КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-сайт для замовлення товарів в інтернет-магазині»

Виконав _____ _____
(Підпис)
Кущ Дмитро Валерійович
(прізвище, ім'я, по батькові)

Керівник д.е.н., професор Плескач Валентина Леонідівна
(прізвище, ім'я, по батькові)

_____ _____
(Резолюція «До захисту»)

Попередній захист:

_____***23.05.2022***_____
_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____ ___ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2022

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	Виконано
3.	Настановча групова співбесіда з питань	21.10.2021	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	Виконано
6.	Підготовка і подання науковому керівнику	21.12.2022	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2022	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту	30.03.2022	Виконано
9.	Подання роботи у першому варіанті	28.04.2022	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2022**	Виконано
12.	Врахування зауважень керівника і подання роботи	27.05.2022	Виконано
13.	Затвердження роботи в цілому	10.06.2022	Виконано
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022	Виконано

Здобувач вищої освіти _____ _____
(підпис)

Керівник _____ _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	14
2	20
3	7
Висновки	2
Перелік використаних джерел	3
Додатки	12

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="2">ДП ХХХХ 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td></tr>
<tr><td>Розробн.</td><td>Кущ Д. В.</td><td></td><td></td><td rowspan="4">Відомість дипломної роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Плескач В.Л.</td><td></td><td></td><td>1</td><td>67</td></tr>
<tr><td>Н/контр.</td><td>Базилюк А.М.</td><td></td><td></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td></td><td></td></tr>
</table>

АНОТАЦІЯ

Дипломна робота: 67 с., 15 рис., 3 табл., 25 джерел, 2 дод.

Ця дипломна робота присвячена проектуванню та розробленню веб-сайту для замовлення товарів в інтернет-магазині.

Метою дипломної роботи є підвищення ефективності роботи е-магазину продуктів завдяки розробленому веб-застосунку.

Для досягнення поставленої мети треба вирішити такі **завдання**:

- дослідити теоретичні основи побудови веб-сайтів для замовлення товарів в інтернет-магазинах;
- здійснити проектування веб-сайту для замовлення товарів з відповідною архітектурою та базою даних MS SQL, реалізацією авторизації та аутентифікації користувача в системі;
- здійснити тестування роботи веб-сайту та його впровадження.

Об’єкт дослідження.

Процеси електронної торгівлі товарами.

Предмет дослідження.

Програмно-технічні, організаційні засади, принципи, підходи щодо побудови веб-сайту для замовлення товарів в інтернет-магазині.

Методи дослідження.

- порівняння; дослідження різниці між видами архітектури (Monolithic,micro-services);
- експеримент; визначення переваг і недоліків шаблону MVC;
- спостереження; дослідження взаємодії веб-серверу з клієнтом.

Ключові слова: веб-сайт, е-комерція, електронні товари, технологія ASP.NET Core.

ABSTRACT

Thesis: 67 pages, 15 figures, 3 tables, 25 sources, 2 appendices.

This thesis is devoted to design and development web site for ordering goods in the online store.

The purpose of the thesis is to increase the efficiency of the grocery store through a developed web application.

To achieve this goal you need to solve the following **tasks**:

● explore the theoretical foundations of building websites for ordering goods in online stores

● design a website for ordering goods with the appropriate architecture and MS SQL database, implementation of authorization and user authentication in the system;

● test and implement the website.

Object of study.

E-commerce processes.

Subject of study.

Software and technical, organizational principles, principles, approaches to building a website for ordering goods in the online store.

Research methods.

• comparison; study of the difference between types of architecture (Monolithic, micro-services)

• experiment; identifying the advantages and disadvantages of the MVC template

• observation; study of web server interaction with the client

Keywords: software system, e-commerce, electronic goods , ASP.NET Core technology.

ЗМІСТ

ЗМІСТ 6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ 7

ВСТУП 8

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ВЕБ-САЙТІВ ЗАМОВЛЕННЯ ТОВАРІВ 10

1.1 Огляд і аналіз принципів і засобів створення веб-сайтів 10

1.2 Переваги та недоліки технологій побудови веб-сайтів 15

1.3 Інструментальні засоби побудови представлень 17

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-САЙТУ ЗАМОВЛЕННЯ ТОВАРІВ 24

2.1 Вибір інформаційної архітектури 24

2.2 Засоби навігації 34

2.3 Види забезпечення для створення веб-сайтів 39

РОЗДІЛ 3 РЕАЛІЗАЦІЯ СИСТЕМИ ЗАМОВЛЕННЯ ТОВАРІВ 44

3.1 Реалізація роботи веб-сайту 44

3.2 Інструкція користування веб-сайтом 45

3.3 Тестування веб-сайту 48

ВИСНОВОК 51

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53

ДОДАТОК А 56

ДОДАТОК Б 58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IDE – Integrated Development Environment.

SSIS - Integration Services

SPA – Single Page Application

ВСТУП

Електронний магазин – це торгова система, яка функціонує цілодобово та забезпечує ведення всього комплексу операцій, пов’язаних з торгівлею та обліком. Різницею інтернет-магазину в порівнянні зі звичайною формою торгівлі є те що інтерактивний магазин може значно збільшити кількість товарів і послуг, і забезпечити більшу кількість споживачів для ухвалення рішення про покупку. Крім того, в часи комп'ютерних технологій можлива персоналізований підхід до всіх користувачів, залежно від його покупок, які він робив раніше.

Актуальність дослідження.

Ось уже кілька років поспіль статистика не вказує нічого принципово нового: сегмент онлайн-шопінгу неухильно зростає... Середня кількість відвідувань онлайн-магазинів по всьому світу збільшується. Цей метод покупок також дуже зручний для жителів маленьких міст з мізерною кількістю необхідних магазинів. Стало зручно використовувати сайти замовлень товарів в інтернеті, тому сфера розробки веб-сайтів розширюється все більше.

Також, за допомогою web-сайту компанії презентують свій сайт в Інтернеті, що в свою чергу сприяє росту аудиторії і підтримці бренду.

Метою кваліфікаційної роботи бакалавра є підвищення ефективності роботи магазину продуктів шляхом розробки веб-застосунку для взаємодії з клієнтом.

Завдання дослідження:

- Провести аналіз і зробити вибір засобів для розробки сайту Інтернет-магазину.
- Розробити архітектуру веб-сайту та шляхи комунікації за протоколом «HTTP», розробити схему роботи з базою даних «MS SQL» та реалізувати авторизацію та аутентифікацію користувача в системі.

- Продемонструвати роботу веб-сайту.

Об’єктом дослідження є процес електронної торгівлі товарами.

Предметом дослідження є веб-сайт для замовлення товарів в інтернет-магазині**.**

Методами дослідження є:

- порівняння; дослідження різниці між видами архітектури(Monolithic,micro-services)
- експеримент; визначення переваг і недоліків шаблону MVC
- спостереження; дослідження взаємодії веб-серверу з клієнтом

Практичне значення одержаних результатів полягає в тому , що веб-сайт стане зручним, мобільним та сучасним рішенням для бізнесу, який в еру нових технологій має просувати себе в Інтернеті. Програмна система може застосовуватись для торгової діяльності магазину з усіма вимогами ринку, які є на сьогоднішній день.

Структура роботи:

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ВЕБ-САЙТІВ ЗАМОВЛЕННЯ ТОВАРІВ

1.1 Огляд і аналіз принципів і засобів створення веб-сайтів

Веб-сайт – це сервіс, який використовують між собою два електронні пристрої що взаємодіють між собою.

Зв'язок зазвичай здійснюється за допомогою файлів JSON або XML.

Веб-служби роблять функції програмного забезпечення доступними через мережу Інтернет, щоб такі програми, як PHP, ASP, JSP, JavaBeans, COM-об'єкт та всі інші наші улюблені віджети, могли робити запит до програми, яка працює на іншому сервері (веб-служба), і використовувати її відповідь на запит на веб-сайті, у WAP-сервісі чи іншому застосунку.

Тому, коли мені потрібно виконати якесь завдання з програмування, і я занадто зайнятий, щоб спробувати це сам, я можу скористатися веб-службою, називаючи це через Інтернет. Передаючи дані параметрів за допомогою запиту, я можу розраховувати на отримання відповіді, що містить отриманий результат веб-служби.

Кожен, хто нещодавно користувався Hotmail, стикався з веб-службами: система автентифікації паспорта є однією з веб-служб за ініціативи Microsoft .NET і наразі доступна безкоштовно, тому розробники можуть використовувати автентифікацію паспорта на своїх власних сайтах.

Наразі існує значна кількість підходів та методів розробки веб-сайтів

Розглянемо основні:

REST розшифровується як REpresentational State Transfer. -

архітектурний стиль для проектування мережевих застосунків. Може використовуватися будь-якою мовою програмування. Особливістю яка відрізняє його є те, що REST сервіси дозволяють використовувати протокол HTTP найкращим чином.

SOAP - протокол обміну структурованими повідомленнями в розподіленому обчислювальному середовищі. Підтримується консорціумом W3C. Цей протокол був створений в 1998 році командою під керівництвом Дейва Вінера, яка працювала в корпорації Microsoft і фірмі Userland, але згодом проект був переданий в консорціум W3C.

SOAP протокол не може відокремити відповідь і виклик процедури, він просто визначає формат посилання зображаючи його у вигляді XML документу.

Після проведення аналізу принципів побудови та засобів створення веб сайтів, було встановлено такі висновки:

-- При створенні веб-сайту використовувати архітектурний стиль REST

-- Основною платформою для розробки сайту обрати ASP.NET.

-- Дані передавати за допомогою протоколу **HTTP**.

-- Основним сервером обрати **Kestrel**

Створений веб-сайт буде надавати свій функціонал для будь-якої платформи або веб-сайту в залежності від вимог користувача. Застосунок є кросплатформним тобто може використовуватися на різних платформах , такі як:Windows , Linux, MacOS та інші.

Основне завдання дипломної роботи є створення веб-сайту, який дозволятиме робити замовлення, проводити авторизацію та аутентифікацію користувача, дозволить користувачу зберігати сесію активною навіть при роз’єднання зв’язку з сайтом, а зберігання даних буде проводитись за

допомогою реляційною бази даних **MS SQL**.

Для виконання поставленої мети необхідно вирішити такі задачі:

- проаналізувати вже існуючі програмні рішення інтернет-магазинів надання послуг.
- Знайти окремі особливості на різних інтернет-ресурсах та інтегрувати їх в сайті;
- створити простий інтерфейс сайту , щоб продемонструвати роботу сайту;
- виконати тестування функціоналу сайту, у разі негативного виконання тестів виправити помилки;
- перевірити працездатність програми в декількох операційних системах, оскільки застосунок повинен бути кросплатформним.

Ресурси

Ресурси є основними елементами веб-платформи. Під час роботи

на REST перше завдання полягає в тому, щоб визначити ресурси та з'ясувати, як вони пов'язані один з одним. Кожен ресурс має унікальний ідентифікатор на веб-платформі, який відомий як універсальний ідентифікатор ресурсу (URI), а найкращим прикладом в Інтернеті є єдиний локатор ресурсів (URL). Кількість URI, які можуть посилатися на ресурс, не обмежена. Наприклад, ми можемо отримати доступ до певної сторінки домену (звичайно, ресурсу), використовуючи http://yahoo.com і http://www.yahoo.com.

У веб-сервісах REST ми використовуємо іменники для визначення типу ресурсу. Інформацію про співробітників з EmpDB можна отримати за наведеною нижче URL-адресою: : //EmployeeService/Employee/1

Методи

Метод – це дія HTTP, як POST, GET PUT, DELETE, OPTIONS тощо.

Давайте спочатку переглянемо запит HTTP. Приклад запиту GET:

GET http://www.w3schools.com/ : HTTP/1.1

Status: HTTP/1.1 200 OK

Accept text/xml,text/html;

Accept-Encoding gzip, deflate, sdch

Accept-Language en-US,en;

Використовуючи URL-адреси, можна визначити ідентичність цільового сервера для зв'язку, але методи HTTP лише вказують вам, яку дію потрібно виконати на хості. Існує багато дій, які клієнт може запустити на хості.

