НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: Веб-застосунок для продажу спортивних товарів

Виконав : студент __4__ курсу, групи _ІО-92_
(шифр групи)
_____Уткін Владислав Антонович_____
(прізвище, ім’я, по батькові) (підпис)

Керівник ____асистент кафедри ОТ Пономаренко А.М._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. викладач Виноградов Ю.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент __доц. каф. ІСТ, к.т.н., доц. Алла Коган_____
(посада, фнауковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Уткіна Владислава Антоновича

1. Тема проєкту *Веб-застосунок для продажу спортивних товарів*
керівник проєкту Пономаренко Артем Миколайович, асистент кафедри ОТ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 31 травня 2023 року №2101-с

2. Термін здачі студентом закінченого проєкту *8 червня 2023 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз предметної області.

Розділ 2. Огляд та порівняння технологій для створення веб-застосунків.

Розділ 3. Проектування та розробка веб-застосунку.

Розділ 4. Дослідження розробленого веб-застосунку.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) компоненти програмного забезпечення (структурна схема), діаграма бази даних (функціональна схема), алгоритм дій програмного забезпечення (принципова схема).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	ст. викладач, Виноградов Ю.М.		

7. Дата видачі завдання «14» березня 2023 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*09.03.2023-14.03.2023*	
2.	*Вивчення та аналіз завдання*	*14.03.2023-23.03.2023*	
3.	*Розробка архітектури та загальної структури системи*	*23.03.2023-07.04.2023*	
4.	*Розробка структур окремих підсистем*	*07.04.2023-17.04.2023*	
5.	*Програмна реалізація системи*	*17.04.2023-24.04.2023*	
6.	*Оформлення пояснювальної записки*	*28.04.2023-16.05.2023*	
7.	*Захист програмного продукту*	*25.05.2023*	
8.	*Передзахист*	*07.06.2023*	
9.	*Захист*	*21.06.2023*	

Студент-дипломник _____ Владислав УТКІН
(підпис)

Керівник проєкту _____ Артем ПОНОМАРЕНКО
(підпис)

АНОТАЦІЯ

У даній роботі було детально розглянуто сферу розробки веб-застосунків. Досліджено основні технології та інструменти для розробки веб-застосунку. Після вибору технологій для розробки було спроектовано та розроблено веб-застосунок для продажу спортивних товарів на мові JavaScript з використанням середовища Node.js, фреймворка Express та бібліотеки React.js та бази даних MongoDB. Розроблений веб-застосунок надає користувачу ряд можливостей: перегляд та фільтрування товарів, додання товарів в кошик, реєстрація та авторизація, використання панелі управління для додання товарів та категорій. Проведено аналіз розробленого веб-застосунку.

ANNOTATION

In this project for a Bachelor`s Degree, the field of web application development was considered in detail. The main technologies and tools for developing a web application are studied. After selecting development technologies, a web application for selling sporting goods was designed and developed in JavaScript using Node.js environment, Express framework and React.js library and MongoDB database. The developed web application provides the user with a number of options: viewing and filtering products, adding products to the cart, registration and authorization, using the control panel to add products and categories. The developed web application was analyzed.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Веб-застосунок для продажу	3		
			спортивних товарів			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Веб-застосунок для продажу	80		
			спортивних товарів			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Веб-застосунок для продажу	1		
			спортивних товарів			
			Компоненти програмного			
			забезпечення			
			(Структурна схема)			
	А4	***ІАЛЦ.467200.005 Д2***	Веб-застосунок для продажу	1		
			спортивних товарів			
			Діаграма бази даних			
			(Функціональна схема)			
	А4	***ІАЛЦ.467200.006 Д3***	Веб-застосунок для продажу	1		
			спортивних товарів			
			Алгоритм дій програмного			
			забезпечення			
			(Принципова схема)			
	А4	***ІАЛЦ.467200.007 Д4***	Веб-застосунок для продажу	15		
			спортивних товарів			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА		
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*			
Розроб		Уткін В.А.			*Веб-застосунок для продажу спортивних товарів* Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Перев		Пономаренко А.М.					
						КПІ ім. Ігоря Сікорського ФІОТ ІО-92	

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-застосунок для продажу спортивних товарів*»

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання стосується розробки веб-застосунку для продажу спортивних товарів, а також подальшої підтримки та вдосконалення розробленого продукту.

Областю застосування є реалізація продажу в Інтернеті спортивних товарів різноманітними компаніями, торговими підприємствами.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даного проекту є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», який був затверджений факультетом "Інформатики та обчислювальної техніки" кафедрою обчислювальної техніки Національного технічного університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка веб-застосунку для продажу спортивних товарів. У проекті реалізований необхідний функціонал для ефективної взаємодії користувача з онлайн-магазином.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Мати зрозумілий та зручний інтерфейс.
- Надати можливість користувачам переглядати та фільтрувати товари.
- Надати можливість користувачам робити пошук товарів за назвою.
- Надати можливість користувачам додавати товари в кошик.
- Надати можливість користувачам реєструватися та авторизуватися в системі.
- Надати можливість адміністратору додавати товари та нові категорії в магазин.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Доступ до мережі Інтернет.
- Веб-браузер за вибором.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core i3-2125 3.30GHz.
- ROM не менше ніж 16 ГБ.
- RAM не менше ніж 4 ГБ.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	09.03.2023-14.03.2023
Вивчення та аналіз завдання	14.03.2023-23.03.2023
Розробка архітектури та загальної структури системи	23.03.2023-07.04.2023
Розробка структур окремих частин системи	07.04.2023-17.04.2023
Програмна реалізація системи	17.04.2023-24.04.2023
Виправлення помилок	24.04.2023-28.04.2023
Оформлення пояснювальної записки	28.04.2023-16.05.2023

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-застосунок для продажу спортивних товарів*»

Київ – 2023

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	База даних
CGI	(Common Gateway Interface) Загальний шлюзовий інтервейс
WAP	(Wireless Application Protocol) Протокол бездротових програм
HTML	(HyperText Markup Language) Мова розмітки гіпертексту
CSS	(Cascading Style Sheets) Каскадні таблиці стилів
MVC	(Model-View-Controller) Модель-вид-контролер
PWA	(Progressive Web Applications) Прогресивний веб-застосунок
SQL	(Structured Query Language) Структурована мова запитів
JSON	(JavaScript Object Notation) Запис об’єктів JavaScript
JWT	(JSON Web Token) Веб-токен JSON
URL	(Uniform Resource Locator) Стандартизований вказівник на ресурс

ВСТУП

В сучасному технологічному світі рівень популярності електронної комерції, як форми покупки, важко переоцінити. На сьогодні, складно уявити будь-яку компанію, спеціалізованій на продажі товарів, яка б не мала свій онлайн-ресурс для реалізації покупок своїми клієнтами. Популярності використанню онлайн-магазинів сприяє їх доступність для користувачів. Достатньо лише мати пристрій з підключенням до Інтернету та встановленим браузером для того, щоб почати взаємодіяти з улюбленим магазином.

З факту про популярність випливає велика конкуренція в сфері веб-розробки онлайн-магазинів. Фахівці з розробки намагаються задовільнити потреби продавців, реалізовуючи продукт високої якості. Веб-розробники прибігають до використання більш продвинутих засобів та технологій, які допомогають створити сучасний та якісний онлайн-магазин.

Метою данного дипломного проекту є реалізація веб-застосунку, який буде виконувати задачі спортивного онлайн-магазина. В ході виконання проекту будуть досліджені теоритичні аспекти веб-розробки, так і веб-технологій загалом. На основі отриманих знань, будуть прийняті рішення по розробці власного веб-застосунку. Додаток повинен мати простий та зрозумілий інтерфейс, в якому користувач може виконувати цілий ряд дій по взаємодії з наповненням магазину. Функціонал веб-застосунку повинен відповідати сучасним вимогам роботи онлайн-магазина.

Розроблений продукт повинен мати потенціал до широкого використання в сфері електронної комерції. Виконання данного дипломного прокту повинно посприяти розвитку моїх навичок та знань в цій сфері.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія розвитку веб-технологій

Для того, щоб отримати більше глибоке розуміння специфіки та проблематики предметної області потрібно дослідити історію веб-технологій. Це допоможе проаналізувати виклики, які стояли перед розробниками на протязі різних етапів розвитку даної сфери. В свою чергу, дана інформація буде корисною для мене, з точки зору розуміння актуальних задач в сфері веб-розробки, які потребують вирішення.

Протягом останніх десятиліть індустрія веб-технологій пройшла багато етапів та подій, які стали визначальними для розвитку сфери.

1.1.1 Протокол HTTP

Протокол HTTP (Hypertext Transfer Protocol) був розроблений Тімом Бернерс-Лі, який вважається одним з батьків Всесвітньої павутини (World Wide Web). У 1989 році, під час роботи в Європейському організаційному інституті для дослідження ядра фізики частинок (CERN), Бернерс-Лі створив систему гіпертексту, яка називалася WorldWideWeb [1].

WorldWideWeb була поєднанням гіпертекстових документів, веб-серверів та веб-браузерів, які дозволяли користувачам зв'язувати інформацію між різними сторінками за допомогою гіперпосилань. Протокол HTTP був розроблений для забезпечення зв'язку між веб-серверами і веб-браузерами.

Перша версія протоколу HTTP, відома як HTTP/0.9 [1], була простою і неповною специфікацією. Вона дозволяла веб-клієнтам (веб-браузерам) зробити запит до веб-сервера і отримати статичну HTML-сторінку. HTTP/0.9 не

підтримувала заголовки запиту та відповіді і не мала можливості передавати інші типи даних, крім HTML.

Поява HTTP є надзвичайно важливою по багатьом причинам, проте я вважаю, що головна - це стандартизація комунікації між веб-клієнтами та веб-серверами. Це дозволило передавати та відображати веб-сторінки на будь-якому пристрої, який підтримую протокол.

1.1.2 CGI та динамічні веб-сторінки

У 1993 році було розроблено Common Gateway Interface (CGI) [2]. CGI - це стандартний протокол взаємодії між веб-сервером і програмою, що генерує динамічний контент. Використання CGI дозволяє веб-серверу виконувати програму та генерувати відповідь, яку він потім надсилає клієнтові (веб-браузеру). За допомогою CGI можна створювати динамічні веб-сторінки, які змінюються в залежності від параметрів, переданих користувачем або іншої інформації.

При використанні CGI, веб-сервер отримує запит від клієнта, виконує вказану програму CGI і передає результат веб-браузеру. Зазвичай програма CGI написана мовою програмування, такою як Perl, Python, PHP або C++, але практично будь-яка мова може бути використана для написання програм CGI.

Дана технологія відкрила двері для подальшої розробки веб-додатків. В свій час це був популярним засобом для створення динамічних веб-сторінок. В сучасному світі розробники застосовують більш продвинуті технології, такі як серверні скриптові мови або фреймворки веб-розробки. Однак, розуміння CGI є дуже важливим для розуміння історії розвитку динамічних веб-сторінок та веб-розробки в цілому.

1.1.3 JavaScript

JavaScript є однією з найпопулярніших скриптових мов програмування, що використовуються для розробки динамічних веб-сторінок і взаємодії на стороні клієнта.

JavaScript був розроблений Бренданом Ейхом в 1995 році під час його роботи в компанії Netscape Communications [3]. Перша назва мови була LiveScript, але згодом була змінена на JavaScript, щоб використовувати популярну назву "Java" через її популярність.

В 1997 році JavaScript був стандартизований організацією Ecma International і отримав назву ECMAScript. Перший стандарт ECMAScript був прийнятий під назвою ECMAScript 1 [3].

Поява JavaScript є дуже важливою для розвитку веб-технологій по багатьом причинам. Я можу виділити основні з них:

- Взаємодія на стороні клієнта: JavaScript дозволив веб-розробникам створювати динамічні та інтерактивні елементи на веб-сторінках, що дозволило користувачам зручно взаємодіювати з інтерфейсом.
- Багатофункціональні веб-додатки: Стало можливим створення повноцінних веб-додатків зі складними функціями, такими як інтерактивні форми, мультимедійні елементи, картографія, графіки, анімації та інше. Завдяки JavaScript інтерфейси користувача дійсно стали багатофункціональними, що приблизило їх в цій характеристиці до десктопних аналогів.
- Розширення можливостей браузерів: Завдяки JavaScript, браузери отримали можливість виконувати складні операції та обробляти багатофункціональні додатки. Мова допомогла розширити можливості веб-браузерів, включаючи підтримку мультимедіа, графіки, мережевої взаємодії та інших функцій.

- Крос-платформеність: JavaScript став стандартом для розробки веб-додатків та наразі підтримується практично всіма сучасними веб-браузерами на різних платформах і пристроях.

1.1.4 WEB 2.0

WEB 2.0 - це поняття, яке використовується для опису другого етапу розвитку Інтернету та веб-технологій. Дане поняття не характеризує якусь конкретну технологію чи мову програмування. WEB 2.0 є результатом поступового розвитку та еволюції Інтернету та веб-технологій, що відбувався протягом кількох років у період з кінця 1990-х по 2000-і роки. Воно охоплює набір нових підходів, технологій та функціональності, які з'явилися в цей проміжок часу.

Існує багато факторів, які посприяли виникненню та розвитку WEB 2.0:

- Зростання швидкості та доступності інтернету
- Розвиток мов програмування та технологій
- Поява соціальних мереж та спільнот
- Збільшення ролі користувача в формуванні змісту веб-сторінок (поява форумів, блогів, вікі-платформ і т.д)
- Розширення функціональності веб-додатків

Загалом, я вважаю що, WEB 2.0 створив основу для подальшого розвитку веб-технологій, особливо з точки зору соціальної взаємодії в онлайн-середовищі.

1.1.5 Mobile Web

Зростання популярності смартфонів та планшетів звісно напряму вплинуло на сферу веб-технологій. Веб-розробники починають зосереджувати свою

увагу на розробці мобільних веб-додатків та адаптації веб-сайтів для мобільних пристроїв.

Якщо казати про зародження мобільної веб-розробки то, мабуть, варто згадати про WAP (Wireless Application Protocol). WAP – це технологія, яка використовувалася для отримання доступу до інтернет-ресурсів за допомогою мобільного телефона.

WAP був розроблений консорціумом під назвою WAP Forum, який був утворений у 1997 році. Цей форум об'єднав такі компанії, як Nokia, Ericsson, Motorola і Phone.com [4].

Основним завданням WAP було створити єдиний стандарт інтернет-доступу на бездротових пристроях. Створення спеціальної технології, яка б забезпечувала ефективний Інтернет-доступ було нетривіальним завданням, адже мобільні пристрої мали свою специфіку з обмеженою кількістю ресурсів.

WAP включав в собі такі основні компоненти:

- WML(Wireless Markup Language): Мова розмітки, подібна до HTML, яка створювала веб-сторінки, адаптовані під мобільні пристрої. В порівнянні із сучасним HTML мав дуже обмежені можливості в розробці.
- WAP-протоколи: спеціальні протоколи, які виконували задачі передачі даних, керування транзакціями та забезпечення безпеки
- WAP-браузери: спеціальні браузери, які функціонували на мобільних пристроях.

WAP був запущений у 1999 році та здобув значну популярність на той час[4].

В порівнянні із сучасними рішеннями мав очевидні недоліки, насамперед це повільна швидкість передачі даних, складний дизайн та обмежені можливості. За допомогою сучасних технологій розробники створюють набагато функціональніші мобільні веб-додатки та сайти. Однак, аналогічно до інших технологій, які ставали “першопроходцями” в своїх сферах

веб-розробки, WAP заклав фундамент для подальшого розвитку мобільної веб-розробки.

1.1.6 HTML5 та CSS3

Незважаючи на те, що історія HTML та CSS, яка бере свій початок біля витоків всієї індустрії веб-технологій, дуже велика та фундаментальна, я вирішив виділити саме останні версії цих технологій в якості окремого етапу в історії веб-розробки.

Саме HTML5 та CSS3 на початку 2010-х років стали новими стандартами для створення веб-сторінок. Дані версії додали багато нових можливостей для розробників, що безумовно створило новий поштовх для розвитку веб-розробки.

Серед причин, чому поява цих версій є важливою, я би виділив такі:

- Розширення можливостей веб-розробки: HTML5 та CSS3 додають багато нових функцій і можливостей, які дозволяють розробникам створювати більш динамічні, інтерактивні та привабливі веб-сторінки
- Сумісність: HTML5 та CSS3 сприяють створенню веб-сторінок, які легше адаптуються до різних пристроїв і екранів. Вони підтримують респонсивний дизайн, що дозволяє сторінкам автоматично змінювати свій вигляд в залежності від розміру екрану.
- Ефективність: Впровадження рішень, які дозволяють створювати більш швидкодіючі, та загалом, більш ефективніші веб-сторінки. Відбувається зменшення завантаження та покращення прискорення роботи веб-додатка за рахунок ефективного використання ресурсів пристрою.

1.1.7 Веб-фреймворки та бібліотеки

На мою думку, саме веб-фреймворки та бібліотеки формують сучасні реалії світу веб-розробки. Сьогодні вони відіграють надзвичайно важливу роль оптимізуючи процес створення веб-додатків.

Простими словами, бібліотеки та фреймворки – це певний код, який вже написаний іншим розробником, який використовується для того щоб вирішити певні однотипні завдання [5].

Бібліотека – це набір функцій, класів, методів, які здатні вирішити конкретну задачу в програмі. Тобто бібліотеки надають готові рішення для виконання певних завдань. Вони допомогають розробникам використовувати такі бібліотеки замість того, щоб самому з нуля писати весь код.

В свою чергу, фреймворки відрізняється від бібліотек тим, вони використовуються для того щоб створити архітектуру програми. Тобто, фреймворки виступають, так би мовити, “скелетом” програми.

До очевидних переваг у використанні фреймворків та бібліотек я можу віднести:

- Швидкість розробки: готові рішення та компоненти допомогають не витрачати час на написання всього функціоналу програми
- Організація та структура: встановлюються загальні правила , за якими слід будувати веб-додаток, що полегшує його розробку, підтримку та масштабування.
- Зменшення дублювання коду: Однотипні завдання вже реалізовано в бібліотеках, тому розробник не повинен писати цей код знову. Це значно підвищує ефективність розробки.
- Велика спільнота: популярність фреймворків та бібліотек посприяла збільшенню бази спільноти, що в свою чергу збільшує кількість корисних ресурсів та форумів, де можна отримати допомогу.

- Надійність та безпека: через активність спільноти, фреймворки та бібліотеки проходять велику кількість тестувань та перевірок. Це допомогає швидко знаходити та виправляти баги та несправності.

1.2 Веб-застосунки

1.2.1 Принцип роботи та архітектура

Дуже часто в спільноті, особливо серед новачків у сфері, постає питання, чим відрізняється веб-застосунок від веб-сайта. Насправді, ці два поняття мають набагато більше відмінностей ніж може здатися на перший погляд.

Веб-сайт – це певна кількість веб-сторінок, які носять здебільшого інформативний характер. Взаємодія користувача з цими сторінками дуже обмежена. Як правило, вміст цих сторінок включає текст та різні медіа-файли.

Веб-застосунок – це вже повноцінна программа з широким функціоналом та інтерактивними елементами, доступ до якої здійснюється через браузер[6]. В цьому випадку, на відміну від веб-сайтів взаємодія з користувачем відіграє головну роль. Користувач є важливим учасником в формуванні контенту веб-застосунка. Такі програми виконують багато важливих функцій такі як: обмін інформації, проведення транзакцій, дистанційна купівля та продаж товарів і т.д

З цієї специфіки веб-застосунків випливає більш складний принцип роботи та архітектура. Отже, давайте більш детально розберемося, як працюють веб-застосунки.

По-перше, потрібно розуміти, що веб-застосунок це багатоскладове програмне забезпечення. Воно складається з великої кількості компонентів, такі як інтерфейс користувача, база даних, екран регістрації/авторизації і т.д.. Тому щоб ефективно керувати всіма ціма компонентами, розробники створили архітектуру для веб-додатків. Така архітектура логічно визначає зв’язки та спосіб взаємодії між складовими застосунка.

Приклад архітектури веб-застосунку зображений на рис. 1.1

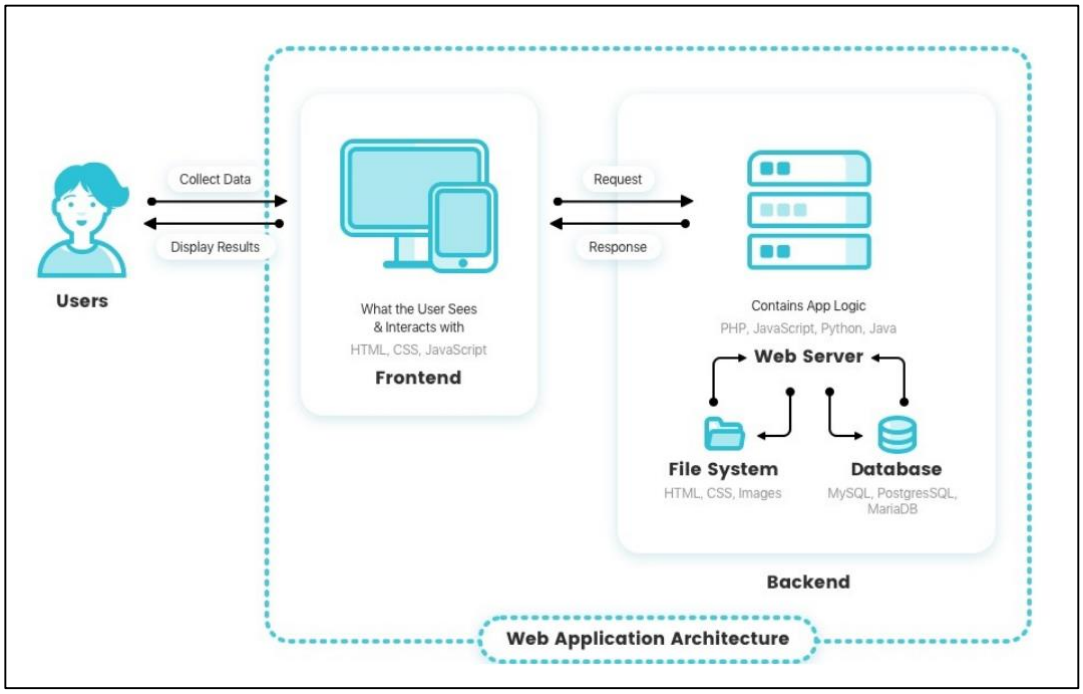


Рисунок 1.1 – Архітектура веб-застосунка [7]

Як можна побачити, кожен веб-застосунок з Front-end(клієнтської частини) та Back-end(серверної частини).

