МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ НА БАЗІ ТЕХНОЛОГІЙ JS, MYSQL ТА ПРЕПРОЦЕСОРУ SASS»

Виконав: здобувач освіти 4 курсу, групи К121-20
Спеціальність 121 Інженерія
програмного забезпечення
(шифр і спеціальність)
Данило ЄВСЕЄВ
(ім’я та ПРІЗВИЩЕ)
Керівник Анна НЕЛАСА
(ім’я та ПРІЗВИЩЕ)

Рецензент доцент кафедри програмних засобів НУ "Запорізька політехніка", доцент, к.т.н. Олександр СТЕПАНЕНКО
(посада, вчене звання, науковий ступінь, ім’я та ПРІЗВИЩЕ)

Запоріжжя
2024

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо–кваліфікаційний рівень фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Т.М. Смолянкова
(підпис)

“ 14 ” червня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ЄВСЕЄВУ Данилу Андрійовичу
(прізвище, ім’я та по– батькові)

1. Тема роботи «Розробка інтернет-магазину одягу на базі технологій JS, MySQL та препроцесору SASS»

Керівник роботи к.т.н. НЕЛАСА Анна Вікторівна
(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від « 30 » листопада 2023 р. № 2004-с

2. Строк подання студентом роботи 01.06.2024

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

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1. Основи веброзробки та аналіз існуючих застосунків
2. Розробка проєкту застосунку
3. Програмна реалізація

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Анна Неласа		

7. Дата видачі завдання 01.12.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	Постановка задачі	Грудень 2023	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2024	виконано
3.	Розробка першого розділу:		
	Огляд інформаційних систем та технологій для їхнього створення	Лютий 2024	виконано
4.	Розробка другого розділу:		
	Розробка проєкту інформаційної системи та вибір технологій	Квітень 2024	виконано
5.	Розробка третього розділу:		
	Розробка застосунку	Травень 2024	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2024	виконано
7.	Захист кваліфікаційної роботи	21.06.2024	виконано

Здобувач освіти _____ Данило ЄВСЕЄВ
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник роботи _____ Анна НЕЛАСА
(підпис) (ім'я ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 49 сторінок, 3 таблиці, 13 ілюстрацій, 17 позицій у переліку посилань, 2 додатки.

Об'єкт дослідження – методи та засоби розробки вебзастосунків для інтернет-магазинів.

Предмет дослідження – середовище розробки вебзастосунків з використанням HTML, SASS (SCSS) та Javascript фреймворку Vue.

Мета дослідження – створення вебзастосунку для інтернет-магазину одягу, що включатиме функції виводу переліку товарів, корзини, процесу покупки, оплати, а також особистого кабінету для керування замовленнями та перегляду історії покупок, забезпечуючи при цьому зручний інтерфейс для користувачів.

У рамках кваліфікаційної роботи було проведено детальний аналіз теоретичних основ систем управління контентом та огляд сучасних ефективних систем для електронної комерції. На основі отриманих знань був спроєктований та розроблений інформаційний вебзастосунок для інтернет-магазину одягу. Проект включав створення бази даних за допомогою мови запитів SQL, що забезпечує гнучке керування даними. Використання фреймворку Express.js дозволило ефективно організувати серверну частину, включаючи розробку API для взаємодії клієнтської частини з базою даних. Для клієнтської частини було обрано фреймворк Vue.js, який сприяв створенню інтуїтивно зрозумілого інтерфейсу, через який користувачі мають змогу переглядати асортимент, вибирати та купувати товари.

ВЕБЗАСТОСУНОК, ОНЛАЙН-МАГАЗИН, HTML, SASS, VUE, SQL, EXPRESS, API, NODEJS, ОДЯГ.

SUMMARY

Thesis: 49 pages, 3 tables, 13 illustrations, 2 appendices, 17 references.

The object of research is methods and means of developing web applications for online stores.

The subject of research is the web application development environment using HTML, SASS (SCSS) and Javascript framework Vue.

The purpose of the research is to create a web application for an online clothing store, which will include the functions of displaying a list of products, shopping cart, purchase process, payment, as well as a personal account for managing orders and viewing purchase history, while providing a user-friendly interface.

A detailed analysis was carried out as part of the qualification project e-commerce systems were conducted. Based on the knowledge gained, an information web application for an online clothing store was designed and developed. The project included the creation of a database using the SQL query language, which provides flexible data management. The use of the Express.js framework allowed us to efficiently organize the server side, including the development of an API for the client side to interact with the database. For the client side, we chose the Vue.js framework, which helped to create an intuitive interface through which users can browse the assortment, select and purchase products.

WEB APPLICATION, ONLINE STORE, HTML, SASS, VUE, SQL, EXPRESS, API, NODEJS, CLOTHING.

ЗМІСТ

ВСТУП

У епоху цифровізації, коли онлайн-торгівля стає все більш популярною, створення вебсайту для інтернет-магазину одягу перетворюється на стратегічний крок для будь-якого бізнесу. Такий вебсайт служить не лише платформою для продажу товарів, а й місцем, де бренд може виразити свою унікальність, залучити клієнтів і надати їм винятковий досвід. Забезпечення зручного інтерфейсу для вибору та покупки одягу, а також інформування про нові колекції та спеціальні пропозиції є ключовими елементами успішного інтернет-магазину.

Ця кваліфікаційна робота має на меті розробити вебзастосунок для інтернет-магазину одягу, використовуючи сучасні вебтехнології, такі як HTML, SASS, та JavaScript з фреймворком Vue.js. Вебзастосунок покликаний забезпечити користувачам легкий доступ до асортименту продукції, можливість здійснити покупку в кілька кліків і отримати всю необхідну інформацію про товари та акції.

Завданнями дослідження є:

– аналіз ринку інтернет-магазинів одягу для визначення кращих практик у цій галузі;

– вивчення основних технологій веброзробки, включаючи html для структурування контенту та sass для стилізації;

– вивчення будування логіки з використанням vue.js для створення інтуїтивно зрозумілого інтерфейсу, який сприяє залученню та утриманню клієнтів.

Об’єкт дослідження охоплює процеси та методики створення вебзастосунків для електронної комерції, з акцентом на інтернет-магазини одягу.

Предмет дослідження зосереджений на використанні конкретних вебтехнологій для розробки таких додатків.

Дослідження використовує виключно онлайн-ресурси та практичні інструменти, наприклад, такі як Visual Studio Code, який є універсальним редактором коду для веброзробників.

Дипломна робота структурована на вступ, три основні розділи, висновки, список використаних джерел та додатки.

У першому розділі представлено огляд основ веброзробки, акцентуючи на новітніх тенденціях і технологіях, які підвищують ефективність та привабливість вебсайтів.

Другий розділ зосереджується на плануванні проєкту, включаючи розробку технічного завдання, визначення вимог до функціональності додатку, а також проектування бази даних і архітектури додатку.

Третій розділ описує процес розробки: від верстки інтерфейсу до імплементації логіки додатку, використовуючи Vue.js, з метою створення зручного і ефективного інтернет-магазину одягу.

1 ОСНОВИ ВЕБ-РОЗРОБКИ ТА АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ

1.1 Веброзробка та її складові

Простими словами, веброзробка – це створення динамічних вебзастосунків та вебсайтів [1]. Веброзробка включає в себе широкий спектр технологій, інструментів та практик, кожна з яких відіграє важливу роль у створенні кінцевого продукту, доступного користувачам через інтернет.

Кожен вебзастосунок поділяється на клієнтську та серверну частину. Клієнтська сторона веброзробки (frontend), зосереджена на тому, як вебзастосунок відображається та взаємодіє з користувачем. Ця сфера включає в себе:

– HTML: мова розмітки, яка використовується для створення та структурування вмісту на вебсторінці;

– CSS: мова стилів, що дозволяє оформляти вебсторінку, включаючи колір, шрифт, розмір елементів, та багато іншого. Препроцесори CSS, такі як SASS, надають додаткові можливості для ефективнішої роботи зі стилями, наприклад, такі як: змінні, міксини, вкладеності тощо;

– JavaScript: мова програмування, що додає інтерактивність вебсторінкам, дозволяючи створювати динамічний вміст та багатофункціональні вебзастосунки.

В цій роботі використовувався фреймворк Vue. Фреймворк – це набір інструментів, бібліотек та правил, який використовується для створення програмних додатків [2]. Vue – це фреймворк, який працює на JavaScript, створений для розробки користувацьких інтерфейсів [3].

