощуючи процес створення складних форм [8].

Yup: JavaScript-бібліотека валідації схем, яка використовується з Formik для визначення та валідації схем форм. Yup дозволяє розробникам створювати міцні правила валідації для полів форми, забезпечуючи цілісність та однорідність даних [9].

MobX: JavaScript бібліотека для управління станом програми, особливо підходить для управління складним станом програми з мінімальним кодом MobX використовує спостережувані структури даних і принципи реактивного програмування для того, щоб автоматично оновлює інтерфейс користувача [10].

MobX React Lite: легковажна бібліотека пов'язок React для MobX, яка пропонує оптимізовану інтеграцію з функціональними компонентами React. MobX React Lite спрощує процес підключення компонентів React до сховищ MobX, забезпечуючи ефективне управління станом у React-додатках.

Swiper: сучасна бібліотека слайдерів для створення відгуків, слайдерів і галерей відгуків у веб-додатках. Swiper підтримує жести сенсорного екрану, клавіатурну навігацію і плавні переходи, бездоганний користувацький досвід на всіх пристроях і з будь-яким розміром екрану.

JWT Decode: бібліотека для декодування JSON Web Tokens (JWT) в JavaScript-додатках. JWT Decode дозволяє розробникам витягувати та отримувати доступ до даних, закодованих у JWT, забезпечуючи аутентифікацію та авторизацію у клієнтському коді.

Технології на стороні сервера.

Express. js: мінімалістичний веб-фреймворк для Node. js, який надає інструменти для обробки HTTP-запитів, маршрутизації, інтеграції проміжного програмного забезпечення та рендерингу шаблонів. RESTful API і часто використовується для розробки веб-серверів та API [11].

Sequelize - ORM-бібліотека для Node.js, призначена для взаємодії з реляційними базами даних за допомогою JavaScript. Sequelize надає потужний набір функцій для визначення моделей, виконання запитів, управління транзакціями баз даних, підвищення продуктивності та збереження даних [12].

PostgreSQL: могутня відкрита система управління реляційними базами даних (RDBMS), відома своєю надійністю, масштабованістю та рядом розширених функцій. PostgreSQL пропонує підтримку ACID-транзакцій, типи даних JSON та розширений індексування, що робить його підходящим для даних, інтенсивних за обсягом, застосунків, таких як інтернет-магазини [13].

JWT (JSON Web Tokens): стандарт для безпечної передачі інформації між сторонами у вигляді JSON-об'єктів, часто використовується для реалізації механізмів аутентифікації та авторизації у веб-додатках. JWT є компактними, URL-безпечними токенами, які можна легко генерувати, перевіряти та декодувати, що дозволяє стане-менше аутентифікацію у розподілених системах [14].

bcrypt: криптографічна хеш-функція, яка зазвичай використовується для безпечного зберігання паролів у базі даних. bcrypt використовує адаптивні хеш-

функції та методи травлення для захисту паролів від атак грубої сили та райдужних таблиць, а також для захисту облікових записів користувачів у веб-додатках від Безпека [15].

Cross-Origin Resource Sharing (CORS): механізм, який дозволяє веб-серверам вказувати джерела, з яких можна отримати доступ до ресурсів на сервері. CORS захищає веб-додатки, обмежуючи запити між різними доменами і є Це важливо для запобігання несанкціонованому доступу до конфіденційних даних.

Express Fileupload: проміжний програмний засіб для обробки завантаження файлів у додатках на основі Express.js, який надає засоби для розбору багатокомпонентних запитів та зберігання завантажених файлів на сервері. Express Fileupload спрощує процес обробки завантаження файлів та інтегрується з веб-сервером Express.js.

Dotenv: модуль для завантаження змінних середовища з файлу .env у об'єкт process.env у додатках Node.js. Dotenv дозволяє розробникам налаштовувати параметри додатка, такі як облікові дані бази даних та ключі API, без жорсткого кодування їх у вихідний код, покращуючи безпеку та переносимість [16].

2.5. Опис структури програми та алгоритмів її функціонування

Система інтернет-магазину електроніки “Dom1store” включає наступні компоненти:

Шапка сайту та “футер”:

– Header.js;
– Footer.js.

Головна сторінка:

– Landing.js;
– Loader.js;
– Advertising.js;
– Privilege.js;

– SwiperActual.js;
– Catalog.js.

Сторінка списку товарів:

– ProdList.js;
– ProdListSerch.js;
– Sort.js;
– Reng.js;
– ProductItem.js.

Сторінка кошику, та вподабаних товарів:

– Basket.js;
– Like.js.

Сторінка реєстрації, авторизації та особистий кабінет:

– Auth.js;
– Register.js;
– PersonalArea.js;
– Desired.js;
– Orders.js;
– Profile.js;
– Sms.js;
– Stock.js.

Сторінка адміністування:

– AdminAuth.js;
– Admin.js;
– AllOrders.js;
– Customers.js;
– ProductAdding.js.

Сторінка 404, модальні вікна, заказ товару:

– NotFound.js;
– Modal.js;
– OrderModal.js;

– Order.js.

За зв'язок с сервером відповідають такі класи як:

– deviceAPI.js;

– orderAPI.js;

– userAPI.js;

– DeviceStore.js;

– OrderStore.js;

– UserStore.js.

Логічна структура програми:

– Express.js та маршрутизатори:

Express.js дозволяє легко створювати бекенд-додатки та керувати HTTP-запитами. Маршрутизатор визначає поведінку програми щодо певних URL-адрес і HTTP-методів. Прикладом такого маршрутизатора є отримання списку користувачів. Можна використовувати такі URL-адреси, як /users, а потім метод GET. Контролер діє як точка входу і відповідає за отримання даних з HTTP-запитів, запуск процесів бази даних і створення відповідей користувачам. У деяких сценаріях контролер, відповідальний за отримання інформації про користувачів, надсилає запит до бази даних, отримує інформацію про кожного користувача зі списку і надає її клієнту у відповіді.

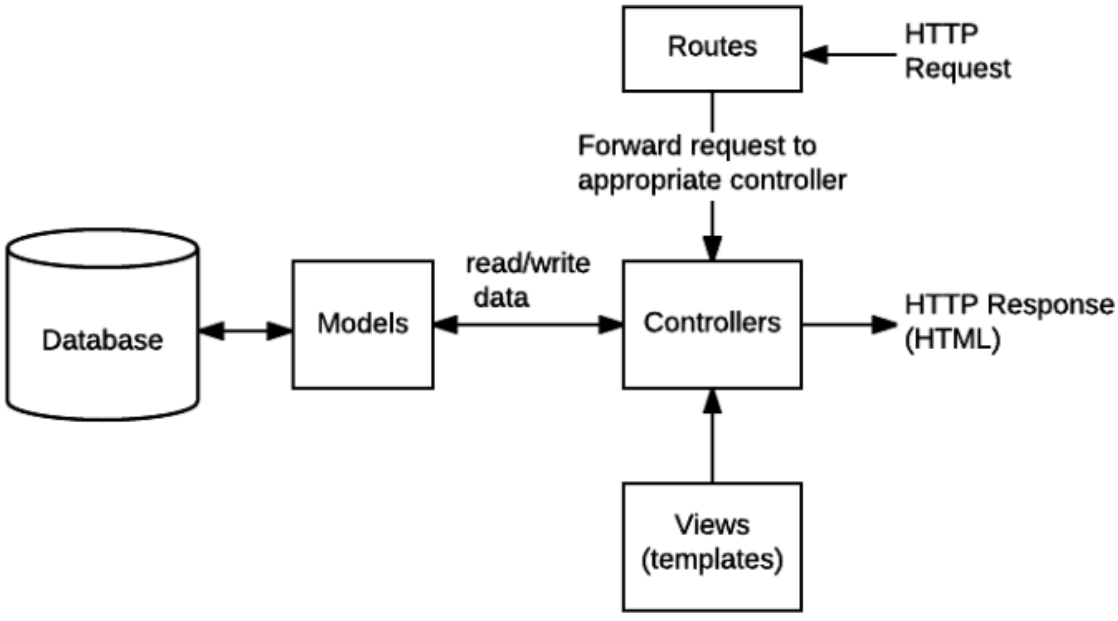

Рис. 2.2. Маршрутизатори у Express.js

– Моделі бази даних Sequelize:

Моделі даних визначають організацію та зв’язки різних об’єктів у базі даних. Наприклад, особа може бути представлена моделлю користувача, яка містить такі поля, як ім’я, адреса електронної пошти та пароль. Sequelize ORM забезпечує зручний спосіб роботи з базою даних за допомогою виразів JavaScript.

– Переваги використання Sequelize ORM:

Простота у використанні: Sequelize ORM дозволяє взаємодіяти з базами даних за допомогою мови JavaScript, що робить процес розробки більш зручним і прозорим. Підтримка широкого спектру баз даних: Sequelize підтримує SQL бази даних, такі як PostgreSQL, MySQL, SQLite і MSSQL, а також NoSQL бази даних, тому ви можете вибрати найкращу базу даних для вашого проекту. Міграція: Sequelize спрощує управління структурою бази даних і надає механізм міграції для легкого оновлення схем баз даних на різних етапах розробки. Асоціації: Sequelize надає зручний спосіб визначення асоціацій між моделями, таких як один-до-одного, один-до-багатьох і багато-до-багатьох. Це спрощує маніпуляції з даними та полегшує виконання складних запитів.Підтримка обіцянок та асинхронізації/очікування: Sequelize підтримує обіцянки та асинхронізацію/очікування, що дозволяє писати чистіший, більш читабельний код без необхідності глибокого вбудовування зворотних викликів. Це дозволяє писати чистіший, більш читабельний код без необхідності глибокого вбудовування зворотних викликів. Перевірка: Sequelize надає можливість перевіряти дані перед тим, як вони будуть збережені в базі даних, щоб уникнути некоректних даних. Підтримка транзакцій: Sequelize дозволяє виконувати операції з базою даних в рамках транзакції, що робить маніпуляції з даними більш надійними [18].

