МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАСОБАМИ .NET»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Б.В. Володський
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Володському Богдану Володимировичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи засобами .NET

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка інформаційної системи засобами .NET.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 03.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.05.2023	
2.	Збір вихідних даних.	20.06.2023	
3.	Обробка методичних та теоретичних	10.07.2023	
	джерел.		
4.	Розробка першого та другого розділу.	25.09.2023	
5.	Розробка третього розділу.	30.10.2023	
6.	Оформлення та нормоконтроль	20.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	14.12.2023	

Студент _____ Б.В. Володський
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи засобами .NET»: 54 с., 16 рис., 4 табл., 11 джерел, 3 додатки.

АРХІТЕКТУРА, БІБЛІОТЕКА КЛАСІВ, ВЕБІНТЕРФЕЙС, ВІРТУАЛЬНА МАШИНА CLR, ДИНАМІЧНИЙ ДИЗАЙН, ЕФЕКТИВНІСТЬ, ЗРУЧНИЙ ІНТЕРФЕЙС, ІНФОРМАЦІЙНА СИСТЕМА, .NET FRAMEWORK.

Об’єкт дослідження – інформаційна система з використанням засобів .NET Framework.

Мета роботи – створення ефективної та надійної інформаційної системи за допомогою інструментів, наданих .NET Framework, яка відповідає вимогам сучасного бізнесу та забезпечує оптимальне використання можливостей .NET технологій.

Метод дослідження – створення вебсайту блогу за допомогою технологій .NET для обговорення та публікації інформації, пов'язаної з розробкою інформаційних систем.

Розроблений вебсайт став важливим ресурсом для спільноти фахівців та студентів, які цікавляться розробкою інформаційних систем з використанням .NET. Завдяки зручному інтерфейсу та функціоналу, сайт отримав позитивний відгук спільноти та став ефективним каналом обміну ідеями та досвідом.

Використані технології включали ASP.NET, MVC, Entity Framework та Bootstrap з метою забезпечення адаптивного та зручного дизайну. Функціонал включав можливість створення, редагування постів, залишання коментарів та категоризацію матеріалів для зручного пошуку.

SUMMARY

Master’s qualifying paper «Development of a Information System Using .NET»: 54 pages, 16 figures., 4 tables, 11 references, 3 supplements.

ARCHITECTURE, CLASS LIBRARY, CLR VIRTUAL MACHINE, DYNAMIC DESIGN, EFFICIENCY, .NET FRAMEWORK, INFORMATION SYSTEM, USER-FRIENDLY INTERFACE, WEB INTERFACE.

The research object is an information system using .NET Framework tools.

The goal of the work is to create an effective and reliable information system using the tools provided by the .NET Framework, which meets the requirements of modern business and ensures optimal use of the capabilities of .NET technologies.

The research method is the creation of a blog website using .NET technologies to discuss and publish information related to the development of information systems.

The developed website has become an important resource for the community of professionals and students interested in developing information systems using .NET. Thanks to the convenient interface and functionality, the site received positive feedback from the community and became an effective channel for exchanging ideas and experiences.

The technologies used included ASP.NET MVC, Entity Framework and Bootstrap to ensure a responsive and user-friendly design. Functionality included the ability to create, edit posts, leave comments and categorize materials for easy searching.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві, де конкуренція та швидкі технологічні зміни визначають умови бізнесу, важливою стає роль ефективних інформаційних систем. Одним із ключових чинників успіху у цьому контексті є використання сучасних технологій розробки програмного забезпечення. У рамках цього дослідження фокус приділяється .NET Framework – технологічному стеку, який визначається своєю потужною та гнучкою платформою для створення різноманітних застосунків [3].

Об'єктом вивчення є розробка інформаційної системи з використанням .NET Framework. Ця робота націлена на розкриття можливостей цієї технології в контексті розробки програмного забезпечення, спрощення процесів створення та підвищення продуктивності. Починаючи з огляду ключових переваг .NET Framework, ми глибше дослідимо основні аспекти розробки інформаційних систем на цій платформі.

Докладніше буде розглянуто архітектурні особливості, інструменти для розробки вебдодатків, мовну незалежність та інші фундаментальні аспекти. Подальше дослідження буде підкріплене прикладом успішної розробки вебсайту блогу, де використання .NET технологій виявилося важливим фактором у досягненні поставлених завдань [2].

Основними завданнями цього дослідження є:

- дослідження .NET Framework – провести глибокий аналіз технологічного стеку .NET Framework, визначивши його ключові складові, можливості та переваги для розробки інформаційних систем;
- розгляд архітектурних аспектів – вивчити архітектурні принципи, які лежать в основі .NET Framework, зокрема віртуальну машину CLR та бібліотеку класів, щоб зрозуміти їх вплив на розробку програмного забезпечення;
- аналіз інструментів розробки – оглянути різноманітні інструменти, які

надає .NET Framework для розробки вебдодатків, включаючи ASP.NET та інші технології;

– вивчення мовної незалежності – з'ясувати, як .NET Framework дозволяє розробникам використовувати різні мови програмування для створення одного програмного продукту;
– створення інформаційної системи – розробити прототип інформаційної системи з використанням .NET технологій, зосереджуючись на визначених аспектах архітектури та ефективних інструментах розробки;
– оцінка впливу .NET технологій – провести оцінку впливу використання .NET Framework на ефективність, безпеку та інші аспекти розробленої інформаційної системи;
– висновки та рекомендації – сформулювати висновки щодо результатів дослідження, надати рекомендації щодо використання .NET технологій у розробці інформаційних систем [3].

1 ВСТУП ДО .NET FRAMEWORK ТА ОБҐРУНТУВАННЯ ДОСЛІДЖЕННЯ

1.1 Визначення .NET Framework

.NET Framework є ключовою технологічною платформою, розробленою корпорацією Microsoft, яка надає високорівневі середовища та інструменти для розробки різноманітних програмних додатків. Ця платформа включає в себе ряд інструментів, що спрощують розробку, розгортання та управління програмами в середовищі Windows.

Віртуальна машина CLR (Common Language Runtime): Однією з ключових складових є CLR, яка відповідає за виконання програм на мові, що компілюється в інтермедіатний байт-код (IL). CLR надає середовище виконання, керує пам'яттю та забезпечує інші сервіси для підтримки виконання програм [9].

Бібліотека класів .NET (Class Library): Вона включає набір готових класів та функцій, які полегшують стандартні завдання програмування. Розробники функціональностей без необхідності писати код з нуля.

.NET Framework підтримує різні мови програмування, такі як C#, Visual Basic, F#, інші. Це означає, що розробники можуть вибирати мову, яка найкраще

відповідає їхнім потребам та експертизі, і все одно взаємодіяти зі спільними компонентами [11].

NET Framework забезпечує зручні засоби інтеграції з іншими технологіями, а також прості механізми розгортання додатків. Завдяки механізмам ASP.NET та Windows Forms, розробники можуть створювати як настільні, так і вебдодатки [8].

Продуктивність: висока швидкодія та оптимізація виконання коду.

Безпека: вбудовані механізми для керування доступом та інші заходи безпеки.

Розширюваність: здатність взаємодіяти з іншими технологіями та мовами

програмування.

.NET Framework став популярним інструментом для розробки програмного забезпечення завдяки своїй ефективності, гнучкості та широкому функціоналу. У наступних розділах буде детально розглянуто архітектурні аспекти та інструменти розробки, які створюють базу для подальших досліджень.

1.2 Важливість .NET Framework у розробці

.NET Framework є невід'ємною частиною сучасного ландшафту програмування, і його значення визначається не лише технічними можливостями, але й стратегічним впливом на процес розробки програмного забезпечення.

З одного боку, розширені можливості розробки, які надає .NET Framework, створюють можливість для розробників реалізувати широкий спектр інноваційних ідей. Від мобільних додатків до великих корпоративних систем – платформа надає високорівневі засоби для творчості [8].

З іншого боку, зручний інтерфейс розробки та інтеграція з іншими технологіями забезпечують не лише ефективність роботи розробників, але й високий рівень адаптованості до сучасних вимог у сфері програмної інженерії.

Імплементація .NET Framework в програмній практиці виявляється як стратегічне рішення через свою важливу роль у побудові інформаційних систем. Зручність розробки завдяки інтегрованому середовищу Microsoft Visual Studio та високорівневі мови програмування, такі як C# та Visual Basic, створюють простір для творчості та впровадження ідей [8].

Однак важливість .NET Framework йде далі ніж лише технічні аспекти. Вона полягає у створенні інновацій та визначенні нових стандартів у розробці програмного забезпечення. Враховуючи широкий спектр сценаріїв використання та його роль у побудові корпоративних рішень, можна стверджувати, що вибір .NET Framework – це стратегічне вирішення, яке визначає подальший успіх

проєктів та їх відповідність вимогам конкурентоспроможності.

Нижче представлена таблиця 1.1 порівняння між фреймворком .NET та бібліотекою React, які використовуються для розробки програмного забезпечення. Важливо враховувати, що .NET – це ширший фреймворк, який включає різноманітні інструменти для розробки різних типів додатків, включаючи веб, мобільні та стільникові додатки. З іншого боку, React – це бібліотека JavaScript, спрямована на розробку інтерфейсів користувача, зокрема вебдодатків.