Серверна сторона відповідає за обробку даних, логіку додатку, взаємодію з базами даних, та забезпечення функціонування вебзастосунків. Ця частина веброзробки залучає такі аспекти:

– мови програмування: такі як python, ruby, php, javascript, що використовуються для розробки серверної логіки;

– бази даних: системи для зберігання та управління даними, включаючи sql (наприклад, postgresql, mysql) та nosql (наприклад, mongodb) бази даних;

– серверне програмне забезпечення: таке як apache, nginx, або сервери, що використовуються в node.js екосистемі, дозволяє управляти запитами до вебсервера та відповідями від нього.

Для створення серверної частини застосунку у межах кваліфікаційної роботи використовувалося середовище Node.js, фреймворк Express та середовище MySQL. Node.js – це однопоточне кросплатформове середовище виконання з відкритим вихідним кодом і бібліотека, яка використовується для запуску вебзастосунків, написаних на JavaScript, поза браузером клієнта [4]. Express – це мінімалістичний та гнучкий фреймворк для вебзастосунків, побудованих на Node.js, що надає широкий набір функціональності [5]. MYSQL – це система управління реляційними базами [6].

Використання сучасних інструментів та підходів у веброзробці дозволяє досягати високого рівня інтерактивності та задовольняти вимоги сучасних користувачів.

1.2 Сучасні тенденції веброзробки

При створенні вебзастосунку у наш час дуже важливо притримуватися та використовувати сучасні тенденції та тренди розробки. Це значно підвищує юзабіліті застосунку.

Однією з тендецій останніх років є адаптивний та респонсивна верстка додатку. У світі, де користувачі мають доступ до вебконтенту з різноманітних пристроїв з різними розмірами екранів, адаптивний та респонсивний дизайн стає не просто бажаним, а обов'язковим. Респонсивний

дизайн забезпечує вебсторінкам здатність автоматично адаптуватися до будь-якого розміру екрану, гарантуючи оптимальне відображення та читабельність контенту без потреби в горизонтальному скролінгу або зміні розміру елементів. Адаптивний дизайн, у свою чергу, передбачає створення кількох версій вебсторінки для різних розмірів екранів, що дозволяє досягти ще більшої специфічної оптимізації. В розробленому додатку використовується як адаптивний, так і респонсивний дизайн.

Використання фреймворку Vue дає змогу створювати односторінкові додатки (SPA), які продовжують набирати популярність завдяки своїй здатності забезпечити швидку та плавну взаємодію користувача з вебзастосунком, імітуючи відчуття роботи з настільним застосунком. SPA завантажує дані в фоновому режимі та оновлює вміст сторінки без необхідності перезавантаження всієї сторінки. Це знижує час завантаження та покращує користувацький досвід.

1.3 Аналіз предметної області

Інтернет-магазин одягу охоплює велику кількість процесів: від вибору та представлення товарів до управління замовленнями та логістики. Ключові аспекти включають каталогізацію продукції, системи фільтрації та пошуку, управління користувацькими профілями, системи оплати та доставки, а також забезпечення високого рівня безпеки даних користувачів.

Серед основних проблем, з якими стикаються інтернет-магазини одягу, можна виділити наступні:

– висока конкуренція: сфера електронної комерції є надзвичайно конкурентною. виділитися серед численних магазинів та привернути увагу потенційних покупців стає все складніше;

– вимоги до якості зображень та опису товарів: детальні та якісні фотографії продукції, а також змістовні описи відіграють ключову роль у прийнятті рішення про покупку;

– персоналізація досвіду користувача: надання індивідуалізованого досвіду покупок, включаючи персоналізовані рекомендації та пропозиції, стає вирішальним фактором у залученні та утриманні клієнтів;

– мобільна адаптація: оптимізація вебзастосунку для мобільних пристроїв є необхідністю, враховуючи зростаючу кількість користувачів, які здійснюють покупки через смартфони.

Розробка нового інтернет-магазину одягу вимагає інноваційного підходу до вирішення зазначених проблем. Актуальність проекту обумовлена потребою в створенні вебзастосунку, який не лише вирішує функціональні завдання, але й пропонує користувачам унікальний та зручний досвід покупок. Важливими аспектами є впровадження сучасних технологій для покращення інтерфейсу та навігації та оптимізація процесів покупки та оплати.

1.4 Огляд існуючих інтернет магазинів

Для огляду оберемо найпопулярніші світові інтернет-магазини одягу: ASOS, Zara, та H&M.

1. ASOS

Технічна реалізація: вебсайт ASOS використовує сучасні вебтехнології для забезпечення плавної взаємодії з користувачем.

Зручність сайту: функціональність пошуку та фільтрації на ASOS є одними з найкращих серед інтернет-магазинів одягу, дозволяючи користувачам легко знайти потрібні товари за різними параметрами. інтерфейс є інтуїтивно зрозумілим і легким для навігації.

2. Zara

Технічна реалізація: сайт Zara виділяється своїм унікальним візуальним стилем, що демонструє високу якість зображень та анімацій. Водночас, це може призводити до додаткового часу завантаження сторінок, особливо на пристроях з обмеженим інтернет-з'єднанням.

Зручність сайту: хоча дизайн сайту Zara є вишуканим, деякі користувачі можуть знайти його менш інтуїтивно зрозумілим порівняно з іншими платформами. навігація зосереджена більше на візуальному досвіді, що може ускладнити пошук конкретних товарів.

3. H&M

Технічна реалізація: H&M використовує стандартний підхід до розробки свого вебсайту, забезпечуючи гарну сумісність з різними пристроями та браузерами. Сайт ефективно використовує кешування та оптимізацію зображень для швидкого завантаження.

Зручність сайту: H&M пропонує зручний та простий у використанні інтерфейс, з ясно структурованими категоріями та легкодоступною інформацією про продукти. Проте, можливості персоналізації та рекомендацій продуктів є менш розвинутими порівняно з ASOS.

Порівняльний аналіз існуючих інтернет-магазинів одягу показує, що успіх у цій сфері залежить від балансу між візуальною привабливістю та технічною ефективністю сайту. Для власного застосунку були враховані ці уроки, щоб забезпечити як високий рівень користувацького досвіду, так і технічну оптимізацію.

У першому розділі кваліфікаційної роботи розглянули ключові аспекти сучасної веброзробки, включаючи технології, методики та найновіші тенденції у цій галузі.

Аналіз існуючих інтернет-магазинів одягу виявив важливість адаптивного дизайну, ефективної взаємодії користувача, а також значення швидкості та зручності навігації сайту. Порівняльний аналіз популярних платформ, таких як ASOS, Zara та H&M, підкреслив необхідність інноваційного підходу до розробки та оптимізації вебзастосунків, що є ключем до залучення та утримання клієнтів у висококонкурентному середовищі електронної комерції.

2 РОЗРОБКА ПРОЄКТУ ВЕБЗАСТОСУНКУ

2.1 Технічне завдання

2.1.1 Найменування і область застосування

Вебзастосунок для магазину одягу "3legant". Даний вебзастосунок призначений для полегшення процесу вибору та придбання одягу, надання інформації про товари, послуги та забезпечення зручного інтерфейсу для користувачів.

2.1.2 Підстава для розробки

Сайт розробляється на підставі наказу «Про затвердження тем кваліфікаційних робіт студентів 4 курсу освітньо-професійного ступеня фаховий молодший бакалавр денної форми здобуття освіти ВСП «Економіко-правничий фаховий коледж ЗНУ» від 30.11.2023 №2004-с.

2.1.3 Призначення розробки

– Надання інформації про магазин одягу "3legant", його асортимент, акції, нові колекції та іншу корисну інформацію.

– Забезпечення можливості онлайн-покупки товарів.

– Надання персоналізованого кабінету для кожного користувача, де він зможе переглядати свої замовлення, змінювати адресу доставки, переглядати історію покупок тощо.

2.1.4 Технічні вимоги до програмного продукту

На етапі початкового проектування були визначені наступні основні вимоги до вебзастосунку магазину одягу "3legant":

– Необхідність реєстрації та авторизації користувачів для доступу до особистих кабінетів.

– Функціонал для пошуку та перегляду товарів за різними параметрами (категорія, розмір, колір, ціна тощо).

