**Міністерство освіти і науки України**

**Луцький національний технічний університет**
(повне найменування закладу вищої освіти)

**Факультет комп'ютерних та інформаційних технологій**
(повне найменування факультету)

**Кафедра комп'ютерної інженерії та кібербезпеки**
(повне найменування кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

# ІНФОРМАЦІЙНА СИСТЕМА ЗАСОБАМИ ПЛАТФОРМИ ASP.NET CORE MVC

# INFORMATION SYSTEM USING THE ASP.NET CORE MVC PLATFORM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Кузьмюк Олександр Васильович

(підпис)

Керівник:
к.т.н., доцент
Пех Петро Антонович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 07 » червня 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н.Чернящук
« 10 » 01 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

*Кузьмюку Олександру Васильовичу*
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інформаційна система засобами платформи ASP.NET CORE MVC*

Керівник роботи *к.т.н., доцент Пех Петро Антонович*

затверджені наказом закладу вищої освіти від «30» грудня 2023 року № 459/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 11.06.2024р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
*Вступ*
*аналіз сучасного стану проблеми, існуючих методів та засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обгрунтування системного наповнення, оцінка ергономічних та надійнісних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування*
*Висновки*

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| *Аналіз проблеми за темою роботи та постановка завдань дослідження* | *Пех П.А. ,доцент* | | |
| *Теоретичне дослідження та практична реалізація* | *Пех П.А. ,доцент* | | |
| *Практична реалізація об'єкта проектування* | *Пех П.А. ,доцент* | | |
| *Нормоконтроль* | *Багнюк Н.В., доцент* | | |
| *Гарант ОП* | *Лавренчук С.В., доцент* | | |
| *Показник запозичень тексту* | ***___%*** | | |
| *Академічна доброчесність* | *Міскевич О.І.,асистент* | | |

7. Дата видачі завдання 10.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | *Розділ 1. Огляд предметної області* | до 15.02.2024 р. | Виконано |
| 2. | *Розділ 2. Проектування архітектури інформаційної системи та розробка бази даних* | до 15.03.2024 р. | Виконано |
| 3. | *Розділ 3. Розробка та дослідження інформаційної системи з використанням засобів платформи ASP.NET CORE MVC* | до 04.05.2024 р. | Виконано |
| 4. | *Висновки та пропозиції* | до 07.05.2025 р. | Виконано |
| 5. | *Формування списку використаних джерел* | до 10.05.2024 р. | Виконано |
| 6. | *Формування додатків* | до 15.05.2024 р. | Виконано |
| 7. | *Оформлення ілюстративного матеріалу* | до 20.05.2024 р. | Виконано |
| 8. | *Нормоконтроль* | до 01.06.2024 р. | Виконано |
| 9. | *Інструментальна перевірка на академічний плагіат* | до 04.06.2024 р | Виконано |
| 10. | *Представлення кваліфікаційної роботи бакалавра до захисту* | до 11.06.2024 р. | Виконано |

**Здобувач вищої освіти** _____ Кузьмюк О.В.
(підпис) (прізвище, ініціали)

**Керівник кваліфікаційної роботи** _____ Пех П.А.
(підпис) (прізвище, ініціали)

# АНОТАЦІЯ

Кузьмюк О.В. Розробка інформаційної системи з використанням засобів платформи ASP.NET Core MVC.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2024. 75 с.

У роботі виконано розробку інформаційної системи, що базується на платформі ASP.NET Core MVC. Проаналізовано архітектуру платформи та її основні компоненти, такі як моделі, подання та контролери. Розглянуто методи інтеграції з базою даних та реалізації основних функцій системи, включаючи обробку запитів користувачів та забезпечення безпеки даних. Проведено тестування створеної системи для оцінки її продуктивності та надійності. Результати дослідження можуть бути корисні для розробників веб-додатків, які використовують платформу ASP.NET Core MVC, а також для студентів та викладачів, які вивчають сучасні технології веб-розробки.

Ключові слова: asp.net core mvc, інформаційна система, веб-розробка, база даних, безпека даних, продуктивність системи, тестування.

# ANNOTATION

Kuzmiuk O.V. Development of an information system using ASP.NET Core MVC platform tools.

Qualification work of the master’s degree program «Computer Engineering», specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2024.

The work deals with the development of an information system based on the ASP.NET Core MVC platform. The architecture of the platform and its main components, such as models, views and controllers, are analyzed. The methods of integration with the database and implementation of the main functions of the system, including processing user requests and ensuring data security, are considered. The created system was tested to evaluate its performance and reliability. The results of the study can be useful for web application developers using the ASP.NET Core MVC platform, as well as for students and teachers studying modern web development technologies.

Keywords: asp.net core mvc, information system, web development, database, data security, system performance, testing.

**ЗМІСТ**

# ВСТУП

Актуальність теми. В сучасному світі, де технології розвиваються швидкими темпами, важливість створення ефективних та надійних інформаційних систем набуває особливого значення. Інформаційні системи, побудовані на основі платформи ASP.NET Core MVC, є критичними для автоматизації бізнес-процесів, управління даними та забезпечення інтерактивної взаємодії користувачів з додатками. Зростання кількості цифрових даних, необхідність обробки великих масивів інформації, а також вимоги до високого рівня безпеки даних роблять розробку таких систем надзвичайно актуальною.

ASP.NET Core MVC як платформа для створення веб-додатків надає розробникам потужні інструменти для побудови масштабованих, продуктивних і безпечних рішень. Використання цієї платформи дозволяє ефективно вирішувати завдання з інтеграції з різноманітними базами даних, забезпечення стабільної роботи додатків під високими навантаженнями, а також реалізації сучасних підходів до розробки, таких як розподілені системи та хмарні технології. Крім того, ASP.NET Core MVC підтримує кросплатформенність, що дозволяє розробникам створювати рішення, які можуть працювати на різних операційних системах, включаючи Windows, macOS та Linux.

У епоху цифрової трансформації підприємства стикаються з необхідністю адаптації своїх бізнес-процесів до сучасних вимог ринку. Це включає впровадження інноваційних технологій, що сприяють підвищенню ефективності роботи, зменшенню операційних витрат і покращенню обслуговування клієнтів. Інформаційні системи, розроблені на базі ASP.NET Core MVC, можуть значно спростити ці процеси, забезпечуючи швидку та надійну обробку даних, зручний доступ до інформації в режимі реального часу та високу ступінь захисту даних.

Зростаюча кількість кіберзагроз та вимоги до захисту персональних даних підкреслюють важливість розробки інформаційних систем з акцентом на безпеку. ASP.NET Core MVC пропонує вбудовані засоби захисту, такі як автентифікація та авторизація, шифрування даних та захист від поширених веб-атак, що дозволяє розробникам забезпечувати високий рівень безпеки своїх

додатків.

Окрім технічних переваг, платформа ASP.NET Core MVC сприяє зменшенню часу та витрат на розробку завдяки своїй модульній архітектурі та можливості повторного використання коду. Це робить її привабливою для малого та середнього бізнесу, який прагне швидко та ефективно розгортати нові інформаційні системи без значних інвестицій у інфраструктуру.

Враховуючи ці фактори, тема розробки інформаційної системи з використанням засобів платформи ASP.NET Core MVC є надзвичайно актуальною. Вона не лише відповідає сучасним технологічним тенденціям, але й сприяє підвищенню продуктивності, безпеки та зручності в користуванні інформаційними системами, що є ключовими аспектами успішного функціонування сучасних підприємств.

Мета дослідження полягає у розробці та впровадженні ефективної інформаційної системи на базі платформи ASP.NET Core MVC, яка забезпечить високий рівень продуктивності, безпеки та зручності для користувачів. Дослідження спрямоване на вивчення архітектури платформи, методів інтеграції з базами даних та реалізації основних функцій системи, включаючи обробку користувацьких запитів і захист даних.

Об’єктом дослідження є процес розробки та впровадження інформаційних систем, що базуються на платформі ASP.NET Core MVC. Це включає в себе аналіз архітектури, компонентів системи, а також методів забезпечення продуктивності та безпеки.

Предметом дослідження є архітектура платформи ASP.NET Core MVC, методи інтеграції з базами даних, технології забезпечення безпеки інформаційних систем, а також практичні аспекти розробки, тестування та впровадження інформаційних систем з використанням цієї платформи.

Завдання дослідження:

– Проаналізувати теоретичні аспекти платформи ASP.NET Core MVC. Вивчити архітектуру та основні компоненти платформи ASP.NET Core MVC. Описати моделі, подання та контролери, які складають основу платформи.

– Дослідити методи інтеграції з базами даних. Проаналізувати різні підходи до інтеграції ASP.NET Core MVC з базами даних. Визначити оптимальні методи для забезпечення ефективного збереження та обробки даних.

– Розробити інформаційну систему. Створити проект інформаційної системи, яка буде реалізована на платформі ASP.NET Core MVC. Розробити та впровадити основні функціональні модулі системи, включаючи обробку користувацьких запитів та управління даними.

– Забезпечити безпеку інформаційної системи. Впровадити механізми автентифікації та авторизації користувачів. Реалізувати заходи захисту даних від поширених веб-загроз та атак.

– Протестувати та оцінити продуктивності системи. Провести комплексне тестування створеної інформаційної системи для виявлення помилок та недоліків. Оцінити продуктивність системи під різними навантаженнями.

– Задокументувати та проаналізувати результати. Підготувати технічну документацію, що описує структуру та функціональні можливості розробленої інформаційної системи. Проаналізувати отримані результати та зробити висновки щодо ефективності розробленої системи.

– Надати рекомендації щодо подальшого вдосконалення системи. Розробити рекомендації щодо можливих напрямків розвитку та вдосконалення інформаційної системи. Оцінити перспективи використання новітніх технологій для підвищення ефективності та функціональності системи.

Методи досліджень. Для вивчення та реалізації теми рекомендується використовувати комплексний підхід, що включає в себе ряд методів дослідження. Це включає літературний огляд, аналіз наукових публікацій та технічних документів з теми, що допоможе засвоїти найновіші тенденції та кращі практики у вибраній області. Далі, виконання емпіричних досліджень у формі створення прототипу інформаційної системи з використанням платформи ASP.NET Core MVC, який потрібно буде систематично тестувати та вдосконалювати. Паралельно, слід проводити аналіз вихідного коду схожих проектів, що допоможе засвоїти кращі практики та уникнути поширених

помилок. Не менш важливим є залучення експертів для отримання консультацій та оцінки розробленого прототипу. Завершальним етапом буде тестування користувацького досвіду, що дозволить оцінити зручність та ефективність використання системи з точки зору кінцевих користувачів. Такий комплексний підхід дозволить отримати всебічне розуміння обраної теми та ефективно реалізувати поставлені завдання дослідження.

Наукова новизна розробки інформаційної системи на базі платформи ASP.NET Core MVC є актуальним і важливим дослідженням, оскільки створення високопродуктивних, безпечних та зручних веб-додатків стає все більш вимогливим завданням у сучасному інформаційному середовищі. Науковою новизною є поєднання теоретичних знань з практичними дослідженнями з метою створення оптимальних та ефективних інформаційних систем, які відповідають вимогам сучасного ринку та користувачів.

Практичне значення отриманих результатів. Отримані результати дослідження мають велике практичне значення для розробників програмного забезпечення, інженерів та бізнес-спеціалістів. Розроблена інформаційна система може бути використана для створення різноманітних веб-додатків, від корпоративних порталів до інтернет-магазинів, що дозволить підприємствам та організаціям ефективно використовувати сучасні технології для автоматизації своєї діяльності та підвищення конкурентоспроможності. Крім того, результати дослідження можуть бути використані для навчання студентів та інших зацікавлених осіб у галузі веб-розробки та програмування, сприяючи підвищенню рівня їхніх знань та навичок у цій області.

Особистий внесок здобувача полягає в розробці та впровадженні інформаційної системи з використанням платформи ASP.NET Core MVC. Результатом цієї роботи є створення ефективного та функціонального веб-додатку, який може бути використаний у різних галузях та сферах діяльності. Розроблений додаток має потенціал сприяти автоматизації бізнес-процесів, полегшувати взаємодію з клієнтами та партнерами, а також підвищувати продуктивність роботи працівників. Крім того, здобувач також вніс значний

внесок у аналіз ринкових потреб та розробку інтерфейсу користувача, що сприяє покращенню користувацького досвіду та забезпечує зручність використання додатку. Розроблена інформаційна система відкриває нові можливості для бізнесу та допомагає досягти стратегічних цілей компанії завдяки своїм інноваційним технологіям та функціоналу.

Структура й обсяг роботи. Робота складається зі вступу, 3 розділів, висновків, списку використаних джерел із 35 найменувань. Повний обсяг роботи становить – 68 сторінок, 54 рисунки та 1 таблиці.

# РОЗДІЛ 1
# ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд платформи ASP.NET

«ASP.NET Core – це безкоштовна, відкрита та кросплатформна веб-платформа від Microsoft для розробки веб-сайтів та веб-застосунків. Вона пропонує широкий спектр функцій та інструментів для створення динамічних та масштабованих веб-рішень (рис. 1.1).


Рисунок 1. 1 – Логотип ASP.NET

Основні переваги ASP.NET Core:

– Легка та модульна ASP.NET Core має чітку структуру та проста у вивченні та використанні. Її модульна природа робить її гнучкою та розширюваною. Ви можете легко додавати нові функціональні можливості до свого застосунку, не впливаючи на існуючий код.

– Кросплатформна ASP.NET Core може працювати на Windows, macOS та Linux. Це робить її ідеальним вибором для розробки застосунків, які потрібно розгорнути в різних середовищах. Вам не потрібно турбуватися про те, щоб писати код, який є специфічним для певної платформи.

– Продуктивна Платформа оптимізована для високої продуктивності та масштабованості. Вона використовує сучасні технології та методи, щоб забезпечити швидке та надійне обслуговування запитів.

– Багата екосистема ASP.NET Core має велику та активну екосистему сторонніх бібліотек та інструментів, які можуть допомогти вам у розробці веб-застосунків. Ви можете знайти бібліотеки для практично будь-якої задачі, яку ви

можете собі уявити, від автентифікації та авторизації до керування контентом та електронної комерції.

Три основні компоненти ASP.NET Core:

– ASP.NET Core Web Forms – це платформа призначена для розробки веб-сайтів за допомогою drag-and-drop та подій. Вона підходить для створення простих веб-сайтів з динамічним контентом.

– ASP.NET Core MVC – це платформа ґрунтується на шаблоні проектування Model-View-Controller (MVC), який розділяє застосунок на три чітко визначені частини: модель, представлення та контролер. Це робить код більш модульним, керованим та тестованим. MVC добре підходить для створення складних веб-застосунків з чітко визначеною логікою бізнесу.

– ASP.NET Core Web API – це платформа використовується для створення RESTful API для веб-браузерів, мобільних пристроїв та інших клієнтів. API дозволяють іншим програмам отримувати доступ до даних та функціональності вашого веб-застосунку.

Додаткові можливості ASP.NET Core:

– Підтримка Razor Pages. Razor Pages – це легкий та динамічний підхід до розробки веб-сторінок без необхідності використовувати MVC. Вони добре підходять для створення простих сторінок з чітко визначеним контентом та логікою.

– SignalR дозволяє створювати веб-застосунки в режимі реального часу. Цей фреймворк використовується для створення чатів, онлайн-ігор та інших програм, які потребують постійного зв'язку між клієнтом та сервером.

– Entity Framework Core – це об'єктно-реляційний ORM (Object-Relational Mapper) для .NET. Він спрощує роботу з базами даних, дозволяючи вам працювати з ними за допомогою об'єктів C#.

Загальні принципи ASP.NET:

– Системне кешування ASP.NET використовує кешування для покращення продуктивності веб-застосунків. Це означає, що часто використовувані дані зберігаються в пам'яті, щоб їх не потрібно було щоразу

отримувати з бази даних або з файлової системи.

– ASP.NET компілює код C# на льоту в машинний код. Це робить код більш ефективним і покращує продуктивність веб-застосунків.

– ASP.NET Web Forms – це модель програмування, яка використовується для створення веб-сторінок з динамічним контентом. Web Forms використовують систему подій, яка дозволяє розробникам реагувати на дії користувачів.

– ASP.NET MVC – це модель програмування, яка ґрунтується на шаблоні проектування Model-View-Controller (MVC). MVC розділяє веб-застосунок на три частини: модель, представлення та контролер. Це робить код більш модульним, керованим та тестованим.

– ASP.NET Web API – це платформа для створення RESTful API. Web API дозволяють іншим програмам отримувати доступ до даних та функціональності веб-застосунку (рис 1.2).

Особливості ASP.NET:

– Легка та модульна ASP.NET – це легка та модульна платформа, що робить її простою у вивченні та використанні. Розробники можуть легко додавати нові функціональні можливості до своїх веб-застосунків, не впливаючи на існуючий код.

– Кросплатформна ASP.NET можна використовувати для розробки веб-застосунків, які працюють на Windows, macOS та Linux. Це робить її ідеальним вибором для розробки веб-застосунків, які потрібно розгорнути в різних середовищах.

– Продуктивна ASP.NET – це продуктивна платформа, яка оптимізована для високої продуктивності та масштабованості. Вона використовує сучасні технології та методи для забезпечення швидкого та надійного обслуговування запитів.

– Багата екосистема ASP.NET має велику та активну екосистему сторонніх бібліотек та інструментів. Це означає, що розробники можуть знайти бібліотеки для практично будь-якої задачі, яку вони можуть собі уявити» [1-4].