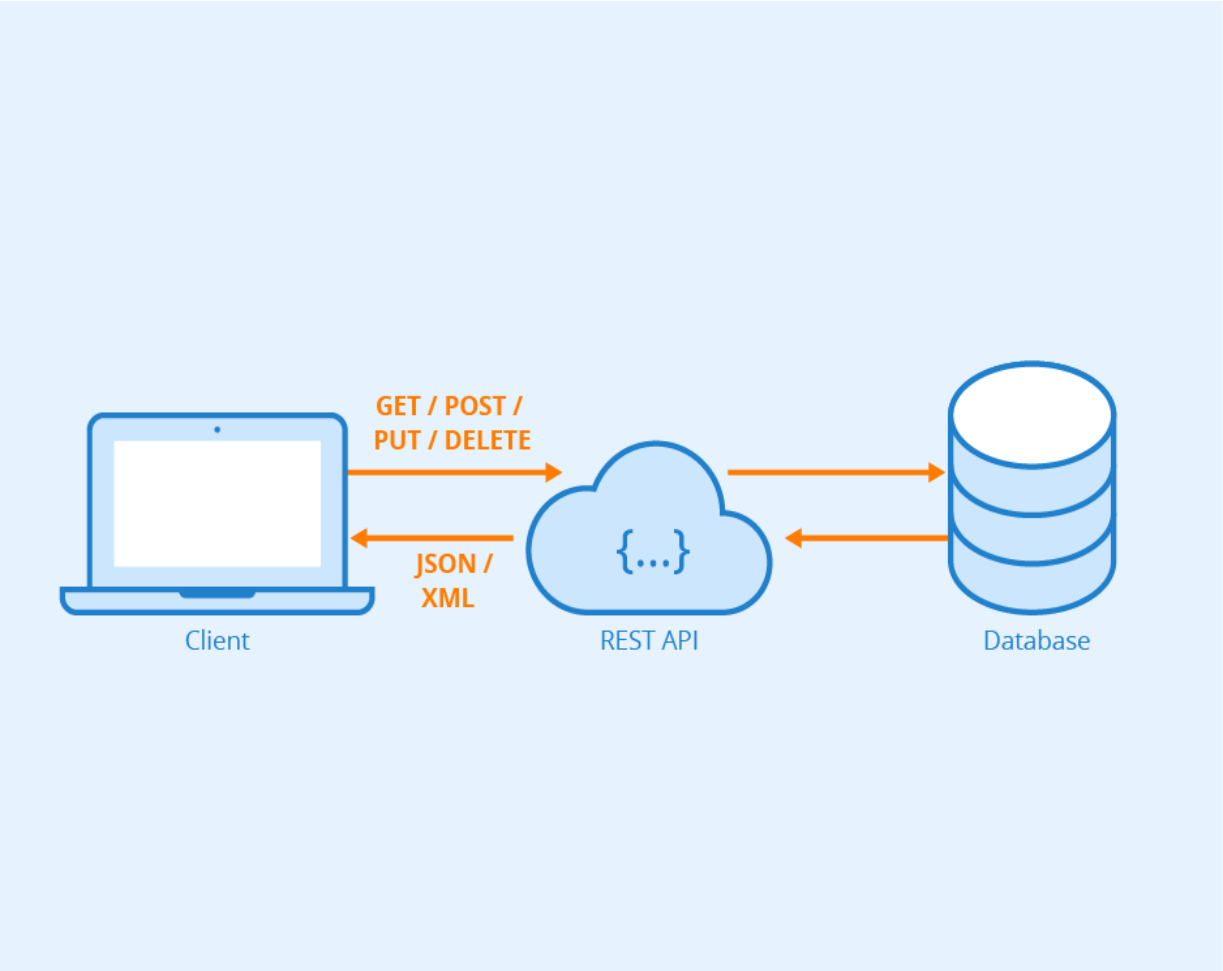


Рисунок 1.1 - Схема роботи клієнт-серверної архітектури за принципами REST

Веб-сайт - це одиниця керованого коду, що викликається віддалено з допомогою протоколу HTTPS, який зазвичай маємо активувати HTTP-викликами . Веб-сайти дозволяють виявляти функціональність існуючого коду через мережу. Коли інша програма виявить веб-сайт вона може використовувати функціональність вашої програми.

1.2 Переваги та недоліки технологій побудови веб-сайтів

Representational State Transfer (REST) — це архітектурний стиль для проектування слабо пов’язаних веб-сервісів. В основному він використовується для розробки легких, швидких, масштабованих і простих у обслуговуванні веб-сервісів, які часто використовують HTTP як засіб зв’язку.

Багато в чому сама всесвітня мережа, яка базується на HTTP, є найкращим прикладом архітектури на основі REST.

Програми RESTful використовують HTTP-запити для публікації даних (створення/оновлення), читання даних (виготовлення запитів) і видалення даних.

REST використовує HTTP для всіх чотирьох операцій CRUD (Create / Read / Update / Delete).

REST визначає Інтернет як розподілений гіпермедіа (гіперпосилання всередині гіпертексту), чиї зв’язані ресурси обмінюються уявленнями про стан ресурсу. Архітектурний стиль REST надає керівні принципи для створення розподілених і слабо пов’язаних застосунків.

REST сам по собі не є стандартом, а є архітектурним стилем, який використовує такі стандарти, як HTTP, XML. Ось чому ви ніколи не побачите організації, які продають набори інструментів на основі REST.

Ми повинні розробити веб-сайт REST таким чином, щоб у результаті були слабо пов’язані веб-сервіси, які відповідають веб-стандартам.

Він також повинен бути сприятливим для розвитку і досить гнучкий, щоб використовувати його для різних нових застосувань.

Розглянемо найкращі методи роботи з REST, які можна

використовувати для дизайну веб-сервісів REST.

Протокол доступу до простих об’єктів (SOAP) залежить насамперед від XML для надання послуг обміну повідомленнями. SOAP використовує різні протоколи для зв'язку, такі як HTTP, SMTP або FTP.

REST, з іншого боку, є архітектурним стилем, який використовує існуючі дії та методи HTTP; і не створює жодних нових стандартів. SOAP, з іншого боку, є протоколом.

REST є більш гнучким порівняно з веб-сервісами SOAP. Він має такі переваги перед SOAP:

- SOAP використовує лише XML для повідомлень. REST підтримує різні формати
- Повідомлення REST менші за розміром і споживають меншу пропускну здатність
- REST кращий з точки зору продуктивності з кращою підтримкою кешування
- Для доступу не потрібен інструмент сторонніх розробників
- Крім того, із сервісами на основі REST навчання легше в порівнянні з SOAP
- Існує менше зв'язків між REST клієнтами (браузерами) і серверами; зміни можна легко внести. Однак клієнт SOAP тісно пов’язаний із сервером, і інтеграція порушиться, якщо буде внесена зміна на будь-якому кінці.

REST слід вибирати, коли вам потрібно розробити високобезпечний і

складний API, який підтримує різні протоколи. Хоча SOAP може бути хорошим вибором, REST може бути кращим, коли вам потрібно розробити легкі API з високою продуктивністю та підтримкою операцій CRUD.

1.3 Інструментальні засоби побудови представлень

Останній крок для взаємодії користувача – це визначення способу продемонструвати ці ресурси клієнтам. REST підтримує всі формати без будь-яких обмежень; тож ви можете використовувати будь-який формат для представлення ресурсів.

Залежно від можливості клієнта та сервера працювати з форматами, ви можете використовувати JSON, XML або будь-який інший формат.

Розробка Front-end частини відбувалась за допомогою двигуна Razor та викостистанням HTML, CSS та бібліотеки стилів Bootstrap.

Razor

Представлений як частина ASP.NET Core, а тепер включений у .NET 5, ASP.NET Razor Pages — це серверна структура, орієнтована на сторінки, яка дозволяє створювати динамічні веб-сайти, керовані даними, з чітким розділенням проблем. Як частина платформи веб-розробки ASP.NET Core від Microsoft, Razor Pages підтримує міжплатформну розробку і може бути розгорнута в операційних системах Windows, Unix і Mac.

Фреймворк Razor Pages є легким і дуже гнучким. Він надає розробнику повний контроль над відтвореним HTML. Razor Pages є рекомендованим фреймворком для міжплатформного генерування HTML на стороні сервера.

Razor Pages використовує популярну мову програмування C# для програмування на стороні сервера та простий у засвоєнні синтаксис шаблонів Razor для вбудовування C# у розмітку HTML для динамічного створення вмісту для браузерів.

Razor Pages підходить для всіх типів розробників від початківців до корпоративного рівня. Він заснований на моделі розробки, орієнтованої на сторінку, пропонуючи веб-розробникам знайомство з іншими сторінковими фреймворками, такими як PHP, Classic ASP, Java Server Pages, ASP.NET Web Pages і ASP.NET Web Forms. Це також відносно легко для початківців, і воно включає в себе всі розширені функції ASP.NET Core (наприклад, ін’єкції залежностей), що робить його так само придатним для великих, масштабованих, командних проектів.

Якщо вам потрібен динамічний веб-сайт, на який регулярно додається вміст, у вас є кілька варіантів. Ви можете використовувати систему керування вмістом (CMS), серед яких є багато з чого вибрати, включаючи WordPress, Umbraco, Joomla!, Drupal, Orchard CMS тощо.

HTML

HTML означає HyperText Markup Language. Він використовується для оформлення веб-сторінок за допомогою мови розмітки. HTML - це комбінація гіпертексту та мови розмітки. Гіпертекст визначає зв'язок між веб-сторінками. Мова розмітки використовується для визначення текстового документа в тегу, який визначає структуру веб-сторінок. Ця мова використовується для анотування (зроблення нотаток для комп’ютера) тексту, щоб машина могла

його зрозуміти та відповідно маніпулювати текстом. Більшість мов розмітки (наприклад, HTML) читаються людиною. Мова використовує теги, щоб визначити, які маніпуляції потрібно виконати з текстом.

HTML – це мова розмітки, що використовується браузером для маніпулювання текстом, зображеннями та іншим вмістом, щоб відобразити його в необхідному форматі. HTML був створений Тімом Бернерсом-Лі в 1991 році. Першою версією HTML був HTML 1.0, але першою стандартною версією був HTML 2.0, опублікований у 1995 році.

Елементи та теги:

HTML використовує попередньо визначені теги та елементи, які повідомляють браузеру, як правильно відображати вміст. Не забудьте включити закриваючі теги. Якщо опущено, браузер застосовує ефект відкриваючого тегу до кінця сторінки.

Структура сторінки HTML: базова структура сторінки HTML наведена на рисунку 1.2. Вона містить основні будівельні елементи (тобто декларацію типу документа, HTML, заголовок, заголовок і елементи основного тексту), на основі яких створюються всі веб-сторінки.

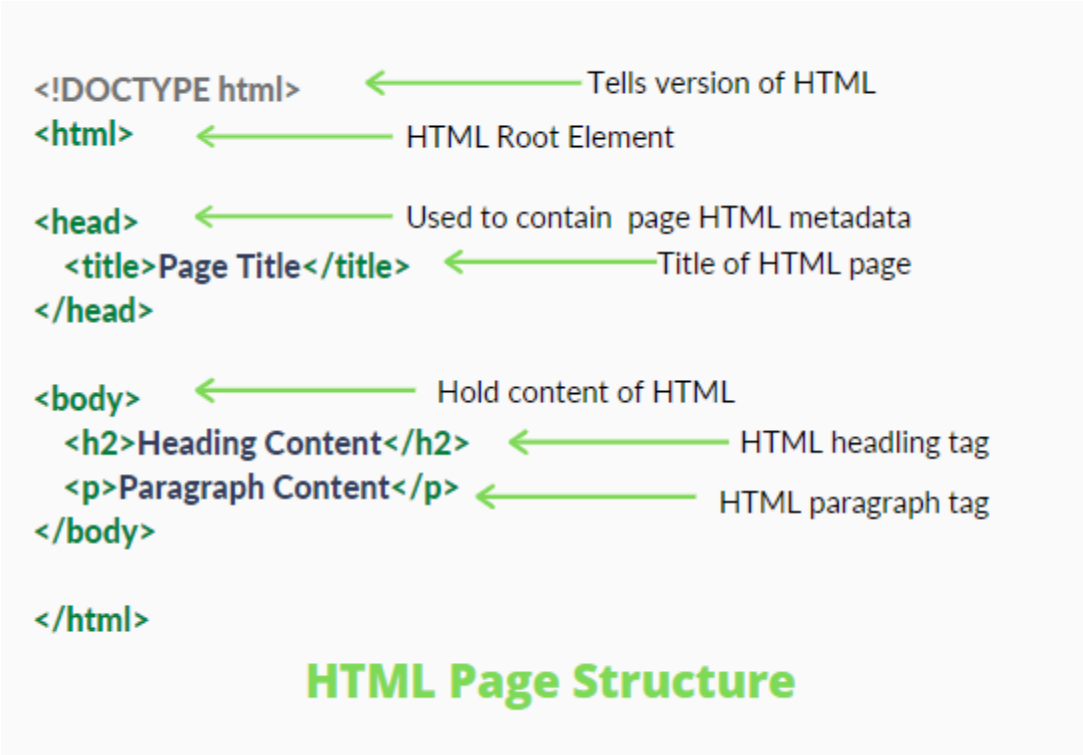


Рисунок 1.2 – Структура html сторінки

<!DOCTYPE html>: це оголошення типу документа (технічно не тег). Він оголошує документ як документ HTML. Оголошення типу документа не чутливе до регістру

<html>: це називається кореневим елементом HTML. Всі інші елементи містяться в ньому.

<head>: тег head містить елементи для веб-сторінки. Елементи в заголовку не видно на передньому плані веб-сторінки. Елементи HTML, що використовуються всередині елемента <head>, включають: стилі, мета-теги та скрипти.

<body>: тег body використовується для охоплення всього видимого вмісту веб-сторінки. Іншими словами, основний вміст – це те, що браузер відображатиме на інтерфейсі.

HTML-документ можна створити за допомогою будь-якого текстового редактора. Збережіть текстовий файл за допомогою .html або .htm. Після збереження у вигляді HTML-документа файл можна відкрити як веб-сторінку в браузері.

CSS

Каскадні таблиці стилів, які називають CSS, — це просто розроблена мова, призначена для спрощення процесу створення презентаційних веб-сторінок. CSS дозволяє застосовувати стилі до веб-сторінок. Що ще важливіше, CSS дозволяє робити це незалежно від HTML, який складає кожну веб-сторінку. Він описує, як має виглядати веб-сторінка: прописує кольори, шрифти, інтервали та багато іншого. Коротше кажучи, ви можете зробити свій веб-сайт виглядати так, як ви хочете. CSS дозволяє розробникам і дизайнерам визначати, як він веде себе, включаючи розташування елементів у браузері.

В той час як html використовує теги, css використовує набори правил.

CSS легко вивчити та зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа.

Переваги CSS:

- CSS економить час: ви можете написати CSS один раз і повторно використовувати той самий аркуш на кількох сторінках HTML.
- Легке обслуговування: щоб внести глобальні зміни, просто змініть стиль, і всі елементи на всіх веб-сторінках оновлюватимуться автоматично.
- Пошукові системи: CSS вважається технікою чистого кодування, що означає, що пошуковим системам не доведеться боротися з «читанням» його вмісту.