– Можливість здійснення покупок онлайн.

– Обов'язкова можливість перегляду деталей замовлення та внесення змін до нього.

– Вимога до адаптивного дизайну, який забезпечує правильне відображення вебзастосунку на різних типах пристроїв та їх екранах.

2.2 Етапи створення вебзастосунку

Процес розробки вебзастосунку для онлайн-покупок одягу "3legant" з використанням Vue.js, SQL та Express.js можна розділити на наступні етапи:

– Визначення цілей створення додатку та складання технічного завдання та вимог до системи. На цьому етапі важливо чітко визначити функціональні та нефункціональні вимоги до додатку, включаючи можливості покупки, пошуку товарів, оплати замовлень та збереження даних користувачів.

– Розробка макетів сторінок додатку з використанням Vue.js. На цьому етапі важливо створити єдиний стиль для всіх сторінок, забезпечити зручну навігацію. Крім того, важливо врахувати адаптивність додатку під різні розширення екранів.

– Проєктування бази даних з використанням мови запитів SQL. База даних повинна бути спроєктована таким чином, щоб забезпечити збереження

інформації про доступні товари, замовлення користувачів та інші необхідні дані для системи покупок одягу.

– Вибір технологій для реалізації додатку. Для фронтенду можна обрати Vue.js, який дозволить побудувати інтерактивний та зручний інтерфейс користувача. Для бекенду можна використовувати Express.js, який надасть можливість створити серверну частину додатку та взаємодіяти з базою даних MySQL.

– Реалізація бекенду та фронтенду додатку. На цьому етапі розробляються серверні API з використанням Express.js, які відповідають за обробку запитів користувачів та взаємодію з базою даних MySQL. Фронтенд реалізується з використанням Vue.js, де відбувається відображення інтерфейсу користувача, обробка подій та взаємодія з сервером через API.

– Тестування додатку. Тестування виконується як на рівні окремих компонентів, так і на рівні взаємодії між компонентами та системою в цілому. Перевіряються функціональність, надійність та безпека додатку.

2.3 Діаграма прецедентів

Діаграма варіантів використання візуально відображає сферу застосування рішення, показуючи акторів, які взаємодіють з рішенням, варіанти використання, з якими вони взаємодіють, і будь-які взаємозв'язки між варіантами використання [7]. Перелік акторів та основних прецедентів представлені в таблиці 2.1.

Таблиця 1.2 – Перелік акторів та прецедентів

Актор	Прецеденти
Адміністратор	Перегляд каталогу Додавання товару до кошика та оформлення замовлення Реєстрація та авторизація
Користувач	Додавання та редагування товарів Керування та моніторинг замовлень Керування користувачами

Діаграма прецедентів, представлена на рисунку 2.1, була розроблена на основі інформації, наведеної у таблиці акторів та прецедентів.

Рисунок 2.1 – Діаграма прецедентів

2.4 Проєктування бази даних

У процесі розробки вебзастосунку було розроблено та створено реляційну базу даних MySQL. Реляційна база даних – це тип бази даних, що зберігає інформацію в електронних таблицях і здійснює пошук даних в одній таблиці на підставі визначених ключових полів іншої таблиці. Програми, які призначені для структурування інформації, розміщення її в таблицях і маніпулювання даними, називаються системами управління базами даних (СУБД) [8]. У роботі була використана СУБД MySQL. Визначені основні сутності:

- Продукт (Products)
- Користувач (Users)
- Категорії (Categories)
- Замовлення (Orders)
- Промокоди (Promocodes)
- Оплати (Payments)
- Повернення (Refunds)

У таблиці "Продукти" зберігається інформація про товари, включаючи назву, ідентифікатор категорії, доступні розміри, кольори, ціну, короткий та детальний опис, фотографії та доступність.

Таблиця "Користувачі" містить дані про користувачів додатку, включно з ім'ям, прізвищем, електронною поштою, паролем, адресою та замовленнями.

У таблиці "Категорії" знаходяться документи з інформацією про категорії продуктів, включаючи назву, опис та фотографію.

Таблиця "Замовлення" містить інформацію про замовлення користувачів, включаючи ідентифікатор користувача, продукти, доступні розміри та статус замовлення.

У таблиці "Промокоди" зберігаються дані про промокоди, включно з кодом, розміром знижки та датою закінчення дії.

"Оплати" – це таблиця, яка відстежує усі платежі, з ідентифікатором замовлення, датою, сумою, статусом та зв'язаним промокодом.

Нарешті, таблиця "Повернення" включає інформацію про повернення коштів, включно з ідентифікатором платежу, датою та причиною повернення.

Перед створенням бази даних важливо мати структурований огляд бази для кращого розуміння її організації. Таким чином, на рисунку 2.2 була створена схема бази даних, яка візуалізує структуру та взаємозв'язки між сутностями.

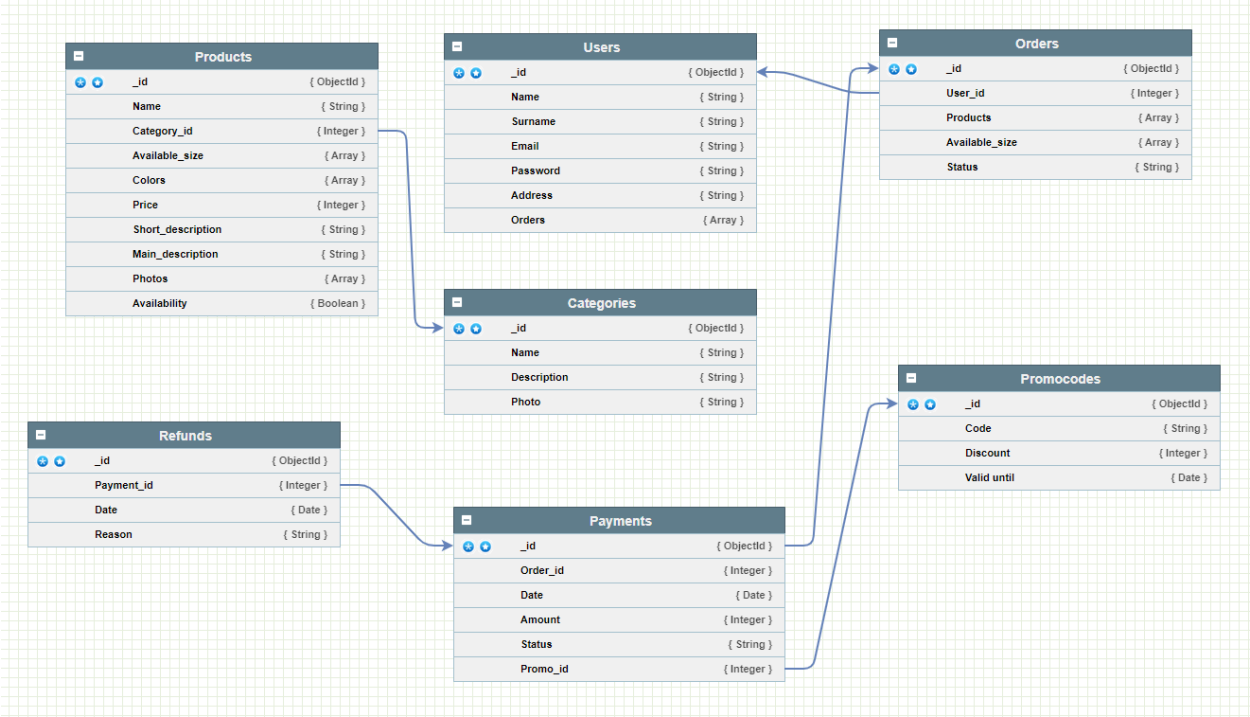


Рисунок 2.2 – Схема бази даних

2.5 Проєктування інтерфейсу

На наступному кроці провели проєктування інтерфейсу шляхом створення макету майбутнього вебзастосунку. Для цих цілей використовувався сервіс Moqups.

На рисунку 2.3 зображено вигляд блоків головної сторінки застосунку. Більша частина блоків головної повинна зображувати користувачу наявний асортимент товарів та різні категорії. Якщо користувач зацікавився товарами, він може зареєструватися та авторизуватися у системі для замовлення.