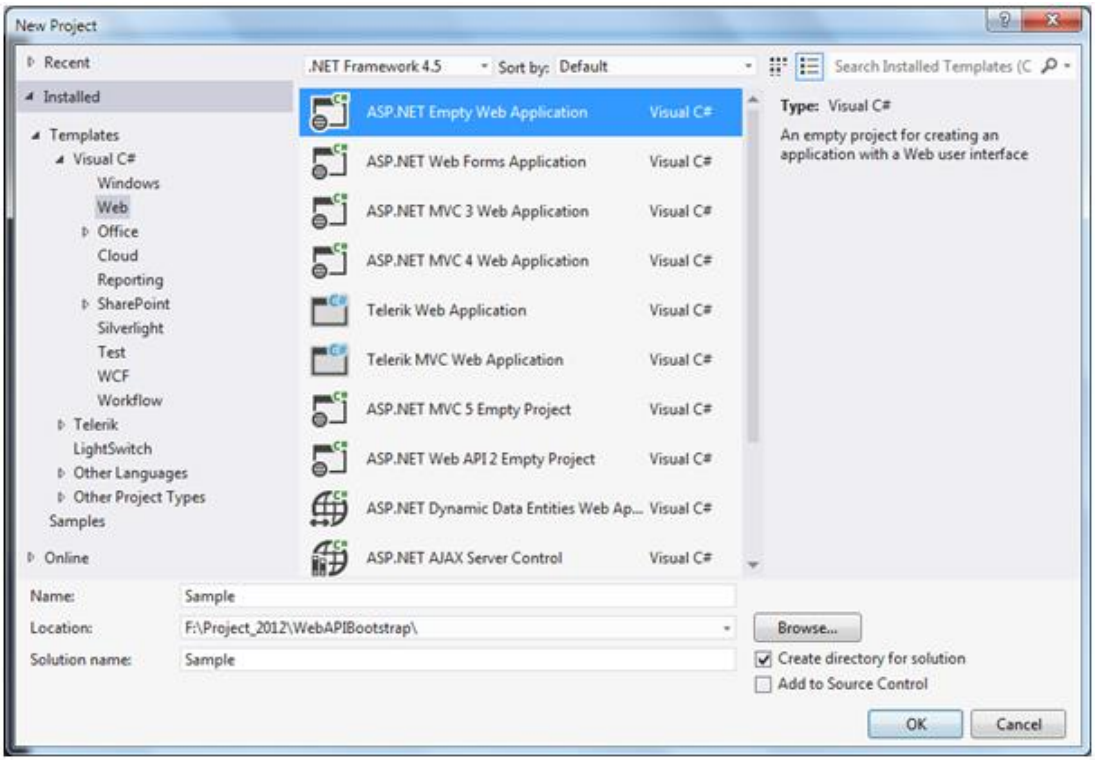



Рисунок 1. 2 – Різновиди ASP.NET

«У своєму проекті я буду вирокисторувати різновид ASP.NET Core MVC – бо це легка та модульна платформа для розробки веб-застосунків на основі .NET, яку підтримує Microsoft. Вона ґрунтується на шаблоні проектування Model-View-Controller (MVC), який розділяє застосунок на три чітко визначені частини:

– Model (Модель) – представляє дані вашого застосунку. Вона може містити класи, які представляють сутності вашої предметної області, а також репозиторії для доступу до даних.

– View (Представлення) – виводить дані моделі користувачеві. Це зазвичай файли HTML, які містять код Razor, який використовується для динамічного створення HTML-сторінок на основі даних моделі.

– Controller (Контролер) – посередник між моделлю та представленням. Він обробляє запити користувача, оновлює модель відповідно та вибирає відповідне представлення для відображення.

Загальні принципи ASP.NET Core MVC:

– Платформа має чітку структуру та проста у вивченні та використанні. Її модульна природа робить її гнучкою та розширюваною. Ви

можете легко додавати нові функціональні можливості до свого застосунку, не впливаючи на існуючий код.

– Кросплатформність ASP.NET Core MVC може працювати на різних операційних системах, включаючи Windows, macOS та Linux. Це робить його ідеальним для розробки застосунків, які потрібно розгорнути в різних середовищах. Вам не потрібно турбуватися про те, щоб писати код, який є специфічним для певної платформи.

– Платформа оптимізована для високої продуктивності та масштабованості. Вона використовує сучасні технології та методи, щоб забезпечити швидке та надійне обслуговування запитів.

– Багата екосистема ASP.NET Core MVC має велику та активну екосистему сторонніх бібліотек та інструментів, які можуть допомогти вам у розробці веб-застосунків. Ви можете знайти бібліотеки для практично будь-якої задачі, яку ви можете собі уявити, від автентифікації та авторизації до керування контентом та електронної комерції [35].

Особливості ASP.NET Core MVC:

– ASP.NET Core MVC використовує залежність від ін'єкції для спрощення тестування та керованості коду. Цей принцип дозволяє чітко визначити залежності між різними частинами вашого застосунку, що робить його більш модульним та простим у тестуванні.

– Підтримка Razor Page – це легкий та динамічний підхід до розробки веб-сторінок без необхідності використовувати MVC. Вони добре підходять для створення простих сторінок з чітко визначеним контентом та логікою. Razor Pages використовують код C# та HTML, що робить їх простими у вивченні та використанні.

– Підтримка маршрутизації ASP.NET Core MVC використовує систему маршрутизації для відповідності запитів користувачів контролерам та діям. Це дозволяє вам створювати чіткі та зрозумілі URL-адреси для вашого веб-застосунку.

– ASP.NET Core MVC використовує фільтри для перехоплення та

обробки запитів перед тим, як вони досягнуть контролера. Фільтри можуть використовуватися для різних цілей, таких як автентифікація, авторизація, кешування та ведення журналів.

– ASP.NET Core MVC використовує систему форматування для зв'язування даних моделі з елементами керування HTML. Це дозволяє вам легко створювати веб-форми, які автоматично оновлюються даними моделі.

Можливості розширення ASP.NET:

– ASP.NET пропонує широкий спектр можливостей розширення, які дозволяють розробникам адаптувати платформу до своїх конкретних потреб. До них належать:

– Модулі – це динамічно завантажувані бібліотеки, які розширюють функціональність ASP.NET. Ви можете створювати власні модулі або використовувати сторонні модулі, доступні в Інтернеті. Наприклад, ви можете використовувати модуль для додавання підтримки електронної комерції до свого веб-застосунку або модуль для покращення продуктивності вашого веб-застосунку.

– Провайдери – це компоненти, які реалізують певну функціональність ASP.NET, наприклад, кешування, автентифікацію або авторизацію. Ви можете створювати власні провайдери або використовувати сторонні провайдери, доступні в Інтернеті. Наприклад, ви можете використовувати провайдер кешування для кешування даних вашого веб-застосунку або провайдер автентифікації для автентифікації користувачів вашого веб-застосунку.

– Системи обробки подій дозволяють розробникам підписуватися на події, які виникають в ASP.NET, і реагувати на них. Це дозволяє розробникам розширювати функціональність ASP.NET без зміни базового коду платформи. Наприклад, ви можете підписатися на подію, яка виникає, коли користувач натискає кнопку на веб-сторінці, і виконати код, який обробляє цю подію.

– Налаштування конфігурації ASP.NET дозволяє розробникам налаштовувати різні аспекти платформи за допомогою файлів конфігурації. Це дозволяє розробникам адаптувати ASP.NET до своїх конкретних потреб.

Наприклад, ви можете налаштувати параметри кешування вашого веб-застосунку або налаштувати параметри автентифікації вашого веб-застосунку.

Інструменти розробки ASP.NET:

– Microsoft пропонує широкий спектр інструментів розробки для ASP.NET, які допомагають розробникам створювати, тестувати та розгортати веб-застосунки. До них належать:

– Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, яке підтримує ASP.NET. Visual Studio надає широкий спектр функцій, які допомагають розробникам створювати веб-застосунки ASP.NET, включаючи редактор коду, відладчик та інтегроване середовище розгортання. Visual Studio також пропонує безліч розширень та інструментів, які можуть допомогти розробникам ASP.NET, таких як розширення для створення веб-форм, розширення для тестування веб-застосунків та розширення для розгортання веб-застосунків.

– ASP.NET Web Forms Designer – це графічний інструмент, який допомагає розробникам створювати веб-форми ASP.NET. Web Forms Designer дозволяє розробникам перетягувати та кидати елементи керування на веб-форму та налаштовувати їхні властивості. Це може значно прискорити процес розробки веб-форм ASP.NET.

– ASP.NET MVC Scaffolding – це інструмент командного рядка, який допомагає розробникам швидко створювати код MVC для веб-застосунків ASP.NET MVC. Scaffolding може генерувати контролери, представлення та моделі для поширених веб-застосунків. Це може заощадити час розробникам, які хочуть швидко розпочати роботу з веб-застосунком ASP.NET MVC.

Найкращі практики використання технології ASP.NET:

– Шаблон проектування MVC – це добре зарекомендований шаблон проектування, який розділяє веб-застосунок на три частини: модель, представлення та контролер. Це робить код більш модульним, керованим та тестованим.

– Залежність від ін'єкції – це принцип програмування, який дозволяє

розробникам чітко визначити залежності між різними частинами веб-застосунку. Це робить код більш модульним та простим у тестуванні.

– ASP.NET пропонує різні можливості кешування, які можуть допомогти покращити продуктивність веб-застосунку. Кешування дозволяє веб-застосунку зберігати дані в пам'яті, щоб їх не потрібно було щоразу отримувати з бази даних або з файлової системи.

– ASP.NET дозволяє розробникам використовувати провайдери для реалізації певної функціональності, наприклад, кешування, автентифікації або авторизації. Це дозволяє розробникам адаптувати ASP.NET до своїх конкретних потреб.

– Системи обробки подій ASP.NET дозволяють розробникам підписуватися на події, які виникають в ASP.NET, і реагувати на них. Це дозволяє розробникам розширювати функціональність ASP.NET без зміни базового коду платформи.

– Microsoft пропонує широкий спектр інструментів розробки ASP.NET, які допомагають розробникам створювати, тестувати та розгортати веб-застосунки. Використання цих інструментів може допомогти розробникам писати кращий код, швидше створювати веб-застосунки та уникати поширених помилок.

– Найкраща практика використання ASP.NET, як і будь-якої іншої технології, полягає в написанні хорошого коду. Це означає писати код, який є чітким, лаконічним, простим для читання та тестування. Використання таких практик програмування, як SOLID та TDD, може допомогти розробникам писати кращий код ASP.NET.

– Важливо ретельно тестувати код ASP.NET. Це допоможе гарантувати, що код працює правильно і не містить помилок. ASP.NET пропонує різні можливості тестування, які можуть допомогти розробникам тестувати свій код.ASP.NET – це потужна та універсальна платформа для розробки веб-застосунків. Використання найкращих практик ASP.NET може допомогти розробникам створювати веб-застосунки, які є надійними, масштабованими та

безпечними» [5-7].

### 1.2 Технології та інструменти розробки інформаційних ресурсів

Існує безліч технологій, які можна використовувати для розробки інформаційних ресурсів. Вибір технології залежить від типу інформаційного ресурсу, який ви створюєте, вашої аудиторії та бюджету.

Ось деякі з найпоширеніших технологій:

Різновиди розробки веб-сайтів:

– Статичні веб-сайти – це веб-сайти складаються з фіксованих сторінок HTML, CSS та JavaScript. Вони прості у створенні та обслуговуванні, але не пропонують багато інтерактивності (рис 1.3).

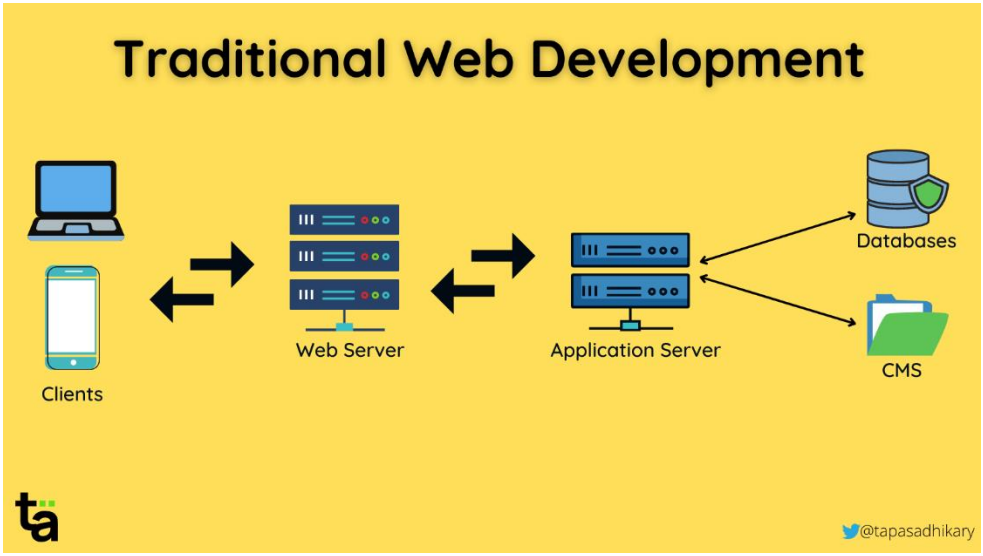



Рисунок 1. 3 – Статичні веб-сайти

– Динамічні веб-сайти – це веб-сайти генеруються на сервері у відповідь на запит користувача. Вони можуть бути більш інтерактивними та динамічними, ніж статичні веб-сайти, але складніші у розробці та обслуговуванні (рис. 1.4).

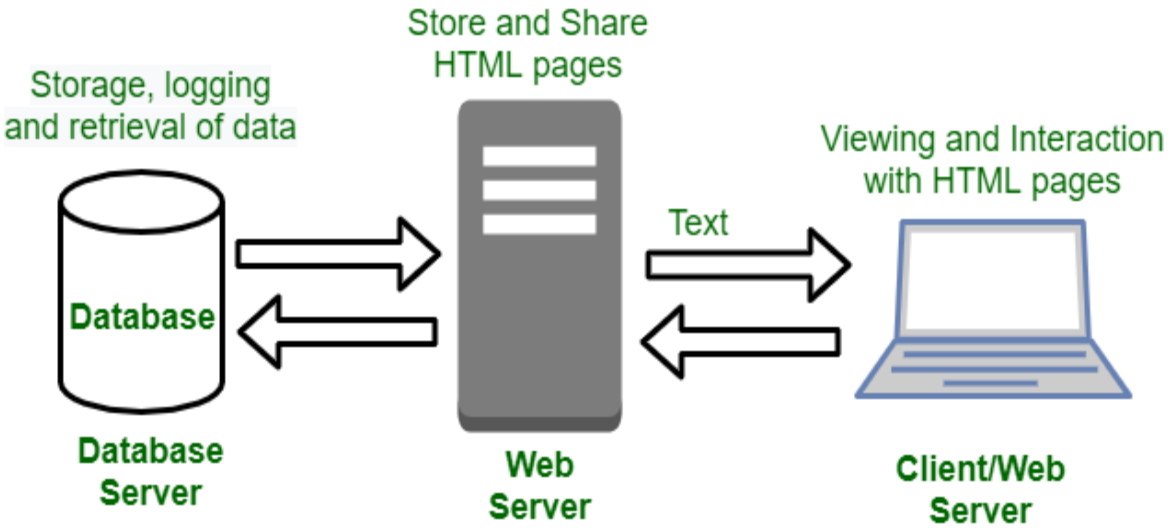



Рисунок 1. 4 – Динамічні веб-сайти

Різновиди розробки веб-застосунків:

– Односторінкові веб-застосунки (SPA) – це веб-застосунки, які завантажують лише одну HTML-сторінку, а потім динамічно оновлюють її вміст за допомогою JavaScript. Вони пропонують плавний користувацький досвід, схожий на мобільний застосунок (рис 1.5).

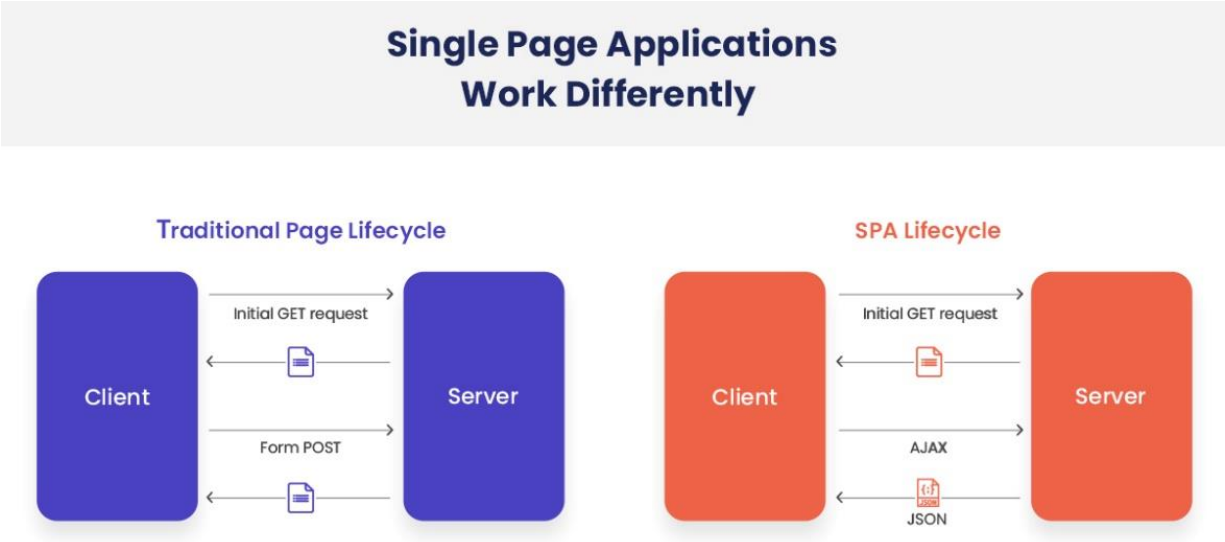



Рисунок 1. 5 – Односторінкові веб-застосунки

– Багатосторінкові веб-застосунки (MPA) – це веб-застосунки складаються з кількох HTML-сторінок, які пов'язані між собою. Вони складніші у розробці, ніж SPA, але можуть бути більш потужними та масштабованими.

Різновиди розробки мобільних застосунків:

– Нативні мобільні застосунки – це застосунки розроблені для певної платформи, такої як iOS або Android. Вони пропонують найкращу продуктивність та користувацький досвід, але можуть бути дорожчими у розробці [34].

– Крос-платформні мобільні застосунки – це застосунки розроблені один раз, а потім компілюються для різних платформ. Вони можуть бути більш економічно ефективними, ніж нативні застосунки, але можуть не пропонувати такий самий рівень продуктивності або користувацького досвіду.

Різновиди розробки настільних програм:

– Традиційні настільні програми – ці програми встановлюються на комп'ютер користувача і запускаються з локального жорсткого диска. Вони пропонують високий рівень контролю та продуктивності, але можуть бути складними у розгортанні та оновленні.

– Веб-застосунки для настільних комп'ютерів – ці веб-застосунки розроблені для роботи в браузері настільного комп'ютера. Вони пропонують зручність використання веб-застосунків, але можуть не мати такого ж рівня продуктивності або функціональності, як традиційні настільні програми.

ASP.NET Core MVC – це потужний фреймворк для розробки веб-застосунків, який пропонує безліч бібліотек та інструментів, які допомагають розробникам створювати динамічні, масштабовані та безпечні веб-застосунки. Ось оновлений опис деяких з найпоширеніших бібліотек та інструментів.

