

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ З ПРОДАЖУ
КРОСІВОК

SOFTWARE IMPLEMENTATION OF AN ONLINE STORE SELLING
SNEAKERS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-221
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

ПОЛТАВСЬКИЙ В. А.
(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.
(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПОЛТАВСЬКОГО Владислава Андрійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інтернет-магазину з продажу кросівок. Software Implementation of an Online Store Selling Sneakers

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Предметна область та постановка задачі. 2. Огляд засобів реалізації. 3. Програмне та технічне забезпечення. 4. Тестування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 14 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Предметна область та постановка задачі.	1 тиждень	Розділ 1
3	Огляд засобів реалізації.	2 тиждень	Розділ 2
4	Програмне та технічне забезпечення.	3-4 тижні	Розділ 3
5	Тестування системи.	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

_____ Владислав ПОЛТАВСЬКИЙ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Євгеній ГОФМАН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 63 с., 4 табл., 22 рис., 2 дод., 13 джерел.

HTML, CSS, JAVASCRIPT, REACT, REDUX TOOLKIT, NODEJS, NESTJS, MYSQL, SEQUELIZE.

Об'єкт дослідження – процес створення програмного забезпечення для онлайн-бізнесу.

Предмет дослідження – методи створення програмного забезпечення для онлайн-бізнесу.

Мета роботи – розробка інтернет-магазину з продажу кросівок.

У ході роботи реалізовано клієнтську частину з можливістю перегляду, фільтрації товарів, додавання до кошика та оформлення замовлень. На серверній стороні створено REST API, що обробляє запити, керує користувачами, товарами та замовленнями. Використано NestJS для побудови серверної логіки, MySQL – як систему управління базами даних, Sequelize – як ORM.

Для захисту передавання даних впроваджено технологію JWT, а для безпечного зберігання паролів – бібліотеку Bcrypt. API протестовано за допомогою Postman.

Розробка здійснювалася із застосуванням сучасного стеку технологій, що забезпечує масштабованість, зручність у розробці та безпеку.

Результат – функціональний вебзастосунок, який дозволяє користувачам взаємодіяти з каталогом товарів, оформляти замовлення, проходити авторизацію та реєстрацію.

Висновок – створено інтернет-магазин з повноцінною логікою роботи, зручним інтерфейсом та надійною системою обробки і захисту даних користувачів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 63 pages, 4 tables, 22 figures., 2 appendixes, 13 sources.

HTML, CSS, JAVASCRIPT, REACT, REDUX TOOLKIT, NODEJS, NESTJS, MYSQL, SEQUELZIE.

Object of research – the process of developing software for online business.

Purpose of the work – the development of an online sneaker store.

During the project, a client-side interface was implemented, allowing users to browse products, filter them by category, add items to a cart, and place orders. On the server side, a REST API was created to handle requests and manage users, products, and orders. NestJS was used for building the backend logic, MySQL served as the database management system, and Sequelize was used as the ORM.

To ensure secure data transmission, JWT technology was implemented. For secure password storage, the Bcrypt library was used. The API was tested using Postman.

The development was carried out using a modern technology stack, ensuring scalability, development efficiency, and data security.

Result – a functional web application that enables users to interact with a product catalog, place orders, and perform authorization and registration.

Conclusion – an online store has been created with complete business logic, a user-friendly interface, and a reliable system for handling and securing user data.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Предметна область та постановка задачі.....	10
1.1 Поняття електронної комерції	10
1.2 Види електронної комерції	11
1.3 Поняття інтернет магазину	12
1.4 Класифікація інтернет-магазинів	13
1.5 Прямі конкуренти в ніші продажу кросівок.....	14
1.6 Висновок	14
2 Огляд засобів реалізації.....	15
2.1 Інструменти для розробки клієнтської частини.....	15
2.2 React для створення клієнтської частини інтернет-магазину.....	17
2.3 Angular для створення клієнтської частини інтернет-магазину.....	19
2.4 Vue.js для створення клієнтської частини інтернет-магазину	20
2.5 Технології для розробки клієнтської частини.....	22
2.6 Інструменти для розробки серверної частини	25
2.7 Express.js для створення серверної частини інтернет-магазину	26
2.8 Koa.js для створення серверної частини інтернет-магазину	27
2.9 Nest.js для створення серверної частини інтернет-магазину.....	28
2.10 Технології для розробки серверної частини інтернет-магазину	31
2.11 Висновок	35
3 Програмне та технічне забезпечення	36
3.1 Засоби реалізації.....	36
3.2 Приклади використання додатку.....	39
4 Тестування системи	43
4.1 Інструменти для тестування системи.....	43
Висновки	45
Перелік джерел посилання	46

Додаток А Текст програми.....	48
Додаток Б Слайди презентації	57

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	– Artificial Intelligence;
API	– Application Programming Interface;
CI	– Continuous Integration;
CLI	– Command line interface;
CRUD	– Create, Read, Update, Delete;
HTTP	– HyperText Transfer Protocol;
IDE	– Integrated Development Environment;
JSON	– JavaScript Object Notation;
JWT	– JSON Web Token;
ML	– Machine Learning;
ORM	– Object-Relational Mapping;
SQL	– Structured Query Language;
UI	– User Interface;
URL	– Uniform Resource Locator;
БД	– база даних;
ЦАП	– цифро-аналоговий перетворювач.

ВСТУП

Інтернет став важливою частиною нашого життя, проникаючи в усі сфери діяльності, змінюючи підходи до бізнесу та пропонуючи нові можливості для розвитку підприємництва. Інформаційні технології стрімко розвиваються, і для підприємців створення інтернет-магазинів стає однією з головних задач. Інтернет-магазини є зручним інструментом для здійснення покупок, оскільки вони дозволяють заощаджувати час, отримувати доступ до широкого асортименту товарів та забезпечують зручні методи оплати і доставки. Зокрема, популярність інтернет-магазинів значно зросла після карантинних обмежень 2020-2021 років.

Одним із напрямків електронної комерції є створення інтернет-магазинів для продажу одягу та взуття, зокрема, кросівок. Кросівки є універсальним товаром, який користується попитом серед різних категорій населення. Інтернет-магазини дозволяють не обмежувати асортимент площею торгового приміщення, що дає можливість запропонувати клієнтам значно більший вибір моделей, розмірів та варіантів.

Розробка інтернет-магазину кросівок потребує врахування специфіки товару та вимог користувачів. Важливими аспектами є забезпечення зручного користувацького інтерфейсу, швидкість оформлення замовлень, безпека платіжних систем, а також надійність роботи сайту, що включає в себе технічні рішення для обробки високих навантажень та безперебійної роботи.

Метою цієї дипломної роботи є розробка інтернет-магазину для продажу кросівок, який відповідатиме сучасним вимогам бізнесу та технічним стандартам. В рамках роботи буде проведено аналіз ринку електронної комерції, визначено основні вимоги до функціоналу інтернет-магазину, а також обрано оптимальні інструменти для розробки і програмної реалізації проєкту.

1 ПРЕДМЕТНА ОБЛАСТЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття електронної комерції

Електронна комерція (E-commerce) – це сфера економіки, в якій просування продуктів, торгові транзакції та інші бізнес-процеси виконуються безпосередньо в інтернеті [1]. Цей термін охоплює різні моделі та практики, що дають змогу виконувати комерційні операції онлайн. Такий формат бізнесу має як свої переваги, так і недоліки. Перевагами електронної комерції для бізнесу є:

- зростає якість надання послуг;
- більш ефективні та прості бізнес-процеси;
- здатність охопити аудиторію у ширшому масштабі;
- ефективна реклама в інтернеті;
- можливість працювати з меншою кількістю персоналу;
- відпадає необхідність витрат на приміщення для торгівлі;
- зменшується кількість зайвих дій та затримок.

Перевагами електронної комерції для клієнта є:

- можливість здійснити покупку з будь-якого місця цілодобово;
- широкий асортимент товарів;
- можливість порівняння товарів і цін у різних магазинах;
- можливість прочитати відгуки інших людей про товар;
- можливість залишити свій коментар;
- доступна і детальна інформація про товар.

Недоліками електронної комерції є:

- велика конкуренція;
- необхідність витрачати значні ресурси на рекламу;
- недовіра до інтернет-покупок з боку деяких споживачів;
- необхідність мати знання про те, як працювати в інтернеті;
- необхідність мати налагоджений механізм доставки.

1.2 Види електронної комерції

Зазвичай виділяють шість основних типів електронної комерції [2], основною відмінністю яких є взаємодіючі сторони (бізнес, споживач):

- B2B (Business to Business): між підприємствами, де компанії продають товари чи послуги іншим компаніям, наприклад, постачання сировини або корпоративного програмного забезпечення (приклади: Microsoft, IBM);
- B2C (Business to Consumer): бізнес продає безпосередньо кінцевому споживачу, що є найбільш поширеним типом (приклади: Rozetka, Amazon);
- C2C (Consumer to Consumer): споживачі продають товари або послуги іншим споживачам (приклади: OLX, eBay);
- C2B (Consumer to Business): споживачі продають свої послуги бізнесу (приклади: Upwork, Freelancehunt);
- B2A або B2G (Business to Administration / Government): бізнес надає послуги урядовим органам (приклади: Microsoft, IBM).

Всі ці види електронної комерції представлені на рис. 1.1 [3].

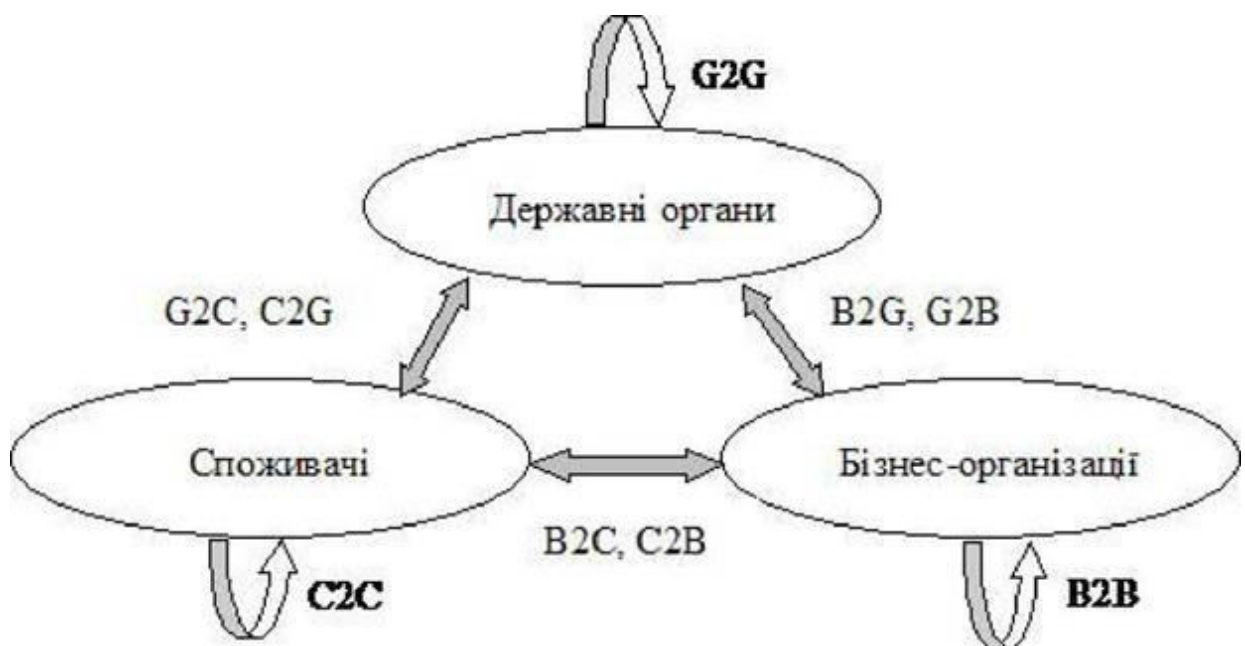


Рисунок 1.1 – Види електронної комерції [3]

1.3 Поняття інтернет магазину

Інтернет-магазин – це вебсайт, який дозволяє продавати товари або послуги споживачам безпосередньо через інтернет. Така модель є однією з головних складових електронної комерції. Інтернет-магазини надають можливість споживачам обирати товари за різними критеріями, оформляти замовлення та обирати спосіб оплати і доставки.