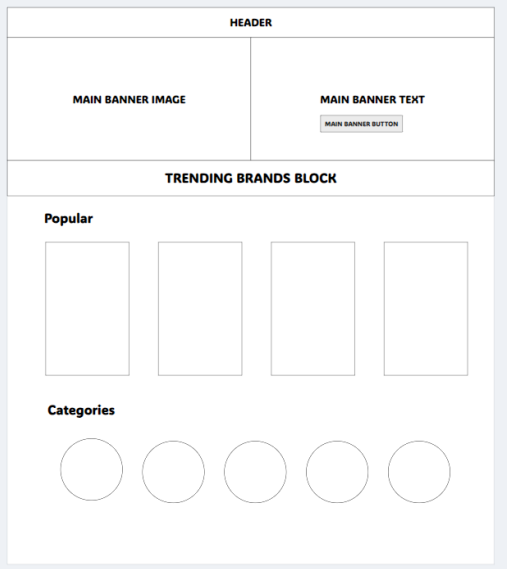


Рисунок 2.3 – Головний екран стартової сторінки

На рисунку 2.4 зображена приблизна схема такої сторінки – вона не повинна бути перевантажена блоками, була ціль розробити її максимально лаконічно та мінімалістично.

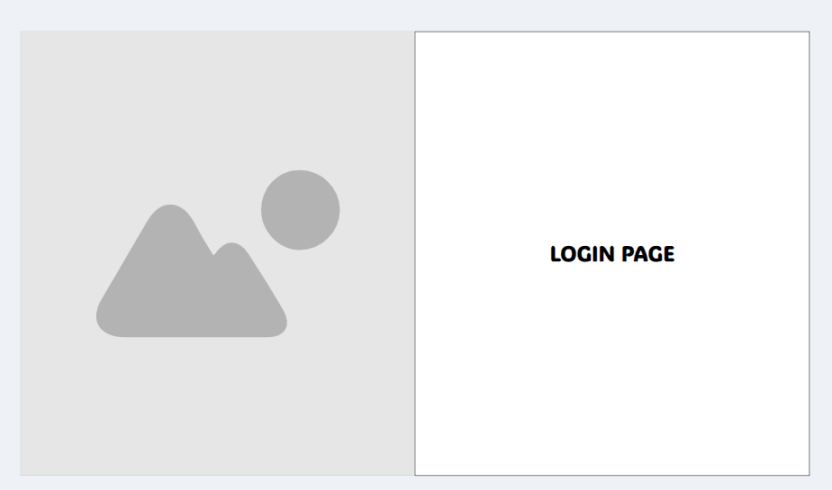


Рисунок 2.4 – Сторінка авторизації

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

В основі програмної реалізації лежить стек, що охоплює такі технології як MySQL, Express, Vue та Node.js. Для розробки інтерфейсу користувача використовувалася технологія Vue, в той час як решта технологій застосовувалася для побудови серверної частини.

3.1 Створення клієнтської частини

Першим етапом було обрано розробку клієнтської частини застосунку за допомогою фреймворку Vue та його компонентів. Під час цього етапу було створено різні види компонентів: картки товарів, форми, кнопки, тощо. Деякі компоненти задля економії часу використовувалися за допомогою сторонніх бібліотек, наприклад, бібліотека Vue Swiper, яка допомагає швидко та зручно створювати різноманітні слайдери.

3.1.1 Дизайн вебзастосунку

Перед створенням застосунку постало завдання про обрання дизайну. На вибір було два варіанти: розробити власноруч або використати існуючі безкоштовні дизайни. Так як, кваліфікаційна робота сконцентрована саме на розробці, то було обрано другий варіант. Для пошуку дизайну та майбутньої верстки став у нагоді сервіс Figma. Figma – це хмарний багатоплатформовий сервіс для дизайнерів інтерфейсів і web-розробників, з яким можна працювати безпосередньо в браузері [9].

На рисунку 3.1 зображено вигляд головного екрану обраного дизайну у застосунку Figma.

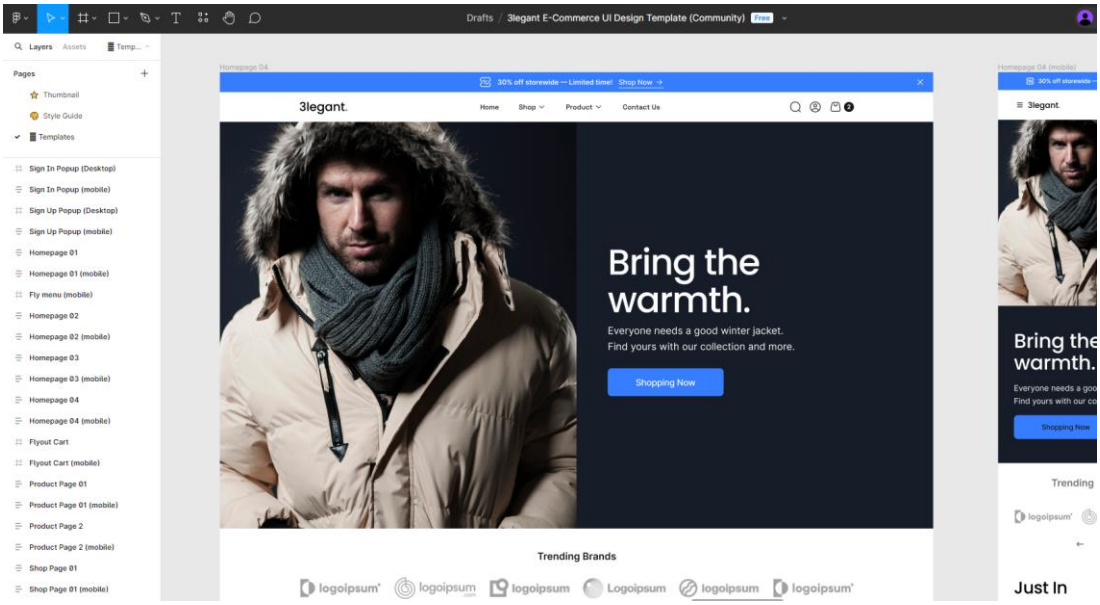


Рисунок 3.1 – Головний екран застосунку у середовищі Figma

Якщо б це був реальний проект створення застосунку, тоді б відходити від узгодженого дизайну було б заборонено, але так як це навчальний проект, то у фінальній верстці є несуттєві зміни.

3.1.2 Формування верстки застосунку

Наступним етапом перейшов до верстки застосунку. Зверстати потрібно було досить багато різних компонентів, шаблонів сторінок. Використовую поняття «шаблони сторінок», тому що, слід пам’ятати, що у розробці на фреймворках не потрібно створювати кожну сторінку окремо – створюються тільки каркаси (шаблони) майбутніх сторінок, а контент вже завантажується динамічно з бази даних.

Верстав одразу з використанням Vue. Верстка на Vue не дуже відрізняється від звичайної верстки на HTML та CSS, однак є свої особливості. Кожна сторінка (view) та кожний компонент (component) має окремий файл під себе. У компоненти виноситься те, що використовується на

сторінках більше одного разу. Це робиться для пришвидшення процесу розробки, щоб не створювати кожного разу один і той самий компонент. Наприклад, компонент картки товару використовується на сайті безліч разів, відповідно, є сенс у створенні окремого компоненту під картку товару. На рисунку 3.2 навів вид декількох компонентів у структурі проекту.

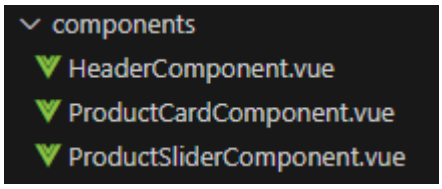


Рисунок 3.2 – Компоненти у структурі проекту

Шаблон будь-якого файлу у Vue містить у собі три необхідних блока – template, style, script [10]. У template прописуємо HTML розмітку майбутньої сторінки або компонента, у style SASS стилі, у script логіку відповідно. На етапі верстки було задіяно тільки блоки template та style. На рисунку 3.3 можна побачити як це виглядає на практиці.