Таблиця 1.1 – Порівняння .NET Framework та React

Особливості	**.NET Framework**	**React**
Мови програмування	Підтримує різні мови , такі як C#, VB.NET	Використовує JavaScript та JSX
Тип додатків	Призначений для розробки різноманітних додатків: веб, мобільні, декстопні	Спеціалізований для розробки інтерфейсів користувача вебдодатків
Архітектура	Забезпечує обширні можливості для створення різноманітних архітектур: MVC, MVVM	Зорієнтований на створення компонентів і заснований на концепції віртуального DOM
Використання	Широко використовується для розробки корпоративних та великих додатків	Особливо популярний в розробці сучасних інтерфейсів вебдодатків
Спільнота та екосистема	Має велику та розширену спільноту, багато інструментів та бібліотек	Має активну спільноту та багатий екосистему React-додатків

1.3 Очікувані результати та значення дослідження

Виходячи із поставлених цілей, мети та конкретних завдань дослідження, наша увага зосереджена на отриманні конкретних результатів, які можуть мати значущий вплив на розробку інформаційних систем за допомогою .NET Framework. Зазначимо ключові аспекти, на які ми спрямовуємо наше дослідження.

Огляд архітектурних особливостей .NET Framework:

- визначення та розкриття тих аспектів архітектури, які можуть бути вирішальними для розробників при створенні складних інформаційних систем;
- аналіз продуктивності та інструментів розробки;
- визначення ефективності інтегрованого середовища розробки Microsoft Visual Studio та інших інструментів, а також їх вплив на загальну продуктивність розробників [3].

Мовна незалежність та інтеграція:

- оцінка гнучкості та зручності використання різних мов програмування під час розробки, а також здатності платформи інтегруватися з іншими технологіями.

Створення прототипу та його оцінка:

- розробка прототипу інформаційної системи, яка дозволить нам оцінити простоту розробки, функціональність та продуктивність платформи у конкретному контексті.

Аналіз переваг та недоліків:

- визначення конкретних переваг та обмежень .NET Framework для розробки інформаційних систем, що може бути важливим для прийняття стратегічних рішень.

Ці очікувані результати та оцінки мають на меті надати інформаційну основу для прийняття розумних та ефективних рішень у виборі технологій розробки програмного забезпечення з використанням .NET Framework [3].

1.4 Роль та перспективи .NET Framework у розробці блогового сайту

Використання .NET Framework у розробці блогового сайту визначає ряд важливих ролей та відкриває перспективи, які можуть значно покращити якість та ефективність вебпроєкту.

Роль .NET Framework:

- потужні інструменти розробки: .NET Framework надає розробникам потужні та високорівневі інструменти, спрощуючи процес створення динамічних вебсайтів (засоби, такі як ASP.NET, дозволяють швидко розробляти сторінки та взаємодіяти з базою даних);
- зручна інтеграція з іншими технологіями: .NET Framework легко інтегрується з іншими технологіями та сервісами, що робить його універсальним рішенням для розробки вебсайтів різного типу (його сумісність із сервісами хмарного зберігання та іншими рішеннями дозволяє легко розширювати функціональність блогу [2]);
- безпека та керування ідентифікацією: вбудовані механізми безпеки .NET Framework гарантують надійний захист вебдодатка від потенційних загроз (системи керування ідентифікацією дозволяють контролювати доступ до різних частин блогу та забезпечують безпеку користувачів);
- підтримка масштабованості: .NET Framework дозволяє масштабувати блоговий сайт відповідно до зростання аудиторії (засоби оптимізації продуктивності та ефективне управління ресурсами гарантують стабільну роботу вебдодатка навіть при великих навантаженнях).

Перспективи використання .NET Framework у майбутньому:

- розвиток та підтримка: .NET Framework продовжує активно розвиватися, отримуючи нові оновлення та функціональні можливості (це забезпечує тривалу підтримку та доступ до останніх технологічних тенденцій для блогового сайту) [2];
- інтеграція з новими технологіями: перспективи включають інтеграцію

.NET Framework з новими технологіями, такими як штучний інтелект, блокчейн або розширена реальність (це може привести до створення більш інноваційних та функціональних можливостей для блогу);

– збільшення зручності зозробки: очікується подальше спрощення процесу розробки за допомогою .NET Framework, зокрема, розширення функціональності та оптимізація інструментів розробки.

1.5 Інформаційні потоки

При роботі будь-якого вебсайту існують два ключових типи інформаційних потоків: зовнішні та внутрішні. Зовнішні потоки стосуються взаємодії користувача з самим сайтом. Коли користувач взаємодіє з вебсайтом, наприклад, заповнює форми, натискуючи кнопки чи переходячи між сторінками, це є зовнішнім потоком. Користувачі надсилають свої запити до сервера, щоб отримати різноманітну інформацію, таку як сторінки, файли або відповіді на конкретні запити.

З іншого боку, внутрішні потоки пов'язані з сервером, який обробляє ці зовнішні запити та надсилає відповіді назад користувачам. Наприклад, коли ви натискайте на кнопку, сервер обробляє цей запит, виконує необхідні операції та надсилає відповідь, яка може бути відображена на сторінці.

Отже, внутрішні та зовнішні інформаційні потоки є важливими компонентами функціонування будь-якого вебсайту. Зовнішні потоки дозволяють користувачам взаємодіяти з сайтом, використовуючи його інтерфейс та функціонал, в той час як внутрішні потоки дозволяють серверу ефективно обробляти та відповідати на ці взаємодії. Це взаємодія є критично важливою для того, щоб користувачі могли отримувати необхідну інформацію та використовувати функціонал сайту зручно та ефективно.

1.6 Висновки до розділу 1

Розділ 1 відкрив широкий огляд важливості та перспектив використання .NET Framework у розробці блогового сайту. Підкреслено, що .NET Framework не лише є потужним інструментом для втілення амбіційних проєктів веброзробки, але й стає ключовою складовою для створення високоякісних та інноваційних вебдодатків.

Аналіз можливостей .NET Framework виявив, що його продуктивність та ефективність роблять його надзвичайно потужним інструментом для будь-яких вебпроєктів. Здатність легко впроваджувати нові ідеї та технології, а також гнучкість для розвитку індивідуальних проєктів роблять .NET Framework невід'ємною частиною сучасного веброзробницького середовища.

Цей розділ наголосив на важливості вибору .NET Framework для успішного розвитку та вдосконалення блогового сайту, підкреслюючи його здатність задовольняти потреби різноманітних проєктів у сфері веброзробки. Інтеграція .NET Framework у вебпроцес робить його не просто інструментом, але ключовим партнером у створенні інноваційних та ефективних вебрішень.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Постановка завдання

Проєкт розробки вебдодатка на основі ASP.NET Core та Razor Pages має на меті створення платформи для створення та перегляду блогів. Основні вимоги та функціональність описані нижче:

Користувачі та авторизація:

– можливість реєстрації нових користувачів;
– авторизація користувачів для доступу до особистого кабінету;
– різні рівні доступу (користувач, автор, адміністратор).

Створення та редагування блогів:

– форма для створення нових записів блогу;
– можливість додавання тексту, зображень та інших медіафайлів до записів;
– редагування та видалення власних записів.

Відображення блогів:

– список всіх блогів на головній сторінці;
– детальний перегляд окремого блогу разом із списком його записів.

Коментарі:

– можливість залишати коментарі під кожним записом блогу;
– модерація коментарів або можливість встановлення фільтрів для уникнення спаму.

Пошук та фільтрація:

– пошук за ключовими словами в записах блогів;
– фільтрація записів за категоріями або тегами.

SEO-оптимізація:

– дружні URL-адреси для кожного запису блогу;
– мета-теги для покращення індексації пошуковими системами.

– підтримка медіафайлів:
– завантаження та зберігання зображень та інших медіафайлів;
– автоматичне масштабування та обрізка зображень.

Адміністративний розділ:

– доступ до адміністративного розділу для адміністраторів;
– можливість управління користувачами, блогами та коментарями.

Аналітика:

– збір та відображення базових аналітичних даних (кількість переглядів, коментарі, тощо).

Мобільна сумісність:

– адаптивний дизайн для зручного перегляду на різних пристроях.

Безпека:

– захист від атак, таких як SQL-ін'єкції та перехоплення сесій.

Це базовий список функціональності, і ви можете розширити його відповідно до конкретних вимог вашого проєкту. З використанням ASP.NET Core і Razor Pages ви матимете доступ до потужних інструментів для реалізації цих функцій.

2.2 Вимоги до інтерфейсу

Інтерфейс вебдодатка розроблено з урахуванням простоти використання та ефективності для користувача. Основні аспекти інтерфейсу відповідають вимогам зручності та доступності:

Форма введення блогу:

– наявність полів для введення заголовку, опису, URL зображення, посилання та автора;
– максимальна довжина заголовку обмежена 40 символами;
– зображення та URL перевіряються на доступність перед відображенням [3].

Відображення списку блогів:

– блоги відображаються в зворотньому порядку (від нових до старих);
– для кожного блогу вказується заголовок, короткий опис та, якщо є, зображення.

Повний перегляд блогу:

– при відсутності зображення виводиться тільки текст блогу, інакше виводиться і зображення та текст;
– посилання на джерело блогу вказується, якщо воно доступне.

Взаємодія з блогами:

– кожен блог може бути відредагований або видалений за допомогою відповідних кнопок;
– відображається ім'я автора блогу, або "Anon", якщо ім'я відсутнє.

Форма редагування блогу:

– форма редагування виводиться при натисканні кнопки "Edit";
– заповнення форми попередньо ініціалізується даними блогу для зручності редагування.

Інші функціональні елементи:

– кнопка "Create new blog" виводить форму для створення нового блогу;
– відображається повідомлення про відсутність доступу до бази даних у разі її відсутності.

Загальна концепція інтерфейсу спрямована на забезпечення зручності введення, перегляду та взаємодії з блогами, зазначаючи важливі елементи інформації та забезпечуючи простоту користування.