Основні характеристики інтернет-магазину:

- каталог товарів: включає інформацію про товари (описи, фотографії, ціни);
- пошук та фільтри: дозволяють користувачам швидко знаходити потрібний товар, обираючи за різними параметрами;
- кошик покупок: дає змогу користувачу зберігати товари для подальшого оформлення замовлення;
- процес оформлення замовлення: вибір способу доставки і оплати, підтвердження товарів у кошику;
- методи оплати: можуть включати банківські перекази, післяплату або онлайн оплати через картки або електронні гаманці.

Схема, як влаштований інтернет-магазин представлено на рис. 1.2 [3].

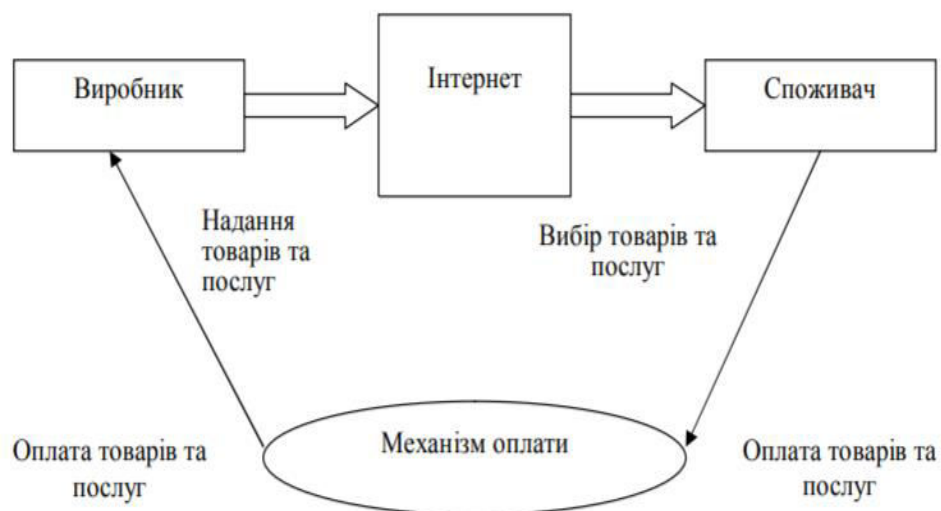


Рисунок 1.2 – Принцип роботи інтернет-магазину [3]

1.4 Класифікація інтернет-магазинів

Існує багато типів інтернет-магазинів, і їх можна класифікувати за різними критеріями. За типом товару:

- універсальні магазини: пропонують широкий асортимент товарів (наприклад, Amazon);
- спеціалізовані магазини: зосереджені на певній категорії товарів (наприклад, магазини одягу або побутової техніки).

За способом взаємодії з клієнтами:

- класичні магазини: повний цикл покупки здійснюється онлайн;
- гібридні магазини: замовлення можна оформити онлайн, але забрати товар в офлайн-магазині.

За технічними особливостями:

- власні магазини: створені і підтримувані вручну програмістами;
- платформи: працюють на готових платформах для створення інтернет-магазинів (наприклад, Shopify, WooCommerce).

За географічним охопленням:

- локальні магазини: працюють в межах одного регіону або країни;
- міжнародні магазини: здійснюють продаж по всьому світу.

За типом товарів:

- фізичні товари: матеріальні товари, які потребують доставки (наприклад, одяг, техніка);
- цифрові товари: продукція, яку можна завантажити або використовувати онлайн (наприклад, програмне забезпечення, підписки).

За моделлю монетизації:

- прямий продаж: безпосередня торгівля товарами через інтернет;
- підписка: надання товарів або послуг на основі підписки;
- маркетплейс: платформи, де продавці можуть розміщувати свої товари для продажу (наприклад, eBay, Etsy).

1.5 Прямі конкуренти в ніші продажу кросівок

У межах дослідження було проаналізовано основних конкурентів у сфері онлайн-продажу кросівок. Зокрема, розглянуто такі платформи, як sneakerstudio, stepin та sneakerbox. Кожен із конкурентів має свої слабкі сторони, як у технічному плані, так і в частині користувацького інтерфейсу. Серед основних проблем можна виділити повільну роботу сайтів, слабку адаптацію під мобільні пристрої, неінтуїтивну навігацію, а також застарілий або перевантажений дизайн (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика конкурентів додатка

Конкурент	Технічні недоліки	UI-недоліки
sneakerstudio	Повільне завантаження сторінок, слабка адаптація під мобільні	Перевантажена головна сторінка, дрібні шрифти
stepin	Відсутність фільтрації за брендами, слабка SEO-оптимізація	Неінтуїтивне меню, застарілий дизайн
sneakerbox	Немобільна версія сайту працює некоректно	Надто багато спливаючих вікон, відволікає користувача

1.6 Висновок

З аналізу видно, що конкуренти мають технічні проблеми (повільна робота, помилки, відсутність адаптації) та недоліки інтерфейсу (застарілий дизайн, складна навігація).

Мій сайт вирішує ці проблеми завдяки сучасному адаптивному дизайну, зручному UX та стабільній технічній реалізації.

2 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Інструменти для розробки клієнтської частини

Всі вебзастосунки так чи інакше фундаментально побудовані за допомогою трьох основних технологій: JavaScript, HTML та CSS. Ці технології працюють разом, щоб створити інтерактивні, зручні та привабливі вебсторінки. JavaScript відповідає за функціональність та динаміку, тоді як HTML та CSS формують структуру та стиль вебсторінки.

HTML є мовою розмітки, яка використовується для створення структури вебзастосунків. Вона складається з елементів та тегів, що визначають різні частини документа, такі як заголовки, абзаци, зображення, посилання тощо. Кожен елемент має відкриваючий і закриваючий тег, що дозволяє браузерам правильно інтерпретувати та відображати контент. Він також підтримує атрибути, які надають додаткову інформацію про елементи, наприклад, ідентифікатори, класи та стилі.

CSS використовується для опису вигляду та форматування HTML-документів. Вона дозволяє розробникам відокремлювати контент від дизайну, забезпечуючи гнучкість та контроль над виглядом вебзастосунків. Визначає стиль елементів HTML, такі як кольори, шрифти, відступи, межі та розташування на сторінці. Його правила застосовуються до елементів через селектори, які можуть бути ідентифікаторами, класами або тегами. CSS підтримує каскадування, що дозволяє застосовувати кілька стилів до одного елемента, враховуючи їхню пріоритетність.

JavaScript – це високорівнева, мультипарадигмова та інтерпретована мова програмування. Підтримує об'єктноорієнтоване, функціональне та подійно-орієнтоване програмування. Це прототипно-орієнтована мова, що дозволяє легку передачу властивостей між об'єктами [4]. Мова була створений у 1995 році Бренданом Айком для компанії Netscape. Спочатку вона була задумана як доповнююча мова для розширення 20 можливостей HTML. Після стрімкого зростання популярності JavaScript, в 1996 році було вирішено

стандартизувати мову. Стандарт ECMAScript був затверджений у 1997 році і продовжував розвиватися. Значними версіями були ES3 (1999), що впровадив більшість основних функцій, і ES5 (2009), що покращив безпеку та додав нові методи для роботи з масивами та об'єктами. З випуском Node.js у 2009 році JavaScript вийшов за межі браузера і став використовуватися для серверних додатків. Це відкрило нові можливості для JavaScript, дозволяючи використовувати його для розробки повноцінних додатків з єдиною мовою програмування. Щорічні оновлення ECMAScript після 2015 року додали багато нових функцій і покращень, таких як стрілкові функції, класи, проміси та модулі. JavaScript продовжує бути однією з найпопулярніших мов програмування, використовуючись не тільки для веброзробки, але й для мобільних та комп'ютерних додатків. Порівняння мови JavaScript з іншими мовами подано в табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

Критерії порівняння	JavaScript	TypeScript	Python
1	2	3	4
Ліцензія	-	Apache License 2.0	Python Software Foundation License
Простота навчання та написання коду	+	+/-	+/-
Наявність зрозумілої та повної документації	+	+	+
Сильна підтримка спільноти	+	+	+/-
Наявність популярних фреймворків	+	+	+/-
Продуктивність	+	+	+

Продовження таблиці 2.1

1	2	3	4
Особливості	Динамічна типізація, основна мова для фронтенду веброзробки	Статична типізація, надбудова над JavaScript для покращення розробки великих проєктів	Універсальність, використовується у різних сферах: від веброзробки до машинного навчання

У сучасній розробці зазвичай використовуються різні фреймворки та бібліотеки. Одними з найпопулярніших фреймворків для розробки клієнтської частини є React, Angular і Vue.js.

2.2 React для створення клієнтської частини інтернет-магазину

Згідно з офіційною документацією React [5] можна дізнатись, що це бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена для спрощення розробки складних вебзастосунків. Основна концепція React базується на компонентах, які є самодостатніми, незалежними частинами інтерфейсу, що можуть бути багаторазово використані. Компоненти можуть приймати дані через аргументи і повертати те, що повинно бути відображено на екрані. Бібліотека використовує JSX, розширення синтаксису JavaScript, що дозволяє писати HTML подібний код безпосередньо в JavaScript. Це робить код більш зрозумілим і легким для підтримки, оскільки логіка відображення тісно пов'язана з логікою компонента. React використовує хуки, такі як `useState` і `useEffect`, що дозволяють додавати в компоненти React функціональність управління станом і побічних ефектів без необхідності використання класових компонентів. Це спрощує створення компонентів і покращує читабельність коду. Популярність React грає важливу роль в

розробці, адже розвинена екосистема пропонує інтеграцію з різними бібліотеками і фреймворками, такими як Redux для управління станом додатка або Next.js для серверного рендерингу та генерації статичних сайтів. Це робить React гнучким інструментом, який може бути використаний для створення як невеликих односторінкових додатків, так і великих комплексних систем. Ще однією важливою особливістю бібліотеки є здатність до інкрементального оновлення. React обробляє оновлення користувацького інтерфейсу за допомогою віртуального DOM, що дозволяє ефективно мінімізувати кількість змін, які потрібно зробити в реальному DOM. Це покращує продуктивність додатків і забезпечує швидкий відгук на дії користувача. React також підтримує розробку нативних додатків через React Native, що дозволяє використовувати ті самі принципи і компоненти для створення додатків для iOS та Android. Це забезпечує можливість розробникам створювати кросплатформені додатки з високим рівнем продуктивності. Поширеність бібліотеки:

- React є однією з найбільш популярних бібліотек для створення користувацьких інтерфейсів;
- вона використовується багатьма великими компаніями, такими як Facebook, Instagram, Airbnb і Netflix.

Переваги бібліотеки:

- компонентна архітектура: дає змогу створювати масштабовані та повторно використовувані компоненти;
- віртуальний DOM: покращує продуктивність, мінімізуючи необхідність маніпуляцій із реальним DOM;
- розвинена екосистема: підтримка бібліотек для управління станом (Redux), маршрутизації (React Router) та серверного рендерингу (Next.js);
- кросплатформеність: можливість створювати додатки для мобільних платформ через React Native.

Недоліки бібліотеки:

- навчальна кривина: складність у розумінні концепції JSX і використання хуків для новачків;
- часті оновлення: постійні зміни у нових версіях можуть призводити до необхідності адаптації старого коду.