- Кращі стилі перед HTML: CSS має набагато ширший набір атрибутів, ніж HTML, тому ви можете надати набагато кращий вигляд своїй HTML-сторінці в порівнянні з атрибутами HTML.
- Офлайн-перегляд: CSS може зберігати веб-програми локально за допомогою автономного кешу. За допомогою цього ми можемо переглядати офлайн-сайти.

Синтаксис CSS:

CSS має правила оформлення, які інтерпретуються, а згодом застосовуються до елементів у вашому документі.

Набір правил стилю складається з селектора та блоку оголошення.

Selector -- h1

Declaration -- {color:blue;font size:12px;}

Селектор вказує на елемент HTML, який потрібно створити.

Блок декларації містить одне або кілька декларацій, розділених крапкою з комою.

Кожна декларація містить назву властивості CSS і значення, розділені двокрапкою.

Наприклад:

–; колір - це властивість, а синій – значення.

–; font-size — це властивість, а 12px — значення.

Оголошення CSS завжди закінчується крапкою з комою, а блоки оголошення оточені фігурними дужками

Bootstrap

Bootstrap — це інтуїтивно зрозумілий і потужний інтерфейсний фреймворк для прискорення роботи і легшої веб-розробки. Він використовує HTML, CSS і Javascript.

Його випустили як продукт з відкритим кодом у серпні 2011 року на GitHub.

Переваги Bootstrap:

- Фреймворк Bootstrap 3 складається зі стилів Mobile first по всій бібліотеці, а не в окремих файлах.
- Підтримка браузера: підтримується всіма популярними браузерами.Легко розпочати: володіючи лише знаннями HTML і CSS, кожен може отримати почав з Bootstrap. Також на офіційному сайті Bootstrap є хороша документація.
- Адаптивний дизайн: адаптивний CSS Bootstrap налаштовується на настільні комп’ютери та планшети і мобільні телефони.
- Адаптивний дизайн.
- Забезпечує чисте та однорідне рішення для створення інтерфейсу для розробників.
- Він містить красиві та функціональні вбудовані компоненти, які легко використовувати, налаштувати.

.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ВЕБ-САЙТУ ЗАМОВЛЕННЯ ТОВАРІВ

2.1 Вибір інформаційної архітектури

Для розробки веб-сайту було обрано мову програмування **C#** та фреймворк ASP.NET Core в якому буде побудова за шаблоном MVC.

Дана технологія була обрана оскільки потрібно забезпечити кросплатформність та стійкість до високих навантажень.

ASP.NET Core — це нова версія веб-фреймворку ASP.NET, яка в основному призначена для роботи на платформі .NET Core.

ASP.NET Core — це безкоштовна кросплатформний застосунок з відкритим вихідним кодом для створення хмарних додатків, таких як веб-програми, програми Інтернету речей і мобільні серверні програми. Він призначений для роботи як у хмарі, так і локально.

Для збереження даних було обрано MS SQL, за його простоту та тісну інтеграцію з ASP.NET.

MSSQL - це набір програмного забезпечення для баз даних, опубліковане корпорацією Майкрософт яке широко використовується серед розробників . Як правило, він включає алгоритм за допомогою якого дані зберігаються в таблицях, рядках і стовпцях, SSIS, що містить інструмент керування даними для експорту та імпорту, служби звітування які використовуються для створення, звітування та обслуговування звітів для кінцевих користувачів.

Microsoft SQL Server (MSSQL) також займає важливу частину в корпоративних проектах. Обрана база даних це велика масштабована система яка керує даними, що включає кілька інструментів ETL (Витяг, перетворення та завантаження) та служби звітності, де дані можна додавати, модифікувати та давати запити з допомогою мови структурних запитів. MSSQL - це платформа даних, що розвивається, що використовується для критично важливих бізнес-рішень та рішень даних на приміщеннях, у хмарі та на гібридних платформах.

Для авторизації та аутентифікації було обрано ASP.NET.Identity

ASP.NET Identity представляє вбудовану систему аутентифікації та авторизації системи ASP.NET. Дана система дозволяє користувачам створювати навчальні записи, аутентифікувати, керувати учбовими записами або використовувати для входу на веб-сайт навчальні записи зовнішніх провайдерів, наприклад, Microsoft, Google , Twitter та інші.

IdentityServer розроблений для гнучкості, і частина цього дає вам змогу використовувати базу даних на ваш розсуд, що полегчить вашим користувачам взаємодію з даними(включаючи хешовані дані). Коли ви починаєте з бази даних, ASP.NET Core Identity - це один із варіантів, який ви можете вибрати. Цей швидкий старт показує, як використовувати ідентифікатор ASP.NET Core з IdentityServer.

Цей підхід до використання ASP.NET Core Identity підходить до створення нового проекту для хосту IdentityServer. Цей новий проект замінить попередній проект IdentityServer, який ми створили в попередніх стартах. Причиною цього нового проекту є різниця в ресурсах користувацького інтерфейсу при використанні ASP.NET Core Identity (головним чином навколо різниць у вході та виході). Усі інші проекти в цьому рішенні (для клієнтів та API) залишаються незмінними.

Для забезпечення простоти та зрозумілості інтерфейсу архітектура веб-сайту була розподілена на 4 основні частини, кожна з яких буде відповідати за певний функціонал.

- Навігаційне меню, перегляд продуктів магазину.
- Меню авторизації користувача.
- Особистий кабінет користувача
- Вікно замовлення продуктів

Для зручності використання, було вирішено використовувати таку схему сайту.(рис 2.1)

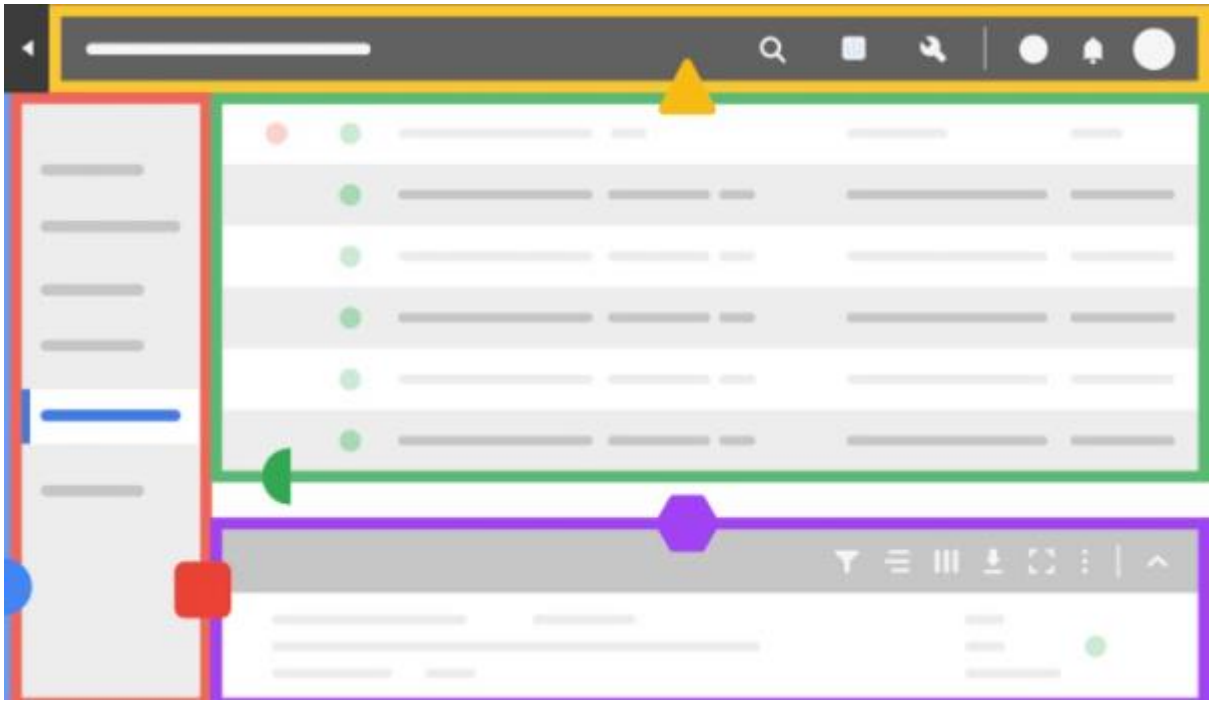


Рисунок 2.1 - Схема побудови компонентів сайту.

Взаємодія клієнта з сервером буде проводитись за допомогою HTTP методів.

Кожним ресурсом в RESTful API управляє кілька певних мінімально необхідних дієслів. В переважній кількості випадків вистачає всього 4 головних запити(HTTP методу)(рис.2.2)

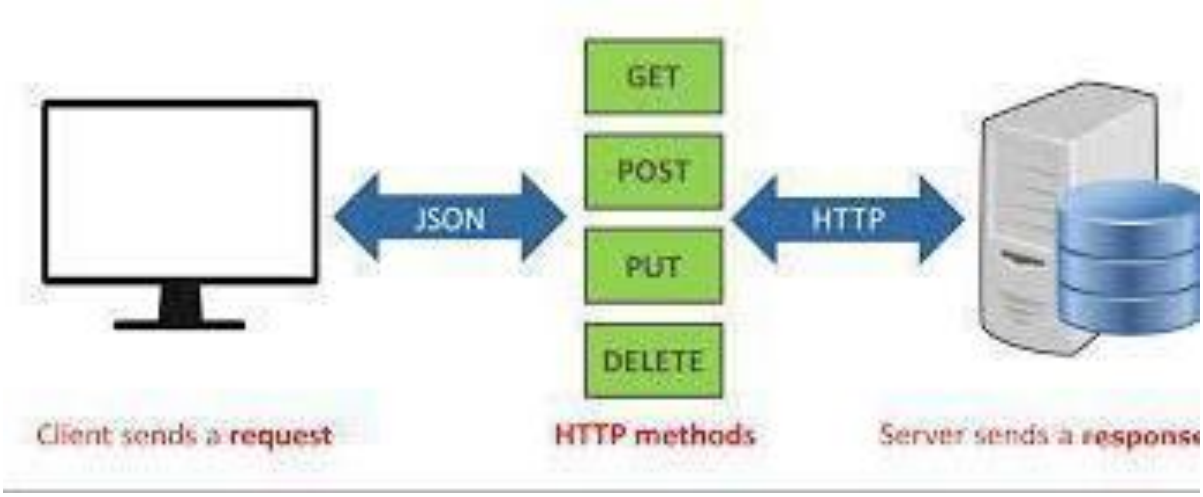


Рисунок 2.2 - Схема взаємодії клієнта(інтерфейс) з сервером

Після прийому сервером запиту йому необхідно створити response на основі даних отриманих з бази даних, схема показана на рисунку 2.3

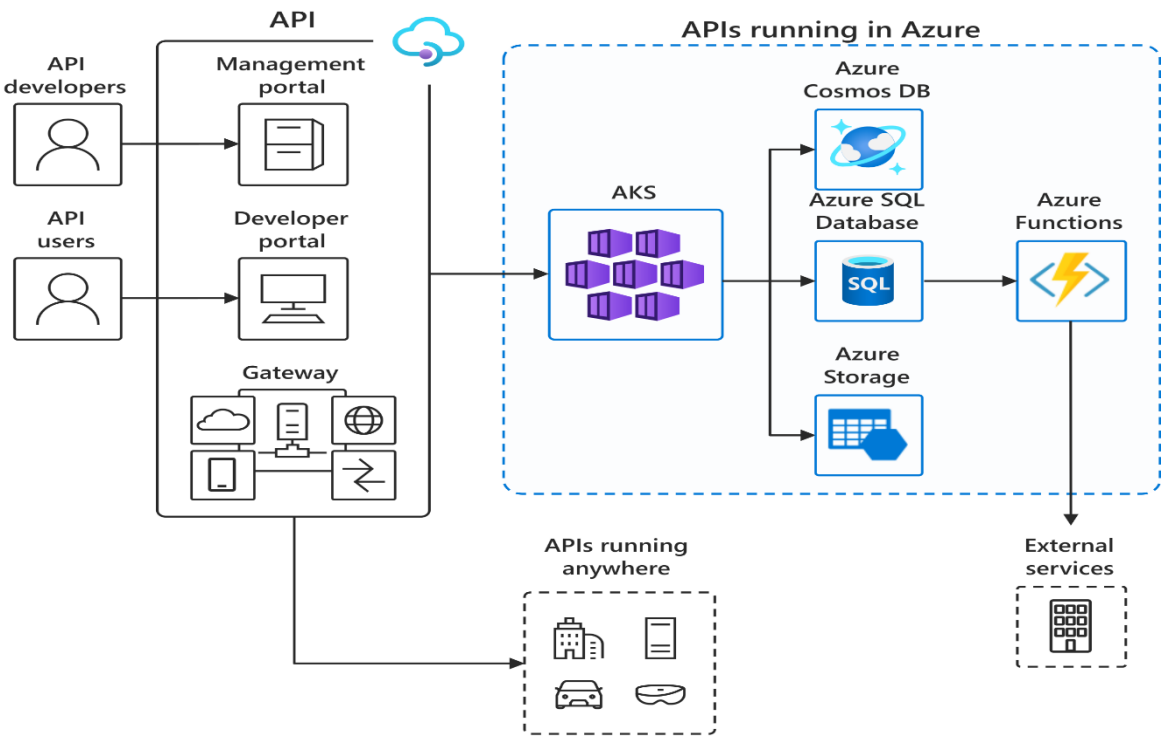


Рисунок 2.3 - Взаємодія API з сервером та БД

При реалізації основних цілей нашого застосунку проаналізовано якісні існуючі програмні рішення та виділено функціонал, який найкраще підходить для створюваного застосунку, має мінімальний необхідний набір функцій для зручного використання користувачем, що не зменшує продуктивність роботи програми.