– Контролери:

Логіка обробки запиту закладена в контролері, який отримує дані з HTTP-запиту, виконує необхідні операції з базою даних і відправляє відповідь клієнту. Контролер, який отримує список користувачів з бази даних - це контролер, який

може відправити запит до бази даних, отримати відповідь зі списком користувачів і відправити його назад клієнту.

Алгоритм та функціонування програми:

– Підключення до бази даних та створення моделей:

Алгоритм починається з підключення до бази даних за допомогою Sequelize і пошуку необхідних моделей. Моделі розроблено на основі схеми бази даних, а також зв’язків між таблицями.

– Обробка HTTP-запитів:

HTTP-запит, і сервер отримає маршрут від URL-адреси до метода. Після цього викликається правильний контролер, який отримує дані з HTTP-запиту та передає їх до бази дани

– Взаємодія з базою даних:

Операції з базою даних, такі як створення, читання, оновлення та видалення даних, є відповідальністю контролерів. Відповідно до викликів методів моделювання Sequelize також генеруватиме SQL-запити.

Опис структури бази даних:

Таблиця 2.1

Таблиця "User"

Поле	Тип даних	Опис
Id	INTEGER	Унікальний ідентифікатор користувача
email	STRING	Електронна адреса, унікальна
password	STRING	Пароль
name	STRING	Ім'я
surname	STRING	Прізвище
phoneNumber	STRING	Номер телефону
patronymic	STRING	По батькові
date	STRING	Дата народження
gender	STRING	Стать

Продовж. табл. 2.1

extension	STRING	Додаткова інформація
address	STRING	Адреса
role	STRING	Роль у системі, за замовчуванням "USER"

Таблиця 2.2

Таблиця "Order"

Поле	Тип даних	Опис
id	INTEGER	Унікальний ідентифікатор замовлення, первинний ключ, автоінкремент
orderEmail	STRING	Електронна адреса замовника
phoneNumber	STRING	Номер телефону замовника
delivery	STRING	Спосіб доставки
name	STRING	Ім'я замовника
surname	STRING	Прізвище замовника
payment	STRING	Спосіб оплати
cost	INTEGER	Вартість замовлення

Таблиця 2.3

Таблиця "OrderDevices"

Поле	Тип даних	Опис
id	INTEGER	Унікальний ідентифікатор товару, первинний ключ, автоінкремент
name	STRING	Назва товару
category	STRING	Категорія товару
count	STRING	Кількість одиниць товару у замовленні

Таблиця 2.4

Таблиця "Device"

Поле	Тип даних	Опис
id	INTEGER	Унікальний ідентифікатор товару, первинний ключ, автоінкремент
name	STRING	Назва товару, унікальна, обов'язкова
price	STRING	Ціна товару, обов'язкова
rating	INTEGER	Рейтинг товару, за замовчуванням 0
img	STRING	Зображення товару, обов'язкове
img2	STRING	Додаткове зображення товару
img3	STRING	Додаткове зображення товару
img4	STRING	Додаткове зображення товару
discount	INTEGER	Знижка
category	STRING	Категорія товару
relevance	STRING	Актуальність товару
img	STRING	Зображення товару, обов'язкове

Таблиця 2.5

Таблиця "Type"

Поле	Тип даних	Опис
id	INTEGER	Унікальний ідентифікатор типу, первинний ключ, автоінкремент
name	STRING	Назва типу, унікальна, обов'язкова
link	STRING	Посилання, унікальне, обов'язкове

Таблиця 2.6

Таблиця "DeviceInfo"

Поле	Тип даних	Опис
id	INTEGER	Унікальний ідентифікатор додаткової інформації, первинний ключ, автоінкремент
title	STRING	Заголовок
description	STRING	Опис

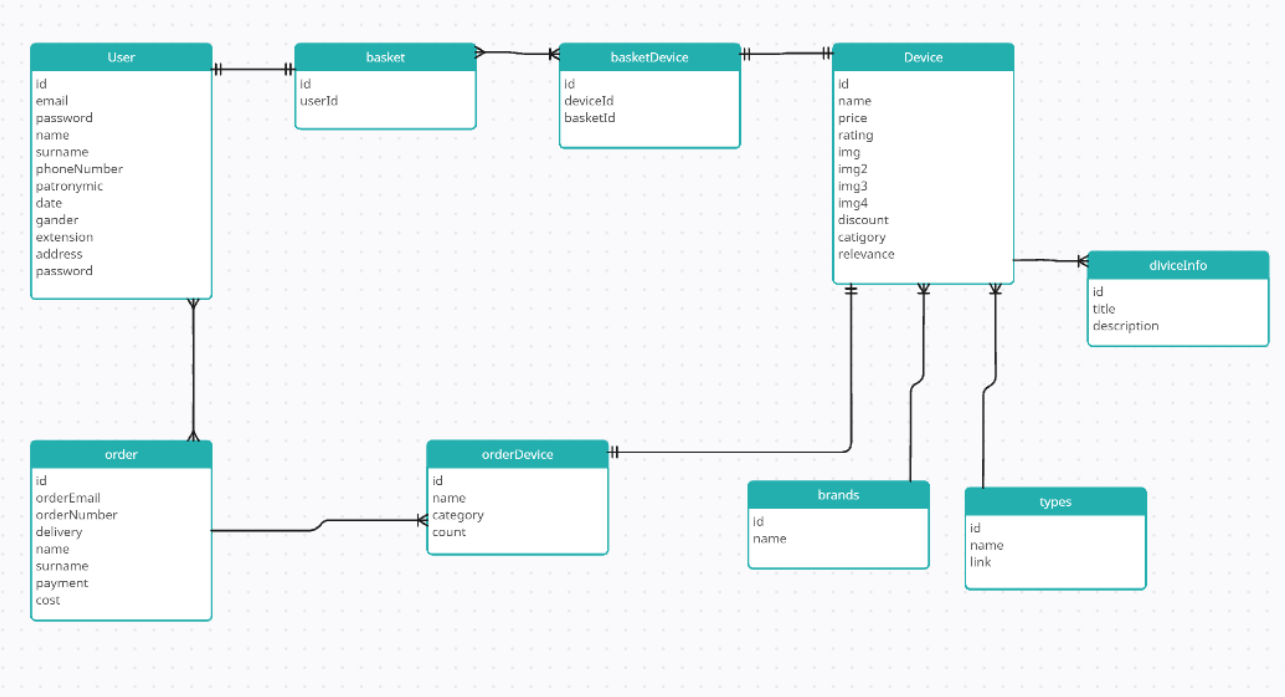

Рис. 2.3. Діаграма бази данних Dom1store

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Характер, організація і попередня підготовка вхідних даних.

Користувацькі дані: Вхідні дані у формі замовлень, реєстрації користувачів, оновлення профілю що надходять через фронтенд за допомогою Formik, які збирають інформацію та відправляють її на сервер за допомогою HTTP запитів з використанням Axios. Попередня валідація здійснюється за допомогою Yup.

Дані каталогу товарів: вони отримуються з бази даних через API, що надається сервером за допомогою Sequelize та Express. Ці дані вже структуровані і готові до відображення на фронтенді.

Авторизація та аутентифікація: вхідні дані для аутентифікації користувача передаються у вигляді токенів, які або вводяться через форму, або розшифровуються за допомогою бібліотеки jwt-decode.

Характер і організація вихідних даних:

Відповіді на запити: сервер надсилає вихідні дані у форматі JSON через API за допомогою Express. Ці дані містять інформацію про користувачів (якщо вони авторизовані), товари, категорії тощо.

Сповіщення та підтвердження: у випадку успішної реєстрації, оформлення замовлення або інших подій, відображає повідомлення на фронтенді.

Формат, опис і спосіб кодування вхідних та вихідних даних:

Формат даних: для вхідних даних використовується формат JSON або форма введення на стороні клієнта, яка передає дані у форматі JSON за допомогою HTTP-запиту.

Опис даних: дані описані у вигляді схеми (наприклад, використовуючи Sequelize моделі для бази даних), або типів даних).

Спосіб кодування: для передачі даних через мережу використовується кодування у форматі UTF-8 для забезпечення сумісності та надійності передачі.

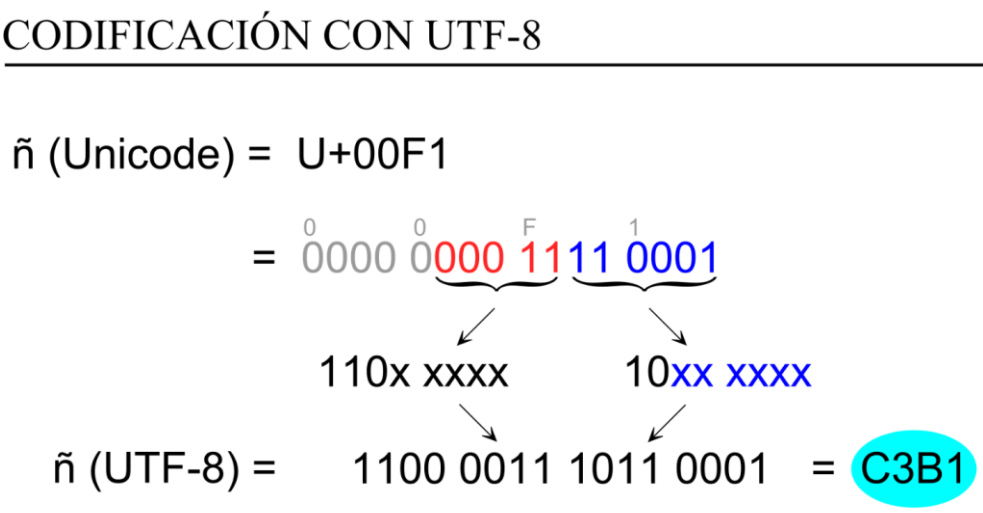

Рис. 2.4. Принцип utf-8 кодування

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Під час розробки інтернет-магазину були активно використані передові технології та інструменти.