Front-end – це все те, що користувач бачить, та те з чим він може взаємодіяти у своєму браузері. Основним завданням клієнтської частини – є збір інформації та даних від користувача. В самому простому варіанті Front-end пишеться за допомогою HTML, CSS та JavaScript.

Back-end – це, в свою чергу, сторона, яка недоступна користувачу. Основним завданням є зберігання та оброблення даних. Сервер обробляє HTTP-запити, які можна сказати “витягують” дані, яких потребує користувач. В написанні Back-end розробники використовують різні мови програмування. Як правило, це PHP, Java, Python, JavaScript і т.д

Також, є третя фундаментальна складова будь-якого веб-додатка – це база баних. Вона забезпечує зберігання та організацію даних, необхідних для функціонування застосунка. Взаємодія з базою даних відбувається через веб-сервер.

Таким чином, ці три складові формують скелет нашого веб-додатка. Головною задачею при створенні застосунка є правильне налагодження зв’язків між цими трьома частинами.

Для того, щоб краще зрозуміти, як функціонує веб-застосунок, я наведу приблизні кроки його роботи:

- Користувач відправляє запит до веб-серверу, в який входить центр керування додатком та бази даних
- Сервер отримує запит та направляє його до бази даних
- Потрібна інформація дістається з бази даних та передається до центру керування
- Після цього вебсервер надсилає клієнту дану інформацію

Типи архітектур веб-застосунків

Я описав загальні принципи та архітектуру стандартного веб-застосунку, які формують загальний фундамент його роботи. Однак в сучасній розробці все трохи ускладнилось. Різна функціональність, логіка додатка викликає потребу в обранні спеціальної архітектури, яка максимально ефективно змогла б вирішити загальну мету продукта. Загалом, на сьогодні можна виділити такі типи архітектур:

- Single Page Application Architecture
- Microservice Architecture
- Serverless Architecture
- Progressive Web Application

Пропоную детально розглянути кожну з них[8]:

1) Single Page Application Architecture:

Така архітектура, як не складно здогадатися, пов’язана з таким типом веб-застосунків, як SPA(Single Page Application). SPA – це застосунок, який взаємодіює з користувачем, шляхом переписування поточної веб-сторінки новими даними з веб-сервера, замість того, щоб завантажувати цілі нові сторінки.

Такі застосунки ставлять за мету вирішити класичні проблеми створення плавних програми, для того щоб забезпечити користувачу інтуїтивно зрозумілий та простий у використанні досвід взаємодії із застосунком.

2) Microservice Architecture

Особливість цієї архітектури полягає в невеликих та легких службах, які виконують одну функцію. Такий підхід має ряд переваг, які дозволяють розробникам підвищити продуктивність та пришвидшити весь процес розгортання. Така архітектура включає в собі компоненти, які не залежать один від одного безпосередньо. Це дозволяє розробникам використовувати різні мови програмування для створення цих компонентів. Таким чином, розробники, які використовують архітектуру мікросервісів, можуть вільно обирати стек технологій, що значно спрощує та пришвидшує розробку

3) Serverless Architecture

Serverless архітектура характеризується підходом до розробки програмного забезпечення, в якому розробникам не потрібно стежити за серверами та інфраструктурою, оскільки інфраструктурні деталі покладаються на провайдера хмарних послуг. У Serverless архітектурі основний акцент робиться на функціях, які виконуються при виникненні певних подій.

Serverless архітектура може бути використана для реалізації різноманітних веб-застосунків, таких як мікросервіси, обробка подій, потокова обробка даних та інші сценарії. Популярні провайдери хмарних послуг, такі як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) та інші, пропонують рішення для розгортання безсерверних застосунків.

4) Progressive Web Applications

Таким тип архітектури пов’язаний із так званими прогресивними веб-додатками(PWA). Головною метою при створенні PWA було розробити багатофункціональні додатки із розширеними можливостями, надійністю та простотою інсталювання. Такі додатки здатні працювати на будь-якому пристрої та будь-якому браузері

1.2.2 Переваги та недоліки веб-застосунків

Веб-застосунки мають свої переваги та недоліки. Пропоную детально їх розглянути для того щоб мати поглиблене розуміння в цій області.

Переваги:

- Кросплатформеність: Веб-застосунки доступні через веб-браузер, що дозволяє їх використовувати на різних операційних системах, таких як Windows, macOS, Linux і т. д. Розробникам не потрібно створювати окремі версії для кожної платформи.
- Легкий доступ: Користувачам не потрібно встановлювати спеціальне програмне забезпечення, оскільки веб-застосунки доступні через браузер. Це зменшує бар'єри для використання і спрощує поширення та розгортання програм.
- Постійне оновлення: Розробники можуть оновлювати веб-застосунки безпосередньо на сервері, і оновлення автоматично відображатимуться для користувачів при наступному завантаженні сторінки. Це дозволяє швидко внести зміни та виправити помилки без необхідності оновлення самого програмного забезпечення на кожному пристрої користувача.
- Зручність роботи в розподіленому середовищі: Веб-застосунки дозволяють користувачам працювати в розподіленому середовищі, де дані зберігаються на серверах. Це забезпечує легкий обмін і спільну роботу над даними між користувачами з різних місць.
- Широкі можливості функціоналу: Веб-додатки можуть надавати широкий спектр функціоналу, включаючи роботу з базами даних, обробку форм, розсилку електронних листів, інтеграцію з зовнішніми сервісами і багато іншого. Це дозволяє реалізувати різноманітні бізнес-потреби та забезпечити багатофункціональність додатків.

- Низькі витрати: Розробка веб-додатків зазвичай вимагає менших витрат порівняно з розробкою настільних або мобільних додатків, оскільки вони не вимагають окремої розробки для кожної платформи.

Недоліки:

- Залежність від Інтернету: Веб-застосунки потребують підключення до Інтернету для доступу до сервера та отримання даних. Без Інтернету вони можуть бути недоступні або функціонувати обмежено.
- Обмежена функціональність: У порівнянні з настільними програмами, веб-застосунки можуть мати обмежену функціональність, особливо якщо для виконання певних завдань потрібний доступ до системних ресурсів пристрою.
- Безпека: Веб-застосунки піддаються ризику вразливостей, таких як хакерські атаки, введення шкідливого коду, крадіжка даних і т. д. Розробники повинні приділяти особливу увагу безпеці, використовуючи захисні заходи для захисту від таких загроз.
- Обмеження доступу до апаратного забезпечення: Веб-застосунки мають обмежений доступ до апаратного забезпечення пристрою, такого як камера, мікрофон, сенсори. Це може обмежити можливості функцій, які вони можуть надавати.

На мою думку, незважаючи на недоліки, веб-додатки є оптимальним рішенням для багатьох задач, як з технічної точки зору, так і з точки зору бізнесу.

Ті переваги, які мають веб-застосунки, допомагають вирішувати актуальні задачі, які постають сьогодні перед розробниками.

1.3 Огляд існуючих спортивних онлайн-магазинів

На сьогоднішній день існує багато веб-додатків спортивних онлайн-магазинів, які надають користувачам можливість придбати спортивний одяг, взуття, аксесуари та спортивне обладнання. Зробимо огляд деяких веб-додатків спортивних онлайн-магазинів:

- Terrasport.ua (головна сторінка магазину зображена на рис. 1.2)

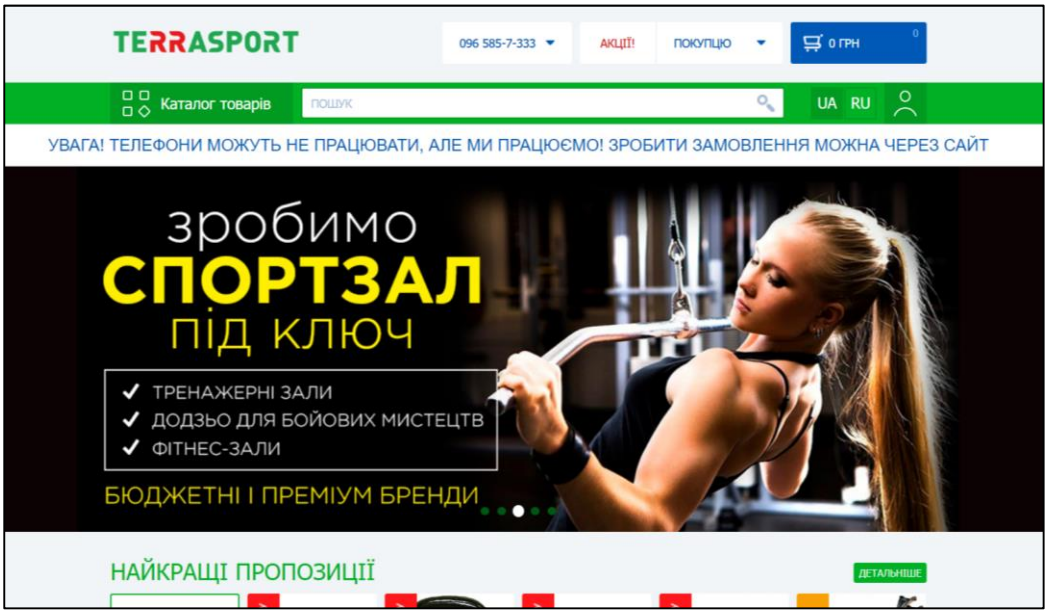


Рисунок 1.2 – Головна сторінка онлайн-магазину Terrasport [10]

З інформації про компанію Terrasport на сайті можна винести наступну корисну інформацію:

- Інтернет-магазин створений в 2008 році ініціативною групою людей, які мають великий досвід в спортивній індустрії
- Поточна версія сайту є четвертою з моменту створення магазина та відповідає всім вимогам безпеки та зручності використання для клієнтів. Сайт постійно вдосконалюється.
- Онлайн-магазин має дуже широкий асортимент товарів, що налічує 80000 товарів
- Компанія має високу репутацію на ринку

Отже, перейдемо до огляду самого веб-додатку:

- На головній сторінці можна побачити велику кількість блоків з різною інформацією. На мій погляд, інтерфейс сторінки є нагромадженим, а блоки мають різну форму, колір. Тому мені, як користувачу, важко швидко зорієнтуватися в інтерфейсі.
- Перейшовши до вкладки «каталог товарів» можна побачити велику кількість категорій, яку можна обрати, що свідчить про величезний вибір товарів. Сторінка зображена на рис. 1.3

Рисунок 1.3 – Каталог товарів онлайн-магазину Terrasport [10]

- Обравши певну категорію товарів, переходимо на сторінку, де можна дуже зручно фільтрувати товари за такими категоріями, як тип, бренд, ціна, матеріал.
- Потрібні товари можно додати в кошик, або в категорію «улюблене»
- В шапці онлайн-магазина можна обрати номер телефона(для зв’язку із службою підтримки), вміст кошика покупця, корисну інформацію для користувача.
- В веб-додатку зроблена функція авторизації та реєстрації. Авторизація передбачає застосування різних соціальних мережей при потребі користувача

- У нижній частині сайту є посилання на корисну інформацію про доставку, оплату, контакти, співробітництво, графік роботи
- Серед особливостей сайту, які рідко можна побачити в інших аналогах, це: окремий блок на головній сторінці для перегляду корисних статей на тему здорового образу життя, а також книга скарг та пропозицій
- На головній сторінці також можна побачити інформацію про тимчасові проблеми з роботою телефонів. В контексті роботи веб-додатка, я вважаю це плюсом, адже працівники дбають про те, щоб надавати своїм клієнтам актуальну інформацію про проблеми.

Загалом веб-додаток онлайн магазину Terrasport можна оцінити позитивно. До основних переваг можу віднести його функціональність, великий вибір товарів. А серед недоліків: незручність інтерфейсу

- Nike.com (Головна сторінка сайту зображена на рис. 1.4)

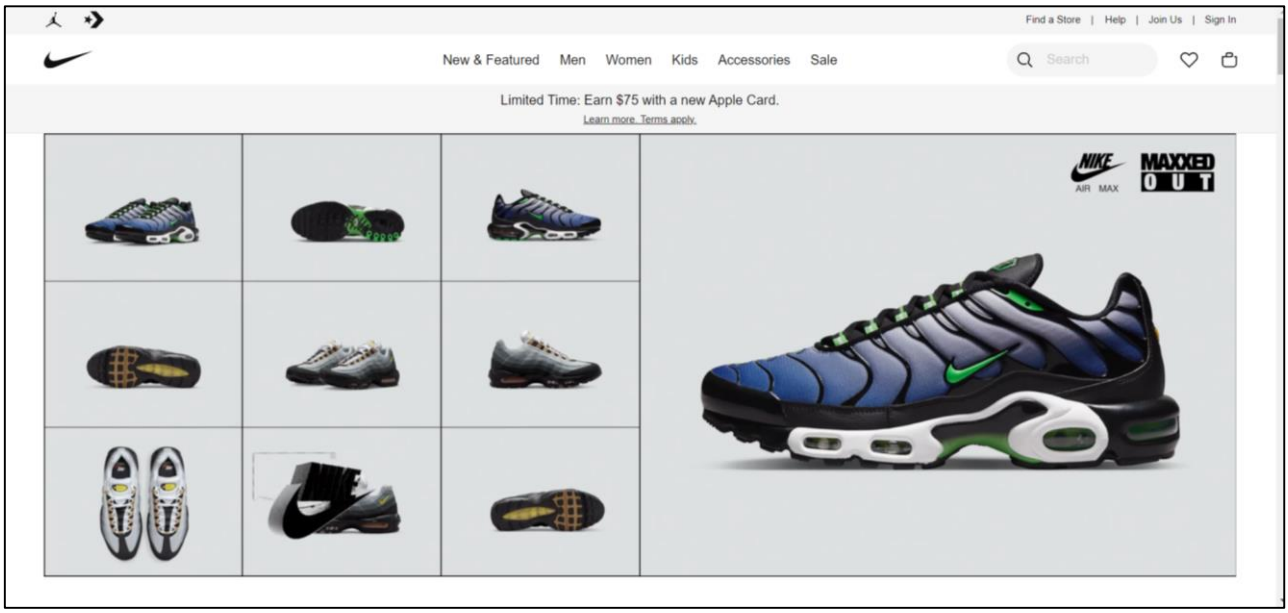


Рисунок 1.4 – Головна сторінка онлайн магазину Nike.com [11]

При обиранні наступного веб-додатка в якості аналога, я вирішив обрати онлайн-магазин всесвітньо відомого бренду Nike. Таким чином, я зможу порівняти два веб-додатки від абсолютно різних по статусу та бюджетом компаній, щоб зрозуміти чи сильно впливає фінансова складова на якість розробленого веб-додатка.

Для аналізу веб-застосунку я зроблю перелік речей, які, з моєї точки зору, є спільними з попереднім аналогом, а також ті, які різні.

Спільне:

- Широкий асортимент продуктів: Nike.com пропонує величезний вибір спортивного одягу та взуття для різних видів спорту.
- Зручний пошук та фільтрація: Веб-додаток має потужну систему пошуку, що дозволяє швидко знаходити потрібні товари за категоріями, стилями, розміром, кольором та іншими параметрами. Користувачі можуть також застосовувати фільтри для точного вибору товарів згідно зі своїми вимогами.
- Створення облікового запису: Користувачі можуть створити обліковий запис на Nike.com, що дозволяє зберігати адреси доставки, переглядати історію замовлень та зберігати обрані товари.
- Зручні способи оплати та доставки: Веб-додаток надає різні варіанти оплати, включаючи платіжні картки, електронні гроші та інші електронні платіжні системи.

Різне:

- Докладна інформація про товари: Кожен товар на Nike.com супроводжується детальним описом, фотографіями з різних кутів, технічними характеристиками та відгуками користувачів.
- Можливість персоналізації: Nike.com пропонує можливість персоналізувати деякі товари, наприклад, взуття або футболки, додавши власну емблему, ім'я або інші деталі.
- Відео та інтерактивний контент: Nike.com пропонує відеоматеріали та інтерактивний контент, які допомагають користувачам краще ознайомитися з продуктами, дізнатися про їх технології та властивості, а також навчитися використовувати їх на практиці.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі було детально розглянуто предметну область дипломної роботи. Була досліджена історія розвитку веб-технологій, а саме ключові етапи та технології, які зробили максимальний внесок в еволюцію веб-розробки. Виходячи із проаналізованих етапів, можно зробити висновок про сучасний стан речей в світі веб-розробки, а саме актуальність розробки веб-застосунків із застосуванням фреймворків та бібліотек.

Було обрано веб-застосунки в якості предмету подальшого дослідження. Розібрали принципи роботи та архітектуру веб-додатка. Зрозуміли його специфіку. Проаналізувавши переваги та недоліки веб-додатків, можна впевнитися в правильності обрання застосунку в якості інструмента вирішення тих задач, які постають переді мною в дипломній роботі.

Після цього, було розглянуто два існуючих рішення в області “веб-додатків для спортивних онлайн-магазинів”. Після результату дослідження можна зробити висновок про схожість багатьох функцій та компонентів. Один з аналогів мав кращі характеристики в плані візуальної складової, інтерактиву, взаємодії з користувачем.

Ціль першого розділу виконана. Я отримав знання в області тематики моєї роботи. Зрозумів масштаб сфери веб-розробки. Отримав інформацію про роботу сучасного веб-додатка. Досліджені аналоги сформували для мене картину, які задачі повинен вирішувати веб-застосунок для онлайн-магазину спортивних товарів. В наступному розділі я розгляну технології за допомогою яких я буду вирішувати ці задачі.

РОЗДІЛ 2. ОГЛЯД ТА ПОРІВНЯННЯ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-ЗАСТОСУНКІВ

В даному розділі розглянемо основні технології для створення веб-застосунка. Розділимо технології на 4 категорії:

- Front end
- Back end
- Бази даних
- Додаткові технологій

В кожній категорії розглянемо рішення, порівняємо їх та оберемо технології для розробки відповідно до поставленого завдання

2.1 Огляд Front-end технологій

Правильний вибір технологій для клієнтської частини є надзвичайно важливим аспектом для розробки веб-застосунка. Такі технології допомогають реалізувати взаємодію з користувачем, створити приємний зовнішній вигляд програми, а також покращити ефективність всього застосунку. Дизайн, структура, анімації та все, що бачить користувач створюється за допомогою таких технологій

- Базові технології

Існує три основні технології, які формують фундамент front-end частини:

- HTML(HyperText Markup Language): Мова маркування, основа будь-якого веб-додатка, застосовується для маркування та створення структури.
- CSS(Cascading Style Sheets): Відповідає за візуальну складову. Використовується для оформлення сторінок, задання стилю елементам

- JavaScript: мова програмування, яка використовується для створення динаміки на веб-сторінці. Використання JavaScript вирішує завдання валідації форм, анімації та багато іншого

HTML, CSS, JavaScript – це ключові технології в веб-розробці та широко використовуються для створення веб-застосунка.

Однак, з часом з'явилися нові технології, фреймворки і інструменти, які розширили можливості веб-розробки і допомагають створювати більш потужні та складні веб-додатки. Деякі з цих технологій використовуються разом з HTML, CSS і JavaScript, доповнюючи їх функціональність. Такі технології полегшують роботу веб-розробників та роблять процес створення додатка простішим та ефективнішим.

Отже, давайте детально розглнемо технології без яких важко уявити сучасну розробку.

- Фреймворки

Існує три найбільш популярних фреймворка, які використовуються в більшості веб-проектах:

- React
- Angular
- Vue

На рис. 2.1 зображено графік порівняння популярності цих фреймворків

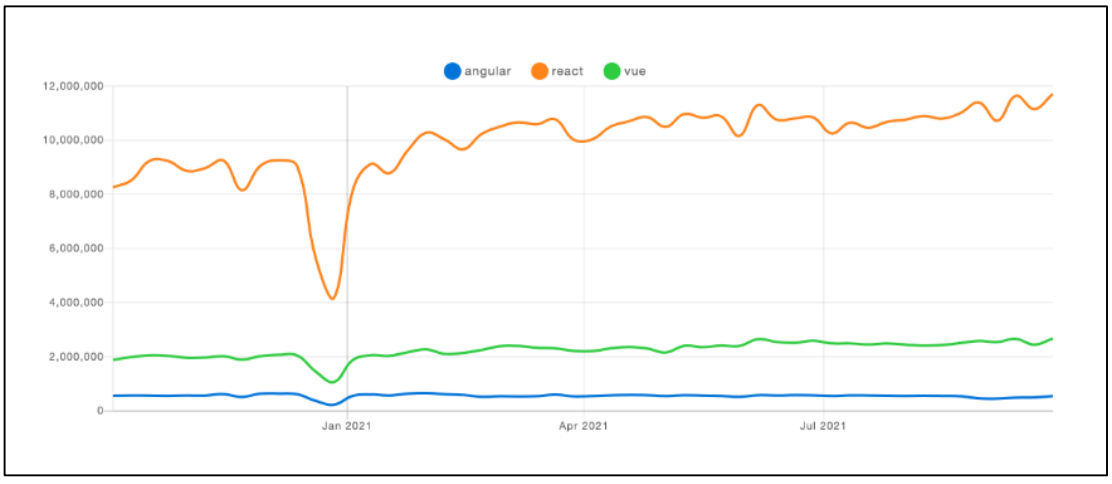


Рисунок 2.1 – Порівняльний графік завантаження фреймворків[12].

За даними популярної серед девелоперів платформи Stack Overflow найбільш використовуваним фреймворком є React, його використовують 35,9% розробників. На другому місці йде Angular – 25.1%, а на останньому Vue – 17.3% [12]. Можна помітити доволі близькі значення, що свідчить про неодностайність думок веб-розробників

Однак, npm тренди показують трохи іншу картину. З рисунку вище, можна побачити беззаперечну перевагу React. Проте, що цікаво, в цьому разі Vue переганяє Angular за скачуваннями.