Бібліотеки на основі Entity Framework Core:

– Entity Framework Core (EF Core) – це об'єктно-реляційний провайдер (ORM), який полегшує роботу з базами даних. Він дозволяє розробникам писати код C#, який відображає структуру бази даних, а не писати SQL-запити. Це може значно спростити розробку веб-застосунків, які потребують доступу до бази

даних.

– EF Core підтримує широкий спектр баз даних, включаючи SQL Server, MySQL, PostgreSQL та SQLite. Він також пропонує безліч функцій, які роблять роботу з базами даних більш простою та ефективною, таких як міграції баз даних, відстеження змін та лениве завантаження.

Бібліотеки на основі ASP.NET Core Identity:

– ASP.NET Core Identity – це система автентифікації та авторизації, яка дозволяє розробникам легко додавати автентифікацію користувачів та авторизацію ролей до своїх веб-застосунків. Вона підтримує різні режими автентифікації, такі як автентифікація на основі форм, автентифікація на основі OAuth та автентифікація на основі Windows.

– ASP.NET Core Identity також пропонує безліч функцій, які роблять управління користувачами більш простим, таких як керування ролями, скидання паролів та генерування токенів доступу.

Бібліотеки на основі ASP.NET Core Razor Pages:

– ASP.NET Core Razor Pages – це альтернативний підхід до розробки веб-сторінок ASP.NET Core MVC, який використовує Razor Pages для створення сторінок HTML з динамічним вмістом. Razor Pages полегшують створення простих веб-сторінок без необхідності писати складний код MVC.

– Razor Pages також добре підходять для створення веб-застосунків з однією сторінкою (SPA), які завантажують лише одну HTML-сторінку, а потім динамічно оновлюють її вміст за допомогою JavaScript.

Бібліотеки на основі ASP.NET Core SignalR:

– ASP.NET Core SignalR – це бібліотека для розробки веб-застосунків у реальному часі. Вона дозволяє розробникам створювати веб-сайти та веб-застосунки, які можуть спілкуватися з клієнтами в режимі реального часу. Це може бути корисно для створення таких функцій, як чат, оновлення в реальному часі та спільні роботи.

– SignalR підтримує різні транспортні протоколи, включаючи WebSocket, Server-Sent Events та Long Polling. Він також пропонує безліч

функцій, які роблять розробку веб-застосунків у реальному часі більш простою, таких як групи, канали та масштабування.

Бібліотеки на основі ASP.NET Core Dependency Injection:

– ASP.NET Core Dependency Injection (DI) – це система залежностей, яка дозволяє розробникам писати більш модульний та керований код. Вона дозволяє розробникам чітко визначити залежності між різними частинами своїх веб-застосунків. Це може зробити код більш простим у читанні, тестуванні та обслуговуванні.

– ASP.NET Core DI підтримує різні стилі ін'єкції залежностей, включаючи конструктор, властивість та метод ін'єкції. Він також пропонує безліч функцій, які роблять DI більш потужним, таких як реєстрація типу та інстанції, а також життєвий цикл об'єкта.

ASP.NET Core MVC пропонує широкий спектр інструментів, які допомагають розробникам створювати, тестувати та розгортати веб-застосунки. Ці інструменти можна розділити на три основні категорії:

Інструменти для розробки:

– Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, яке підтримує ASP.NET Core. Visual Studio надає широкий спектр функцій, які допомагають розробникам створювати веб-застосунки ASP.NET, включаючи: редактор коду Visual Studio підтримує синтаксис підсвічування, автозаповнення коду та інші функції, які допомагають розробникам писати код швидше та ефективніше, відладчик Visual Studio дозволяє розробникам крок за кроком виконувати код їхніх веб-застосунків, встановлювати точки зупинки та переглядати значення змінних, Visual Studio надає інтегроване середовище розгортання, яке дозволяє розробникам легко розгортати свої веб-застосунки на локальному сервері або в хмарі (рис. 1.6)

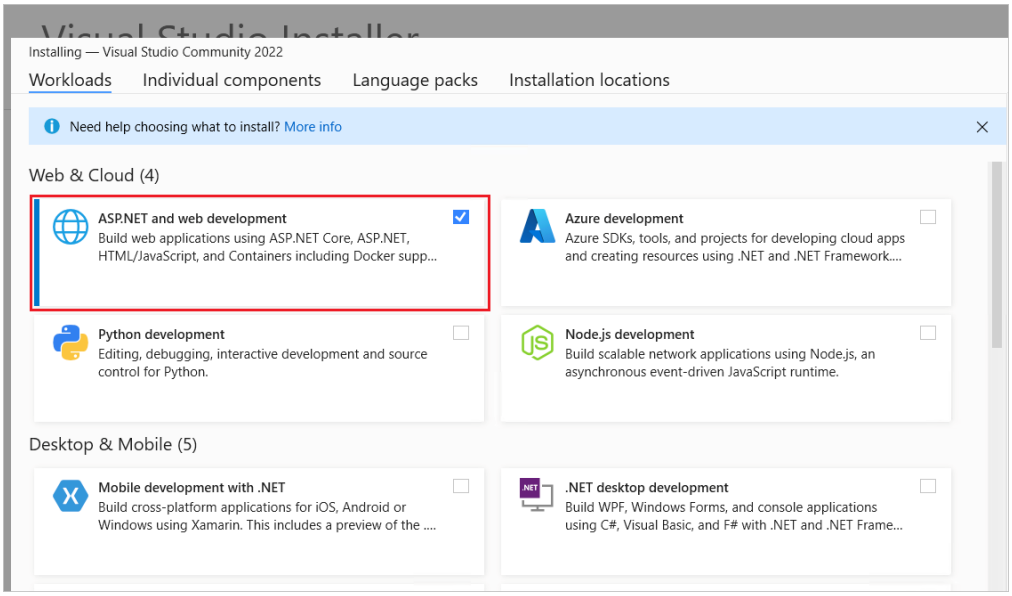



Рисунок 1. 6 – Visual Studio ASP.NET середовище

– ASP.NET Core Web Forms Designer – це графічний інструмент, який допомагає розробникам створювати веб-форми ASP.NET Core. Web Forms Designer дозволяє розробникам перетягувати та кидати елементи керування на веб-форму, налаштовувати властивості елементів керування, переглядати макет веб-форми в режимі реального часу.

– ASP.NET Core MVC Scaffolding – це інструмент командного рядка, який допомагає розробникам швидко створювати код MVC для веб-застосунків ASP.NET Core MVC. Scaffolding може генерувати: контролери, представлення, моделі та код для доступу до даних.

– NuGet – це менеджер пакетів для .NET Framework, який дозволяє розробникам легко знаходити, встановлювати та оновлювати сторонні бібліотеки. NuGet пропонує широкий спектр пакетів для ASP.NET Core, які можуть допомогти розробникам: розширити функціональність своїх веб-застосунків, використовувати популярні бібліотеки та фреймворки, залишатися в курсі останніх оновлень безпеки.

Інструменти тестування:

– ASP.NET Core Unit Tests – це фреймворк для тестування одиниць, який допомагає розробникам тестувати код своїх веб-застосунків ASP.NET Core. Unit Tests дозволяє розробникам: тестувати невеликі частини коду, не

запускаючи весь веб-застосунок, гарантувати, що код працює правильно, покращити якість коду.

– ASP.NET Core Integration Tests – це фреймворк для інтеграційного тестування, який допомагає розробникам тестувати, як різні частини веб-застосунку ASP.NET Core працюють разом. Integration Tests дозволяє розробникам: тестувати взаємодії між контролерами, представленнями та моделями, гарантувати, що веб-застосунок працює правильно як ціле, знаходити та виправляти помилки інтеграції.

– ASP.NET Core End-to-End Tests – це фреймворк для тестування кінцевого користувача, який допомагає розробникам тестувати, як веб-застосунок ASP.NET Core працює.

Інструменти розгортання:

– IIS (Internet Information Services) – це веб-сервер від Microsoft, який підтримує ASP.NET Core. IIS – це популярний вибір для розгортання веб-застосунків ASP.NET Core в середовищі Windows.

– Kestrel – це веб-сервер з відкритим кодом, який використовується для розробки та розгортання ASP.NET Core. Kestrel є легким і продуктивним веб-сервером, який можна використовувати в різних середовищах, включаючи Windows, Linux і macOS (рис. 1.7).

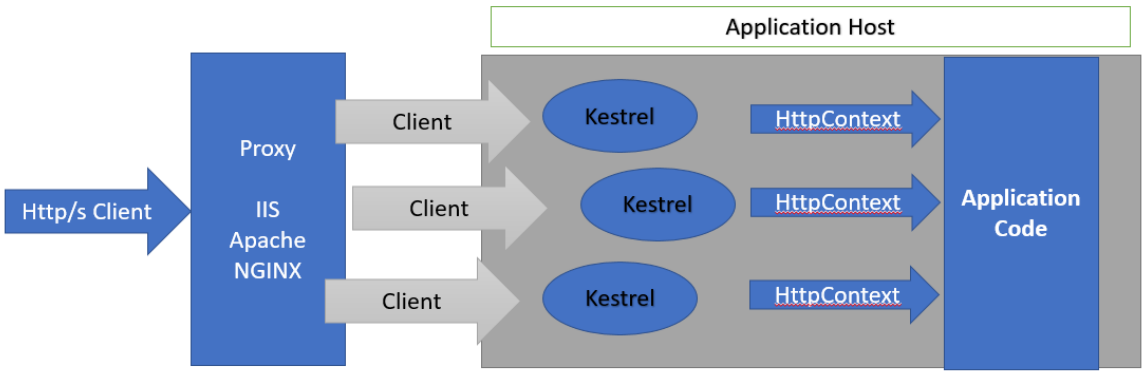



Рисунок 1. 7 – Інструмент розгортання Kestrel

– Azure – це хмарна платформа від Microsoft, яка пропонує різні послуги для розгортання та обслуговування веб-застосунків ASP.NET Core.

Azure пропонує такі послуги, як Azure App Service – це платформа як послуга (PaaS), яка дозволяє розробникам легко розгортати та обслуговувати веб-застосунки ASP.NET Core в хмарі. Azure Functions – це безсерверна платформа, яка дозволяє розробникам запускати код ASP.NET Core у відповідь на події, такі як HTTP-запити, зміни даних у базі даних або повідомлення в черзі.

## 1.3 Порівняння платформи ASP.NET Core MVC з іншими аналогічними рішеннями

ASP.NET Core MVC – це популярна платформа для розробки веб-додатків, яка пропонує безліч функцій та переваг. Однак існують й інші аналогічні рішення, які можуть бути кращим вибором для деяких розробників (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз ASP.NET з альтернативами

| Функція | ASP.NET Core MVC | Ruby on Rails | Python Django | Node.js with Express |
|---|---|---|---|---|
| Мова програмування | C# | Ruby | Python | JavaScript |
| Крива навчання | Середня | Низька | Середня | Низька |
| Продуктивність | Висока | Середня | Середня | Низька |
| Масштабованість | Висока | Середня | Середня | Висока |
| Спільнота | Велика та активна | Велика та активна | Велика та активна | Велика та активна |
| Вартість | Безкоштовна | Безкоштовна | Безкоштовна | Безкоштовна |

Детальний опис платформи ASP.NET Core MVC:

– ASP.NET Core MVC – це фреймворк веб-застосунків з відкритим кодом на основі .NET, розроблений Microsoft. Він є наступником ASP.NET MVC і пропонує ряд нових функцій та вдосконалень, включаючи: Підтримку .NET Core, яка є крос-платформною, що означає, що веб-застосунки ASP.NET Core MVC можна запускати на Windows, Linux і macOS. Підтримку Dependency

Injection, яка є фреймворком для побудови компонентно-орієнтованих веб-застосунків. Підтримку Razor Pages, які є способом створення веб-сторінок без використання MVC.

- ASP.NET Core MVC – це потужний і гнучкий фреймворк, який можна використовувати для створення широкого спектра веб-застосунків, від простих веб-сайтів до складних веб-застосунків з єдиним входом.
- ASP.NET Core MVC має велику та активну спільноту, що означає, що вам буде легко знайти допомогу, якщо вам це знадобиться.

Детальний опис платформи Ruby on Rails:

- Ruby on Rails – це фреймворк веб-застосунків на Ruby, який відомий своєю простотою використання та швидкістю розробки. Rails має низьку криву навчання, що робить його хорошим вибором для початківців. Rails також пропонує високий рівень продуктивності та масштабованості.
- Rails використовує концепцію Convention over Configuration, що означає, що багато спільних завдань розробки веб-застосунків виконуються автоматично, без необхідності писати багато коду. Це може зробити розробку веб-застосунків Rails швидшою та простішою.
- Rails має велику та активну спільноту, що означає, що вам буде легко знайти допомогу, якщо вам це знадобиться.

Детальний опис платформи Python Django:

- Python Django – це фреймворк веб-застосунків на Python, який відомий своєю елегантністю та гнучкістю. Django має трохи вищу криву навчання, ніж Rails, але він пропонує ширший спектр функцій та можливостей. Django також є дуже продуктивним і масштабованим фреймворком.
- Django використовує архітектуру Model-View-Template (MVT), яка є поширеною архітектурою для веб-застосунків. MVT робить код Django чітким і простим для розуміння.
- Django має велику та активну спільноту, що означає, що вам буде легко знайти допомогу, якщо вам це знадобиться.

Детальний опис платформи Node.js with Express:

- Node.js – це платформа JavaScript з відкритим кодом, яка дозволяє розробникам писати серверний код JavaScript. Express – це популярний фреймворк веб-застосу нків для Node.js, який пропонує простий і гнучкий спосіб створення веб-застосунків.
- Node.js – це подієво-орієнтована платформа, що робить її дуже масштабованою та ефективною. Node.js також є дуже гнучкою платформою, що робить її хорошим вибором для розробки веб-застосунків, які потребують високої продуктивності та масштабованості.
- Node.js має велику та активну спільноту, що означає, що вам буде легко знайти допомогу, якщо вам це знадобиться.

Ось деякі додаткові фактори, які слід врахувати при виборі платформи для розробки веб-застосунку:

- Досвід розробки. Якщо ви досвідчений розробник, ви можете вибрати більш складну платформу, таку як ASP.NET Core MVC або Python Django. Якщо ви початківець, ви можете вибрати простішу платформу, таку як Ruby on Rails або Node.js з Express.
- Вимоги до продуктивності. Якщо вам потрібен веб-застосунок з високою продуктивністю, вам слід вибрати платформу, яка відома своєю продуктивністю, таку як ASP.NET Core MVC або Node.js.
- Вимоги до масштабованості. Якщо вам потрібен веб-застосунок, який може масштабуватися для обслуговування великої кількості користувачів, вам слід вибрати платформу, яка відома своєю масштабованістю, таку як ASP.NET Core MVC або Node.js.
- Особисті уподобання. Зрештою, найкраща платформа для розробки веб-застосунку – це та, з якою вам найзручніше працювати.

Найкраща платформа для розробки веб-застосунку залежить від конкретних потреб. Якщо ви шукаєте потужний і гнучкий фреймворк з великою та активною спільнотою, ASP.NET Core MVC – це хороший вибір. Якщо ви шукаєте фреймворк, який простий у використанні та швидкий у розробці, Ruby

on Rails або Node.js з Express можуть бути кращим вибором. Якщо ви шукаєте елегантний і гнучкий фреймворк, Python Django може бути кращим вибором.

# РОЗДІЛ 2
# ПРОЕКТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА РОЗРОБКА БАЗИ ДАНИХ

## 2.1 Архітектура інформаційної системи

«Архітектура інформаційної системи, розробленої на платформі ASP.NET Core MVC, базується на трирівневій архітектурі, що складається з наступних основних компонентів: презентаційний шар (UI), бізнес-логіка (BLL) та шар доступу до даних (DAL). Кожен з цих компонентів виконує свою унікальну роль у системі, що сприяє підвищенню її гнучкості, масштабованості та зручності обслуговування.

Презентаційний шар(UI) відповідає за взаємодію з користувачем та відображення даних. У контексті ASP.NET Core MVC, цей шар представлений контролерами та представленнями (Views). Контролери обробляють запити від користувачів, викликають відповідні методи бізнес-логіки, а потім повертають результати у вигляді представлень. Представлення, у свою чергу, містять HTML-шаблони, що відображають дані, отримані від контролерів.

Бізнес-логіка (BLL), шар бізнес-логіки є серцем інформаційної системи, де зосереджені всі бізнес-правила та логіка обробки даних. Він включає в себе сервіси та менеджери, які виконують операції з даними, отриманими від шару доступу до даних, та підготовляють їх для відображення у презентаційному шарі. Цей шар дозволяє забезпечити модульність та повторне використання коду, а також спрощує тестування та обслуговування системи [33].

Шар доступу до даних (DAL), шар доступу до даних відповідає за взаємодію з базою даних. У ASP.NET Core MVC, цей шар зазвичай реалізується за допомогою Entity Framework Core або інших ORM (Object-Relational Mapping) бібліотек. Він містить репозиторії, які виконують CRUD-операції (створення, читання, оновлення, видалення) з даними. Така структура дозволяє ізолювати бізнес-логіку від деталей зберігання даних, що робить систему більш гнучкою та легкою в обслуговуванні.

Інтеграція компонентів – це взаємодія між цими трьома шарами

здійснюється через добре визначені інтерфейси та залежності. Це забезпечує чітке розмежування обов'язків та сприяє підтримці принципу єдиного відповідальності. Такий підхід дозволяє легко оновлювати або замінювати окремі компоненти системи без значних змін у решті коду.

Діаграма архітектури, для наочності, загальну архітектуру інформаційної системи можна зобразити у вигляді діаграми, яка показує взаємозв'язки між різними компонентами та потоками даних. Це допоможе краще зрозуміти структуру та функціонування системи.

Узагальнюючи, архітектура інформаційної системи на базі ASP.NET Core MVC забезпечує високу гнучкість, масштабованість та зручність обслуговування, що є критично важливими для сучасних веб-додатків. Розділення на презентаційний шар, бізнес-логіку та шар доступу до даних дозволяє ефективно управляти складністю системи та забезпечити її надійність та ефективність» [8-12].

## 2.2 Технології та інструменти розробки інформаційної системи

«Розробка інформаційної системи на базі платформи ASP.NET Core MVC вимагає використання сучасних технологій та інструментів, які забезпечують ефективність, надійність та зручність у розробці та підтримці веб-додатків. У цьому підрозділі розглянуто основні технології та інструменти, які використовуються для створення даної інформаційної системи.

