МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

"До захисту допущений"
Зав. кафедри комп'ютерних наук
та прикладної математики
д.т.н., професор Турбал Ю.В.
«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
ТЕМА:
«ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБДОДАТКУ ДЛЯ ПРОДАЖУ ПОБУТОВОЇ ХІМІЇ»

Виконав: Негодюк Владислав Юрійович
студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки
групи КН-41

_____
підпис

Керівник: к.ф.-м. н , доцент Гладун Любомир Володимирович

_____
підпис

Рівне 2023

ЗМІСТ

Національний університет водного господарства та природокористування

ННІ інститут **автоматики, кібернетики та обчислювальної техніки**

Кафедра **комп'ютерних наук та прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань

Спеціальність **122 Комп'ютерні науки**

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

та прикладної математики

д.т.н., професор Турбал Ю.В.

«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

НЕГОДЮКУ ВЛАДИСЛАВУ ЮРІЙОВИЧУ

1. Тема роботи: Проектування та розробка веб додатку для продажу побутової хімії.

керівник роботи: Гладун Любомир Володимирович, к.ф.-м.н., доцент кафедри комп'ютерних наук та прикладної математики,

затверджені наказом вищого навчального закладу від "19" квітня 2023 року № С449

2. Термін подання роботи студентом: 5 червня 2023 р.

3. Вихідні дані до роботи: дані анкетування

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): розділ 1 - дослідження проблеми, розділ 2 - тасування елементів масиву випадковим чином, розділ - 3 розробка кінцевого програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): фрагменти основних функцій та скриптів, візуалізація основних етапів створення програмного продукту, діаграма розробленого алгоритму

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	Гладун Л.В.	30.10.22	30.10.22
Розділ 2	Гладун Л.В..	12.01.23	12.01.23
Розділ 3	Гладун Л.В.	15.03.23	15.03.23

7. Дата видачі завдання 10 жовтня 2022 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Опрацювання ідеї	30.10.2022 - 01.11.2022	
2	Вибір інструментів для розробки	01.11.2022 – 03.12.2022	
3	Розробка та реалізаціія бази данних	12.01.2023- 14.01.2023	
4	Розробка серверної частини додатка	14.01.2023- 20.01.2023	Розробка та підключення серверу до існуючої бази

			даних
5	Розробка UI та UX дизайну додатку	24.03.2023 – 26.03.2023	Написання html та css
6	Розробка візуальної частини веб додатку	28.03.2023 – 05.04.2023	Розробка Angular проекту
7	Тестування та удосконалення	06.04.2023 – 10.04.2023	

Студент Негодюк В.Ю

підпис прізвище та ініціали

Керівник роботи Гладун Л.В

підпис прізвище та ініціали

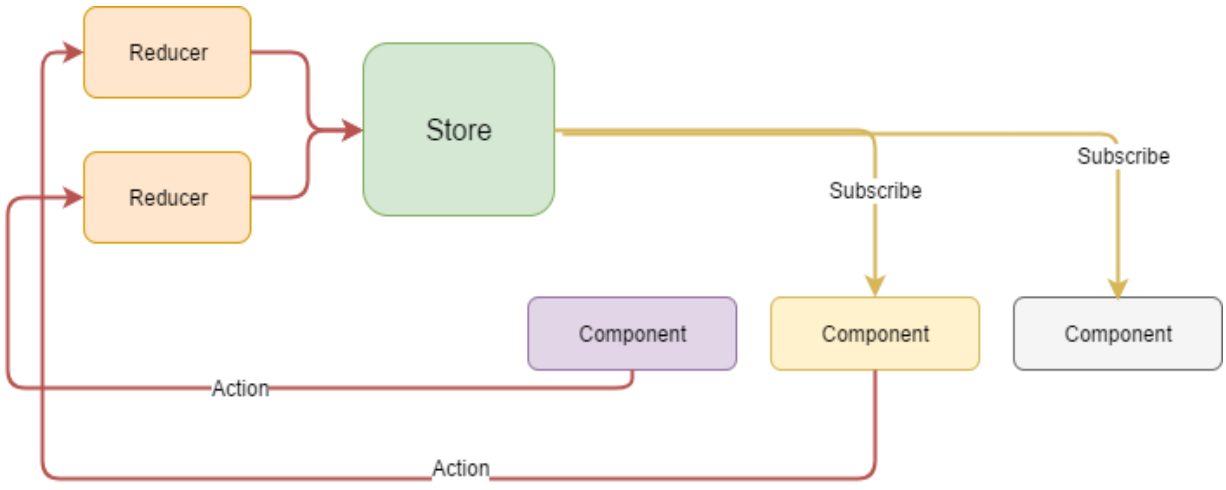
ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Технічне завдання полягає в проектуванні та розробці вебдодатку для продажу побутової хімії, також передбачає опис загальних вимог, які включають функціональність та інтерфейс додатку.

Основні вимоги до функціональності включають такі елементи як пошук та фільтрація продуктів, наявність каталогу продуктів, можливість складання замовлення через корзину та оформлення замовлення, а також можливість керування замовленнями.

Щодо інтерфейсу, він має бути зручним та привабливим для користувачів. Важливим аспектом є логічна навігація між сторінками, яка дозволяє користувачам швидко знаходити потрібну інформацію та здійснювати покупки без зайвих зусиль.

Загальна мета технічного завдання полягає в створенні веб додатку, який забезпечує зручний та ефективний процес продажу побутової хімії, забезпечує безпеку даних користувачів та надійність оплати, а також відповідає останнім тенденціям у дизайні та функціональності.



У розробці вебдодатку для продажу побутової хімії будуть використовуватись сучасні методи розробки, такі як Agile-підхід (наприклад, Scrum або Kanban), який забезпечує гнучкість та швидкість розробки. Також буде використовуватись архітектурний стиль REST API для стандартизованої взаємодії між клієнтською та серверною частиною додатку. Це дозволить

забезпечити ефективну комунікацію між компонентами системи та реалізувати доступ до функціональності додатку через HTTP-протокол.

РЕФЕРАТ

Кваліфікаційна робота: 50 сторінок, 22 рисунка, 5 таблиць, 17 джерел

Мета роботи: проектування та розробка вебдодатку для продажу побутової хімії. Робота спрямована на створення зручного та привабливого інтерфейсу, забезпечення безпеки та захисту даних користувачів, а також надання функціональності, необхідної для успішного ведення онлайн-бізнесу з продажу побутової хімії.

Засоби розробки: ASP.NET CORE API, Entity Framework, MSSQL, Angular, RxJs, Typescript, GitHub

Актуальність роботи: полягає у розробці зручного веб-сервісу для продажу побутової хімії.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI (User Interface) – Інтерфейс користувача. Тобто те, що бачить користувач при використанні додатку.

RxJs – бібліотека для реактивного асинхронного програмування в Angular.

СУБД – система управління базами даних.

API - (Application Programming Interface) це набір готових функцій, процедур, протоколів та інструментів, які дозволяють різним програмним компонентам взаємодіяти між собою. API визначає, які запити можуть бути виконані та які дані можуть бути передані між програмними компонентами.

IDE - (Integrated Development Environment) це інтегроване середовище розробки, що надає зручний інтерфейс для розробки програмного забезпечення. IDE поєднує в собі різні інструменти, сервіси та редактори, які допомагають програмістам ефективно працювати над розробкою програмного коду.

JSON - (JavaScript Object Notation) є легким форматом обміну даними, який базується на синтаксисі JavaScript. Він використовується для представлення структурованих даних у вигляді тексту, що може бути легко зрозумілим для людей та легко оброблюваним комп'ютерами.

EF – Entity Framework

ВСТУП

Розвиток сучасних технологій та інтернету значно змінив спосіб, яким ми спілкуємося, шукаємо інформацію та здійснюємо покупки. Віртуальний простір став важливим каналом для комерційної діяльності, а електронна комерція стала невід'ємною частиною сучасного бізнесу.

З розвитком інтернет-технологій з'явилися нові можливості для продажу товарів та послуг через веб додатки. Одним з таких сегментів є продаж побутової хімії, який відіграє важливу роль у повсякденному житті багатьох людей.

Ця дипломна робота має на меті проектування та розробку веб додатку для продажу побутової хімії, що відповідає потребам сучасних споживачів. Веб додаток надасть зручний та привабливий інтерфейс для вибору та замовлення продуктів побутової хімії, забезпечуючи швидку та безпечну покупку з будь-якого пристрою, що має доступ до Інтернету.

У рамках розробки вебдодатку будуть використані сучасні методи розробки, такі як Agile-підхід, який забезпечить ефективне керування проектом та швидкий розвиток додатку. Також буде використовуватись REST API для забезпечення взаємодії між клієнтською та серверною частиною додатку.

Для досягнення успіху на ринку побутової хімії важливо також враховувати обставини на ринку, такі як зростання популярності електронної комерції, конкурентний характер ринку, зміни в споживацьких звичках та впровадження інновацій.

РОЗДІЛ 1. СТЕК ВИКОРИСТАНИХ У ПРОЕКТІ ТЕХНОЛОГІЙ

1.1 Загальні технології

Інтернет є світовою мережею комп'ютерів, що з'єднує мільйони пристроїв по всьому світу. Це глобальна система комунікації, яка дозволяє обмінюватися інформацією та взаємодіяти з іншими користувачами, незалежно від їх географічного розташування [1].

Інтернет заснований на протоколах передачі даних, зокрема Internet Protocol (IP), що визначає правила передачі даних через мережу. Завдяки цим протоколам, комп'ютери та інші пристрої можуть обмінюватися пакетами даних і спілкуватися один з одним.

Основною характеристикою Інтернету є його відкритість та децентралізованість. Кожен користувач може підключитися до Інтернету та вільно обмінюватися інформацією без обмежень. Інтернет не має централізованого управління, а замість цього базується на принципі розподіленої мережі, де кожен вузол мережі може взаємодіяти з іншими вузлами.

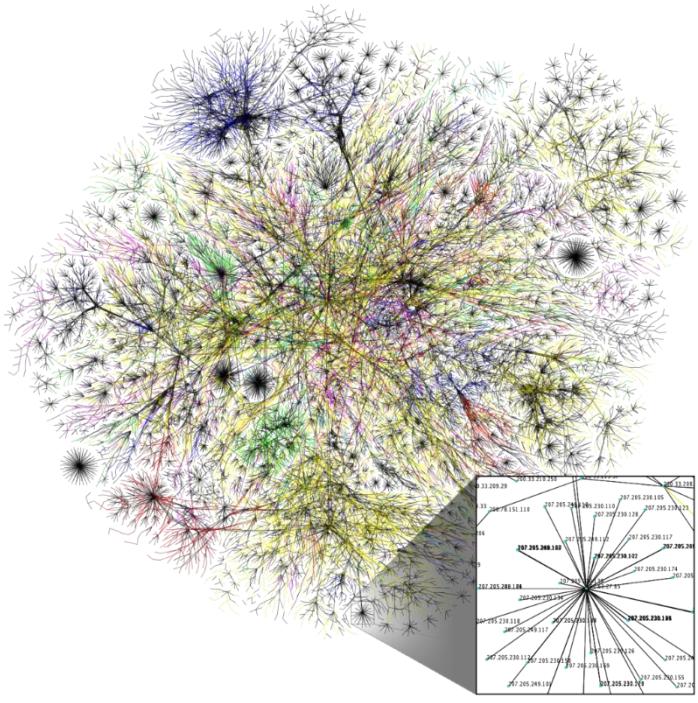


Рис. 1.1 Візуалізація усіх маршрутів інтернету від Opte Project

Веб-технології - це сукупність інструментів, протоколів, мов програмування та стандартів, які використовуються для розробки та функціонування веб-додатків і веб-сайтів. Вони дозволяють створювати динамічні, інтерактивні та зручні для використання веб-інтерфейси, забезпечують передачу та обробку даних між клієнтською та серверною частинами додатків.

Основні компоненти веб-технологій включають HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) і JavaScript. HTML використовується для структуризації та відображення контенту на веб-сторінках, CSS використовується для визначення стилів та вигляду цих сторінок, а JavaScript - для створення динамічних ефектів та взаємодії з користувачем [2].

Також веб-технології включають протоколи передачі даних, такі як HTTP (Hypertext Transfer Protocol), що використовується для передачі даних між веб-сервером та клієнтом, і WebSocket, який дозволяє зберігати постійне з'єднання між клієнтом та сервером для обміну даними в режимі реального часу.

Додаткові технології, такі як бази даних, серверні мови програмування (наприклад, PHP, Python, Ruby) і фреймворки (наприклад, Node.js, Django, Ruby on Rails), дозволяють розширити можливості веб-додатків та створити більш потужні та комплексні системи.