Тому для того, щоб краще зрозуміти, який фреймворк використовувати для клієнтської частини проекту, пропоную детально розглянути дані технології.

1) Angular був створений в 2009 році компанією Google [12]. На сьогодні відомий, як JavaScript фреймворк. Angular – це MVC фреймворк, тобто підтримує архітектуру Model-view-controller

Переваги:

- Підтримка архітектури MVC
- Ремонтнопридатність
- Дозволяє керувати архітектурою мікроінтерфейсу
- Просте блокове тестування
- Angular-CLI(командний рядок) підвищує швидкість розробки проекту

Недоліки:

- Для роботи з ним повинен певний період навчання
- Великий обсяг коду
- Використання великої кількості ресурсів для завантаження
- Відносно негнучкий через свою структуру

2) React був випущений Facebook у 2013 році в якості бібліотеки JavaScript [12]. Даний фреймворк на відміну від Angular не підходить для побудови архітектури MVC. Його найбільшою перевагою є те, що він

використовує віртуальний DOM, що дуже позитивно впливає на час завантаження.

Переваги:

- Швидке завантаження даних
- В одному файлі можна розмістити, як логіку, так і розмітку(JSX)
- Дуже легкий поріг входу, не потребує багато часу на навчання
- Легкість інтеграції з різними технологіями

Недоліки:

- Фактично, це лише бібліотека, а не фреймворк
- Відсутня можливість застосувати архітектуру MVC

3) Vue – фреймворк на основі JavaScript, який використовується для створення так званих SPA(Single-Page Applications) програм. На відміну від інших фреймворків, Vue придатний до поступового впровадження. Ядро Vue в першу чергу вирішує задачі рівня представлення, що допомогає його інтегрувати з іншими проекатми або бібліотеками. Vue – це неймовірно гнучкий фреймворк. Я би охарактеризував цей фреймворк, як поєднання гарних сторін Angular та React.

Переваги:

- Багатий вибір інструментів
- Гнучкість, простота в застосуванні
- Найлегший в навчанні
- Детальна документація

Недоліки:

- Невелика спільнота, в порівнянні з React та Angular
- Невелика кількість плагінів та бібліотек
- Спільнота здебільшого неангломовна

Підсумувавши даний підрозділ можу зробити висновок, що для Front-end частини свого веб-додатка я буду використовувати React

Серед причин такого рішення:

- Наявність досвіду в використанні
- Велика спільнота
- Швидке завантаження даних(дуже важливо при роботі онлайн-мазагина)
- Легкість інтеграції

2.2 Огляд Back-end технологій

Вибір технологій для серверної частини веб-додатка дуже важливий. Обрані рішення будуть відповідати за багато задач, такі як: обробка запитів користувача, взаємодія з базою даних, надання необхідних даних для фронтенду і т.д

Розглянемо різні мови програмування та фреймворки у галазі back-end розробки, порівняємо їх та оберемо технології для власної розробки.

2.2.1 PHP та фреймворк Lavarel

PHP – дуже поширена скриптова мова з відкритим вихідним кодом. Мова має довгу історію використання в веб-розробці та надає широкий набір інструментів та функцій, спрямованих на обробку запитів, взаємодію з базами даних, роботу з файлами та інші завдання серверної частини веб-додатку[13].

Переваги PHP:

- Синтаксис і простота використання: PHP має синтаксис, легкий для вивчення і використання. Вона також пропонує широкий набір

функцій та бібліотек, які полегшують розробку та прискорюють процес створення веб-застосунків.

- Широке співтовариство розробників: PHP має велику та активну спільноту розробників, яка надає підтримку, документацію, пакети та багато іншого. Це означає, що можна легко знайти рішення для багатьох проблем або підключити готові компоненти для розширення функціональності вашого додатку.
- Підтримка баз даних: PHP має вбудовану підтримку для різних систем управління базами даних, таких як MySQL, PostgreSQL, SQLite і багато інших. Ви можете легко взаємодіяти з базами даних, виконувати запити, отримувати та зберігати дані.
- Рентабельність: завдяки наявності безкоштовних спільнот із відкритим кодом програмувати за допомогою PHP досить зручно.

Недоліки:

- Падіння популярності: На сьогодні рідко можна побачити розробників, які хочуть почати вчити цю мову.
- Відсутність розширених бібліотек робить PHP менш привабливим в порівнянніз з іншими сучасними технологіями
- Відкритий вихідний код призводить до неправильного використання помилок в програмі

Найпопулярнішим фреймворком PHP – є Laravel. Дана технологія вважається однією з найкращих для back-end розробки. Серед особливостей, можу відзначити здатність надавати різні способи доступу до реляційних баз даних, а також наявність утиліт для розгортання та обслуговування додатків.

Переваги:

- Відкритий код фреймворка
- Модульна архітектура для створення масштабованих програм
- Використання архітектуру MVC для організації коду
- Вбудований інтерфейс командного рядка

- Вбудована система аутентифікації та реєстрації користувачів

Недоліки:

- Повільний розвиток в порівнянні з іншими аналогами
- Обмежена вбудована підтримка
- Ризиковані та неоднозначні оновлення
- Велика кількість компонентів може зробити його важким для опанування.

2.2.2 Python та фреймворк Django

Мова програмування Python є дуже популярним вибором для back-end розробки. Завдяки різноманітим стандартним бібліотекам можна спростити весь процес розробки. Інструменти веб-сервісів, протоколи, бібліотеки дозволяють розробникам застосовавати готові скрипти, що допомогає скоротити час розробки.

Однією з головних причин, чому Python обирають для розробки серверної частини – є високий рівень безпеки. За допомогою проекту OWASP(Open-Source Web Application Security Project for Python) розробники мають можливість створити захищену версію програми та зменшити вразливість до атак [14].

Також, перевагою Python є стисліть та лаконічність написання коду. Дана мова програмування є простішою в порінянні з іншими серверними технологіями.

Найпопулярнішим фреймворком для розробки веб-застосунків з виористанням Python – є Django. Фреймворк надає повний набір інструментів для створення складних та потужних веб-додатків, включаючи систему маршрутизації, ORM (Object-Relational Mapping) для взаємодії з базами даних, систему аутентифікації, адміністративний інтерфейс та багато іншого [15].

Переваги використання Django:

- Швидкість розробки
- Масштабованість
- Безпзпека
- Адміністративний інтерфейс

Недоліки:

- Розгортання усіх компонентів одночасно
- Відсутня можливість обробки декількох запитів одночано
- Відсутня вбудована підтримка створення API

2.2.3 Java та фреймворк Spring

Java – добре відома об’єктно-орієнтована мова програмування. В сучасній розробці має широке застосування для створення десктопних та мобільних застосунків, для розробки back-end, для обробки великої кількості даних.

За данмим компанії Oracle, яка володіє Java, кількість пристроїв, які використовують дану мову налічує близько 3 мільярдів. Це робить Java однією з найпопулярніших мов програмування.

Серед причин використання Java для розробки back-end можна виділити такі[16]:

- Масштабовність та стійкість:

  Мова є довговічною через надійний механізм перевірки типів. Java може працювати будь-де через динамічне зв’язування та безпечне середовище, яке забезпечує віртуальна машина. За допомогою автоматичного керування пам’яттю можна досягти масштабованості.

- Бібліотека з відкритим кодом:

  Багато бібліотек Java є безкоштовними та мають відкритий вихідний код. Застосування подібних бібліотек пришвидшує розробку.

Бібліотеки можуть виконувати різні функції, такі як: парсинг JSON, обмін повідомленнями, криптографія і т.д

- Різноманітність:

Через велику популярність мови розширилися варіанти її використання, починаючи додатками для обробки даних, закінчуючи програмами машинного навчання

- Безпека:

Java зменшує ризики порушення безпеки, через свою специфіку, при якій зменшується кількість випадків небажаного доступу до пам’яті.

- Незалежність платформи:

Зкомпільований код не залежить від обраної платформи та має здатність працювати будь-де, незалежно від операційної системи. Код можна запустити на будб-якій машині, яка може підтримувати JVM. Схема такої роботи показана на рис.2.2

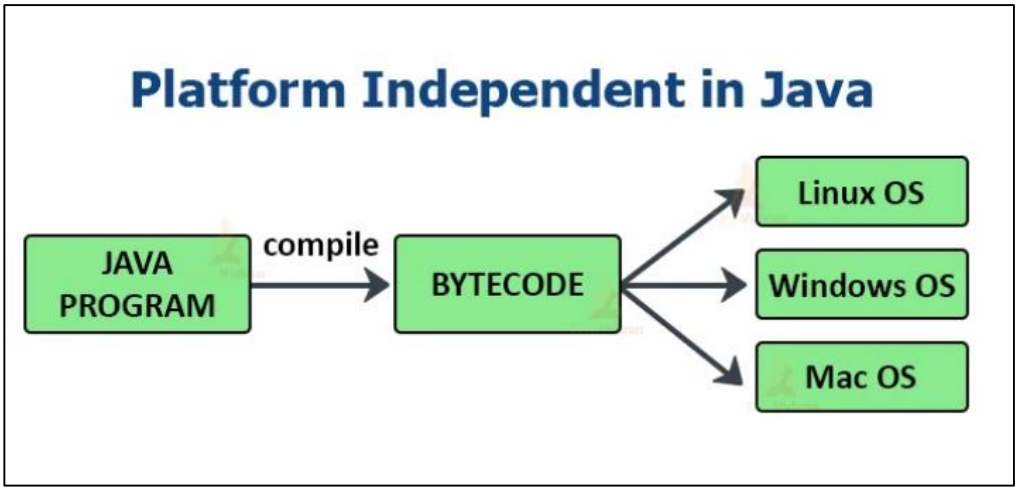


Рисунок 2.2 – Незалежність платформи в Java [16]

До недоліків Java можна віднести:

- Відсутність резервного копіювання:

Мова акцентує увагу лише на зберіганні даних

- Складність коду:

Код має великий розмір та багато складних конструкцій, що знижує читабельність коду

- Швидкість та рівень продуктивності:

  Багато рівнів компіляції та абстракції, а також потреба в інтерпретації біт коду в код машинного рівня негативно впливають на загальну продуктивність

- Ємність пам’яті:

  В порівнянні із іншими аналогами мова використовує великий обсяг пам’яті, що впливає на ефективність та продуктивність.

Найкращим фреймворком Java для розробки back-end – є Spring. Spring надає потужні інструменти та компоненти, які допомагають розробникам побудувати масштабовані, надійні та ефективні системи[17].

Можна виділити декілька причин, чому варто використовувати Spring для back-end розробки:

- Інверсія керування (IoC) та впровадження залежностей (DI):

  Spring базується на принципах Інверсії керування та впровадження залежностей, що сприяють створенню слабо зв'язаних компонентів і полегшують тестування та розширення системи. Ви можете використовувати DI, щоб легко внедрювати залежності між класами та змінювати їх безпосередньо, що сприяє гнучкості та обмежує залежність від конкретних реалізацій.

- Spring Boot:

  Spring Boot - це частина екосистеми Spring, яка спрощує налаштування та розгортання Spring додатків. Він надає конвенції над конфігурацією, що дозволяє розробникам швидко створювати самостійні, готові до використання додатки з мінімальними зусиллями. Spring Boot також включає в себе вбудований контейнер для управління веб-додатками, що полегшує їх розгортання та виконання.

- Архітектурні шаблони та рішення:

  Spring пропонує різні архітектурні шаблони та рішення, такі як MVC, RESTful веб-сервіси, обробка подій та багато інших. Ці шаблони

допомагають розробникам побудувати добре організовані та розширюван

- Інтеграція з іншими технологіями:

Spring інтегрується з багатьма іншими технологіями та бібліотеками, що дозволяє вам використовувати різноманітні інструменти для розробки веб-додатків. Наприклад, ви можете використовувати Spring Data для роботи з базами даних, а Spring Security для забезпечення безпеки.

2.2.4 Ruby та фреймворк Ruby on Rails.

Ruby – це високорівнева мова програмування. Отримала велику популярність завдяки своєму фреймворку Ruby on Rails, який почали часто використовувати для розробки веб-додатків.

Головною особливістю фреймворка є його структура, яка надає розробникам можливість використовувати визначені рішення для виконання повторюваних завдань.

Розглянемо головні аспекти фреймворку Ruby on Rails[17] :

- Принципи “Convention over Configuration”

Такий підхід означає використання фреймворком стандартних конфігурацій та шаблонів для того, щоб зменшити потребу в ручній конфігурації. Такий підхід дозволяє швидше приступати до роботи над проектом.

- Архітектура MVC:

Архітектура Model-View-Controller дозволяє розділити логіку застосунка на окремі частини. Застосування такої архітектури спрощує розробку та підтримку веб-додатків

- Active Record

Фреймворк використовує такий принцип по роботі з базами даних, що дозволяє класам моделей мати методи для повної взаємодії із записами у базі даних. Такий підхід спрощує роботу з базами даних

- Restful маршрутизація

Така маршрутизація дозволяє створювати логічні URL-шаблони для взаємодії з різними діями додатку

- Швидкість розробки

Фреймворк покладається на покращення зручності та надає багато інструментів для більш швидкої розробки.

2.2.5 JavaScript, середовище Node.js та фреймворк Express

В попередньому розділі, при огляді технологій для розробки Front-end мною була розглянута мова JavaScript. Дана мова займає передову позицію в розробці клієнтської частини. Однак, в останні роки веб-розробники дедалі частіше використовують JS для розробки back-end. Такі зміни відбулись за рахунок розвитку технологій та інструментів на базі JavaScript. Пропоную детально познайомитися з такими технологіями.

- Node.js

Node.js – це відкрита платформа для виконання JavaScript на сервері. В основі нього лежить двигун V8, який є виконавчим середовищем для JavaScript, розробленим Google для веб-переглядача Chrome [18]. Однак Node.js використовує V8 для виконання JavaScript на сервері замість веб-браузера.

Node.js має відкритий код, таким чином розробники можуть працювати з ним безпкоштовно. Платформа має велику спільноту розробників, які постійно продовжують розвивати її

Головною особливістю даної платформи є можливість запускати програми на мові JS окремо від браузера.

Про неймовірну популярність платформи свідчить рис.2.3, на якому можна побачити, що node.js займає перше місце за популярністю серед всіх технологій за даними порталу stack overflow[19].

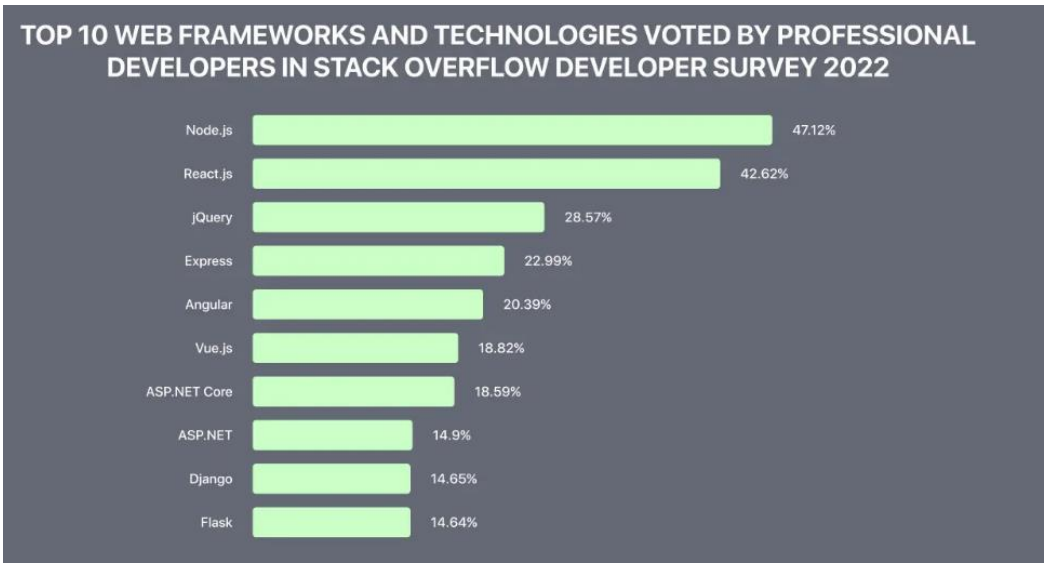


Рисунок 2.3 – Графік популярності веб-технологій за 2022 рік[19]

Можна виділити такі плюси в використанні node.js[19]:

- Висока продуктивність: Node.js побудований на основі подій та неблокуючого вводу/виводу (non-blocking I/O), що дозволяє досягти високої продуктивності та ефективності в обробці багатьох запитів одночасно.
- Єдина мова програмування: JavaScript є однією з найпоширеніших мов програмування, яка використовується як на стороні клієнта (фронтенд), так і на стороні сервера (бекенд).
- Швидка розробка: Node.js пропонує зручні інструменти та бібліотеки, які допомагають прискорити процес розробки. Наявність пакетного менеджера npm дозволяє легко встановлювати та використовувати готові модулі та бібліотеки для різних завдань.
- Масштабованість: Node.js добре підходить для побудови масштабованих систем. Він підтримує горизонтальне масштабування, тобто можливість додавати більше серверів для обробки більшого навантаження.

- Активна спільнота: Node.js має велику та активну спільноту розробників, яка постійно внесе внески в екосистему Node.js.

- Express

Express – це компактний та швидкий фреймворк Node.js, який надає всі необхідні інструменти розробки серверних програм. Express допомогає спростити розробку веб-додатка на стороні сервера, дозволяючи легко створювати динамічні сторінки[20].

ExpressJS дозволяє розробникам використовувати різні бібліотеки та пакети, які виступаютб в ролі плагінами. Це робить процес розробки швидшим, ефективнішим і менш схильним до помилок.

2.3 Огляд баз даних

База даних є дуже важливою складовою розробки та адміністрування веб-застосунка. Головною функцією бази даних – є зберігання та організація даних відвовідним чином для того щоб іх можна було швидко опрацювати та отримати у відповідь на запит користувача. Також база даних повинна гарантувати безпеку та конфіденційність

На ринку існує велика кількість баз даних, які можуть вирішувати такі завдання. Тому варто розділити різні бази даних за типом, а також розглянути конкретні рішення.

Загалом, існує два типи баз даних:

- SQL: Реляційні бази даних з реляційною структурою. Такий тип підходить для структурованих даних
- NoSQL: Відсутня реляційна структура. Підходить для взаємодії із неструктурованими даними

В табл. 2.1 описані головні відмінності між SQL та NoSQL базами даних.

Таблиця 2.1 – Порівняльна таблиця типів баз даних

Типи баз даних	
SQL	**NoSQL**
Закртий вихідний код	Відкритий вихідний код
Реляційна структура	Відстутня реляційна структура
Підходить для структурованих даних	Підходить для неструктурованих даних
Вертикально масштабована	Горизонтально масштабована
Фіксована схема	Динамічна схема

На сьогодні існує багато якісних баз даних для вирішення тих задач, які від них вимагають. Пропоную розглянути дві найпопулярніші з них(одна SQL, інша NoSQL) [21].

1) MongoDB – це документоорієнтована NoSQL база даних, придатна до масштабування. Використовує JSON-подібні документи з динамічними схемами для спрощення зберігання та запиту даних. Дуже гарне рішення для неструктурованих даних

Особливості:

- Кожна база даних має колекції, які в свою чергу зберігають документи
- Кожен документ може мати різний розмір, зміст та кількість полів
- Структура документа подібна до того, як розробники конструюють програмний код
- Ряди можуть не мати якусь конкретну схему
- Є можливість відтворювати ієрархічні відношення

2) MySQL – це реляційна база даних, непридатна до масштабування. Має складності із зберіганням даним через фіксовану схему. Однак, таке рішення добре підійде до структурованих даних, наприклад пов`язаних з фінансами.

Особливості:

- Може бути інтегрована з різними платформами
- Підтримка управління декількома версіями паралелізма
- Багатошаровий дизайн з незалежними модулями
- Повністю багатопотоковий
- Наявні вбудовані інструменти для аналіза запитів
- Можливість обробляти будь-який об’єм даних

ВИСНОВОК ДО РОЗДІЛУ 2

В другому розділі мною були розглянуті технології для вирішення задачі розробки веб-додатка. Були досліджені конкретні рішення в таких областях, як: Front-end, Back-end та бази даних.

В області Front-end розробки були розглянуті та порівняні три головних фреймворка: React, Angular та Vue. Після результату аналізу технологій, прийняте рішення, що для розробки власного магазину мною обрано фреймворк React. Причини такого рішення були обгрунтовані.

В області Back-end розробки були розглянуті найбільш популярні мови програмування та відповідні фреймворки та платформи. З представлених технологій для свого веб-застосунку мною прийняте рішення обрати Node.js та Express. Таке рішення обгрунтовано бажанням застосовувати одну мову програмування для розробки, що допоможе легко об’єднати проект в цільну структуру. Платформа Node.js має велику спільноту та має інструменти для пришвидшення процесу розробки. Фреймворк Express також відповідає моїм вимогам для розробки серверної частини проекту.

Після огляду та порівнняння рішень в області баз даних, мною зроблено вибір – використати MongoDB в якості бази даних мого веб-застосунку. Таке рішення обумовлене бажанням використовувати неструктуровані дані в форматі документів.

Таким чином, підсумовуючи все вище сказане, обрано стек технологій для розробки веб-застосунка, а саме:

- React
- Node.js
- Express
- MongoDB

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ЗАСТОСУНКА

3.1 Проектування роботи веб-застосунка

Враховуючи результати попередніх досліджень можна сказати, що сфера розробки онлайн-магазанів дуже розвинута. Після перегляду існуючих аналогів можна зробити висновок, що сучасні онлайн-магазини спортивних товарів мають весь необхідний функціонал для зручної та ефективної роботи з ними. Таким чином, ставити за мету – перевершити за функціоналом подані аналоги вважаю недоцільним. Вважаю правильним рішенням, зосередити увагу на розробці конкретних складових онлайн-магазина, які будуть відповідати потребам користувача.