ASP.NET Core MVC є основним фреймворком для створення веб-додатків на платформі .NET. Він поєднує в собі переваги архітектури Model-View-Controller (MVC), що дозволяє розробникам чітко розділити бізнес-логіку, дані та представлення, що сприяє покращенню модульності, тестованості та підтримки коду. ASP.NET Core MVC підтримує крос-платформенність, тобто додатки можна запускати на Windows, Linux та macOS. Крім того, ASP.NET Core забезпечує високу продуктивність і підтримує сучасні веб-стандарти, що дозволяє створювати швидкі та надійні веб-додатки.

Entity Framework Core (EF Core) – це ORM (Object-Relational Mapping) бібліотека для .NET, яка дозволяє працювати з базами даних, використовуючи об'єктно-орієнтовані моделі. EF Core забезпечує зручний та ефективний доступ до даних, підтримуючи різні СУБД, включаючи SQL Server, MySQL, PostgreSQL та інші. За допомогою EF Core можна виконувати CRUD-операції (створення, читання, оновлення, видалення), а також складні запити до бази даних, не пишучи SQL-коду. EF Core також підтримує міграції бази даних, що дозволяє легко керувати змінами у структурі даних під час розвитку проекту.

Для зберігання та управління даними використовується реляційна база даних Microsoft SQL Server. Це потужна та надійна СУБД, яка забезпечує високу продуктивність та масштабованість, а також має велику кількість інструментів для адміністрування та аналізу даних. SQL Server підтримує розширені можливості безпеки, резервного копіювання та відновлення даних, що робить його ідеальним вибором для корпоративних додатків [32].

Основним середовищем розробки для створення ASP.NET Core MVC додатків є Visual Studio. Це інтегроване середовище розробки (IDE), яке надає широкий спектр інструментів для написання, тестування та налагодження коду. Visual Studio підтримує роботу з різними мовами програмування, включаючи C#, та забезпечує інтеграцію з системами контролю версій, такими як Git. Visual Studio також пропонує потужні засоби для роботи з базами даних, веб-розробки та хмарними технологіями.

Для контролю версій та спільної роботи над проектом використовуються Git та GitHub. Git – це система розподіленого контролю версій, яка дозволяє відслідковувати зміни у коді та працювати над ним у команді. GitHub є хостингом для Git-репозиторіїв, який також надає інструменти для управління проектами, виявлення помилок та організації командної роботи. Використання GitHub також дозволяє автоматизувати процеси інтеграції та розгортання додатків за допомогою GitHub Actions.

Для створення адаптивного та зручного інтерфейсу користувача використовується фреймворк Bootstrap. Він забезпечує готові компоненти та

стилі для швидкого створення сучасних веб-сторінок, які добре виглядають на різних пристроях. Bootstrap дозволяє розробникам швидко створювати макети сторінок з використанням готових CSS-класів та компонентів, що значно прискорює процес розробки інтерфейсу.

jQuery – це популярна JavaScript-бібліотека, яка спрощує роботу з DOM, обробку подій, анімацію та AJAX-запити. Використання jQuery дозволяє значно скоротити кількість коду та підвищити його читабельність. jQuery також має велику екосистему плагінів, що дозволяє легко додавати функціональність до веб-додатків.

Для автоматизації процесів розгортання та тестування додатків використовується платформа Azure DevOps. Вона забезпечує безперервну інтеграцію (CI) та безперервне розгортання (CD), що дозволяє автоматично тестувати та розгортати додатки на різних середовищах. Azure DevOps також пропонує інструменти для управління проектами, планування робіт та відстеження помилок.

Додаткові інструменти:

Swagger – це інструмент для автоматичної генерації документації API, що полегшує тестування та інтеграцію з іншими системами.

Postman – це інструмент для тестування API, який дозволяє розробникам відправляти запити до сервера та аналізувати відповіді.

Docker – це платформа для контейнеризації додатків, що дозволяє створювати легкі, портативні та самодостатні контейнери для запуску додатків.

Узагальнюючи, використання сучасних технологій та інструментів у розробці інформаційної системи на базі ASP.NET Core MVC дозволяє створити ефективний, надійний та зручний у підтримці веб-додаток, який відповідає вимогам сучасних користувачів та бізнесу. Інтеграція цих технологій та інструментів забезпечує високу продуктивність, масштабованість та безпеку розробленої системи» [13-20].

## 2.3 Процес розробки та впровадження інформаційної системи

Процес розробки та впровадження інформаційної системи на базі платформи ASP.NET Core MVC включає кілька ключових етапів, кожен з яких забезпечує поступове досягнення кінцевого результату – створення ефективної, надійної та зручної у використанні системи. У цьому підрозділі детально розглянуто основні етапи процесу розробки та впровадження інформаційної системи.

Першим кроком у розробці інформаційної системи є визначення вимог до системи. Це включає аналіз потреб користувачів, визначення функціональних та нефункціональних вимог, а також створення технічного завдання. На цьому етапі активно залучаються замовники та кінцеві користувачі, щоб забезпечити точне розуміння того, які задачі повинна вирішувати система та які характеристики вона повинна мати.

На етапі проектування створюється архітектура інформаційної системи, визначаються її основні компоненти та їх взаємодія. Проектування включає розробку схем бази даних, визначення структур даних, моделювання основних процесів та визначення інтерфейсів користувача. Важливим аспектом цього етапу є створення прототипів інтерфейсів та моделювання користувацьких сценаріїв.

Етап розробки включає написання коду для всіх компонентів системи відповідно до проектної документації. Використовуючи ASP.NET Core MVC, розробники створюють контролери, моделі та представлення, які відповідають за обробку запитів, маніпуляцію даними та відображення інформації користувачам. У процесі розробки використовуються сучасні інструменти та технології, такі як Visual Studio, Entity Framework Core та інші.

Після розробки основних компонентів системи проводиться її тестування. Тестування включає різні види тестів: юніт-тести для перевірки окремих модулів, інтеграційні тести для перевірки взаємодії між компонентами, функціональні тести для перевірки відповідності вимогам та користувацькі тести

для оцінки зручності використання системи. Автоматизоване тестування виконується за допомогою спеціалізованих інструментів, що дозволяє швидко виявляти та виправляти помилки.

Після успішного завершення тестування система готова до впровадження. Впровадження включає розгортання додатку на продуктивному сервері, налаштування бази даних та інших сервісів, а також навчання користувачів. Важливим аспектом цього етапу є забезпечення безперервності бізнес-процесів під час переходу на нову систему.

Після впровадження система потребує постійної підтримки та розвитку. Це включає моніторинг продуктивності, вирішення проблем, які можуть виникати під час експлуатації, а також впровадження нових функцій відповідно до змінних вимог користувачів та бізнесу. Регулярні оновлення та покращення системи забезпечують її актуальність та ефективність [21-25].

Інструменти та методи, використані у процесі розробки:

Гнучкі методології розробки, такі як Agile та Scrum, забезпечують ітеративний підхід до розробки, дозволяючи адаптуватися до змінних вимог та забезпечуючи постійний зворотний зв'язок з користувачами.

Безперервна інтеграція та безперервне розгортання (CI/CD) з використанням Azure DevOps забезпечують автоматизацію процесів тестування та розгортання, що підвищує якість та швидкість випуску нових версій системи.

Регулярні перевірки коду (Code Review) сприяють покращенню якості коду, виявленню помилок та забезпеченню відповідності стандартам кодування.

Автоматизовані юніт-тести забезпечують перевірку окремих модулів системи, що дозволяє швидко виявляти та виправляти помилки на ранніх етапах розробки.

Використання контейнеризації з Docker спрощує розгортання та забезпечує консистентність середовища розробки та продуктивного середовища.

Узагальнюючи, процес розробки та впровадження інформаційної системи на базі ASP.NET Core MVC включає кілька ключових етапів, кожен з яких спрямований на створення ефективної, надійної та зручної у використанні

системи. Використання сучасних технологій та інструментів забезпечує високу якість та продуктивність розробленої системи, а також дозволяє швидко реагувати на зміни вимог та забезпечувати безперервний розвиток системи [26-30].

# РОЗДІЛ 3
# РОЗРОБКА ТА ДОСЛІДЖЕННЯ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ ЗАСОБІВ ПЛАТФОРМИ ASP.NET CORE MVC

## 3.1 Етапи розробки інформаційної системи

Розпочинаємо розробку інформаційної системи із запуску нашого інтегроване середовище розробки (IDE), таке як Visual Studio, запустивши бачимо вікно Visual Studio 2022 та перелік можливостей, а саме Clone a repository, Open a project or solution, Open a local folder, Create a new project (рис. 3.1) [31].

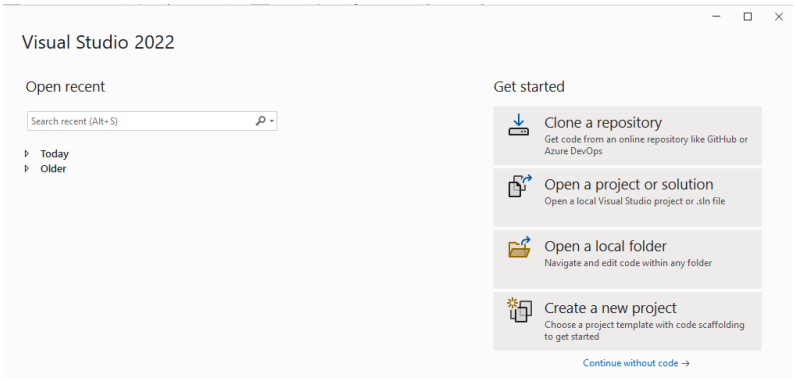



Рисунок 3. 1 – Стартове меню Microsoft Visual Studio

Потім обираємо опцію "Create a new project" у відповідному меню з можливими варіаціями (рис. 3.2).

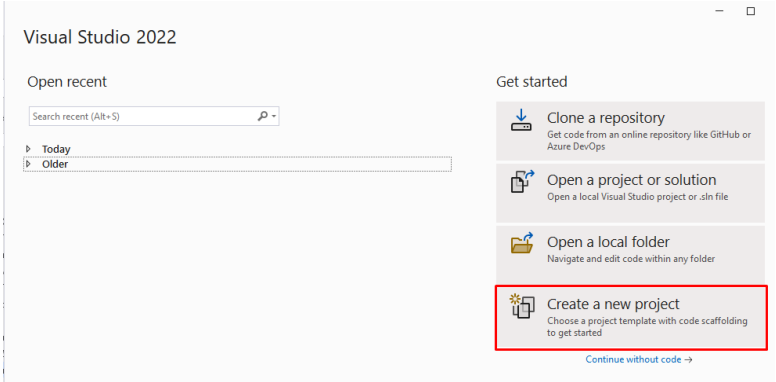



Рисунок 3. 2 – Створення нового проекту

Після натискання у головному вікні на пункт "Create a new project”

обираємо шаблон ASP.NET Core Web App (Model-View-Controller) і натискаємо «Next» (рис. 3.3).

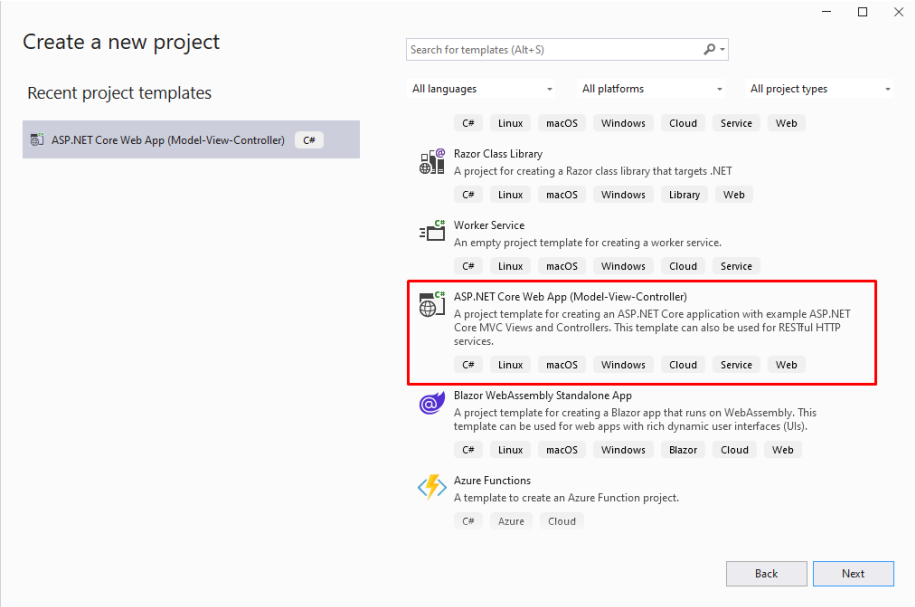



Рисунок 3. 3 – Обрання потрібного шаблону для проекту

Обираємо місце збереження проекту та переходимо далі, в меню конфігурацій обираємо версію Framework'у та створюємо проект (рис. 3.4).

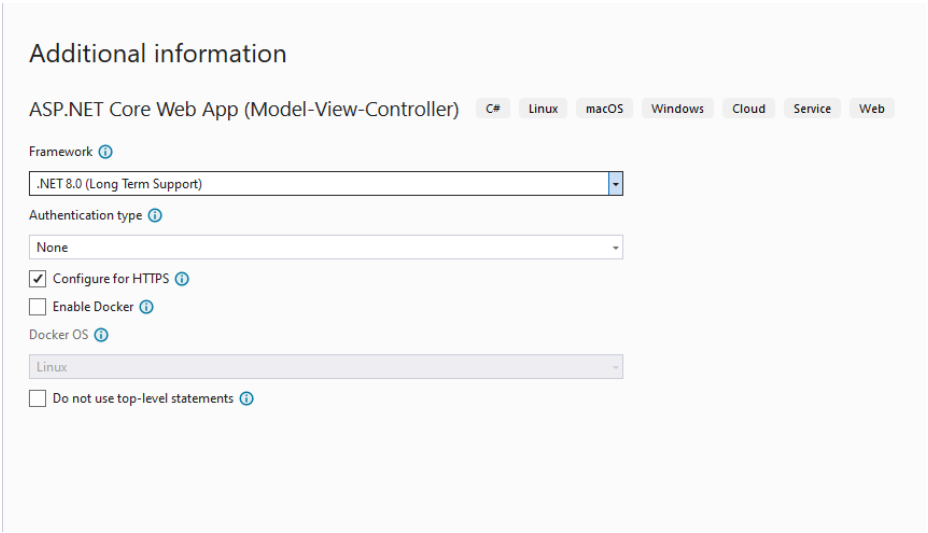



Рисунок 3.4 – Конфігурація проекту

Після створення проекту ми можемо побачити основні папки проекту з котрими в подальшому будемо працювати (рис. 3.5).

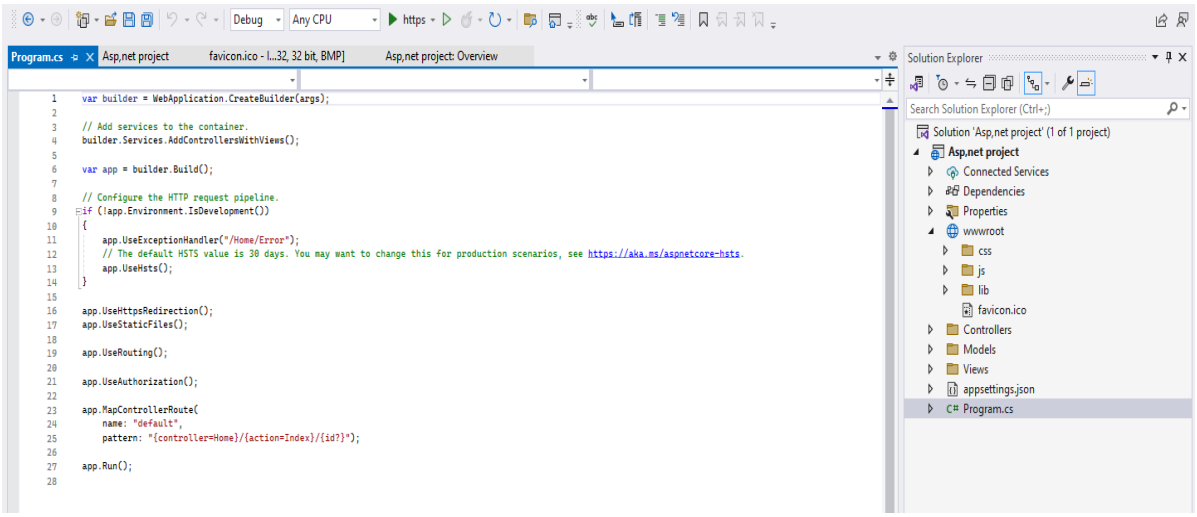



Рисунок 3.5 – Папки та код проекту

Запускаємо проект для перевірки роботи веб-сторінки, до внесення будь-яких змін (рис. 3.6).

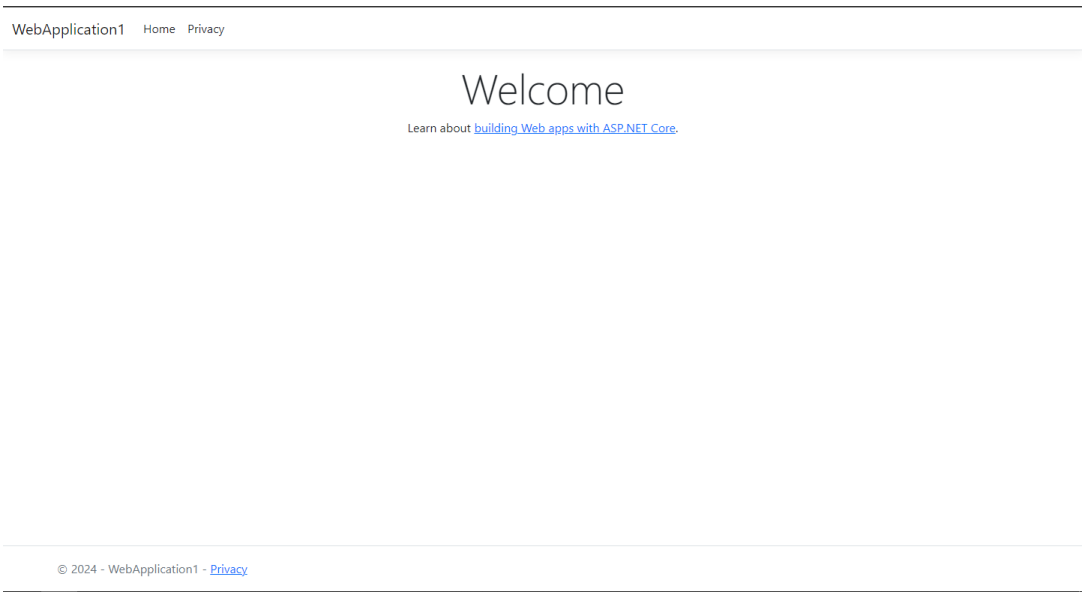



Рисунок 3.6 – Головна сторінка веб-сторінки

## 3.2 Розробка та дослідження бази даних інформаційної системи

Створюємо бази даних та додаємо у нього таблиці для подальшого зберігання інформації на веб-сторінці (рис. 3.7–3.8).