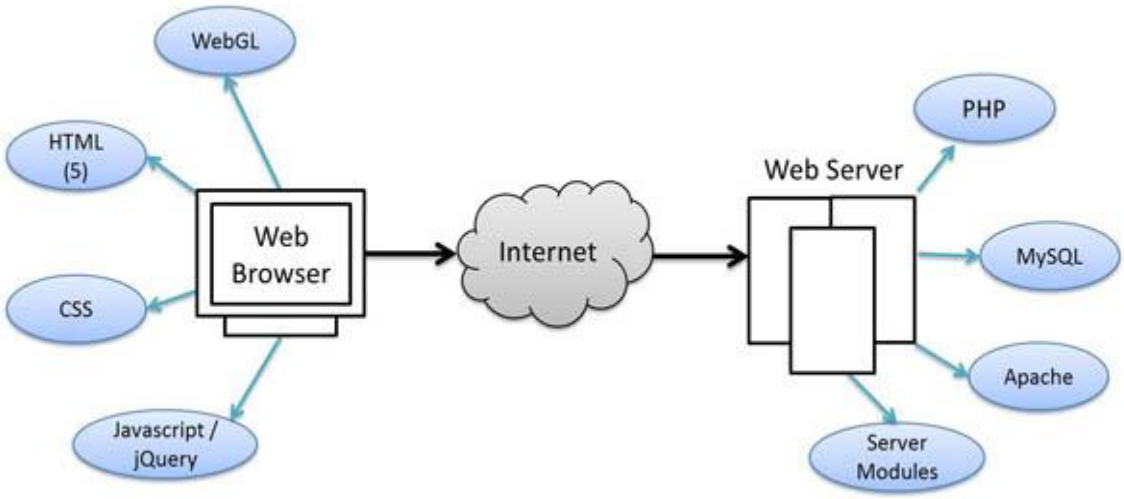


Рис. 1.2 Схематичний рисунок роботи веб-сервісів

1.2 Front-End стек

1.2.1 Angular

Angular - це веб-фреймворк, розроблений компанією Google, який використовується для створення високопродуктивних односторінкових веб-додатків. Він базується на мові програмування TypeScript і пропонує потужні інструменти для розробки і управління складними веб-інтерфейсами [3].

Angular пропонує компонентний підхід до розробки, де додаток будується з окремих компонентів, які мають свою логіку, шаблони та стилі. Це спрощує розподіл функціональності та роботу в команді, а також полегшує тестування та підтримку коду.

Один з головних принципів Angular - це двостороннє зв'язування даних, що дозволяє автоматично синхронізувати дані між моделью та представленням. Це забезпечує зручну роботу з формами та динамічне оновлення інтерфейсу при зміні даних [4].

Один з переваг Angular - це його активна спільнота розробників, яка надає багато ресурсів, документацію, приклади та розширення для спрощення розробки. Крім того, Angular має широкий набір інструментів, таких як Angular CLI, що допомагають в створенні, тестуванні та розгортанні додатків.

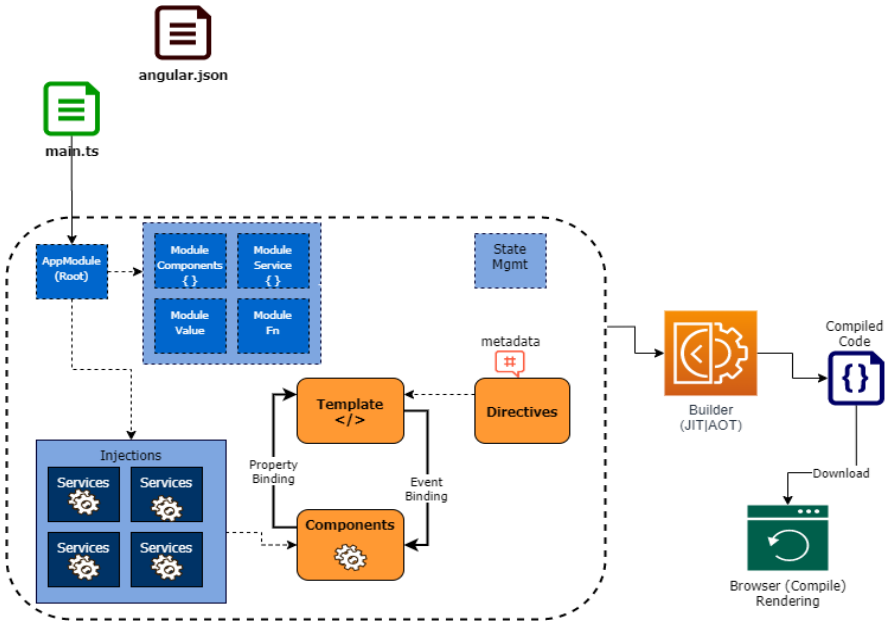


Рис. 1.3 Архітектура Angular проекту [5]

1.2.2 RxJs

RxJS (Reactive Extensions for JavaScript) - це бібліотека для програмування на мові JavaScript, яка реалізує парадигму реактивного програмування. Вона надає потужні інструменти для роботи з асинхронними та подійно-орієнтованими потоками даних [6].

Основною концепцією RxJS є спостережувані (Observables) - це послідовності подій, які можна спостерігати та підписуватися на них. Вони дозволяють асинхронно отримувати дані та реагувати на їх зміни.

RxJS надає багатий набір операторів, які дозволяють трансформувати, комбінувати та фільтрувати дані в потоці. Це дозволяє зручно працювати з асинхронними запитами, обробкою подій, обробкою помилок та багато іншого.

Одним з ключових переваг RxJS є можливість створювати складні асинхронні логіки, такі як злиття потоків, затримки, повторення, кешування і багато іншого, за допомогою невеликої кількості чітко визначених операторів.

RxJS можна використовувати в різних областях програмування, таких як розробка веб-додатків, мобільних додатків, розробка на боці сервера та багато іншого. Вона інтегрується з багатьма популярними фреймворками, такими як Angular, React, Vue.js, що робить її важливим інструментом для розробників [7].

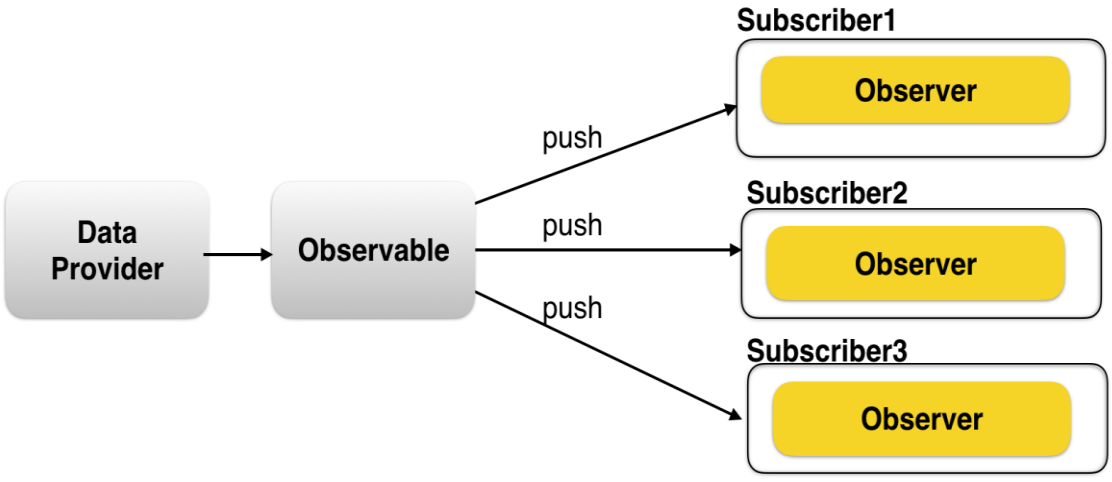


Рис. 1.4 Схематична архітектура RxJs

1.2.3 Typescript

TypeScript - це мова програмування, яка розширює стандартний синтаксис JavaScript, додаючи сильну типізацію та деякі нові функціональні можливості. Вона розроблена компанією Microsoft та є відкритим стандартом.

Основна мета TypeScript - полегшити розробку складних програм та забезпечити більшу надійність коду. Вона дозволяє визначати типи для змінних, параметрів функцій, властивостей об'єктів та інших елементів коду. Це допомагає знаходити та усувати помилки на етапі розробки, полегшує рефакторинг та підвищує читабельність коду.

TypeScript компілюється до звичайного JavaScript, що дозволяє запускати його на будь-якому сучасному браузері або виконувати на серверній стороні за допомогою платформи Node.js. Вона інтегрується з багатьма популярними фреймворками, такими як Angular, React та Vue.js, і підтримує розширення функціональності за допомогою сторонніх пакетів.

Одним з ключових переваг TypeScript є його здатність до розробки масштабованих проектів та роботи в команді. Він дозволяє використовувати модульну структуру, визначати імпорти та експорти, що полегшує організацію коду на великих проектах [8].

1.2.3 CSS/HTML

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для визначення зовнішнього вигляду веб-сторінок, написаних мовою HTML. Вона дозволяє розміщувати, форматувати та оформлювати елементи веб-сторінки, включаючи кольори, шрифти, розміри, межі, фони та інші стилістичні властивості.

CSS використовує селектори для вибору елементів сторінки та правила стилю для визначення їх вигляду. Завдяки каскадному природі CSS, стилі можуть успадковуватись, змінювати або переозначати в залежності від пріоритету селекторів та порядку підключення [9].

HTML (HyperText Markup Language) - це основна мова розмітки для створення веб-сторінок. Вона використовується для структурування та організації вмісту сторінки, включаючи текст, зображення, посилання, таблиці, форми та інші елементи.

HTML використовує теги для визначення типу та значення кожного елемента. Завдяки правильній структурі HTML-документу, браузери можуть коректно інтерпретувати та відображати вміст сторінки для користувачів.

CSS та HTML зазвичай використовуються разом для створення привабливих та функціональних веб-сторінок, де HTML відповідає за структуру та вміст, а CSS - за зовнішній вигляд та стилізацію [10].

1.2.4 NPM менеджер

NPM (Node Package Manager) - це пакетний менеджер для мови програмування JavaScript, який використовується в середовищі Node.js. Він дозволяє легко управляти залежностями проекту, встановлювати, оновлювати та видаляти пакети, а також керувати версіями пакетів.

NPM є найбільш поширеним пакетним менеджером для JavaScript і має велику кількість відкритих пакетів, які розроблені спільнотою розробників. Він дозволяє швидко використовувати готові рішення та бібліотеки, що спрощує розробку веб-додатків [11].

1.3 Back-End стек

ASP.NET Core є відкритою та переносною платформою розробки веб-додатків, розроблених компанією Microsoft. Вона є модульним та легким фреймворком, який дозволяє розробникам створювати швидкі та масштабовані додатки з використанням мов програмування C# або F#.

ASP.NET Core базується на попередній версії ASP.NET, але вона була повністю переписана, забезпечуючи новий підхід до розробки веб-додатків. Основні переваги ASP.NET Core включають кросс-платформену підтримку, високу продуктивність, модульність та гнучкість, а також інтеграцію з сучасними фронтенд технологіями.

ASP.NET Core є досить популярним фреймворком у розробці веб-додатків, особливо для побудови корпоративних та масштабованих додатків. Використання ASP.NET Core дозволяє розробникам ефективно використовувати мову програмування C# для створення потужних та безпечних веб-додатків, які можуть працювати на різних платформах [12].