Перелік функцій, які буде виконувати розроблений веб-застосунок:

- Перегляд товарів
- Фільтрування товарів за категоріями та цінами
- Пошук товара за назвою
- Додання обраних товарів в кошик
- Реєстрація та авторизація користувачів
- Розподілення функціоналу веб-застосунка за ролями(Користувач та Адміністратор)
- Додання категорій та товарів, редагування існуючих товарів адміністратором
- Редагування інформації про свій профіль користувачем

- Принцип роботи веб-застосунка

Для того, щоб описати принцип роботи власного веб-додатка пропоную розглянути блок-схему в якій описується весь алгоритм функціонування. На рис.3.1 зображена дана блок-схема.

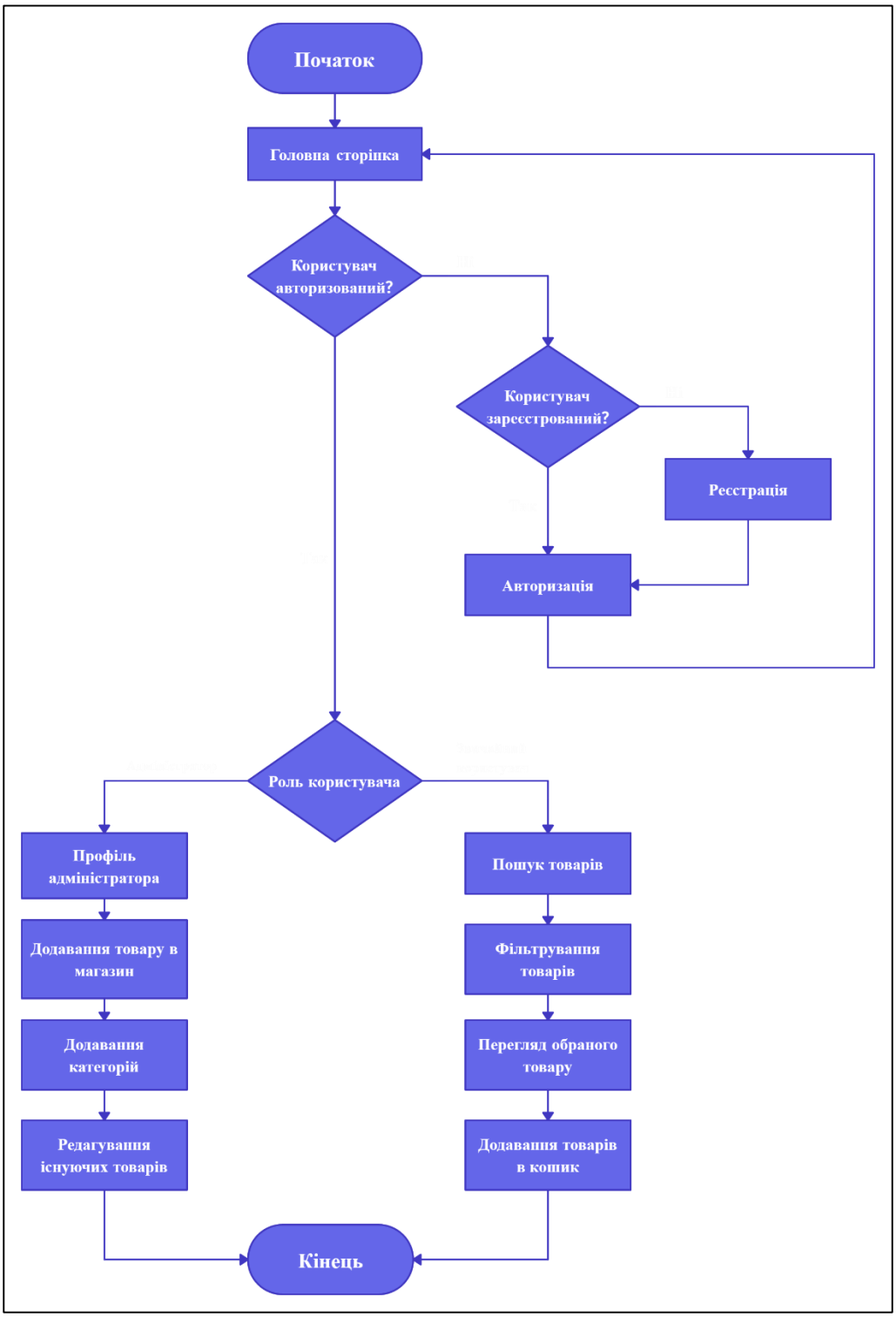


Рисунок 3.1– Блок-схема алгоритму дій

3.2 Розробка та реалізація серверної частини

- **Вибір та завантаження пакетів для розробки**

Для того щоб зробити розробку серверної частини більш продуктивною та зручною будемо використовувати спеціальні пакети з бібліотеками, утилітами та драйверами.

Такі пакети будуть важливим інструментом для вирішення таких задач, як: маршрутизація, валідація даних, робота з базою даних і т.д.

Перелік пакетів, які будемо використовувати:

- Axios : Бібліотека для здійснення HTTP-запитів з Node.js або з веб-браузера.
- body-parser: Middleware для розбору (парсингу) тіла запиту.
- concurrently: Утиліта для одночасного виконання декількох команд.
- config: Бібліотека для керування конфігурацією проекту.
- cookie-parser: Middleware для обробки cookie.
- cors: Middleware для налаштування політики обмеження розподілу ресурсів між різними доменами.
- dotenv: Бібліотека для завантаження змінних середовища з файлу .env.
- express: Фреймворк для створення веб-додатків на Node.js.
- express-jwt: Middleware для перевірки та обробки JWT-токенів.
- express-validator: Middleware для перевірки та валідації даних запиту.
- formidable: Бібліотека для обробки форм та завантаження файлів.
- jsonwebtoken: Бібліотека для генерації та перевірки JWT-токенів.
- lodash: Утилітарна бібліотека, яка надає розширені функції для маніпулювання даними.
- mongodb: Офіційний драйвер MongoDB для Node.js.
- mongoose: ORM для роботи з MongoDB, який надає спрощений доступ до бази даних та визначення моделей.

- nodemon: Утиліта для автоматичного перезапуску сервера при зміні файлів у проекті.
- uuid: Бібліотека для генерації унікальних ідентифікаторів.

3.2.1 Робота з базою даних

Для цього проеку, як було зазначено мною раніше, була використана база даних – MongoDB.

Для роботи була використана MongoDB Atlas – хмарна платформа для керування базами даних MongoDB. Зареєструвашись на офіційному ресурсі був створений кластер для зберіганні та обробки даних. На рис.3.2 зображений створений кластер.

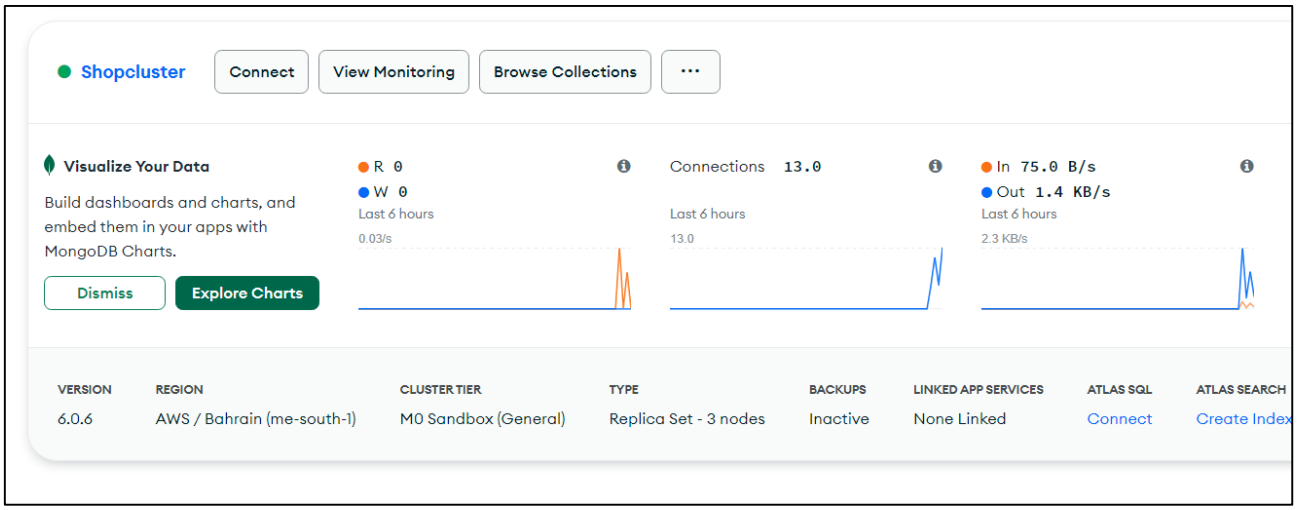


Рисунок 3.2 – Кластер бази даних MongoDB

Даний кластер містить базу даних, яка в свою чергу зберігає колекції. Кожна колекція відповідає за зберігання та обробку окремого виду даних.

Мій проект має 3 колекції:

- Товари
- Категорії
- Користувачі

Колекції містять в собі документи, в кожному з яких міститься інформація про конкертний товар, категорію чи користувача. Приклад збергання документу зображений на рис. 3.3

Рисунок 3.3 – Зберігання документів в базі даних

На рис 3.4 зображена загальна структура бази даних

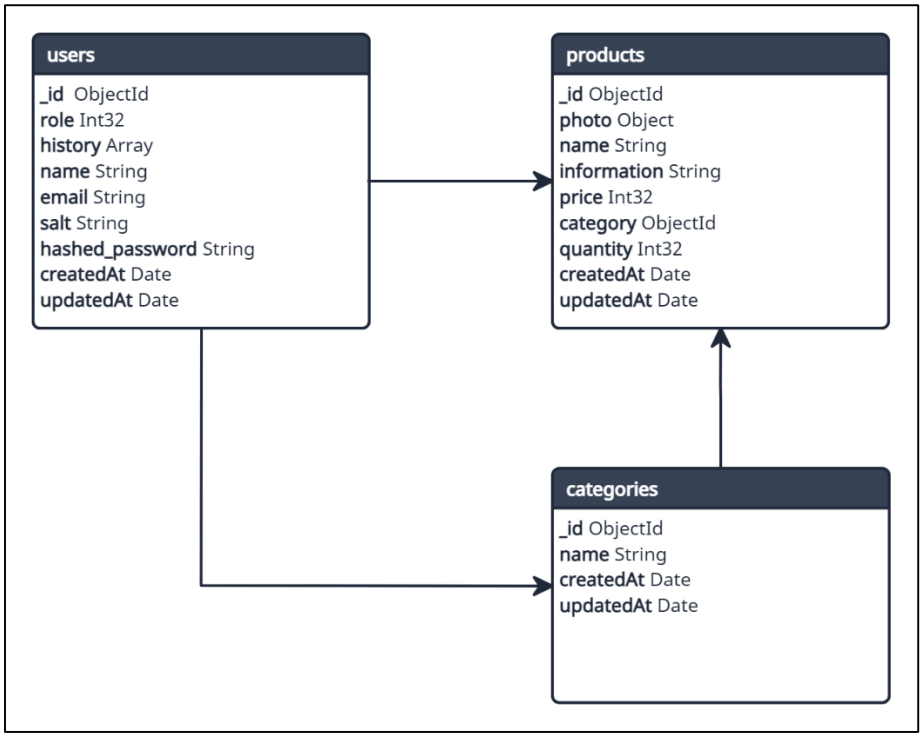


Рисунок 3.4 – Структура бази даних

Після того, як ми створили та налаштували базу даних її необхідно підключити до проекту. Кроки підключення бази даних описані на рис.3.5.

Спочатку інсталюємо відповідний драйвер до програмного середовища, а потім додаємо рядок-підключення до програмного коду.

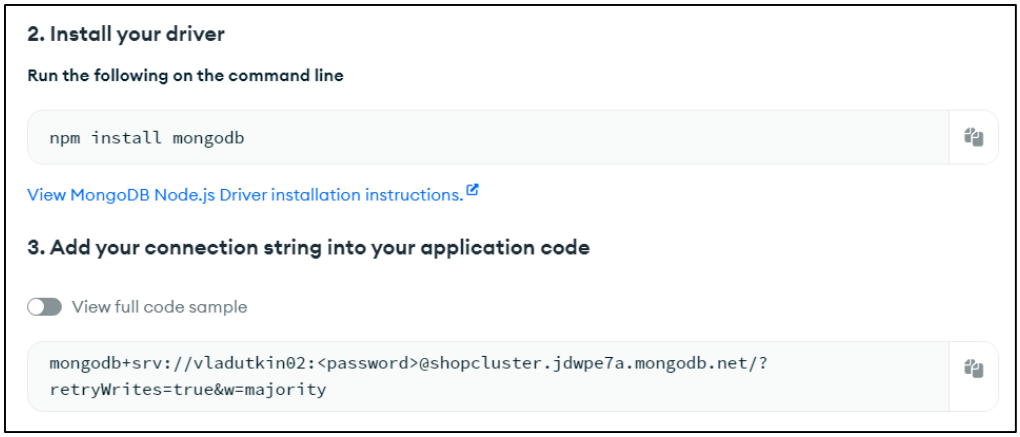


Рисунок 3.5 – Кроки для підключення бази даних

Наступним кроком потрібно реалізувати підключення до БД в програмному коді. Для цього використаємо Mongoose - спеціальну ODM-бібліотеку, яка використовується для створення зв'язку між mongoDB та середовищем виконання JS.

Викликаємо метод mongoose.connect, в якому вказано адрес на базу даних, який ми попередньо зберегли у файлі .env (рис. 3.6)

```
const connectMongo = async () => {
  try {
    await mongoose.connect(
      process.env.MONGO_URI,
      {
        useNewUrlParser: true,
        useUnifiedTopology: true,
        useCreateIndex: true,
        useFindAndModify: false,
      }
    );
    console.log('База даних успішно підключена');
  } catch (err) {
    console.error(err.message);
    process.exit(1);
  }
};
connectMongo();
```

Рисунок 3.6 – Підключення БД

Запустивши сервер, можемо впевнитися про успішне підключення БД(рис. 3.7)

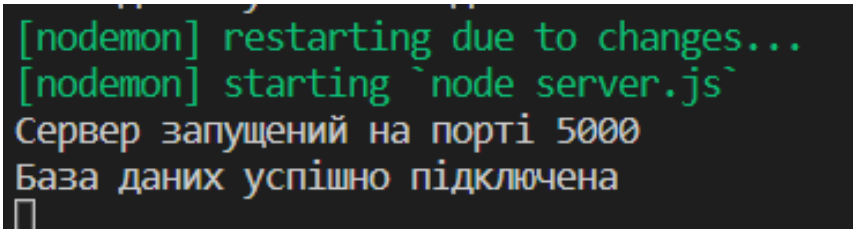


Рисунок 3.7 – Перевірка підключення БД

Після того, як ми підключили БД, потрібно створити моделі даних, з якими потрібно буде взаємодіяти. Відповідно до тої структури БД, яку описано вище, були створені моделі:

- User

Поля:

- Name (тип: String)
- Email (тип: String)
- Hashed_password (тип: String)
- Salt (тип: String)
- Role (тип: String)
- History (тип: Array)

- Products

Поля:

- Name (тип: String)
- Information (тип: String)
- Price (тип: Number)
- Category (тип: ObjectId)
- Quantity (тип: String)
- Photo (тип: Array)

- Categories

Поле:

- Name (тип: String)

3.2.2 Роутери та контролери

Роутери та контролери є надзвичайно важливими компонентами при розробці серверної частини веб-застосунка.

Роутери визначають URL-шляхи, до яких можна зробити запит у веб-застосунку. Вони вказують, який контролер або функція контролера повинна обробити цей запит. Таким чином, роутери встановлюють зв'язок між URL-шляхами та діями, які необхідно виконати для обробки запиту.

В свою чергу, контролери виконують обробку запитів, які надійшли через роутери.

Для реалізації роутерів у власному проекті мною було використано бібліотеку Express.js, а саме - вбудований в дану бібліотеку об’єкт Router. Створимо папку Routes, в якій реалізуємо маршрути для товарів, категорій та користувачів та аутентифікації.

Наведу приклад роботи роутерів з товарами(рис 3.8)

```
router.get('/products', list);
router.get('/products/search', listSearch);
router.post('/products/by/search', listBySearch);
router.get('/products/associated/:productId', listAssociated);
router.get('/products/categories', listCategories);
router.get('/product/photo/:productId', photo);

router.param('userId', findUserById);
router.param('productId', findProductById);

module.exports = router;
```

Рисунок 3.8 – Робота роутерів

Бачимо, що мною були використані різні методи роутера, такі як router.get(), router.post() для визначення маршрутів. В кожному маршруті вказаний відповідний контролер, який буде обробляти запит. Також використано router.param() для визначення конкретних параметрів, які повинні бути передані в запиті

Для контролерів також була створена окрема папка під назвою controllers, яка містить файли, в яких реалізовано контролери відповідно до кожного роутера.

Загалом, вміст файлів з контролерами виглядає наступним чином:

- Auth.js

Реалізовано фнукції:

- Signup - реєстрація нового користувача
- Signin - авторизація користувача
- Signout – вихід з профіля
- checkAuth – middleware функція перевірки авторизації
- checkAdmin – middleware функція перевірки ролі адміністратора

- Category.js

Реалізовано фнукції:

- findCategoryById – Пошук категорії за її ідентифікатором у базі даних
- create – Додавання нової категорії
- read – Зчитування всіх категорій
- update – Оновлення інформації про категорію
- remove – Видалення існуючих категорій
- list – Отримання списку всіх категорій

- Product.js
  - findProductById – пошук товару по ID
  - read – зчитування всіх товарів
  - create – додавання нових товарів
  - remove – видалення існуючих товарів
  - update – оновлення інформації про товари
  - list – отримання списку всіх товарів
  - listAssociated – отримання списку сумісних товарів

- listBySearch – пошук товарів за заданими параметрами
- photo – надання зображення товару
- listSearch – пошук товарів за запитом

- User.js
  - findUserById – пошук користувача по ID
  - read – зчитування даних про користувача
  - update – оновлення інформації про користувача

Пропоную детально розібрати реалізацію однієї із функцій контролера. (рис. 3.9)

```
exports.list = (req, res) => {
  let limit = req.query.limit ? parseInt(req.query.limit) : 6;
  let order = req.query.order? req.query.order: 'asc';
  let sortBy = req.query.sortBy ? req.query.sortBy : '_id';

  Product.find()
    .select('-photo')
    .populate('category')
    .sort([[sortBy, order]])
    .limit(limit)
    .exec((err, products) => {
      if (err) {
        return res.status(400).json({
          error: 'Товари не найдені',
        });
      }
      res.json(products);
    });
};
```

Рисунок 3.9 Функція отримання списку товарів

На даному рисунку представлена реалізація функції, яка повертає список товарів з бази даних.

Опис:

- Функція отримає два параметри: ‘res’(запит) та ‘req’(відповідь)
- Відбувається деконструкція об’єкту req.query для отримання значень параметрів запиту: ‘sortBy’, ‘limit’. Якщо параметри не вказані, встановлюється значення за замовчуванням.

- Викликає метод find() моделі Product
- За допомогою методу select('-photo') вибираються всі поля товару, за винятком поля photo.
- Викликається метод populate('category'), який заповнює поле category об'єкта товару посиланням на відповідний документ категорії.
- Викликається метод sort([[sortBy, order]]), який сортує товари за вказаним полями.
- Викликається метод limit, який обмежує кількість повернутих товарів відповідно до вказаного значення.
- Завершується функція викликом exec((err, products) => { ... }), який виконує запит до бази даних. У випадку помилки під час виконання запиту або якщо товари не знайдено, відправляється відповідь зі статусом 400 (Bad Request) і повідомленням про помилку. У разі успішного виконання запиту, список товарів повертається у відповіді у форматі JSON.

3.2.3 Реєстрація та авторизація

Для реалізації системи реєстрації та авторизації було використано JWT токени. JWT – це спосіб представлення даних у вигляді токена доступу на базі JSON, який використовується для передачі даних.

Загалом, структура JWT складається з 3 частин:

- Header (JSON-об’єкт, який складається з двух полів: типу токена та алгоритму хешування)

- Payload(JSON-об’єкт, який містить корисну інформацію про киористувача або інші дані. Може містити, як стандартні поля, так і поля, визначені користувачем

- Signature(Підпис, який використовується для перевірки цілісності та та автентичності токена. Для його генерації відбувається підписування закодованого заголовка та заявки з секретним ключем)

Структура JWT представлена на рис. 3.10.[22]

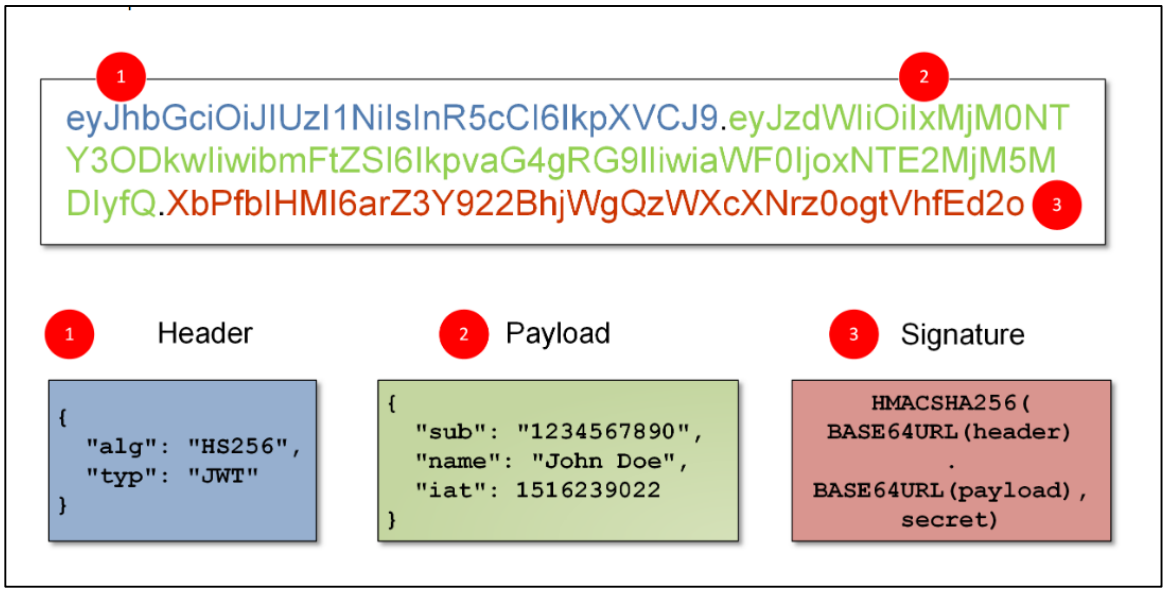


Рисунок 3.10 – Структура JWT-токену[22]

В нашому проекті алгоритм роботи з JWT-токенами відбувається наступним чином:

- Після аутентифікації користувача генерється JWT-токен.
- Згенерований токен зберігається на клієнтському боці в локальному сховищі(LocalStorage).
- Після отримання токена від сервера, клієнт включає його в кожен наступний запит до сервера.
- Сервер після отримання запиту перевіряє цілісність та автентичність токена. Відбувається перевірка підпису. Також відбувається розкодування токену, щоб отримати користну інформацію про користувача.
- Після перевірки та розкодування токена, сервер виконує авторизацію, перевіряючи, чи має користувач, представлений в токені, доступ до запитуваного ресурсу або дії.