Для контролю версій і управління кодом було обрано Git. Це дає змогу ефективно відстежувати зміни та забезпечувати цілісність кодової бази.

В якості основної мови програмування для розробки на стороні сервера було використано JavaScript та Node.js, що гарантує високу продуктивність та масштабованість серверного коду, дозволяючи йому ефективно справлятися з високими навантаженнями в магазині.

Для створення інтерфейсу клієнтської частини навантаження було обрано фреймворк React, який пропонує високу продуктивність та легке оновлення користувацького інтерфейсу, а також надає багато готових компонентів та бібліотек для швидкої розробки.

В якості бази даних - PostgreSQL, надійна і потужна реляційна база даних, яка забезпечує надійне зберігання і обробку даних і гарантує відповідність властивостям ACID, забезпечує цілісність і надійність даних в пам'яті.

Figma - для розробки та прототипування макетів користувацького інтерфейсу. Вона допомагає створювати привабливі та функціональні макети для покращення користувацького досвіду, а також ви можете створювати спільні макети, щоб отримати зворотній зв'язок від вашої команди та покращити дизайн поверхні.

Також було використано розгалуження і злиття в Git для ефективної співпраці та управління змінами в коді.

2.7.2. Використані програмні засоби

Операційна система, на якій базується проект - Windows.

Операційна система Windows має кілька ключових переваг, які сприяють успіху системи: Windows є однією з найпоширеніших операційних систем у світі. Це означає, що більшість користувачів вже знайомі з цією платформою, що спрощує впровадження та використання наших систем. Різноманітність програмного забезпечення: Windows має широкий спектр програмного забезпечення, що підтримується на цій платформі. Це означає, що для розробки, тестування та підтримки проектів можна використовувати широкий спектр інструментів і додатків. Інтерфейс: Windows відома своїм інтуїтивно зрозумілим і зручним інтерфейсом. Це робить наші системи більш доступними та зручними для користувачів, сприяючи їх популярності та визнанню громадськістю. Сумісність з апаратним забезпеченням: Windows підтримує широкий спектр апаратного забезпечення, що дозволяє працювати з різноманітними пристроями та конфігураціями без суттєвих обмежень. Підтримка: корпорація Майкрософт забезпечує активну підтримку та регулярні оновлення операційної системи, підтримуючи її в актуальному та безпечному стані. Загалом, операційна система Windows надає інструменти та середовище, необхідні для успішної розробки, впровадження та підтримки проектів, забезпечуючи надійність, доступність та простоту використання [19].

Для написання коду та розробки функціональності проекту використовуються різні програмні інструменти, такі як Visual Studio Code. Цей текстовий редактор є одним з найпопулярніших і найпотужніших текстових редакторів для розробки програмного забезпечення. Він має кілька важливих переваг, які роблять його ідеальним вибором для написання коду та розробки функціональності проекту Легкий і швидкий: Visual Studio Code - це легкий текстовий редактор, що означає, що він запускається миттєво і має мінімальний вплив на продуктивність системи. Він не обтяжує вашу систему зайвими ресурсами, тому ви можете швидко розпочати роботу над своїм проектом. Розширюваність: VS Code має широкий спектр розширень, які дозволяють розширити його функціональність відповідно до потреб вашого проекту. Інтеграція з Git: VS Code має вбудовану підтримку системи контролю версій Git.

Це полегшує співпрацю та керування кодовою базою вашого проекту, оскільки ви можете переносити коди безпосередньо з редактора, переглядати перенесені файли та керувати гілками. Підтримка різних мов програмування: Visual Studio Code підтримує JavaScript, Python, Java, C# та багато інших мов програмування і фреймворків. Це робить його універсальним інструментом розробки, незалежно від мови програмування, яка використовується у вашому проекті. Активна спільнота та підтримка: VS Code має велику спільноту розробників і користується активною підтримкою від Microsoft. Це означає, що ви можете легко знайти відповіді на свої запитання та отримати поради та підказки щодо роботи з редактором [20].

Інструмент Postman використовується для тестування API і взаємодії з сервером Postman - це інструмент тестування API, який дозволяє розробникам створювати, тестувати і налагоджувати HTTP-запити і відповіді. Деякі важливі причини для використання Postman: Запит Створення: Postman надає зручний інтерфейс для створення різних типів HTTP-запитів, таких як GET, POST, PUT і DELETE. Це дозволяє швидко і легко тестувати різні кінцеві точки API. Конфігурація параметрів: інструмент дозволяє налаштовувати різні параметри запиту, такі як заголовки, тіло запиту і параметри запиту. Це дозволяє запускати різні тестові сценарії і тестувати різні взаємодії API. Організація колекцій: Postman дозволяє організувати велику кількість запитів у колекції, що полегшує керування та навігацію між колекціями. Це особливо корисно, коли потрібно протестувати різні частини або функції API [21].

Управління версіями коду здійснюється за допомогою системи управління версіями Nginx Nginx — це високоефективний веб-сервер, що зазвичай використовується як проксі-сервер, а також для збалансування навантаження, обробки запитів HTTPS, обслуговування веб-сайтів з високим навантаженням, а також для доставки контенту через протоколи HTTP, HTTPS, SMTP, POP3 і IMAP.

Основні переваги Nginx:

– Висока продуктивність: розроблений з думкою про високу продуктивність, Nginx оптимізований для великої кількості одночасних з'єднань. Його архітектура розроблена таким чином, щоб ефективно обробляти тисячі одночасних запитів. Низьке споживання ресурсів: використання пам'яті і центрального процесора у Nginx дуже ефективне, що робить його ідеальним вибором для серверів з обмеженими ресурсами.

– Висока масштабованість: Nginx може легко масштабуватися горизонтально за допомогою різних методів, включаючи балансування навантаження на сервер, розподіл навантаження і кешування.

– Проста конфігурація: конфігурація Nginx здійснюється за допомогою текстових файлів, що робить процес конфігурування веб-сервера досить простим і зрозумілим.

– Проксі-сервери і балансувальники навантаження: Nginx часто використовується як проксі-сервер, який може перенаправляти запити до різних сервісів, віддалених серверів або додатків. Він також може розподіляти навантаження між різними серверами, підвищуючи надійність і ефективність інфраструктури [22].

2.7.3. Виклик та завантаження програми

Ось як викликати та завантажити програму інтернет-магазину Dom1store:

– Щоб розпочати роботу з інтернет-магазином, спочатку потрібно клонувати репозиторій з відповідного носія даних. Для цього використовуйте команду: git clone [URL репозиторію]

– Переконайтеся, що встановлено Git на вашому пристрої перед виконанням цієї команди.

– Після завершення клонування перейдіть у каталог, де ви хочете мати копію репозиторію. Це можна зробити, відкривши термінал та використовуючи команду: cd [назва папки проекту] Тепер, для завантаження необхідних залежностей для React, використовуйте команду: npm install

– Після завершення встановлення залежностей перейдіть до папки "client" за допомогою команди: cd client .Використайте команду "npm install" ще раз, щоб встановити залежності для клієнтської частини проекту. npm install Після завершення цього кроку перейдіть до кореневої папки проекту за допомогою команди: cd ..

– Далі, перейдіть до папки "server" за допомогою команди: cd server Використовуйте команду "npm install" для встановлення залежностей для серверної частини проекту. Перед цим переконайтеся, що Node.js встановлено на вашому пристрої. npm install Після завершення встановлення залежностей, перейдіть до кореневої папки проекту . Тепер, щоб запустити проект, переконайтеся, що ви знаходитесь в кореневій папці (server). Запустіть сервер, використовуючи команду: npm run dev

– Після успішного запуску сервера відкрийте нове вікно терміналу. Перейдіть до папки "client" за допомогою команди: cd client. Запустіть клієнт, використовуючи команду: npm start. Якщо все пройшло успішно, програма буде доступна за адресою "http://localhost:3000/".

– Перевірте консоль, щоб переконатися, що немає помилок.

2.7.4. Опис інтерфейсу користувача

Головна сторінка наведена на рис.2.4. та містить важливу інформацію про акції, новинки та найпопулярніші товари. На головній сторінці є слайдер нових товарів, де користувачі можуть побачити останні додані товари. Також є слайдер "Популярні товари", який дозволяє користувачам швидко ознайомитися з найбільш популярними та затребуваними товарами. Посилання на основні категорії товарів, акції та рекламні банери дозволяють користувачам легко переходити на відповідні сторінки.

Управління:

– Слайдер новинок: клікабельні об'єкти, які дозволяють переглядати нові товари.

– Свайпер популярних товарів: клікабельні об'єкти, що дозволяють швидко ознайомитися з популярними товарами.

– Головні категорії товарів: клікабельні посилання для переходу до відповідних категорій товарів.

– Акції: клікабельні об'єкти для перегляду акцій та спеціальних пропозицій.

– Рекламні банери: клікабельні посилання, які привертають увагу користувача та надають доступ до різних акцій та пропозицій.