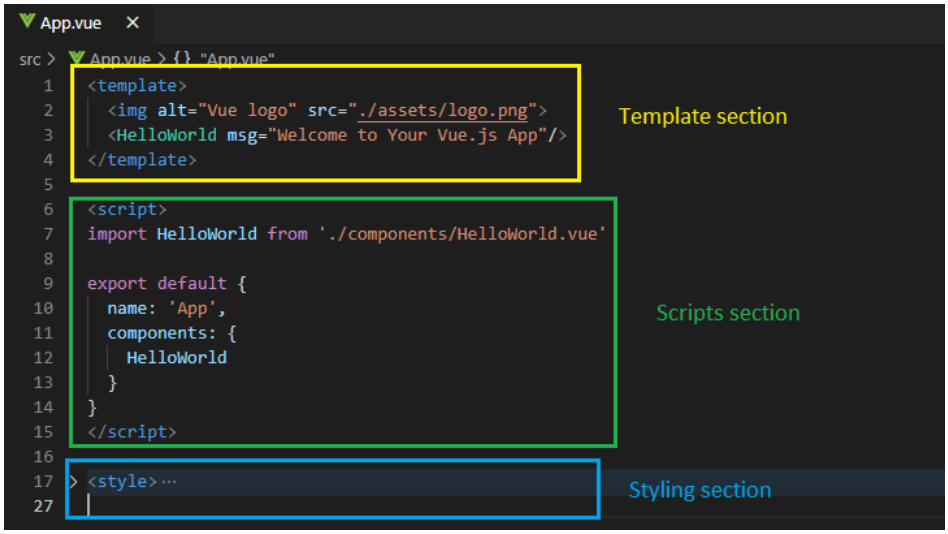


Рисунок 3.3 – Секції (блоки) у Vue файлі [11]

Для більш швидкої та простішої розробки також використовувався Vite. Vite – це сучасний, блискавично швидкий інструмент для створення

скелетів і групування проєктів, який швидко стає популярним завдяки майже миттєвій компіляції коду та швидкій гарячій заміні модулів [12]. Vite допоміг у багатьох ситуаціях, наприклад, для того, щоб всі зміни у коді миттєво відображалися на сторінці.

Слід зауважити, що браузер не сприймає SASS синтаксис, тому необхідно інсталювання додаткових бібліотек, які автоматично компілюють sass до css.

При верстці застосунку намагався зробити його більш «живим» - для цього, наприклад, додавав анімації на різні елементи при наведенні на них. Для цього використовується псевдоклас hover. На рисунку 3.4 можна побачити цікавий ефект при якому при наведенні на логотипи відомих брендів, вони з чорно-білих стають кольоровими. (було наведено на логотип бренду «Tommy Hilfiger»)

Рисунок 3.4 – Hover ефект на логотипи брендів

Після створення макетів основних сторінок не варто нехтувати адаптацією під мобільні пристрої, бо згідно останніх досліджень, більше 67% користувачів заходять в інтернет саме з мобільного телефону, при цьому ще 5 років тому цей відсоток був меншим за 50% та продовжує збільшуватися [13]. Для цього у CSS використовуються медіа-запити. Медіа запити (media queries) – це правила CSS, що дозволяють керувати стилями елементів залежно від значень технічних параметрів (висота та ширина браузера, роздільна здатність сторінки, типу пристрою, його орієнтації, щільності пікселів тощо) [14]. На рисунку 3.5 можна побачити вигляд мобільної версії головного екрану застосунку.

Рисунок 3.5 – Мобільна версія головного екрану

3.1.3 Логіка застосунку

Для програмування логіки застосунку використовувався фреймворк Vue. Цей фреймворк дозволяє створювати користувацькі інтерфейси за допомогою маршрутизації, плагінів, вбудованих компонентів та методів. Тобто значно пришвидшує розробку, для чого він і створювався.

Для того, щоб краще зрозуміти як працює Vue нижче наведено частину коду слайдеру товарів, який виводить товари у слайдер з бази даних.

```
<template>
  <div class="product-slider">
    <div class="product-slider-title">
      Just In
    </div>
    <swiper :modules="modules" :pagination="{ clickable: true }"
:spaceBetween="5" :slidesPerView="5" :slidesPerGroup="5">
      <swiper-slide v-for="product in products" :key="product.id">
        <product-card :product="product"></product-card>
      </swiper-slide>
    </swiper>
  </div>
</template>
```

До шостої строки бачимо звичайну HTML розмітку. У шостій строчці бачимо елемент слайдеру, який працює за допомогою бібліотеки Swiper. У нього задано конкретні параметри – доступність кнопок пагінації, кількість видимих слайдів, відстань між слайдами (картками товарів). Потім використовується вбудована директива Vue v-for, яка циклом проходиться по всім товарам у об’єкті products та виводить їх усіх.

На рисунку 3.6 можна побачити, як це виглядає у застосунку.

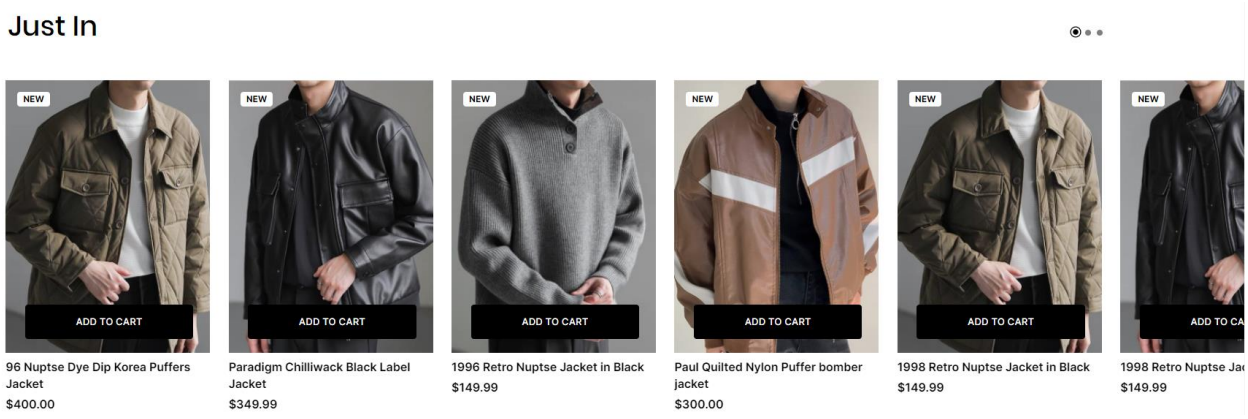


Рисунок 3.6 – Слайдер товарів головної сторінки

Працюючи з SPA додатками слід пропрацювати та зрозуміти роутинг додатку – для цього використовується Vue Router. Vue Router – це офіційна бібліотека від розробників Vue, яка дозволяє зручно налаштувати маршрутизацію у додатку. За допомогою цієї технології перехід між сторінками проходить без перезавантаження сторінки. Нижче можна побачити приклад коду, який налаштовує маршрутизацію для головної сторінки та сторінки товарів. Налаштування не складне – повідомляємо роутеру необхідні шляхи та сторінки, які повинні з’являтися при переході на них.

```
const router = createRouter({
 history: createWebHistory(import.meta.env.BASE_URL),
 routes: [
  {
   path: '/',
   name: 'home',
   component: HomeView
  },
  {
   path: '/shop',
   name: 'shop',
   component: ShopView
  },
```

Далі можна побачити, як це виглядає на сторінці. У цьому випадку у шапку сайту було додане посилання через router-link на необхідну сторінку.

```
<li class="header-list-link"><router-link to="/shop">Shop</router-link></li>
```

Важливою частиною фронтенду інтернет-магазину також є створення фільтрів та сортування товарів, тому що це значно підвищує користувацький досвід. Для цього використовувалися конструкції, методи, що за замовчуванням вбудовані у Vue та JS. Розглянемо, наприклад, сортування

товарів – за збільшенням та зменшенням ціни, популярністю та новизною. Для цього використовувався такий код:

```
return products.sort((a, b) => {
        if (this.sortOption === 'price-low-to-high') {
          return a.price - b.price;
        } else if (this.sortOption === 'price-high-to-low') {
          return b.price - a.price;
        } else if (this.sortOption === 'newest') {
          return new Date(b.createdAt) - new Date(a.createdAt);
        } else {
          return a.popularity - b.popularity;
        }
      }).slice(0, this.currentPage * this.itemsPerPage);
```

Цей фрагмент коду сортує масив products відповідно до вибраного варіанта сортування (sortOption). Якщо обраний варіант сортування - ціна від нижчої до вищої, товари сортуються за зростанням ціни. Якщо ціна від вищої до нижчої - за спаданням ціни. Якщо вибрано новизну - за датою створення (найновіші спочатку). Якщо обрано популярність - за значенням популярності (від меншої до більшої). Після сортування, метод slice обрізає масив до розміру, відповідного поточній сторінці (currentPage), обмежуючи кількість відображених товарів до itemsPerPage.