Таблиця 2.1 - Функціональність веб-сайту замовлення продуктів в магазині

Назва функціоналу	Опис функціоналу
Реєстрація та логін	Користувач може ввійти на сайт ввівши свій емейл та пароль, або зареєструватись на сайті, ввівши емейл, Можлива двофакторна ідентифікація

Продовження таблиці 2.1

Створення кошику	У користувача є можливість створити кошик де будуть міститися товари, які він додав
Додавання на видалення товару з кошику	Користувач може додавати або видаляти товар з кошику в середині нього або в списку товарів, які переглядаються.
Заповнення власного кабінету	Функція дає змогу користувачу спростити подальші замовлення, оскільки не потрібно буде повторно вводити дані.
Змінити пароль	Зареєстрований користувач може змінити пароль для його облікового запису
Змінити емейл	Зареєстрований користувач може змінити емейл в його обліковому записі.
Редагування кошику	Клієнт може модифікувати вміст кошику залежно від суми замовлення.

Продовження таблиці 2.1

Створення замовлення	Користувач може створити замовлення , але якщо він не заповнив обліковий запис, потрібно заповнити деталі замовлення
Підтвердження замовлення	Користувач підтверджує замовлення після того , як перевірить правильність вводу даних.
Відправка замовлень на пошту	Користувачу відправляється його замовлення на вказаний ним email.

Це є основний функціонал, що вимагається від нашого веб-серверу. Він простий для розуміння та може працювати стабільно, навіть при великій кількості запитів.

Пропонується наступний принцип роботи з даним інтерфейсом. Якщо користувач заходить на сайт у нього може бути збережена активна сесія або навпаки. Для збереження сесії потрібно прийняти правила зберігання даних в cookies. Після цього користувачу надається унікальний ідентифікатор сесії , який згодом зберігається в базі даних. При роз’єднанні з сайтом користувач може повернутись в свою активну сесію. В базі даних буде йти пошук по ідентифікатору сесії і якщо вона знайдеться то користувачу повернеться активна сесія. Він отримає відповідну початкову сторінку на якій він завершив попередній візит на сайті.

Коли у користувача немає активної сесії для нього створиться ідентифікатор сесії і якщо він прийме правила зберігання даних в cookies він в подальшому матиме змогу користуватися таким функціоналом.

Схему роботи API наведено на рисунку 2.4

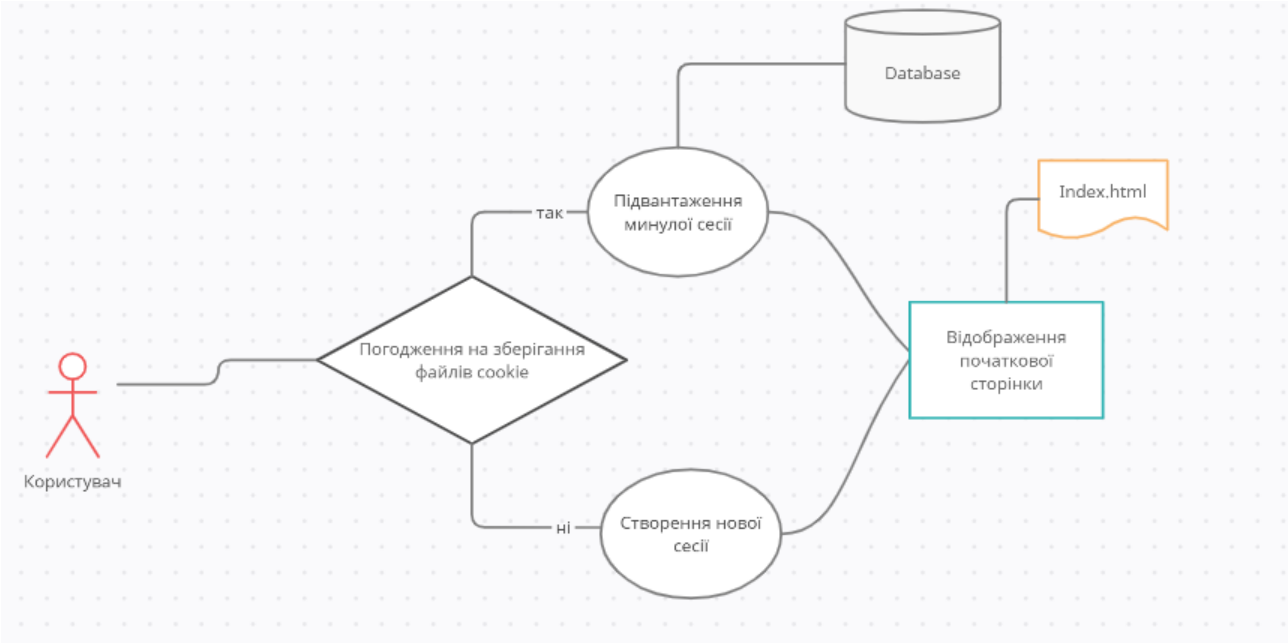


Рисунок 2.4 - Схема роботи API з клієнтом

З метою створення оптимальних колекцій для даних я обрав функціонал класів, які знаходяться в бібліотеці System.Collections.Generic.

Всередині програми ми будемо використовувати контролери, які і будуть маніпулювати нашими колекціями. Це PieController та ShoppingCartController.

В середині контролерів інкапсульовані методи які модифікують наші колекції.

AddToShoppingCart(int pieId) -метод , який додає певний об’єкт до списку.

Параметр який передається в метод є унікальним ідентифікатором об’єкту. За допомогою нього в методі проходить пошук по базі даних і у разі знаходження

елемент додається. RemoveFromShoppingCart(int pieId) -метод, аналогічний по методу роботи, але він видаляє елемент зі списку.

Збереження колекцій в базі даних , які має певний об’єкт здійснюється завдяки внутрішньому зв’язку через foreign key, де в таблиці буде міститися посилання на колекцію з іншої таблиці.

Таблиця 2.2 - Розподіл відповідальності між класами по вимогам MVC веб-сайту замовлення товарів в магазині

Model	
Назва класу	Опис класу
Pie	Структура класу Pie яка описує всі відомості про об’єкт Pie.
Category	Інформація про категорію пирога, клас містить список пирогів, які належать до даної категорії.
ShoppingCart	Містить відомості про стан кошику. Має список продуктів. Зберігає посилання на нього в базі даних.
Order	Містить особисту інформацію про користувача, яку він надає після підтвердження в кошику. Вміщає дату замовлення на адресу доставки.

Продовження таблиці 2.2

Controller	
IdentityController	Клас підрозділу Контролер, що відповідає за логін та реєстрацію, також надає зареєстрованому користувачу його унікальний токен, що засвідчує його авторизацію та аутентифікацію
PieController	Клас підрозділу Контролер, що відповідає за надання даних про продукт, список продуктів. Містить метод деталі продукту.
ShoppingCartController	Клас підрозділу Контролер, що відповідає за створення та редагування товарів в кошику. Містить окремий компонент підсумовування кількості товарів на передачу його в навігаційне меню.
OrderController	Клас підрозділу Контролер, що відповідає за надання, створення та редагування інформації про замовлення. Дозволяє користувачу редагувати дані щодо замовлення

Продовження таблиці 2.2

View	
List	Клас, який забезпечує відображення списку товарів в залежності від категорії, яка передається як параметр.
Details	Клас, який забезпечує відображення деталей продукту. В собі містить partial view для товару.
Login	Клас, який забезпечує відображення розділу логінізації та містить валідаційні перевірки для логіну та паролю.
Register	Клас, який забезпечує відображення розділу реєстрації та передачі даних для занесення в базу даних..

2.2 Засоби навігації

У каркасі проекту на рисунку 2.5 представлено внутрішню структуру програми.

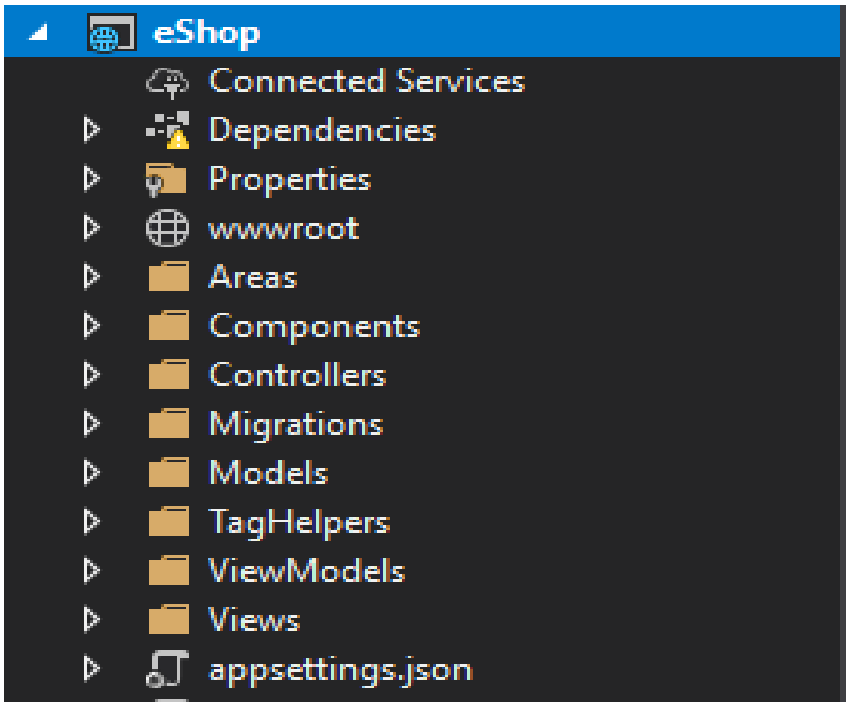


Рисунок 2.5 - Структура проекту для сайту замовлень

Основу проекту складають 3 основні папки Model View Controller. Інші каталоги складають допоміжний функціонал для оптимізації, передачі даних з бази даних до представлень.

Таблиця 2.3 - Опис системних папок веб-сайту замовлення товарів

Назва папки	Опис
properties	Папка в якій знаходиться налаштування для запуску веб-сайту який обрано.
Areas	Папка повністю відповідальна за авторизацію та аутентифікацію користувача.
Controllers	Папка у якій знаходяться усі контролери, що відповідають за кінцеві точки розроблених запитів (endpoints). Приклад контролеру знаходиться у додатку.
Migrations	Папка містить всі класи в який є якась модифікація бази даних(зміна структури або заповнення даними)
Models	Папка містить опис розглянутих вище моделей у форматі .cs. Також ви можете подивитись на приклад моделі у додатку А
TagHelpers, ViewModels	У цих папках розміщені допоміжні класи для більш зручної передачі даних у View.

Продовження таблиці 2.3

Views	Папка в якій знаходяться всі представлення у форматі .cshtml , які згодом компілюються в html Razor-двигуном.

Навігація по сторінці

Навігація сторінкою – це процес переходу з однієї сторінки на іншу на вашому веб-сайті. Існує багато способів переходу з однієї сторінки на іншу в ASP.NET.

- Навігація на стороні клієнта
- Розміщення між сторінками
- Переспрямування браузера на стороні клієнта
- Передача на стороні сервера

Навігація на стороні клієнта

Навігація на стороні клієнта означає, що ваш браузер (клієнт) або HTML запитує веб-сторінку, натиснувши HyperLink. Елемент керування HyperLink — це найпростіший спосіб перейти на іншу веб-сторінку. Він створює посилання на іншу веб-сторінку. Елемент керування HyperLink відтворює тег прив’язки HTML, <a>. Властивість Text використовується для відображення тексту в HyperLink. Ми також можемо відобразити зображення на цьому елементі керування замість тексту. Щоб відобразити зображення, ми повинні встановити властивість ImageUrl

Home Page

Коли користувач натискає елемент керування HyperLink, браузер викликає веб-сторінку Welcome.aspx. Ви також можете використовувати JavaScript для навігації на стороні клієнта. Для цього слід використовувати кнопку введення HTML. Об'єкт документа представляє веб-сторінку в клієнтському JavaScript. Ви можете викликати певну веб-сторінку, вказавши назву сторінки aspx до її властивості Document location.

2.3 Види забезпечення для створення веб-сайтів

Visual Studio — це інтегроване середовище розробки (IDE), розроблене Microsoft для розробки графічного інтерфейсу користувача (графічного інтерфейсу користувача), консолі, веб-додатків, веб-програм, мобільних додатків, хмарних програм і веб-сервісів тощо. За допомогою цієї IDE ви можете створювати керований код, а також рідний код. Він використовує різні платформи програмного забезпечення для розробки програмного забезпечення Microsoft, наприклад Windows Store, Microsoft Silverlight, Windows API тощо. Це не специфічна для мови IDE, оскільки ви можете використовувати її для написання коду на C#, C++, VB (Visual Basic), Python, JavaScript та багато інших мов. Він забезпечує підтримку 36 різних мов програмування. Він доступний як для Windows, так і для macOS.

Деякі популярні функції Visual Studio, які покращують вашу продуктивність під час розробки програмного забезпечення.

Squiggles and Quick Actions

Squiggles — це хвилясті підкреслення, які сповіщають вас про помилки або потенційні проблеми у вашому коді під час введення. Ці візуальні підказки допоможуть вам негайно виправити проблеми, не чекаючи виявлення помилок під час збірки або виконання. Якщо ви наведете курсор на завитку, ви побачите більше інформації про помилку. На лівому полі також може з’явитися лампочка, яка показує швидкі дії, які можна виконати, щоб виправити помилку.