2.3 Вимоги до продуктивності

Для досягнення високої продуктивності вебдодатка визначено декілька важливих аспектів. Проєкт приділяє увагу оптимізації швидкості завантаження сторінок та реакції інтерфейсу на дії користувача. Забезпечення ефективності

бази даних, оптимізація передачі даних та мінімізація завантаження ресурсів – основні аспекти, на які акцентується розробка.

Використання кешування та зменшення кількості запитів до сервера сприяє підвищенню швидкодії завантаження сторінок. Також розроблено механізми асинхронного завантаження контенту, що сприяє відчутному покращенню реакції інтерфейсу на дії користувача.

Оптимізація роботи зображень та перевірка доступності URL сприяють ефективному відображенню блогів, забезпечуючи швидке завантаження контенту. Використання асинхронних запитів та оптимальних методів обробки даних зменшує час очікування відповіді від сервера.

Узагальнюючи, архітектурні рішення та оптимізації, впроваджені у вебдодатку, спрямовані на максимізацію продуктивності та забезпечення швидкодії реакції інтерфейсу, що призводить до покращення загального користувацького досвіду.

2.4 Вибір середовища програмування для серверної частини

Середовищем функціонування сервера було обрано ASP.NET.

Нижче наведена таблиця 2.1 порівняння середовища функціонування ASP.NET та інших з деякими іншими середовищами, як наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння середовища функціонування ASP.NET

Функція/особливість	ASP.NET	Ruby
Мова програмування	C# або VB.NET	Ruby
Тип	Серверна технологія	Серверна технологія
Асинхронність	Повністю асинхронний	Асинхронні опції зі сторонніми бібліотеками
Спрощена обробка HTTP	Вбудований http модуль, або сторонні бібліотеки	Ruby on Rails фреймворк

Продовження табл. 2.1

Пакетний менеджер	NuGet	RubyGems
Ком’юніті та екосистема	Активна участь в розвитку, регулярні випуски нових версій, підтримка спільноти.	Активна спільнота, багато готових рішень у вигляді гемів
Платформозалежність	Кросплатформений	Кросплатформений

Отже, обравши ASP.NET для серверної частини свого програмного забезпечення, користувач обраєплатформу, яка добре впорається з великою кількістю одночасних підключень та обробкою значних обсягів даних.

ASP.NET має вбудований не блокуючий ввід/вивід (non-blocking I/O) підхід, що робить його особливо ефективним у випадках, коли сервер повинен обробляти багато запитів паралельно [3] (як наведено у табл. 2.1).

Цей підхід до асинхронного програмування, де код може продовжувати виконуватися, не чекаючи завершення операцій вводу/виводу (I/O), стає ключовим для створення ефективних та високопродуктивних серверів.Зокрема, коли маєш справу з великою кількістю одночасних підключень в реальному часі, наприклад, при роботі з багатокористувацькими вебзастосунками або мережевими додатками.

ASP.NET також дозволяє легко масштабувати додатки за допомогою кластеризації та інших методів, що робить його зручним вибором для розробки серверної частини додатків, які мають багато користувачів та великі обсяги даних [3].

Загалом, обрання ASP.NET вказує на те, що розробник приділяє увагу ефективності, швидкості та масштабованості у веброзробці, особливо в умовах великої кількості одночасних запитів [3].

Підкреслюючи основні аспекти вашого висновку:

Не блокуючий ввід/вивід (non-blocking I/O): ASP.NET використовує асинхронний підхід до програмування, що дозволяє ефективно обробляти багато

запитів паралельно, особливо в ситуаціях з великою кількістю одночасних підключень.

Масштабованість: ASP.NET надає можливості для легкої масштабованості додатків, зокрема за допомогою кластеризації, що робить його відмінним вибором для вебзастосунків з великою кількістю користувачів та обсягами даних [5].

Швидкодія: асинхронний підхід та оптимізації ASP.NET спрямовані на досягнення високої продуктивності та швидкодії, що робить його ефективним для обробки запитів у реальному часі.

Можливості масштабування на рівні кластера: кластеризація та інші методи масштабування ASP.NET дозволяють легко розширювати додатки для відповіді на зростання користувачів та завдань [5].

Обираючи ASP.NET для серверної частини програмного забезпечення, розробник визначається з потужним та ефективним інструментарієм для реалізації вимог щодо високої продуктивності та обробки багатьох запитів одночасно.

2.5 Обгрунтування вибору середовища розробки системи

Для зручної та більш продуктивної розробки проєкту була обрана програма Visual Studio Code (VS Code).

Обираючи Visual Studio Code (VS Code) для розробки проєкту, розробник обирає популярний редактор вихідного коду, який вражає своєю зручністю та здатністю прискорювати робочий процес.

Такі основні характеристики, як кросплатформеність, легкість та продуктивність, роблять його популярним серед розробників вебдодатків, як наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння Visual Studio Code з іншим редактором

Особливість	VS Code	SublimeText
Кросплатформеність	Так	Так
Легкість та продуктивність	Швидкий запуск, низькі вимоги до ресурсів	Швидкий та ефективний, малі вимоги до ресурсів
Вбудовані інструменти	Великий вибір вбудованих інструментів, включаючи вкладчик, Git інтеграцію, підсвічування синтаксису	Зручний інтерфейс для роботи з Git, можливості підсвічування синтаксису
Розширюваність	Велика кількість розширень у магазині розширень	Розширення та пакети підтримуються, але не настільки широко як у VS Code
Відкритий вихідний код	Так	Ні
Інтеграція з іншими інструментами	Широкі можливості для інтеграції з різними інструментами розробки	Інтеграція з багатьма інструментами, але може вимагати деяких налаштувань

2.6 Обґрунтування вибору середовища функціонування системи

Веббраузери в сучасному інтернет-світі відіграють невід'ємну роль, будучи ключовим інструментом для доступу до вебдодатків та ресурсів Інтернету.

Веббраузери служать невід'ємною частиною нашої онлайн-експерієнції, дозволяючи нам переглядати вебсторінки, спілкуватися в соціальних мережах, використовувати вебпродукти та робити покупки в Інтернеті. Вони

використовуються для відображення змісту, розробки та запуску вебдодатків.

Важливою особливістю веббраузерів є їхні можливості взаємодії з вебдодатками, які використовують різноманітні технології, такі як HTML, CSS, JavaScript та інші. Браузери мають дві основні функції: завантаження та відображення вебсторінок та виконання скриптів, що дозволяє робити сторінки динамічними та взаємодійними.

Розроблений проєкт є високопереносним, оскільки призначений для функціонування на різних операційних системах, таких як Windows, Mac OS X, Linux, OS/2, та інші, якщо пристрій має браузер. Це відкриває можливість для користувачів з різних платформ отримати доступ до проєкту та використовувати його зручно та ефективно [8].

Основні цільові браузери, такі як Google Chrome, Safari, Opera і Firefox, відіграють ключову роль у проєкті. Обрання саме цих браузерів є стратегічним рішенням, оскільки вони належать до найпопулярніших та широко використовуваних. Це дозволяє забезпечити широку сумісність та надати найкращий користувацький досвід для багатьох користувачів.

2.7 Функціональність системи

Вебдодаток реалізовано з врахуванням різноманітних функцій для зручного та ефективного управління блогами. Основна функціональність охоплює наступні аспекти:

- додавання нових блогів (див. рис 2.1) – користувач може вводити дані нового блогу через форму, вказуючи заголовок, опис, URL зображення, посилання та автора (опціонально);
- редагування та видалення блогів (див. рис. 2.2) – користувач може редагувати та видаляти блоги, натисканням відповідних кнопок;
- повний перегляд блогу (див. рис. 2.3) – користувач може переглядати весь вміст блогу, включаючи зображення, опис та іншу інформацію;

– перевірка доступності URL та зображень (див. рис. 2.4) – використано функцію «CanFetch» для перевірки доступності URL та зображень перед їх відображенням на сторінці.

```
const postNewBlog = async (title, description, img, url, maker) => {

    const response = await fetch(databaseURL);
    if (!response.ok) {
        throw new Error("Could not fetch data from that resource");
    }
    const existingBlogs = await response.json();
    const maxId = Math.max(...existingBlogs.map(blog => blog.id));
    let id = maxId + 1;
    const newBlog = { id, title, description, img, url, maker };
    console.log(newBlog);
    fetch(databaseURL, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(newBlog),
    }).then((res) => {
        if (!res.ok) {
            throw Error("could not fetch data from that resource");
        }
        console.log("Blog added");
        location.reload()
    });
};
```

Рисунок 2.1 – Код функції створення нового блогу

Одним із сценаріїв використання fetch-запитів може бути перевірка доступності URL та зображень перед їх відображенням на сторінці. Це може бути корисно для оптимізації завантаження сторінок та покращення користувацького досвіду.