3.1.4 Валідація та обробка помилок

При будуванні форм застосунку обов’язковим етапом також було впровадження валідації полів. Валідація – це процес перевірки значень за певним раніше затвердженим стандартом, вимогою, правилом [15]. Тобто,

для того, щоб на сервер заходила тільки коректна інформація необхідно використовувати різні методи валідації.

Валідацію на Vue можно організувати багатьма варіантами. Найпопулярнішими є додавання сторонніх бібліотек або створення валідації кодом власноруч. У цьому проекті було обрано другий варіант, який довший в реалізації, але за допомогою нього можна значно підвищити експертизу.

Нижче наведено приклад коду, який перевіряє чи всі поля форми реєстрації було заповнено. Важливо передати на сервер максимально повну інформацію для подальшої роботи з нею. На рисунку 3.7 можна побачити як це працює у застосунку.

```
if (!this.name || !this.username || !this.email || !this.password) {
  this.errorMessage = 'All fields are required.';
  return;
}
```

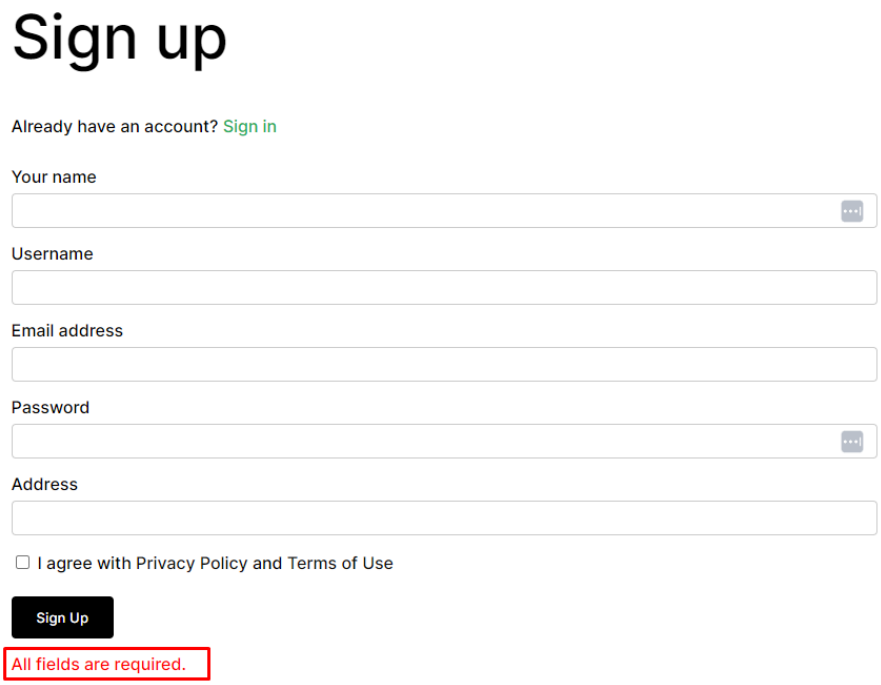


Рисунок 3.7 – Приклад валідації форми реєстрації

3.2 Створення серверної частини

Основу сервера складають технології MySQL, Express та Node.js, які забезпечують високу продуктивність та гнучкість при розробці сучасних вебзастосунків.

3.2.1 Створення моделей таблиць

Опишемо процес створення моделей для взаємодії з базою даних за допомогою ORM-бібліотеки Sequelize. Моделі визначають структуру таблиць у базі даних і встановлюють правила для полів, які вони містять. Перш за все необхідно було створити модель головної таблиці товарів. Для цього було використано такий код:

```
module.exports = (sequelize, DataTypes) => {
   const Products = sequelize.define("Products", {
     name: {
       type: DataTypes.STRING,
       allowNull: false
     },
     category_id: {
       type: DataTypes.INTEGER,
       allowNull: false
     },
     price: {
       type: DataTypes.INTEGER,
       allowNull: false
     },
     short_description: {
```

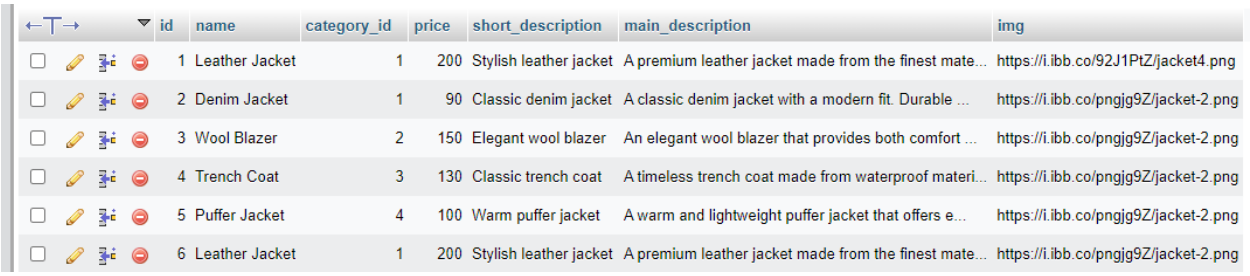
```
      type: DataTypes.STRING,
      allowNull: false
    },
    main_description: {
      type: DataTypes.STRING,
      allowNull: false
    },
    img: {
      type: DataTypes.STRING,
      allowNull: false
    }
  }, {
    timestamps: false
  });

  return Products;
};
```

У підсумку, за допомогою Sequelize визначили модель, яка описує структуру таблиці в базі даних. Модель Products містить всі необхідні поля для зберігання інформації про продукти, включаючи їх назви, категорії, ціни, описи та зображення.

3.2.2 Запуск локального серверу з базою даних

Наступним етапом необхідно створити підключення до бази даних MySQL. Для цього необхідно було запустити сервер за базою даних – для таких завдань можна використовувати застосунок Open Server Panel. Для створення таблиць та заповнення базовою інформацією став у нагоді

phpMyAdmin. phpMyAdmin - це безкоштовний програмний інструмент, написаний на PHP, який призначений для адміністрування сервера баз даних MySQL або MariaDB [16]. На рисунку 3.8 можна побачити таблицю сутності товарів з тестовими даними.



id	name	category_id	price	short_description	main_description	img
1	Leather Jacket	1	200	Stylish leather jacket	A premium leather jacket made from the finest mate...	https://i.ibb.co/92J1PtZ/jacket4.png
2	Denim Jacket	1	90	Classic denim jacket	A classic denim jacket with a modern fit. Durable ...	https://i.ibb.co/pngjg9Z/jacket-2.png
3	Wool Blazer	2	150	Elegant wool blazer	An elegant wool blazer that provides both comfort ...	https://i.ibb.co/pngjg9Z/jacket-2.png
4	Trench Coat	3	130	Classic trench coat	A timeless trench coat made from waterproof materi...	https://i.ibb.co/pngjg9Z/jacket-2.png
5	Puffer Jacket	4	100	Warm puffer jacket	A warm and lightweight puffer jacket that offers e...	https://i.ibb.co/pngjg9Z/jacket-2.png
6	Leather Jacket	1	200	Stylish leather jacket	A premium leather jacket made from the finest mate...	https://i.ibb.co/pngjg9Z/jacket-2.png

Рисунок 3.8 – Таблиця товарів у phpMyAdmin

3.2.3 Створення серверу Express

Створення базового сервера в Express включає кілька ключових етапів: ініціалізація Express-додатка, налаштування маршрутів, що обслуговують запити клієнта, та запуск сервера на певному порті. Також, для більш зручного створення застосунку була додана бібліотека Sequelize.