Наведу приклад по роботі з JWT-токеном в моєму програмному коді. На рис. 3.11 представлена реалізація функції авторизації в магазин.

```
exports.signin = (req, res) => {
  const { email, password } = req.body;
  User.findOne({ email }, (err, user) => {
    if (err || !user) {
      return res.status(400).json({
        error: "Користувача з цим email не існує. Будь-даска, зареєструйесь",
      });
    }
    if (!user.authenticate(password)) {
      return res.status(401).json({
        error: "Email та пароль не співпадають",
      });
    }
    const token = jwt.sign(
      { _id: user._id },
      process.env.JWT_SECRET
    );
    res.cookie('t', token, { expire: new Date() + 100000 });
    const { _id, name, email, role } = user;
    return res.json({ token, user: { _id, email, name, role } });
  });
};
```

Рисунок 3.11 – Приклад роботи з JWT-токеном

У випадку успішної авторизації генерується JWT-токен з використанням функції jwt.sign(). У токені вказуємо ідентифікатор (_id) та секретний ключ(JWT_SECRET), який використовується для підпису токена. Токен зберігається у вигляді cookie з ім’ям ‘t’ та з датою закінчення терміну дії. Дані про користувача, які мають бути надіслані на клієнт, виділяються з об'єкту користувача (user) і повертаються у відповіді на клієнт разом з токеном.

Також представлю приклад включення токену в HTTP-запит клієнта до серверу(рис. 3.12)

```
headers: {
  Accept: 'application/json',
  'Content-Type': 'application/json',
  Authorization: `Bearer ${token}`,
}
```

Рисунок 3.12 – Використання токену в HTTP-запиті

3.3 Розробка та реалізація клієнтської частини

- **Початок роботи та застосування залежностей**

Розпечнемо огляд реалізації front-end, як і в випадку з back-end, з опису залежностей, які будуть використані в якості додаткових інструментів, які зроблять розробку більш ефективною.

Перелік застосованих залежностей:

- @material-ui/core і @material-ui/icons: Набір компонентів Material-UI, які використовуються для створення інтерфейсу користувача.
- @testing-library/jest-dom, @testing-library/react і @testing-library/user-event: Бібліотеки для тестування React компонентів та взаємодії з ними у тестовому середовищі.
- config: Бібліотека для завантаження конфігураційних файлів, які містять параметри налаштування проекту.
- fontsource-roboto: Пакет, який надає шрифт Roboto для використання у проекті.
- moment: Бібліотека для роботи з датами та часом, яка допомагає форматувати, відображати та обробляти різні формати дат.
- query-string: Пакет для розбору та створення рядків запитів URL, що допомагає обробляти та робити запити з параметрами у проекті.
- react і react-dom: Основні бібліотеки React для створення та відображення компонентів.
- react-router-dom: Бібліотека маршрутизації для React, яка дозволяє керувати навігацією та переходами між сторінками в онлайн-магазині.

- react-scripts: Скрипти для розробки та збірки проекту React, які забезпечують зручні команди для запуску, розробки та побудови проекту.

Для початку роботи з клієнтською частиною створимо окрему папку client, в якій буде знаходиться вся реалізація. Це зробимо для того, щоб розділити в програмі Front-end і Back-end та зробити проект більш структурованим.

В даній папці розгорнули початковий react проект, використавши команду npm create-react-app.

Загалом, файлова структура виглядає наступним чином:

- Папка Src (містить весь розроблений функціонал):
  - Папка Admin

    В даній директорії будуть знаходитися файли, в яких буде реалізований функціонал адміністратора в онлайн-магазині
  - Папка Auth

    Містить файли, в яких буде реалізовано набір функцій для взаємодії з сервером, пов’язаним з аутентифікацією та реєстрацією користувачів. Також тут будуть реалізовані приватні маршрути.
  - Папка Core

    Тут буде реалізовано більшу частину клієнтської частини веб-додатку. В окремих файлах буде розроблена головна сторінка, меню, сторінка товару, кошик, фільтрація за категоріями та цінами, картки товарів і т.д
  - Папка User

    Буде реалізовано сторінка профіля користувача в якій, в залежності від ролі, буде знаходиться панель адміністратора або користувача. Також тут буде міститься реалізація сторінки реєстрації та авторизації.

- Файл Routes.js

  Визначення конфігурації маршрутів React

- Файл Index.js

  Точка входу для React-додатку, де рендериться застосунок та відображається на сторінці

- **Реєстрація та авторизація**

Для реалізації реєстрації нових користувачів в папці User був створений компонент Signup.js. На рис. 3.13 зображена частина реалізації даного компоненту

```
export default function Signup() {
  const [values, setValues] = useState({
    name: '',
    email: '',
    password: '',
    error: '',
    success: false,
  });

  const { name, email, password, success, error } = values;

  const handleChange = (name) => (event) => {
    setValues({ ...values, error: false, [name]: event.target.value });
  };

  const clickSubmit = (event) => {
    event.preventDefault();
    setValues({ ...values, error: false });
    signup({ name, email, password }).then((data) => {
      if (data.error) {
        setValues({ ...values, error: data.error, success: false });
      } else {
        setValues({
          ...values,
          name: '',
          email: '',
          password: '',
          error: '',
          success: true,
        });
      }
    });
  };
```

Рисунок 3.13 – Частина реалізації реєстрації

Для реалізації даного компоненту був використаний хук ‘useState’ для управління станом компонента, зокрема, для збереження значень полів форми та відображення повідомлень про помилки або успішу реєстрацію.

Більш детально розбираючи реалізацію, можна виділити такі складові:

- Визначення початкового стану за допомогою хука ‘usestate’.
- Функція ‘handleChange’ оновлює значення полів форми при їх зміні.
- Функція ‘clickSubmit’ оброблює подію натискання на кнопку “Зареєструватися” і відправляє дані форми на сервер за допомогою функкції ‘signup’.
- Функції ‘showError’ та ‘showSuccess’, які відображають повідомлення про помилки або успішну реєстрацію.
- Форма реєстрації, в якій використовуються компоненти з бібліотеки Material-UI, такі як TextField, Button, Grid, Container, Typography та інші.
- Виклик функції signUpForm для відображення форми реєстрації.
- Повернення розмітки компонента з використанням компонента Layout, який обгортає форму реєстрації і надає заголовок та опис сторінки.

Для реалізації авторизації в папці User був створений компонент Signin. Загалом, з точки зору клієнтської реалізації компонент має таку ж структура, як і попередній компонент Signup

До відмінностей між розробкою авторизації та реєстрації можна віднести:

- Різні властивості стану при використанні хуку ‘useState’.
- Різні операції зі станом. У випадку успішної авторизації стан ‘redirectToReferrer’ встановлюється на true, що призводить до перенаправлення користувача на відповідну сторінку(рис.3.14). У випадку реєстрації, стна success встановлюється на true, щоб відобразити повідомлення про успішну реєстрацію

- Різний вигляд та розташування компонентів

```
const redirectUser = () => {
  if (redirectToReferrer) {
    if (user && user.role === 1) {
      return <Redirect to='/admin/panel' />;
    } else {
      return <Redirect to='/user/panel' />;
    }
  }
  if (isAuthenticated()) {
    return <Redirect to='/' />;
  }
};
```

Рисунок 3.14 – Перенаправлення користувача після авторизації

- **Домашня сторінка**

На домашній сторінці нашого магазина будуть відображені картки товарів, які будуть за замовчування відсортовані за датою додання товару. Таким чином користувачі зможуть одразу побачити нові надходження, як тільки перейшли на наш сайт.

Також, на цій сторінці буде можливість знайти товар за назвою та категорією. Для цього реалізуємо відповідну форму.

Перейдемо до реалізації:

В папці Core був створений компонент Home, який відображає домашню сторінку веб-додатку. В ньому використовується стан для зберігання списку продуктів та помилок, які можуть виникнути під час завантаження продуктів.

За допомогою хука ‘useState’ створюємо стан, який містить порожній масив ‘productsByArrival’. Далі створюємо функцію loadProductsByArrival, в якій виконуємо запит на сервер за допомогою ‘getProducts('createdAt')’. Викликаємо цю функцію за допомогою хука ‘useEffect’. Таким чином ми завантажили товари за часом їх додання в базу даних (рис 3.15).

```
const Home = () => {
  const [error, setError] = useState([]);
  const [productsByArrival, setProductsByArrival] = useState([]);


  const loadProductsByArrival = () => {
    getProducts('createdAt').then((data) => {
      if (data.error) {
        setError(data.error);
      } else {
        setProductsByArrival(data);
      }
    });
  };

  useEffect(() => {
    loadProductsByArrival();
  }, []);
```

Рисунок 3.15 – Завантаження товарів за датою додання

В папці Core створено компонент Search, в якому реалізовано строку пошуку товарів по назві.

- **Сортуваня товарів**

На сторінці магазину реалізована можливість сортування товарів за категоріями та цінами.

Реалізація даного функціоналу відбувається в компоненті Shop.

Головну роль при розробці відіграють наступні складові:

- loadFilteredResults – функція завантаження відфільтрованих результатів при зміні фільтрів
- getFilteredProducts – функція отримання відфільтрованих даних
- filteredResults – стан компонента в якому зберігаються отримані дані
- handleFilters – функція, яка викликається при зміні фільтрів та оновлює значення у стані компонента в залежності від обраних фільтрів
- handlePrice – функція отримання диапазону цін за вибраним значенням цінового фільтру.

На рис 3.16 продемонстрована частина коду, яка відповідає за фільтрування товарів.

```
useEffect(() => {
  init();
  loadFilteredResults(skip, limit, myFilters.filters);
}, []);

const handleFilters = (filters, filterBy) => {
  const newFilters = { ...myFilters };
  newFilters.filters[filterBy] = filters;

  if (filterBy === 'price') {
    let priceValues = handlePrice(filters);
    newFilters.filters[filterBy] = priceValues;
  }
  loadFilteredResults(myFilters.filters);
  setMyFilters(newFilters);
};
```

Рисунок 3.16 – Частина реалізації фільтрування товарів

- **Картки товарів**

Товари в нашому магазині будуть відображатися у виді карток, які в собі будуть містити:

- Фото товару
- Назву
- Опис
- Інформація про ціну, категорію та час додання товару в магазин
- Кнопка переходу на сторінку огляду товару
- Кнопка додання товару в кошик
- Кнопка видалення товару з кошика
- Відображення статусу наявності товару
- Поле вводу кількості товарів, доданих в кошик

Для реалізації було ствоерно компонент ‘Card’. Пропоную детально основні елменти коду:

- Використання хуку useState для збереження стану компонента
- Функція showViewButton відповідає за відображення кнопки "Показати товар" з посиланням на деталі товару.

- Функція addToCart викликається при натисканні на кнопку "Додати в кошик" і додає товар до кошика за допомогою функції addItem.
- Функція shouldRedirect перенаправляє користувача на сторінку кошика, якщо змінна redirect встановлена в true.
- Функція showAddToCartBtn відповідає за відображення кнопки "Додати в кошик".
- Функція showStock відображає статус наявності товару (в наявності або немає в наявності).
- Функція handleChange викликається при зміні кількості товару в полі введення. Вона оновлює стан count і оновлює кількість товару в кошику за допомогою функції updateItem.
- Функція showCartUpdateOptions відповідає за відображення опцій оновлення кількості товару в кошику.
- Функція showRemoveButton відображає кнопку "Видалити товар" і викликає функцію removeItem для видалення товару з кошика.

Компонент Card повертає розмітку карточки товару, яка містить зображення товару, назву, опис, ціну, категорію, дату додавання, статус наявності, кнопки "Показати товар", "Додати в кошик" та "Видалити товар", а також поле введення для оновлення кількості товару в кошику.

- **Сторінка профілю користувача**

В проекті також реалізовано сторінку профілю користувача, яка містить інформацію про нього: ім'я, електронна пошта, роль

Також на цій сторінці реалізована панель управління, вміст якої буде залежить від ролі, яку має користувач.

У випадку із звичайним користувачем буде можливість перейти за посиланням до свого кошику, а також змінити інформацію про свій профіль.

Якщо користувач – адміністратор, то панель управління надаватиме можливість додавати нові товари та категорії, а також редагувати та видаляти існуючі товари.

На рис. 3.17 представлена реалізація панелі управління адміністратора.

```
const AdminPanel = () => {
  const {
    user: { _id, name, email, role },
  } = isAuthenticated();

  const adminTools = () => {
    return (
      <div className='card'>
        <h4 className='card-header'>Посилання</h4>
        <ul className='list-group'>
          <li className='list-group-item'>
            <Link className='nav-link' to='/create/category'>
              Додати категорію
            </Link>
          </li>
          <li className='list-group-item'>
            <Link className='nav-link' to='/create/product'>
              Додати товар
            </Link>
          </li>
          <li className='list-group-item'>
            <Link className='nav-link' to='/admin/products'>
              Управляти товарами
            </Link>
          </li>
        </ul>
      </div>
    );
  };
```

Рисунок 3.17 – Реалізація адмін-панелі

- **Кошик товарів**

Сторінка кошику товарів буде містити товари, які користувач додав в кошик при перегляді товарів. Товари, представлені у вигляді карток, аналогічно до реалізації інших сторінок

У користувача на сторінці є можливість видалити товар з кошику, обрати кількість, які він хоче додати до замовлення.

З правої частини буде відображати загальна сума, які клієнт повинен сплати за обрані товари.

- **Сторінка перегляду товару**

Дану сторінку можна розділити умовно на ліву та праву частини. На лівій - користувач може ознайомитися з детальною інформацією про товар, а на правій – користувачу буде запропонавано супутні товари. В випадку моєї реалізації супутні товари – це товари із спільної категорії.

На рис 3.18 можна побачити частину реалізації компонента Product.

```
const Product = (props) => {
  const [product, setProduct] = useState({});
  const [associatedProduct, setAssociatedProduct] = useState([]);
  const [error, setError] = useState(false);

  const loadSingleProduct = (productId) => {
    read(productId).then((data) => {
      if (data.error) {
        setError(data.error);
      } else {
        setProduct(data);
        listAssociated(data._id).then((data) => {
          if (data.error) {
            setError(data.error);
          } else {
            setAssociatedProduct(data);
          }
        });
      }
    });
  };
```

Рисунок 3.18 – Частина реалізації компоненту Product

В даному компоненті розроблено функцію ‘loadSingleProduct’, яка відповідає за завантаження обраного товару та супутнього товару.

ВИСНОВОК ДО РОЗДІЛУ 3

В третьому розділі було спроектовано та реалізовано веб-застосунок. На початку роботи визначилися з функціоналом веб-додатка. Були поставлені конкретні задачі, які повинен виконувати розроблений додаток. Розробили блок-схему алгоритму, яка чітко описує принцип роботи веб-застосунка.

Розроблено та реалізувано серверну частину програми. Завантажили необхідні пакети для підвищення ефективності розробки. Виконали роботу з базою даних MongoDB: зареєструвались на офіційному ресурсі та створили кластер бази даних, налаштували БД відповідно до нашого проекту, описали структура БД, підключили БД до проекту. В ході виконання серверної частини створені моделі даних. Реалізовані роутери та контролери, які виступають фундаментом back-end частини проеку. За допомогою технології JWT-токенів реалізована система реєстрації та авторизації.

Розроблено та реалізовано клієнтську частину програми. Встановлено окремі залежності для покращення розробки. Описано файлову структуру клієнтської частини проекту. Реалізовано форми реєстрації та авторизації, домашню сторінку, сортування товарів, картки товарів, сторінку профілю користувача, кошик товарів, сторінку перегляду товарів.

Таким чином, повністю реалізовано веб-застосунок для продажу спортивних товарів. В наступному розділі буде показана робота веб-додатку та перевірка його функціоналу.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО ВЕБ-ЗАСТОСУНКУ

4.1 Огляд та тестування роботи веб-застосунку

- **Реєстрація та авторизація**

Спочатку зробимо огляд форми реєстрації нових користувачів. На рисунку представлений процес реєстрації. Ввели всі необхідні поля та спробували зареєструватися. Продемонструю реалізацію одного із варіантів валідації форми (рис.4.1).

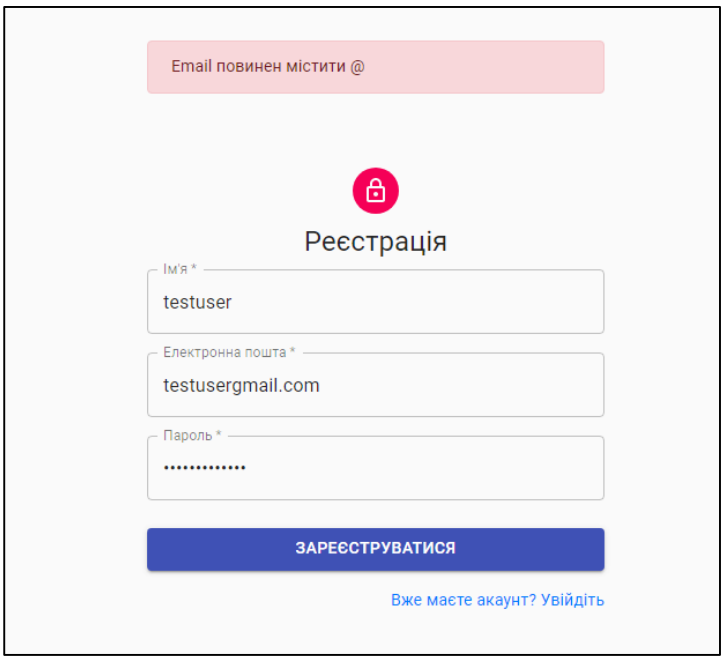


Рисунок 4.1 – Форма реєстрації

Після того, як користувач був успішно створений, система надає інформацію про це (рис. 4.2).

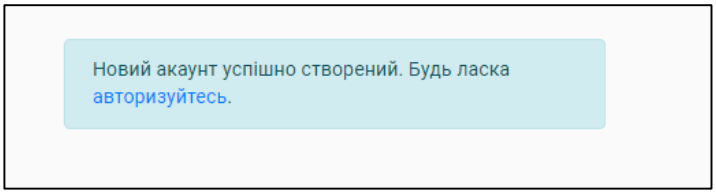


Рисунок 4.2 – Повідомлення про успішну реєстрацію

Переходимо на сторінку авторизації, на якій вводимо дані, вказані при реєстрації. Протестуємо роботу форми авторизації, навмисно вказавши неправильний пароль(рис 4.3) .

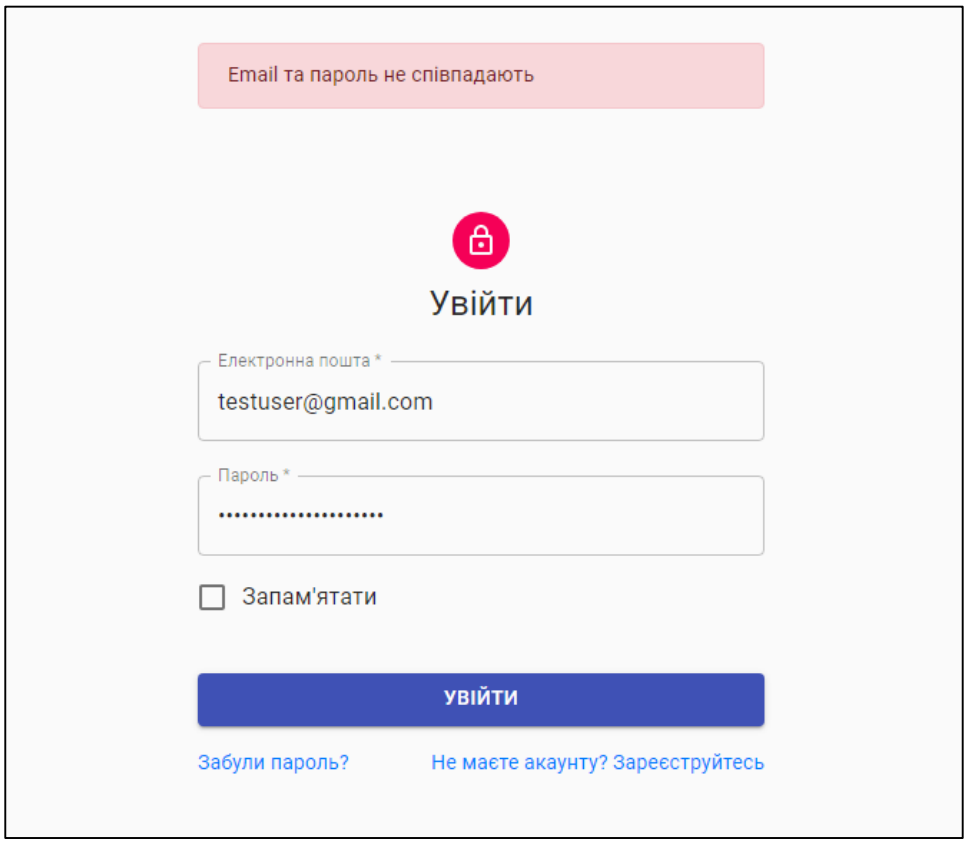


Рисунок 4.3 – Форма авторизації

Після того, як ввели правильні дані, користувач перенаправляється на домашню сторінку онлайн-магазина. На рис.4.4 можна побачити, що зареєстрований користувач додан в базу даних, де вказана його роль, зашифрований пароль, дата реєстрації і т.д.


```
_id: ObjectId('647f43c0e320962b8c302763')
role: 0
history: Array
name: "testuser"
email: "testuser@gmail.com"
salt: "1f1fef70-0477-11ee-aea6-8da0bcfeab47"
hashed_password: "0d5b4febdb8de21be5170c50d199921370c99984"
createdAt: 2023-06-06T14:33:36.491+00:00
updatedAt: 2023-06-06T14:33:36.491+00:00
__v: 0
```


Рисунок 4.4 – Додання користувача в базу даних

- **Домашня сторінка**

Розпочнемо огляд роботи із домашньої сторінки застосунка. На рис. 4.5 можна побачить пошукову строку та список товарів відсортованих за часом додання в магазин.