Також, Сучасні браузери накладають обмеження на Fetch API для забезпечення безпеки. Запити, виконані зі сторінок, які використовують HTTPS, можуть взаємодіяти лише з іншими HTTPS-ресурсами (не забороненими за політикою Same-Origin Policy). Це покращує безпеку передачі даних [11].

```
const updatePostedBlog = (title, description, img, url, maker) => {
    const blog = {selectedId, title, description, img, url, maker};

    console.log(blog);

    fetch(databaseURL + `/${selectedId}`, {
        method: "PATCH",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(blog),
    }).then((res) => {
        if (!res.ok) {
            throw Error("could not fetch data from that resource");
        }
        console.log("Blog updated");
        location.reload()
    });
};

const deleteBlog = (idToDelete) => {
    let confirmMenu = window.confirm("Are you sure you want to delete this blog?");

    if (confirmMenu) {
        fetch(databaseURL + `/${idToDelete}`, {
            method: "DELETE",
        }).then((res) => {
            if (!res.ok) {
                throw Error("could not redact data");
            }
            console.log("Blog deleted");
            location.reload()
        });
    }
}
```

Рисунок 2.2 – Код функції відправлення редагованогоблогу та видалення блогів

```
@section Scripts {
    <script>
        const blog = @Html.Raw(Json.Serialize(Model.NeededBlog));
    </script>
}

@if (Model.NeededBlog is object)
{
    <div class="container d-flex flex-column gap-3 justify-content-center">
            <section class="full-card-with-blog">
                @if (!string.IsNullOrEmpty(@Model.NeededBlog.Image))
                {
                    <div style="background-image: url('@Model.NeededBlog.Image');" class="full-blog-img">
                        <div class="title-img text-white-with-bg">
                            <h5 class="m-0 fs-4">@Model.NeededBlog.Title</h5>
                        </div>
                    </div>

                }
                else
                {
                    <div class="title-img">
                        <h5 class="m-0 fs-4">@Model.NeededBlog.Title</h5>
                    </div>
                }

            <div class="full-text pb-0">
                    <div class="d-flex flex-column justify-content-start align-items-center gap-2">
                        <p class="full-description-width m-0">
                            @Model.NeededBlog.Description
                        </p>
                    @if (!string.IsNullOrEmpty(@Model.NeededBlog.Maker))
                        {
                        <p class="fw-bold m-0">Writed by: @Model.NeededBlog.Maker</p>
                        }
                        else
                        {
                            <p class="fw-bold m-0">Writed by: Anon</p>
                        }
                    </div>

                </div>
                <div class="d-flex flex-row mx-3 mb-2 gap-3 justify-content-center">
                    @if (!string.IsNullOrEmpty(@Model.NeededBlog.Url))
                        {
                            <a href="@Model.NeededBlog.Url" class="link-violet">Pined link</a>
                        }
                </div>
            </section>
        <a class="btn btn-primary" href="/">Back to home page</a>
    </div>
```

Рисунок 2.3 – Код сторінки перегляду блогу

```
public bool CanFetch(string imageUrl)
{
    using (HttpClient httpClient = new HttpClient())
    {
        try
        {
            HttpResponseMessage response = httpClient.GetAsync(imageUrl).Result;
            return response.IsSuccessStatusCode;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error fetching image: {ex.Message}");
            return false;
        }
    }
}
```

Рисунок 2.4 – Код функції перевірки доступності URL та зображень

Ця функціональність забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувача, щоб вони могли легко додавати, редагувати, видаляти та переглядати блоги на вебсайті (як наведено у рис. 2.4).

2.8 UML діаграма класів

UML (Unified Modeling Language) діаграма класів – це графічний інструмент, який використовується для візуалізації та моделювання структури програмного забезпечення.

У застосунку використовуються декілька класів, кожен з яких відповідає за якусь певну дію.

На рисунку 2.5 зображено взаємозв’язок класів у проєкті.

BlogCreateSection відображає компонент створення нового блогу в системі. Містить методи та властивості, які дозволяють вводити та зберігати інформацію для нового блогу, таку як заголовок, зміст, автора, картинку.

BlogController відповідає за обробку HTTP-запитів, що стосуються блогів. Містить методи відображення сторінок, створення нових блогів, відображення конкретного блогу.

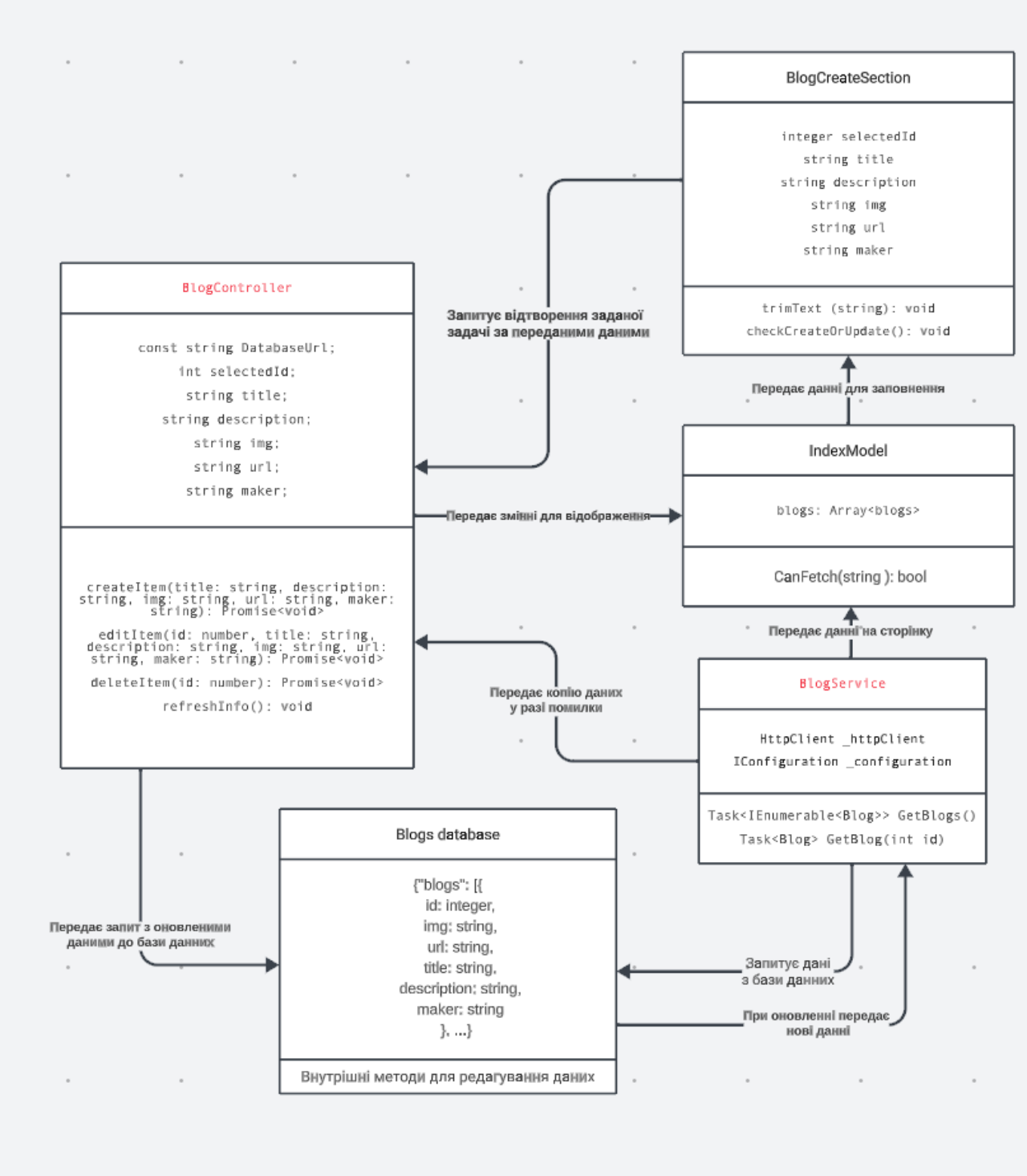


Рисунок 2.5 – UML діаграма класів

IndexModel представляє модель для відображення сторінки зі списком блогів або головної сторінки. Містить дані, які потрібно відобразити на цій сторінці, такі як список останніх блогів, категорії

BlogService представляє сервіс, який надає функціональність для роботи з блогами. Містить методи для створення, редагування, видалення блогів, а також

інші операції, пов’язані з управлінням блогами.

BlogDateBase представляє роботу з базою даних для зберігання блогів. Містить методи для взаємодії з базою даних, такі як додавання, оновлення, видалення блогів, а також запити для отримання блогів із бази даних.

Узагальнюючи, ці класи ілюструють важливі компоненти для створення та управління блогами в вебдодатку. Кожен клас виконує свою роль у системі та спільно сприяє правильному функціонуванню функціоналу для блогів як зображено на рисунку 2.5.

2.9 Діаграма розгортання

Діаграма розгортання – це вид діаграми, який вказує на те, як компоненти системи фізично розміщені та як вони взаємодіють між собою на рівні апаратного забезпечення.

На рисунку 2.6. зображена діаграма взаємодії бази даних, сервера та користувача (рис. 2.6).

Рисунок 2.6 – Діаграма розгортання

База даних фізично розташована на локальному сервері розгортання “Live server” та взаємодіє з іншими компонентами.

Сервер представлений локальний, на якому розгорнуті додатки та служби.

Функції користувача взаємодіють з сервером для надання запитів, отримання даних та виконання необхідних функцій.

Зв’язки – це лінії між компонентами, які вказують на напрямок взаємодії.

2.10 Висновки до розділу 2

У даному розділі проведено детальне проєктування вебдодатку, призначеного для універсального блогу, тобто користувач сам підлаштовується під себе та обирає власну тему для свого блогу. Під час аналізу вимог до системи були визначені ключові аспекти та їх ролі у вебзастосунку. Цей аналіз дозволив отримати чітке уявлення про потреби користувачів та взаємодії, які відбуватимуться в системі.

Аналіз вимог дозволяє також зрозуміти, як саме користувачі будуть взаємодіяти з системою та які функції їм будуть доступні.

Діаграма розгортання та діаграма UML, що були розроблені, детально відображають взаємодії, які відбуваються в системі, наприклад, між користувачем та сервером.