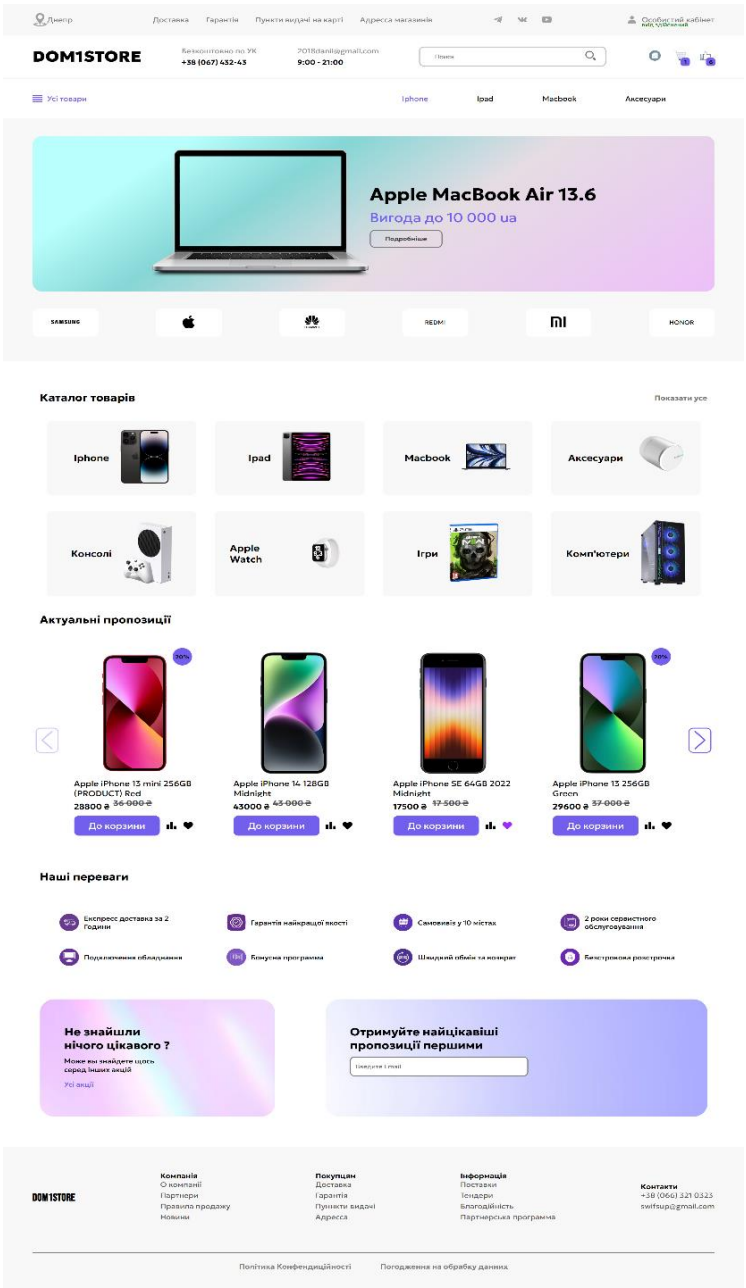

Рис. 2.4. Головна сторінка

Категорії товарів:

Опис: список категорій товарів, що дозволяє користувачам швидко перейти до потрібної категорії товарів.

Управління:

– Категорії товарів: клікабельні посилання для перегляду підкатегорій та товарів в цій категорії.

– Підкатегорії: при кліці на кожну категорію відображаються підкатегорії.

– Товари: при кліці на кожну підкатегорію відображаються товари в цій категорії.

Ілюстрація:

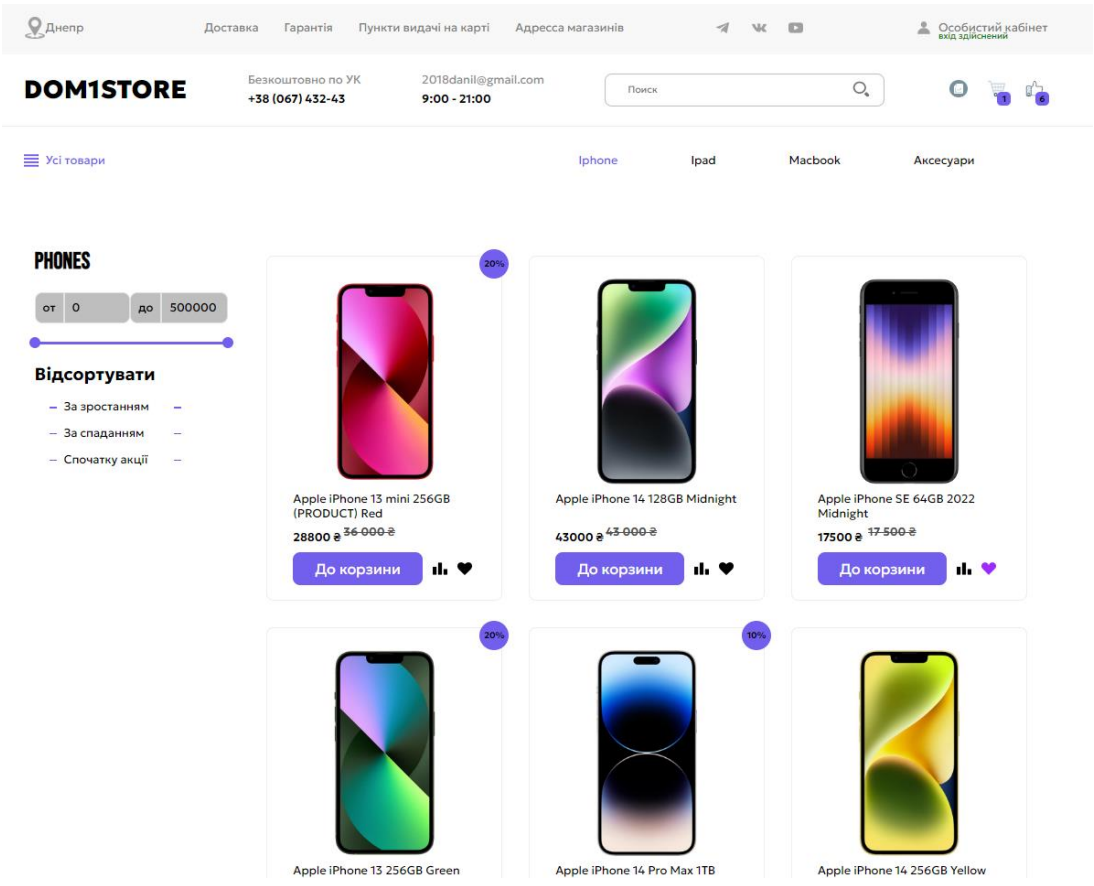

Рис. 2.5. Список категорій

Пошук:

Опис: можливість здійснення пошуку товарів за назвою, категорією тощо.

Управління:

– введення тексту для пошуку та вибір параметрів пошуку.

Ілюстрація:

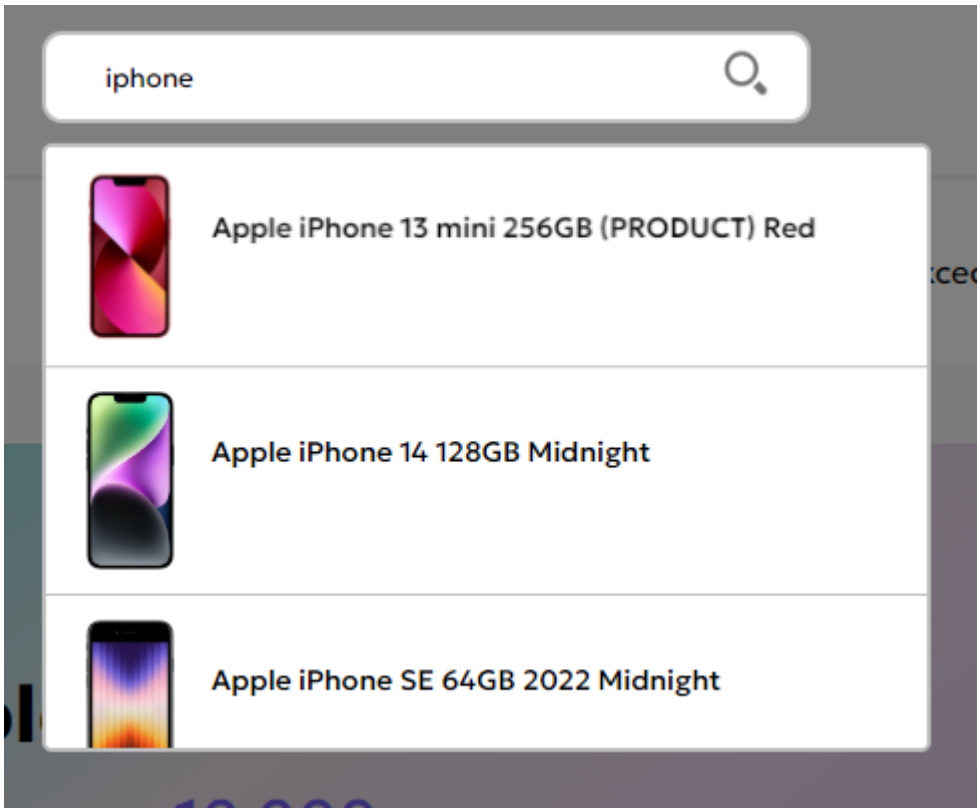

Рис. 2.6. Управління пошуком

Кошик.

Опис: перегляд товарів, які додані до кошика, обчислення загальної вартості та оформлення замовлення.

Управління:

– Додавання товарів: кнопка "Додати до кошика" для додавання товару в кошик.

– Видалення товарів: можливість видалення товарів з кошика.

– Зміна кількості товарів: можливість змінити кількість одиниць товару для замовлення.