2.3 Angular для створення клієнтської частини інтернет-магазину

Проаналізувавши документацію Angular [6], можна зробити висновок, що це платформа для розробки вебдодатків на основі TypeScript, яка включає фреймворк, бібліотеки і інструменти для побудови масштабованих та високопродуктивних додатків. Дозволяє створювати динамічні односторінкові додатки (SPA) з використанням структурованого підходу. Ключовою особливістю фреймворку є його використання TypeScript, що додає статичну типізацію і сучасні можливості JavaScript, роблячи код більш надійним і легким для підтримки. Angular застосовує компонентну архітектуру, де кожен компонент складається з шаблону, логіки та стилів, що сприяє модульності та повторному використанню коду. Даний фреймворк використовує директиви та прив'язку даних для управління шаблонами і динамічного оновлення інтерфейсу користувача. Директиви дозволяють розширювати HTML з новою поведінкою, тоді як прив'язка даних забезпечує синхронізацію між моделлю даних та інтерфейсом користувача. Двостороння прив'язка даних дозволяє автоматично оновлювати модель при зміні даних в інтерфейсі і навпаки. Angular CLI надає інструменти для автоматизації процесів розробки, таких як створення нових проєктів, генерація компонентів, сервісів та інших частин додатка, а також для тестування і збірки проєкту. Використання CLI значно спрощує робочий процес і підвищує продуктивність розробників. Поширеність бібліотеки:

- Angular – це популярний фреймворк для створення масштабованих і складних вебдодатків;

- він активно використовується у великих підприємствах і стартапах.

Переваги бібліотеки:

- TypeScript: статична типізація забезпечує більшу надійність і зручність підтримки коду;
- модульність: чітка структура додатків, що дозволяє ефективно працювати в команді;
- багато вбудованих функцій: підтримка двосторонньої прив'язки даних, директив, форм, HTTP-запитів;
- масштабованість: добре підходить для великих проєктів і корпоративних додатків.

Недоліки бібліотеки:

- висока складність: потребує більше часу для освоєння в порівнянні з іншими фреймворками;
- важкість у налаштуванні: для новачків може бути складно налаштувати проєкт з нуля.

2.4 Vue.js для створення клієнтської частини інтернет-магазину

Розглянемо документацію ще одного фреймворку Vue.js [7]. З неї дізнаємось, що це фреймворк для створення користувацьких інтерфейсів. Він розроблений, щоб бути інтуїтивно зрозумілим, продуктивним і гнучким, що дозволяє його використовувати як для побудови простих односторінкових додатків, так і складних інтерфейсів. Він базується на стандартних технологіях HTML, CSS та JavaScript, що робить його доступним для розробників з будь-яким рівнем досвіду. Основою Vue.js є його реактивна система, яка забезпечує високу продуктивність шляхом автоматичної оптимізації рендерингу компонентів. Даний фреймворк також підтримує компонентну архітектуру, що сприяє модульності та повторному використанню коду. Багата екосистема дозволяє адаптуватися до вимог більшості проєктів. Vue Router забезпечує

маршрутизацію, а Vuex надає можливості для керування станом додатка. Також доступні інструменти для розробки, такі як Vue CLI, що спрощує налаштування проєктів і управління залежностями. Поширеність бібліотеки:

- Vue.js набирає популярність завдяки своїй простоті і гнучкості;
- багато розробників вибирають його для проєктів різного масштабу, від простих додатків до великих систем.

Переваги бібліотеки:

- реактивність: автоматична синхронізація даних між моделлю та відображенням на інтерфейсі;
- гнучкість: дозволяє комбінувати використання лише необхідних частин фреймворку або повноцінну інтеграцію;
- прекрасна інтеграція з іншими бібліотеками та фреймворками.

Недоліки бібліотеки:

- менша популярність у порівнянні з React та Angular: хоча Vue дуже популярний серед спільноти, він все ще менш використовуваний в порівнянні з іншими фреймворками;

Як React, Angular та Vue оцінюються за різними критеріями, що дозволяє зробити більш точний вибір залежно від потреб конкретного проєкту, можна побачити у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика фреймворків

Критерії порівняння	React	Angular	Vue
Ліцензія	MIT License	MIT License	MIT License
Простота навчання та написання коду	+-	-	+
Наявність зрозумілої та повної документації	+	+	+

Продовження таблиці 2.2

Критерії порівняння	React	Angular	Vue
Сильна підтримка спільноти	+	+	+
Гнучкість і масштабованість	+	+	+/-
Продуктивність	+	+	+
Підтримка інструментів розробки	+	+	+
Використання у великих проєктах	+	+	+/-

З урахуванням порівняння, ми обрали React для розробки клієнтської частини інтернет-магазину. Це рішення обґрунтовано його високою популярністю, чудовою екосистемою, гнучкістю та широкими можливостями для масштабування. React дозволяє створювати інтерактивні та швидкі вебдодатки, що є критичним для успіху сучасних інтернет-магазинів.

2.5 Технології для розробки клієнтської частини

Розглянемо основні бібліотеки, які будуть необхідні під час написання коду. Перш за все почнемо з управління станами. Для цього було обрано одну бібліотеку.

На даний момент для управління станом у вебдодатках активно використовуються різні технології, і однією з найбільш популярних є Redux разом з Redux Toolkit. Ці інструменти дозволяють централізовано керувати станом додатка та забезпечують ефективну і зручну роботу з даними. В рамках цієї роботи було обрано використання Redux з Redux Toolkit для управління станом у клієнтській частині інтернет-магазину, а також CSS модулів для стилізації інтерфейсу замість використання Bootstrap.

Redux та Redux Toolkit для управління станом:

- Redux є бібліотекою для управління станом додатка, яка базується на централізованому сховищі (store), що дозволяє зберігати всі дані в одному місці. Це полегшує синхронізацію стану між різними компонентами додатка. Використання Redux Toolkit значно спрощує налаштування Redux, надаючи розробникам зручні інструменти для створення дій, редукторів та асинхронних запитів;
- простота використання: завдяки Redux Toolkit процес налаштування Redux став значно простішим. Бібліотека автоматизує створення редукторів і асинхронних дій, що значно зменшує кількість необхідного коду для роботи з глобальним станом.
- управління станом: використання Redux дозволяє централізувати весь стан додатка в одному сховищі, що спрощує доступ до даних і передавання їх між компонентами без зайвих зусиль. Це також полегшує масштабування додатка, оскільки всі дані зберігаються в одному місці і мають чітку структуру;
- продуктивність: завдяки автоматичному відстеженню змін у стані та оптимізації рендерингу, Redux з Redux Toolkit допомагають знизити кількість непотрібних перерендерів компонентів, що забезпечує покращену продуктивність додатка;
- гнучкість та масштабованість: Redux Toolkit дає змогу ефективно організовувати складні стани і працювати з асинхронними запитами. Це робить бібліотеку такою, яка підходить для масштабованих проєктів, таких як інтернет-магазини, де необхідно керувати великим обсягом даних;
- підтримка спільноти та документація: Redux і Redux Toolkit мають велику і активну спільноту розробників, що забезпечує регулярні оновлення і полегшує процес вирішення проблем. Документація містить детальні приклади і інструкції, що значно полегшує процес навчання та інтеграції цих технологій.

CSS модулі для стилізації інтерфейсу:

- у якості альтернативи Bootstrap, в даному проєкті використано CSS модулі для стилізації інтерфейсу. Це дозволяє уникнути проблем з глобальними конфліктами класів, забезпечуючи локалізацію стилів на рівні компонентів;
- локалізація стилів: завдяки використанню CSS модулів, кожен компонент має свої унікальні стилі, що не впливають на інші частини додатка. Це дозволяє уникнути проблем, пов'язаних з глобальним стилем, і забезпечує більшу організованість і чистоту коду;
- простота використання: CSS модулі легко інтегруються в проєкт, і кожен клас автоматично отримує унікальне ім'я, що мінімізує можливість помилок у CSS і дає змогу чітко контролювати стилі для кожного компонента;
- масштабованість: використання CSS модулів дозволяє створювати компоненти, що можуть бути легко повторно використані в інших частинах додатка, без ризику змішування стилів. Це робить код більш підтримуваним і адаптованим до змін у проєкті.

React Router для маршрутизації:

- React Router був обраний для реалізації навігації між сторінками інтернет-магазину. Це потужна бібліотека для створення багатосторінкових додатків з динамічною навігацією без перезавантаження сторінки;
- гнучкість: React Router підтримує вкладені маршрути, динамічні параметри URL та захищені маршрути, що дозволяє створювати складні структури навігації;
- інтуїтивний API: завдяки компонентам BrowserRouter, Routes, Route та Link, управління навігацією є простим і зрозумілим.

Axios для здійснення HTTP запитів:

- для здійснення HTTP запитів у додатку було використано Axios – бібліотеку [8], що дозволяє ефективно працювати з асинхронними запитами;
- підтримка промісів: Axios дозволяє зручно обробляти асинхронні операції і працювати з ними без зайвих труднощів;

- інтуїтивно зрозумілий синтаксис: API Axios є простим у використанні, що дозволяє швидко освоїти бібліотеку і ефективно застосовувати її для взаємодії з сервером;
- конфігурація запитів: Axios підтримує налаштування запитів, включаючи тайм-аути та заголовки, що дозволяє ефективно налаштувати взаємодію з API.

2.6 Інструменти для розробки серверної частини

На даний момент існує багато мов програмування та способів розробки серверної частини або ж бекенду в цілому, в даній роботі акцент був зроблений на технологіях, які використовують JavaScript під капотом. Всі вони побудовані на базі Node.js – це основа для побудови серверної частини на JavaScript. Він дозволяє виконувати JavaScript код поза браузером і забезпечує високопродуктивне асинхронне середовище. Завдяки своїй подійній моделі та неблокуючому введенню/виведенню він може обробляти велику кількість одночасних з'єднань, що робить його хорошим рішенням для роботи в реальному часі та підтримання додатків з високим навантаженням. Він підтримує написання коду з використанням сучасних стандартів JavaScript, включаючи ECMAScript 6 і новіші версії, що дозволяє розробникам використовувати останні можливості мови. Крім того, його інтеграція з такими фреймворками, як 30 Express.js, полегшує створення вебзастосунків та API, забезпечуючи ефективне управління маршрутизацією, обробкою запитів і відповідей, а також середовищем виконання. Node.js також використовується для створення інструментів командного рядка та інших утиліт, що робить його універсальним інструментом для різних типів проєктів. Його гнучкість, ефективність і активна спільнота розробників роблять Node.js одним із найпопулярніших виборів для створення серверних додатків і сервісів у сучасній веброзробці.

2.7 Express.js для створення серверної частини інтернет-магазину

Перший розглянутий варіант – це Express.js, з офіційної документації [9] можна дізнатись, що це фреймворк для Node.js, який спрощує створення вебзастосунків і API. Він забезпечує тонкий шар фундаментальних функцій без нав'язування структури, що дозволяє розробникам будувати застосунки із зручною для них архітектурою. Express.js використовує середовище виконання Node.js для обробки HTTP-запитів і відповідає за маршрутизацію, обробку запитів і відповідей, а також управління middleware. Основною концепцією Express.js є маршрутизація, яка дозволяє визначати різні маршрути для обробки запитів HTTP-методів (GET, POST, PUT, DELETE тощо). Це досягається шляхом створення маршрутів за допомогою функцій `app.get()`, `app.post()`, `app.put()`, `app.delete()` та інших. Ці функції приймають шлях маршруту і обробник, який виконується, коли запит відповідає цьому маршруту. Express.js підтримує middleware, що дозволяє виконувати код між отриманням запиту і відправкою відповіді. Даний принцип може бути використаний для обробки аутентифікації, логування, обробки помилок та інших задач. Це робить його потужним інструментом для розробки додатків з багаторівневою архітектурою, де кожен рівень виконує певну функцію. Фреймворк також підтримує шаблонізатори, такі як Pug, EJS та Handlebars, що дозволяє динамічно генерувати HTML-сторінки на сервері. Це корисно для додатків, які потребують рендерингу сторінок на сервері з даними, що надходять у запиті. Express.js інтегрується з різними базами даних через сторонні бібліотеки, такі як Mongoose для MongoDB або Sequelize для SQL баз даних. Це спрощує розробку додатків, які потребують збереження і маніпуляції даними в базах даних. Завдяки своїй популярності фреймворк має 31 активну спільноту та великий набір плагінів і модулів, що дозволяє розширювати його функціональність. Поширеність бібліотеки:

- Express.js – найпопулярніший фреймворк для Node.js;

- широко використовується як у малих, так і у великих проєктах (Netflix, Uber).