Отже, розділ з детальним проєктуванням визначив ключових учасників, їх функції та способи взаємодії в системі, створюючи тим самим фундамент для подальшого розроблення та впровадження вебзастосунку для ведення васного блогу.

Виразивши детальні аспекти взаємодії у системі, важливо відзначити, що глибоке проєктування вирізняє ключові елементи, які визначають успішну реалізацію вебдодатку. Одним із них є гнучкість, надана користувачам управління власним блогом та вибором тематики. Це дозволяє створювати унікальні та персоналізовані простори для вираження індивідуальності.

Враховуючи важливість забезпечення безпеки та вірогідності прав доступу до файлів, стає очевидним, що цей етап проєктування також враховує турботу про конфіденційність та цілісність інформації, яку користувачі довіряють системі.

Важливо наголосити, що визначені у розділі 2.10 аспекти створюють основу для подальшої розробки та імплементації вебзастосунку. Забезпечення доступності, ефективного обміну даними та врахування потреб користувачів дозволяють забезпечити високу якість вебдодатку та його успішне впровадження в реальне використання.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЗАСОБАМИ .NET

3.1 Проєктування дизайну застосунку

При розробці дизайну вебдодатка було акцентовано на створенні чистого та зрозумілого інтерфейсу для користувачів. Дизайн визначається мінімалізмом та простотою з метою покращення зручності використання та навігації.

Колірна палітра складається з приглушених тонів, що дозволяє виділяти ключові елементи та забезпечує приємний вигляд. Використання білого тла сприяє читабельності контенту та відокремленню елементів.

Зручність користування досягнута за допомогою логічного розташування елементів у вигляді блоків. Форма введення блогу та повна інформація про блог відображаються відмінно, забезпечуючи структурованість та легкість взаємодії.

Для кнопок та інших важливих елементів використано контрастні кольори для виділення та покращення їх видимості. Такий підхід робить функціональність додатка інтуїтивно зрозумілою для користувача.

Типографіка зосереджена на читабельності, використовуючи шрифти з добре визначеною структурою та виглядом.

Резюмуючи, дизайн вебдодатка спрямований на створення приємного та ефективного користувацького досвіду, забезпечуючи зручність взаємодії та чітку інформативність.

Дизайн розроблено з урахуванням адаптивності для різних пристроїв та розмірів екранів. Це забезпечує однаково зручний користувацький досвід як на комп'ютері, так і на мобільному пристрої.

Форми введення блогу та інші елементи мають інтуїтивно зрозумілі мітки та вказівки, що полегшує користувачам їх використання та заповнення.

Використання приємних кольорів та елементів дизайну спрямоване на викликання позитивних емоцій у користувачів, покращуючи їх враження від використання додатка.

На рис 3.1 зображена головна сторінка вебдодатку.

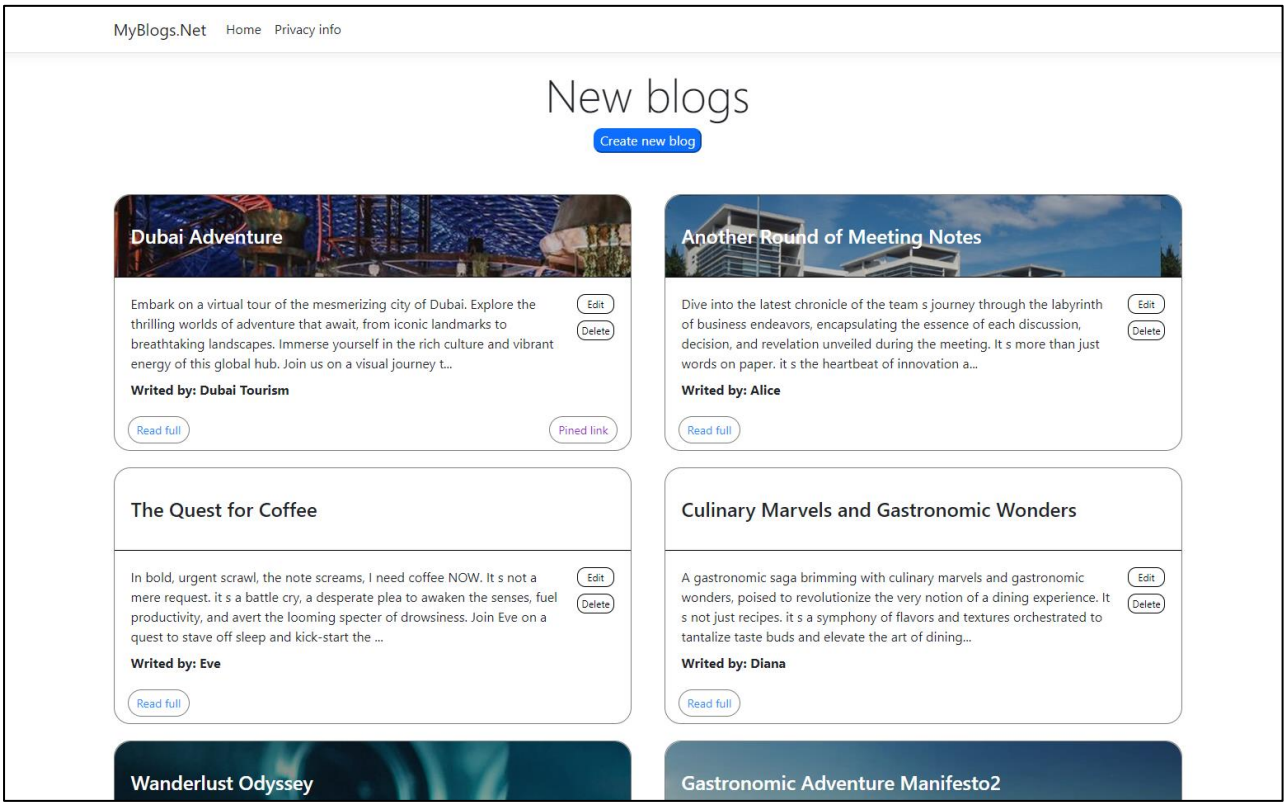


Рисунок 3.1 – Дизайн застосунку

3.2 Розгортання бази даних застосунку

Для розгортання бази даних вебдодатка використовується проста та ефективна стратегія. Даний застосунок використовує JSON-файл як базу даних, що зберігається локально на сервері. Це рішення є зручним для невеликих та середніх проєктів, де необхідно вести невеликий обсяг даних.

JSON (JavaScript Object Notation) є форматом обміну даними, який використовується для зберігання та передавання структурованих інформаційних об'єктів між комп'ютерами.

JSON є текстовим форматом, легким для людей читати і писати, а також легким для машин розуміти та обробляти. Він є часто використовуваним форматом для обміну даними вебдодатками.

JSON використовується для представлення структурованої інформації у вигляді пар ключ-значення, де ключі є рядками, а значення можуть бути рядками, числами, логічними значеннями, масивами, об'єктами або null.

JSON-файл містить інформацію про блоги, а його структура відповідає моделі блогу в застосунку. Це дозволяє легко зберігати та отримувати дані без необхідності використання складних систем управління базами даних.

Json-файл містить вміст даних розробленого проєкту, а саме:

– картинка блогу;
– посилання;
– назва блогу;
– опис блогу.
– ім’я користувача.

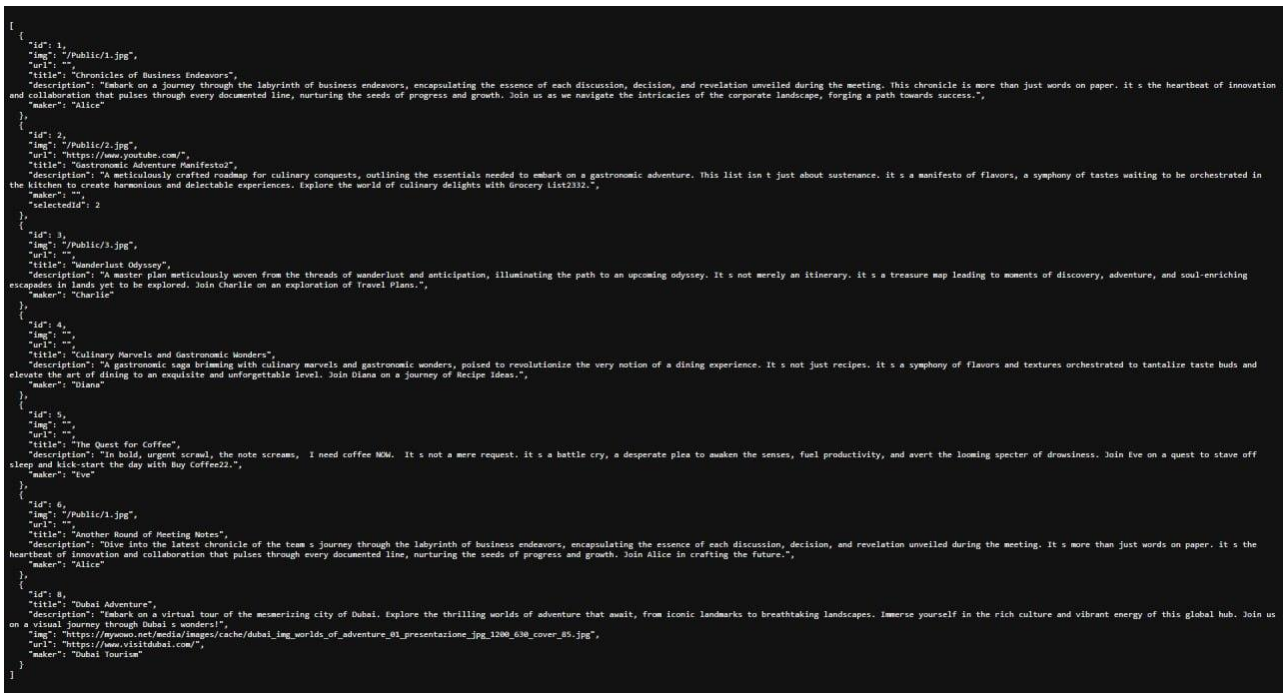
Під час розгортання додатку достатньо переконатися, що JSON-файл доступний для читання та запису на сервері за вказаною адресою. Додаток використовує HttpClient для здійснення запитів до цього файлу та обміну даними.