Ілюстрація:

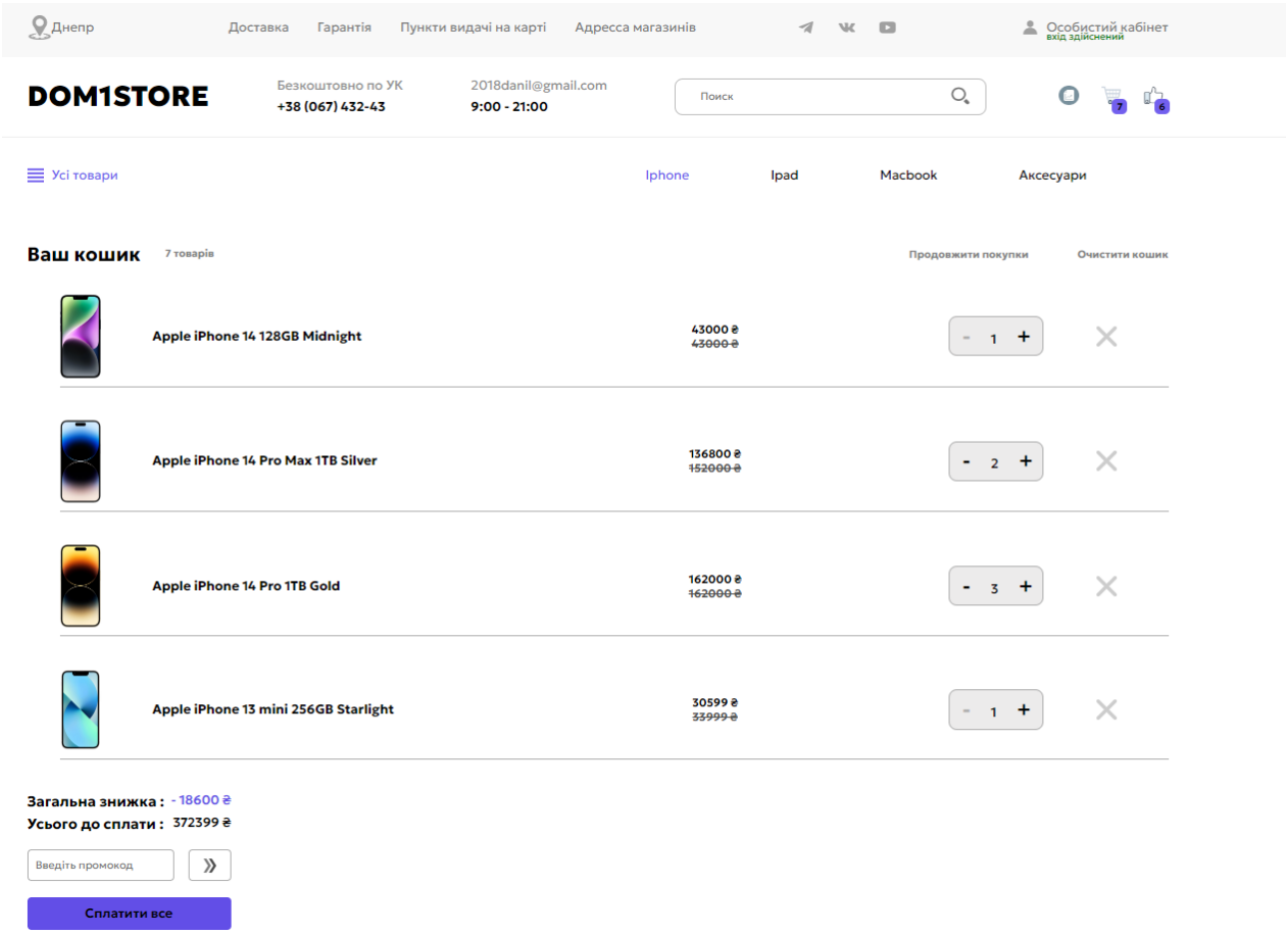

Рис. 2.7. Кошик

Особистий кабінет.

Опис: сторінка для авторизованих користувачів, де вони можуть переглядати свої замовлення, редагувати профіль тощо.

Управління:

– Перегляд особистих даних: користувач може переглядати інформацію, яка стосується його особистих даних, таку як ім'я, адреса електронної пошти, контактний номер телефону тощо.

– Редагування профілю: користувач може змінювати свої особисті дані, такі як ім'я, адреса електронної пошти, контактний номер телефону тощо.

– Зміна адреси доставки: користувач може змінювати адресу доставки для майбутніх замовлень.

– Перегляд історії замовлень: користувач може переглядати історію своїх замовлень, включаючи статус доставки та іншу важливу інформацію.

– Перегляд доставлених товарів: Користувач може переглядати і підтверджувати доставку отриманих товарів.

Ілюстрація:

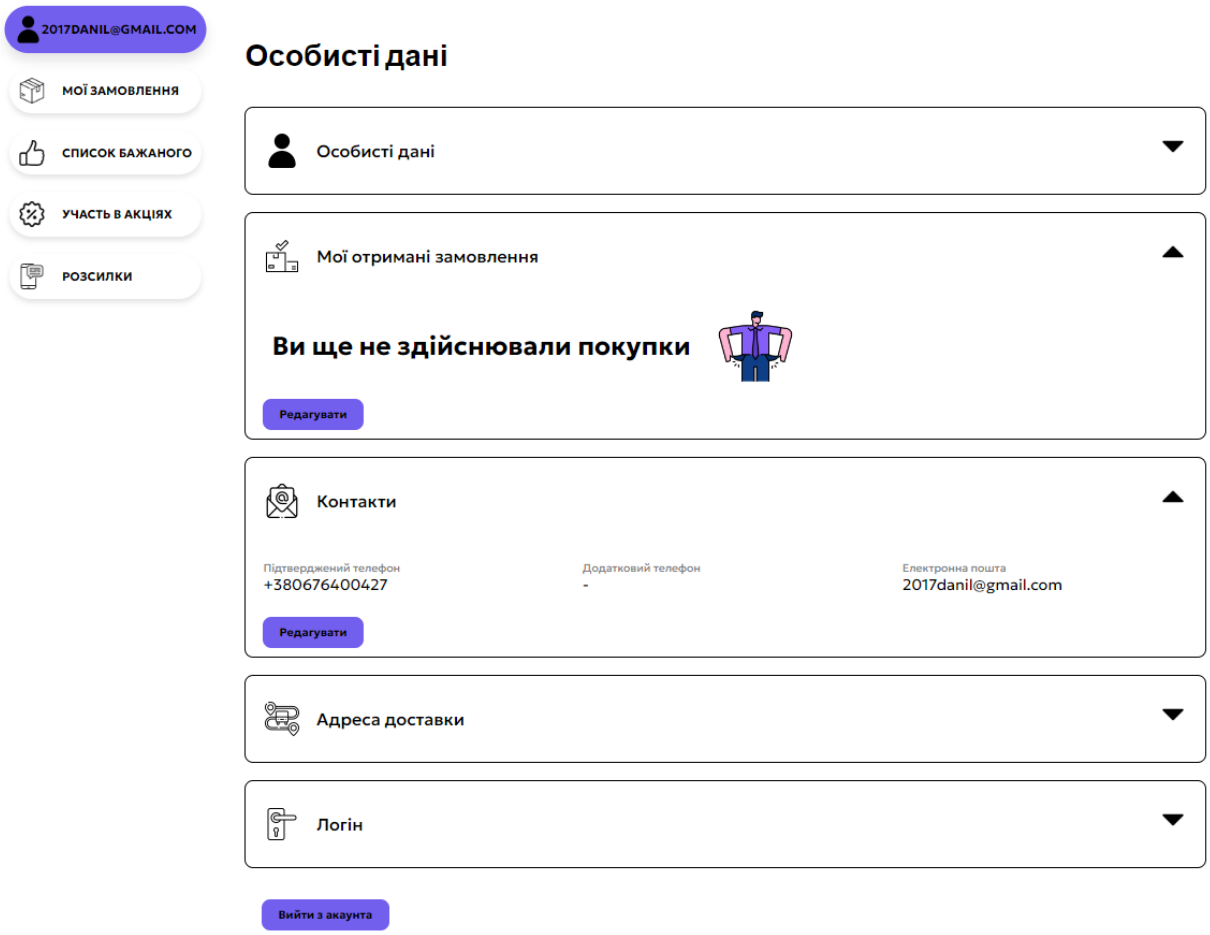

Рис. 2.8. Особистий кабінет

Оформлення замовлення.

Опис: сторінка, де користувачі можуть переглядати свій кошик, вводити дані для доставки та оплати, а також підтверджувати своє замовлення.

Управління:

– Перегляд кошика: користувач може переглядати список товарів у своєму кошику, включаючи назви, кількість, ціну кожного товару та загальну вартість замовлення.

– Введення даних для доставки: користувач може ввести або вибрати свою адресу доставки для поточного замовлення.

– Введення платіжних даних: користувач може ввести свої платіжні дані, такі як інформація про кредитну або дебетову карту, або вибрати інші способи оплати.

– Вибір способу доставки: користувач може вибрати бажаний спосіб доставки, наприклад, стандартна доставка, експрес-доставка тощо.

– Підтвердження замовлення: користувач може переглянути всі деталі замовлення, включаючи товари, адресу доставки, спосіб оплати та доставку, а також підтвердити своє замовлення.

Ілюстрація:

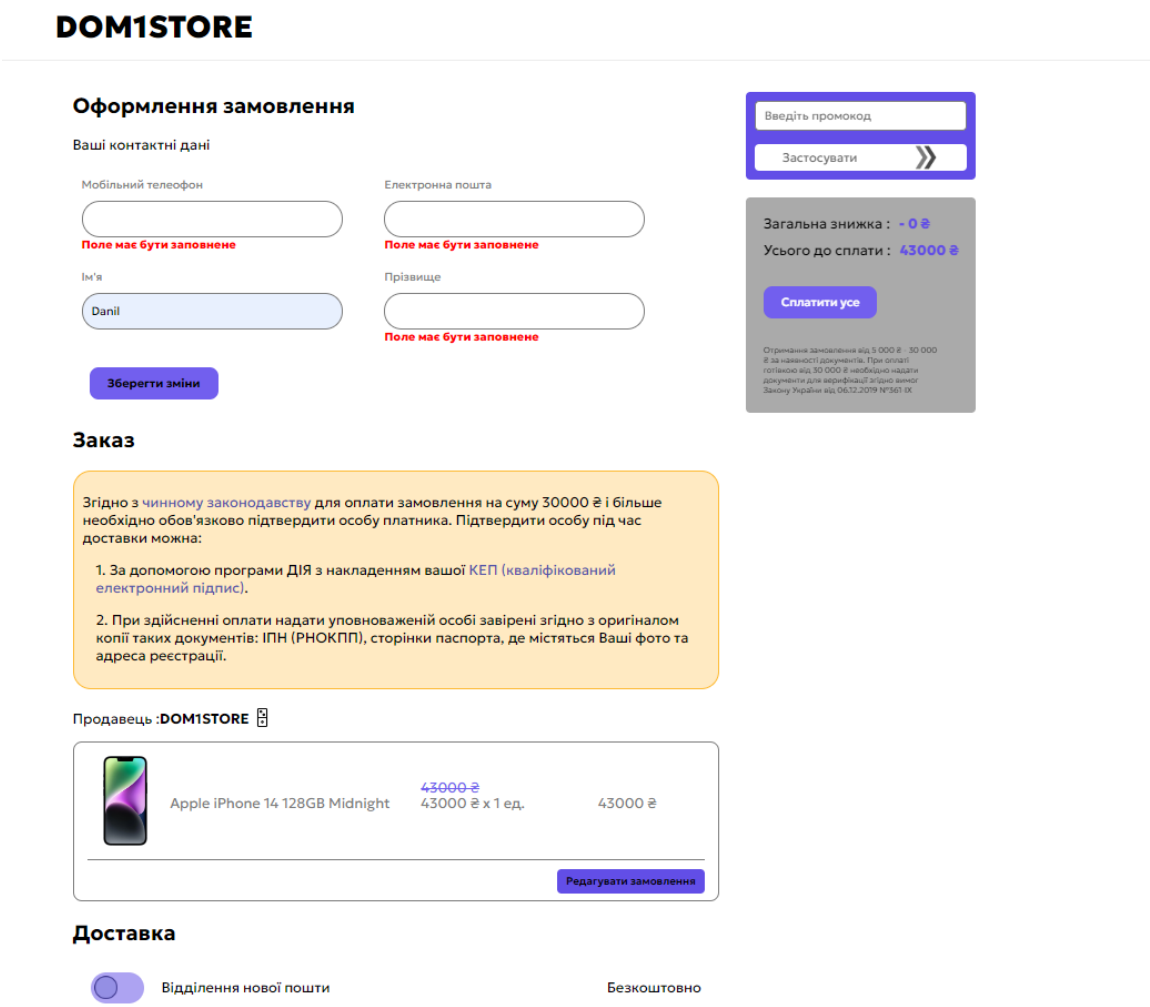

Рис. 2.9. Оформлення замовлення

Реєстрація та авторизація.

Опис: сторінка, де нові користувачі можуть створити обліковий запис, а зареєстровані користувачі можуть увійти у свій особистий кабінет.

Управління:

– Реєстрація нового користувача: користувач може створити новий обліковий запис, ввівши свої особисті дані, такі як ім'я, адреса електронної пошти, пароль, контактний номер телефону тощо.

– Вхід для зареєстрованих користувачів: користувач може увійти у свій обліковий запис, ввівши свою адресу електронної пошти та пароль.

– Захист даних: сторінка забезпечує захист особистих даних користувачів через шифрування та інші заходи безпеки.

Ілюстрація:

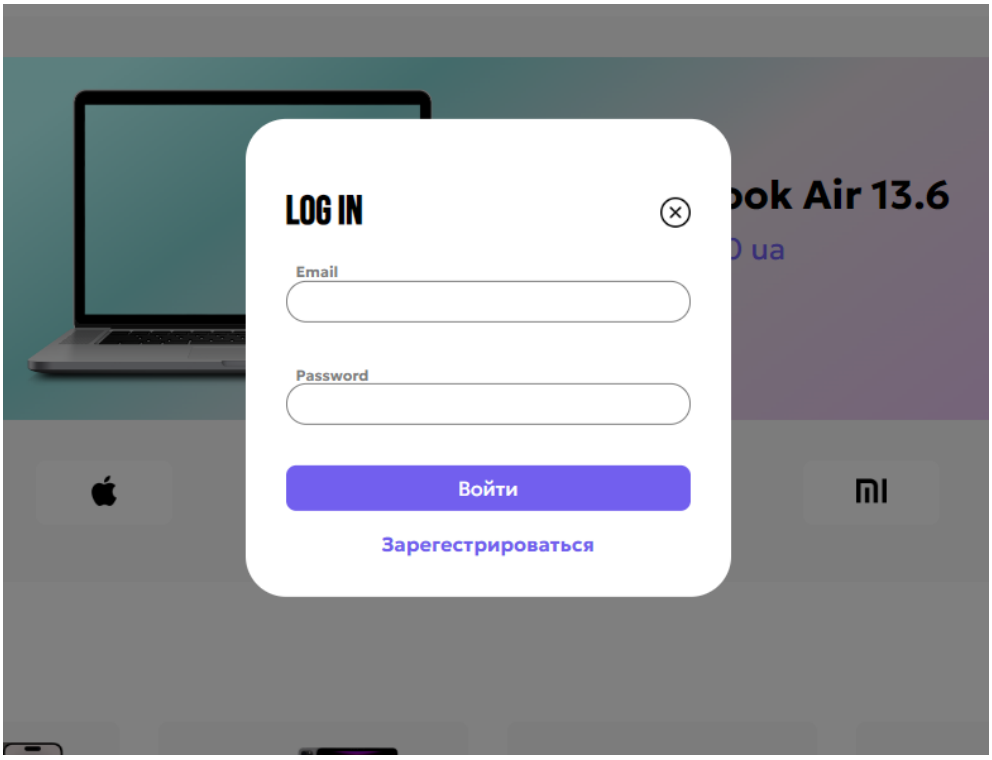

Рис. 2.10. Авторизація

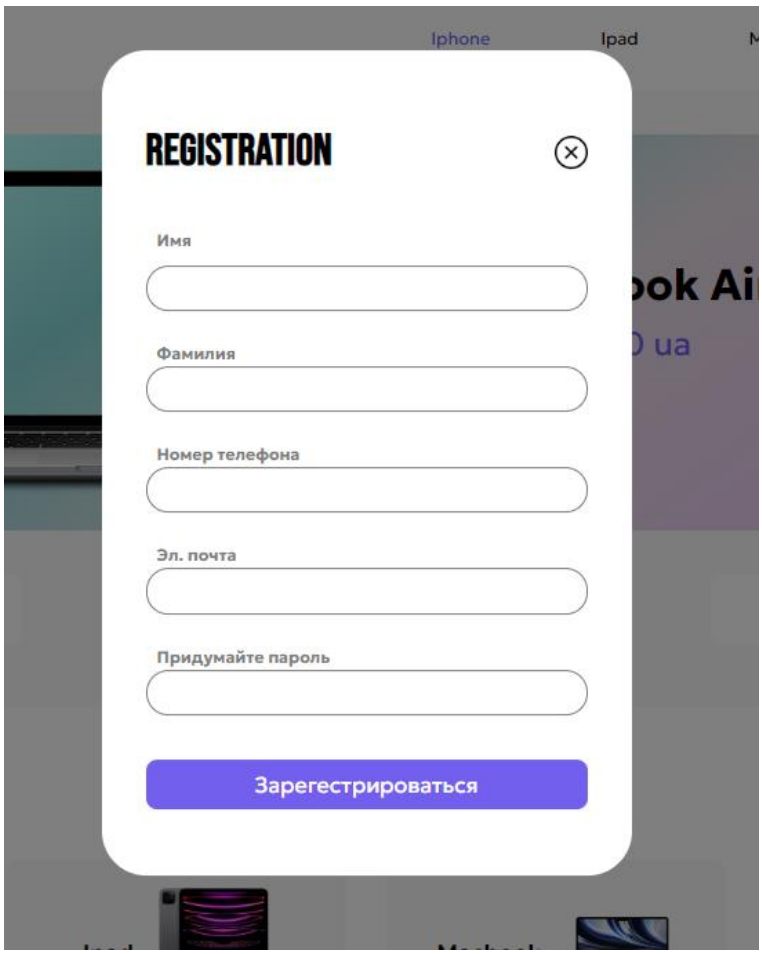

Рис. 2.11. Реєстрація

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2228;

2. коефіцієнт складності програми – 1,5;

3. коефіцієнт корекції програми в ході її розробки – 0,3;

4. годинна заробітна плата програміста – 208,87 грн/год;

За інформацією з сайту https://jobs.dou.ua/salaries/?period=2023-12&position=Junior%20SE, на 5 червня 2024 року середня зарплата для посади Junior Software Engineer становить $950. Згідно з Національним банком України, курс долара становить 40,05 гривень за долар , це еквівалентно 36 125 гривням. При стандартному графіку роботи 173 години на місяць, середня зарплата за годину складатиме приблизно 208,87 гривень.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1;

7. вартість машино-години ЕОМ – 1,4 грн/год.

Розрахунок вартості машино-години ЕОМ виконувався наступним чином: загальний час написання кваліфікаційної роботи склав 7 місяців. Протягом цього періоду в середньому кожного дня витрачалося по 5 годин на роботу, не враховуючи вихідних. Загальна кількість витрачених годин безперервного використання комп'ютера протягом 7 місяців, не враховуючи вихідні, склала 1050 годин.

Згідно тарифного плану, кВт/год коштує 2,64 грн з урахуванням ПДВ. Комп’ютер використовуваної моделі споживає близько 400 Вт на годину під час активної роботи. Витрати під час роботи на електроенергію складають 0,4 * 2,64

= 1,05 грн. Додатково, підключення до Інтернет-провайдера «Тріолан» передбачає щомісячну оплату у розмірі 250 грн. Таким чином, вартість використання ЕОМ на годину становить 0,35 грн. Отже, загальна вартість ЕОМ дорівнює 1,4 грн.

Можна визначити трудомісткість розробки програмного продукту за допомогою спеціальної формули.:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q - передбачуване число операторів (2228);

C - коефіцієнт складності програми (1,5);

p - коефіцієнт корекції програми в ході її розробки (0,3).

Звідси умовне число операторів в програмі:

$$Q = 1{,}5 \cdot 2228 \cdot (1 + 0{,}3) = 4344$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \qquad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

З урахуванням коефіцієнта кваліфікації $k = 1$, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (4344 \cdot 1{,}2) / (75 \cdot 1) = 69 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot k}, \text{ людино-годин,} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 4344 / (20 \cdot 1) = 217.2 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин,} \qquad (3.5)$$

t_n = 4344 / (25 · 1) = 173.76 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}, \text{ людино-годин,} \qquad (3.5)$$

$t_{отл}$ = 4344 / (5 · 1) = 869 людино-годин.