Sequelize – це інструмент для ефективної роботи з базами даних у додатках Node.js. У контексті розробки, він діє як бібліотека Object-Relational Mapping (ORM), полегшуючи взаємодію з базами даних. Sequelize надає абстракцію даних у вигляді об’єктів, позбавляючи розробників від прямої роботи з SQL-запитами [17]. db.sequelize.sync() синхронізує всі моделі з базою даних, створюючи або оновлюючи таблиці відповідно до визначень моделей. Після успішної синхронізації запускається сервер Express на порті 3001, який виводить повідомлення "Server running on port 3001".

```
const express = require ("express");
const app = express();
const cors = require('cors');

app.use(express.json());
app.use(cors());

const db = require("./models");

db.sequelize.sync().then(() => {
   app.listen (3001, () => {
      console.log("Server running on port 3001");
   });
});
```

3.2.4 Створення серверної маршрутизації

Для налаштування маршрутизації використовується фреймворк Express.js. Створюється маршрутизатор, який обробляє різні маршрути та виконує відповідні функції для кожного з них. У цьому прикладі показано, як налаштувати маршрут для отримання списку продуктів з бази даних.

```
router.get("/", async (req, res) => {
   const listOfProducts = await Products.findAll();
   res.json(listOfProducts);
});
```

Цей маршрутизатор оголошує маршрут GET /, який обробляє запити на отримання списку всіх продуктів. Використовується асинхронна функція для

взаємодії з базою даних за допомогою методу findAll з моделі Products. Отримані дані відправляються у відповідь на запит у форматі JSON. Така маршрутизація була створена під усі необхідні дії. У підсумку як це виглядає на сайті можна побачити на рисунку 3.9. Повний лістинг коду цього компоненту можна побачити на додатку Б.

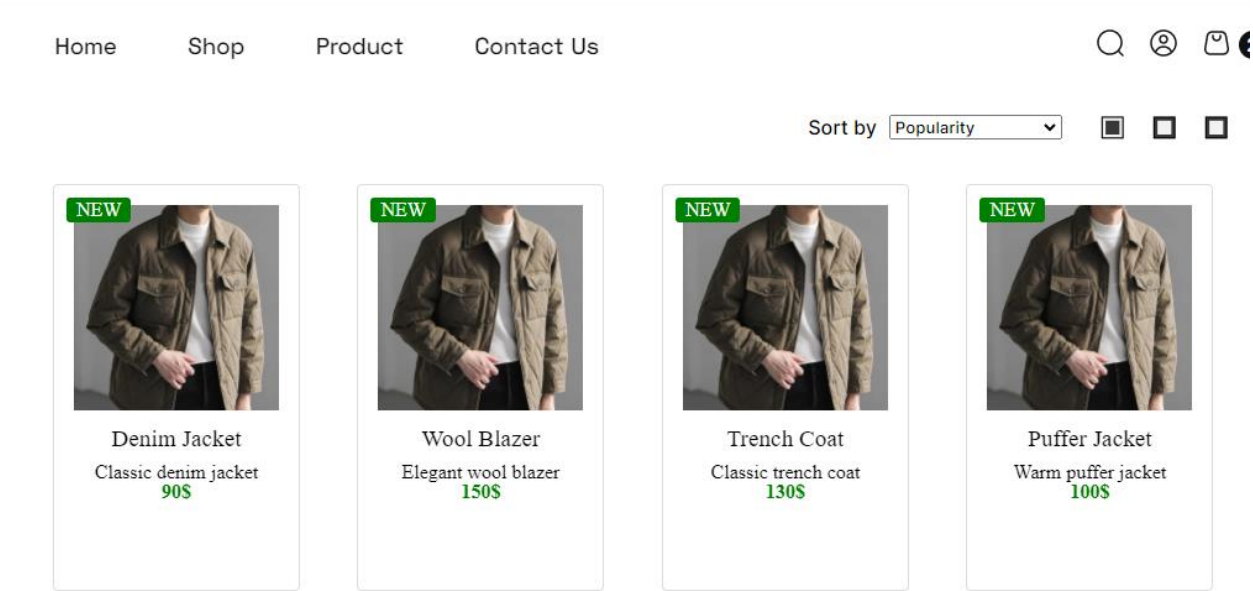


Рисунок 3.9 – Відображення усіх товарів

Також, була розроблена маршрутизація та запити для роботи з аккаунтами користувачі, наприклад, вхід у аккаунт або зміна даних аккаунту. Можемо побачити частину коду, яка повертає інформацію про користувача з бази даних при запиті.

```
router.get('/', authenticateToken, async (req, res) => {
  try {
    const user = await User.findByPk(req.user.id);
    res.json(user);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

Цей код створює захищений маршрут на Express.js, який вимагає автентифікації токеном для доступу до даних користувача. Коли користувач надсилає GET запит на кореневий шлях '/', функція authenticateToken перевіряє валідність токена. Якщо токен валідний, маршрут використовує отримане ID користувача для пошуку його в базі даних через ORM і повертає інформацію про користувача у форматі JSON. У разі помилки сервер відправляє відповідь з кодом 500 та повідомленням про помилку. Повний лістинг маршрутизації для роботи з користувачем викладено в додатку А.

ВИСНОВКИ

У дипломній роботі було розглянуто розробку вебзастосунку для інтернет-магазину одягу з використанням Vue.js, MySQL і Express. Було проведено детальний аналіз вимог до сучасних інформаційних систем для електронної комерції та визначено необхідні функціональність та особливості додатку. На основі цього було сформульоване технічне завдання та встановлено технічні вимоги до розробки.

Під час виконання дипломної роботи було зроблено:

1. Дослідження поняття веброзробки та засобів для створення сучасних вебзастосунків. Було вивчено основні концепції веброзробки та інструменти, що використовуються для створення інтерактивних і функціональних вебзастосунків, зокрема, Vue.js для фронтенду, Express для бекенду та MySQL для зберігання даних.

2. Визначено архітектуру додатку та його компоненти. Було спроектовано архітектуру додатку, яка включає фронтенд на Vue.js, бекенд на Express і базу даних MySQL. Було визначено ключові компоненти системи та їх взаємодію.

3. Реалізовано серверну логіку з використанням Express. Серверна частина була реалізована з використанням Express, яка надає API для обробки запитів на додавання продуктів, отримання списку продуктів, оформлення замовлень тощо. Для взаємодії з базою даних MySQL було використано ORM-бібліотеку Sequelize.

4. Розроблено клієнтський інтерфейс з використанням Vue.js. Було створено інтерактивний інтерфейс користувача для інтернет-магазину з використанням Vue.js. Інтерфейс включає сторінки каталогу товарів, деталізації продуктів, кошика покупок та оформлення замовлення.

5. Забезпечено комунікацію між клієнтом та сервером за допомогою HTTP-запитів. Комунікація між клієнтом і сервером здійснювалася за допомогою HTTP-запитів. Фронтенд надсилав запити на

сервер для отримання та відправки даних, таких як список продуктів, деталі продукту та оформлення замовлення.

6. Виконано тестування додатку. Було проведено тестування додатку для перевірки його функціональності, виявлення помилок та забезпечення якості програмного забезпечення. Тестування включало перевірку роботи API, інтерфейсу користувача та взаємодії між фронтендом і бекендом.

Розроблений вебзастосунок для інтернет-магазину одягу відповідає сформованому технічному завданню та технічним вимогам. Було реалізовано повний цикл розробки від визначення архітектури до тестування і забезпечення якості програмного забезпечення. Використання сучасних технологій, таких як Vue.js, Express і MySQL, дозволило створити функціональний і зручний у використанні вебзастосунок.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основи Web UI розробки (проєкт «Ти можеш усе! Можливості безмежні!»). Prometheus. URL: https://prometheus.org.ua/course/course-v1:LITS+114+YCE (дата звернення: 20.05.2024).

2. Що таке фреймворк: пояснюємо простими словами. brainlab.com.ua. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prostymy-slovamy (дата звернення: 20.05.2024).

3. Вступ | Vue.js. Vue.js - Прогресивний JavaScript фреймворк | Vue.js. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення: 20.05.2024).

4. Легше, швидше, масштабніше. Чому топові компанії використовують Node.js. Projector – Creative & Tech Online Institute. URL: https://prjctr.com/mag/whynode (дата звернення: 20.05.2024).

5. Express - це фреймворк для веб-застосунків, побудованих на Node.js. Express - Node.js web application framework. URL: https://expressjs.com/uk/ (дата звернення: 20.05.2024).

6. FREEhost.UA. Хостинг в Украине – купить украинский хостинг сайтов от провайдера FreeHost. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-mysql/ (дата звернення: 20.05.2024).

7. Maxym Z. Варіанти використання та сценарії (Use Cases and Scenarios). Maxym Zosym. URL: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 20.05.2024).

8. Основні поняття реляційних БД: нормалізація, зв'язок та ключі ⋆ ДПА и ЗНО онлайн. ДПА и ЗНО онлайн. URL: https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/ (дата звернення: 20.05.2024).

9. Омельчук Є. Що таке Figma: функції, інструменти та переваги - академія Wezom. Академия Wezom - Обучаем IT технологиям с нуля. URL:

https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/ (дата звернення: 20.05.2024).

10. Understanding Application Structure Of Vue.js. C# Corner - Community of Software and Data Developers. URL: https://www.c-sharpcorner.com/article/understanding-application-structure-of-vue-js/ (дата звернення: 20.05.2024).

11. Vela E. P. Getting Started with Fundamental Library Styles in Vue Project. Medium. URL: https://medium.com/fundamental-library/getting-started-with-fundamental-library-styles-in-vue-project-6a344ff4d16d (дата звернення: 20.05.2024).

12. Vite – наступник WebPack?. https://medium.com/. URL: https://medium.com/@jstify.community/vite-наступник-webpack-9f2fdc847c79 (дата звернення: 20.05.2024).

13. Mobile First: мобільна версія сайту. Як створити, особливості, дизайн, функціонал. Захищена сторінка. URL: https://www.site2b.ua/ua/web-blog-ua/princip-mobile-first-dlya-rozrobki-dizajnu-sajtiv.html (дата звернення: 20.05.2024).

14. FREEhost.UA. Хостинг в Украине – купить украинский хостинг сайтов от провайдера FreeHost. URL: https://freehost.com.ua/ukr/faq/articles/chto-takoe-media-zaprosi-css-i-dlja-chego-oni-nuzhni/ (дата звернення: 20.05.2024).

15. Валідація. Тестування валідації. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/validation-testing/ (дата звернення: 20.05.2024).

16. Вступ – phpMyAdmin 6.0.0-dev documentation. Welcome to phpMyAdmin’s documentation! – phpMyAdmin 5.1.4 documentation. URL: https://docs.phpmyadmin.net/uk/latest/intro.html (дата звернення: 20.05.2024).

17. Sequelize що це за фреймворк і його основні характеристики. FoxmindEd. URL: https://foxminded.ua/sequelize-shcho-tse/ (дата звернення: 20.05.2024).

ДОДАТОК А

Серверна маршрутизація для роботи з користувачами

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { User } = require('../models');
const router = express.Router();

const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

  if (token == null) return res.sendStatus(401);

  jwt.verify(token, 'key', (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
};

router.get('/', authenticateToken, async (req, res) => {
  try {
    const user = await User.findByPk(req.user.id);
    res.json(user);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
```

```
});

router.put('/', authenticateToken, async (req, res) => {
  const { name, email, oldPassword, newPassword } = req.body;

  try {
    const user = await User.findByPk(req.user.id);

    if (oldPassword && newPassword) {
      const isMatch = await bcrypt.compare(oldPassword, user.password);
      if (!isMatch) {
        return res.status(400).json({ error: 'Old password is incorrect' });
      }
      const hashedPassword = await bcrypt.hash(newPassword, 10);
      user.password = hashedPassword;
    }

    user.name = name || user.name;
    user.email = email || user.email;

    await user.save();
    res.json({ message: 'Account updated successfully!' });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

module.exports = router;
```