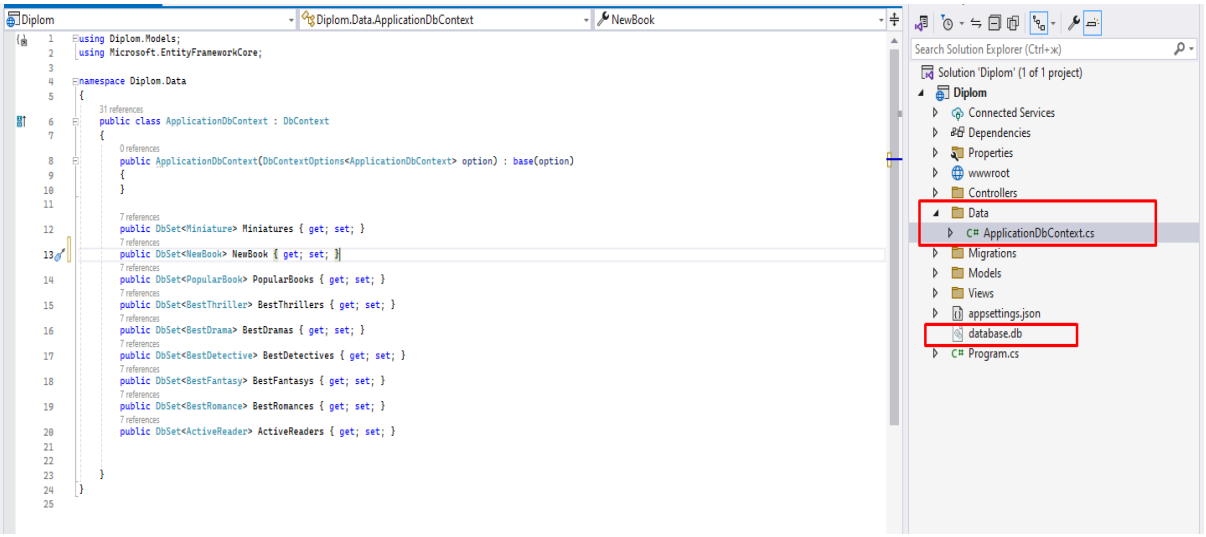



Рисунок 3.7 – Файли бази даних та частина коду для відображення таблиць

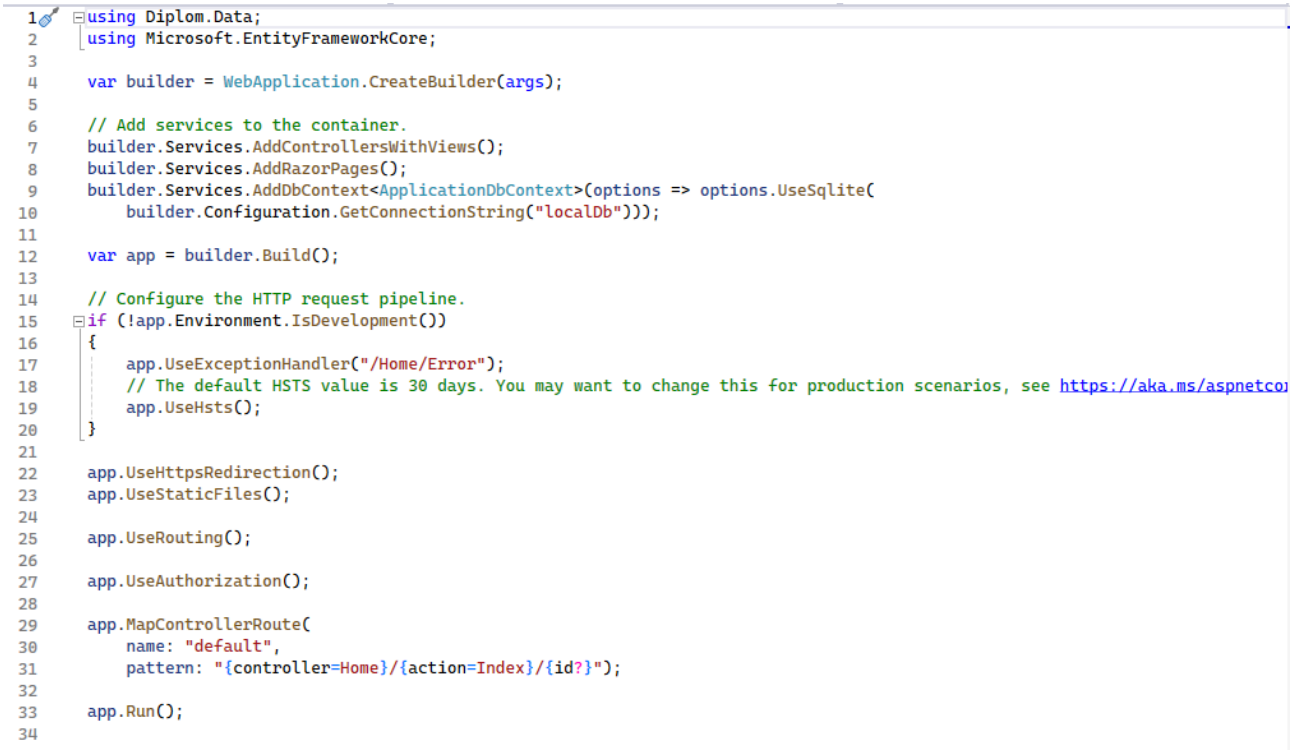
```
using Diplom.Models;
using Microsoft.EntityFrameworkCore;

namespace Diplom.Data
{
    31 references
    public class ApplicationDbContext : DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> option) : base(option)
        {
        }

        7 references
        public DbSet<Miniature> Miniatures { get; set; }
        7 references
        public DbSet<NewBook> NewBook { get; set; }
        7 references
        public DbSet<PopularBook> PopularBooks { get; set; }
        7 references
        public DbSet<BestThriller> BestThrillers { get; set; }
        7 references
        public DbSet<BestDrama> BestDramas { get; set; }
        7 references
        public DbSet<BestDetective> BestDetectives { get; set; }
        7 references
        public DbSet<BestFantasy> BestFantasys { get; set; }
        7 references
        public DbSet<BestRomance> BestRomances { get; set; }
        7 references
        public DbSet<ActiveReader> ActiveReaders { get; set; }


    }
}
```

Рисунок 3. 8 – Код для відображення таблиць

В даному коді ми можемо побачити які таблиці будуть відображенні на інформаційній сторінці, а саме таблиця NewBook, PopularBooks, BestThrillers, BestDramas, BestDetectives, BestFantasys, BestRomances, ActiveReaders (рис. 3.9-3.10).

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "localDb" : "Data Source=database.db"
  }
}
```

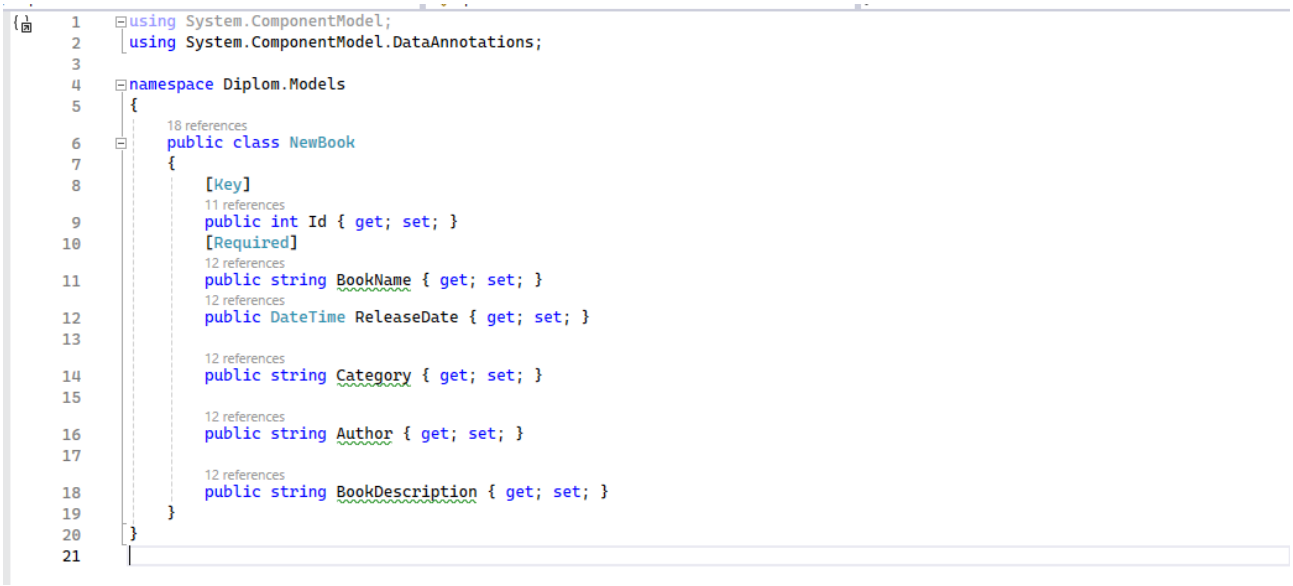
Рисунок 3. 9 – Код для підключення локальної бази даних до проекту



```
using Diplom.Data;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages();
builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseSqlite(
    builder.Configuration.GetConnectionString("localDb")));

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcor
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Рисунок 3. 10 – Код для впровадження залежностей з базою даних

Тепер створюємо за допомогою коду таблиці для різних жанрів книг, а саме таблиця з новинками, найпопулярніші книги, найкращі трилери, найкращі драми, найкращі детективи, найкращі фентезі, найкращі романи, найактивніші читачі (рис. 3.11–3.12).



```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{
    18 references
    public class NewBook
    {
        [Key]
        11 references
        public int Id { get; set; }
        [Required]
        12 references
        public string BookName { get; set; }
        12 references
        public DateTime ReleaseDate { get; set; }

        12 references
        public string Category { get; set; }

        12 references
        public string Author { get; set; }

        12 references
        public string BookDescription { get; set; }
    }
}
```

Рисунок 3.11 – Код для створення таблиці з новинками

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{
    public class PopularBook
    {
        [Key]

        public int Id { get; set; }
        [Required]

        public int BookPlace { get; set; }

        public string? BookName { get; set; }

        public DateTime ReleaseDate { get; set; }

        – references
        public string? Category { get; set; }

        – references
        public string? Author { get; set; }

        12 references
        public string? BookDescription { get; set; }
    }
}
```

Рисунок 3.12 – Код для створення таблиці найпопулярніші книги

В даному коді ми можемо побачити з чого складається таблиця найпопулярніші книги, а саме з стовпцю Id, стовпцю BookPlace, BookName, ReleaseDate, Category, Author, BookDescription (рис. 3.13-3.19).

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{
    – references
    public class BestThriller
    {
        [Key]
        – references
        public int Id { get; set; }
        [Required]

        – references
        public int BookPlace { get; set; }
        12 references
        public string? BookName { get; set; }
        12 references
        public DateTime ReleaseDate { get; set; }

        12 references
        public string? Category { get; set; }

        12 references
        public string? Author { get; set; }

        12 references
        public string? BookDescription { get; set; }
    }
}

```

Рисунок 3. 13 – Код для створення таблиці найкращі трилери

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{

    public class BestDrama
    {
        [Key]

        public int Id { get; set; }
        [Required]


        public int BookPlace { get; set; }

        public string? BookName { get; set; }

        public DateTime ReleaseDate { get; set; }


        public string? Category { get; set; }


        public string? Author { get; set; }

        – references
        public string? BookDescription { get; set; }
    }
}

```

Рисунок 3. 14 – Код для створення таблиці кращі драми

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{

    public class BestDetective
    {
        [Key]

        public int Id { get; set; }
        [Required]


        public int BookPlace { get; set; }

        public string? BookName { get; set; }

        public DateTime ReleaseDate { get; set; }


        public string? Category { get; set; }


        public string? Author { get; set; }


        public string? BookDescription { get; set; }
    }
}

```

Рисунок 3. 15 – Код для створення таблиці захоплюючі детективи

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{

    public class BestFantasy
    {
        [Key]

        public int Id { get; set; }
        [Required]


        public int BookPlace { get; set; }

        public string? BookName { get; set; }

        public DateTime ReleaseDate { get; set; }


        public string? Category { get; set; }


        public string? Author { get; set; }


        public string?   BookDescription { get; set; }
    }
}

```

Рисунок 3. 16 – Код для створення таблиці фентезі

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{

    public class BestRomance
    {
        [Key]

        public int Id { get; set; }
        [Required]


        public int BookPlace { get; set; }

        public string? BookName { get; set; }

        public DateTime ReleaseDate { get; set; }


        public string? Category { get; set; }


        public string? Author { get; set; }


        public string? BookDescription { get; set; }
    }
}
```

Рисунок 3. 17 – Код для створення таблиці зворушливі романи

```
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{

    public class ActiveReader
    {
        [Key]

        public int Id { get; set; }
        [Required]

        public string Name { get; set; }

        public string SurName { get; set; }

        public DateTime RegistrationDate { get; set; }


        public DateTime DateOfBirth { get; set; }


        public int PhoneNumber { get; set; }


        public int ReadedCount { get; set; }
    }
}
```

Рисунок 3. 18 – Код для створення таблиці наші читачі

```
using System.ComponentModel.DataAnnotations;

namespace Diplom.Models
{
    4 references
    public class Contacts
    {
        [Display(Name = "Введіть ім'я")]
        [Required(ErrorMessage = "Вам потрібно ввести ім'я")]
        3 references
        public string Name { get; set; }

        [Display(Name = "Введіть прізвище")]
        [Required(ErrorMessage = "Вам потрібно ввести прізвище")]
        3 references
        public string Surname { get; set; }

        [Display(Name = "Введіть вік")]
        [Required(ErrorMessage = "Вам потрібно ввести вік")]
        3 references
        public int Age { get; set; }

        [Display(Name = "Введіть e-mail")]
        [Required(ErrorMessage = "Вам потрібно ввести пошту")]
        3 references
        public string Email { get; set;}

        [Display(Name = "Введіть повідомлення")]
        [StringLength(30, ErrorMessage = "Текст менше 30 символів")]
        [Required(ErrorMessage = "Вам потрібно ввести повідомлення")]
        3 references
        public string Message { get; set;}

    }
}

```

Рисунок 3. 19 – Код для створення таблиці зворотній зв’язок

Після створення усіх таблиць, потрібно зробити міграцію кожної таблиці за допомогою коду в базу даних і щоб вона виводилась на веб-ресурсі, для прикладу продемонструю фото коду міграції трьох таблиць (рис. 3.20–3.26).

Цей код виконує міграцію бази даних за допомогою Entity Framework Core в ASP.NET Core. Він додає нову таблицю «NewBook» до бази даних і надає спосіб видалення цієї таблиці.

Функції коду:

– Імпорти.

– Оголошення класу.

– Метод Up, який виконується при застосуванні міграції, створює нову таблицю «NewBook».

– Метод Down, який виконується при відкаті міграції, видаляє таблицю «NewBook».