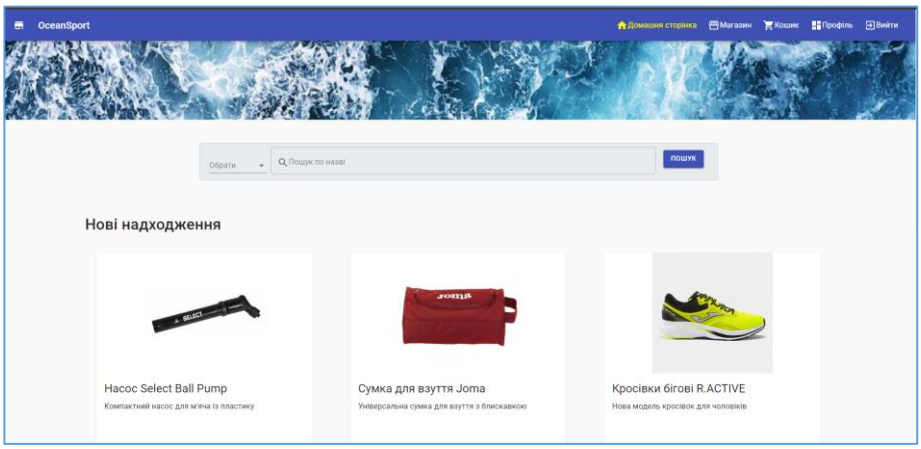


Рисунок 4.5 – Домашня сторінка

Протестуємо роботу пошукової строки. Введемо назву товару та спробуємо знайти товар(рис 4.6).

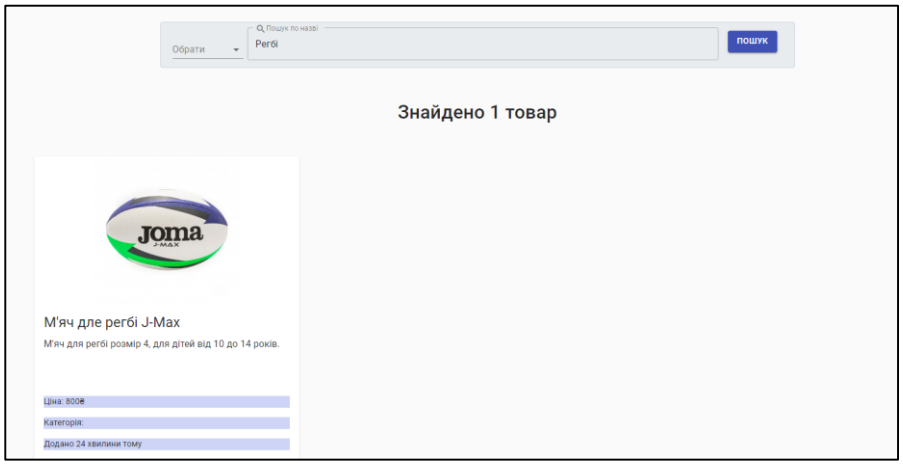


Рисунок 4.6 – Пошук товару по назві

- **Картки товарів**

Проведемо детальний огляд карток товарів. З рис. 4.7 можна побачити, що картки містять в собі фотографію товара, назву, короткий опис, ціну, назву категорії до якої відноситься товар та час додання товару в магазин. Також реалізовано кнопки: перегляд товару та додання в кошик. Окрему увагу слід приділити напису, який інформує користувача про наявність товару на складі.

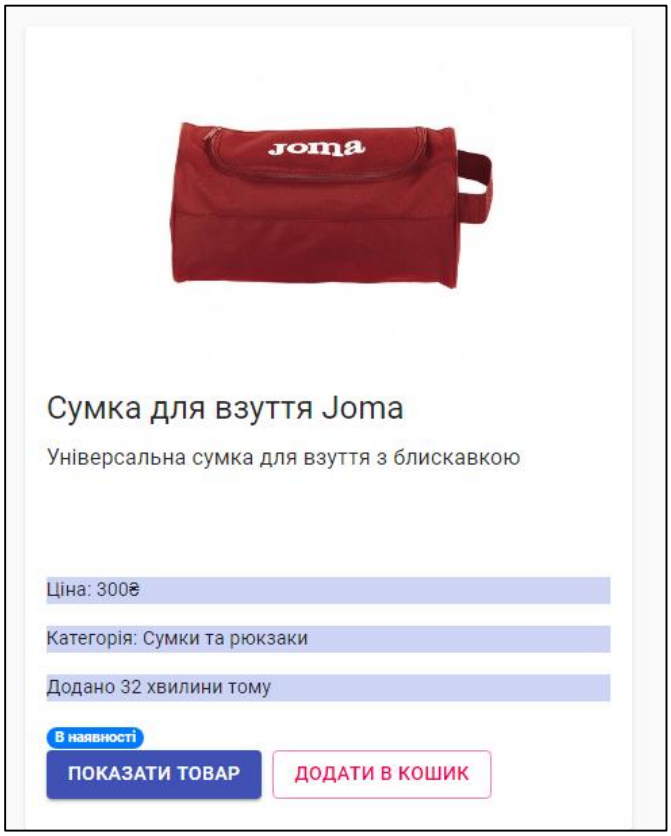


Рисунок 4.7 – Картка товару

Для того щоб, перевірити достовірність інформації про наявність товару на складі виконаємо наступні дії: перейдемо до управляння бази даних та у документі, який відповідає за поданий товар поставимо полю quantity(кількість) значення 0. На рис. 4.8 наведені успішні реазультати такого тестування.

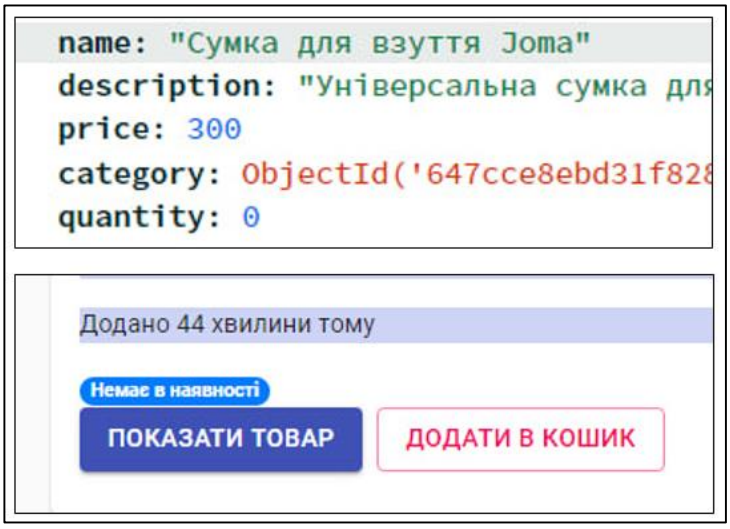


Рисунок 4.8 – Тестування наявності товару

- **Сторінка магазину**

На сторінці магазина реалізовано можливість фільтрування товарів за категоріями та цінами. На рис. 4.9 продемонстрована успішна фільтрація товару за обраними значеннями.

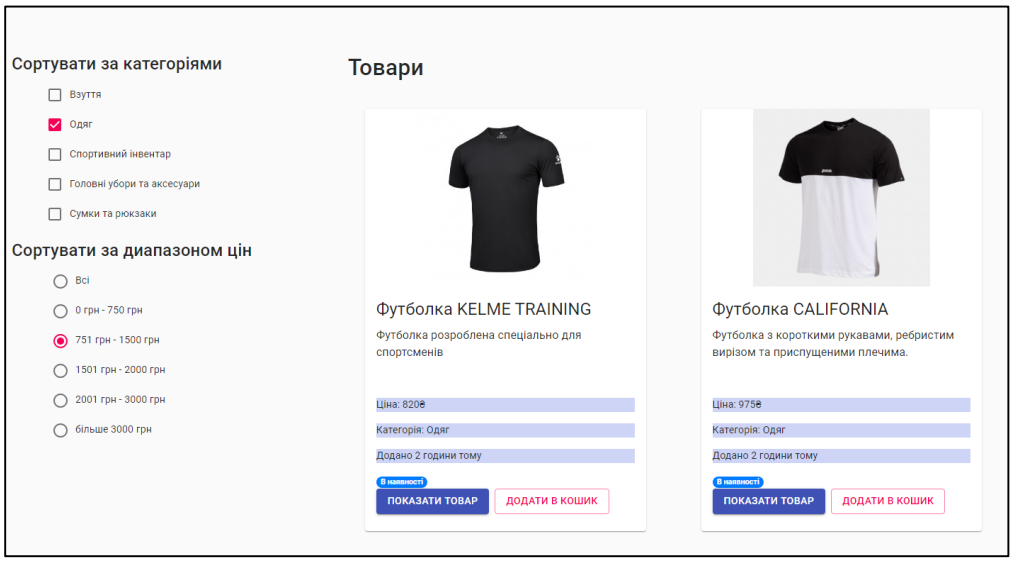


Рисунок 4.9 – Фільтрування товарів

- **Профіль користувача**

Після того, як користувач авторизувався в системі, він може перейти в свій профіль за допомогою відповідного посилання в меню. Сторінка профілю користувача продемонстрована на рис. 4.10. На ній можна побачити поточну інформацію про користувача: ім’я, пошта та роль.

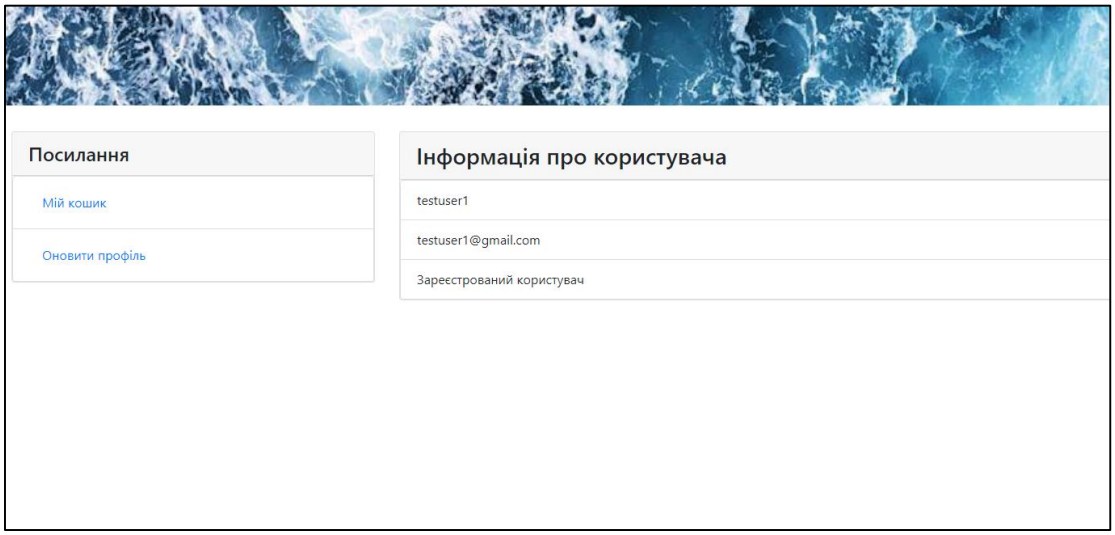


Рисунок 4.10 – Профіль користувача

Також, як можна помітити, реалізована панель управління, вміст якої відрізняється в залежності від ролі користувача. Спочатку розглянемо можливість звичайного користувача змінити оновити інформацію про профіль (рис. 4.11)

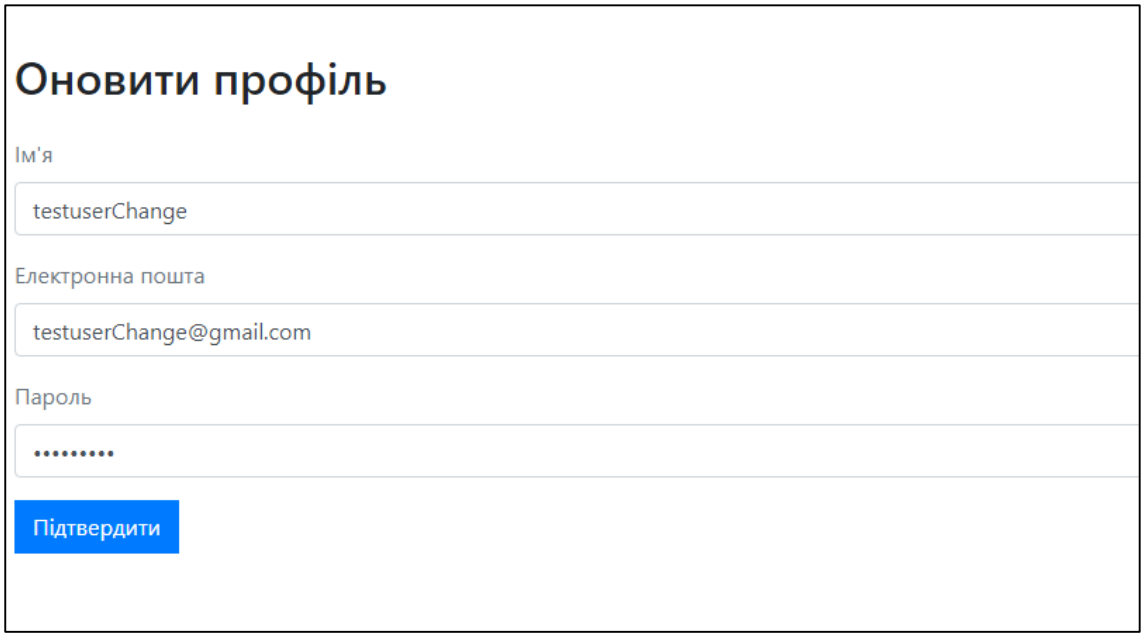


Рисунок 4.11 – Оновлення профілю

Вводимо у відповідні поля нові дані та натискаємо кнопку “Підтвердити”. Перевіряємо змінені дані на сторінці профіля (рис. 4.12).

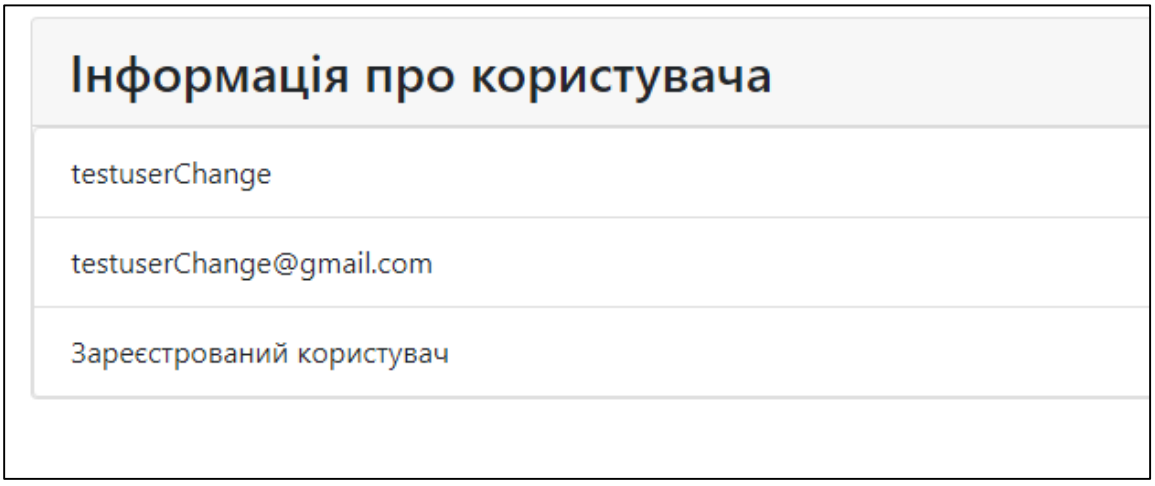


Рисунок 4.12 – Зміна інформації про користувача

Тепер перейдемо до огляну панелі управління адміністратора. За допомогою неї можна буде взаємодіяти з товарами магазину: додати нові позиції, категорії, редагувати існуючі товари. На рис. 4.13 показана сторінка профілю адміністратора.

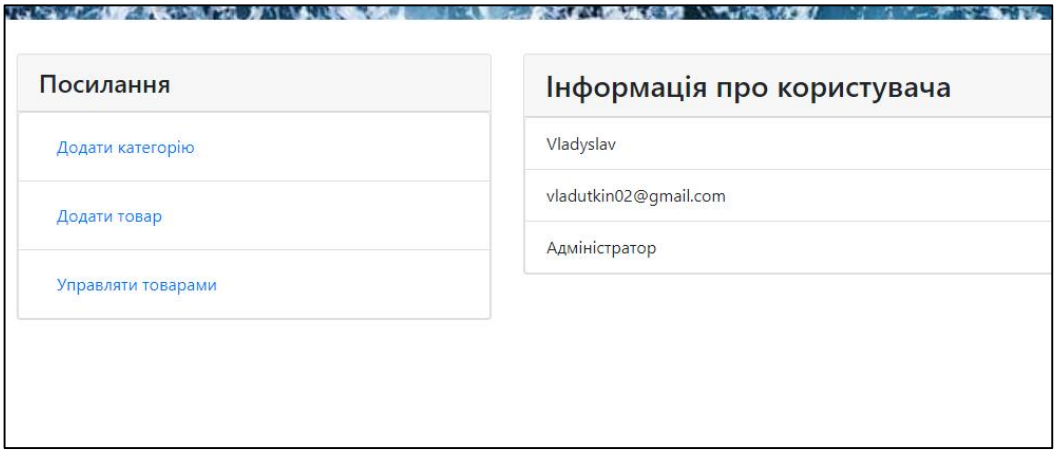


Рисунок 4.13 – Профіль адміністратора

Спробуємо скористатися панелю адміністратора, додавши новий товар до магазину. Натискаємо на відповідне посилання в панелі управління. Переходимо на сторінку з формою, яку потрібно заповнити. Наприклад, додамо набір гантелей до нашого магазину(рис 4.14).

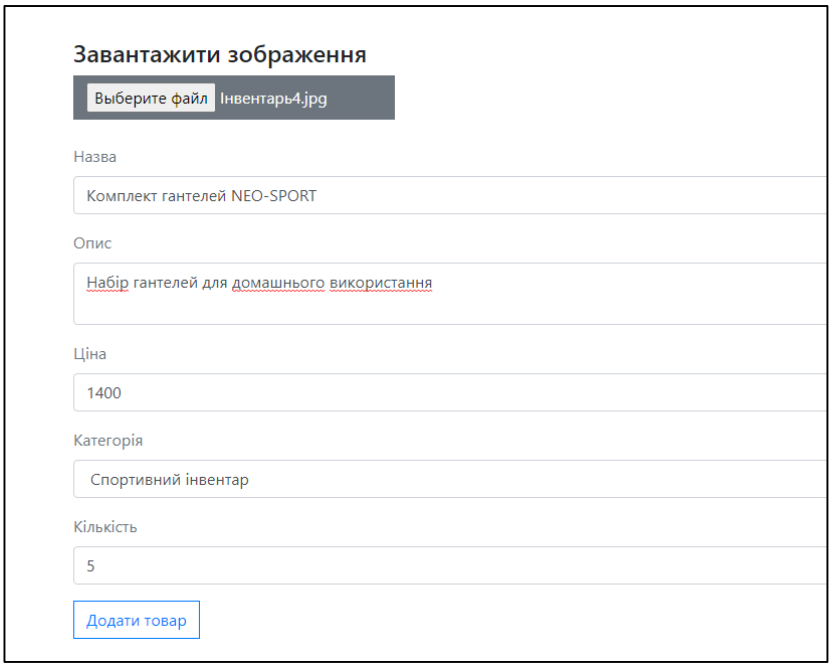


Рисунок 4.14 – Форма додання товару в магазин

Перевіряємо наявність доданого товару в магазині, перейшовши на домашню сторінку застосунка(рис 4.15).