Code Cleanup

Натиснувши кнопку, ви можете відформатувати свій код та застосувати будь-які виправлення коду, запропоновані налаштуваннями стилю коду, умовами .editorconfig та аналізаторами Roslyn. Очищення коду, яке наразі доступне лише для коду C#, допомагає вирішити проблеми у вашому коді, перш ніж він буде перевірятися.

IntelliSense

IntelliSense — це набір функцій, які відображають інформацію про ваш код безпосередньо в редакторі і, в деяких випадках, записують невеликі фрагменти коду для вас. Це як мати базову документацію в редакторі, тож вам не доведеться шукати інформацію про типи в іншому місці.

Live Share

Спільно редагуйте та налагоджуйте з іншими в режимі реального часу, незалежно від типу вашої програми чи мови програмування. Ви можете миттєво та безпечно поділитися своїм проектом. Ви також можете ділитися сеансами налагодження, екземплярами терміналів, веб-програмами localhost, голосовими дзвінками тощо.

Visual Studio search

Меню, параметри та властивості Visual Studio іноді можуть здаватися надто значними. Пошук у Visual Studio або Ctrl+Q — це чудовий спосіб швидко знайти функції та код IDE в одному місці.

Call Hierarchy

У вікні «Ієрархія викликів» показано методи, які викликають вибраний метод. Ця інформація може бути корисною, коли ви думаєте про зміну чи видалення методу, або коли ви намагаєтесь відстежити помилку.

CodeLens

CodeLens допомагає знайти посилання на код, зміни коду, пов’язані помилки, робочі елементи, огляди коду та модульні тести, не виходячи з редактора.

Postman

Postman — один із найпопулярніших інструментів тестування програмного забезпечення, який використовується для тестування API. За допомогою цього інструменту розробники можуть легко створювати, тестувати, обмінюватися та документувати API.

При розробці я буду використовувати програмне забезпечення Postman для надсилання та отримання запитів, даних POST на сервер.

Postman — це окрема платформа API для тестування програмного забезпечення (Інтерфейс програмного забезпечення) для створення, тестування, проектування, модифікації та документування API. Це простий графічний інтерфейс користувача для надсилання та перегляду HTTP-запитів і відповідей.

Під час використання Postman для цілей тестування не потрібно писати код мережі клієнта HTTP. Замість цього ми створюємо набори тестів, які називаються колекціями, і дозволяємо Postman взаємодіяти з API.

У цей інструмент вбудовано майже будь-яку функціональність, яка може знадобитися будь-якому розробнику. Цей інструмент має можливість робити різні типи запитів HTTP, як-от GET, POST, PUT, PATCH, і перетворювати API на код для таких мов, як JavaScript і Python.

Postman заснований на широкому спектрі надзвичайно зручних електроінструментів. Для більш ніж 8 мільйонів користувачів Postman став інструментом зручності. Нижче наведено причини, чому було обрано Postman:

- Доступність – його можна використовувати де завгодно після встановлення Postman на пристрій, просто увійшовши в обліковий запис.
- Використання Collections-Postman дозволяє користувачам створювати колекції для своїх API-викликів. Кожен набір може створювати кілька запитів і вкладених папок. Це допоможе організувати тестові набори.
- Розробка тесту. Щоб перевірити контрольні точки, перевірка успішного статусу відповіді HTTP має бути додана до кожного виклику API.
- Автоматичне тестування-тести можна виконувати з кількома повтореннями або ітераціями за допомогою Collection Runner або Newman, що економить час для повторних тестів.
- Створення середовищ. Дизайн кількох середовищ призводить до меншої реплікації тестів, оскільки можна використовувати ту саму колекцію, але для інших параметрів.
- Налагодження. Щоб ефективно налагоджувати тести, консоль листоноші допомагає відстежувати, які дані витягуються.
- Співпраця. Ви можете імпортувати або експортувати колекції та середовища, щоб покращити спільний доступ до файлів. Ви також можете використовувати пряме з’єднання для обміну колекціями.

- Безперервна інтеграція – може підтримувати безперервну інтеграцію.

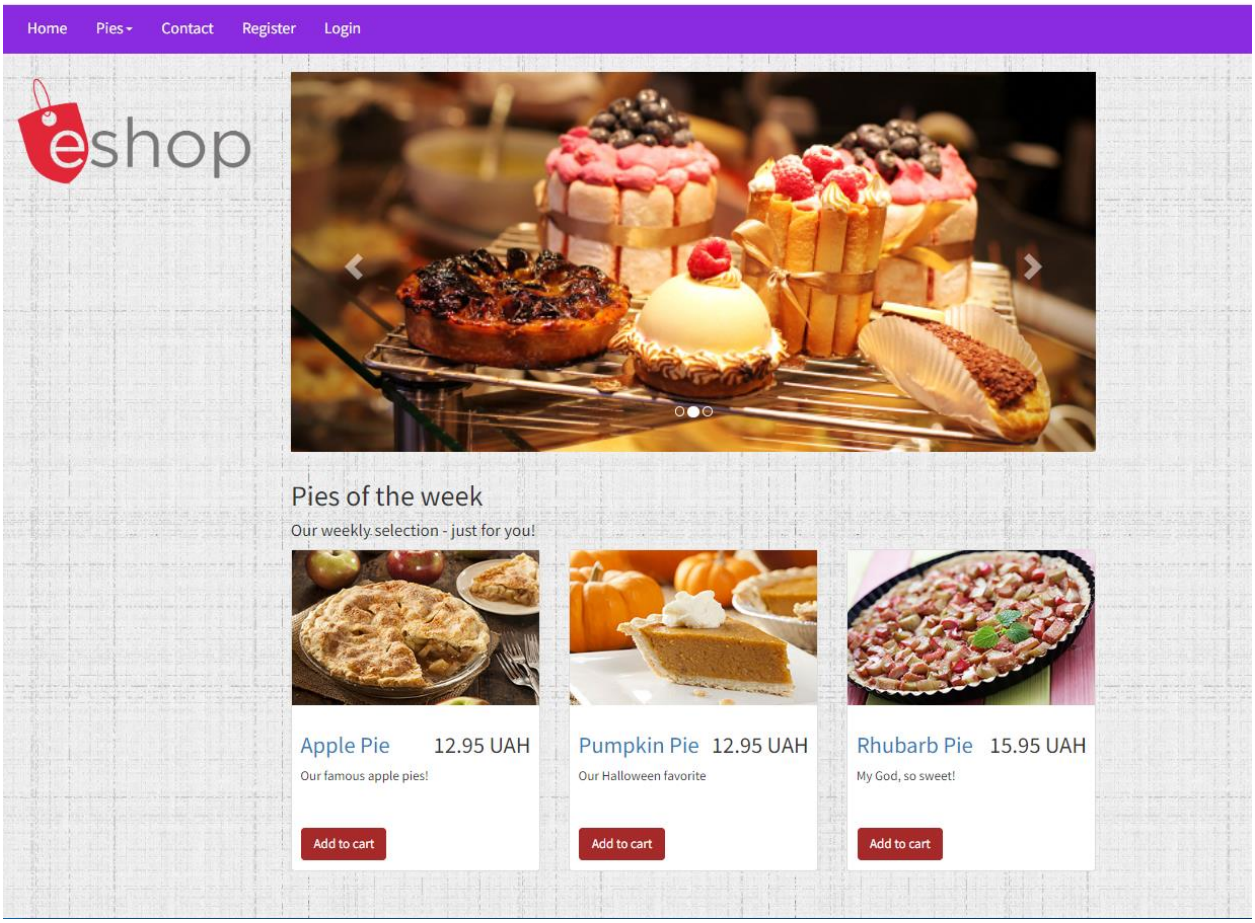
РОЗДІЛ 3

РЕАЛІЗАЦІЯ СИСТЕМИ ЗАМОВЛЕННЯ ТОВАРІВ

3.1 Реалізація роботи веб-сайту

Для демонстрації роботи веб-сайту був створений застосунок із використанням Razor-сторінок, для візуального оформлення було використано css-бібліотеку Bootstrap.

Для початку перегляду сайту потрібно запустити сервер. Вибираємо



сервер Kestrel. Огляд сайту починається з стартової сторінки (рис 3.1)

Рисунок 3.1 - Початкова сторінка

Після цього користувач може вільно пересуватися по сайту використовуючи навігаційне меню зверху, яке містить:

- Домашня сторінка
- Продукти(у нашому випадку пироги)
- Контактна сторінка, де ви можете зв’язатись з магазином.
- Реєстрація
- Логінізація, якщо аккаунт вже створено

Можна створити кошик товарів і вже після дії підтвердження покупки наш ресурс запропонує користувачу зареєструватися або ввійти в свій аккаунт, якщо він вже створений.

3.2 Інструкція користування веб-сайтом

Нижче наведено форму для реєстрації та входу відповідно(рис. 3.2, рис 3.3).