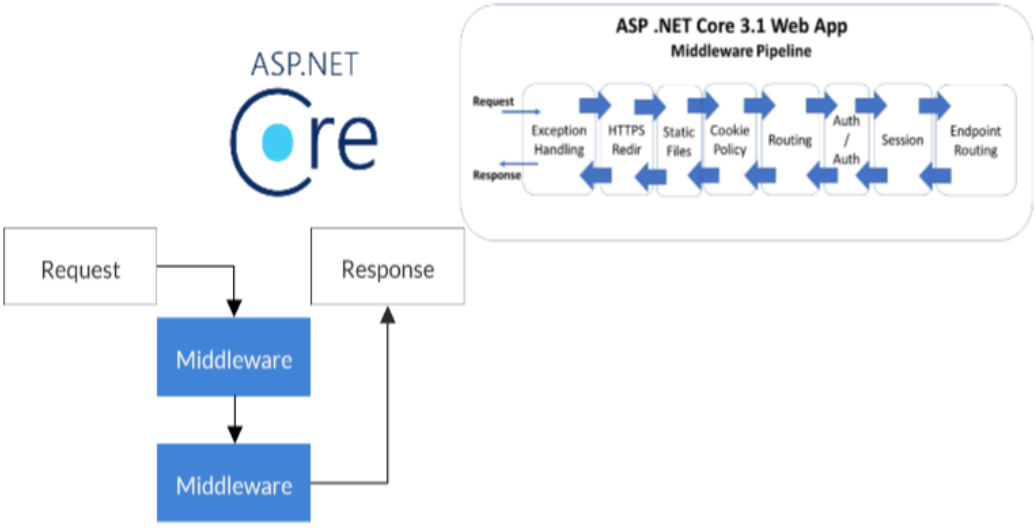


Рис. 1.5 Схема роботи ASP.NET CORE

ASP.NET Core також має кілька можливостей, що роблять його привабливим для розробки веб-додатків:

1. Крос-платформеність: ASP.NET Core підтримує роботу на різних платформах, включаючи Windows, macOS та Linux. Це дає розробникам велику свободу вибору оптимальної платформи для розгортання їх додатків.
2. Висока продуктивність: Завдяки оптимізаціям та вдосконаленням, ASP.NET Core забезпечує високу продуктивність та швидку обробку запитів. Він працює на базі потужного рантайму .NET Core, який оптимізований для швидкої виконавчої швидкості.
3. Модульна структура: ASP.NET Core використовує концепцію Middleware, що дозволяє розробникам легко додавати та налаштовувати компоненти для обробки запитів. Це робить розробку додатків гнучкою та розширюваною.
4. Інтеграція з сучасними технологіями: ASP.NET Core має гарну підтримку для роботи з сучасними фронтенд-фреймворками, такими як Angular, React та Vue.js. Це дозволяє розробникам легко інтегрувати фронтенд технології з серверною частиною додатку.
5. Висока безпека: ASP.NET Core надає широкий набір інструментів та функцій для забезпечення безпеки додатків, включаючи автентифікацію, авторизацію, захист від атак, обробку вразливостей та інші механізми безпеки.

Entity Framework (EF) - це фреймворк для роботи з базами даних у програмах на платформі .NET. Він надає зручність та продуктивність у розробці додатків, що працюють з базами даних, через використання об'єктно-орієнтованого підходу.

Основні можливості Entity Framework включають:

- Об'єктно-реляційне відображення: EF дозволяє розробникам працювати з базою даних, використовуючи об'єктно-орієнтовану модель. Таблиці бази даних відображаються на класи, а стовпці - на властивості цих класів. Це

спрощує роботу з даними та дозволяє зосередитися на бізнес-логіці додатку.

- Мова запитів LINQ: EF підтримує мову запитів LINQ (Language Integrated Query), що дозволяє розробникам виразно та зручно вибирати, фільтрувати та маніпулювати даними. Запити LINQ відображаються на відповідні запити SQL, які виконуються базою даних.
- Міграції бази даних: EF надає механізм міграцій, що дозволяє автоматично змінювати схему бази даних відповідно до змін у моделі додатку. Це спрощує процес розвитку та підтримки додатку, забезпечуючи автоматичне оновлення бази даних.
- Підтримка різних провайдерів баз даних: EF може працювати з різними провайдерами баз даних, такими як Microsoft SQL Server, MySQL, PostgreSQL та інші. Це дає розробникам свободу вибору бази даних, з якою вони хочуть працювати.

1.3.3 SQL

SQL (Structured Query Language) - це стандартна мова запитів та управління реляційними базами даних. Вона використовується для створення, модифікації та керування даними в базі даних. SQL дозволяє виконувати різні операції з даними, такі як вибірка, вставка, оновлення, видалення, сортування та групування.

РОЗДІЛ 2. СЕРВЕРНА ЧАСТИНА ВЕБ ДОДАТКУ

NuGet є пакетним менеджером для платформи розробки Microsoft .NET. Він дозволяє розробникам легко встановлювати, оновлювати та керувати залежностями програмного забезпечення в їх проектах.

НАЗВА	ВЕРСІЯ	ОПИС
EntityFramework	6.4.4	Фреймворк для маніпуляцій з базою даних
Microsoft.AspNet.Identity. EntityFramework	2.2.3	EF для роботи з identity
Microsoft.AspNetCore.Identity	2.2.0	Бібліотека для операцій з користувачем
Microsoft.AspNetCore.Identity. EntityFrameworkCore	6.0.16	EF для кросс-платформи
Microsoft.EntityFrameworkCore .Abstractions	7.0.5	Абстракції для EF
Microsoft.EntityFrameworkCore .Design	7.0.5	Бібліотека для опису дезайн-компонентів в моделях
Microsoft.EntityFrameworkCore .SqlServer	7.0.5	Драйвер для роботи з SQL базою даних
Microsoft.EntityFrameworkCore .Tools	7.0.5	Інструменти для EF
Swashbuckle.AspNetCore	6.5.0	Інструмент для підключення Swagger

В проекті база даних реалізована на MSSQL, тому що: SQL база даних може використовуватись для збереження будь-яких типів даних, включаючи тексти, числа, дати, зображення та багато іншого. Вона також може підтримувати комплексні запити з використанням умов, фільтрів та сортування для вибору потрібних даних [13].

База даних містить моделі даних. Моделі в базах даних визначають структуру та організацію даних. Вони використовуються для визначення сутностей, атрибутів та зв'язків між даними. Сутності представляють реальні або абстрактні об'єкти, про які зберігається інформація. Кожна сутність в базі даних має унікальний ідентифікатор.

Атрибути визначають характеристики або властивості сутностей. Кожен атрибут має назву і тип даних, який відображає його значення.

Зв'язки визначають взаємозв'язки між сутностями. Наприклад, можуть бути визначені зв'язки один до одного, один до багатьох або багато до багатьох. Зв'язки допомагають організувати взаємодію між різними сутностями в базі даних [14].

Моделі баз даних можуть бути представлені у різних форматах, таких як діаграми ER (Entity-Relationship), UML (Unified Modeling Language) або XML (Extensible Markup Language).

Назва	**Опис**
Product	Товар, який розміщується в магазині, містить усю інформацію про себе (назва, опис, ціна, статус товара, вага і тд.)
Producer	Модель виробника товарів
Category	Модель, яка представляє категорії товарів
ProductStatus	Відображає статус продукту (В наявності, закінчується, немає в наявності)

AspNetUser	Модель користувача в Identity
AspNetUsersRole	Модель ролі до користувача
AspNetUserLogin	Модель логіну користувача
AspNetRoles	Модель ролі (користувач, адмін і тд)
AspNetUsersToken	Модель токену користувача

Також базу даних можна показати у вигляді діаграми таблиць (рис 2.1)

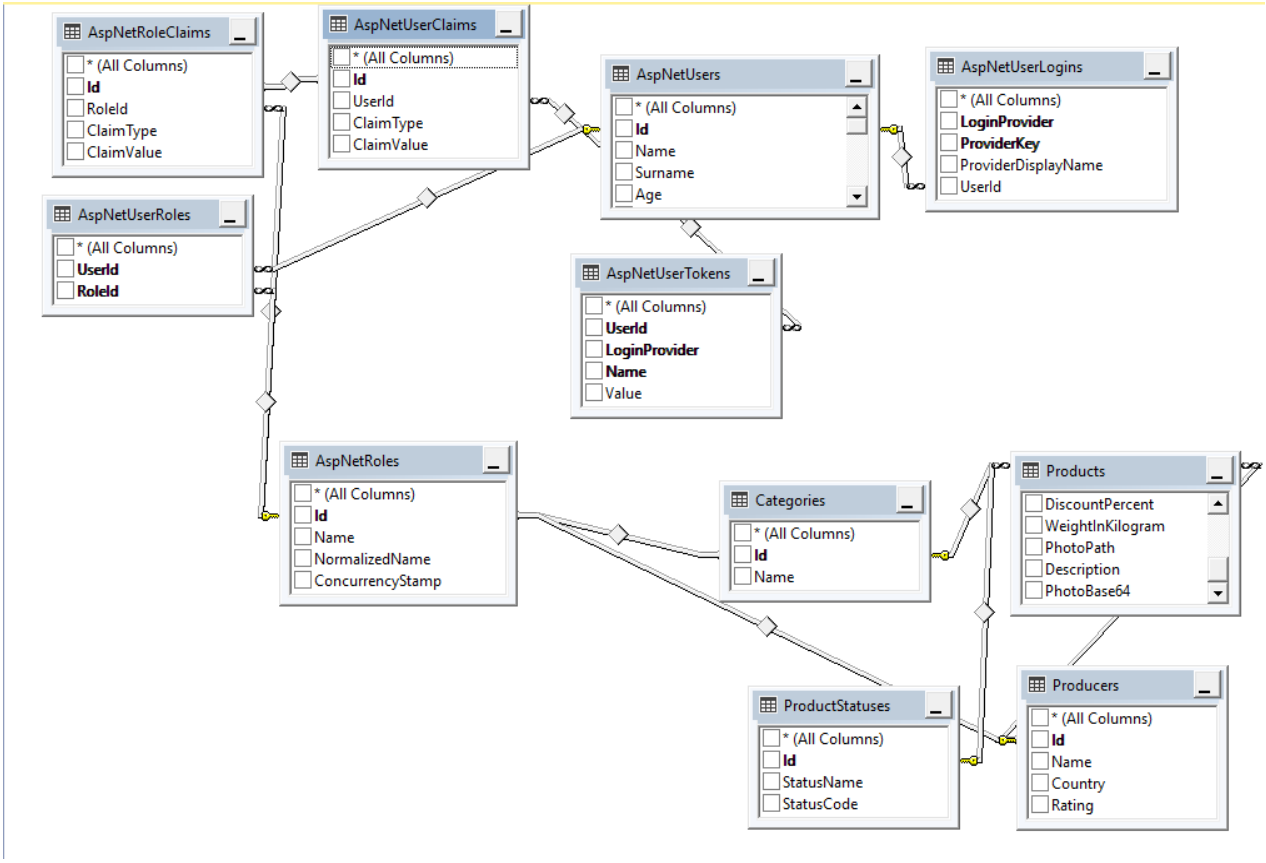


Рис. 2.1 Схема бази даних

А тепер давайте детальніше розглянемо моделі баз даних, у вигляді коду і з стороно EF.

1. Product model

```
public class Product
{
    public Product()
    {
        this.PhotoBase64 = "";
    }

    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public string Id { get; set; }
    public string Name { get; set; }
    public string CategoryId { get; set; }
    public string Article { get; set; }
    public string StatusId { get; set; }
    public float Price { get; set; }
    public string ProducerId { get; set; }
    public float DiscountPercent { get; set; }
    public float WeightInKilogram { get; set; }
    public string PhotoPath { get; set; }
    public string Description { get; set; }
    public string PhotoBase64 { get; set; }

    public virtual Category Category { get; set; }
    public virtual Producer Producer { get; set; }
    public virtual ProductStatus Status { get; set; }
}
```

Клас Product є моделлю, яка представляє продукт в системі. Ця модель містить різні властивості, що характеризують продукт:

- **Id** - унікальний ідентифікатор продукту (ключ)
- **Name** - назва продукту
- **CategoryId** - ідентифікатор категорії, до якої належить продукт
- **Article** - артикул продукту
- **StatusId** - ідентифікатор статусу продукту
- **Price** - ціна продукту
- **ProducerId** - ідентифікатор виробника продукту

- **DiscountPercent** - відсоток знижки на продукт
- **WeightInKilogram** - вага продукту в кілограмах
- **PhotoPath** - файловий шлях до фотографії продукту
- **Description** - опис продукту
- **PhotoBase64** - фотографія продукту у форматі Base64

Крім того, модель Product має віртуальні навігаційні властивості, які дозволяють звертатися до зв'язаних об'єктів:

- **Category** - категорія, до якої належить продукт
- **Producer** - виробник продукту
- **Status** - статус продукту

Ця модель використовує атрибути **Key**, **DatabaseGenerated** для визначення унікального ключа та автоматичної генерації значення для властивості **Id**. Це допомагає забезпечити унікальність та ідентифікацію продукту в базі даних.

2.
```
public class Category
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public string Id { get; set; }
    public string Name { get; set; }
}
```

Клас **Category** є моделлю, яка представляє категорію продукту. Ця модель має наступні властивості:

- **Id** - унікальний ідентифікатор категорії (ключ)
- **Name** - назва категорії

Атрибути **Key** та **DatabaseGenerated** використовуються для визначення унікального ключа та автоматичної генерації значення для властивості Id. Це допомагає забезпечити унікальність та ідентифікацію категорії в базі даних.

Ця модель використовується для організації категорій, до яких належать продукти. Кожна категорія має унікальний ідентифікатор та назву, що дозволяє класифікувати продукти за певними категоріями.