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5\cdot t_{отл}, \text{ людино-годин,} \qquad (3.6)$$

$t^k_{отл}$ = 1,5 · 869 = 1303.5 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \qquad (3.7)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}, \text{ людино-годин,} \qquad (3.8)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\text{, людино-годин,} \quad (3.9)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 4344 / (18 · 1) = 241.33 людино-годин.

$t_{до}$ = 0,75 · 241.33 = 180.9 людино-годин.

$t_{д}$ = 241.33 + 180.9 = 422.23 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 69 + 217.2+ 173.76 + 869 + 422.23 = 1801.19 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{, грн,} \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}\text{, грн,} \quad (3.11)$$

де: t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

Середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1801{,}19 \cdot 208{,}87 = 376\ 311{,}145 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн,} \qquad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (1,4).

Визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 869 \cdot 1{,}4 = 1216{,}6 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = = 1216{,}6 + 376\ 311{,}145 = 377\ 527{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.,} \qquad (3.12)$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=173 годин);

t – загальна трудомісткість, людино годин()

Звідси витрати на створення програмного продукту:

$$T = 1801{,}19 / 1 \cdot 173 \approx 10.4 \text{ міс.}$$

Висновок: для розробки ігрового застосунку з використанням рушія Unity загальна трудомісткість становить 1801,19 людино-годин, тому очікуваний період створення програмного додатку становитиме, приблизно, 10.4 місяців, а сумарні витрати на створення - 377 527,6 грн

ВИСНОВКИ

Розроблене програмне забезпечення є ефективним і працездатним інструментом для автоматизації діяльності інтернет-магазину електронної техніки. Воно дозволяє здійснювати автоматизований збір, обробку і маніпулювання даними, спрощуючи процеси продажу, обліку та контролю за товарами в мережі магазинів.

Програмне забезпечення надає можливість виконувати різноманітні операції, такі як додавання, видалення і редагування інформації про товари, клієнтів, постачальників, продажі, покупців, а також перегляд інформації про них.

Метою кваліфікаційної роботи було поставлено: розробити інтернет-магазину електронної техніки з використанням React, MobX, Sequelize.

Актуальність розробленого програмного забезпечення обумовлена широким попитом на такі програмні продукти, що надають можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Практичне призначення:

Для бізнесу: означає можливість ефективно представляти свої товари в онлайн просторі, що розширює аудиторію та збільшує потенційні продажі.

Для клієнтів: це зручний доступ до широкого асортименту електронної техніки, можливість порівняння товарів та їх характеристик, а також зручний процес оформлення замовлення та сплати онлайн.

Розроблене програмне забезпечення інформаційної системи призначене для застосування в будь-якій мережі інтернет-магазинів з подібним родом діяльності і схожими функціональними вимогами.

Створений додаток дозволить оптимізувати та спростити дії по веденню бази даних мережі магазинів, скоротити час на оформлення продажу, підвищити ефективність діяльності компанії шляхом електронного ведення документації з

продажу і можливістю аналізу наявних даних і надання необхідних звітів і супутньої ділової документації.

Структура програми складається з клієнтського додатку, написаного на мові програмування високого рівня JavaScript (з використанням React та Node.js), що взаємодіє з базою даних, розробленою мовою SQL в системі управління базами даних.

Для розробки проекту, призначеного для комерційної діяльності інтернет магазину з продажу електронних товарів, необхідно витратити 1801,19 людино-годин. З урахуванням цих даних можна припустити, що очікувана тривалість розробки продукту складатиме 10,4 місяців при стандартному робочому тижні та робочому місяці. Вартість цього програмного продукту складатиме 377 527,6 грн, включаючи заробітну плату фахівців, витрати на ЕОМ та витрати на його придбання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Touch URL: https://touch.com.ua (дата звернення: 11.04.2024).
2. Foxtrot URL: https://www.foxtrot.com.ua (дата звернення: 11.04.2024).
3. Що таке JavaScript URL: https://cases.media/en/article/sho-take-javascript (дата звернення: 11.04.2024).
4. Що таке NodeJs URL: https://devzone.org.ua/post/shcho-take-nodejs-osnovy-servernoyi-rozrobky-na-javascript (дата звернення: 11.04.2024).
5. Початок роботи з React URL: https://developer.mozilla.org/ru/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started (дата звернення: 11.04.2024).
6. Redux Керівництво
   URL: https://highload.today/redux-react/ (дата звернення: 11.04.2024).
7. HTTP клієнт для браузера і node.js на основі Promise URL: https://axios-http.com/ru/docs/intro (дата звернення: 11.04.2024).
8. React Formik + styled-components URL: https://medium.com/flyparakeet/react-formik-styled-components-add78b37971f (дата звернення: 11.04.2024).
9. JS: Предметно-орієнтований дизайн URL: https://ru.hexlet.io/courses/js-ddd/lessons/validation/theory_unit (дата звернення: 11.04.2024).
10. Просте масштабоване управління станом. URL: https://www.jscamp.app/ru/docs/state-management/reactnativestate00 (дата звернення: 11.04.2024).
11. Веб-фреймворк Express (Node.js/JavaScript)
    URL: https://www.jscamp.app/ru/docs/state-management/reactnativestate00 (дата звернення: 11.04.2024).
12. Як Sequelize допомагає з базами даних в Node.js URL: https://foxminded.ua/ru/sequelize-chto-eto/ (дата звернення: 11.04.2024).

13. PostgreSQL що це таке і чому це важливо?URL: https://foxminded.ua/ru/postgresql-chto-eto/#:~:text=Это%20свободная%20и%20открытая%20система,СУБД)%20с%20открытым%20исходным%20кодом. 0 (дата звернення: 11.04.2024).
14. Структура JWT (JSON Web Token) . URL: https://www.linkedin.com/pulse/10-питань-про-jwt-які-вас-можуть-спитати-сергей-s--ihxee (дата звернення: 11.04.2024).
15. Bcrypt – довідка. URL: https://reporter.zp.ua/bcrypt-dovidka.html (дата звернення: 11.04.2024).
16. Використання змінних середовища в Node.js URL: https://habr.com/ru/companies/ruvds/articles/351254/ (дата звернення: 11.04.2024).
17. Глибоке занурення в унікальні ідентифікатори . URL: https://www.tempmail.us.com/uk/javascript/Створення-унікальних-ідентифікаторів-у-javascript-посібник-із-uuid-та-guid (дата звернення: 11.04.2024).
18. Введение в Sequelize ORM для Node.js . URL: https://coursehunter.net/course/vvedenie-v-sequelize-orm-dlya-node-js (дата звернення: 11.04.2024).
19. ЯКУ ОС КРАЩЕ ОБРАТИ? URL: https://www.nspace.ua/info/windows-linux-ta-macos-chim-vidriznyayutsya-ta-yaku-os-obrati (дата звернення: 11.04.2024).
20. Використання Visual Studio Code URL: https://learn.microsoft.com/uk-ua/power-pages/configure/vs-code-extension (дата звернення: 11.04.2024).
21. Postman - must have для QA URL: https://qagroup.com.ua/publications/znajomtes-postman-must-have-dlia-qa/ (дата звернення: 11.04.2024).
22. Nginx — вебсервер URL: https://brander.ua/technologies/nginx (дата звернення: 11.04.2024).

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

NodeJs models

```
const sequelize = require("../db");
const { DataTypes } = require("sequelize");

const User = sequelize.define("user", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  email: { type: DataTypes.STRING, unique: true },
  password: { type: DataTypes.STRING },
  name: { type: DataTypes.STRING },
  surname: { type: DataTypes.STRING },
  phoneNumber: { type: DataTypes.STRING },
  patronymic: { type: DataTypes.STRING },
  date: { type: DataTypes.STRING },
  gender: { type: DataTypes.STRING },
  extension: { type: DataTypes.STRING },
  address: { type: DataTypes.STRING },
  role: { type: DataTypes.STRING, defaultValue: "USER" },
});

const Basket = sequelize.define("basket", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

const BasketDevice = sequelize.define("basket_device", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});
const Order = sequelize.define("order", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  orderEmail: { type: DataTypes.STRING },
  phoneNumber: { type: DataTypes.STRING },
  delivery: { type: DataTypes.STRING },
  name: { type: DataTypes.STRING },
  surname: { type: DataTypes.STRING },
  payment: { type: DataTypes.STRING },
  cost: { type: DataTypes.INTEGER },
```

```
});

const OrderDevices = sequelize.define("order_device", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true },
  category: { type: DataTypes.STRING },
  count: { type: DataTypes.STRING },
});

const Device = sequelize.define("device", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
  price: { type: DataTypes.STRING, allowNull: false },
  rating: { type: DataTypes.INTEGER, defaultValue: 0 },
  img: { type: DataTypes.STRING, allowNull: false },
  img2: { type: DataTypes.STRING, allowNull: false },
  img3: { type: DataTypes.STRING, allowNull: false },
  img4: { type: DataTypes.STRING, allowNull: false },
  discount: { type: DataTypes.INTEGER, allowNull: false },
  catigory: { type: DataTypes.STRING, allowNull: false },
  relevance: { type: DataTypes.STRING, allowNull: false },
});

const Type = sequelize.define("type", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
  link: { type: DataTypes.STRING, unique: true, allowNull: false },
});

const Brand = sequelize.define("brand", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, unique: true, allowNull: false },
});

const Rating = sequelize.define("rating", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  rate: { type: DataTypes.INTEGER, allowNull: false },
});

const DeviceInfo = sequelize.define("device_info", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  title: { type: DataTypes.STRING, allowNull: false },
```

```
  description: { type: DataTypes.STRING, allowNull: false },
});

const TypeBrand = sequelize.define("type_brand", {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
});

User.hasOne(Basket);
Basket.belongsTo(User);

User.hasMany(Rating);
Rating.belongsTo(User);

Basket.hasMany(BasketDevice);
BasketDevice.belongsTo(Basket);

Order.hasMany(OrderDevices, { as: "devices" });
OrderDevices.belongsTo(Order);

Type.hasMany(Device);
Device.belongsTo(Type);

Brand.hasMany(Device);
Device.belongsTo(Brand);

Device.hasMany(Rating);
Rating.belongsTo(Device);

Device.hasMany(BasketDevice);
BasketDevice.belongsTo(Device);

Device.hasMany(DeviceInfo, { as: "info" });
DeviceInfo.belongsTo(Device);

Type.belongsToMany(Brand, { through: TypeBrand });
Brand.belongsToMany(Type, { through: TypeBrand });

module.exports = {
  User,
  Basket,
  BasketDevice,
  Device,
```

```
  Type,
  Brand,
  Rating,
  TypeBrand,
  DeviceInfo,
  Order,
  OrderDevices,
};
```