Переваги бібліотеки:

- простий у вивченні та використанні;
- гнучкий, не нав'язує структуру;
- велика кількість middleware та активна спільнота;
- підходить для швидкого створення API.

Недоліки бібліотеки:

- відсутність єдиної структури проєкту;
- мало функцій "з коробки" – багато налаштовується вручну;
- менш зручний для великих систем без додаткової організації.

2.8 Koa.js для створення серверної частини інтернет-магазину

Другий розглянутий фреймворк мною був Koa.js. Koa.js – це сучасний вебфреймворк для Node.js, розроблений творцями Express.js з метою створення більш компактного, гнучкого та ефективного інструменту для побудови серверних застосунків. Основна ідея Коа полягає в тому, щоб надати розробнику мінімальний набір інструментів, залишаючи більшість функцій на вибір через підключення зовнішніх middleware. Такий підхід дає повний контроль над архітектурою додатку та дозволяє уникнути нав'язаної структури або непотрібних залежностей.

Koa.js був побудований з урахуванням сучасних можливостей JavaScript, зокрема повної підтримки `async/await`. Це дозволяє уникнути “callback hell” і зробити код більш читабельним та підтримуваним. У Коа використовується унікальний механізм обробки запитів через каскадний ланцюжок middleware (так званий `onion model`), де кожен middleware може виконати певну логіку до та після виклику наступного елемента в ланцюгу. Такий підхід забезпечує високу гнучкість та контроль над кожним етапом обробки запиту.

На відміну від Express, Koa не має вбудованого роутингу, обробки тіла запиту, логування або обробки помилок. Усе це реалізується через окремі пакети, такі як `@koa/router`, `koa-bodyparser`, `koa-logger` тощо. Це дозволяє створювати легковагові та високопродуктивні серверні додатки, які містять тільки необхідні компоненти. Поширеність бібліотеки:

- Koa.js розроблений творцями Express.js, але менш популярний;
- застосовується переважно в проєктах, де потрібна висока продуктивність і гнучкість.

Переваги бібліотеки:

- сучасний і легкий фреймворк з мінімалістичним підходом;
- повна підтримка `async/await` з самого початку;
- висока продуктивність та контроль над обробкою запитів.

Недоліки бібліотеки:

- менше готових рішень та `middleware` у порівнянні з Express;
- більш складний у налаштуванні для новачків;
- відсутність стандартної структури застосунку.

2.9 Nest.js для створення серверної частини інтернет-магазину

Після аналізу технологій для розробки серверної частини і вивчення документації [10] був обраний Nest.js. Це фреймворк для створення серверних додатків Node.js, побудований на TypeScript. Він застосовує модульну архітектуру, яка дозволяє легко організувати код і забезпечити високу масштабованість додатків. Nest.js використовує принципи об'єктноорієнтованого програмування, функціонального програмування та архітектури, керованої метаданими. Фреймворк забезпечує інверсію контролю через ін'єкцію залежностей, що спрощує тестування та повторне використання коду. Основні компоненти Nest.js включають модулі, контролери та провайдери. Модулі є основними будівельними блоками додатка, контролери відповідають за обробку HTTP-запитів і повернення відповідей, а провайдери

реалізують бізнес-логіку і взаємодію з базами даних або іншими сервісами. Nest.js підтримує створення RESTful API, GraphQL API та мікросервісну архітектуру. Завдяки інтеграції з різними бібліотеками та інструментами, такими як TypeORM для роботи з базами даних і Passport для аутентифікації, Nest.js дозволяє швидко створювати потужні та масштабовані серверні додатки. Фреймворк також надає вбудовані інструменти для тестування, включаючи підтримку юніт-тестів і інтеграційних тестів через Jest, що забезпечує високу якість коду і полегшує процес розробки. Поширеність бібліотеки:

- NestJS активно використовується в корпоративних проєктах;
- з кожним роком зростає його популярність завдяки підтримці

TypeScript та архітектурі, схожій на Angular.

Переваги бібліотеки:

- чітка структура та модульна архітектура;
- повна підтримка TypeScript із коробки;
- підтримка мікросервісів, GraphQL, WebSockets, REST;
- вбудовані засоби для тестування.

Недоліки бібліотеки:

- більш крута крива навчання через складність архітектури;
- менш підходить для простих або невеликих проєктів;
- строгість структури може обмежувати гнучкість для деяких розробників.

Порівняння цих фреймворків подано у табл. 2.3.

Таблиця 2.3 – Порівняльна характеристика серверних фреймворків

Критерії порівняння	Nest.js	Express.js	Koa.js
Ліцензія	MIT License	MIT License	MIT License
Простота навчання та написання коду	+ -	+	+ -
Наявність зрозумілої та повної документації	+	+	+ -
Сильна підтримка спільноти	+	+	+ -
Гнучкість і масштабованість	+	+ -	+
Продуктивність	+	+	+
Архітектура	Структурована (OOP, DI, модулі)	Мінімальна, без наві'язаної структури	Мінімалістична, middleware-based
Підтримка TypeScript	+ (вбудована)	+ - (через налаштування)	+ - (через сторонні типи)
Підтримка інструментів тестування	+	+ -	+ -
Підтримка мікросервісної архітектури	+	+ -	+ -

2.10 Технології для розробки серверної частини інтернет-магазину

Необхідна річ в розробці серверної частини – це ORM система, до прикладу Sequelize. Вона спрощує взаємодію з реляційними базами даних, такими як MySQL, PostgreSQL, SQLite і Microsoft SQL Server.

Однією з ключових складових для зберігання даних у вебдодатку є система керування базами даних. У цьому проєкті було обрано MySQL – одну з найпопулярніших реляційних СКБД з відкритим вихідним кодом [11]. MySQL має широке застосування в реальних проєктах, надає потужні можливості для роботи з транзакціями, індексацією, зв'язками між таблицями, а також підтримує масштабованість та високу продуктивність. Крім того, вона має активну спільноту, багатофункціональні GUI-інструменти (наприклад, MySQL Workbench), і легко інтегрується з Node.js-проєктами через ORM-бібліотеки, такі як Sequelize.

Саме через її стабільність, зрозумілу архітектуру, підтримку складних SQL-запитів та широке поширення серед хостинг-провайдерів, MySQL є надійним вибором для створення серверної частини додатку.

Використовуючи Sequelize, розробники можуть працювати з базами даних через об'єкти JavaScript замість написання сирих SQL-запитів. Основною перевагою Sequelize є його здатність автоматизувати багато аспектів управління базами даних, таких як створення таблиць, відносин між таблицями, а також валідація та маніпуляція даними. Sequelize використовує моделі для представлення таблиць у базі даних. Модель визначається за допомогою JavaScript-класів, що забезпечує високий рівень абстракції і зручність використання. Використання Sequelize включає три основні етапи: визначення моделей, синхронізація з базою даних та виконання CRUD-операцій (створення, читання, оновлення, видалення). Крім того, Sequelize підтримує складні запити, включаючи агрегації, групування, об'єднання таблиць та інші операції, що дозволяє виконувати складні операції з даними з мінімальними зусиллями. Також надає можливості міграції баз даних, що

дозволяє керувати змінами в структурі бази даних у часі. Це важливо для великих проєктів, де структура бази даних може змінюватися протягом усього життєвого циклу проєкту. Міграції дозволяють автоматично застосовувати зміни до бази даних, зберігаючи історію змін і забезпечуючи можливість відкату. Sequelize надає потужні інструменти для валідації даних і управління зв'язками між таблицями, включаючи один до одного, один до багатьох та багато до багатьох зв'язки. Це робить його гнучким інструментом для роботи з будь-якою реляційною базою даних.

Для безпечного передавання важливої інформації між клієнтом і сервером необхідно використати JWT. Інформація в ньому є цифрово підписаною за допомогою алгоритму HMAC або асиметричними ключами RSA або ECDSA, що забезпечує цілісність і достовірність даних. JWT складається з трьох частин, розділених крапками: заголовок (header), корисного навантаження (payload) і підпису (signature). Заголовок містить інформацію про тип токена і алгоритм підпису. Корисне навантаження містить заяви (claims), які можуть бути про користувача або про сам токен. Підпис використовується для перевірки достовірності повідомлення. Основні переваги використання JWT включають компактність, самодостатність і незалежність від стану, що дозволяє легко передавати токени між різними сервісами. JWT зазвичай використовується для аутентифікації та авторизації у вебдодатках, зберігаючи токен на клієнтській стороні (наприклад, у локальному сховищі або куку) і передаючи його з кожним запитом до сервера. JWT забезпечує надійну і масштабовану систему аутентифікації, дозволяючи серверам перевіряти достовірність запитів без необхідності зберігати стан сесій. Це особливо корисно в мікросервісних архітектурах і додатках з високим навантаженням, де ефективність і масштабованість є ключовими факторами. JWT також підтримує кілька типів заяв, такі як зарезервовані заяви (типу "iss" – видавець токена, "exp" – час закінчення дії), публічні заяви (визначені користувачами) і приватні заяви (обидві сторони узгоджують їх

використання). Це робить JWT дуже гнучким і універсальним інструментом для передачі безпечної інформації між клієнтом і сервером.

Наступна бібліотека це Bcrypt. Основна перевага Bcrypt полягає в його стійкості до атак словникового типу та атак за допомогою перебору паролів. Він використовує алгоритм шифрування Blowfish, який вимагає значних обчислювальних ресурсів для генерації кожного хешу, що робить злом паролів значно складнішим і більш витратним для злоумисників.

Однією з ключових особливостей Bcrypt є можливість налаштовувати так звану вартість (cost factor) хешування, яка визначає кількість обчислень, необхідних для генерації хешу. Це дозволяє регулювати складність обчислення хешу в залежності від потужності вашого обладнання та вимог до безпеки. Зі збільшенням вартості процес хешування стає більш складним і займає більше часу, що значно ускладнює атаки на паролі.

Bcrypt також забезпечує автоматичне використання унікальних сольових значень (salt) для кожного пароля. Сіль додається до пароля перед хешуванням, що робить кожен хеш унікальним навіть для однакових паролів. Це ефективно захищає від атак за попередньо обчисленими хешами (rainbow tables), значно підвищуючи безпеку зберігання паролів.

Бібліотека Bcrypt легко інтегрується з різними мовами програмування і фреймворками, що робить її зручним інструментом для розробників. Вона широко використовується в індустрії для забезпечення безпеки користувацьких даних у вебдодатках, мобільних додатках і інших програмних продуктах.

Завдяки своїй надійності та ефективності, Bcrypt став стандартом де-факто для хешування паролів. Використання цієї бібліотеки дозволяє розробникам забезпечити високий рівень захисту даних користувачів, зменшуючи ризик успішних атак на паролі. Інтуїтивний API і добре задокументовані функції Bcrypt роблять його доступним для розробників будь-якого рівня, сприяючи створенню безпечних і надійних додатків.

Щоб зробити розробку зручнішою та прозорішою, варто інтегрувати в проєкт Swagger. Це набір інструментів для автоматизації документування API. Основний компонент – це Swagger UI, який надає інтерактивну документацію для вашого API, дозволяючи розробникам легко тестувати і взаємодіяти з різними кінцевими точками без необхідності написання додаткового коду. Swagger використовує специфікацію OpenAPI для опису структури API, що робить його стандартизованим та зрозумілим для машин і людей. Файли специфікації написані у форматі YAML або JSON і містять всю інформацію про кінцеві точки, типи запитів, параметри, відповіді та інші метадані. Необхідність Swagger на проєкті зумовлена кількома факторами. Насамперед, він забезпечує автоматизоване створення документації, що економить час і зусилля розробників. Також інтерактивна документація дозволяє швидко тестувати API, що полегшує відладку і забезпечує високу якість продукту. Swagger також полегшує комунікацію між командами розробників і зацікавленими сторонами, забезпечуючи зрозумілий і чіткий опис функціональності API. Це особливо важливо у великих проєктах або у випадках, коли API надається зовнішнім користувачам або партнерам. Як влаштована взаємодія між клієнтом, сервером та базою даних чітко показано на рис. 2.1 [12].