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Diplom.Migrations
{
    /// <inheritdoc />
    public partial class AddNewBookTableToDb : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "NewBook",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    BookName = table.Column<string>(type: "TEXT", nullable: false),
                    ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                    Category = table.Column<string>(type: "TEXT", nullable: false),
                    Author = table.Column<string>(type: "TEXT", nullable: false),
                    BookDescription = table.Column<string>(type: "TEXT", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_NewBook", x => x.Id);
                });
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "NewBook");
        }
    }
}
```

Рисунок 3.20 – Код міграції таблиці Нові книги в базу даних

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Diplom.Migrations
{
    /// <inheritdoc />
    public partial class AddPopularBooksTableToDb : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "ActiveReaders",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Name = table.Column<string>(type: "TEXT", nullable: false),
                    SurName = table.Column<string>(type: "TEXT", nullable: false),
                    RegistrationDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                    DateOfBirth = table.Column<DateTime>(type: "TEXT", nullable: false),
                    PhoneNumber = table.Column<int>(type: "INTEGER", nullable: false),
                    ReadedCount = table.Column<int>(type: "INTEGER", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_ActiveReaders", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "BestDetectives",
                columns: table => new
                {
                    Id = table.Column<int>(type: "INTEGER", nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
```

Рисунок 3. 21 – Код міграції таблиці Популярні книги в базу даних

```
                BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
                BookName = table.Column<string>(type: "TEXT", nullable: false),
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                Category = table.Column<string>(type: "TEXT", nullable: false),
                Author = table.Column<string>(type: "TEXT", nullable: false),
                BookDescription = table.Column<string>(type: "TEXT", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_BestDetectives", x => x.Id);
            });

        migrationBuilder.CreateTable(
            name: "BestDramas",
            columns: table => new
            {
                Id = table.Column<int>(type: "INTEGER", nullable: false)
                    .Annotation("Sqlite:Autoincrement", true),
                BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
                BookName = table.Column<string>(type: "TEXT", nullable: false),
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                Category = table.Column<string>(type: "TEXT", nullable: false),
                Author = table.Column<string>(type: "TEXT", nullable: false),
                BookDescription = table.Column<string>(type: "TEXT", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_BestDramas", x => x.Id);
            });

        migrationBuilder.CreateTable(
            name: "BestFantasys",
            columns: table => new
            {
                Id = table.Column<int>(type: "INTEGER", nullable: false)
                    .Annotation("Sqlite:Autoincrement", true),
                BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
                BookName = table.Column<string>(type: "TEXT", nullable: false),
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
```

Рисунок 3. 22 – Код міграції таблиці Популярні книги в базу даних

```
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                Category = table.Column<string>(type: "TEXT", nullable: false),
                Author = table.Column<string>(type: "TEXT", nullable: false),
                BookDescription = table.Column<string>(type: "TEXT", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_BestFantasys", x => x.Id);
            });

        migrationBuilder.CreateTable(
            name: "BestRomances",
            columns: table => new
            {
                Id = table.Column<int>(type: "INTEGER", nullable: false)
                    .Annotation("Sqlite:Autoincrement", true),
                BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
                BookName = table.Column<string>(type: "TEXT", nullable: false),
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                Category = table.Column<string>(type: "TEXT", nullable: false),
                Author = table.Column<string>(type: "TEXT", nullable: false),
                BookDescription = table.Column<string>(type: "TEXT", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_BestRomances", x => x.Id);
            });

        migrationBuilder.CreateTable(
            name: "BestThrillers",
            columns: table => new
            {
                Id = table.Column<int>(type: "INTEGER", nullable: false)
                    .Annotation("Sqlite:Autoincrement", true),
                BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
                BookName = table.Column<string>(type: "TEXT", nullable: false),
                ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
                Category = table.Column<string>(type: "TEXT", nullable: false),
                Author = table.Column<string>(type: "TEXT", nullable: false),
```

Рисунок 3. 23 – Код міграції таблиці Популярні книги в базу даних

```
114                     Author = table.Column<string>(type: "TEXT", nullable: false),
115                     BookDescription = table.Column<string>(type: "TEXT", nullable: false)
116                 },
117                 constraints: table =>
118                 {
119                     table.PrimaryKey("PK_BestThrillers", x => x.Id);
120                 });
121 
122             migrationBuilder.CreateTable(
123                 name: "PopularBooks",
124                 columns: table => new
125                 {
126                     Id = table.Column<int>(type: "INTEGER", nullable: false)
127                         .Annotation("Sqlite:Autoincrement", true),
128                     BookPlace = table.Column<int>(type: "INTEGER", nullable: false),
129                     BookName = table.Column<string>(type: "TEXT", nullable: false),
130                     ReleaseDate = table.Column<DateTime>(type: "TEXT", nullable: false),
131                     Category = table.Column<string>(type: "TEXT", nullable: false),
132                     Author = table.Column<string>(type: "TEXT", nullable: false),
133                     BookDescription = table.Column<string>(type: "TEXT", nullable: false)
134                 },
135                 constraints: table =>
136                 {
137                     table.PrimaryKey("PK_PopularBooks", x => x.Id);
138                 });
139         }
140 
141         /// <inheritdoc />
            0 references
142         protected override void Down(MigrationBuilder migrationBuilder)
143         {
144             migrationBuilder.DropTable(
145                 name: "ActiveReaders");
146 
147             migrationBuilder.DropTable(
148                 name: "BestDetectives");
149 
150             migrationBuilder.DropTable(
151                 name: "BestDramas");
```

Рисунок 3. 24 – Код міграції таблиці Популярні книги в базу даних

```
151                 name: "BestDramas");
152 
153             migrationBuilder.DropTable(
154                 name: "BestFantasys");
155 
156             migrationBuilder.DropTable(
157                 name: "BestRomances");
158 
159             migrationBuilder.DropTable(
160                 name: "BestThrillers");
161 
162             migrationBuilder.DropTable(
163                 name: "PopularBooks");
164         }
165     }
166 }
```

Рисунок 3. 25 – Код міграції таблиці Популярні книги в базу даних

```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Diplom.Migrations
{
    /// <inheritdoc />
    1 reference
    public partial class AddBestDetectivesTableToDb : Migration
    {
        /// <inheritdoc />
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AlterColumn<string>(
                name: "BookDescription",
                table: "PopularBooks",
                type: "TEXT",
                nullable: true,
                oldClrType: typeof(string),
                oldType: "TEXT");
        }

        /// <inheritdoc />
        0 references
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AlterColumn<string>(
                name: "BookDescription",
                table: "PopularBooks",
                type: "TEXT",
                nullable: false,
                defaultValue: "",
                oldClrType: typeof(string),
                oldType: "TEXT",
                oldNullable: true);
        }
    }
}
```

Рисунок 3. 26 – Код міграції таблиці Кращі детективи в базу даних

### 3.3 Розробка та дослідження контролерів інформаційної системи та стилізація сторінок на веб-ресурсі

Після створення усіх таблиць та міграції їх до бази даних потрібно створити Controllers, це можна створити натиснувши правою кнопкою миші на папку Controllers та обрати пункт Add та вибрати підпункт New Scaffolded Item (рис. 3.27–3.32).

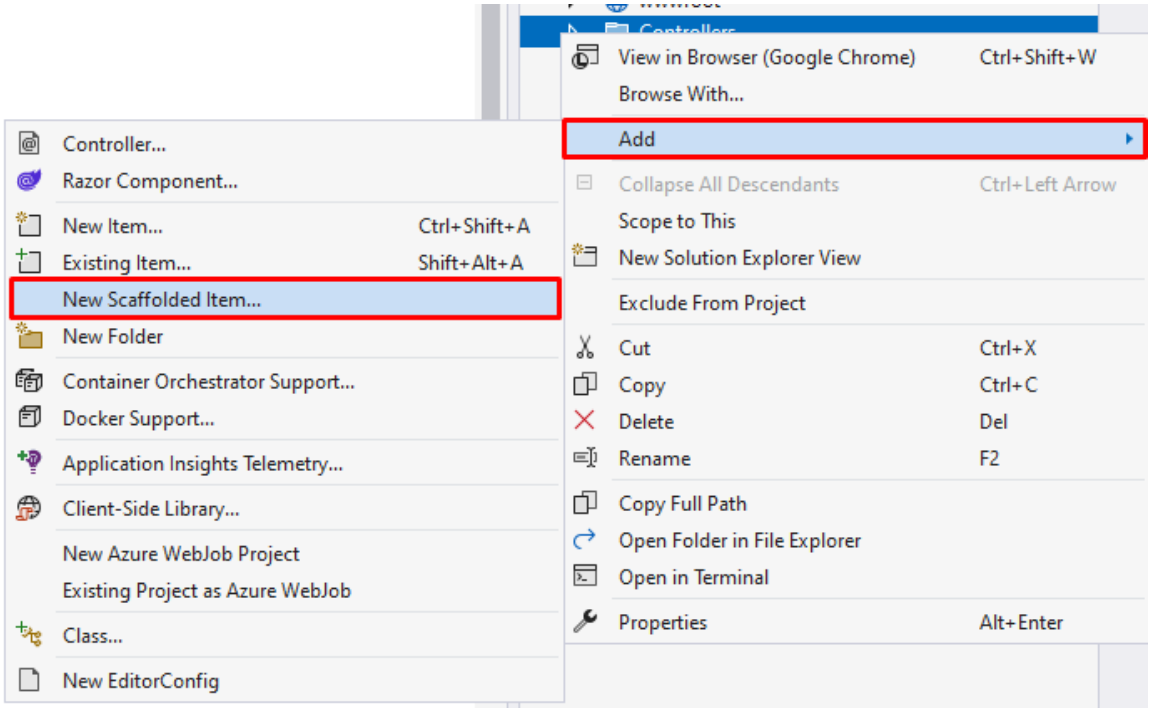



Рисунок 3.27 – Додання контролера

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Diplom.Data;
using Diplom.Models;

namespace Diplom.Controllers
{
    1 reference
    public class PopularBooksController : Controller
    {
        private readonly ApplicationDbContext _context;

        0 references
        public PopularBooksController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: PopularBooks
        3 references
        public async Task<IActionResult> Index()
        {
            return View(await _context.PopularBooks.ToListAsync());
        }

        // GET: PopularBooks/Details/5
        0 references
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var popularBook = await _context.PopularBooks
```

Рисунок 3. 28 – Код після створення контролера для таблиці Популярні книги

```
            .FirstOrDefaultAsync(m => m.Id == id);
        if (popularBook == null)
        {
            return NotFound();
        }

        return View(popularBook);
    }

    // GET: PopularBooks/Create
    0 references
    public IActionResult Create()
    {
        return View();
    }

    // POST: PopularBooks/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> Create([Bind("Id,BookPlace,BookName,ReleaseDate,Category,Author,BookDescription
    {
        if (ModelState.IsValid)
        {
            _context.Add(popularBook);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(popularBook);
    }

    // GET: PopularBooks/Edit/5
    0 references
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
```

Рисунок 3. 29 – Код після створення контролера для таблиці Популярні книги

```
        }

        var popularBook = await _context.PopularBooks.FindAsync(id);
        if (popularBook == null)
        {
            return NotFound();
        }
        return View(popularBook);
    }

    // POST: PopularBooks/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> Edit(int id, [Bind("Id,BookPlace,BookName,ReleaseDate,Category,Author,BookDescr
    {
        if (id != popularBook.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(popularBook);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!PopularBookExists(popularBook.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
```

Рисунок 3. 30 – Код після створення контролера для таблиці Популярні книги

```
            }
            return RedirectToAction(nameof(Index));
        }
        return View(popularBook);
    }

    // GET: PopularBooks/Delete/5
    0 references
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var popularBook = await _context.PopularBooks
            .FirstOrDefaultAsync(m => m.Id == id);
        if (popularBook == null)
        {
            return NotFound();
        }

        return View(popularBook);
    }

    // POST: PopularBooks/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var popularBook = await _context.PopularBooks.FindAsync(id);
        if (popularBook != null)
        {
            _context.PopularBooks.Remove(popularBook);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
```

Рисунок 3. 31 – Код після створення контролера для таблиці Популярні книги

```
151
        1 reference
152     private bool PopularBookExists(int id)
153     {
154         return _context.PopularBooks.Any(e => e.Id == id);
155     }
156 }
157 }
158
```

Рисунок 3. 32 – Код після створення контролера для таблиці Популярні книги

Цей код визначає контролер PopularBooksController в ASP.NET Core MVC додатку, який обробляє CRUD (створення, читання, оновлення, видалення) операції для об'єктів PopularBook.

Контролер PopularBooksController керує операціями з моделлю PopularBook.

Контекст використовує ApplicationDbContext для взаємодії з базою даних (рис. 3.33–3.34).

```
using Diplom.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

namespace Diplom.Controllers
{
    3 references
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        0 references
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        0 references
        public IActionResult Index()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        0 references
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

Рисунок 3. 33 – Код після створення контролера для Головної сторінки

```
using Diplom.Models;
using Microsoft.AspNetCore.Mvc;

namespace Diplom.Controllers
{
    0 references
    public class ContactsController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        0 references
        public IActionResult Check(Contacts contact)
        {
            if(ModelState.IsValid)
            {
                return Redirect("/");
            }

            return View("Index");
        }
    }
}
```

Рисунок 3. 34 – Код після створення контролера для сторінки Зворотній зв’язок

Після додавання контролерів, веб-сторінка повністю готова до стилізації та заповнення інформацією. Для більш зручної стилізації сторінок використовував Bootstrap стилі, щоб сторінка мала більш привабливий вигляд та була адаптивною (рис. 3.35–3.36).

```
@model IEnumerable<Diplom.Models.NewBook>

@{
    ViewData["Title"] = "Новинки";
}

<div class="container">
    <div class="row">
        <div class="col">
            <h1 class="mt-4 mb-4 text-center font-weight-bold">Що читати в 2024 році, найочікуваніші книжкові новинки.</h1>
            <h3 class="mb-4">
                Завершуючи цей рік, українські видавництва не просто підводять підсумки 2023-го, але й висувають на передній план свої амбіційні плани на майбутнє. Вони плекають уяви, готують сюрпризи та збираються пригощати читачів запам'ятовуваною літературою.
                Особливим подією стане виходження найкращого художнього роману 2023 року, який отримав свою славу за версією впливової платформи Goodreads. Це миттєво стане об'єктом бажання для багатьох поціновувачів літератури, які прагнуть поглибитися у світ художньої магії та відкрити для себе нові відчуття та емоції.
                Крім того, книжкові полиці збагатяться ще однією перлиною, яка вже підкорила серця читачів — твір, що вразив своєю оригінальністю та змістом. Його популярність стане неабияким випробуванням для книготоргових платформ та збере величезну армію прихильників.
                З нетерпінням очікується виходження найбільш очікуваних книжкових релізів 2024 року, завертаючи сторінки нових історій, які промовлять до душі та даруватимуть читачам безліч незабутніх моментів.
            </h3>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <a asp-action="Create" class="btn btn-primary mb-4">Створити новий запис</a>
            <table class="table">
                <thead>
                    <tr>
                        <th>@Html.DisplayNameFor(model => model.BookName)</th>
                        <th>@Html.DisplayNameFor(model => model.ReleaseDate)</th>
                        <th>@Html.DisplayNameFor(model => model.Category)</th>
                        <th>@Html.DisplayNameFor(model => model.Author)</th>
                        <th>@Html.DisplayNameFor(model => model.BookDescription)</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
```

Рисунок 3. 35 – Код зі стилями для сторінки «Книжні новинки»

```
                    @foreach (var item in Model)
                    {
                        <tr>
                            <td>@Html.DisplayFor(modelItem => item.BookName)</td>
                            <td>@Html.DisplayFor(modelItem => item.ReleaseDate)</td>
                            <td>@Html.DisplayFor(modelItem => item.Category)</td>
                            <td>@Html.DisplayFor(modelItem => item.Author)</td>
                            <td>@Html.DisplayFor(modelItem => item.BookDescription)</td>
                            <td>
                                <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-sm btn-primary mb-2">Редагувати</a>
                                <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-sm btn-info mb-2">Деталі</a>
                                <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-danger mb-2">Видалити</a>
                            </td>
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <h3 class="mt-4 mb-4">Завершуючи наш огляд книжкових новинок 2024 року, ми бачимо, що світ літератури продовжує нас дивувати й надихати. Цей рік приніс нам низку захоплюючих творів, які розкривають нові світи, перспективи та ідеї. Незалежно від вашого смаку чи уподобань, у світі книг завжди є місце для вас. Так що нехай кожна нова сторінка, яку ви перевертаєте, веде вас до нових відкриттів і пригод, а читання залишається найвідміннішим засобом розвитку та насолоди.</h3>
        </div>
    </div>
</div>
```

Рисунок 3. 36 – Код зі стилями для сторінки «Книжні новинки»

Написавши стилі для сторінки переглядаємо, який вона має вигляд у браузері (рис. 3.37).

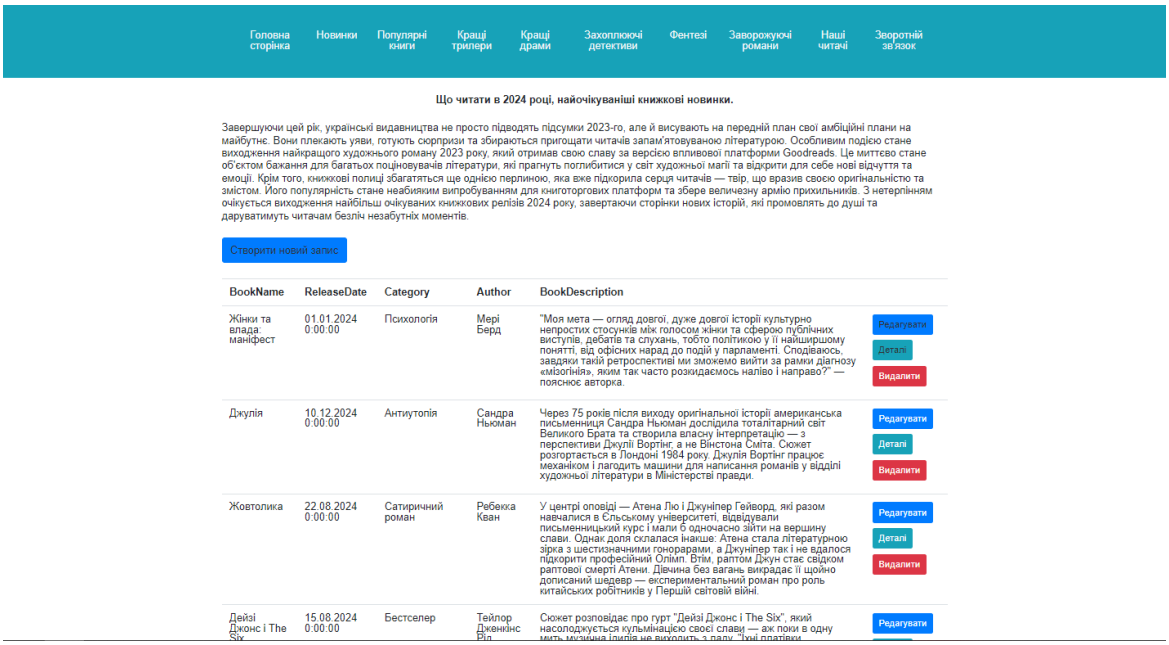
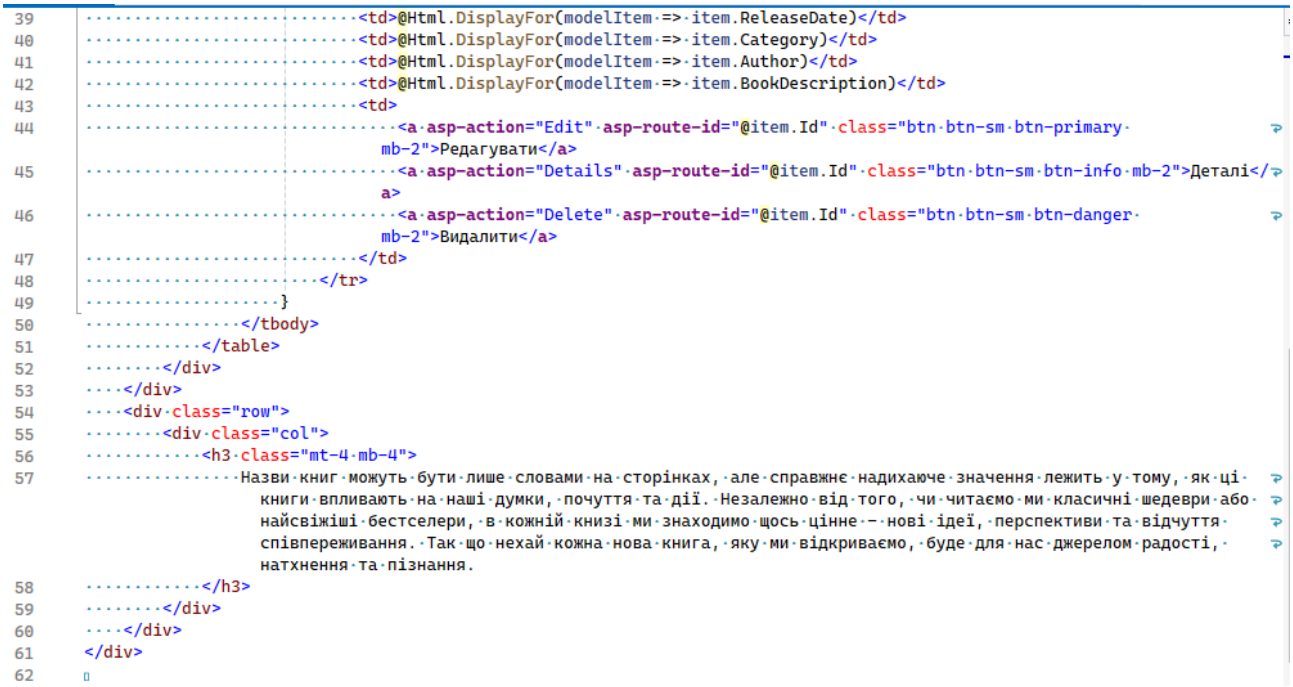



Рисунок 3.37 – Вигляд сторінки «Книжні новинки» після додавання стилів

На прикладі даної сторінки, проводимо стилізацію для усіх сторінок та заповняємо сторінки інформацією (рис. 3.38–3.41).