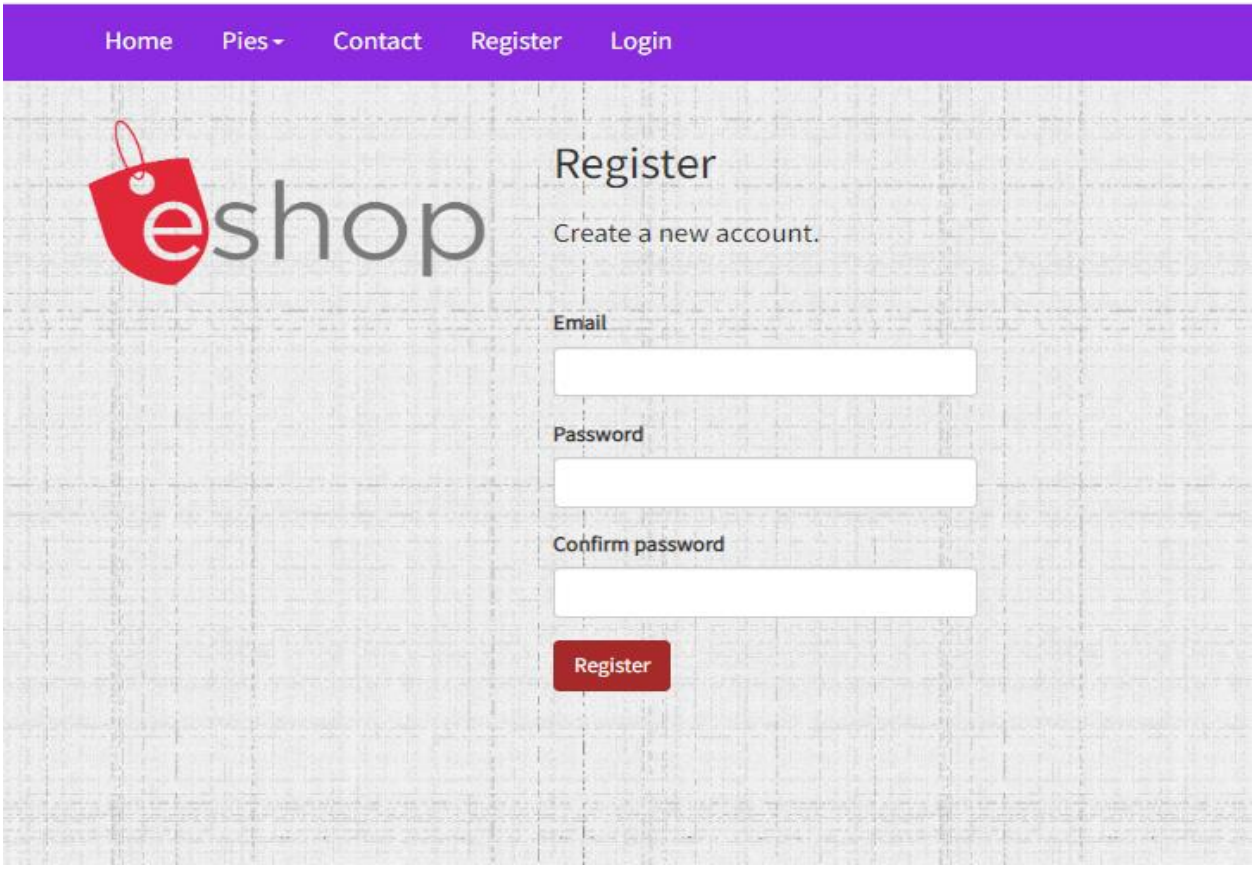


Рисунок 3.2 - Форма реєстрації

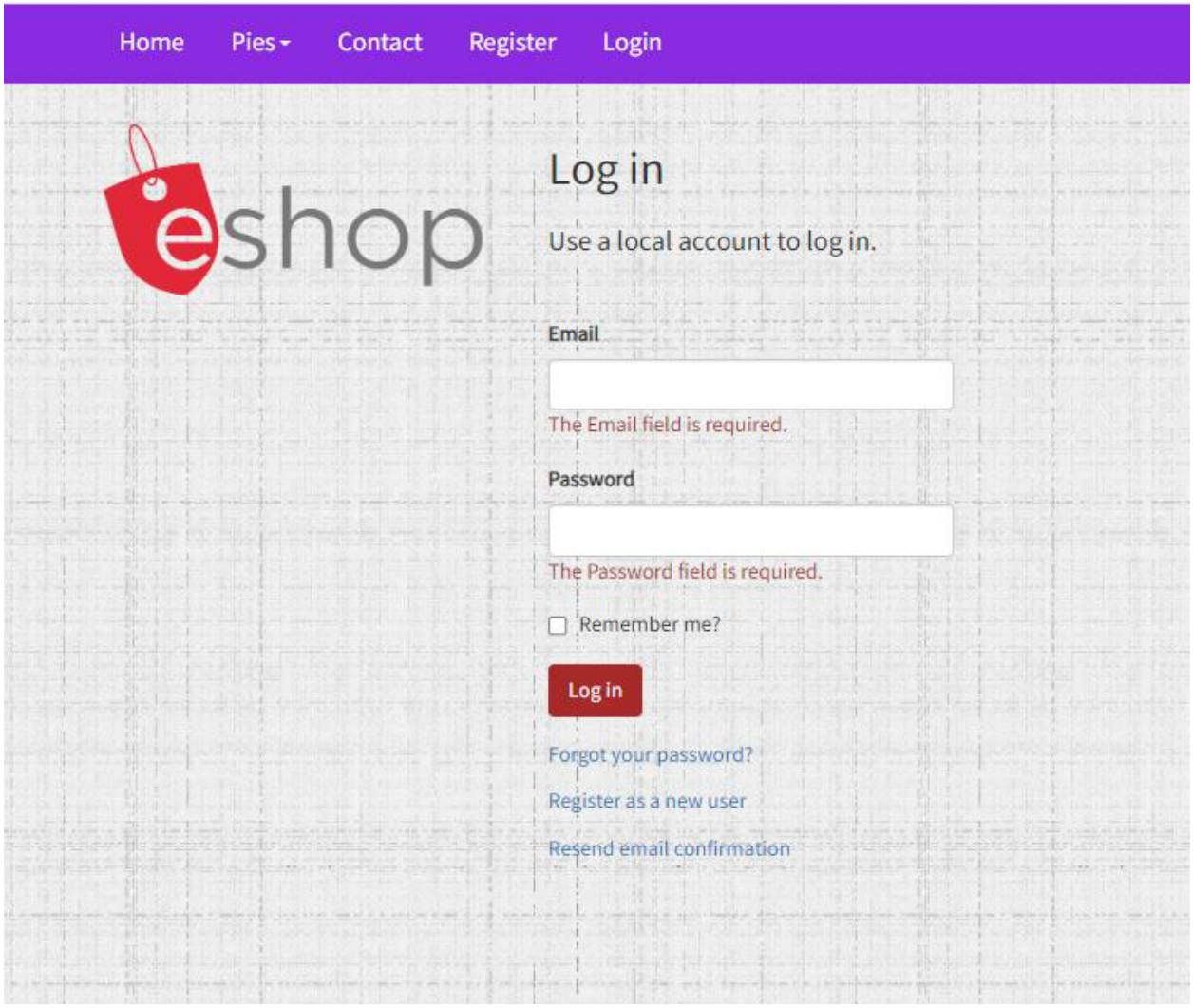


Рисунок 3.3 - Форма входу

В формах реєстрації та входу присутні шаблони перевірки введених даних. Якщо користувача з таким email-ом не знайдено або пароль неправильний, буде показано відповідну помилку. Також є checkbox Remember me? і якщо він активний при наступному відвідуванні сайту користувачу не потрібно буде заново вводити логін та пароль.

Якщо користувач не може згадати пароль він може змінити його натиснувши посилання Forgot your password? і на вказану пошту прийде лист з підтвердженням на зміну паролю.

Вже тепер у користувача є власний кабінет, де він може додати особисту інформацію для полегшення заповнення бланку замовлення. Особистий кабінет зображено на рис. 3.4.

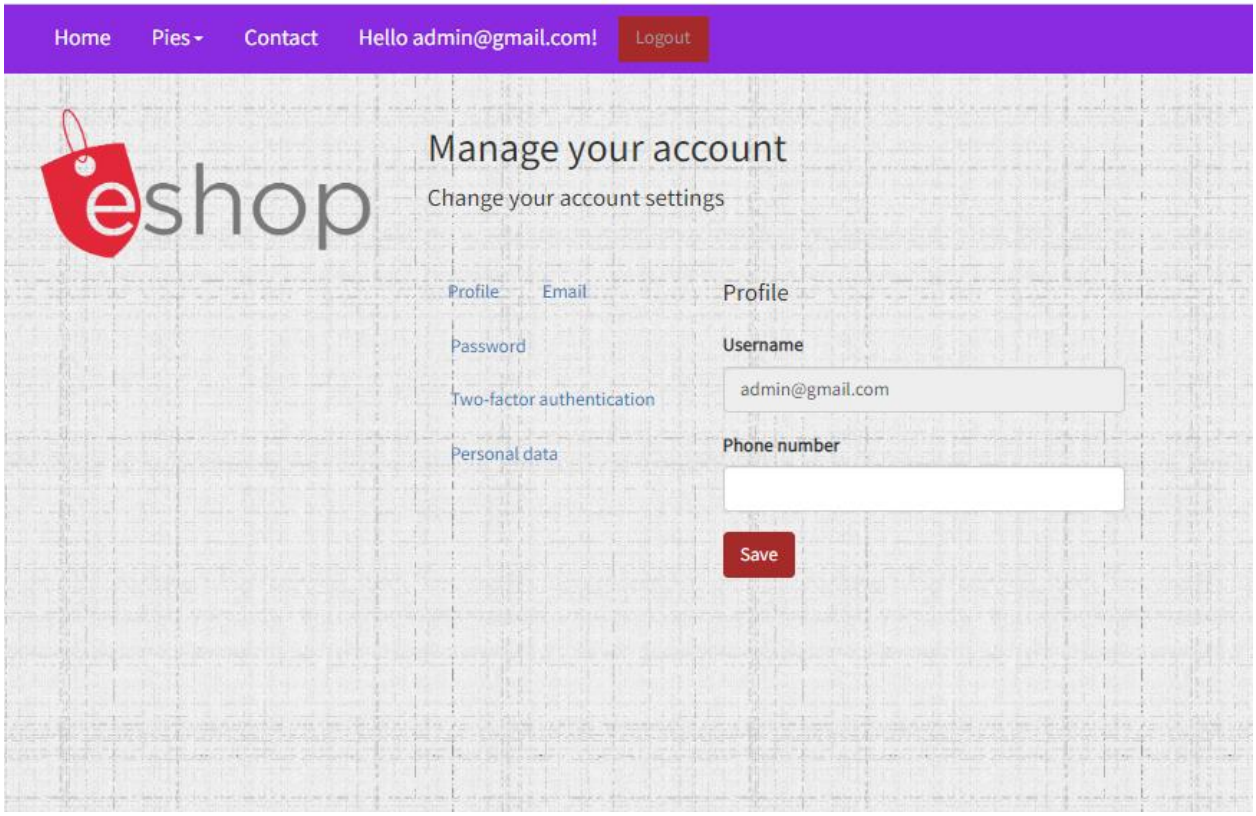


Рисунок 3.4 - Персональний кабінет користувача

Тут користувач може виконувати менеджмент аккаунтом. Він може змінити email або пароль. Додати або змінити номер телефону, встановити персональні дані.

Функція двофакторної аутентифікації також присутня (рис 3.5).

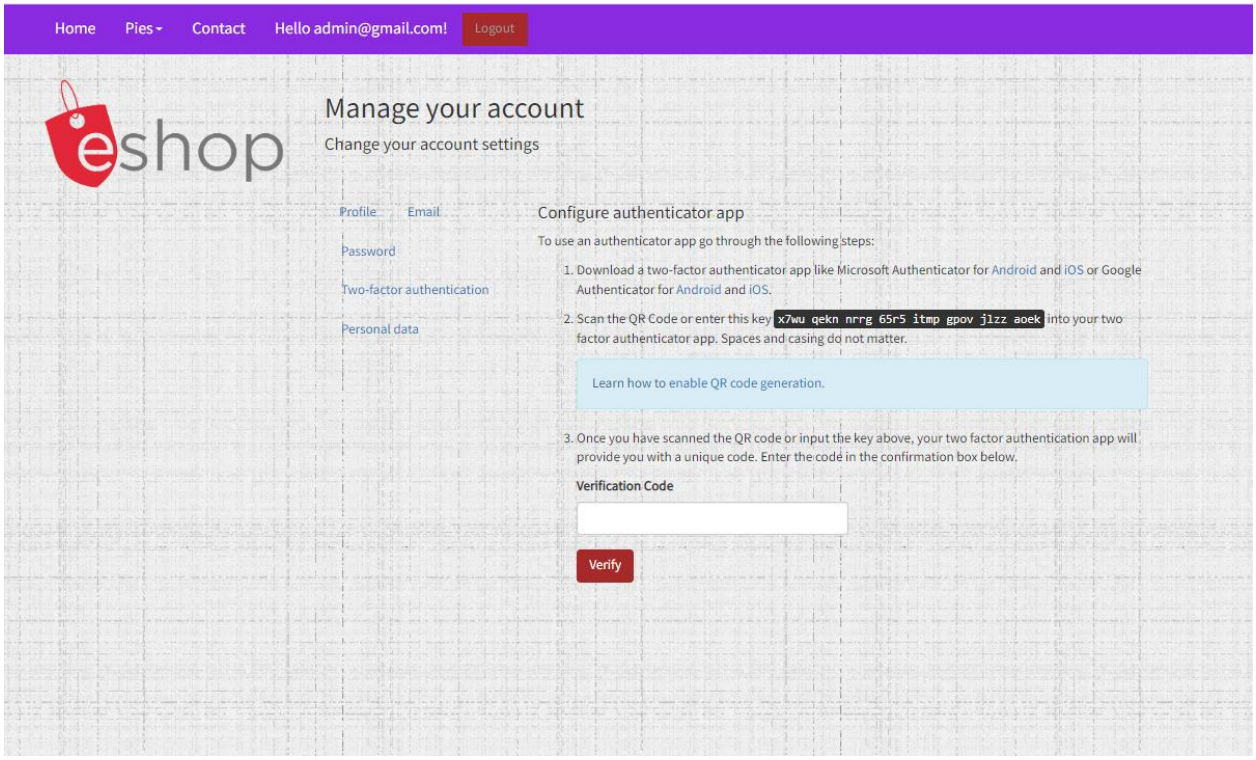


Рисунок 3.5 - Меню двофакторної аутентифікації

Користувач має змогу прив’язати до свого аккаунту застосунок і він зможе автентифікуватися за допомогою цього застосунку.

3.3 Тестування веб-сайту

В процесі навігації по сайту користувач вибере певні продукти, які будуть знаходитись в кошику (рис. 3.6).

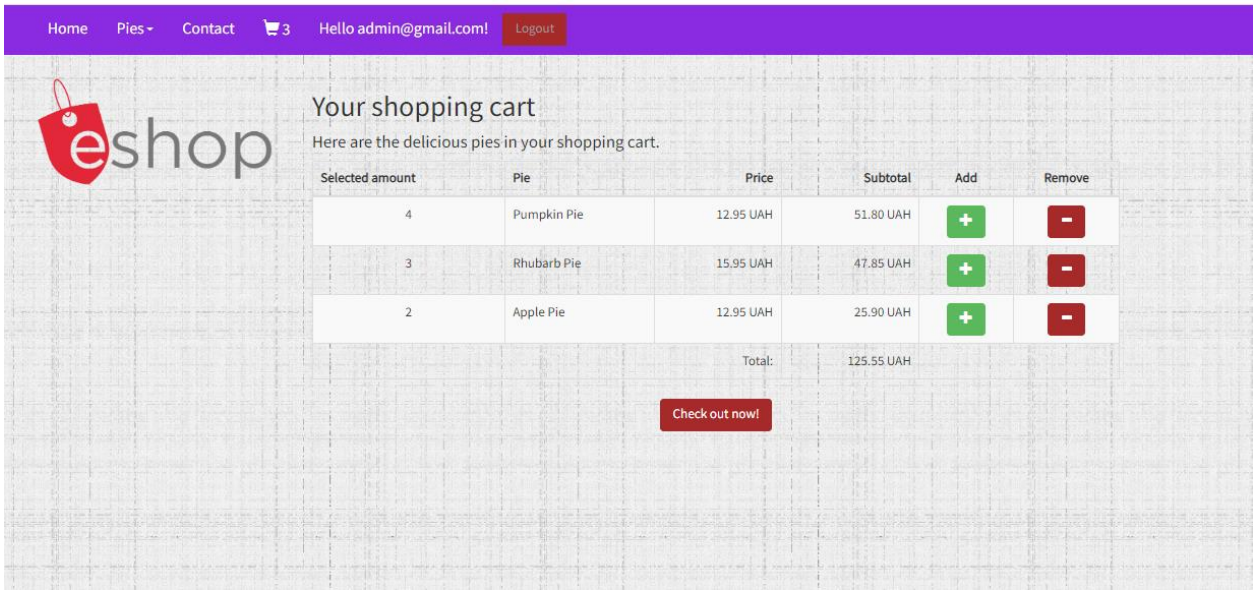


Рисунок 3.6 - Кошик користувача

Зайшовши туди, він матиме змогу переглянути свої замовлення та редагувати їх(додавати кількість або зменшувати). Після цього, щоб зробити замовлення, потрібно натиснути на клавішу Checkout (Розрахуватися). Користувачу буде надана форма заповнення замовлення, яка показана на рисунку 3.7. Після цього замовлення буде передане в магазин і буде направлене клієнту. Отже, на прикладі цього сайту, показано, що його можна інтегрувати на різні ресурси. Він є кросплатформним, тому його можна використовувати на різних платформах таких, як Windows, Linux, Mac та інші.

Присутня можливість додавання на видалення адміном товарів. Коли користувач авторизується він переходить на стартову сторінку, але якщо у нього є адмін права навігаційне меню буде містити додаткову вкладку Settings. Перейшовши туди він побачить всі товари у вигляді таблиці, де будуть присутні функції додавання та видалення. Після виконання операції його буде перенаправлено на сторінку списку товарів.