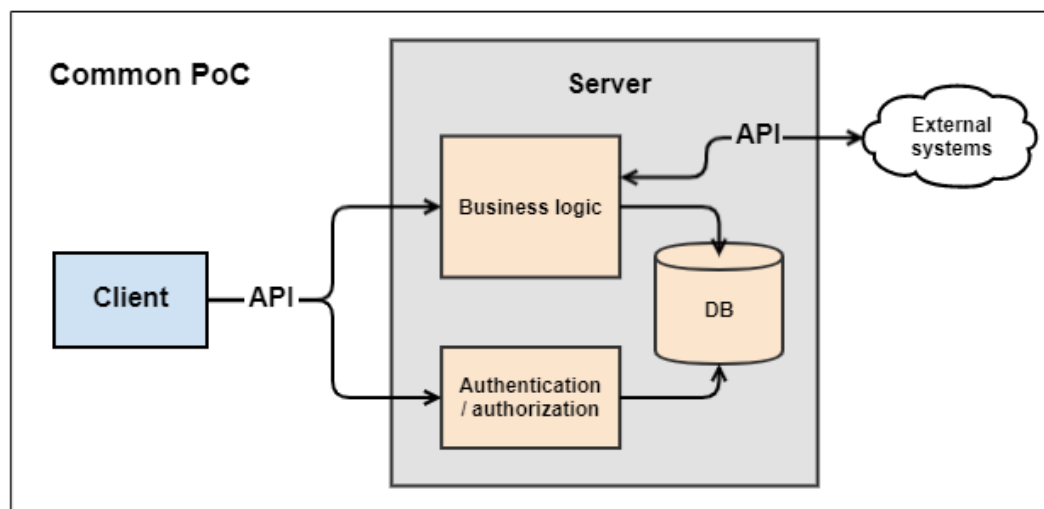


Рисунок 2.1 – Діаграма клієнт-серверного проєкту [12]

2.11 Висновок

Обрав для серверної частини Node.js, тому що це сучасне, високопродуктивне середовище виконання JavaScript, яке дозволяє створювати швидкі, масштабовані та асинхронні вебзастосунки. Node.js має широку екосистему пакетів, активну спільноту та підтримує останні стандарти JavaScript, що робить його потужною основою для серверної розробки.

Серед фреймворків для Node.js було обрано NestJS, оскільки він забезпечує чітку архітектуру, модульність і високий рівень масштабованості, що є особливо важливим для проєктів середнього та великого розміру. NestJS поєднує у собі переваги об'єктноорієнтованого підходу та функціонального програмування, пропонує інтеграцію з популярними бібліотеками (наприклад, TypeORM, Swagger, Passport) і підтримує розробку REST API, GraphQL, мікросервісів. Завдяки використанню TypeScript забезпечується типізація та висока якість коду, що значно спрощує супровід і масштабування застосунку в майбутньому.

Таким чином, комбінація Node.js та NestJS є оптимальним вибором для створення надійної, ефективної та добре структурованої серверної частини сучасного вебзастосунку.

3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби реалізації

Для ефективної програмної реалізації вебзастосунку для створення, аналізу та експорту рекламних креативів необхідно обрати правильні інструменти.

Перший – це розподілена система контролю версій Git, яка використовується для відстеження змін у вихідному коді під час розробки програмного забезпечення. Вона дозволяє кільком розробникам працювати над одним проектом одночасно, забезпечуючи збереження історії змін і можливість повернення до попередніх версій коду. GitHub – це платформа для зберігання і управління Git-репозиторіями з додатковими функціями, такими як управління проектами, інтеграція з іншими інструментами та співпраця. Для застосування їх в проекті необхідно створити репозиторій на GitHub і клонувати його на локальний комп'ютер. Кожен розробник може створити власну гілку для роботи над новими функціями або виправленнями, не впливаючи на основну гілку проекту. Після завершення роботи зміни можуть бути злиті з основною гілкою через pull request, що дозволяє іншим членам команди переглянути і затвердити зміни. Тому я створив свою власну гілку `dev_vlad` і робив pull request та merge у гілку `dev`.

Git дозволяє розробникам здійснювати коміти для збереження змін на локальному рівні з описом виконаної роботи. Це забезпечує детальну історію змін, що робить відстеження прогресу та виявлення проблем значно простішим. Крім того, Git підтримує теги для створення версій проекту, що полегшує випуск релізів. GitHub забезпечує централізоване зберігання репозиторіїв з можливістю доступу з будь-якого місця, де є інтернет-з'єднання. Платформа також надає інструменти для управління проектами, такі як issues для відстеження завдань та багів, а також wiki для документації проекту. Інтеграція з такими інструментами, як неперервна інтеграція (CI), дозволяє автоматизувати процеси тестування і розгортання. Крім того, GitHub

підтримує pull requests, що є ефективним способом для рецензування коду. Кожен pull request може містити коментарі і обговорення, що допомагає підвищити якість коду за рахунок командної співпраці. Прикладом створеної моєї гілки на git показано на рис 3.1.

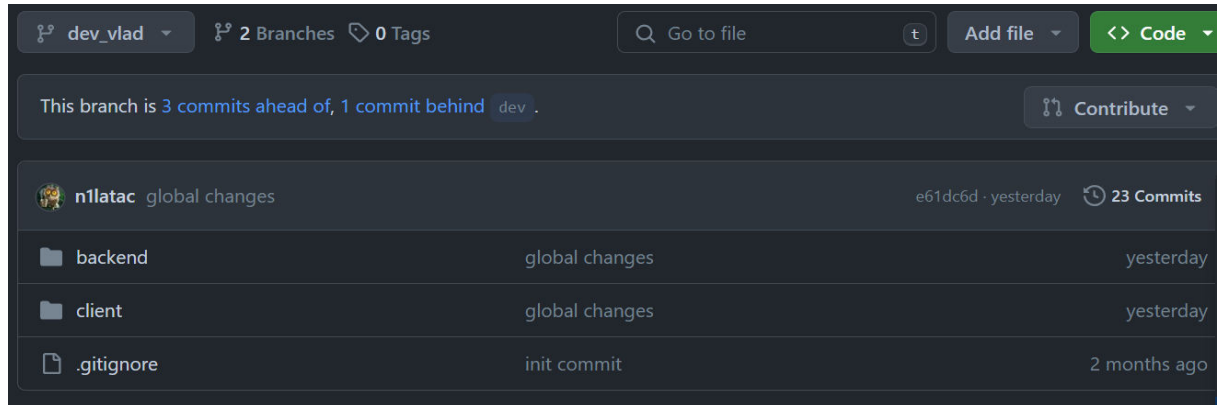


Рисунок 3.1 – Створення власної гілки на github

Не менш важливим є вибір IDE або ж середовища розробки. Одне з найкращих середовищ це PhpStorm. PhpStorm – це комерційний інтегрований середовище розробки (IDE), розроблене компанією JetBrains. Це потужний інструмент, який надає розробникам все необхідне для ефективної роботи з PHP, а також підтримку для HTML, CSS, JavaScript, і багатьох інших мов програмування. PhpStorm забезпечує високу продуктивність завдяки розширеним функціям редагування коду, таким як інтелектуальна автозаповнення, підсвічування синтаксису, автоматичний рефакторинг і багато інших.

Однією з ключових особливостей PhpStorm є його глибока інтеграція з популярними фреймворками та системами управління контентом (CMS) для PHP, такими як Symfony, Laravel, Drupal, WordPress та інші. Це дозволяє розробникам швидко і ефективно створювати вебдодатки, використовуючи відомі інструменти та методології. Крім того, PhpStorm надає вбудовану підтримку для систем контролю версій, таких як Git, SVN, Mercurial та інші, що значно полегшує роботу в команді та управління версіями проєктів.

PhpStorm включає потужні інструменти для налагодження коду, які дозволяють встановлювати точки зупину, крокувати через код, переглядати змінні та виконувати інші операції для діагностики та виправлення помилок. Підтримка налагодження доступна для PHP, JavaScript і багатьох інших мов, що робить PhpStorm універсальним інструментом для веброзробки. Вбудований профайлер допомагає розробникам знаходити вузькі місця в продуктивності додатків і оптимізувати їх.

Ще однією важливою особливістю PhpStorm є його розширюваність за рахунок підтримки численних плагінів. Розробники можуть додавати нові функціональні можливості, такі як підтримка додаткових мов програмування, інструменти для тестування, засоби для інтеграції з хмарними сервісами і багато іншого. Плагіни можна легко встановлювати і налаштовувати через вбудований менеджер плагінів.

Для роботи з проектами PhpStorm надає зручні інструменти для керування файлами і папками, а також підтримує функцію робочих областей (workspaces), що дозволяє організувати кілька проектів у межах одного середовища розробки. Вбудований термінал дозволяє розробникам виконувати командні скрипти безпосередньо з IDE, що підвищує продуктивність і зручність роботи.

Загалом, PhpStorm є відмінним вибором для розробників, які шукають потужне і зручне середовище для створення вебдодатків на PHP. Його розширені можливості, інтеграція з популярними інструментами і фреймворками, а також висока продуктивність роблять його незамінним інструментом для професійних розробників.

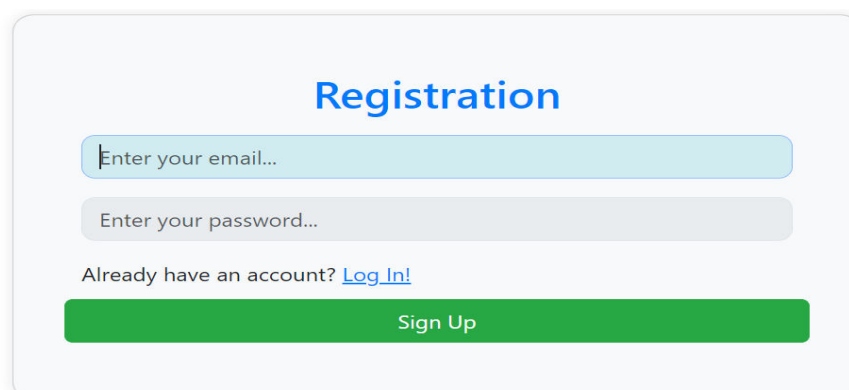
Як інструмент для збірки оберемо Vite – це сучасний інструмент збірки, розроблений для забезпечення швидкого та ефективного середовища розробки для сучасних вебпроектів, зокрема для React-додатків. Він використовує модулі ES (ESM) і забезпечує миттєвий запуск сервера, що дозволяє розробникам починати роботу майже миттєво, незалежно від розміру проекту. Однією з основних переваг Vite є його надзвичайно швидке оновлення

модулів, що дозволяє зміни в коді відображатися в браузері миттєво, без перезавантаження сторінки та втрати стану додатку. Це значно покращує продуктивність і ефективність розробки, оскільки розробники можуть бачити результати своїх змін у реальному часі. Vite також пропонує оптимізовану збірку використовуючи Rollup як пакетувальник. Це забезпечує ефективне розділення коду, що дозволяє зменшити розмір кінцевих файлів і покращити продуктивність завантаження сторінки. Інструмент підтримує багатий набір плагінів, що дозволяє легко розширювати функціональність та інтегрувати різні препроцесори, такі як TypeScript, JSX, CSS та інші. Плагіни можуть бути використані для налаштування збірки під конкретні потреби проєкту, забезпечуючи гнучкість та адаптивність.

Основні та допоміжні технології, які були обрані для розробки вебзастосунку описані у розділі 2, а саме React.js для клієнтської частини та Nest.js для серверної.