3.
```
public class Producer
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public string Id { get; set; }

    public string Name { get; set; }
    public string Country { get; set; }
    public int Rating { get; set; }
}
```

Клас **Producer** є моделлю, яка представляє виробника продукту. Ця модель має наступні властивості:

- **Id** - унікальний ідентифікатор виробника (ключ)
- **Name** - назва виробника
- **Country** - країна, де знаходиться виробник
- **Rating** - рейтинг виробника

Атрибути **Key** та **DatabaseGenerated** використовуються для визначення унікального ключа та автоматичної генерації значення для властивості Id. Це допомагає забезпечити унікальність та ідентифікацію виробника в базі даних.

Модель **Producer** дозволяє зберігати інформацію про виробників продуктів, включаючи їх назву, країну розташування та рейтинг. Ця інформація може бути корисною при відображенні деталей про виробника на веб-сайті або в додатку.

4.
```
public class ProductStatus
{
    // Code 0 - avaliable
    // Code 1 - run out
    // Code 2 - not avaliable


    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public string Id { get; set; }
    public string StatusName { get; set; }
    public int StatusCode { get; set; }
}
```

Клас **ProductStatus** є моделлю, яка представляє статус продукту. Він має наступні властивості:

- **Id** - унікальний ідентифікатор статусу продукту (ключ)
- **StatusName** - назва статусу
- **StatusCode** - код статусу

Атрибути **Key** та **DatabaseGenerated** використовуються для визначення унікального ключа та автоматичної генерації значення для властивості Id.

Цей клас містить статуси, що відповідають різним кодам:

1. Код 0 відповідає статусу "available" (доступний)
2. Код 1 відповідає статусу "run out" (розпродано)
3. Код 2 відповідає статусу "not available" (не доступний)

Також проект містить таблиці з бібліотеки **Identity.** Бібліотека Identity ASP.NET Core надає розширену функціональність для роботи з аутентифікацією та авторизацією в додатках на основі ASP.NET Core. Вона включає в себе кілька таблиць для зберігання даних користувачів, ролей, клеймів та інших важливих елементів безпеки. Давайте розглянемо деякі з цих таблиць:

Таблиця **AspNetUsers**:

- **Id**: унікальний ідентифікатор користувача
- **UserName**: ім'я користувача
- **NormalizedUserName**: нормалізоване ім'я користувача для пошуку та порівняння
- **Email**: електронна адреса користувача
- **NormalizedEmail**: нормалізована електронна адреса користувача для пошуку та порівняння
- **PasswordHash**: хеш пароля користувача

Таблиця **AspNetRoles**:

- **Id**: унікальний ідентифікатор ролі
- **Name**: назва ролі
- **NormalizedName**: нормалізована назва ролі для пошуку та порівняння

Таблиця **AspNetUserRoles**:

- **UserId**: ідентифікатор користувача, пов'язаний з роллю
- **RoleId**: ідентифікатор ролі, пов'язаної з користувачем

Ці таблиці, разом з іншими, які надає бібліотека Identity ASP.NET Core, дозволяють зберігати, керувати та перевіряти дані про користувачів, ролі та їх відношення в системі. Вони є основою для реалізації системи аутентифікації та авторизації в додатках ASP.NET Core [15].

2.3 Доступ до бази даних

Для доступу в базу даних в проекті використовується паттерн “Репозиторій”.

Паттерн репозиторій є одним зі структурних паттернів проектування і використовується для розділення бізнес-логіки програми від деталей доступу до даних. Він надає абстракцію над джерелом даних і дозволяє здійснювати операції збереження, отримання, оновлення та видалення об'єктів у базі даних.

Основна ідея паттерна репозиторій полягає в тому, що весь код, пов'язаний з доступом до даних, виокремлюється в окремий клас-репозиторій. Цей клас визначає інтерфейс, який включає методи для виконання різних операцій з об'єктами доменної моделі. Класи, які використовують цей репозиторій, взаємодіють з даними тільки через цей інтерфейс, а не безпосередньо з базою даних.

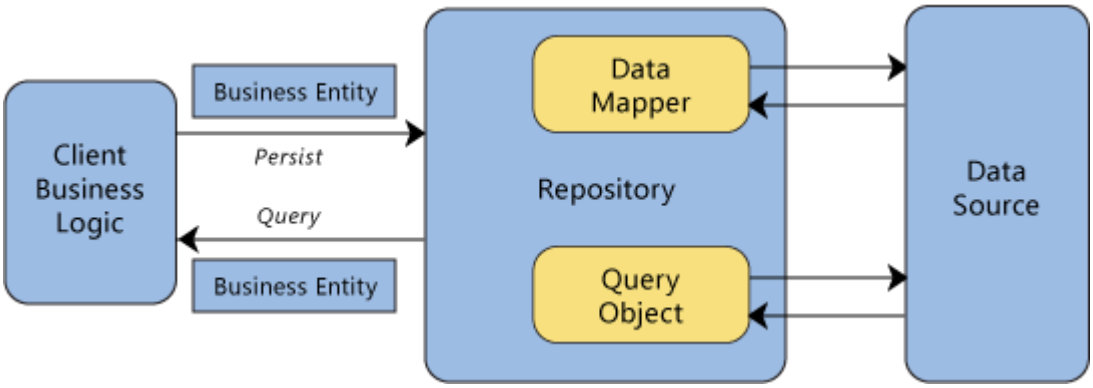


Рис. 2.2 Схематична архітектура паттерну “Репозиторій”

В проекті реалізована абстракція у вигляді інтерфейсу **IRepository,** який містить декілька методів.

```
public interface IRepository<T> where T : class
{
    IEnumerable<T> GetAll();
    T GetById(string EntityId);
    void Insert(T entity);
    void Delete(string EntityId);
    int GetCount();
    void Save();
}
```

Назва	Опис
`IEnumerable<T> GetAll()`	Отримати усі сутності з бази даних
`T GetById(string EntityId)`	Отримати сутність по **id** ідентифікатору
`void Insert(T entity)`	Додати сутність в базу даних
`void Delete(string EntityId)`	Видалити сутність з бази даних
`int GetCount()`	Отримати кількість елементів в таблиці
`void Save()`	Зберегти стан контексту

Також для доступу у базу даних в Entity Framework потрібно створити контекст, який містить усі таблиці і способи доступу до них. Приклад буде нижче:

```
public class EFDbContext : IdentityDbContext<DbUser, IdentityRole, string,
IdentityUserClaim<string>,
    IdentityUserRole<string>, IdentityUserLogin<string>,
    IdentityRoleClaim<string>, IdentityUserToken<string>>
    {
        public EFDbContext(DbContextOptions<EFDbContext> options) : base(options)
        {
            Database.EnsureCreated();
        }

        public DbSet<Product> Products { get; set; }
        public DbSet<Producer> Producers { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<ProductStatus> ProductStatuses { get; set; }
}
```

2.4 Контроллери

Контролери в ASP.NET Core API відповідають за обробку запитів, маршрутизацію та відповідь на них. Вони є основною частиною реалізації паттерну проектування MVC (Model-View-Controller) у веб-додатку.

Контролери визначаються як класи, що успадковуються від базового класу Controller. У цих класах визначаються методи, які відповідають на різні HTTP-запити (GET, POST, PUT, DELETE і т.д.) та обробляють логіку для цих запитів.

Контролери використовують атрибути для визначення шляху маршрутизації (Route), типу запиту (HttpGet, HttpPost і т.д.), а також параметрів запиту (FromBody, FromRoute, FromQuery і т.д.). Завдяки цим атрибутам контролери встановлюють зв'язок між шляхом URL-запиту та методом, що його обробляє.

У методах контролерів можна виконувати різні дії, такі як отримання даних з бази даних, створення нових записів, оновлення чи видалення існуючих записів, а також відправлення відповідей на запити.

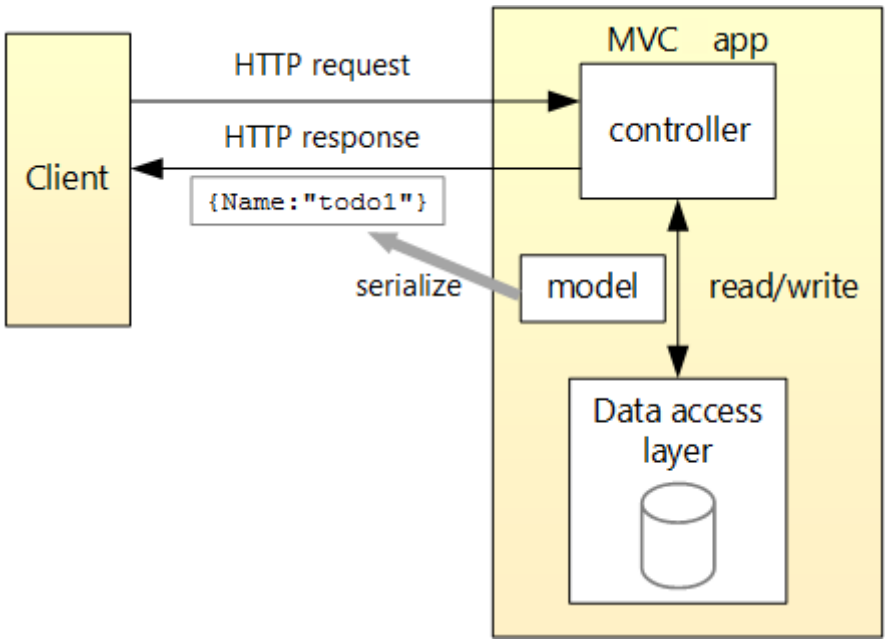


Рис. 2.3 Діаграма взаємодій ASP.NET Core

В проекті реалізована абстракція у вигляді **ApiController.**

```
[ApiController]
[Route("api/[controller]")]
[Produces("application/json")]
public class ApiController : ControllerBase
{
}
```

Атрибути **[ApiController]**, **[Route("api/[controller]")]** та **[Produces("application/json")]** вказують на специфікацію контролера в ASP.NET Core API.

- **[ApiController]**: Цей атрибут позначає клас контролера як контролер API, що надає різноманітні вигоди та автоматичні налаштування для обробки API-запитів. Використання цього атрибута спрощує роботу з Web API, включаючи автоматичну валідацію моделей, форматування відповідей та обробку помилок.
- **[Route("api/[controller]")]**: Цей атрибут визначає шаблон маршруту, за яким будуть розпізнаватися запити до контролера. У даному випадку, шлях маршруту буде формуватися на основі назви контролера. Наприклад, якщо назва контролера - "ProductController", то шлях маршруту буде "api/product".
- **[Produces("application/json")]**: Цей атрибут вказує, що контролер повинен генерувати відповіді у форматі JSON. Всі відповіді, пов'язані з цим контролером, будуть автоматично серіалізовані в JSON формат.

Клас **ApiController** успадковується від **ControllerBase** і надає базовий функціонал для реалізації контролерів API. Цей клас можна розширити, додавши власну логіку для обробки API-запитів та взаємодії з джерелами даних.

Застосування цих атрибутів та успадкування від ControllerBase допомагає створити потужний та зручний в розробці контролер для обробки API-запитів у форматі JSON [16].

Список усіх контроллерів:

Назва	Опис
ApiController	Контроллер, від якого наслідуються усі контроллери в програмі. Контроллер наслідує ControllerBase.
ProductController	Контроллер, який відповідає за продукти (товари).
CategoryController	Контроллер, який відповідає за категорії товарів.
ProducrCategory	Контроллер, який відповідає за виробників товарів.

2.5 Сервіси

Сервіси в ASP.NET є компонентами, що надають послуги та виконують функції в програмній системі. Вони дозволяють розділити відповідальності та функціональність системи на окремі компоненти, що сприяє кращій організації коду та полегшує його управління.

Однією з переваг використання сервісів є можливість перевикористання коду. Сервіси можуть бути використані у різних частинах програми, що зменшує дублювання коду та сприяє його ефективному використанню.

Крім того, сервіси допомагають покращити тестованість програмного коду. Оскільки вони виконують окремі функції та послуги, їх можна легко тестувати незалежно від інших компонентів системи. Це сприяє полегшенню розробки тестів та забезпечує високу якість програмного продукту.

Крім того, використання сервісів сприяє легкості підтримки та розширення програми. Завдяки розділенню функціональності на сервіси, модифікація або розширення одного сервісу не впливає на інші компоненти

системи. Це дозволяє зручно вносити зміни та розширювати функціональність без необхідності змінювати весь код.