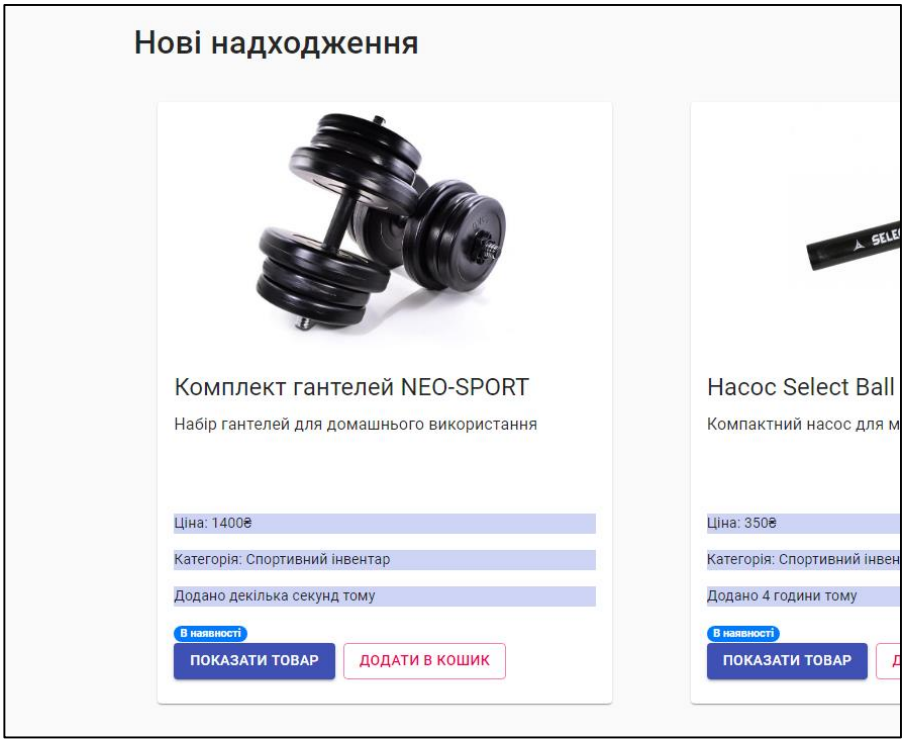


Рисунок 4.15 – Наявність доданого товару в магазині

Також, у адміністратора є можливість редагувати існуючі товари. Переходимо по відповідному посиланню в панелі управління(рис. 4.16). Кнопка Update дозволяє редагувати існуючі товари, за допомогою форми, аналогічній той, яку ми використовуєм при доданні товару. Також можна видалити існуючий товар.

Рисунок 4.16 – Панель редагування товарів

- **Кошик товару**

Для демонстрації реалізованої сторінки кошику додамо в неї якийсь товар (рис. 4.17).

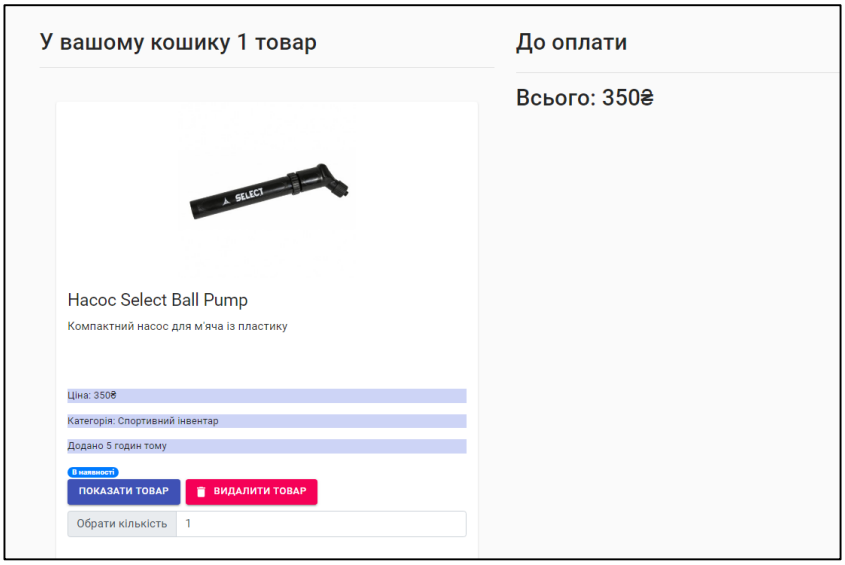


Рисунок 4.17 – Сторінка кошику товарів

Сторінка містить список товарів, доданих в кошик. Є можливість видалити товар, а також обрати кількість одиниць обраного товару. На правій частині сторінки розраховується сума всіх обраних товарів, яку необхідно буде сплатити клієнту. Для тестування змінемо, наприклад, кількість одиниць обраного насосу (рис. 4.18).

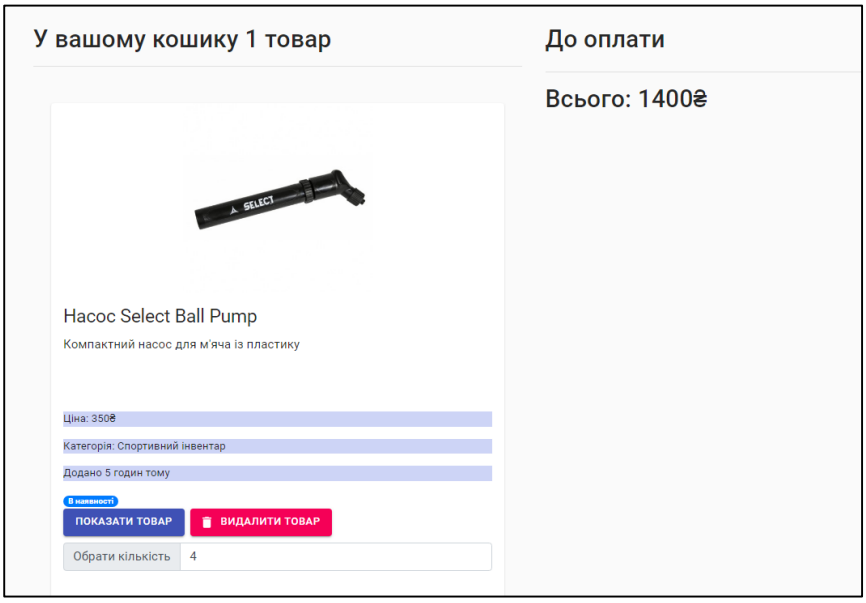


Рисунок 4.18 – Зміна кількості одиниць товару в кошику

- **Перегляд товару**

На рис. 4.19 представлена реалізована сторінка перегляду товару. На ній можна детальніше ознайомитися з товаром та подивитися запропоновані супутні товари. Як я казав в попередньому розділі, супутні товари виступають товарами з тієї категорії, що й обраний товар.

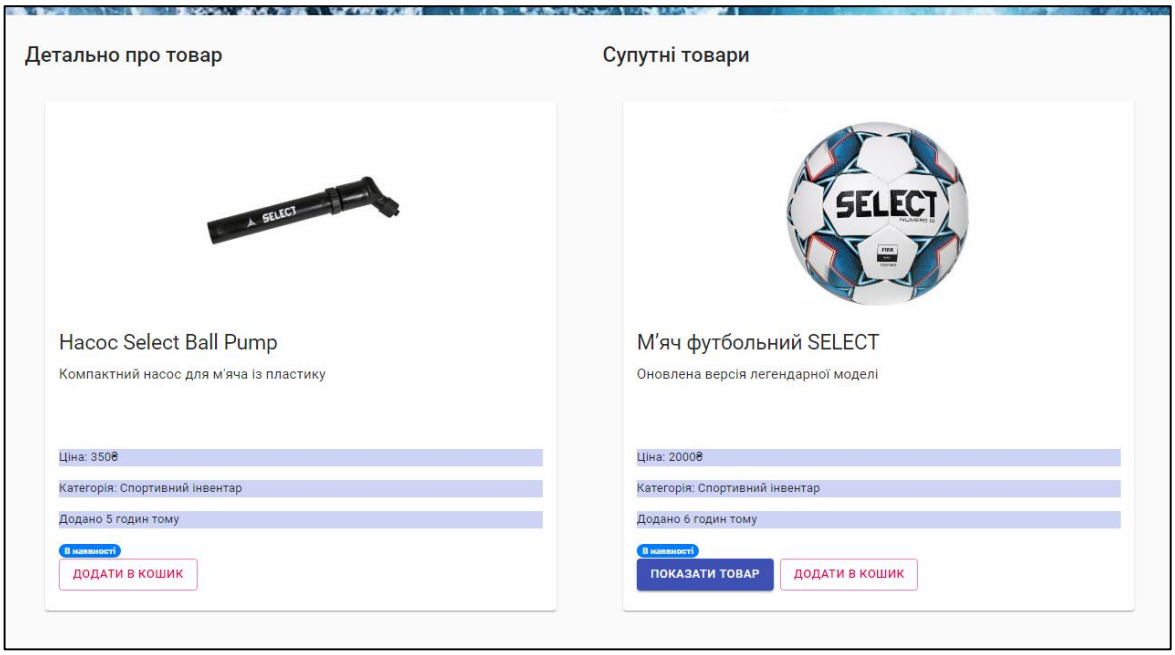


Рисунок 4.19 – Сторінка перегляду товару

4.2 Шляхи вдосконалення розробленого додатка

Після огляду та тестування можна зробити висновок, що веб-застосунок успішно реалізовано та він відповідає мінімальним вимогам до спортивного онлайн-магазина.

Однак, для того щоб веб-застосунок можна було назвати конкурентноспроможним на тлі існуючих аналогів, він потребує в покращенні.

По-перше, онлайн-магазин потребує в реалізації оформлення заказів та приймання платежів. Це необхідний крок, для того щоб розроблений проект, дійсно можна було використовувати в бізнес-сфері. Для цього слід приділити велику увагу автоматизації та безпеці веб-застосунку.

По-друге, застосунок потребує в розширенні функціоналу. До цього пункту можна віднести: можливість підписатися на розсилку, реалізація чат-боту, можливість додавати товару в категорію “улюблене”. Також, гарною ідеєю, саме для спортивного магазина, є реалізація сторінки із популярними статтями про спорт та здоровий образ життя загалом.

Також, варто зробити інтерфейс більш динамічним, додати велику кількість анімацій, що буде відповідати сучасним трендам онлайн-магазинів.

ВИСНОВОК ДО РОЗДІЛУ 4

В четвертому розділі було виконано повне дослідження роботи розробленого веб-застосунку.

Проведено огляд та тестування роботи додатка. Були розглянуті такі складові реалізації:

- Реєстрація та авторизація

Досліджені форми та протестовані деякі варіанти валідації. Успішно зареєстровано нового користувача.

- Домашня сторінка

Зроблено огляд домашньої сторінки. Окрему уваги приділено роботі пошукової строки.

- Картки товарів

Проведено детальний огляд. Шляхом зміни кількості товару в базі даних досліджена робота одного із елементу картки.

- Сторінка магазину

Продемонстрована реалізація фільтрування товарів за обраними значеннями

- Профіль користувача

Розглянуто сторінку профілю та досліджено функціонал панелі управління в ролі адміністратора та звичайного користувача.

- Кошик товару

Розглянуто сторінку та досліджено можливості зміні кількості одиниць товару.

- Перегляд товару

Проведено аналіз розробленого застосунку та описано шляхи вдосконалення продукту.

ВИСНОВКИ

В ході виконання дипломного проекту була досліджена галузь розробки веб-застосунків як з теоретичної сторони, так і з практичної.

В першому розділі проведено аналіз предметної області розробки. Досліджено загальну історію розвитку веб-технологій. Розглянуто галузь веб-застосунків та зроблено огляд існуючих рішень в сфері спортивних магазинів.

В другому розділі зроблено огляд та порівняння технологій для створення веб-застосунків. Розглянуті технології по трьом складовим: клієнтська частина, серверна частина та база даних. В результаті обрано найбільш оптимальний набір технологій для реалізації веб-застосунка.

В третьому розділі було описано проектування та розробку веб-застосунку. В ході початкового аналізу визначено задачі, які повинен виконувати додаток. Описана загальна структура роботи веб-застосунку. Розроблено клієнтську та серверну частини застосунку. Ключові моменти реалізації підкріплені рисунками та описом конкретних елементів коду.

В четвертому розділі проведено дослідження розробленого веб-застосунку. Проведено огляд та тестування роботи додатка. Роглянуто всі сторінки онлайн-магазину з описом функціоналу. Були наведені шляхи для подальшого вдосконалення продукту.

Підсумовуючи, варто зазначити, що розроблений веб-застосунок потребує в невеликому допрацюванні, щоб дійсно бути конкурентноспроможним із іншими відомими спортивними онлайн-магазинами. Метою розробки було створити продукт, який виконує конкретний ряд задач, які задовільняють загальні потреби користувача веб-застосунку.

Веб-застосунок відповідає, поставленим на початку розробки, вимогам. Робота виконана успішно. Завдання на дипломний бакалаврський проект виконано повністю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Grygorik I. Brief History of HTTP [Електронний ресурс] / Ilya Grygorik. – 2012. – Режим доступу до ресурсу: https://www.oreilly.com/library/view/http-protocols/9781492030478/ch01.html
2. 1993: CGI Scripts and Early Server-Side Web Programming [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://webdevelopmenthistory.com/1993-cgi-scripts-and-early-server-side-web-programming/
3. JavaScript History [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.w3schools.com/js/js_history.asp
4. Brien Posey. Wireless Application Protocol (WAP) [Електронний ресурс] / Brien Posey. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchmobilecomputing/definition/WAP
5. Навіщо потрібні фреймворки та бібліотеки [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/251-zachem-nuzhny-freymvorki-i-biblioteki
6. ЩО ТАКЕ ВЕБ ДОДАТОК? РІЗНИЦЯ МІЖ САЙТОМ, ВЕБ-ДОДАТКОМ, SPA І PWA [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/
7. The Fundamentals of Web Application Architecture [Електронний ресурс] // 2019 – Режим доступу до ресурсу: https://reinvently.com/blog/fundamentals-web-application-architecture/
8. Web Application Architecture – Detailed Explanation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.interviewbit.com/blog/web-application-architecture/

9. Swapnil B. What is Web Application Architecture? Components, Models, and Types [Електронний ресурс] / Banga Swapnil. – 2022. – Режим доступу до ресурсу: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more
10. Спортивний магазин Terrasport [Електронний ресурс] – Режим доступу до ресурсу: https://terrasport.ua/ua/
11. Онлайн-магазин бренду Nike [Електронний ресурс] – Режим доступу до ресурсу: https://www.nike.com/
12. Stoyko T. Angular Vs React Vs Vue: The Main Differences And Use Cases [Електронний ресурс] / Tetiana Stoyko. – 2022. – Режим доступу до ресурсу: https://incora.software/insights/react-vs-angular-vs-vue
13. Is PHP Used In Backend Or Frontend? [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.intervue.io/blog/is-php-used-in-backend-or-frontend
14. Top Backend Technologies in 2023: choose the right one for your solution [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://doit.software/blog/backend-technologies
15. Stoyko T. The Major Benefits Of Django For Backend Development [Електронний ресурс] / Tetiana Stoyko. – 2022. – Режим доступу до ресурсу: https://incora.software/insights/benefits-of-django-for-backend
16. Fugbuyiro D. Why You Should Use Java for Backend Development [Електронний ресурс] / David Fugbuyiro. – 2023. – Режим доступу до ресурсу: https://www.freecodecamp.org/news/java-for-backend-web-development/
17. Sushil Kumar Tripathi. Top 7 Backend Web Development Frameworks 2019 [Електронний ресурс] / Sushil Kumar Tripathi. – 2019. – Режим доступу до ресурсу: https://www.kellton.com/kellton-tech-blog/top-7-backend-web-development-frameworks-2019

18. Explain V8 engine in Node.js [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/explain-v8-engine-in-node-js/
19. Yurii Luchaninov. Using Node.js for Backend Web Development in 2023 [Електронний ресурс] / Yurii Luchaninov. – 2022. – Режим доступу до ресурсу: https://mobidev.biz/blog/node-js-for-backend-development
20. What Is Express.js? Everything You Should Know [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://kinsta.com/knowledgebase/what-is-express-js/
21. Durga Prasad Acharya. MongoDB vs MySQL: Which Is the Better Database Management System? [Електронний ресурс] / Durga Prasad Acharya. – 2022. – Режим доступу до ресурсу: https://kinsta.com/blog/mongodb-vs-mysql/
22. Sajdak M. JWT (JSON Web Token) (in)security [Електронний ресурс] / Michal Sajdak. – 2019. – Режим доступу до ресурсу: https://research.securitum.com/jwt-json-web-token-security/

ДОДАТОК 1

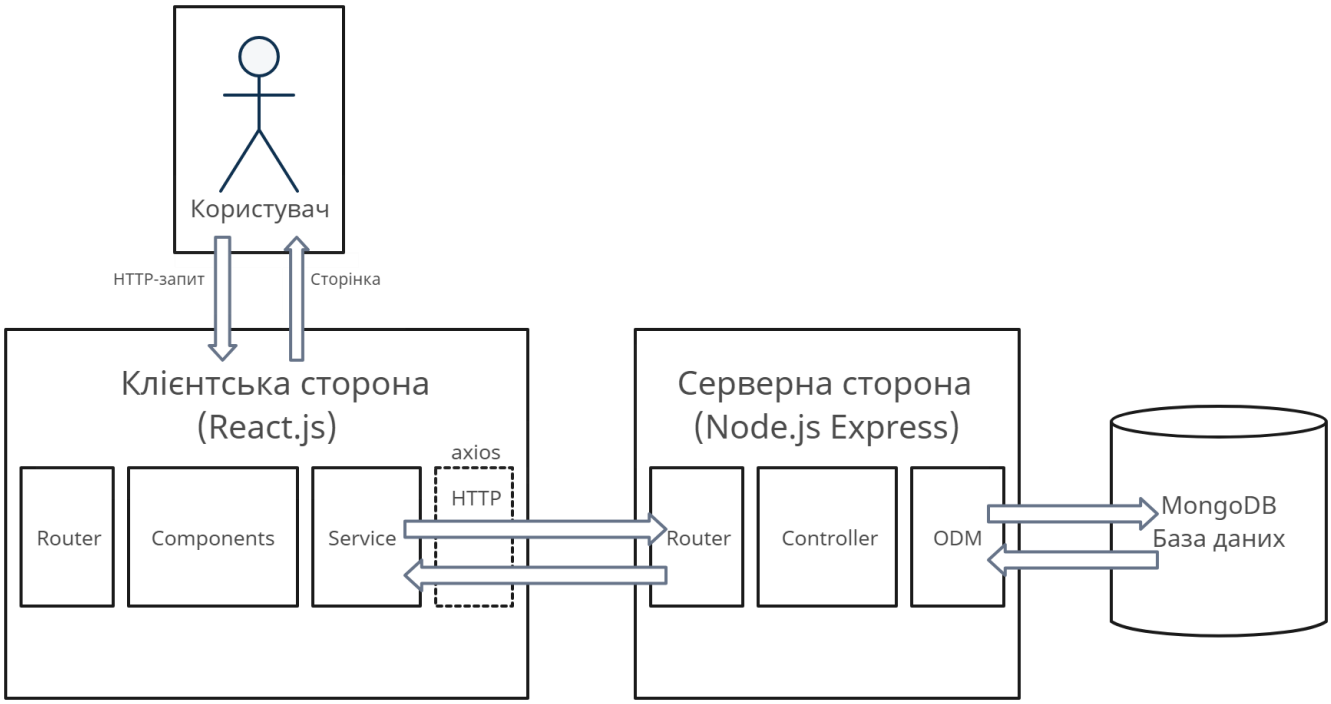
Веб-застосунок для продажу спортивних товарів

Компоненти програмного забезпечення (Структурна схема)

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.004 Д1		
		№ докум.	Підпис	Дата			
Розробив		Уткін В.А.			**Веб-застосунок для продажу спортивних товарів** Компоненти програмного забезпечення (структурна схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Пономаренко А.М.					
						КПІ ім. Ігоря Сікорського ФІОТ ІО-92	
Н. Контр.		Виноградов Ю.М.					
Затвердив		Стіренко С.Г.					

ДОДАТОК 2

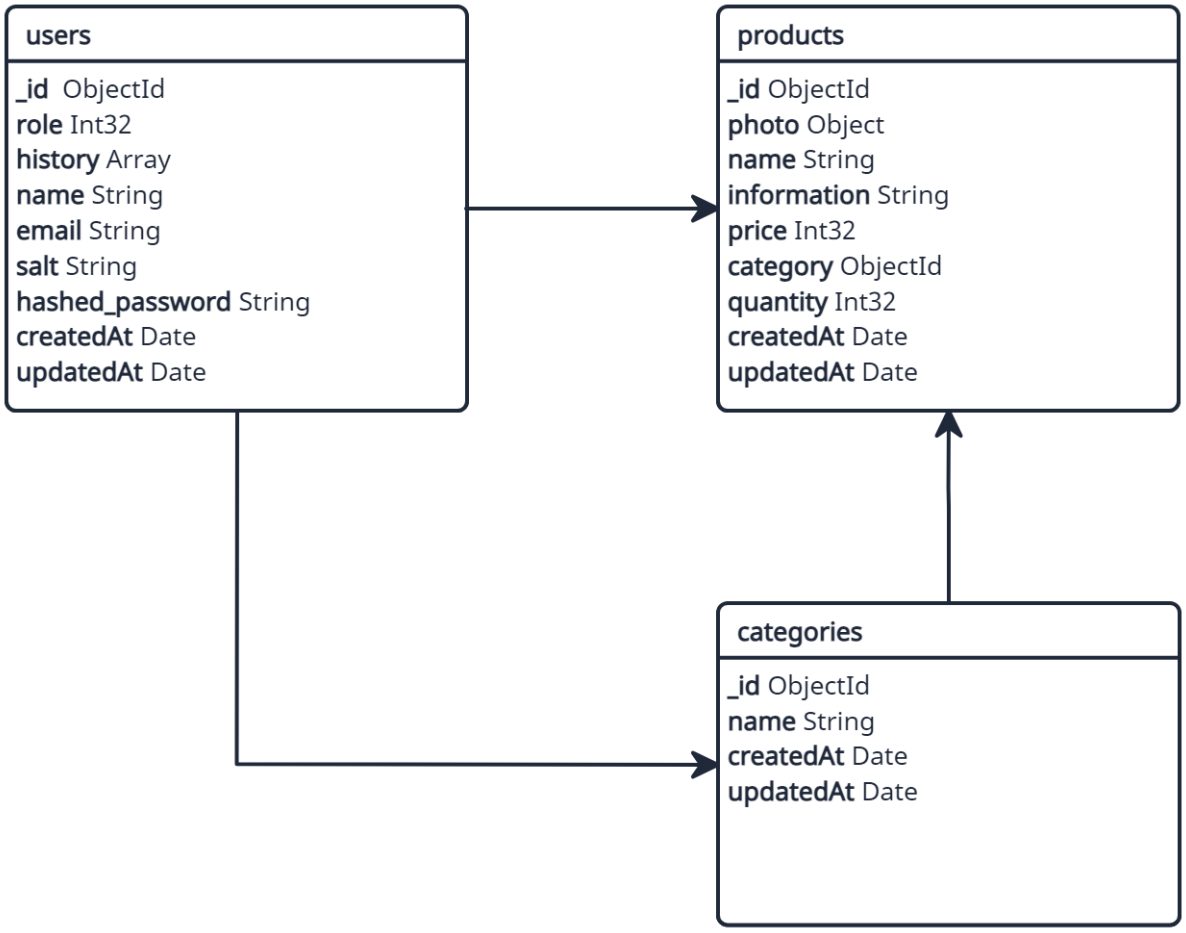
Веб-застосунок для продажу спортивних товарів

Діаграма бази даних (Функціональна схема)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Уткін В.А.			**Веб-застосунок для продажу спортивних товарів** Діаграма бази даних (функціональна схема)	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Пономаренко А.М.				
Н. Контр.		Виноградов Ю.М.				***КПІ ім. Ігоря Сікорського ФІОТ ІО-92***
Затвердив		Стіренко С.Г.				