3.2 Приклади використання додатку

Для доступу до функцій системи користувачу потрібно зареєструватись або увійти в акаунт, якщо такий існує. Відповідно для реєстрації потрібно вказати пошту та пароль, а для входу тільки пошту та пароль. Також є можливість відновити свій пароль у разі втрати попереднього. Це представлено на рис. 3.2.

The image shows a registration form titled "Registration" in blue text. It contains two input fields: "Enter your email..." and "Enter your password...". Below these fields is a link that says "Already have an account? [Log In!](#)". At the bottom of the form is a green button labeled "Sign Up".

Registration

Enter your email...

Enter your password...

Already have an account? [Log In!](#)

Sign Up

Рисунок 3.2 – Вікно авторизації

Після успішного входу в застосунок користувача зустрічає сторінка магазину, що представлено на рис. 3.3.

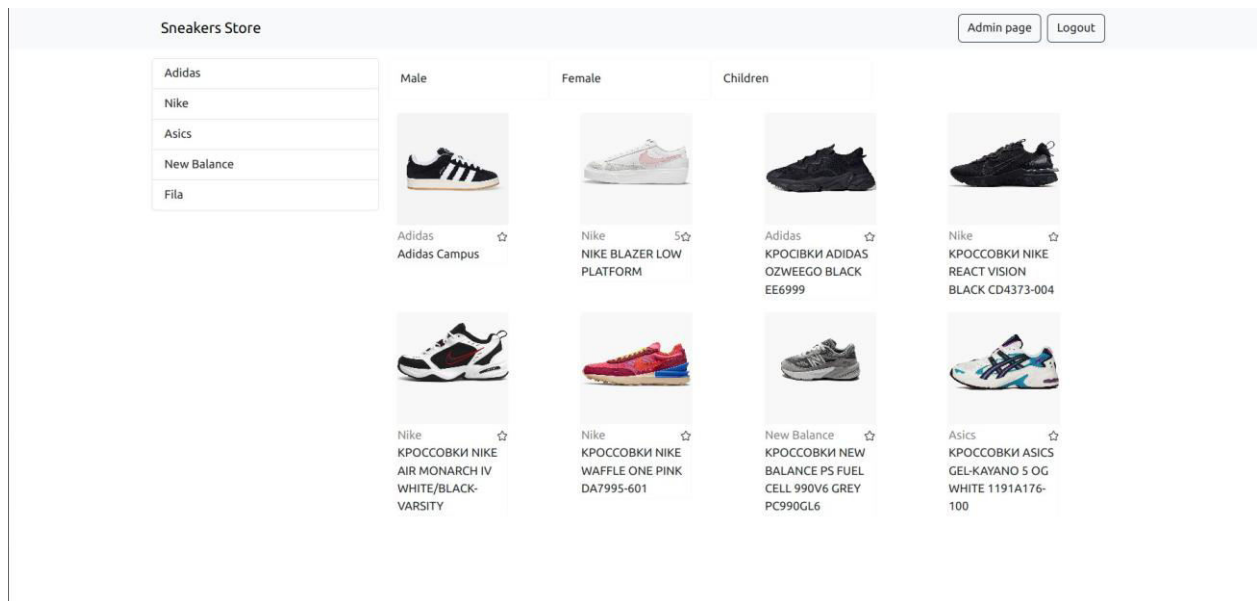


Рисунок 3.3 – Головна сторінка сайту

Дана сторінка існує для перегляду кросівок. Зверху та збоку є панель управління де можна вибирати кросівки за типом, або брендом. Обравши відповідний бренд та тип, можемо переглядати кросівки, які нам потрібні. На карточках кросівок є мінімальні характеристики (ціна, рейтинг).

При кліку на карточку якихось кросівок, ти переходиш на сторінку цих кросівок, де ти можеш побачити вже весь опис та всі характеристики їх. Також можна додати їх до корзини.

Також на сайті є своя адмін панель, заходячи під адміністратором, зверху у NavBar з'являється кнопка Admin page, клікаючи на неї, відкривається сторінка з 3 кнопками, кожна з яких відкриває своє модальне вікно зі своїми діями. Перше модальне вікно – це додавання типу, де ти вказуєш тільки ім'я типу. Це представлено на рис. 3.4.

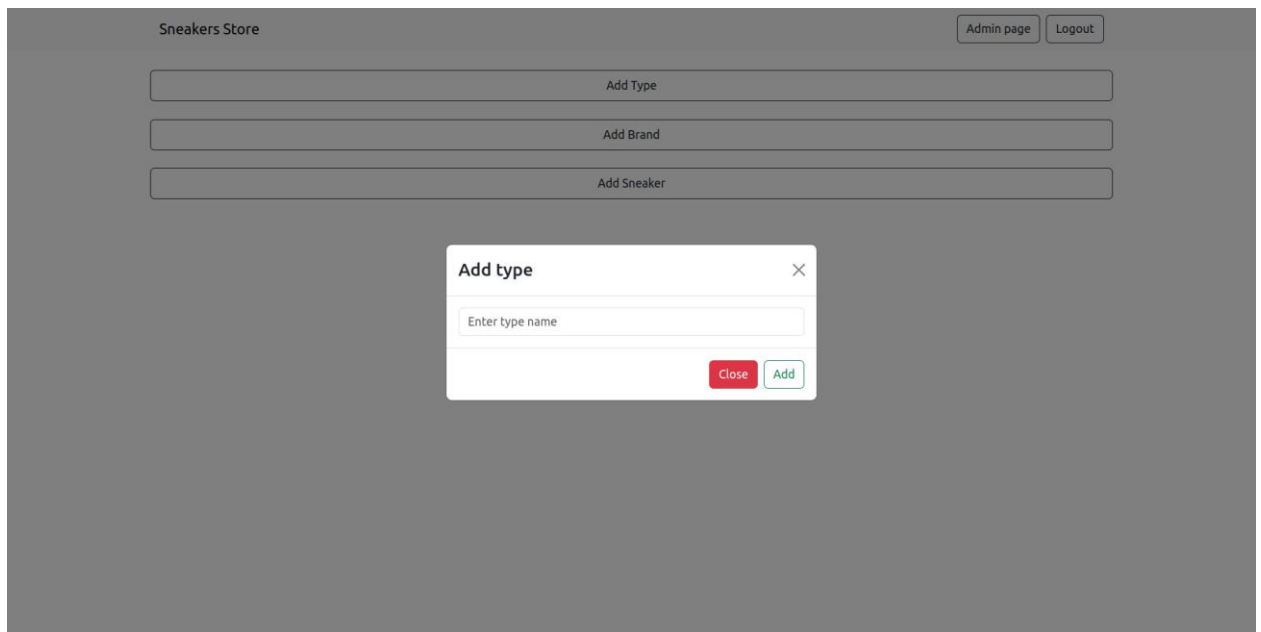


Рисунок 3.4 – Модальне вікно з додаванням типу

Друге модальне вікно аналогічне до першого, тільки там вказуєш назву бренду, а третє – це додавання самих кросівок, вказуючи існуючі бренді та типи, даючи характеристики та вказуючи ціну, що представлено на рис 3.5.

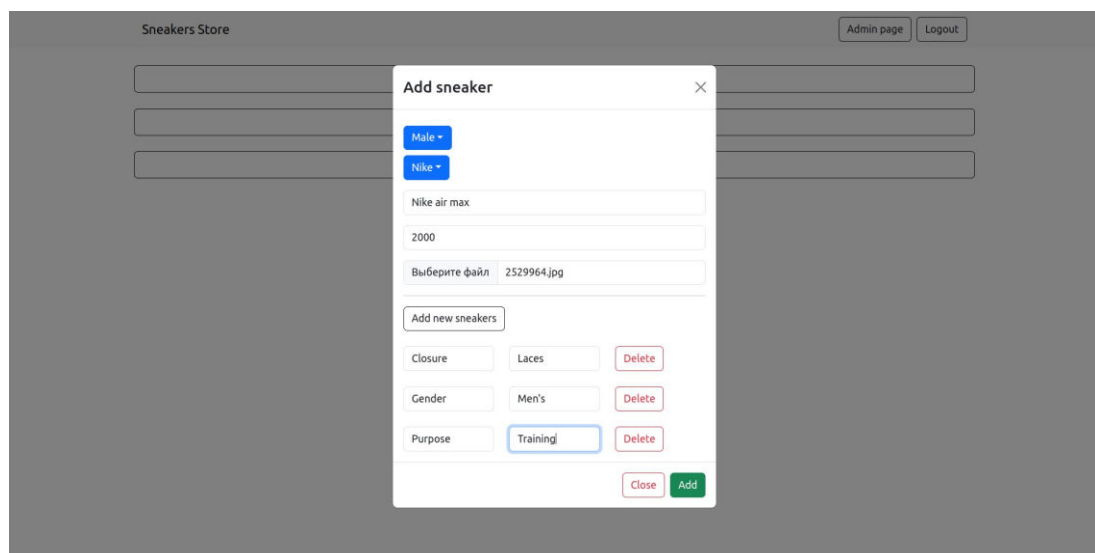


Рисунок 3.5 – Модальне вікно з додаванням кросівок

Для зручності роботи з системою, вся інформація про кросівки, типи, бренди та користувачів зберігається в базі даних, що дозволяє ефективно керувати даними та здійснювати швидкий доступ до них. У БД

використовуються різні таблиці, які містять дані про кожен з елементів: кросівки, бренди, типи, а також користувачів та їхні замовлення. Всі ці таблиці зв'язані між собою через відповідні ключі, що забезпечує цілісність і зручність у роботі з даними. Схема зв'язків між таблицями представлена на рис. 3.6.

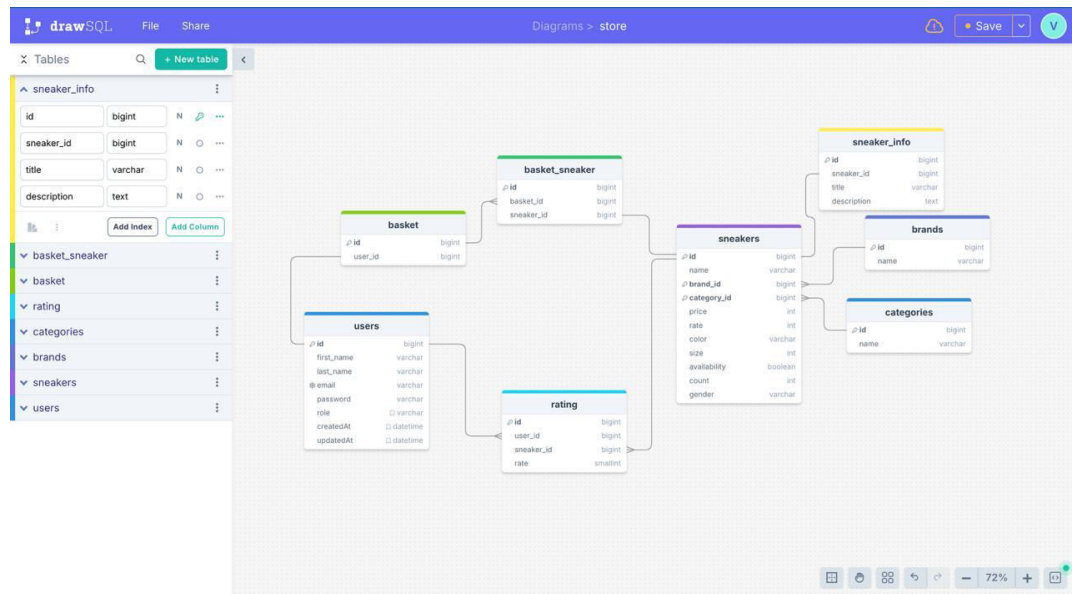


Рисунок 3.6 – Схема зв'язків таблиць у БД системи

Крім того, Postman має багато корисних функцій, таких як можливість автоматизувати тестування API за допомогою колекцій тестів, підтримка зберігання запитів і відповідей для подальшого використання, а також можливість зберігати та спільно використовувати ці дані з іншими членами

команди. Це робить Postman незамінним інструментом для колективної роботи над проєктами, особливо в умовах розробки API.