Приклад сервісу:

```
public static class ImageService
    {
        public static string ImageToBase64(string path)
        {
            var currentDirectory = System.IO.Directory.GetCurrentDirectory() +
"\\Uploaded\\ProductsImages\\" + path + ".jpg";

            using (Image image = Image.FromFile(currentDirectory))
            {
                using (MemoryStream m = new MemoryStream())
                {
                    image.Save(m, image.RawFormat);
                    byte[] imageBytes = m.ToArray();

                    string base64String = "" +
Convert.ToBase64String(imageBytes);
                    return base64String;
                }
             }
          }
      }
```

Код зверху представляє статичний клас **ImageService**, який містить метод **ImageToBase64**. Цей метод використовується для перетворення зображення на формат **Base64**.

В методі **ImageToBase64** передається шлях до зображення. Використовуючи цей шлях, метод завантажує зображення з файлу за допомогою **Image.FromFile**. Після цього, створюється об'єкт **MemoryStream**, в який зображення зберігається у форматі, що відповідає його оригінальному формату.

Далі, зображення перетворюється на масив байтів за допомогою **m.ToArray()**. Цей масив байтів потім перетворюється на рядок **Base64** за допомогою **Convert.ToBase64String(imageBytes)**. Кінцевий рядок **Base64** містить інформацію про тип зображення та самі байти зображення.

На останньому кроці, до отриманого рядка **Base64** додається префікс **""**, який вказує на тип зображення. Після цього, результат повертається з методу.

Цей код корисний, коли потрібно перетворити зображення на формат **Base64**, наприклад, для передачі його через API або включення в HTML-розмітку.

РОЗДІЛ 3. ВІЗУАЛЬНА ЧАСТИНА ВЕБ ДОДАТКУ

Візуальна частина додатку включає в себе HTML, CSS та JavaScript, які використовуються для створення структури, оформлення та надання інтерактивності веб-сторінок. Також можуть використовуватись фреймворки та бібліотеки для швидкої розробки та покращення вигляду додатку. Графіка та мультимедіа елементи, такі як зображення, іконки, відео та аудіо, також можуть бути використані для створення візуальних ефектів та надання контенту.

Візуальна частина додатку є важливою, оскільки вона визначає вигляд і взаємодію користувача з додатком. Завдяки HTML можна створювати різноманітні елементи, такі як кнопки, поля вводу, таблиці та інші, які відображаються на сторінках додатку. CSS дозволяє стилізувати ці елементи, визначаючи кольори, розміри, шрифти, відступи та інші атрибути. JavaScript додає інтерактивність, динаміку та можливість обробки подій на сторінках додатку.

Фреймворки та бібліотеки, такі як Bootstrap, Foundation або Material UI, надають готові компоненти, стилізацію і розмітку, що спрощує розробку та забезпечує єдиноформний вигляд додатку.

Використання графіки та мультимедіа елементів додає візуальну привабливість до додатку. Зображення, іконки, відео та аудіо можуть бути використані для передачі інформації, навігації, створення ефектів або підкреслення важливих деталей.

Загалом, візуальна частина додатку грає важливу роль у створенні привабливого та функціонального інтерфейсу, який забезпечує зручну та приємну взаємодію користувача з додатком.

3.1 Домашня сторінка веб додатку

Домашня сторінка веб додатку виглядає наступним чином:

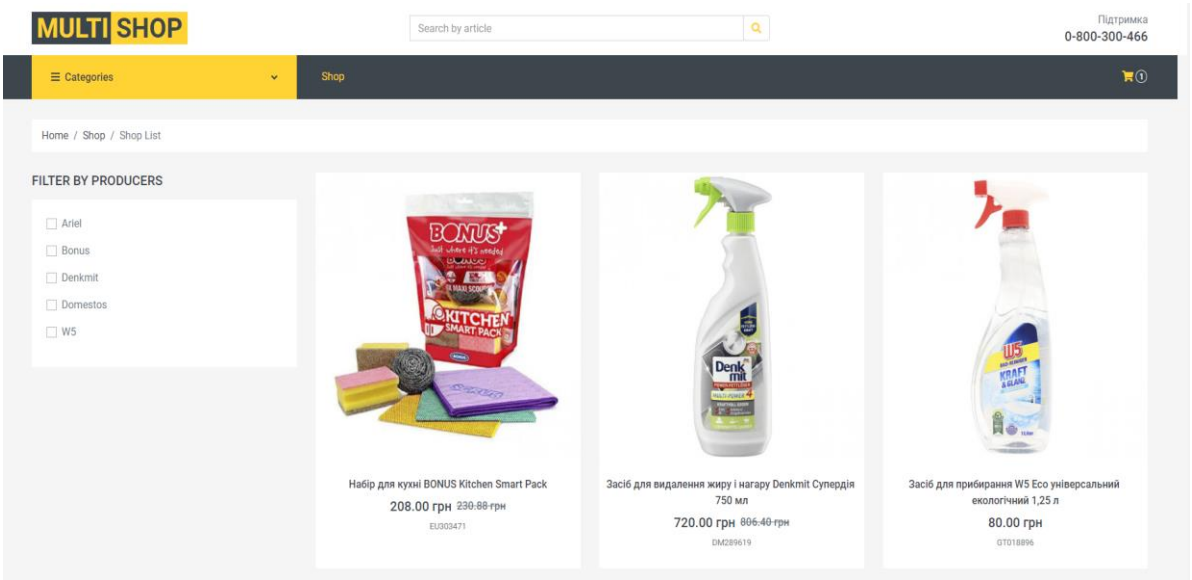


Рис. 3.1 Домашня сторінка

Домашня сторінка інтернет-магазину побутової хімії є ключовим елементом, що привертає увагу відвідувачів і створює перший враження про магазин. Основними елементами домашньої сторінки це:

- Логотип і назва магазину: Розміщення логотипу і назви магазину у верхній частині сторінки надає відвідувачам чітку ідентифікацію бренду.
- Навігаційне меню: Навігаційне меню зазвичай розташовується у верхній частині сторінки і містить посилання на основні категорії товарів або сторінки, такі як "Категорії", "Магазин" та інші.
- Рекомендовані товари або пропозиції: На домашній сторінці відображені рекомендовані товари або спеціальні пропозиції, що привертають увагу відвідувачів і спонукають їх до покупок.
- Пошукова рядок: Пошукова рядок зазвичай розташовується видною місці, де відвідувачі можуть шукати конкретні товари або категорії. На скріншоті строка пошуку розташована зверху по центру.
- Категорії товарів: Домашня сторінка може містить відображення основних категорій товарів, які продаються в магазині, з можливістю переходу до відповідних сторінок категорій. Також показується список виробників.

Ці елементи домашньої сторінки інтернет-магазину побутової хімії створюють зручне та привабливе середовище для відвідувачів, допомагають їм

зорієнтуватися в асортименті товарів, знайти необхідні продукти та зробити покупку. Для створення естетичного та функціонального дизайну домашньої сторінки можуть використовуватися сучасні веб-технології, CSS-анімації, а також принципи UX-дизайну.

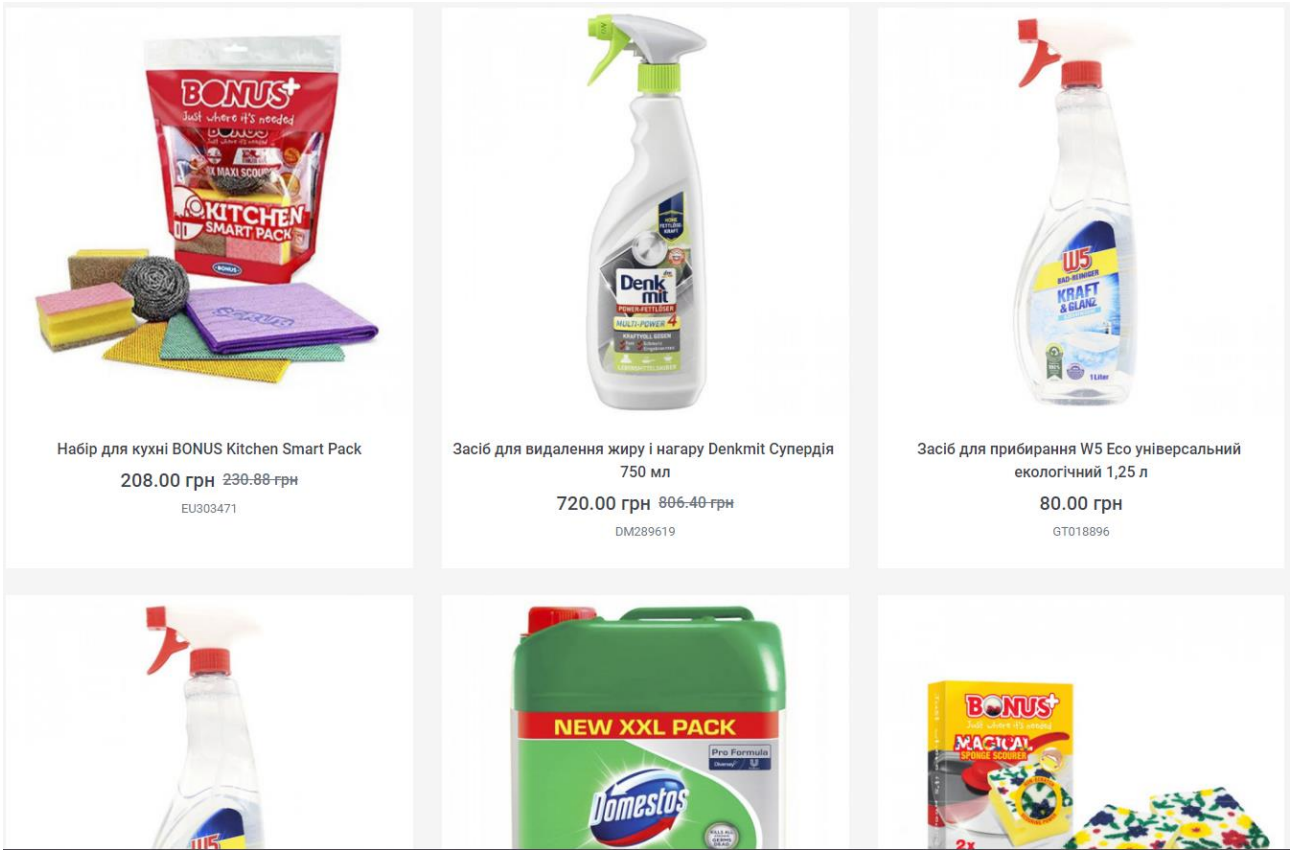


Рис. 3.2 Список товарів

На сторінці зі списком товарів (рис 3.2) можна побачити широкий вибір побутової хімії, представленої в веб додатку. Тут ви зможете ознайомитися з різноманітними продуктами, їх характеристиками та цінами.

Ця сторінка надає зручний спосіб перегляду товарів, де ви можете швидко прокручувати список та зупинятися на тих позиціях, які вас зацікавили. Кожен товар супроводжується зображенням, що дозволяє вам отримати візуальне уявлення про продукт.

Будьте упевнені, що ми надаємо вичерпну інформацію про кожен товар, включаючи назву, опис, характеристики, ціну та наявність на складі. Ви також можете знайти додаткові деталі.

Для вашої зручності ми можемо надати різні фільтри та опції сортування, щоб користувач міг швидко знайти потрібний товар. Наприклад, користувач може використовувати фільтри за категоріями, брендами і тд. Таким чином, ви зможете зосередитися на продуктах, які відповідають вашим потребам і вимогам.