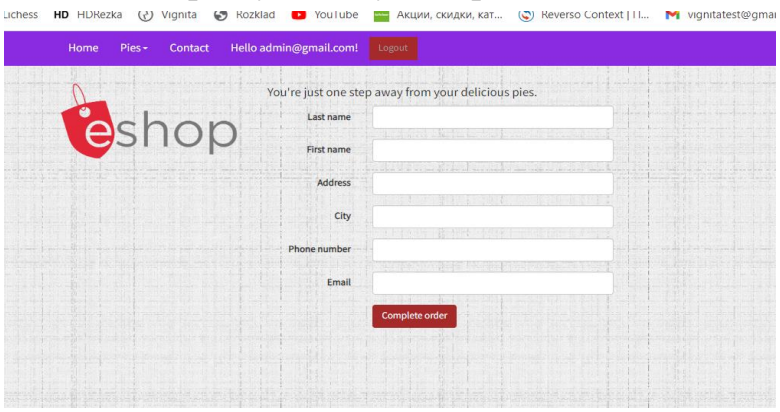
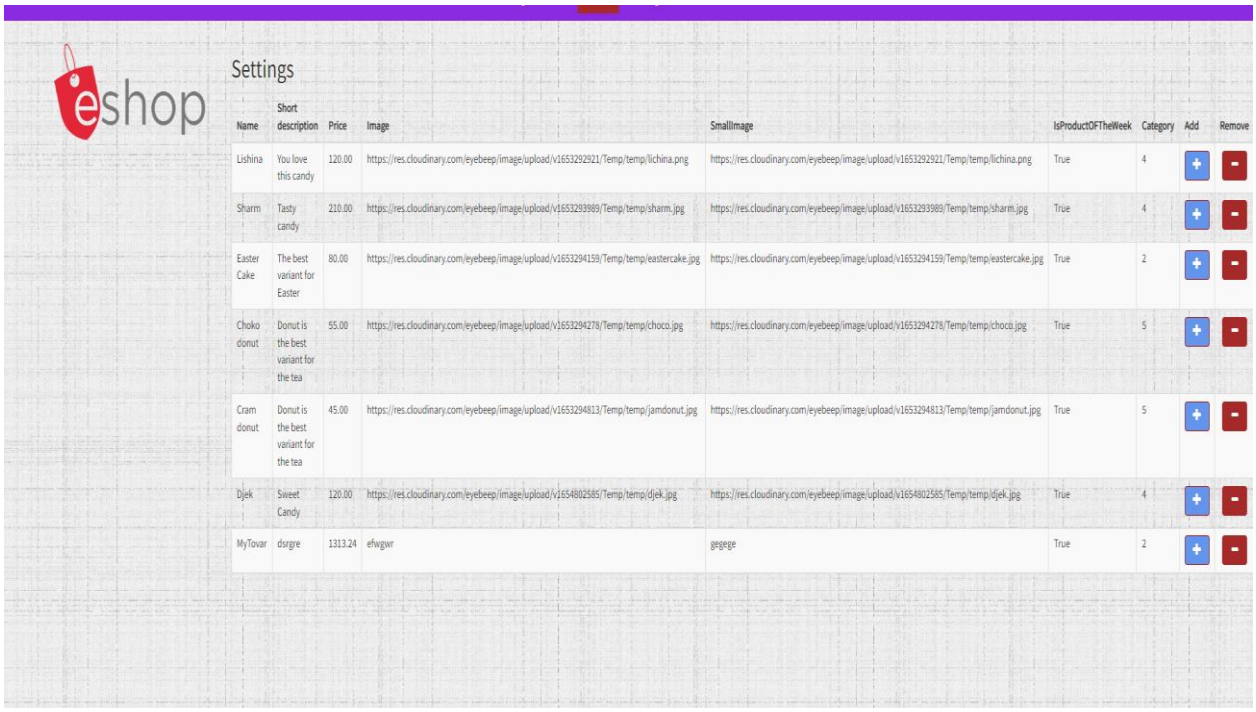


Рисунок 3.8 - Список товарів для адміна



ВИСНОВОК

У результаті виконання кваліфікаційної роботи бакалавра було спроектовано та розроблено веб-сайт замовлення продуктів в Інтернет-магазині, зокрема:

- здійснено аналіз і зроблено вибір засобів для розроблення сайту Інтернет-магазину;
- розроблено архітектуру веб-сайту та шляхи комунікації за протоколом «HTTP», розроблено схему роботи з базою даних MS SQL та реалізовано авторизацію та аутентифікацію користувача в системі;
- впроваджено веб-сайт .

Під час виконання дипломної роботи було проведено низку досліджень щодо архітектурних рішень. Були обрані гнучкі технології, які забезпечують розширюваність та кросплатформність цього сайту. Зібрано матеріали щодо проектування системи, тестування сайту, а також забезпечення авторизації та аутентифікації користувача.

Були вивчено низку технологій побудови архітектури сайтів; засоби забезпечення безпеки користувачів.

Застосунок реалізований мовою програмування **C#**, за основу взято платформу **ASP.NET Core** , а як базу даних обрано реляційну БД **MS SQL**. Для демонстрації роботи було створено сайт, де візуалізація коду забезпечено **Razor**-двигуном, який компілює .cshtml в html. Стилі та розмітку забезпечено бібліотекою **Bootstrap**.

Головною особливістю цієї програми є кросплатформність та дружність інтерфейсу, і її можна інтегрувати в будь-яку систему, незалежно від платформи або мови програмування. Основні параметри швидкодії програми є на високому рівні.

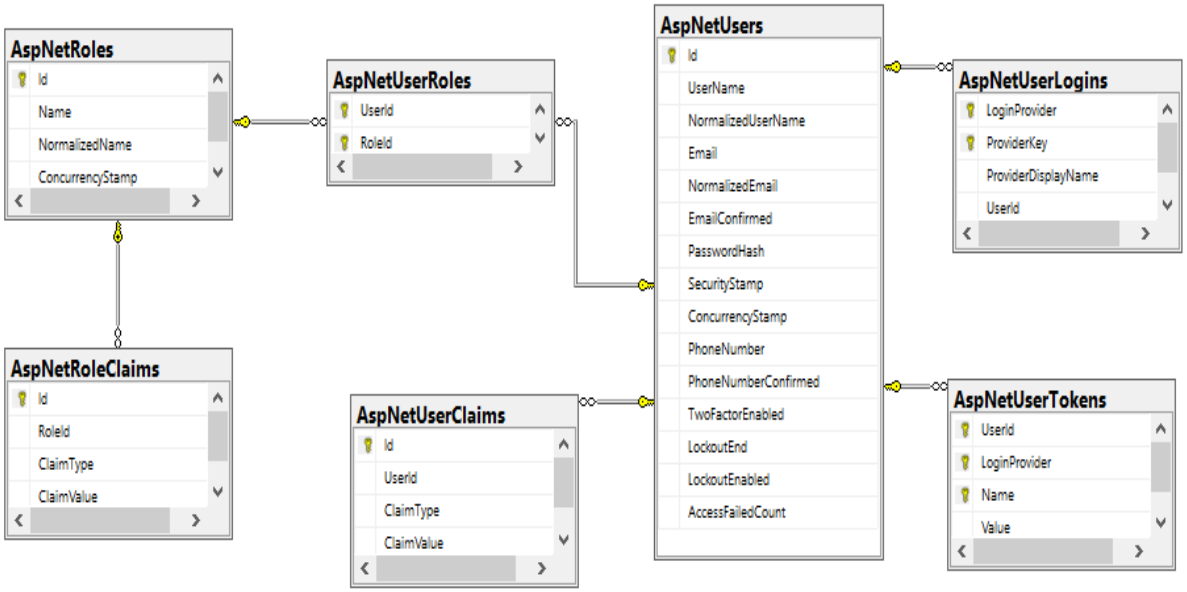
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Core веб-сайт. URL: https://metanit.com/sharp/aspnet5/ (переглянуто 10 квітня 2022 року) .
2. Патерн MVC веб-сайт. URL: https://habr.com/ru/post/181772/ (переглянуто 10 квітня 2022 року) .
3. What is MSSQL? веб-сайт. URL: https://www.atlantic.net/vps-hosting/what-is-mssql/ (переглянуто: 10 квітня 2022 року) .
4. Generics веб-сайт. URL: https://metanit.com/sharp/tutorial/3.12.php (переглянуто: 11 квітня 2022 року) .
5. Рейки веб-інтеграції. REST і SOAP веб-сайт. URL: https://www.intervolga.ru/blog/projects/relsy-veb-integratsii-rest-i-soap/(переглянуто: 11 квітня 2022 року) .
6. Підхід до вибору архітектури інтернет-магазину веб-сайт. URL: https://qna.habr.com/q/654851 переглянуто: 12 квітня 2022 року) .
7. Архітектура ASP.NET MVC 5 веб-сайт. URL: https://professorweb.ru/my/ASP_NET/mvc/level1/1_7.php (переглянуто: 12 квітня 2022 року) .
8. Тестування веб-застосунків веб-сайт. URL: https://metanit.com/sharp/mvc5/18.1.php (переглянуто: 13 квітня 2022 року) .
9. Тестування застосунків MVC ASP.NET Core веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/test-asp-net-core-mvc-apps переглянуто: 13 квітня 2022 року) .
10. 5 принципів створення структури сайту інтернет-магазину веб-сайт. URL: https://evo.business/5-principov-sozdaniya-struktury-sajta-internet-magazina/ (переглянуто: 13 квітня 2022 року) .
11. Розробка сучасних веб-застосунків за допомогою ASP.NET Core і Azure веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/ (переглянуто: 13 квітня 2022 року) .

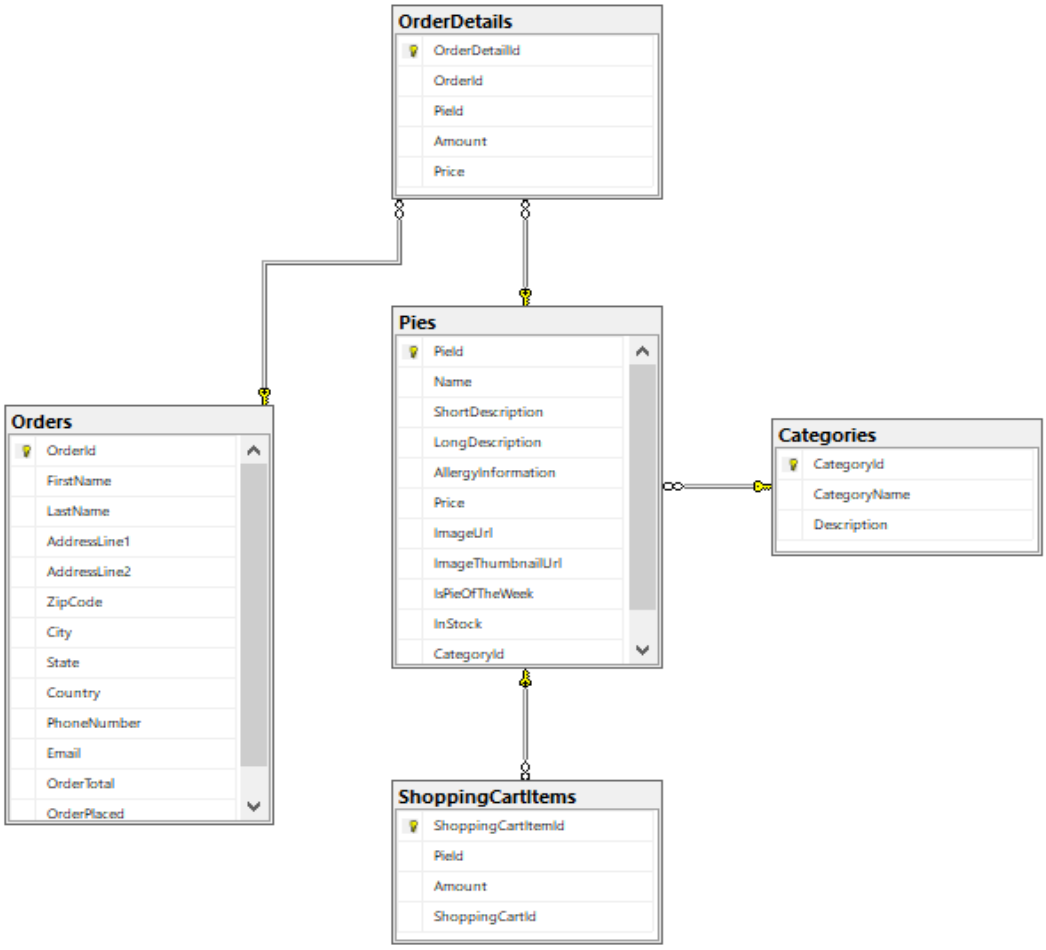
12. Фрімен Е. Вивчаємо HTML, XHTML і CSS = Head First HTML with CSS & XHTML. 2010. 656 с. (переглянуто: 13 квітня 2022 року) .
13. Тіттел Е., Ноубл Д. HTML, XHTML і CSS для чайників, 7-е видання HTML, XHTML & CSS For Dummies, 7th Edition.: «Діалектика», 2011. 400с. (переглянуто: 13 квітня 2022 року) .
14. Kestrel web server *Microsoft Software Developers Network* : веб-сайт. URL:https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-5.0 (переглянуто 3 травня 2022 року).
15. ASP.NET Cache *Web Training Room*: веб-сайт. URL: https://www.webtrainingroom.com/aspnetmvc/caching (переглянуто 3 травня 2022 року)
16. Введение в REST API — RESTful веб-сервіси - *Хабр* : веб-сайт. URL: https://habr.com/ru/post/483202/ (переглянуто 3 травня 2022 року).
17. What is REST – *REST API Tutorial*: веб-сайт. URL: https://restfulapi.net/ (переглянуто 3 травня 2022 року).
18. Роберт М. Чиста архітектура, 2021. – 352р.
19. Плескач В.Л., Затонацька Т.Г. Електронна комерція: Підручник. — Київ: Знання, 2007. — 535 с.
20. Diagrams of All The OpenID Connect Flows– *Medium*: веб-сайт. URL: https://darutk.medium.com/diagrams-of-all-the-openid-connect-flows-6968e3990660 (переглянуто 3 травня 2022 року).
21. Unit testing ASP.NET Core Identity– *Samueleresca*: веб-сайт. URL: https://samueleresca.net/unit-testing-asp-net-core-identity (переглянуто 3 травня 2022 року).
22. Фрімен А. Entity Framework Core 2 для ASP.NET Core MVC для професіоналов – Діалектика, 2019. – 626с.