Ще однією важливою перевагою Postman є його кросплатформенність. Він доступний для використання на різних операційних системах, таких як Windows, macOS та Linux, що дозволяє розробникам працювати з ним на будь-якому обладнанні.

Отже, можна зробити висновок, що Postman – це потужний і зручний інструмент для тестування API та розробки, який забезпечує широкий функціонал і простий у використанні інтерфейс. Він дозволяє розробникам ефективно працювати з API та забезпечує зручну інструментальну платформу для спільної роботи над проєктами.

Також, для тесту даних, які мені приходять на сервер, я використовую DTO. DTO (Data Transfer Object) – це об'єкт, який використовується для передачі даних між клієнтом і сервером. DTO дозволяє чітко визначити структуру даних, які очікуються у запиті або які повертаються у відповіді. Це допомагає уникати передачі зайвої інформації, покращує безпеку та робить код більш передбачуваним і зручним для підтримки.

У NestJS DTO зазвичай описуються у вигляді TypeScript-класів з використанням декораторів для валідації (наприклад, з пакету class-validator). Це дозволяє одразу перевіряти дані, які приходять від користувача, і повертати помилки у разі невідповідності. Переваги використання DTO:

- забезпечення валідності вхідних даних;
- відділення логіки запиту від бізнес-логіки додатку;
- спрощення документування та тестування API;
- зменшення ризику помилок через надлишкові або відсутні поля.

Такий підхід дозволяє контролювати структуру і якість даних, що передаються в додатку, і значно полегшує підтримку великого проєкту.

ВИСНОВКИ

У першому розділі ми розглянули концепцію електронної комерції та її переваги для бізнесу і споживачів. Провели аналіз видів електронної комерції, таких як B2B, B2C, C2C, B2A, C2A, зосередившись на їх характеристиках і прикладах. Детально вивчили поняття інтернет-магазину, його основні характеристики та переваги для покупців та бізнесу. Не забули про класифікацію інтернет-магазинів за різними критеріями, що дозволяє краще розуміти різноманітність цього сегменту електронної комерції.

У другому розділі про вибір інструментів для реалізації розглянуто основні технології, такі як JavaScript, HTML та CSS, а також популярні фреймворки і бібліотеки, включаючи React, Angular та Vue.js. Особливу увагу приділено React, обраного для проєкту завдяки його зручності, компонентній архітектурі та розвиненій екосистемі. Детально описано бібліотеки для управління станом, маршрутизації, здійснення HTTP-запитів, роботи з картами та візуалізації даних. Для серверної частини обрано Node.js як основу для побудови асинхронного середовища виконання. Розглянуто фреймворки Express.js та Nest.js, що забезпечують ефективне управління запитами та відповідями, а також роботу з базами даних через ORM систему Sequelize [13]. Для забезпечення безпеки використано JWT, а для роботи з об'єктним зберіганням MySQL.

У третьому розділі пояснюється важливість використання Git та GitHub для контролю версій та співпраці між розробниками. PhpStorm виділено як потужний редактор коду, що підтримує багато мов програмування та розширюваність встроєні плагіни. Основна увага приділяється детальному опису функцій вебзастосунку із візуальним представленням, що дозволяє ознайомитись з інтерфейсом та можливостями даного сервісу. Загалом, в даній роботі демонструється системний підхід до розробки вебзастосунку, враховуючи всі аспекти від вибору інструментів до впровадження та тестування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке електронна комерція? [Електрон. ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/elektronna-komertsiya>.
2. Види електронного бізнесу [Електрон. ресурс]. – Режим доступу: https://mpa.vntu.edu.ua/fdb/699/%D0%A2%D0%B5%D0%BC%D0%B0_13.__%D0%92%D0%B8%D0%B4%D0%B8_%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B1%D1%96%D0%B7%D0%BD%D0%B5%D1%81%D1%83..pdf?utm_source=chatgpt.com.
3. Електронна комерція як складова електронного бізнесу. [Електрон. ресурс]. – Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%20%D0%86.%D0%A1.%20%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9%20%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA%20%D0%B7%20%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B1%D1%96%D0%B7%D0%BD%D0%B5%D1%81%D1%83/page8.html.
4. The nature and evolution of JavaScript [Electronic resource]. – Access mode: https://www.theseus.fi/bitstream/handle/10024/353557/Chau_Dao.pdf?sequence=2&isAllowed=y.
5. React.js Documentation [Electronic resource]. – Access mode: <https://react.dev/>.
6. Angular.js Documentation [Electronic resource]. – Access mode: <https://angular.io/docs%20w>.
7. Vue.js Documentation [Electronic resource]. – Access mode: <https://vuejs.org/>.

8. Axios Documentation [Electronic resource]. – Access mode: <https://axios-http.com/>.
9. Express.js Documentation [Electronic resource]. – Access mode: <https://expressjs.com/>.
10. Nest.js Documentation [Electronic resource]. – Access mode: <https://nestjs.com/>.
11. MySQL Documentation [Electronic resource]. – Access mode: <https://dev.mysql.com/doc/>.
12. Best practices для клієнт-серверного проєкта [Електрон. ресурс]. – Режим доступу: <https://habr.com/ru/articles/534084/>.
13. Sequelize Documentation [Electronic resource]. – Access mode: <https://sequelize.org/docs/v6/getting-started/>.

ДОДАТОК А
Текст програми

A.1 Файл Shop.js

```

import React, { useEffect } from 'react';
import cl from '../styles/Home.module.css';
import TypeBar from '../components/TypeBar/TypeBar';
import BrandBar from '../components/BrandBar/BrandBar';
import SneakersList from '../components/SneakersList/SneakersList';

import { useDispatch, useSelector } from 'react-redux';
import { RootState } from '../store';

import { useGetTypesQuery } from '../api/typeApi';
import { useGetBrandsQuery } from '../api/brandApi';
import { useGetSneakersQuery } from '../api/sneakerApi';

import { setTypes } from '../store/typeSlice';
import { setBrands } from '../store/brandSlice';
import { setSneakers } from '../store/sneakerSlice';
const HomePage: React.FC = () => {
  const dispatch = useDispatch();

  const selectedType = useSelector(
    (state: RootState) => state.type.selectedType,
  );
  const selectedBrand = useSelector(
    (state: RootState) => state.brand.selectedBrand,
  );
  const page = useSelector((state: RootState) => state.sneaker.page);

  const { data: types } = useGetTypesQuery();
  const { data: brands } = useGetBrandsQuery();
  const { data: sneakersData } = useGetSneakersQuery({
    type_id: selectedType?.id,
    brand_id: selectedBrand?.id,
    page,
    limit: 8,
  });

  useEffect(() => {
    if (types) {
      dispatch(setTypes(types?.data));
    }
  }, [types?.data, dispatch]);

  useEffect(() => {

```

```

    if (brands) {
      dispatch(setBrands(brands?.data));
    }
  }, [brands?.data, dispatch]);

  useEffect(() => {
    if (sneakersData) {
      dispatch(setSneakers(sneakersData?.data?.sneakers));
    }
  }, [sneakersData?.data, dispatch]);

  return (
    <div className={cl.shopContainer}>
      <div className={cl.sidebar}>
        <TypeBar />
      </div>
      <div className={cl.mainContent}>
        <BrandBar />
        <SneakersList />
        { /* <Pages /> */ }
      </div>
    </div>
  );
};

export default HomePage;

```

A.2 Файл SneakerPage.js

```

import React, {useEffect, useState} from 'react';
import {useParams} from "react-router-dom";
import RatingBigStar from '../img/RatingBigStar.png'
import {getOneSneaker} from "../api/sneakerApi";

const SneakerPage = () => {

  const [sneaker, setSneaker] = useState({info: []})
  const {id} = useParams()
  useEffect(() => {
    getOneSneaker(id).then(data => {
      setSneaker(data)
    })
  }, [])

  return (

```

```

    <Container className="mt-3" style={{paddingTop: '60px'}}>
      <Row>
        <Col md={4}>
          <Image width={300} height={300}
src={process.env.REACT_APP_API_URL + 'uploads/' + sneaker.img}/>
        </Col>
        <Col md={4}>
          <Row className="d-flex flex-column align-items-center">
            <h2>{sneaker.name}</h2>
            <div
              className="d-flex align-items-center justify-content-center"
              style={{background: `url(${RatingBigStar}) no-repeat center
center`, width:100, height: 100, backgroundSize: 'cover', fontSize:34, color:
'black'}}
            >
              {sneaker.rating}
            </div>
          </Row>
import React from 'react';
import { useParams } from 'react-router-dom';
import cl from '../styles/SneakerById.module.css';
// import RatingBigStar from '../img/RatingBigStar.png';
import { useGetOneSneakerQuery } from '../api/sneakerApi';
import { Sneaker } from '../interfaces';

const SneakerPage = () => {
  const { id } = useParams<{ id: string }>();
  const { data, isLoading, isError } = useGetOneSneakerQuery(Number(id));

  const sneaker = data?.data;

  if (isLoading) return <div className={cl.loader}>Loading...</div>;
  if (isError || !sneaker)
    return <div className={cl.error}>Error loading sneaker.</div>;

  return (
    <div className={cl.container}>
      <div className={cl.row}>
        <div className={cl.col}>
          <img
            className={cl.image}
            src={` ${process.env.REACT_APP_API_URL}uploads/${sneaker.img}`}
            alt={sneaker.name}
          />
        </div>
      </div>
    </div>
  )

```

```

<div className={cl.col}>
  <h2 className={cl.title}>{sneaker.name}</h2>
  <div
    className={cl.rating}
    // style={{
    //   backgroundImage: url(`${RatingBigStar}`),
    // }}
  >
    {sneaker.rating}
  </div>
</div>

<div className={cl.col}>
  <div className={cl.card}>
    <h3 className={cl.price}>From: {sneaker.price} €</h3>
    <button className={cl.button}>Add to Basket</button>
  </div>
</div>
</div>

{(sneaker?.sneakersInfo ?? []).length > 0 && (
  <div className={cl.infoSection}>
    <h1>Characteristics</h1>
    {(sneaker?.sneakersInfo ?? []).map((info, index) => (
      <div
        key={info.id}
        className={` ${cl.infoRow} ${index % 2 === 0 ? cl.even : cl.odd}`}
      >
        {info.title}: {info.description}
      </div>
    ))}
  </div>
)}
</div>
);
};

```

```
export default SneakerPage;
```

A.3 Файл Auth.js

```

import React, { useState } from 'react';
import { NavLink, useLocation, useNavigate } from 'react-router-dom';
import { login as loginAction } from '../store/userSlice';
import cl from '../styles/Auth.module.css';

```

```

import { useLoginMutation, useRegistrationMutation } from '../api/userApi';
import {
  HOME_ROUTE,
  LOGIN_ROUTE,
  REGISTRATION_ROUTE,
} from '../utils/constants';
import { useDispatch } from 'react-redux';

const Auth: React.FC = () => {
  const dispatch = useDispatch();
  const location = useLocation();
  const navigate = useNavigate();
  const isLogin = location.pathname === LOGIN_ROUTE;

  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const [loginApi, { isLoading: loginLoading, error: loginError }] =
    useLoginMutation();
  const [registerApi, { isLoading: regLoading, error: regError }] =
    useRegistrationMutation();

  const handleClick = async () => {
    try {
      const user = isLogin
        ? await loginApi({ email, password }).unwrap()
        : await registerApi({ email, password }).unwrap();

      dispatch(loginAction(user));
      navigate(HOME_ROUTE);
    } catch (err: any) {
      alert(err.data?.message || 'Ошибка авторизации');
    }
  };

  const loading = loginLoading || regLoading;

  return (
    <div className={cl.authContainer}>
      <div className={cl.authCard}>
        <h2 className={cl.authTitle}>
          {isLogin ? 'Authentication' : 'Registration'}
        </h2>

        <div className={cl.formGroup}>