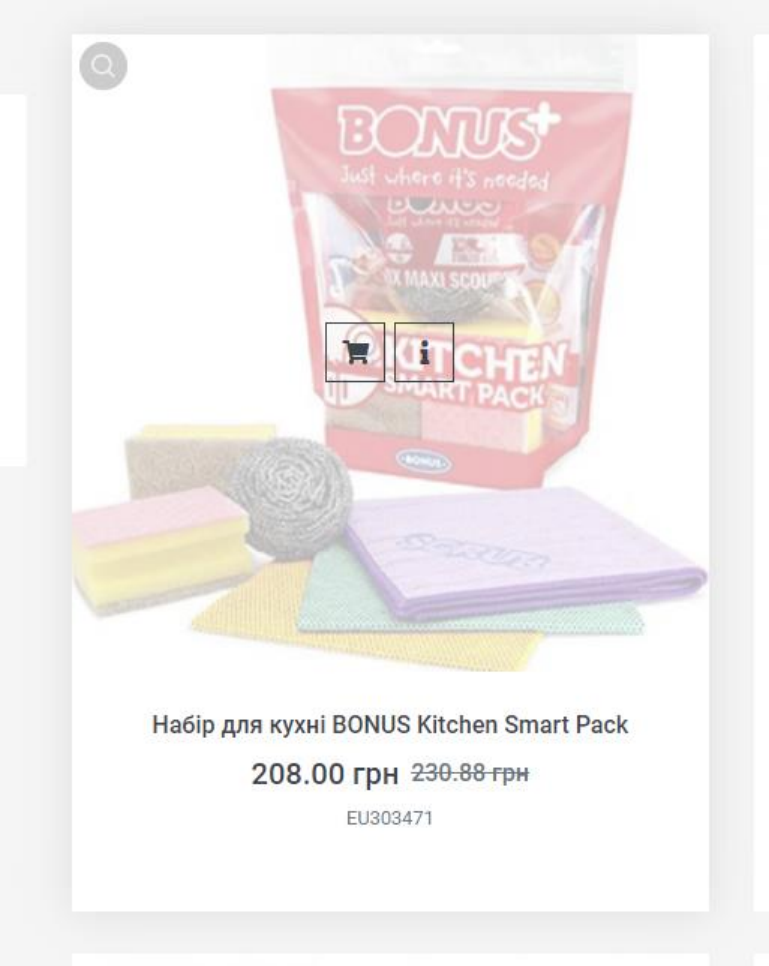


Рис. 3.3 Наведення курсора на продукт

На рисунку 3.3, ми бачимо картку з продуктом та всієї інформацією про нього таку як: фото продукту, ціна, знижка (якщо є), та артикль по якому можна легко знайти товар.

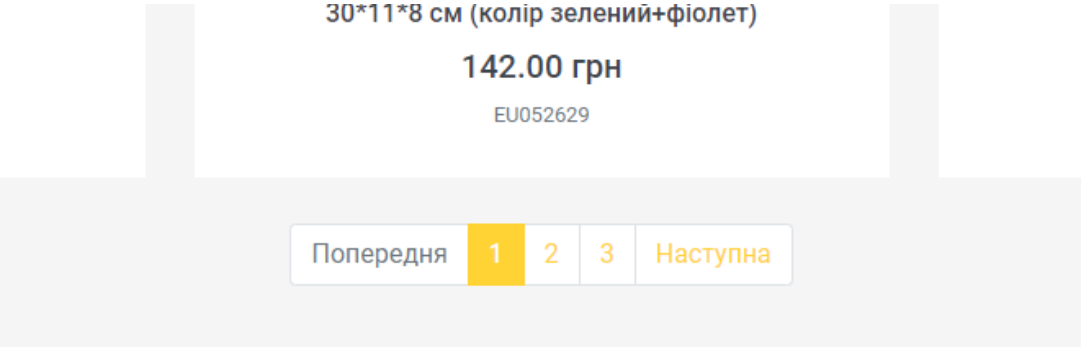


Рис. 3.4 Інтерфейс пагінації

Пагінація - це техніка розподілу великого обсягу даних на окремі сторінки. Вона застосовується для поділу списку товарів, статей, користувачів або будь-яких інших даних на більш зручні порції. Замість відображення всіх записів на одній сторінці, пагінація дозволяє користувачам переходити між різними сторінками, щоб проглядати різні частини даних [17].

Користувач може вибрати номер сторінки (рис. 3.4) або використовувати кнопки "Попередня" та "Наступна" для переходу між сторінками. Кількість записів, які відображаються на кожній сторінці, може бути налаштована і зазвичай встановлюється з урахуванням зручності користувача та швидкодії сторінки.

Пагінація забезпечує кращу навігацію і полегшує користувачам знаходження потрібних даних. Вона також сприяє поліпшенню швидкодії сторінок, оскільки завантаження всіх даних одночасно може призвести до затримок і перевантаження сторінки.

Для реалізації пагінації використовуються різні техніки та компоненти, зокрема вбудовані функції веб-фреймворків або сторонні бібліотеки. При належному використанні пагінація допомагає створити зручний та ефективний досвід користувача при роботі з великими обсягами даних.

3.2 Фільтрація по категоріям і виробникам

Фільтрація товарів в інтернет-магазині дозволяє користувачам вибрати товари за певними критеріями, полегшуючи процес пошуку і забезпечуючи зручність покупок. Елементи фільтрації включають категорії товарів, ціновий діапазон, бренди або виробників, характеристики товарів та рейтинг або відгуки. Користувачі можуть вибрати певні категорії, встановити межі цінового діапазону, обрати певні бренди або виробників, вибрати характеристики товарів та врахувати рейтинг або відгуки. Ці фільтри допомагають скоротити кількість відображуваних товарів, показуючи тільки ті, що відповідають обраним критеріям. Фільтри представлені у вигляді чекбоксів, радіокнопок, випадаючих списків або інших елементів інтерфейсу, що дозволяють користувачеві зручно вибирати критерії фільтрації.

В проекті реалізована фільтрація по категоріям (рис. 3.5) і по виробникам (рис. 3.6).

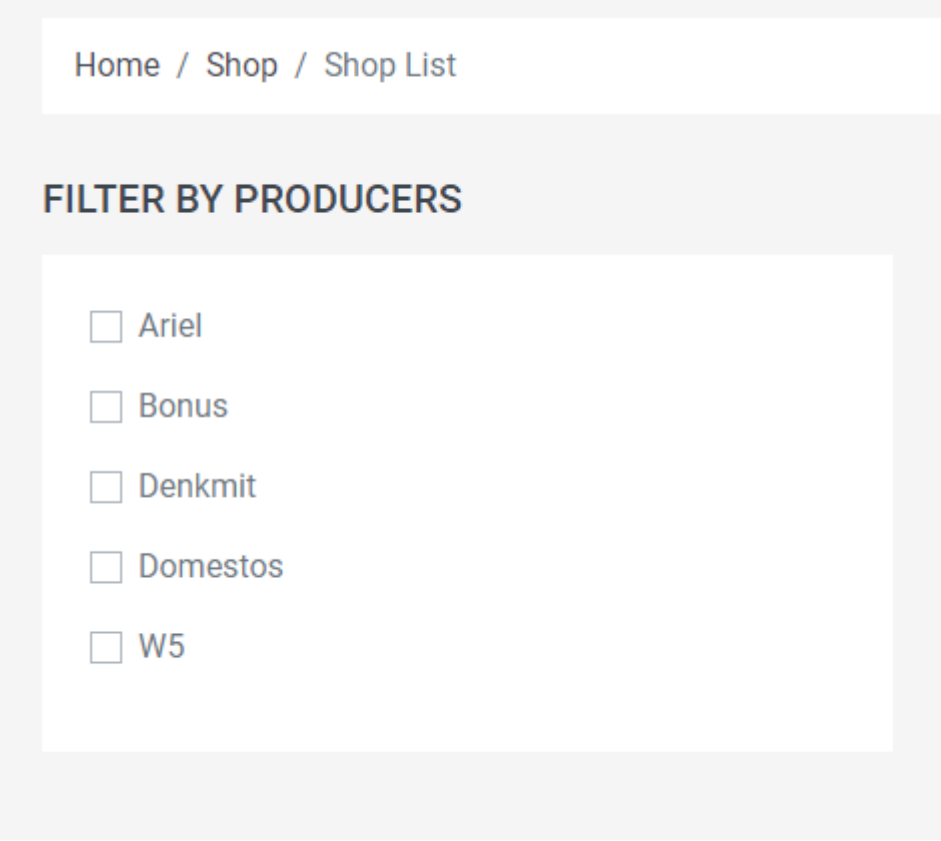


Рис. 3.5 Список виробників

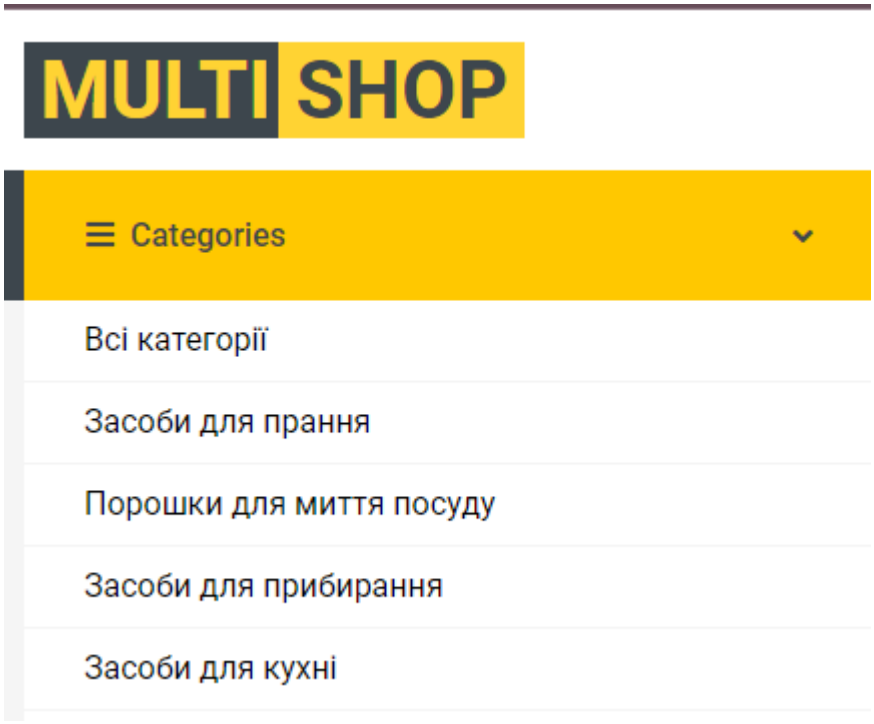


Рис. 3.6 Список категорій

3.3 Корзина

Корзина є важливою частиною багатьох інтернет-магазинів і дозволяє користувачам збирати і зберігати товари перед оформленням замовлення. Основна функціональність корзини включає додавання товарів, вилучення товарів, оновлення кількості товарів і розрахунок загальної суми замовлення. Корзина дозволяє користувачам зручно керувати своїми покупками, переглядати список доданих товарів, вносити зміни і продовжувати процес покупок. Вона забезпечує зручний зв'язок між користувачем і системою замовлення, спрощуючи процес вибору і оплати товарів.

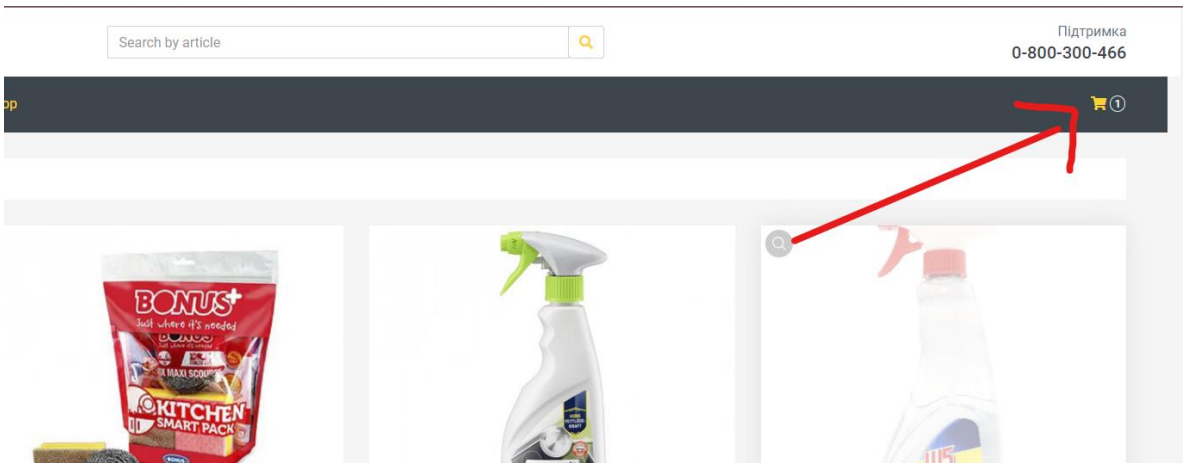


Рис. 3.7 Іконка корзини на головній (домашній) сторінці

Після того, як ми натиснули на іконку корзини на товарі, товар потрапляє в корзину. Справа зверху (рис. 3.7) ми бачимо іконку корзини і кількість товарів в ній. Ми можемо відкрити корзину натиснувши на іконку.