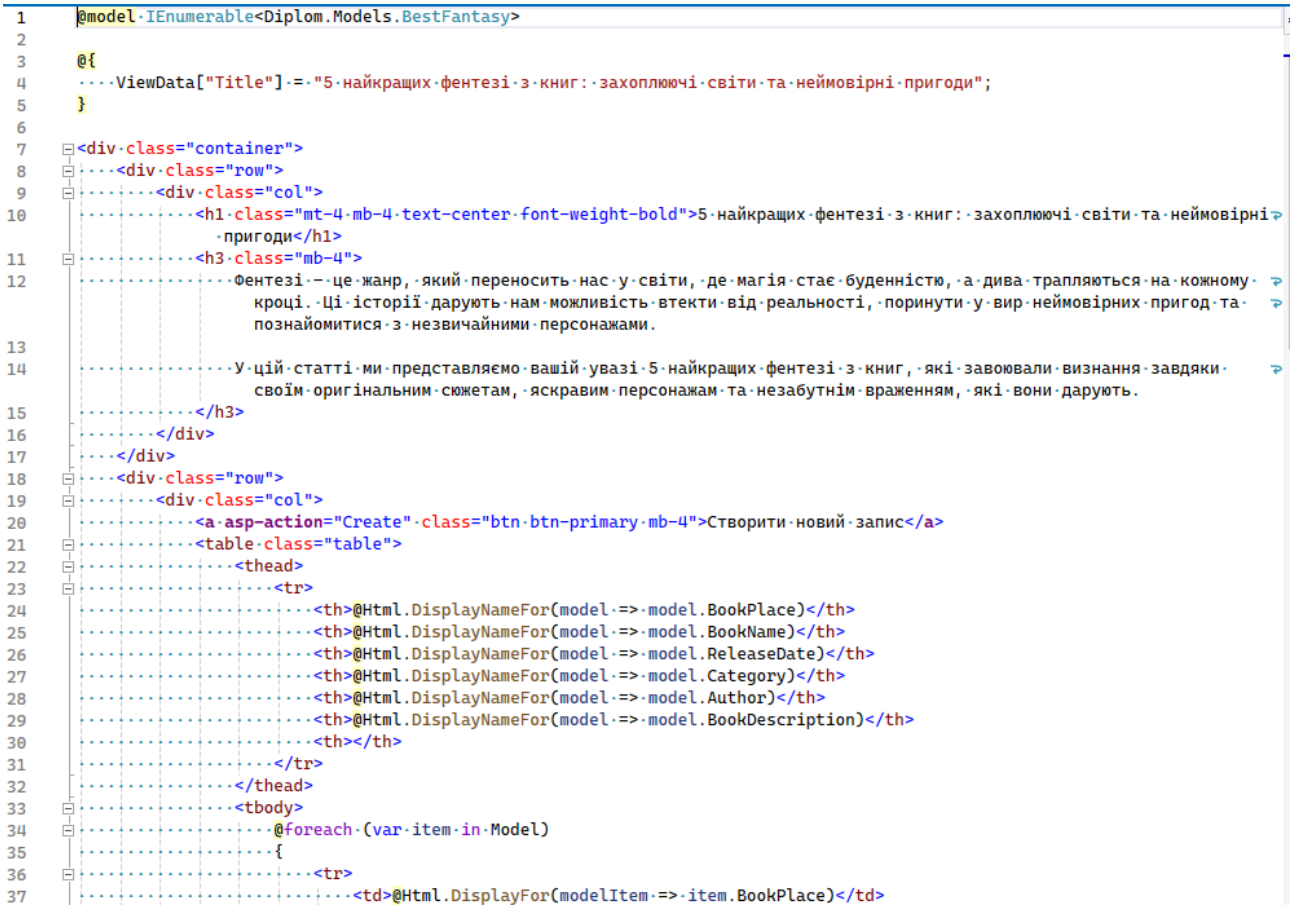
```
@model IEnumerable<Diplom.Models.BestThriller>

@{
    ViewData["Title"] = "Найкращі трилери";
}

<div class="container">
    <div class="row">
        <div class="col">
            <h1 class="mt-4 mb-4 text-center font-weight-bold">Найкращі трилери</h1>
            <h3 class="mb-4">
                Трилери – це жанр літератури, який тримає читача в напрузі від першої до останньої сторінки. Завдяки захоплюючим сюжетам, динамічному розвитку подій та несподіваним поворотам вони змушують наше серце битися швидше, а мозок працювати на повну потужність.

                У цій статті ми представляємо вам 5 найкращих трилерів, які неодмінно занурять вас у світ таємниць, небезпек та захоплюючих пригод.
            </h3>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <a asp-action="Create" class="btn btn-primary mb-4">Створити новий запис</a>
            <table class="table">
                <thead>
                    <tr>
                        <th>@Html.DisplayNameFor(model => model.BookPlace)</th>
                        <th>@Html.DisplayNameFor(model => model.BookName)</th>
                        <th>@Html.DisplayNameFor(model => model.ReleaseDate)</th>
                        <th>@Html.DisplayNameFor(model => model.Category)</th>
                        <th>@Html.DisplayNameFor(model => model.Author)</th>
                        <th>@Html.DisplayNameFor(model => model.BookDescription)</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var item in Model)
                    {
                        <tr>
                            <td>@Html.DisplayFor(modelItem => item.BookPlace)</td>
                            <td>@Html.DisplayFor(modelItem => item.BookName)</td>
```

Рисунок 3. 38 – Код зі стилями для сторінки «Найкращі трилери»



```
                        <td>@Html.DisplayFor(modelItem => item.ReleaseDate)</td>
                        <td>@Html.DisplayFor(modelItem => item.Category)</td>
                        <td>@Html.DisplayFor(modelItem => item.Author)</td>
                        <td>@Html.DisplayFor(modelItem => item.BookDescription)</td>
                        <td>
                            <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-sm btn-primary mb-2">Редагувати</a>
                            <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-sm btn-info mb-2">Деталі</a>
                            <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-danger mb-2">Видалити</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
<div class="row">
    <div class="col">
        <h3 class="mt-4 mb-4">
            Назви книг можуть бути лише словами на сторінках, але справжнє надихаюче значення лежить у тому, як ці книги впливають на наші думки, почуття та дії. Незалежно від того, чи читаємо ми класичні шедеври або найсвіжіші бестселери, в кожній книзі ми знаходимо щось цінне – нові ідеї, перспективи та відчуття співпереживання. Так що нехай кожна нова книга, яку ми відкриваємо, буде для нас джерелом радості, натхнення та пізнання.
        </h3>
    </div>
</div>
</div>
```

Рисунок 3. 39 – Код зі стилями для сторінки «Найкращі трилери»



```
@model IEnumerable<Diplom.Models.BestFantasy>

@{
    ViewData["Title"] = "5 найкращих фентезі з книг: захоплюючі світи та неймовірні пригоди";
}

<div class="container">
    <div class="row">
        <div class="col">
            <h1 class="mt-4 mb-4 text-center font-weight-bold">5 найкращих фентезі з книг: захоплюючі світи та неймовірні пригоди</h1>
            <h3 class="mb-4">
                Фентезі — це жанр, який переносить нас у світи, де магія стає буденністю, а дива трапляються на кожному кроці. Ці історії дарують нам можливість втекти від реальності, поринути у вир неймовірних пригод та познайомитися з незвичайними персонажами.

                У цій статті ми представляємо вашій увазі 5 найкращих фентезі з книг, які завоювали визнання завдяки своїм оригінальним сюжетам, яскравим персонажам та незабутнім враженням, які вони дарують.
            </h3>
        </div>
    </div>
    <div class="row">
        <div class="col">
            <a asp-action="Create" class="btn btn-primary mb-4">Створити новий запис</a>
            <table class="table">
                <thead>
                    <tr>
                        <th>@Html.DisplayNameFor(model => model.BookPlace)</th>
                        <th>@Html.DisplayNameFor(model => model.BookName)</th>
                        <th>@Html.DisplayNameFor(model => model.ReleaseDate)</th>
                        <th>@Html.DisplayNameFor(model => model.Category)</th>
                        <th>@Html.DisplayNameFor(model => model.Author)</th>
                        <th>@Html.DisplayNameFor(model => model.BookDescription)</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var item in Model)
                    {
                        <tr>
                            <td>@Html.DisplayFor(modelItem => item.BookPlace)</td>
```

Рисунок 3. 40 – Код зі стилями для сторінки «Найкращі фентезі»

```
                        <td>@Html.DisplayFor(modelItem => item.BookName)</td>
                        <td>@Html.DisplayFor(modelItem => item.ReleaseDate)</td>
                        <td>@Html.DisplayFor(modelItem => item.Category)</td>
                        <td>@Html.DisplayFor(modelItem => item.Author)</td>
                        <td>@Html.DisplayFor(modelItem => item.BookDescription)</td>
                        <td>
                            <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-sm btn-primary
                            mb-2">Редагувати</a>
                            <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-sm btn-info mb-2">Деталі</
                            a>
                            <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-danger
                            mb-2">Видалити</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>
<div class="row">
    <div class="col">
        <h3 class="mt-4 mb-4">
            Це лише п'ять з безлічі захоплюючих фентезі-романів, які здатні перенести вас у світи, де панує чари та
            дива. Вибір найкращого фентезі залежить від ваших особистих уподобань, адже кожна з цих історій володіє
            унікальною атмосферою та емоціями, які роблять їхніми читачів.

            Порада: Не зупиняйтеся на цій п'ятірці! Продовжуйте досліджувати світ фентезі-літератури, адже він криє в
            собі безліч скарбів, які очікують на те, щоб їх відкрили.
        </h3>
    </div>
</div>
```

Рисунок 3. 41 – Код зі стилями для сторінки «Найкращі фентезі»

Стилізувавши усі сторінки, розпочинаємо вручну додавати дані до нашої бази даних, для цього у нас є спеціальна кнопка котра відразу додає дані на сторінці, після заповнення інформаційних рядків (рис. 3.42–3.47).

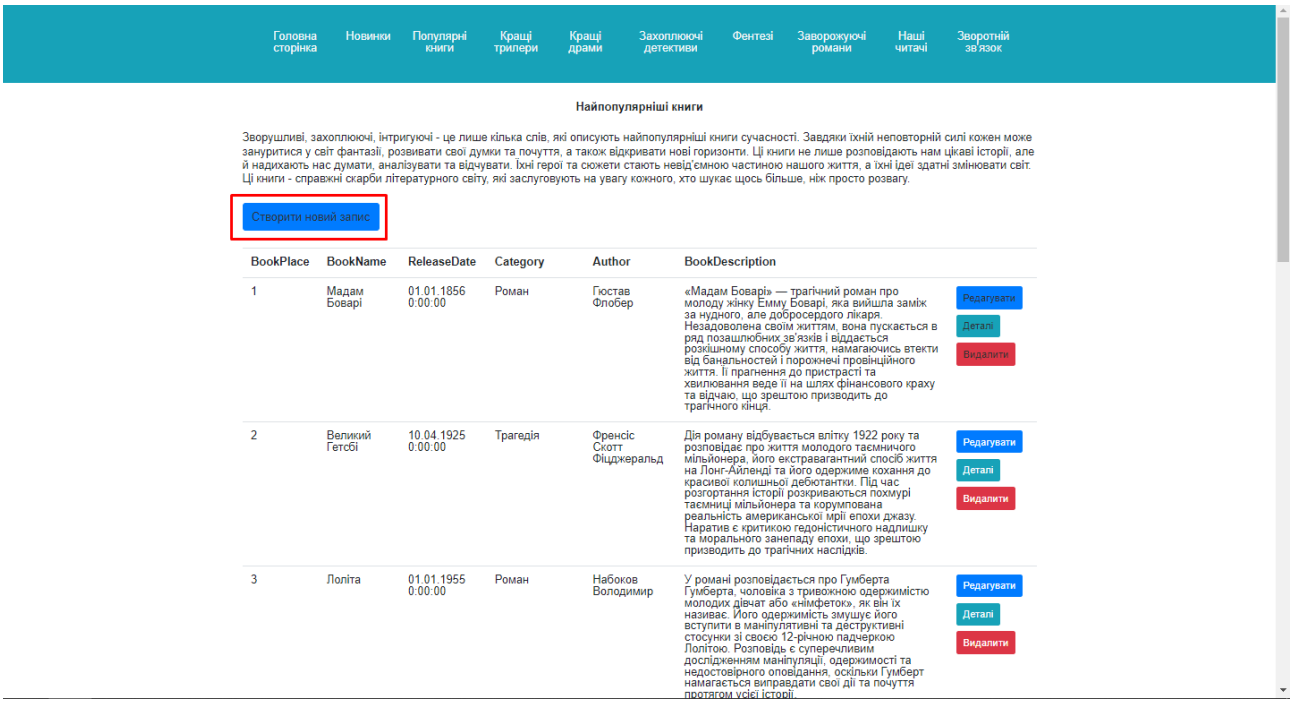



Рисунок 3. 42 – Вигляд сторінки з доданою інформацією та кнопкою через яку додається інформація