23. LINQ to SQL — *Microsoft*: веб-сайт. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/sql/linq/ (переглянуто 7 травня 2022 року).
24. SHA-1 — *Вікіпедія*: веб-сайт. URL: https://uk.wikipedia.org/wiki/SHA-1 (переглянуто 7 травня 2022 року).
25. Sass — *Sass Lang*: веб-сайт. URL: https://sass-lang.com (переглянуто 7 травня 2022 року).

ДОДАТОК А



Структура таблиць для авторизації та аутентифікації.



Діаграма зв’язків в базі даних MS SQL

ДОДАТОК Б

```
using eShop.Models;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.Collections.Generic;
using System.Linq;

namespace eShop.Areas.Identity.Pages.Account
{
    public class AdminModel : PageModel
    {

        private readonly AppDbContext _appDbContext;
        public List<Product> Products { get; set; }
        public string displayModal { get; set; } = "none";

        public AdminModel(AppDbContext appDbContext)
        {
            _appDbContext = appDbContext;
            Products = _appDbContext.Products.Skip(8).ToList();
        }

        public void ChangeModalState()
        {
            var arr = new[] { "none", "block" };
            displayModal = displayModal == arr[0] ? arr[1] : arr[0];
        }


    }
}
@page
@model eShop.Areas.Identity.Pages.Account.AdminModel
@{
    ViewData["Title"] = "Settings";
}
<h2>@ViewData["Title"]</h2>
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>Name</th>
            <th>Short description</th>
            <th>Price</th>
            <th>Image</th>
            <th>IsProductOFTheWeek</th>
            <th>Category</th>
            <th>Add</th>
            <th>Remove</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var product in Model.Products)
        {
            <tr>
                <td>@product.Name</td>
                <td>@product.ShortDescription</td>
                <td>@product.Price</td>
                <td>@(product.ImageUrl)</td>
                <td>@(product.IsProductOfTheWeek)</td>
                <td>@(product.CategoryId)</td>
```

```
using Microsoft.AspNetCore.Mvc;
using eShop.Models;
using eShop.ViewModels;

namespace eShop.Components
{
    public class EndingCart: ViewComponent
    {
        private readonly OrderingBacket _orderingBacket;

        public EndingCart(OrderingBacket orderingBacket)
        {
            _orderingBacket = orderingBacket;
        }

        public IViewComponentResult Invoke()
        {

            var product = _orderingBacket.GetItemInTheBacket();
            _orderingBacket.ShoppingCartItems = product;

            var cartView = new ShoppingCartViewModel
            {
                OrderingBacket = _orderingBacket,
                ShoppingCartTotal = _orderingBacket.GetOrderingBacketSum()

            };
            return View(cartView);
        }
    }
}
using Microsoft.AspNetCore.Mvc;
using eShop.Models;
using System.Linq;

namespace eShop.Components
{
    public class SortMenu : ViewComponent
    {
        private readonly ICategoryRepository _repo;
        public SortMenu(ICategoryRepository repo)
        {
            _repo = repo;
        }

        public IViewComponentResult Invoke()
        {
            var sorting = _repo.AllCategories.OrderBy(c => c.CategoryName);
            return View(sorting);
        }



    }
}
using eShop.Models;
using eShop.ViewModels;
using Microsoft.AspNetCore.Mvc;
```

```
namespace eShop.Controllers
{
    public class HomeController : Controller
    {
        private readonly IProductRepository _myRepository;

        public IActionResult Index()
        {
            var home = new HomeViewModel();


            home.ProductsOfTheWeek = _myRepository.ProductsOfTheWeek;

            return View(home);

        }
        public HomeController(IProductRepository myRepository)
        {
            _myRepository = myRepository;
        }


    }
}
using eShop.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;


namespace eShop.Controllers
{
    [Authorize]
    public class OrderController : Controller
    {
        private readonly IZamovlennyaRepository _zamovlennyaRepository;
        private readonly OrderingBacket _orderingBacket;

        public OrderController(OrderingBacket orderingBacket, IZamovlennyaRepository
zamovlennyaRepository)
        {
            _orderingBacket = orderingBacket;
            _zamovlennyaRepository = zamovlennyaRepository;
        }

        public IActionResult Checkout()
        {
            return View();
        }


        [HttpPost]
        public IActionResult Checkout(Order order)
        {
            var list = _orderingBacket.GetItemInTheBacket();
            _orderingBacket.ShoppingCartItems = list;

            if (_orderingBacket.ShoppingCartItems.Count == 0)
            {
```

```
                ModelState.AddModelError("", "Your backet is empty, add some products
first");
            }

            if (!ModelState.IsValid) return View(order);
            _zamovlennyaRepository.CreateOrder(order);
            _orderingBacket.ClearBacket();
            return RedirectToAction("CheckoutIsFinished");
        }

        public IActionResult CheckoutIsFinished()
        {
            ViewBag.CheckoutCompleteMessage = "Your order is accepted. You'll retrieve
our sweetness!";
            return View();
        }




    }
}
using eShop.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;


namespace eShop.Controllers
{
    [Authorize]
    public class OrderController : Controller
    {
        private readonly IZamovlennyaRepository _zamovlennyaRepository;
        private readonly OrderingBacket _orderingBacket;

        public OrderController(OrderingBacket orderingBacket, IZamovlennyaRepository
zamovlennyaRepository)
        {
            _orderingBacket = orderingBacket;
            _zamovlennyaRepository = zamovlennyaRepository;
        }

        public IActionResult Checkout()
        {
            return View();
        }


        [HttpPost]
        public IActionResult Checkout(Order order)
        {
            var list = _orderingBacket.GetItemInTheBacket();
            _orderingBacket.ShoppingCartItems = list;

            if (_orderingBacket.ShoppingCartItems.Count == 0)
            {
                ModelState.AddModelError("", "Your backet is empty, add some products
first");
            }
```

```
            if (!ModelState.IsValid) return View(order);
            _zamovlennyaRepository.CreateOrder(order);
            _orderingBacket.ClearBacket();
            return RedirectToAction("CheckoutIsFinished");
        }

        public IActionResult CheckoutIsFinished()
        {
            ViewBag.CheckoutCompleteMessage = "Your order is accepted. You'll retrieve
our sweetness!";
            return View();
        }




    }
}
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using eShop.Models;
using eShop.ViewModels;
using Microsoft.AspNetCore.Mvc;


namespace eShop.Controllers
{
    public class ProductController : Controller
    {
        private readonly ICategoryRepository _categoryStore;
        private readonly IProductRepository _productStore;



        public ProductController(IProductRepository productStore, ICategoryRepository
categoryStore)
        {
            _productStore = productStore;
            _categoryStore = categoryStore;
        }
        public IActionResult Details(int id)
        {
            var element = _productStore.GetProductById(id);
            if (element == null)
                return NotFound();

            return View(element);
        }

        public RedirectToActionResult RemoveProduct(int productId)
        {
            var myProduct = _productStore.AllProducts.FirstOrDefault(p => p.ProductId ==
productId);

            if (myProduct != null)
            {
                _productStore.RemoveProduct(myProduct.ProductId);
            }
            return RedirectToAction("List");
```

```
        }

        public IActionResult EditProduct()
        {
            return View();
        }


        [HttpPost]
        public IActionResult EditProduct(Product product)
        {
            _productStore.EditProduct(product);
            return RedirectToAction("List");
        }

        public ViewResult List(string category)
        {
            IEnumerable<Product> products;
            var currentCategory = "";

            if (string.IsNullOrEmpty(category))
            {
                products = _productStore.AllProducts.OrderBy(p => p.ProductId);
                currentCategory = "All products";
            }
            else
            {
                products = _productStore.AllProducts.Where(p => p.Category.CategoryName
== category)
                    .OrderBy(p => p.ProductId);
                currentCategory = _categoryStore.AllCategories.FirstOrDefault(c =>
c.CategoryName == category)?.CategoryName;
            }

            return View(new ProductsListViewModel
            {
                Products = products,
                CurrentCategory = currentCategory
            });
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace eShop
{
}
using eShop.Models;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;

namespace eShop
{
    public class Startup
    {
        }

@model HomeViewModel


<partial name="_Carousel" />

<h2>Products of the week</h2>
<h4>Our weekly selection - just for you!</h4>

<div class="row">

}
using eShop.Models;
using eShop.ViewModels;
using Microsoft.AspNetCore.Mvc;


namespace eShop.Controllers
{

    }
} @model Order

<form asp-action="Checkout" method="post" class="form-horizontal" role="form">
    <h4>You're just one step away from your delicious sweetness.</h4>

    <div asp-validation-summary="All" class="text-danger"></div>


    <div class="form-group">
        <label asp-for="LastName" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="LastName" class="form-control" />
            <span asp-validation-for="LastName" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
        <label asp-for="FirstName" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="FirstName" class="form-control" />
            <span asp-validation-for="FirstName" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
        <label asp-for="AddressLine1" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="AddressLine1" class="form-control" />
            <span asp-validation-for="AddressLine1" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
```

```
        <label asp-for="City" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="City" class="form-control" />
            <span asp-validation-for="City" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
        <label asp-for="PhoneNumber" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="PhoneNumber" class="form-control" />
            <span asp-validation-for="PhoneNumber" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
        <label asp-for="Email" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="Email" class="form-control" />
            <span asp-validation-for="Email" class="text-danger"></span>
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-5">
            <input type="submit" class="btn btn-primary" value="Complete order" />
        </div>
    </div>

</form>

@model ShoppingCartViewModel

<h2>Your shopping cart</h2>
<h4>Here are the delicious sweetness in your shopping cart.</h4>
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>Selected amount</th>
            <th>Product</th>
            <th class="text-right">Price</th>
            <th class="text-right">Subtotal</th>
            <th class="text-center">Add</th>
            <th class="text-center">Remove</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var line in Model.OrderingBacket.ShoppingCartItems)
        {
        <tr>
            <td class="text-center">@line.Amount</td>
            <td class="text-left">@line.Product.Name</td>
            <td class="text-right">@line.Product.Price UAH</td>
            <td class="text-right">
                @((line.Amount * line.Product.Price)) UAH
            </td>
            <td class="text-center">
                <a asp-controller="OrderingBacket" asp-action="AddToOrderingBacket" asp-
route-productId="@line.Product.ProductId"
                   type="button" class="btn btn-success" data-type="plus" data-
field="quant[2]">
```

```
                    <span class="glyphicon glyphicon-plus"></span>
                </a>
            </td>
            <td class="text-center">
                <a asp-controller="OrderingBacket" asp-action="RemoveFromOrderingBacket"
asp-route-productId="@line.Product.ProductId"
                   type="button" class="btn btn-danger" data-type="minus" data-
field="quant[2]">
                    <span class="glyphicon glyphicon-minus"></span>
                </a>
            </td>

        </tr>
        }
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3" class="text-right">Total:</td>
            <td class="text-right">
                @Model.ShoppingCartTotal.ToString("F2") UAH
            </td>
        </tr>
    </tfoot>
</table>


<div class="text-center">
    <a class="btn btn-primary" asp-controller="Order" asp-action="Checkout">Check out
now!</a>
</div>

@model Product

<form asp-action="EditProduct" method="post" class="form-horizontal" role="form">


    <div class="form-group">
        <label asp-for="Name" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="Name" class="form-control" />
        </div>
    </div>

    <div class="form-group">
        <label asp-for="ShortDescription" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="ShortDescription" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="LongDescription" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="LongDescription" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Price" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="Price" class="form-control" />
        </div>
    </div>
```

```
    <div class="form-group">
        <label asp-for="ImageUrl" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="ImageUrl" class="form-control" />
        </div>
    </div><div class="form-group">
        <label asp-for="ImageThumbnailUrl" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="ImageThumbnailUrl" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="IsProductOfTheWeek" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="IsProductOfTheWeek" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="CategoryId" class="col-md-2 control-label"></label>
        <div class="col-md-5">
            <input asp-for="CategoryId" class="form-control" />
        </div>
    </div>


    <div class="form-group">
        <div class="col-md-offset-2 col-md-5">
            <input type="submit" class="btn btn-primary" value="Update" />
        </div>
    </div>

</form>

@model Product

<h2>@Model.Name</h2>

<div class="thumbnail">
    <img style="height:500px" alt="@Model.Name" src="@Model.ImageUrl">
    <div class="caption-full">
        <h3 class="pull-right">@Model.Price</h3>
        <h3>
            <a href="#">@Model.Name</a>
        </h3>
        <h4>@Model.ShortDescription</h4>
        <p>@Model.LongDescription</p>
        <div class="addToCart">
            <p class="button">
                <a class="btn btn-primary" asp-controller="OrderingBacket" asp-
action="AddToOrderingBacket"
                   asp-route-productId="@Model.ProductId">Add to cart</a>
            </p>
        </div>
    </div>
</div>
```