Важливим аспектом при розгортанні бази даних є забезпечення відповідності безпеці та правильності прав доступу до файлу, щоб уникнути несанкціонованого доступу чи втрати даних.

Такий підхід дозволяє швидко та ефективно розгортати базу даних, зменшуючи складність конфігурації та управління, що є особливо важливим для невеликих проєктів.

Розгортання бази даних у цьому вебдодатку обумовлено переконанням у доступності JSON-файлу для читання та запису на сервері за конкретною адресою. Використання HttpClient взаємодіє з файлом, забезпечуючи ефективний обмін даними між вебдодатком та базою даних.

Також важливою частиною цього процесу є гарантування безпеки та коректності прав доступу до файлу, щоб уникнути неправомірного доступу чи можливої втрати інформації. Такий підхід дозволяє оперативно та ефективно розгортати базу даних, спрощуючи конфігурацію та управління, що виявляється особливо корисним для розробки проєктів невеликого масштабу (див. рис. 3.2).



```
[
  {
    "id": 1,
    "img": "/Public/1.jpg",
    "url": "",
    "title": "Chronicles of Business Endeavors",
    "description": "Embark on a journey through the labyrinth of business endeavors, encapsulating the essence of each discussion, decision, and revelation unveiled during the meeting. This chronicle is more than just words on paper. it s the heartbeat of innovation and collaboration that pulses through every documented line, nurturing the seeds of progress and growth. Join us as we navigate the intricacies of the corporate landscape, forging a path towards success.",
    "maker": "Alice"
  },
  {
    "id": 2,
    "img": "/Public/2.jpg",
    "url": "https://www.youtube.com/",
    "title": "Gastronomic Adventure Manifesto2",
    "description": "A meticulously crafted roadmap for culinary conquests, outlining the essentials needed to embark on a gastronomic adventure. This list isn t just about sustenance. it s a manifesto of flavors, a symphony of tastes waiting to be orchestrated in the kitchen to create harmonious and delectable experiences. Explore the world of culinary delights with Grocery List2332.",
    "maker": "",
    "selectedId": 2
  },
  {
    "id": 3,
    "img": "/Public/3.jpg",
    "url": "",
    "title": "Wanderlust Odyssey",
    "description": "A master plan meticulously woven from the threads of wanderlust and anticipation, illuminating the path to an upcoming odyssey. It s not merely an itinerary. it s a treasure map leading to moments of discovery, adventure, and soul-enriching escapades in lands yet to be explored. Join Charlie on an exploration of Travel Plans.",
    "maker": "Charlie"
  },
  {
    "id": 4,
    "img": "",
    "url": "",
    "title": "Culinary Marvels and Gastronomic Wonders",
    "description": "A gastronomic saga brimming with culinary marvels and gastronomic wonders, poised to revolutionize the very notion of a dining experience. It s not just recipes. it s a symphony of flavors and textures orchestrated to tantalize taste buds and elevate the art of dining to an exquisite and unforgettable level. Join Diana on a journey of Recipe Ideas.",
    "maker": "Diana"
  },
  {
    "id": 5,
    "img": "",
    "url": "",
    "title": "The Quest for Coffee",
    "description": "In bold, urgent scrawl, the note screams,  I need coffee NOW.  It s not a mere request. it s a battle cry, a desperate plea to awaken the senses, fuel productivity, and avert the looming specter of drowsiness. Join Eve on a quest to stave off sleep and kick-start the day with Buy Coffee22.",
    "maker": "Eve"
  },
  {
    "id": 6,
    "img": "/Public/1.jpg",
    "url": "",
    "title": "Another Round of Meeting Notes",
    "description": "Dive into the latest chronicle of the team s journey through the labyrinth of business endeavors, encapsulating the essence of each discussion, decision, and revelation unveiled during the meeting. It s more than just words on paper. it s the heartbeat of innovation and collaboration that pulses through every documented line, nurturing the seeds of progress and growth. Join Alice in crafting the future.",
    "maker": "Alice"
  },
  {
    "id": 8,
    "title": "Dubai Adventure",
    "description": "Embark on a virtual tour of the mesmerizing city of Dubai. Explore the thrilling worlds of adventure that await, from iconic landmarks to breathtaking landscapes. Immerse yourself in the rich culture and vibrant energy of this global hub. Join us on a visual journey through Dubai s wonders!",
    "img": "https://mywowo.net/media/images/cache/dubai_img_worlds_of_adventure_01_presentazione_jpg_1200_630_cover_85.jpg",
    "url": "https://www.visitdubai.com/",
    "maker": "Dubai Tourism"
  }
]
```


Рисунок 3.2 – Представлення бази даних у вигляді JSON-файлів

3.3 Використання Json серверу

JSON Server – це інструмент для швидкого створення та запуску фейкового REST API з мінімальними зусиллями. Це надає можливість розробникам працювати з фейковим сервером, який веде себе як реальний API, але не використовує реальні дані бази даних. JSON Server використовує JSON файли для зберігання даних та автоматично генерує API, що надає CRUD (створення, читання, оновлення, видалення) операції, як наведено у таблиці 3.1 характеристики JSON Server.

Дані також зберігаються в JSON форматі, що робить їх легкими для редагування та розуміння. Кожен JSON файл може представляти ресурс (наприклад, колекцію об'єктів) у вашому API.

Використання JSON Server може значно полегшити процес розробки та тестування, особливо в ситуаціях, коли реальна база даних ще не готова або доступна. Він дозволяє розробникам швидко створювати та тестувати API, спрощуючи їхню роботу і зменшуючи час, необхідний для початкового

налаштування середовища.

Розробники можуть легко додавати, оновлювати та видаляти дані з їхнього фейкового API без необхідності налаштовувати складні запити тощо.

Таблиця 3.1 – Характеристика JSON Server

Основні риси та функціональність	JSON Server
Швидке налаштуваня	Запуск фейкового сервера на JSON Server можливий за декілька хвилин. Вам потрібно лише вказати JSON файл з даними, і сервер готовий до використання.
RESTFUL API	JSON Server генерує RESTful API, що дозволяє виконувати стандартні HTTP-запити, такі як GET, POST, PUT, PATCH та DELETE. Це робить його ідеальним інструментом для тестування та розробки вебдодатків, що використовують HTTP-запити
Маршрутизація	Можливість налаштувати маршрути, щоб визначити, які дані ви хочете використовувати та як обробляти різні типи запитів
Фільтрація та сортування	JSON Server підтримує фільтрацію та сортування даних за різними критеріями, що дозволяє вам тестувати різні сценарії запитів

Нижче представлена блок схема роботи json файлу.Створення блок-схеми роботи з JSON файлом допомагає візуалізувати процес обробки даних, полегшує аналіз та вдосконалення програми, сприяє комунікації в команді, використовується для навчання та документації, а також допомагає виявляти та виправляти помилки, як на рисунку 3.3.

JSON Server часто використовується в тестуванні та розробці вебдодатків, коли реальний сервер чи база даних ще не готові, а розробникам потрібно швидко створити фейковий API для взаємодії з фронтендом. Це також корисний

інструмент для навчання та експериментів з роботою з RESTful API.

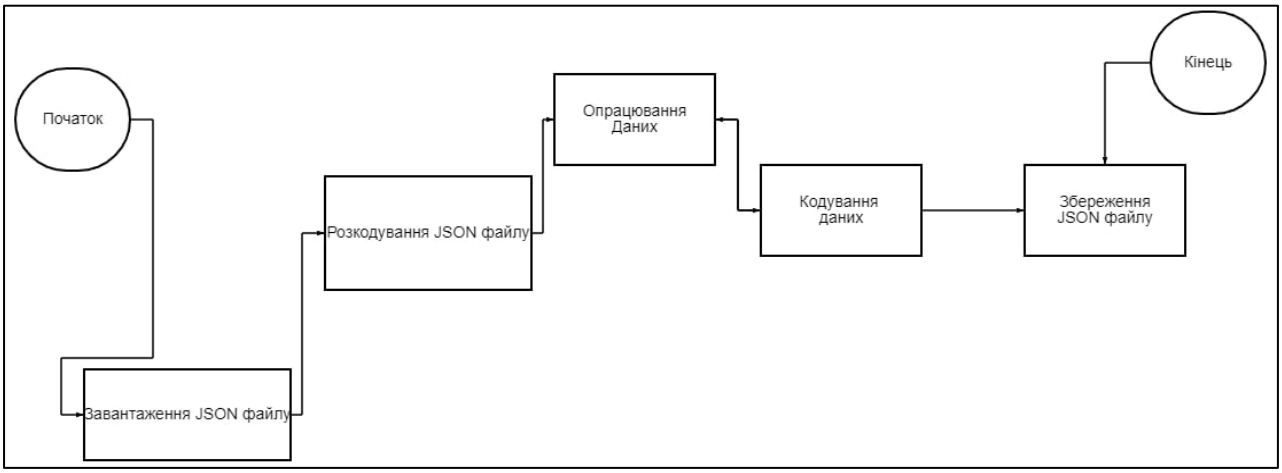


Рисунок 3.3 – Блок схема роботи json файлу

RESTful API (Representational State Transfer API) – це архітектурний стиль для побудови вебсервісів, які використовують стандартні HTTP-методи наприклад, GET, POST, PUT, DELETE) для взаємодії між клієнтом та сервером. Термін "RESTful" вказує на те, що API відповідає принципам REST.