Device controller

```
const uuid = require("uuid");
const path = require("path");
const { Device, DeviceInfo } = require("../models/models");
const { Op } = require("sequelize");
const ApiError = require("../error/ApiError");

class DeviceController {
  async create(req, res, next) {
    try {
      let {
        name,
        price,
        brandId,
        typeId,
        info,
        discount,
        catigory,
        relevance,
      } = req.body;
      const { img, img2, img3, img4 } = req.files;

      let fileName = uuid.v4() + ".jpg";
      img.mv(path.resolve(__dirname, "..", "static", fileName));

      let fileName2 = uuid.v4() + ".jpg";
      img2.mv(path.resolve(__dirname, "..", "static", fileName2));

      let fileName3 = uuid.v4() + ".jpg";
      img3.mv(path.resolve(__dirname, "..", "static", fileName3));

      let fileName4 = uuid.v4() + ".jpg";
      img4.mv(path.resolve(__dirname, "..", "static", fileName4));
```

```
    const device = await Device.create({
      name,
      price,
      brandId,
      typeId,
      img: fileName,
      img2: fileName2,
      img3: fileName3,
      img4: fileName4,
      discount,
      catigory,
      relevance,
    });

    if (info) {
      info = JSON.parse(info);
      info.forEach((el) => {
        DeviceInfo.create({
          title: el.title,
          description: el.description,
          deviceId: device.id,
        });
      });
    }

    return res.json(device);
  } catch (e) {
    next(ApiError.badRequest(e.message));
  }
}
async getAll(req, res) {
  let { brandId, typeId, limit, page, sortBy } = req.query;
  page = page || 1;
  limit = limit || 9;
  let offset = page * limit - limit;
  let device;

  let order = [];
  if (sortBy === "priceAscending") {
    order.push(["id", "ASC"]);
  } else if (sortBy === "priceDescending") {
```

```
    order.push(["id", "DESC"]);
  }

  if (!brandId && !typeId) {
    device = await Device.findAndCountAll({ limit, offset, order });
  }
  if (brandId && !typeId) {
    device = await Device.findAndCountAll({
      where: { brandId },
      limit,
      offset,
      order,
    });
  }
  if (!brandId && typeId) {
    device = await Device.findAndCountAll({
      where: { typeId },
      limit,
      offset,
      order,
    });
  }
  if (brandId && typeId) {
    device = await Device.findAndCountAll({
      where: { brandId, brandId },
      limit,
      offset,
      order,
    });
  }
  return res.json(device);
}
async getOne(req, res) {
  const { id } = req.params;
  const device = await Device.findOne({
    where: { id },
    include: [{ model: DeviceInfo, as: "info" }],
  });
  return res.json(device);
}
async getActualDevices(req, res) {
  let { limit, page } = req.query;
```

```
    page = page || 1;
    limit = limit || 9;
    let offset = page * limit - limit;
    let device;

    try {
      device = await Device.findAndCountAll({
        where: { relevance: "actual" },
        limit,
        offset,
      });

      return res.json(device);
    } catch (error) {
      return res.status(500).json({ error: "Internal Server Error" });
    }
  }

  async getSearchDevices(req, res) {
    let { limit, page, searchTerm } = req.query;
    page = page || 1;
    limit = limit || 9;
    let offset = page * limit - limit;

    try {
      let devices;

      devices = await Device.findAndCountAll({
        where: {
          name: { [Op.iLike]: `%${searchTerm}%` },
        },
        limit,
        offset,
      });

      return res.json(devices);
    } catch (error) {
      return res.status(500).json({ error: "Internal Server Error" });
    }
  }
}
```

```
module.exports = new DeviceController();
```

Device API

```
import { $authHost, $host } from ".";

export const createType = async (type) => {
  const { data } = await $authHost.post("api/type", type);
  return data;
};

export const fetchTypes = async () => {
  const { data } = await $host.get("api/type");
  return data;
};

export const createBrand = async (brand) => {
  const { data } = await $authHost.post("api/brand", brand);
  return data;
};

export const fetchBrands = async () => {
  const { data } = await $host.get("api/brand");
  return data;
};

export const createDevice = async (device) => {
  const { data } = await $authHost.post("api/device", device);
  return data;
};

export const fetchDevices = async (typeId, brandId, page, limit = 5) => {
  const { data } = await $host.get("api/device", {
    params: {
      typeId,
      brandId,
      page,
      limit,
    },
  });
  return data;
};
```

```
export const fetchOneDevice = async (id) => {
  const { data } = await $host.get("api/device/" + id);
  return data;
};

export const fetchActualDevices = async (page, limit = 5) => {
  const { data } = await $host.get("api/device/actual", {
    params: {
      page,
      limit,
      relevance: "actual",
    },
  });
  return data;
};
// export const fetchSearchDevices = async (page, limit = 5) => {
//   const { data } = await $host.get("api/device/search", {
//     params: {
//       page,
//       limit,
//     },
//   });
//   return data;
// };
export const fetchSearchDevices = async (page, limit = 5, searchTerm = "") => {
  try {
    const { data } = await $host.get("api/device/search", {
      params: {
        page,
        limit,
        searchTerm,
      },
    });
    return data;
  } catch (error) {
    console.error("Error fetching search devices:", error);
    throw error;
  }
};
```

React router

```
function AppRouter() {
  const { users } = useContext(CustomContext);
  return (
    <Routes>
      {users.userRole == "ADMIN" &&
        authRoutes.map(({ path, element, children }) => (
          <Route key={path} path={path} element={element}>
            {children &&
              children.map(({ path: childPath, element: childElement }) => (
                <Route
                  key={childPath}
                  path={childPath}
                  element={childElement}
                />
              ))}
          </Route>
        ))}
      {publicRoutes.map(({ path, element, children }) => (
        <Route key={path} path={path} element={element}>
          {children &&
            children.map(({ path: childPath, element: childElement }) => (
              <Route key={childPath} path={childPath} element={childElement} />
            ))}
        </Route>
      ))}
      <Route path="*" element={<NotFound />} />
    </Routes>
  );
}

export default AppRouter;
import Prod from "../AllProducts/ProdList";
import ProdListSerch from "../AllProducts/ProdListSerch";

import Landing from "../MainLanding/Landing";
import PersonalArea from "../PersonalArea/PersonalArea";
import Desired from "../PersonalArea/components/Desired";
import Orders from "../PersonalArea/components/Orders";
import Profile from "../PersonalArea/components/Profile";
import Sms from "../PersonalArea/components/Sms";
```

```
import Stock from "../PersonalArea/components/Stock";
import Product from "../Product/Product";
import Basket from "../ToolsPages/pages/Basket";
import Like from "../ToolsPages/pages/Like";
import Admin from "../Admin/Admin";
import {
  ADMIN_ROUT,
  BASKET_ROUT,
  DESIRED_ROUT,
  LIKE_ROUT,
  LOGIN_ROUT,
  ORDERS_ROUT,
  PERSONAL_ROUT,
  PRODUCTS_ROUT,
  PRODUCT_ROUT,
  PRODUCT_SEARCH_ROUT,
  REGISTRATION_ROUT,
  SHOP_ROUT,
  SMS_ROUT,
  STOCK_ROUT,
} from "../utils/consts";
import Order from "../ordering/Order";
import Auth from "../Authoriz/page/Auth";
import Register from "../Authoriz/page/Register";
import AdminAuth from "../Admin/AdminAuth";
import AllOrders from "../Admin/components/AllOrders";
import ProductAdding from "../Admin/components/ProductAdding";
import Customers from "../Admin/components/Customers";

export const authRoutes = [
  {
    path: "/admin",
    element: <Admin />,
    children: [
      {
        path: "",
        element: <ProductAdding />,
      },
      {
        path: "orders",
        element: <AllOrders />,
      },
```

```
      {
        path: "customers",
        element: <Customers />,
      },
    ],
  },
];
export const publicRoutes = [
  {
    path: "/admin",
    element: <AdminAuth />,
  },
  {
    path: "/login",
    element: <Auth />,
  },
  {
    path: "/order",
    element: <Order />,
  },
  {
    path: "/registration",
    element: <Register />,
  },
  {
    path: "/products/:type",
    element: <Prod />,
  },
  {
    path: "/:type/:name/:id",
    element: <Product />,
  },
  {
    path: "/personal",
    element: <PersonalArea />,
    children: [
      {
        path: "",
        element: <Profile />,
      },
      {
        path: "orders",
```

```
        element: <Orders />,
      },
      {
        path: "desired",
        element: <Desired />,
      },
      {
        path: "stock",
        element: <Stock />,
      },
      {
        path: "sms",
        element: <Sms />,
      },
    ],
  },
  {
    path: "/search/:inquiry",
    element: <ProdListSerch />,
  },

  {
    path: "/like",
    element: <Like />,
  },
  {
    path: "/basket",
    element: <Basket />,
  },
  {
    path: "/",
    element: <Landing />,
  },
];
```

ДОДАТОК Б

ВІДГУК

Керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

«Розробка інтернет-магазину електронних пристроїв та аксесуарів з використанням React, MobX, Sequelize»

Студента групи 121-20-1 Гордієнка Данила Дмитровича

Керівник економічного розділу
доц. каф. ПЕП та ПУ,
к.е.н **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_ Гордієнко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_ Гордієнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Dom1store.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Гордієнко.ppt	Презентація кваліфікаційної роботи