ДОДАТОК Б

Компонент виводу товарів з бази даних

```
<template>
  <div class="container">
    <div class="sidebar">
      <div class="filter">
        <h3>Filter</h3>
        <div class="categories">
          <h4>CATEGORIES</h4>
          <ul>
            <li @click="filterByCategory('all')" :key="'all'">
              <input type="radio" name="category" /> All Clothes
            </li>
            <li
              v-for="category in categories"
              :key="category.id"
              @click="filterByCategory(category.id)"
            >
              <input type="radio" name="category" /> {{ category.name }}
            </li>
          </ul>
        </div>
        <div class="price">
          <h4>PRICE</h4>
          <ul>
            <li
              v-for="range in priceRanges"
              :key="range"
```

```
          @click="filterByPrice(range)"
        >
          <input type="checkbox" /> {{ range }}
        </li>
      </ul>
    </div>
  </div>
</div>
<div class="main">
  <div class="header">
    <h1>Catalog</h1>
    <div class="sort">
      <span>Sort by</span>
      <select v-model="sortOption" @change="sortProducts">
        <option value="price-low-to-high">Price: Low to High</option>
        <option value="price-high-to-low">Price: High to Low</option>
      </select>
      <div class="view-icons">
      </div>
    </div>
  </div>
  <div class="product-list">
    <div
      class="product"
      v-for="product in filteredProducts"
      :key="product.id"
    >
      <img :src="product.img" :alt="product.name" />
      <div class="product-info">
        <span class="badge">NEW</span>
```

```
            <h2>{{ product.name }}</h2>
            <p>{{ product.short_description }}</p>
            <p class="price">{{ product.price }}$</p>
            <button @click="addToCart(product)" class="add-to-cart-btn">
              Add to cart
            </button>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
import axios from "axios";

export default {
  data() {
    return {
      products: [],
      categories: [],
      priceRanges: [
        "$0.00 - $99.99",
        "$100.00 - $199.99",
        "$200.00 - $299.99",
        "$300.00 - $399.99",
        "$400.00+",
      ],
      selectedCategory: "all",
      selectedPriceRange: [],
```

```
    sortOption: "price-low-to-high",
    currentPage: 1,
    itemsPerPage: 10,
  };
},
computed: {
  filteredProducts() {
    let products = this.products;

    if (this.selectedCategory !== "all") {
      products = products.filter(
        (product) => product.category_id === this.selectedCategory
      );
    }

    if (this.selectedPriceRange.length > 0) {
      products = products.filter((product) => {
        return this.selectedPriceRange.some((range) => {
          const [min, max] = range.replace("$", "").split(" - ").map(Number);
          return product.price >= min && (max ? product.price <= max : true);
        });
      });
    }

    return products
      .sort((a, b) => {
        if (this.sortOption === "price-low-to-high") {
          return a.price - b.price;
        } else if (this.sortOption === "price-high-to-low") {
          return b.price - a.price;
```

```
        } else if (this.sortOption === "newest") {
          return new Date(b.createdAt) - new Date(a.createdAt);
        } else {
          return a.popularity - b.popularity;
        }
      })
      .slice(0, this.currentPage * this.itemsPerPage);
  },
},
mounted() {
  this.loadCategories();
  this.loadProducts();
},
methods: {
  async loadCategories() {
    try {
      const response = await axios.get("http://localhost:3001/category");
      this.categories = response.data;
    } catch (error) {
      console.error("Error loading categories:", error);
    }
  },
  async loadProducts() {
    try {
      const response = await axios.get("http://localhost:3001/shop");
      this.products = response.data;
    } catch (error) {
      console.error("Error loading products:", error);
    }
  },
```

```
filterByCategory(categoryId) {
  this.selectedCategory = categoryId;
},
filterByPrice(range) {
  if (this.selectedPriceRange.includes(range)) {
    this.selectedPriceRange = this.selectedPriceRange.filter(
      (r) => r !== range
    );
  } else {
    this.selectedPriceRange.push(range);
  }
},
sortProducts() {},
addToCart(product) {
  const token = localStorage.getItem("token");
  if (!token) {
    alert("Please log in to add items to the cart.");
    this.$router.push('/signin');
    return;
  }

  const cart = JSON.parse(localStorage.getItem("cart")) || [];
  const existingProduct = cart.find(item => item.id === product.id);

  if (existingProduct) {
    existingProduct.quantity += 1;
  } else {
    cart.push({ ...product, quantity: 1 });
  }
```

```
      localStorage.setItem("cart", JSON.stringify(cart));
      alert("Product added to cart!");
    }
  },
};
</script>
```

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничого фахового коледжу ЗНУ»

Я, Євсеєв Данило Андрійович, здобувач освіти 4-го курсу, спеціальності «Інженерія програмного забезпечення», освітньо-професійної програми «Розробка програмного забезпечення», групи К121-20, адреса електронної пошти danilevseev1245@gmail.com:

- підтверджую, що написана мною дипломна робота на тему «Розробка інтернет-магазину одягу на базі технологій JS, MySQL та препроцесору SASS» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 15.06.2024 Підпис  Данило ЄВСЕЄВ

Дата 17.06.2024 Підпис  Анна НЕЛАСА