3.4 Виконання програми

Для того, щоб розпочати роботу необхідно відкрити проєкт у програмному середовищі «Visal Studio Code». Запустити консоль консоль Terminal-New terminal (Термінал – Новий термінал).

Завантажити сайт за допомогою кнопки «RUN» та сервер за допомогою команди «npx json-server». Після заватаження сайту перейти за адресою http://localhost:8000/blogs, як на рисунку 3.4.

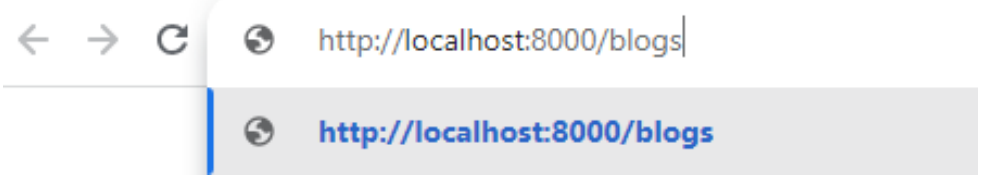


Рисунок 3.4 – Вхід до вебдодатку з браузера

На рисунку 3.5. представлений код створення віртуального хосту.

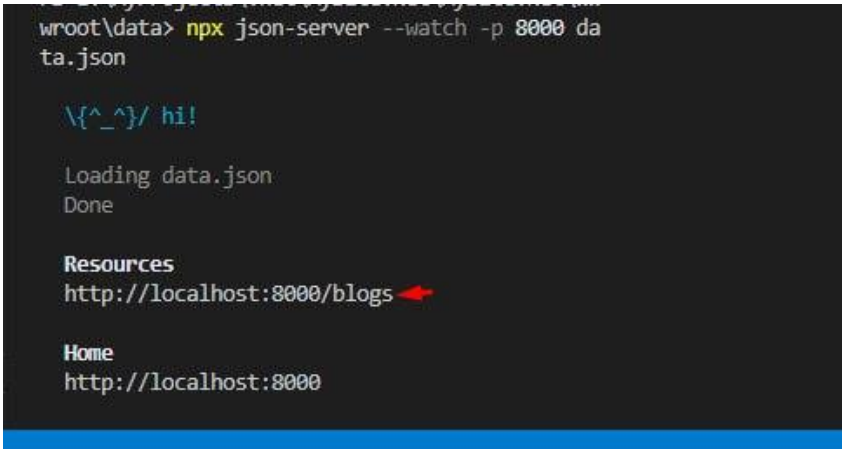


Рисунок 3.5 – Створення локального хосту

Після переходу за посиланням на вебсайт, користувач знаходить себе на головній сторінці блогу, де користувач може побачити як буде виглядати його майбутній блог як наведено у рисунку 3.6.

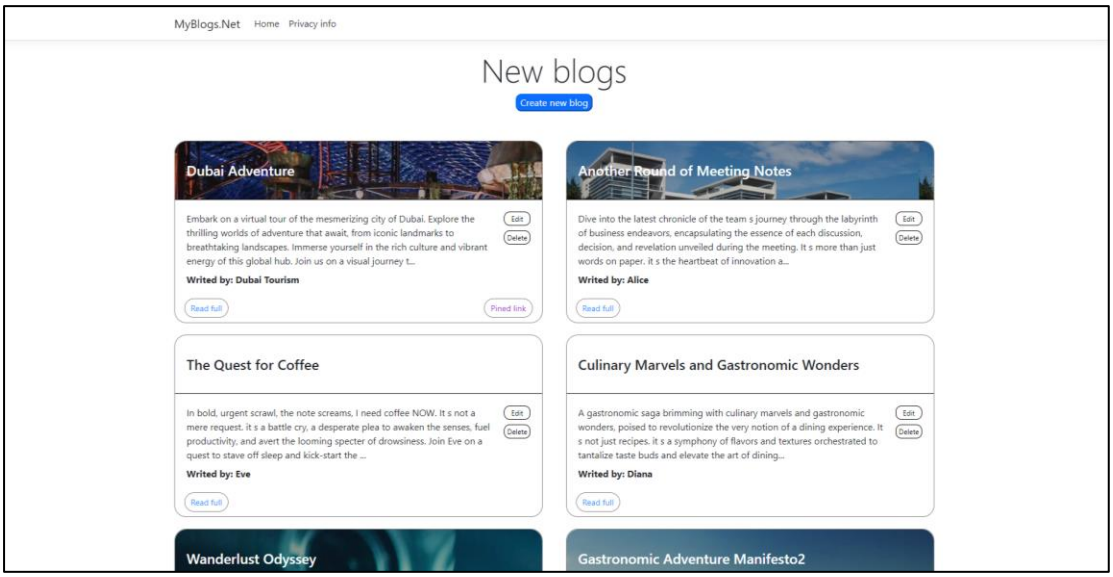


Рисунок 3.6 – Зображення головної сторінки універсального блогу

Для того щоб створити власний блог треба натиснути кнопку «Create new blog», яка знаходиться на головній сторінці та підсвічена синім кольором. Після натискання користувачем цієї кнопки, користувач бачить вікно реєстрації, яке зображено на рисунку 3.7.

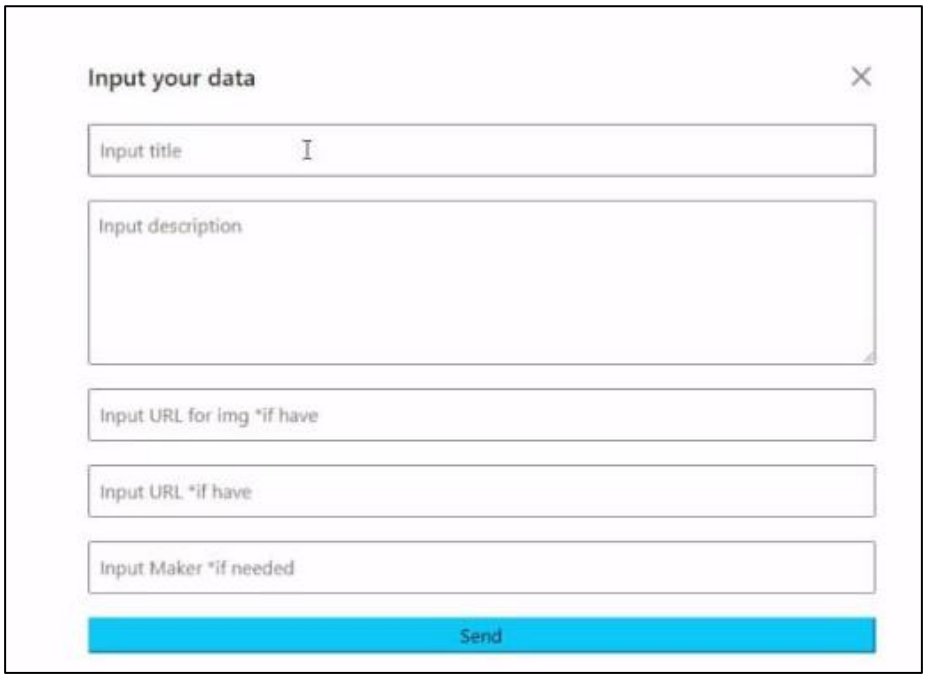


Рисунок 3.7 – Вікно створення нового блогу

Для того, щоб створити власний блог треба заповнити відповідні поля, такі як тема блогу, його короткий опис, посилання на картинку, якщо його маєте, та засіб вводу, якщо він потрібен. Після заповнення вікна створення, треба натиснути кнопку «SEND», як зображено на рисунку 3.8 та 3.9.

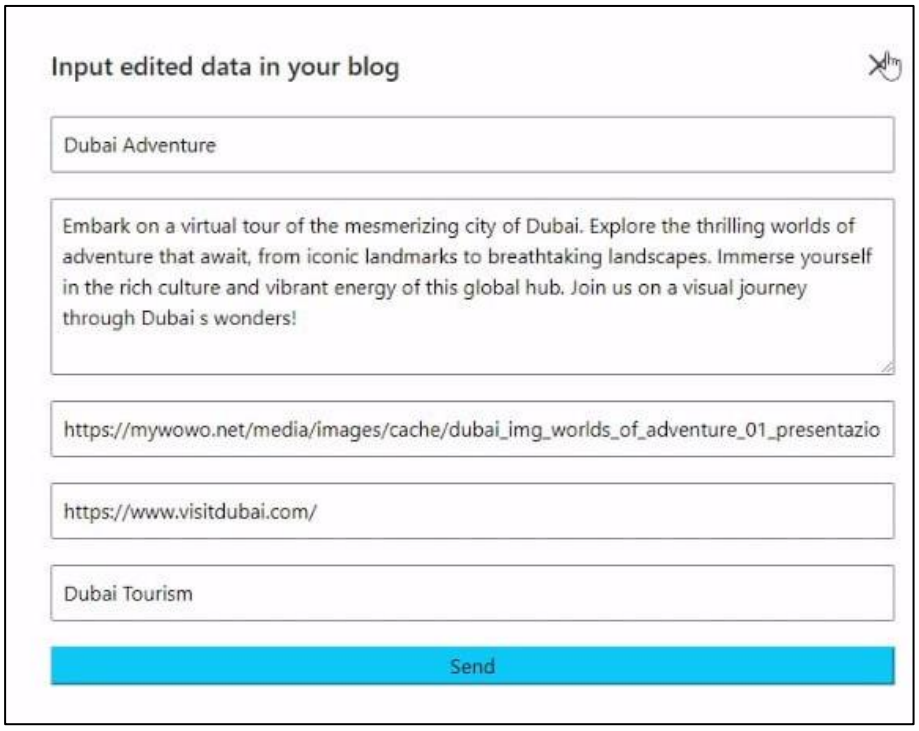


Рисунок 3.8 – Заповнене вікно «Створення блогу»