ДОДАТОК 3

Веб-застосунок для продажу спортивних товарів

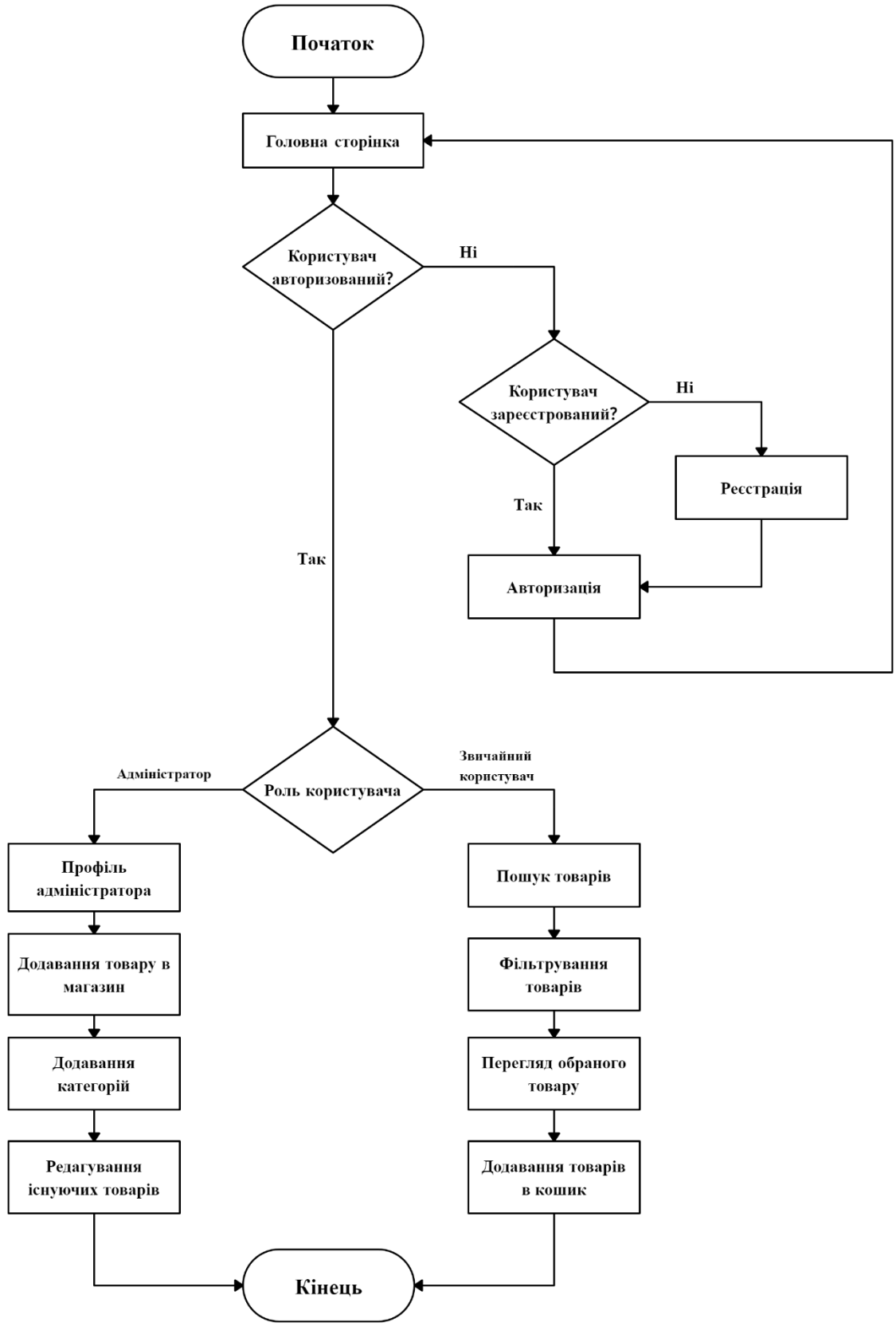
Алгоритм дій програмного забезпечення

(Принципова схема)

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2023 р



					ІАЛЦ.467200.006 Д3		
		№ докум.	Підпис	Дата			
Розробив		Уткін В. А.			**Веб-застосунок для продажу спортивних товарів** Алгоритм дій програмного забезпечення (принципова схема)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Пономаренко А.М.					
Н. Контр.		Виноградов Ю.М.				***КПІ ім. Ігоря Сікорського ФІОТ ІО-92***	
Затвердив		Стіренко С.Г.					

ДОДАТОК 4

Веб-застосунок для продажу спортивних товарів

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 15

Київ 2023 р

index.js

```
// Імпортування роутерів
const RoutesAuth = require('./routes/auth');
const RoutesUser = require('./routes/user');
const RoutesCategory = require('./routes/category');
const RoutesProduct = require('./routes/product');

const app = express();

// Підключення бази даних
const connectMongo = async () => {
  try {
    await mongoose.connect(
      process.env.MONGO_URI,
      {
        useNewUrlParser: true,
        useCreateIndex: true,
        useFindAndModify: false,
      }
    );
    console.log('База даних успішно підключена');
  } catch (err) {
    console.error(err.message);
    process.exit(1);
  }
};
connectMongo();
// Встановлення Middleware-функцій
app.use(morgan('dev'));
app.use(bodyParser.json());
app.use(cookieParser());
app.use(expressValidator());
app.use(cors());

// Маршрутування Middleware-функцій
app.use('/api', RoutesAuth);
app.use('/api', RoutesUser);
app.use('/api', RoutesCategory);
app.use('/api', RoutesProduct);

if (process.env.NODE_ENV === 'production') {
  app.use(express.static('client/build'));
  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'client', 'build', 'index.html'));
  });
}
```

```
const PORT = process.env.PORT;

app.listen(PORT, () => {
  console.log(`Сервер запущений на порті ${PORT}`);
});
```

routes/product.js

```
// Реалізація роутера товару

const express = require('express');
const router = express.Router();
const {
  create,
  findProductById,
  read,
  update,
  remove,
  list,
  listAssociated,
  listCategories,
  listBySearch,
  photo,
  listSearch
} = require('../controllers/product');
const { requireSignin, checkAuth, checkAdmin } = require('../controllers/auth');
const { FindUserById } = require('../controllers/user');

router.get('/product/:productId', read);
router.post('/product/create/:userId', requireSignin, checkAuth, checkAdmin,
create);
router.delete(
  '/product/:productId/:userId',
  requireSignin,
  checkAuth,
  checkAdmin,
  remove
);

router.put(
  '/product/:productId/:userId',
  requireSignin,
  checkAuth,
  checkAdmin,
  update
);
router.get('/products', list);
router.get('/products/search', listSearch);
router.get('/products/associated/:productId', listAssociated);
router.get('/products/categories', listCategories);
router.post('/products/by/search', listBySearch);
router.get('/product/photo/:productId', photo);
```

```
router.param('userId', findUserById);
router.param('productId', findProductById);

module.exports = router;
```

routes/user.js

```
// Роутер для користувача

const express = require('express');
const router = express.Router();

const { requireSignin, checkAuth, checkAdmin } = require('../controllers/auth');

const {
  findUserById,
  read,
  update,
} = require('../controllers/user');
router.get('/secret/:userId', requireSignin, checkAuth, checkAdmin, (req, res) =>
{
  res.json({
    user: req.profile,
  });
});

router.get('/user/:userId', requireSignin, checkAuth, read);
router.put('/user/:userId', requireSignin, checkAuth, update);
router.get('/orders/by/user/:userId', requireSignin, checkAuth);
router.param('userId', findUserById);

module.exports = router;
```

controllers/product.js

```
const formidable = require('formidable');
const _ = require('lodash');
const fs = require('fs');
const Product = require('../models/product');
const { errorHandler } = require('../helpers/MyErrorHandler');

exports.findProductById = (req, res, next, id) => {
  Product.ById(id)
    .populate('category')
    .exec((err, product) => {
      if (err || !product) {
        return res.status(400).json({
          error: 'Товар не знайдено',
        });
      }
      req.product = product;
```

```
      next();
    });
};

exports.read = (req, res) => {
  req.product.photo = undefined;
  return res.json(req.product);
};

exports.create = (req, res) => {
  let form = new formidable.IncomingForm();
  form.keepExtensions = true;
  form.parse(req, (err, fields, files) => {
    if (err) {
      return res.status(400).json({
        error: 'Зображення не може бути завантажене',
      });
    }
    // Перевірка для всіх полів
    const { name, description, price, category, quantity} = fields;

    if (
      !name ||
      !information ||
      !price ||
      !category ||
      !quantity
    ) {
      return res.status(400).json({
        error: 'Всі поля необхідно заповнити',
      });
    }

    let product = new Product(fields);


    if (files.photo) {
      if (files.photo.size > 1000000) {
        return res.status(400).json({
          error: 'Зображення повинно бути менше 1мб',
        });
      }
      product.photo.data = fs.readFileSync(files.photo.path);
      product.photo.contentType = files.photo.type;
    }

    product.save((err, result) => {
      if (err) {
        console.log('Помилка додання товару ', err);
        return res.status(400).json({
          error: errorHandler(err),
```

```
        });
      }
      res.json(result);
    });
  });
};

exports.remove = (req, res) => {
  let product = req.product;
  product.remove((err, deletedProduct) => {
    if (err) {
      return res.status(400).json({
        error: errorHandler(err),
      });
    }
    res.json({
      message: 'Товар успішно видалено',
    });
  });
};

exports.update = (req, res) => {
  let form = new formidable.IncomingForm();
  form.parse(req, (err, fields, files) => {
    if (err) {
      return res.status(400).json({
        error: 'Зображення неможливо завантажити',
      });
    }

    let product = req.product;
    product = _.extend(product, fields);


    if (files.photo) {
      if (files.photo.size > 1000000) {
        return res.status(400).json({
          error: 'Зображення повинно бути менше 1мб',
        });
      }
      product.photo.data = fs.readFileSync(files.photo.path);
      product.photo.contentType = files.photo.type;
    }

    product.save((err, result) => {
      if (err) {
        return res.status(400).json({
          error: errorHandler(err),
        });
      }
      res.json(result);
```

```
    });
  });
};


exports.list = (req, res) => {
  let limit = req.query.limit ? parseInt(req.query.limit) : 6;
  let order = req.query.order ? req.query.order : 'asc';
  let sortBy = req.query.sortBy ? req.query.sortBy : '_id';

  Product.find()
    .select('-photo')
    .populate('category')
    .sort([[sortBy, order]])
    .limit(limit)
    .exec((err, products) => {
      if (err) {
        return res.status(400).json({
          error: 'Товари не найдені',
        });
      }
      res.json(products);
    });
};

exports.listAssociated = (req, res) => {
  let limit = req.query.limit ? parseInt(req.query.limit) : 6;
  Product.find({ _id: { $ne: req.product }, category: req.product.category })
    .limit(limit)
    .populate('category', '_id name')
    .exec((err, products) => {
      if (err) {
        return res.status(400).json({
          error: 'Товари не найдені',
        });
      }
      res.json(products);
    });
};

exports.listCategories = (req, res) => {
  Product.distinct('category', {}, (err, categories) => {
    if (err) {
      return res.status(400).json({
        error: 'Категорії не найдені',
      });
    }
    res.json(categories);
  });
};
```

```
exports.listBySearch = (req, res) => {
  let order = req.body.order ? req.body.order : 'desc';
  let sortBy = req.body.sortBy ? req.body.sortBy : '_id';
  let limit = req.body.limit ? parseInt(req.body.limit) : 100;
  let skip = parseInt(req.body.skip);
  let findArgs = {};


  for (let key in req.body.filters) {
    if (req.body.filters[key].length > 0) {
      if (key === 'price') {
        findArgs[key] = {
          $gte: req.body.filters[key][0],
          $lte: req.body.filters[key][1],
        };
      } else {
        findArgs[key] = req.body.filters[key];
      }
    }
  }

  Product.find(findArgs)
    .select('-photo')
    .populate('category')
    .sort([[sortBy, order]])
    .limit(limit)
    .exec((err, data) => {
      if (err) {
        return res.status(400).json({
          error: 'Товари не знайдені',
        });
      }
      res.json({
        size: data.length,
        data,
      });
    });
};

exports.photo = (req, res, next) => {
  if (req.product.photo.data) {
    res.set('Content-Type', req.product.photo.contentType);
    return res.send(req.product.photo.data);
  }
  next();
};

exports.listSearch = (req, res) => {
  // створити об'єкт запиту для зберігання значення пошуку та значення категорії
  const query = {};
  // призначити пошукове значення query.name
```

```
  if (req.query.search) {
    query.name = { $regex: req.query.search, $options: 'i' };
  // присвоєння значення категорії для query.category
  if (req.query.category && req.query.category != 'All') {
      query.category = req.query.category;
    }
    // знайти продукт на основі об'єкта запиту з 2 властивостями
    // пошук і категорія
    Product.find(query, (err, products) => {
      if (err) {
        return res.status(400).json({
          error: errorHandler(err),
        });
      }
      res.json(products);
    }).select('-photo');
  }
};
```

user/AdminPanel.js

```
//Реалізація адмін-панелі

import React from 'react';
import Layout from '../core/Layout';
import { isAuthenticated } from '../auth';
import { Link } from 'react-router-dom';

const AdminPanel = () => {
  const {
    user: { _id, name, email, role },
  } = isAuthenticated();

  const adminTools = () => {
    return (
      <div className='card'>
        <h4 className='card-header'>Посилання</h4>
        <ul className='list-group'>
          <li className='list-group-item'>
            <Link className='nav-link' to='/create/category'>
              Додати категорію
            </Link>
          </li>
          <li className='list-group-item'>
            <Link className='nav-link' to='/create/product'>
              Додати товар
            </Link>
          </li>
          <li className='list-group-item'>
            <Link className='nav-link' to='/admin/products'>
              Управляти товарами
            </Link>
```

```
          </li>
        </ul>
      </div>
    );
  };

  const adminInfo = () => {
    return (
      <div className='card mb-5'>
        <h3 className='card-header'>Інформація про користувача</h3>
        <ul className='list-group'>
          <li className='list-group-item'>{name}</li>
          <li className='list-group-item'>{email}</li>
          <li className='list-group-item'>
            {role === 1 ? 'Адміністратор' : 'Зареєстрований користувач'}
          </li>
        </ul>
      </div>
    );
  };

  return (
    <Layout
      title=''
      description=''
      className='container-fluid'
    >
      <div className='row'>
        <div className='col-md-3'>{adminTools()}</div>
        <div className='col-md-9'>{adminInfo()}</div>
      </div>
    </Layout>
  );
};

export default AdminPanel;
```

user/signup.js

```
//Реалізація реєстрації

import Layout from '../core/Layout';
import { signup } from '../auth';

const useStyles = makeStyles((theme) => ({
  paper: {
    marginTop: theme.spacing(7),
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
```

```
  },
  avatar: {
    margin: theme.spacing(1),
    backgroundColor: theme.palette.secondary.main,
  },
  form: {
    width: '100%',
    marginTop: theme.spacing(1),
  },
  submit: {
    margin: theme.spacing(3, 0, 2),
  },
}));

export default function Signup() {
  const [values, setValues] = useState({
    name: '',
    email: '',
    password: '',
    error: '',
    success: false,
  });

  const { name, email, password, success, error } = values;

  const handleChange = (name) => (event) => {
    setValues({ ...values, error: false, [name]: event.target.value });
  };

  const clickSubmit = (event) => {
    event.preventDefault();
    setValues({ ...values, error: false });
    signup({ name, email, password }).then((data) => {
      if (data.error) {
        setValues({ ...values, error: data.error, success: false });
      } else {
        setValues({
          ...values,
          name: '',
          email: '',
          password: '',
          error: '',
          success: true,
        });
      }
    });
  };

  const showError = () => (
    <div
      className='alert alert-danger'
```

```
        style={{ display: error ? '' : 'none' }}
      >
        {error}
      </div>
    );

    const showSuccess = () => (
      <div
        className='alert alert-info'
        style={{ display: success ? '' : 'none' }}
      >
        Новий акаунт успішно створений. Будь ласка <Link
to='/signin'>авторизуйтесь</Link>.
      </div>
    );

    const classes = useStyles();
    const signUpForm = () => (
      <Container component='main' maxWidth='xs'>
        {showSuccess()}
        {showError()}
        <CssBaseline />
        <div className={classes.paper}>
          <Avatar className={classes.avatar}>
            <LockOutlinedIcon />
          </Avatar>
          <Typography component='h1' variant='h5'>
            Реєстрація
          </Typography>
          <form className={classes.form} noValidate>
            <Grid container spacing={2}>
              <Grid item xs={12}>
                <TextField
                  autoComplete='off'
                  onChange={handleChange('name')}
                  type='text'
                  name='name'
                  value={name}
                  variant='outlined'
                  required
                  fullWidth
                  id='name'
                  label="Ім'я"
                  autoFocus
                />
              </Grid>
              <Grid item xs={12}>
                <TextField
                  variant='outlined'
                  required
                  fullWidth
```

```
                id='email'
                label='Електронна пошта'
                name='email'
                onChange={handleChange('email')}
                type='email'
                value={email}
                autoComplete='off'
              />
            </Grid>
            <Grid item xs={12}>
              <TextField
                variant='outlined'
                required
                fullWidth
                name='password'
                label='Пароль'
                type='password'
                id='password'
                onChange={handleChange('password')}
                value={password}
                autoComplete='current-password'
              />
            </Grid>
          </Grid>
          <Button
            type='submit'
            fullWidth
            variant='contained'
            color='primary'
            className={classes.submit}
            onClick={clickSubmit}
          >
            Зареєструватися
          </Button>
          <Grid container justify='flex-end'>
            <Grid item>
              <Link to='/signin' variant='body2'>
                Вже маєте акаунт? Увійдіть
              </Link>
            </Grid>
          </Grid>
        </form>
      </div>
    </Container>
  );

  return (
    <Layout
      title=''
      description=''
      className='container col-md-8 offset-md-2'
```

```
    >
      {signUpForm()}
    </Layout>
  );
}
```

core/shop.js

```
// Реалізація сторінки магазину

import Search from './Search';
import { prices } from './RangePrices';

const Shop = () => {
  const [myFilters, setMyFilters] = useState({
    filters: { category: [], price: [] },
  });

  const [categories, setCategories] = useState([]);
  const [error, setError] = useState(false);
  const [limit, setLimit] = useState(6);
  const [skip, setSkip] = useState(0);
  const [size, setSize] = useState(0);
  const [filteredResults, setFilteredResults] = useState([]);

  const init = () => {
    getCategories().then((data) => {
      if (data.error) {
        setError(data.error);
      } else {
        setCategories(data);
      }
    });
  };

  const loadFilteredResults = (newFilters) => {
    getFilteredProducts(skip, limit, newFilters).then((data) => {
      if (data.error) {
        setError(data.error);
      } else {
        setFilteredResults(data.data);
        setSize(data.size);
        setSkip(0);
      }
    });
  };

  const loadMore = () => {
    let toSkip = skip + limit;
    getFilteredProducts(toSkip, limit, myFilters.filters).then((data) => {
      if (data.error) {
        setError(data.error);
      } else {
```

```
      setFilteredResults([...filteredResults, ...data.data]);
      setSize(data.size);
      setSkip(toSkip);
    }
  });
};

const useStyles = makeStyles((theme) => ({
  btn: {
    background: 'black',
    borderRadius: 3,
    border: 0,
    color: 'white',
    height: 48,
    padding: '0 20px',
    boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
  },
}));

const classes = useStyles();

const loadMoreButton = () => {
  return (
    size > 0 &&
    size >= limit && (
      <Button onClick={loadMore} variant='contained' className={classes.btn}>
        Завантажити ще
      </Button>
    )
  );
};

useEffect(() => {
  init();
  loadFilteredResults(skip, limit, myFilters.filters);
}, []);

const handleFilters = (filters, filterBy) => {
  const newFilters = { ...myFilters };
  newFilters.filters[filterBy] = filters;

  if (filterBy === 'price') {
    let priceValues = handlePrice(filters);
    newFilters.filters[filterBy] = priceValues;
  }
  loadFilteredResults(myFilters.filters);
  setMyFilters(newFilters);
};

const handlePrice = (value) => {
  const data = prices;
```

```
    let array = [];

    for (let key in data) {
      if (data[key]._id === parseInt(value)) {
        array = data[key].array;
      }
    }
    return array;
  };

  return (
    <Layout
      title=''
      description=''
      className='container-fluid'
    >
      <Search />
      <div className='row'>
        <div className='col-md-3'>
          <h4>Сортувати за категоріями</h4>
          <ul>
            <Checkbox
              categories={categories}
              handleFilters={(filters) => handleFilters(filters, 'category')}
            />
          </ul>

          <h4>Сортувати за диапазоном цін</h4>
          <div>
            <RadioBox
              prices={prices}
              handleFilters={(filters) => handleFilters(filters, 'price')}
            />
          </div>
        </div>
        <div className='col-md-9'>
          <h2 className='mb-2'>Товари</h2>
          <div className='row'>
            {filteredResults.map((product, i) => (
              <div key={i} className='col-xl-4 col-lg-6 col-md-12 col-sm-12'>
                <Card product={product} />
              </div>
            ))}
          </div>
          <hr />
          {loadMoreButton()}
        </div>
      </div>
    </Layout>
  );
};
```