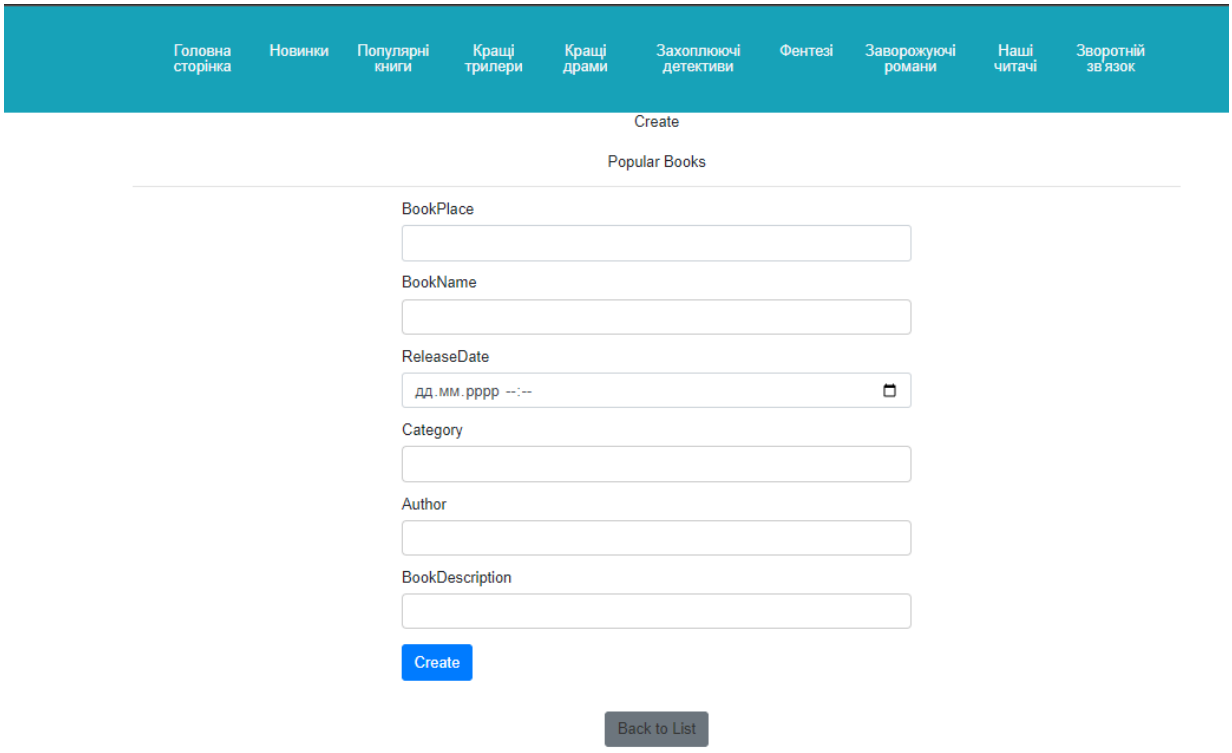



Рисунок 3. 43 – Вигляд сторінки де можна додавати інформацію

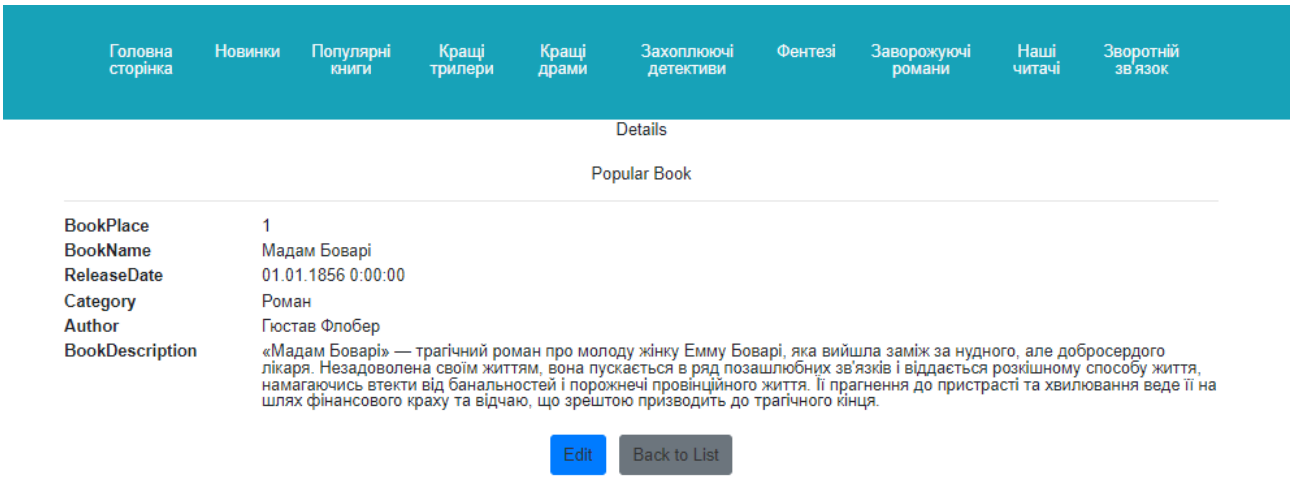



Рисунок 3. 44 – Вигляд сторінки Деталі

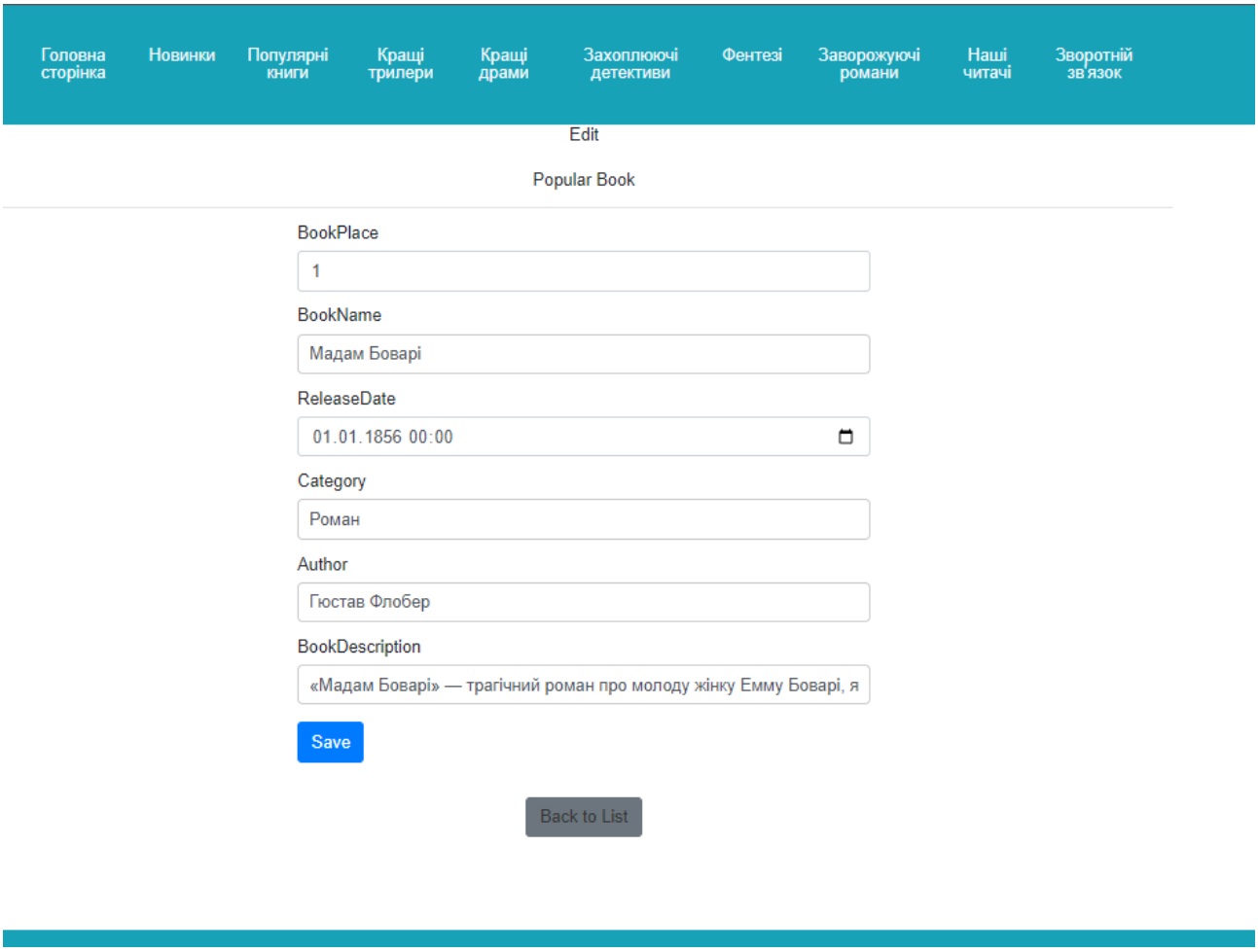



Рисунок 3. 45 – Вигляд сторінки Редагування

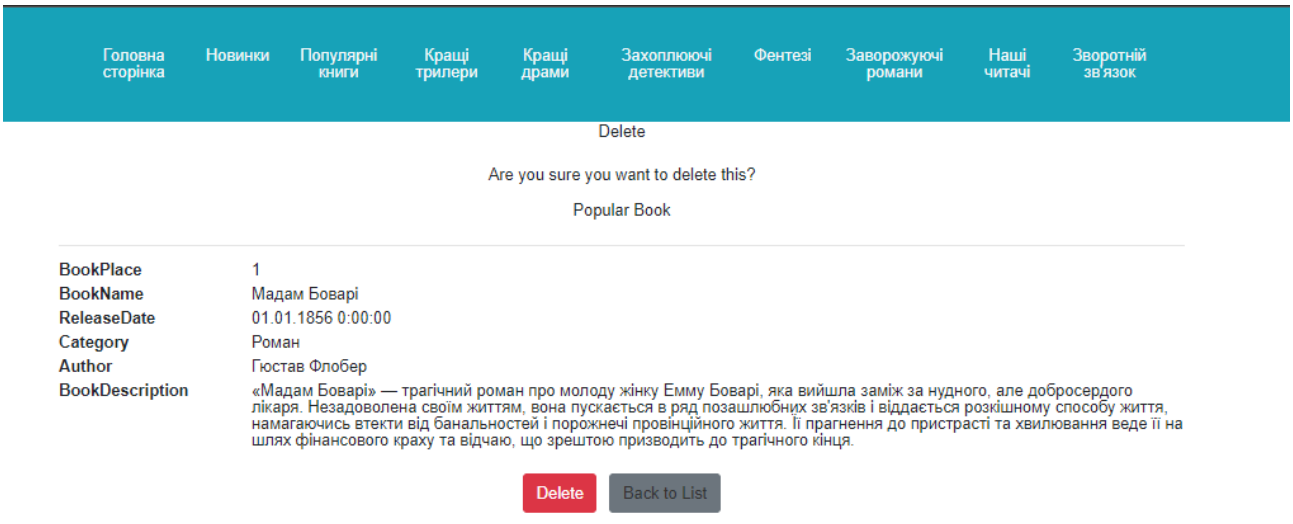



Рисунок 3. 46 – Вигляд сторінки видалення

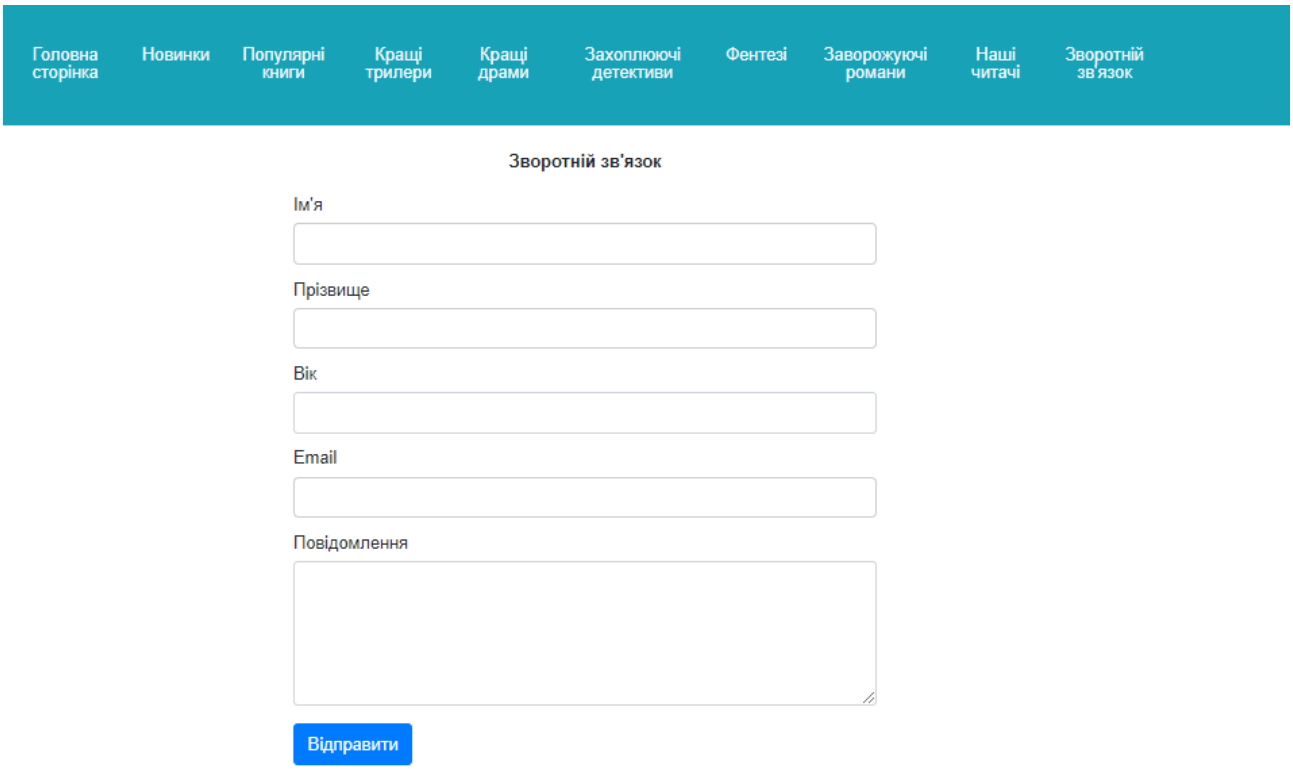



Рисунок 3. 47 – Вигляд сторінки Зворотній зв’язок

# ВИСНОВКИ

У кваліфікаційній роботі було проведено аналіз теоретичних аспектів платформи ASP.NET Core MVC, що включав дослідження архітектури та основних компонентів цієї платформи. Було детально вивчено моделі, подання та контролери, які складають її основу.

Також досліджено методи інтеграції з базами даних, проаналізовано різні підходи до інтеграції ASP.NET Core MVC з базами даних та визначено оптимальні методи для забезпечення ефективного збереження та обробки даних.

В рамках дослідження розроблено проект інформаційної системи, що реалізований на платформі ASP.NET Core MVC. Були створені та впроваджені основні функціональні модулі системи, зокрема обробка користувацьких запитів та управління даними.

Для забезпечення безпеки інформаційної системи впроваджено механізми автентифікації та авторизації користувачів, а також реалізовано заходи захисту даних від поширених веб-загроз та атак.

Було проведено комплексне тестування створеної інформаційної системи з метою виявлення помилок та недоліків. Продуктивність системи оцінено під різними навантаженнями.

Підготовлено технічну документацію, яка описує структуру та функціональні можливості розробленої інформаційної системи. Отримані результати проаналізовано та зроблено висновки щодо ефективності розробленої системи.

Надано рекомендації щодо подальшого вдосконалення системи, включаючи можливі напрямки розвитку та вдосконалення інформаційної системи. Оцінено перспективи використання новітніх технологій для підвищення ефективності та функціональності системи.

На першому етапі роботи проведено детальний аналіз предметної області, що включав дослідження платформи ASP.NET Core MVC. Визначено її основні переваги та можливості, такі як модульність, розширюваність та підтримка сучасних стандартів веб-розробки. Розглянуто компоненти платформи, зокрема

моделі, контролери та представлення, які дозволяють створювати масштабовані та підтримувані веб-додатки.

Подальший етап роботи включав процес проектування та розробки інформаційної системи. Було розроблено загальну архітектуру системи, визначено основні компоненти та їх взаємодію. У ході розробки використовувалися сучасні технології та інструменти, такі як Entity Framework Core для роботи з базою даних, а також Visual Studio для розробки та налагодження коду. Значна увага приділялася процесу розробки та впровадження, включаючи визначення вимог, проектування, кодування, тестування та впровадження системи.

У рамках роботи також створено інформаційну сторінку, яка забезпечує зручний та інтуїтивно зрозумілий доступ до необхідної інформації. Використання сучасних технологій та інструментів дозволило реалізувати ефективний функціонал для перегляду та управління даними.

Результатом роботи стала розроблена архітектура інформаційної системи на базі ASP.NET Core MVC, яка забезпечує високу продуктивність, масштабованість та підтримуваність. Використання гнучких методологій розробки, таких як Agile та Scrum, дозволило забезпечити адаптивність до змінних вимог користувачів. Застосування автоматизованого тестування та практик перевірок коду сприяло забезпеченню високої якості програмного продукту.

Практичне значення одержаних результатів полягає у можливості використання розробленої інформаційної системи в різних галузях для ефективного управління даними та покращення взаємодії з користувачами. Система є гнучкою та розширюваною, що дозволяє легко адаптувати її до специфічних вимог та потреб різних організацій. Наукова новизна роботи включає застосування новітніх підходів та технологій у розробці веб-додатків, що сприяє підвищенню ефективності процесу розробки та якості кінцевого продукту.

Загалом, результати проведеного дослідження та розробки підтвердили

високу ефективність платформи ASP.NET Core MVC для створення сучасних інформаційних систем, що задовольняють вимоги користувачів щодо зручності, продуктивності та надійності.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Blog – Microsoft DevBlogs. Microsoft. URL: https://devblogs.microsoft.com/aspnet (date of access: 20.05.2024).

2. ASP.NET Core Authentication – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authentication (date of access: 20.05.2024).

3. ASP.NET Core Documentation – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/aspnet/core (date of access: 20.05.2024).

4. ASP.NET Core in Visual Studio – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-aspnet-core?view=vs-2019 (date of access: 20.05.2024).

5. ASP.NET Core MVC Tutorial – Tutorialspoint. Tutorialspoint. URL: https://www.tutorialspoint.com/asp.net_core/index.htm (date of access: 20.05.2024).

6. ASP.NET Core Performance Best Practices – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/aspnet/core/performance/best-practices (date of access: 20.05.2024).

7. ASP.NET Core Security – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/aspnet/core/security (date of access: 20.05.2024).

8. ASP.NET Core Tutorials – DotNetTricks. DotNetTricks. URL: https://www.dotnettricks.com/learn/aspnet-core (date of access: 20.05.2024).

9. ASP.NET Core vs ASP.NET MVC – The Complete Guide – Stackify. Stackify. URL: https://stackify.com/aspnet-core-vs-aspnet-mvc (date of access: 20.05.2024).

10. Bogard J. ASP.NET Core in Action, 2nd ed. Manning Publications, 2021. 832 p.

11. Building Web Applications with ASP.NET Core MVC – Pluralsight. Pluralsight. URL: https://www.pluralsight.com/courses/asp-dot-net-core-mvc-building-web-applications (date of access: 20.05.2024).

12. Burns B. ASP.NET Core 5 and Angular: Full-stack web development with

.NET 5 and Angular 11. Packt Publishing, 2021. 690 p.

13. Entity Framework Core – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/ef/core (date of access: 20.05.2024).

14. Esposito D., Santoro A. Modern Web Development with ASP.NET Core 3: An end to end guide covering the latest features of Visual Studio 2019, Blazor and Entity Framework, 2nd ed. Packt Publishing, 2020. 746 p.

15. Exploring ASP.NET Core Razor Pages – Learn.microsoft.com. Microsoft. URL: https://learn.microsoft.com/en-us/aspnet/core/razor-pages (date of access: 20.05.2024).

16. Fazio R., Pereyra A. ASP.NET Core 5 for Beginners: Kick-start your ASP.NET web development journey with the help of step-by-step tutorials and examples. Packt Publishing, 2020. 780 p.

17. Firtman M. ASP.NET Core 5 for Mobile Developers: Build cloud-connected mobile apps using C# and ASP.NET Core. Packt Publishing, 2020. 728 p.

18. Freeman A., Pro Angular 9. Apress, 2020. 800 p.

19. Freeman A., Sanderson A. Pro ASP.NET Core 3. Apress, 2019. 1080 p.

20. Getting Started with ASP.NET Core MVC – Learn.microsoft.com. Microsoft. URL: https://learn.microsoft.com/en-us/training/modules/build-web-api-aspnet-core (date of access: 20.05.2024).

21. Migrating from ASP.NET MVC to ASP.NET Core – Microsoft Docs. Microsoft. URL: https://docs.microsoft.com/en-us/aspnet/core/migration/proper-to-2x (date of access: 20.05.2024).

22. Murach J. Murach's ASP.NET Core MVC. Mike Murach & Associates, 2020. 886 p.

23. Porebski T., De Carvalho P., Esposito D. Programming ASP.NET Core, 2nd ed. O'Reilly Media, 2021. 1024 p.

24. Shah A. Building Web Applications with ASP.NET Core MVC, 2nd ed. Springer, 2019. 512 p.

25. Understanding ASP.NET Core Middleware – Code Maze. Code Maze. URL: https://code-maze.com/aspnetcore-middleware (date of access: 20.05.2024).

26. Грищенко М.С. Веб-розробка на основі ASP.NET Core MVC. Харків, 2023. 320 с.

27. Іванов О.М. Бази даних: проектування, реалізація та адміністрування. Харків, 2022. 576 с.

28. Коваленко І.П. Інтернет речей: концепції, технології та застосування. Дніпро, 2023. 352 с.

29. Коваленко І.П. Машинне навчання та штучний інтелект. Львів, 2022. 512 с.

30. Кучеренко О.М. Кібербезпека: захист інформаційних систем. Дніпро, 2020. 384 с.

31. Петренко О.В. Інформаційні системи: концепції та проектування. Київ, 2020. 456 с.

32. Петров С.В. Хмарні обчислення: принципи та практики. Київ, 2021. 416 с.

33. Пономаренко А.І. Інформаційні технології в бізнесі. Одеса, 2020. 320 с.

34. Сидоренко А.І. Розробка мобільних додатків на Android. Одеса, 2021. 288 с.

35. Сидоренко М.В. Захист даних та інформаційна безпека. Львів, 2019. 400 с.