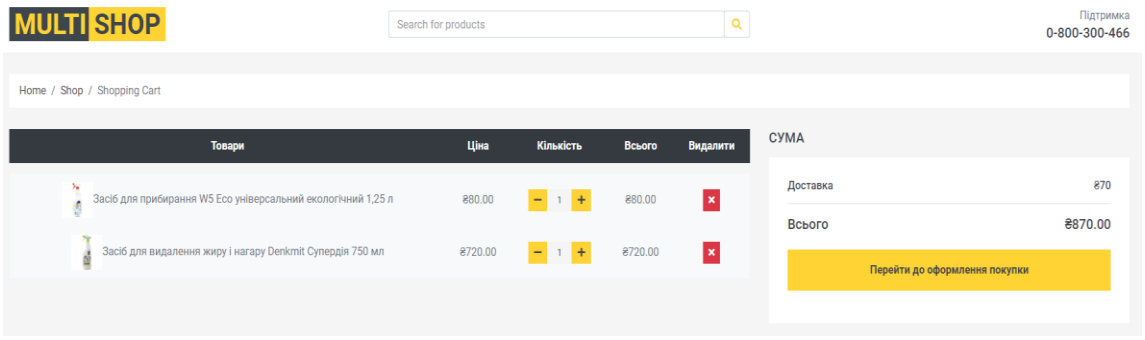


Рис. 3.8 Сторінка корзини

В корзині реалізовані наступні речі:

1. Список товарів: Корзина відображає список товарів, які були додані користувачем. Кожен товар має назву, зображення, ціну та кількість одиниць.
2. Кількість товарів: Користувач має можливість змінювати кількість одиниць товару в корзині. Це реалізовано за допомогою кнопок збільшення або зменшення кількості, або введенням числа в поле.
3. Вилучення товарів: Користувач має можливість видалити товар з корзини, якщо він вирішив не купувати його.
4. Загальна сума замовлення: Корзина відображає загальну суму замовлення, яка **автоматично перераховується** при зміні кількості товарів або додаванні/вилученні товарів.
5. Кнопка оформлення замовлення: Користувач має можливість перейти до процесу оформлення замовлення, коли він готовий придбати товари з корзини. Це може бути реалізовано за допомогою кнопки "Оформити замовлення".

3.4 Строка пошуку товарів

Строка пошуку товарів - це елемент інтерфейсу, який дозволяє користувачеві ввести ключові слова або фрази для пошуку конкретних товарів. Це поле вводу зазвичай знаходиться на сторінці зі списком товарів або в рядку навігації. Користувач може ввести назву товару, його опис, артикул або будь-яку іншу інформацію, що допоможе знайти потрібний товар. Після введення пошукового запиту користувач натискає кнопку пошуку або використовує клавішу Enter, і система виконує пошук за заданими критеріями. Результати пошуку можуть відображатися у вигляді списку товарів, які відповідають пошуковому запиту.

Строка пошуку товарів дозволяє забезпечити швидкий та зручний доступ до потрібної інформації та полегшує процес знаходження конкретного товару в інтернет-магазині.

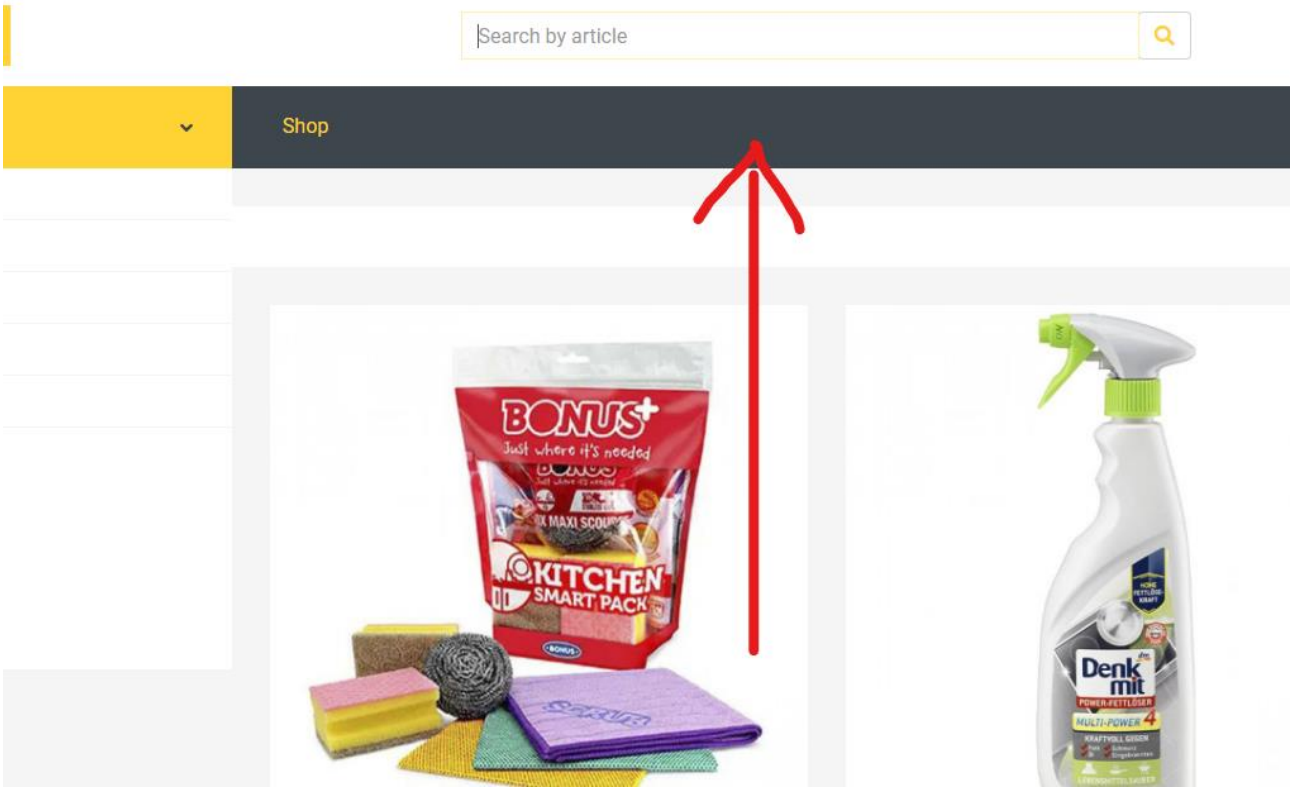


Рис. 3.9 Строка пошуку

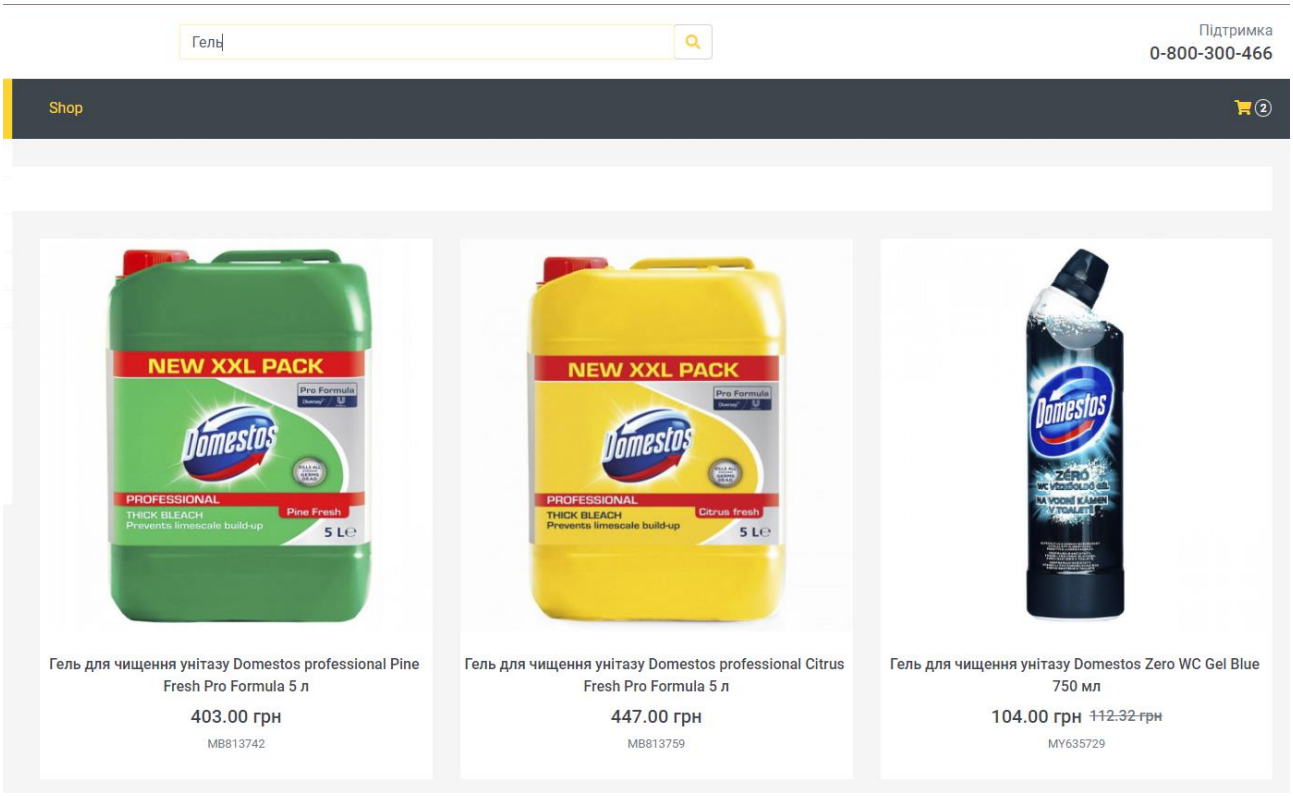


Рис. 3.10 Результат пошуку по запросу “гель”

3.5 Інформація про товар

Сторінка інформації про товар містить зображення товару, назву, опис, ціну, кнопку "Додати в корзину", відгуки та оцінки покупців, додаткові характеристики та артикль. Вона дозволяє користувачеві отримати повну інформацію про товар і прийняти рішення щодо його придбання.

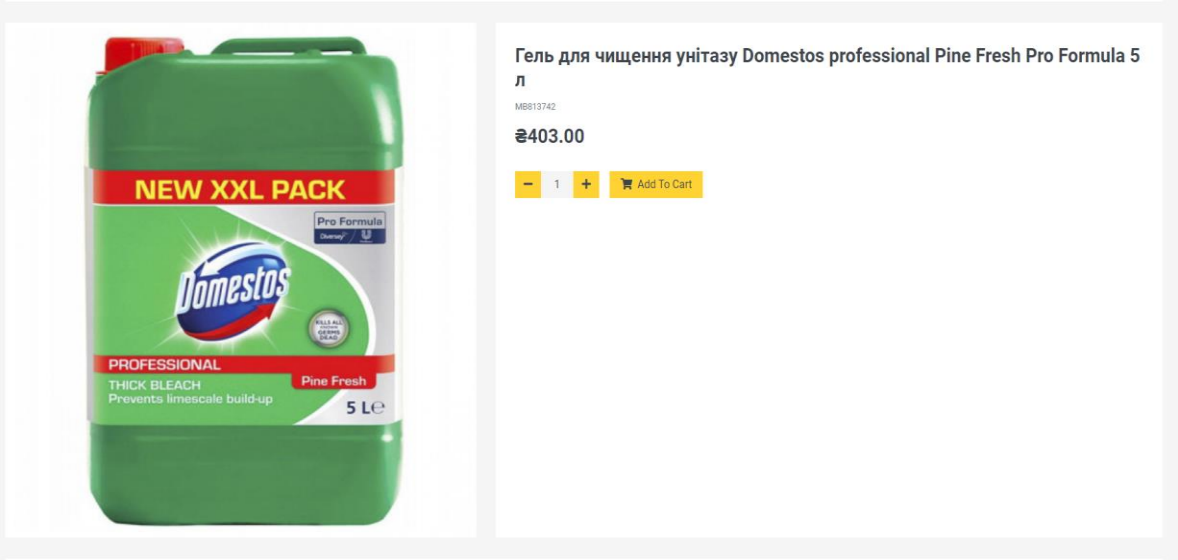


Рис. 3.11 Верхня частина сторінки інформації про товар

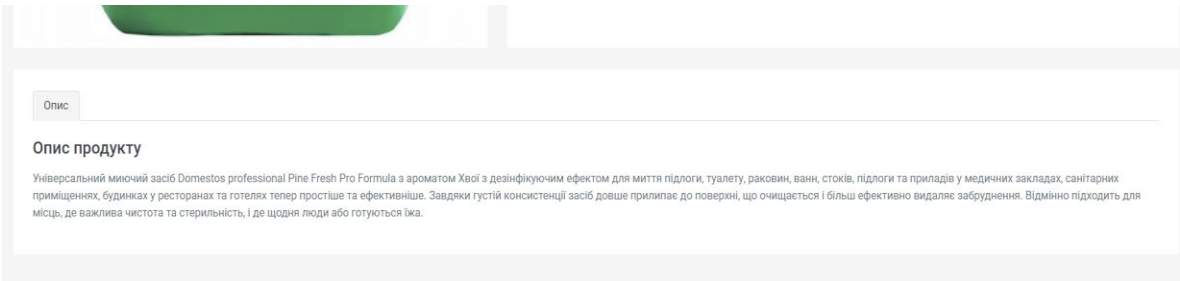


Рис. 3.12 Опис товару

3.6 Авторизація

Авторизація - процес перевірки і підтвердження ідентичності користувача для забезпечення доступу до обмежених ресурсів або функціональності системи. Вона забезпечує контроль над доступом і дозволяє системі відрізняти між авторизованими і неавторизованими користувачами. Авторизація може вимагати введення коректних облікових даних, таких як ім'я користувача і пароль, або використання інших методів, таких як токени або сертифікати. Успішна авторизація дає користувачеві можливість виконувати певні дії або отримувати доступ до конфіденційної інформації.