```

```

<label htmlFor="email" className={cl.formLabel}>
  Email
</label>
<input
  id="email"
  type="email"
  className={cl.formControl}
  placeholder="Enter your email..."
  value={email}
  onChange={(e) => setEmail(e.target.value)}
  disabled={loading}
/>
</div>

<div className={cl.formGroup}>
  <label htmlFor="password" className={cl.formLabel}>
    Password
  </label>
  <input
    id="password"
    type="password"
    className={cl.formControl}
    placeholder="Enter your password..."
    value={password}
    onChange={(e) => setPassword(e.target.value)}
    disabled={loading}
  />
</div>

<div className={cl.formFooter}>
  <div className={cl.toggleText}>
    {isLoggedIn ? (
      <>
        Don't have an account? {' '}
        <NavLink className={cl.link} to={REGISTRATION_ROUTE}>
          Sign Up!
        </NavLink>
      </>
    ) : (
      <>
        Already have an account? {' '}
        <NavLink className={cl.link} to={LOGIN_ROUTE}>
          Log In!
        </NavLink>
      </>
    )}
  </div>
</div>

```

```

    })
  </div>
  <button
    className={cl.button}
    onClick={handleClick}
    disabled={loading}
  >
    {loading ? 'Please wait...' : isLogin ? 'Log In' : 'Sign Up'}
  </button>
</div>

(({loginError || regError) && (
  <div className={cl.errorText}>
    {(loginError as any)?.data?.message ||
      (regError as any)?.data?.message}
  </div>
)})
</div>
</div>
);
};

export default Auth;

```

A.4 Файл SneakerStorage.js

```

import { configureStore } from '@reduxjs/toolkit';
import typeReducer from './typeSlice';
import brandReducer from './brandSlice';
import sneakerReducer from './sneakerSlice';
import { api } from './api';
import userSlice from './userSlice';

export const store = configureStore({
  reducer: {
    type: typeReducer,
    brand: brandReducer,
    sneaker: sneakerReducer,
    user: userSlice,
    [api.reducerPath]: api.reducer,
  },
  middleware: (getDefault) => getDefault().concat(api.middleware), //
  // опционально для RTK Query
});

```

```
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

A.5 Файл User.js

```
import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { User } from '../interfaces';
```

```
interface UserState {
  user: User | null;
  isAuth: boolean;
}
```

```
const initialState: UserState = {
  user: null,
  isAuth: false,
};
```

```
const userSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    login(state, action: PayloadAction<User>) {
      state.user = action.payload;
      state.isAuth = true;
    },
    logout(state) {
      state.user = null;
      state.isAuth = false;
    },
  },
});
```

```
export const { login, logout } = userSlice.actions;
export default userSlice.reducer;
```

ДОДАТОК Б
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

**Програмна реалізація інтернет-магазину з продажу
кросівок**

ПОЛТАВСЬКИЙ Владислав
ст. групи КНТ-221
ГОФМАН Євгеній
к.т.н., доцент НУ «Запорізька політехніка»



Рисунок Б.1 - Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження: процес створення програмного забезпечення для онлайн бізнесу.

Предмет дослідження: методи створення програмного забезпечення для онлайн бізнесу.

Мета роботи: розробка інтернет магазину з продажу кросівок.

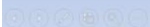


Рисунок Б.2 - Слайд 2

Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Функціональні вимоги:

- перегляд товарів;
- реєстрація користувачів;
- додавання товарів до кошика;
- видалення товарів з кошика;
- оформлення замовлення;
- фільтрація товарів за певним критерієм;
- оплата товару;
- вибір товарів за категорією;

Рисунок Б.3 - Слайд 3

Порівняння мов програмування

Критерії порівняння	JavaScript	TypeScript	Python
Ліцензія	-	Apache License 2.0	Python Software Foundation License
Простота навчання та написання коду	+	+-	+-
Наявність зрозумілої та повної документації	+	+	+
Сильна підтримка спільноти	+	+	+-
Наявність популярних фреймворків	+	+	+-
Продуктивність	+	+	+

Рисунок Б.4 - Слайд 4

Порівняння середовищ розробки

Критерії порівняння	Visual Studio Code	PhpStorm	Sublime Text
Наявність безкоштовної версії(open source)	++	+-	+-
Підтримка бібліотек та налаштувань	++	++	+-
Кросс-платформеність	++	++	++
Швидкість роботи	+	+-	++
Підтримка та оновлення	++	++	+

Рисунок Б.5 - Слайд 5

Mvc схема для користувача

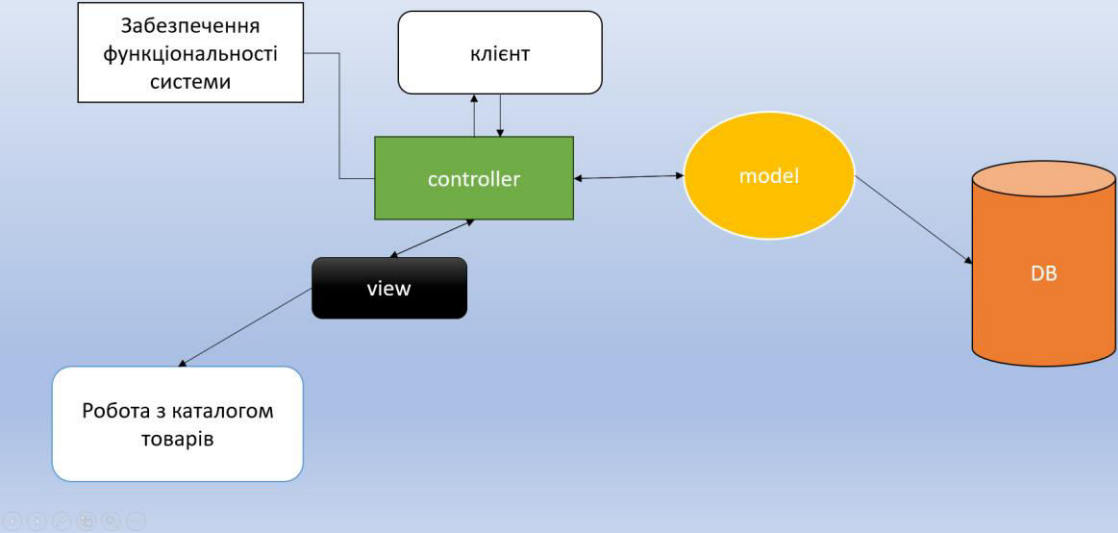


Рисунок Б.6 - Слайд 6

Схема бази даних

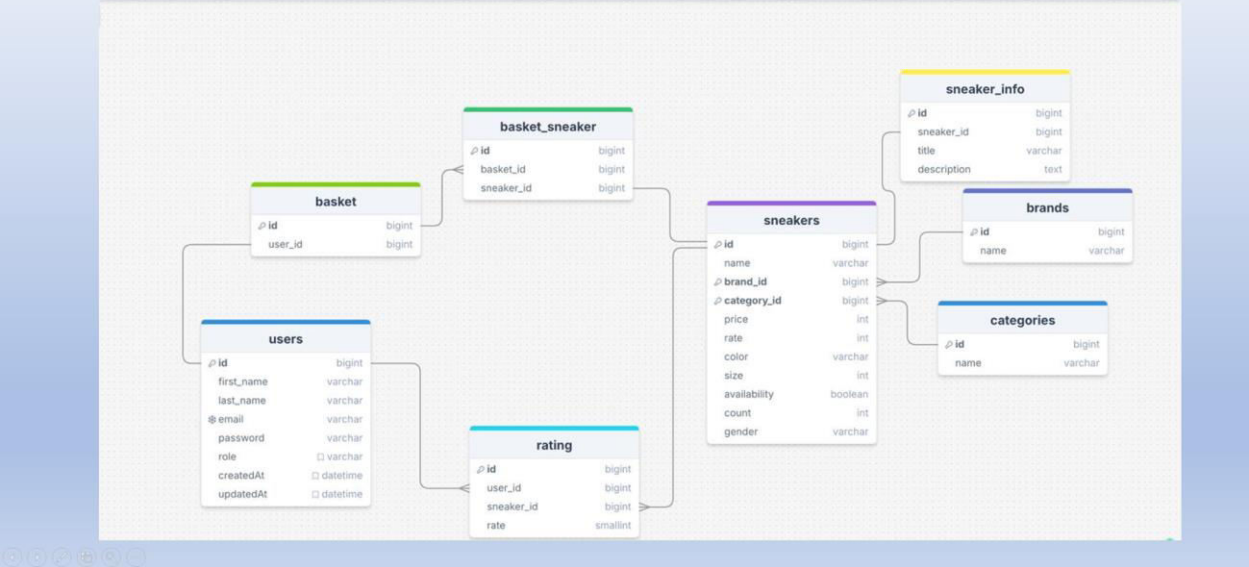


Рисунок Б.7 - Слайд 7

Приклади сторінок веб застосунку

The image shows a registration form with the following elements:

- Registration** (Section Header)
-
-
- Already have an account? [Log In!](#)
-

Рисунок Б.8 - Слайд 8

Приклади сторінок веб застосунку

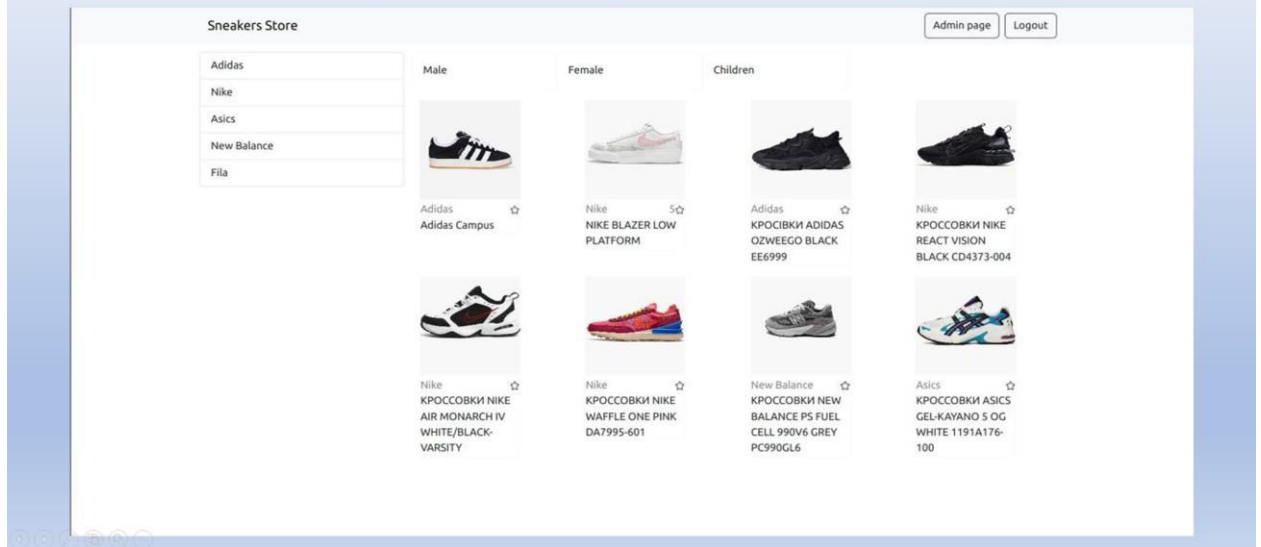


Рисунок Б.9 - Слайд 9

Висновки

В ході роботи було виконано наступні завдання:

- Розглянуто проблемну область;
- Проаналізовано сучасні аналоги онлайн-магазинів продажу кросівок;
- Виконано проектування веб-застосунку;
- Розроблено веб застосунок;

Розроблений застосунок виконує наступні функції:

- Реєстрація та авторизація усіх груп користувачів;
- Ознайомлення із каталогом товарів;
- Оформлення замовлення;
- Оновлення каталогу товарів;

В результаті виконання завдання було отримано систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Рисунок Б.10 - Слайд 10



Рисунок Б.11 - Слайд 11