В проекті авторизація реалізована за допомогою ASP.NET Identity, який є вбудованою бібліотекою для управління користувачами, ролями і авторизацією в ASP.NET додатках. ASP.NET Identity надає готові засоби для реєстрації користувачів, аутентифікації, управління ролями та дозволами.

Під час реєстрації користувачів вводять свої облікові дані, такі як ім'я, електронна пошта та пароль. Після успішної реєстрації їм присвоюється унікальний ідентифікатор користувача.

Для аутентифікації користувачів використовується механізм токенів. Користувачі вводять свої облікові дані, і їх ідентичність перевіряється збереженими в базі даних. Після успішної аутентифікації користувачам надається доступ до обмежених ресурсів або функціональності.

ASP.NET Identity також надає можливість управляти ролями та дозволами користувачів. Адміністратор системи може призначати ролі різним користувачам і надавати їм відповідні дозволи в залежності від їхніх прав доступу.

Усі дані про користувачів, ролі та дозволи зберігаються в базі даних, що дозволяє зберігати та керувати інформацією про користувачів з безпекою і масштабованістю.

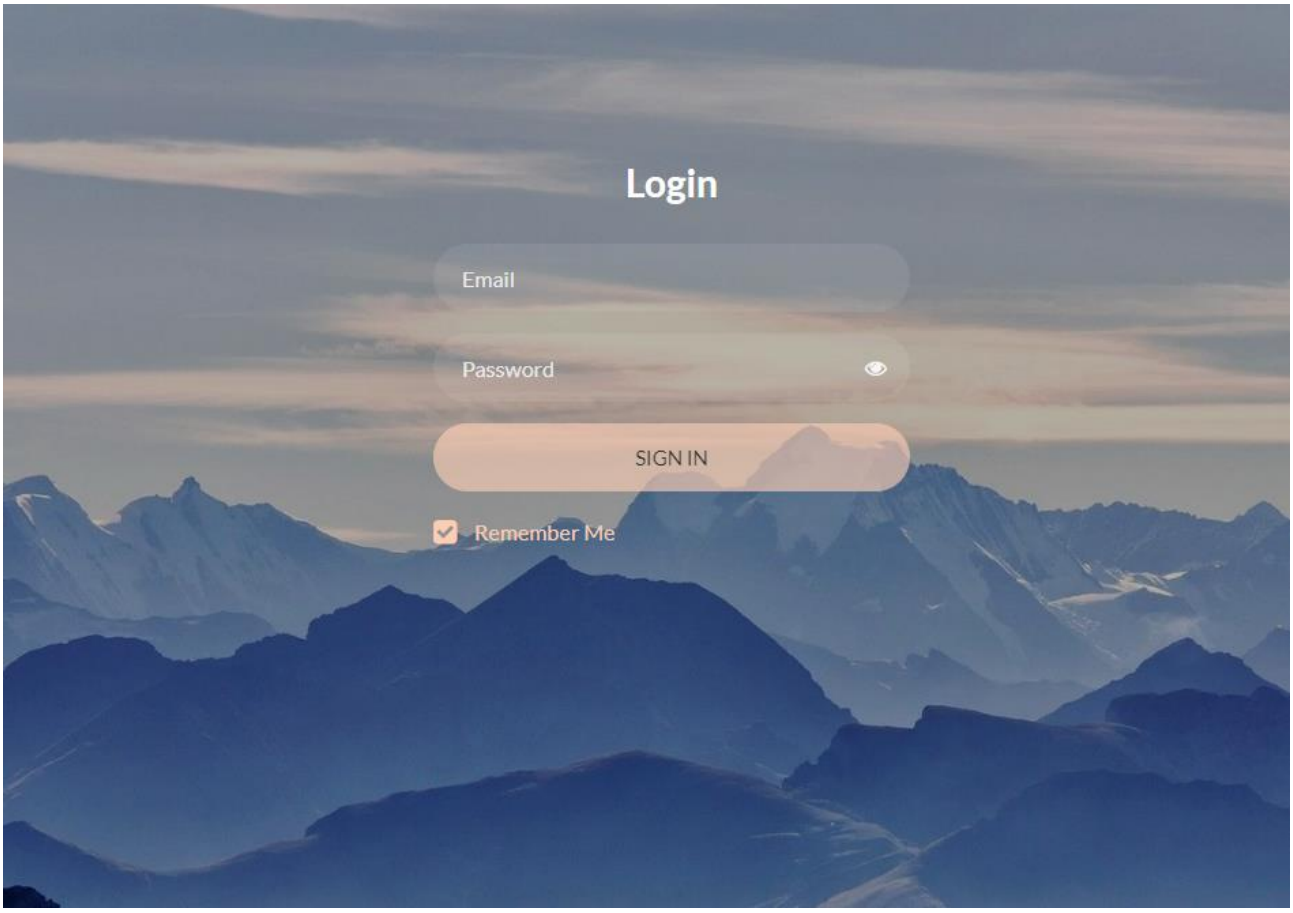


Рис. 3.13 Сторінка логіна

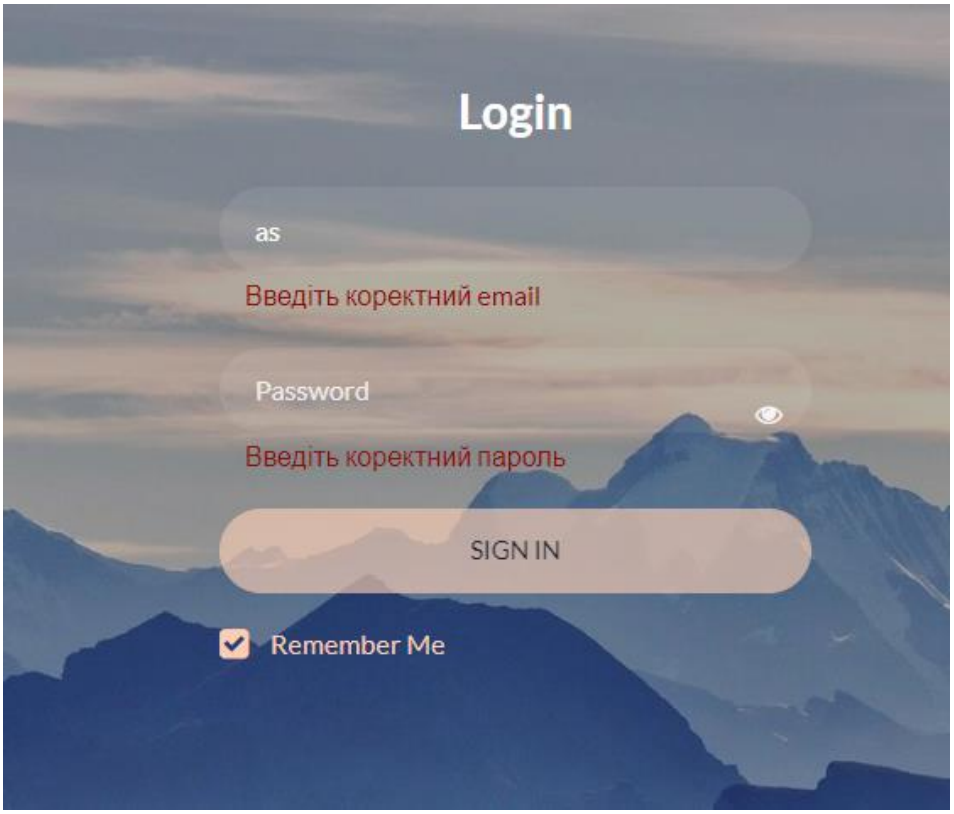


Рис. 3.14 Валідація данних

ВИСНОВКИ

У ході розробки веб-додатку були задіяні різні аспекти пов'язані з розробкою веб-додатків, базами даних, фреймворками та інструментами програмування. Були розглянуті такі теми як технічне завдання, веб технології, мови програмування (наприклад, **TypeScript**), бази даних (**SQL**), фреймворки (**Angular**, **ASP.NET Core**), а також різні концепції інтернету і розробки програмного забезпечення.

Протягом розробки веб додатку, ми спробували надати короткі описи та пояснення, щоб допомогти зрозуміти основні поняття і концепції в області програмування та розробки веб-додатків. Для кожної теми, ми надавали основні відомості та підтримували це посиланнями на джерела, які можуть бути використані для детальнішого вивчення.

В цілому, розробка веб-додатків є комплексним процесом, який включає в себе використання різних інструментів, технологій та фреймворків. Розуміння основних понять та концепцій у цій галузі може допомогти стати більш успішним розробником і зробити впевнені кроки у своїй кар'єрі.

Даний додаток є веб-додатком для продажу побутової хімії. Він призначений для створення онлайн-магазину, де користувачі можуть переглядати, вибирати та придбавати різноманітні товари, пов'язані з побутовою хімією, такі як мийні засоби, засоби для прибирання, косметичні засоби і т.д.

Цей додаток дозволяє зручно та ефективно продавати побутову хімію, забезпечуючи зручний досвід для клієнтів і спрощуючи процес управління магазином для власників. Він може бути використаний різними компаніями, які спеціалізуються на продажу побутової хімії, а також сприяє автоматизації та оптимізації їхньої діяльності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Contributors to Wikimedia projects. Internet - Wikipedia. Wikipedia, the free encyclopedia. URL: **https://en.wikipedia.org/wiki/Internet** (дата звертання: 17.06.2023).
2. Learn web development | MDN. MDN Web Docs. URL: **https://developer.mozilla.org/en-US/docs/Learn** (дата звертання: 17.06.2023).
3. Angular. Angular. URL: **https://angular.io/** (дата звертання: 17.06.2023).
4. Angular. Angular. URL: **https://angular.io/guide/architecture** (дата звертання: 17.06.2023).
5. Angular Components, Modules and Services. Mehtalogy. URL: **https://www.mehtalogy.in/2020/11/angular-components-modules-and-services.html** (дата звертання: 17.06.2023).
6. RxJS. RxJS. URL**: https://rxjs.dev/** (дата звертання: 17.06.2023).
7. RxJS in Action. Manning Publications. URL: **https://www.manning.com/books/rxjs-in-action** (дата звертання: 17.06.2023).
8. The starting point for learning TypeScript. TypeScript: JavaScript With Syntax For Types. URL: **https://www.typescriptlang.org/docs/** (дата звертання: 17.06.2023).
9. CSS: Cascading Style Sheets | MDN. MDN Web Docs. URL: **https://developer.mozilla.org/en-US/docs/Web/CSS** (дата звернення: 17.06.2023).
10. HTML: HyperText Markup Language | MDN. MDN Web Docs. URL: **https://developer.mozilla.org/en-US/docs/Web/HTML** (дата звертання: 17.06.2023).
11. npm Docs. npm Docs. URL: **https://docs.npmjs.com/** (дата звертання: 17.06.2023).
12. ASP.NET documentation. Microsoft Learn: Build skills that open doors in your career. URL: **https://docs.microsoft.com/en-us/aspnet/core/** (дата звертання: 17.06.2023).
13. SQL Tutorial. W3Schools Online Web Tutorials. URL: **https://www.w3schools.com/sql/** (дата звертання: 17.06.2023).
14. Introduction to the Oracle Database. Moved. URL: **https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.html** (дата звернення: 17.06.2023).
15. Overview of ASP.NET Core Authentication. Microsoft Learn: Build skills that open doors in your career. URL: **https://learn.microsoft.com/en-**

us/aspnet/core/security/authentication/?view=aspnetcore-7.0 (дата звернення: 17.06.2023).

16. Create web APIs with ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. URL: **https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0#api-controllers** (дата звернення: 17.06.2023).
17. Hanna K. T., Wigmore I. What is pagination? – TechTarget Definition. WhatIs.com. URL: **https://www.techtarget.com/whatis/definition/pagination#:~:text=Pagination%20is%20the%20process%20of,the%20sequential%20order%20of%20pages**. (дата звертання: 18.06.2023).