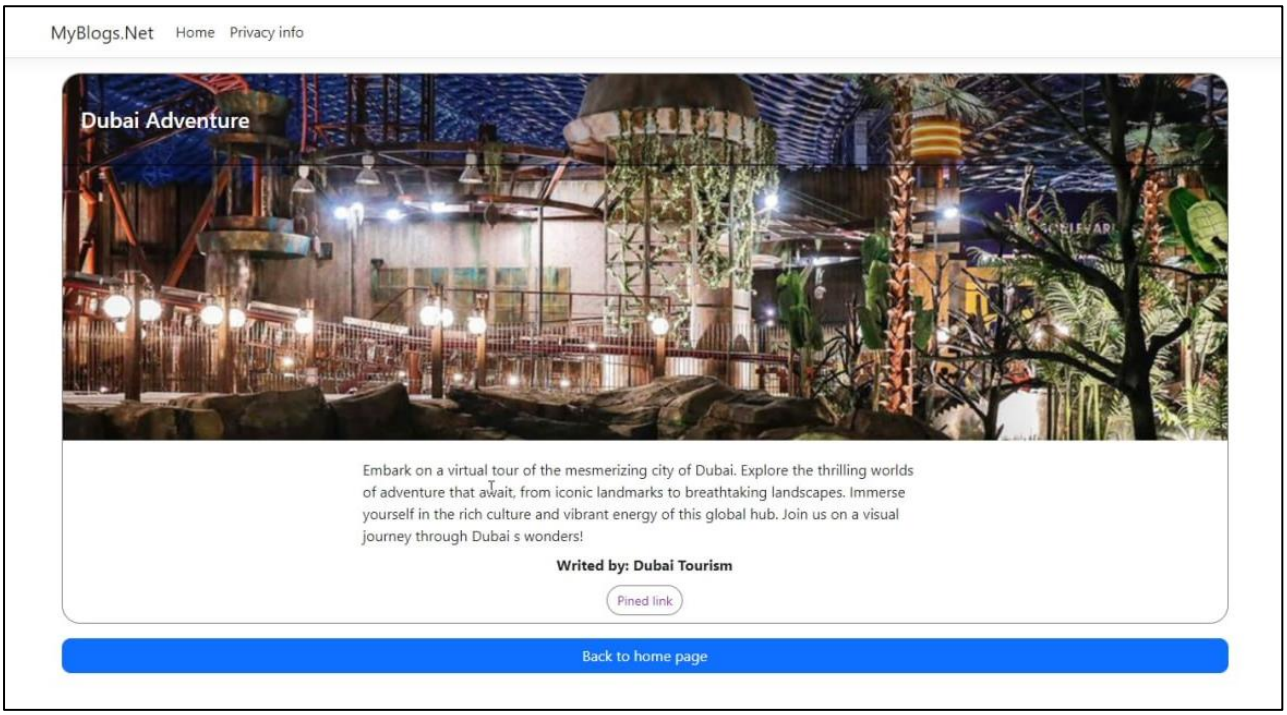


Рисунок 3.9 – Зображення блогу

Після створення власного блогу, користувач також може його видалити за допомогою кнопки «DELETE», яка зображена червоного кольору. Видалення блогу зображено на рисунку 3.10.

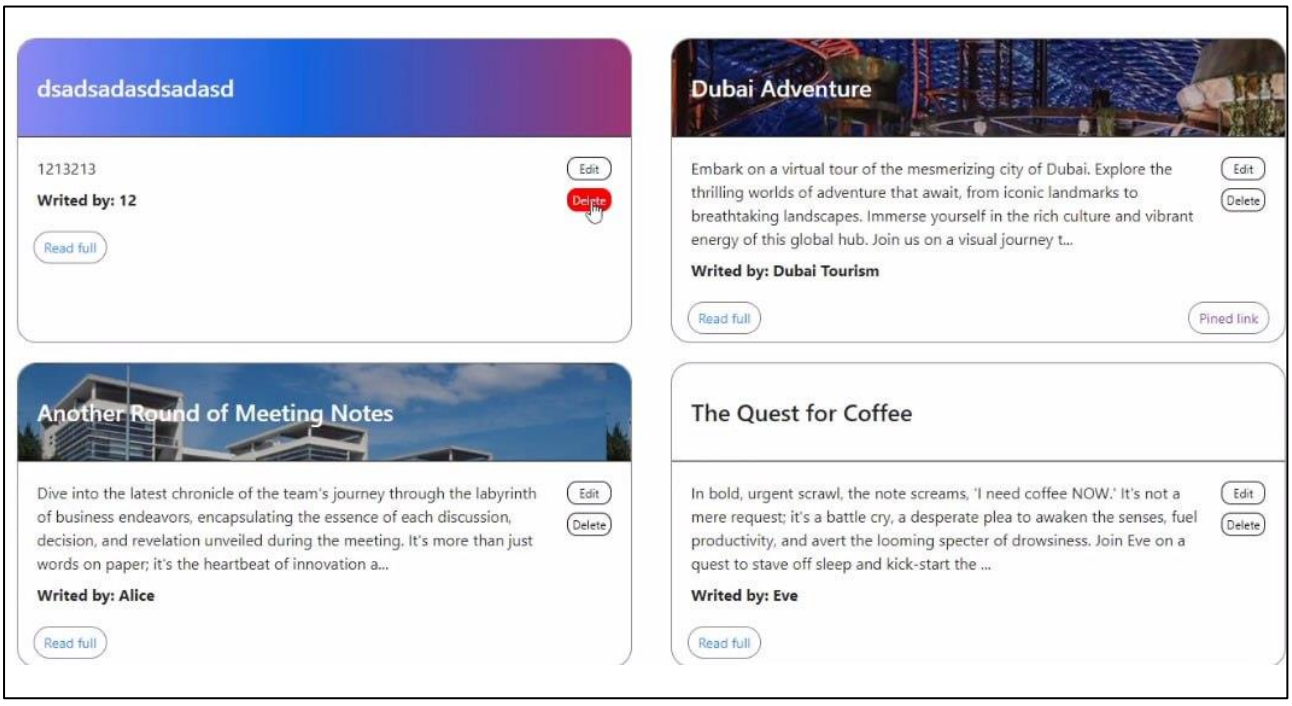


Рисунок 3.10 – Видалення блогу

3.5 Висновок до розділу 3

Вебдодаток розгортався за допомогою Visual Studio Code, використовуючи JSON Server для імітації роботи RESTful API. Після старту додатку користувач може переглядати, створювати та видаляти блоги через зручний інтерфейс.

Цей підхід дозволяє швидко розробляти та тестувати вебдодатки, особливо корисний на етапі розробки, коли реальний сервер чи база даних ще не готові

Щоб зробити процес ще більш повноцінним, рекомендується використовувати npm для управління пакетами, розділяти frontend і backend, використовувати модульні фреймворки, перевіряти RESTful API, додавати логування, обробку помилок, і використовувати тести для забезпечення стабільності та надійності коду. Документація та можливість легкої співпраці з іншими розробниками також допомагають у поліпшенні якості проєкту.

Використання npm, розділення frontend і backend, використання модульних фреймворків і уважне тестування створюють основу для стабільного та гнучкого додатка.

Цей підхід особливо корисний на етапі розробки, коли необхідно швидко випробувати концепції та функціонал. Досягнутий рівень модульності та тестування полегшує майбутні розширення та інтеграцію з реальними серверами та базами даних.

Завдяки цим практикам розробка стає більш ефективною, а вихідний код стає готовим для подальших етапів розвитку та впровадження в продакшн.

Розділення frontend і backend надає гнучкість для майбутнього розширення та інтеграції з реальними серверами та базами даних.

ВИСНОВКИ

Під час розробки вебдодатка на основі .NET та використання різних технологій, вдало врахував ключові аспекти, які дозволяють створити ефективний та зручний продукт. Підхід до дизайну, бази даних та використання JSON Server являє собою ретельно продуманий план, спрямований на досягнення кращого користувацького досвіду та оптимальної продуктивності.

Підхід до дизайну, зосереджений на мінімалізмі, чистоті та зручності використання, створює приємний та ефективний користувацький досвід. Кольорова палітра, логічне розташування елементів та контрастні кольори роблять інтерфейс інтуїтивно зрозумілим та привабливим.

Використання JSON-файлу для зберігання даних є ефективним та зручним рішенням, особливо для невеликих та середніх проєктів. Це дозволяє швидко розгортати додаток та спрощує управління та конфігурацію бази даних.

Впровадження JSON Server дозволяє швидко створювати та тестувати фейкове REST API, полегшуючи взаємодію з фронтендом та дозволяючи перевіряти різні сценарії запитів.

Загальний висновок полягає в тому, що підхід до розробки відзначається пристосуванням до потреб користувачів, ефективністю використання технологій та легкістю управління проєктом. Результатом є стабільний, зручний та ефективний вебдодаток, готовий до використання в реальних умовах.

Вебдодаток, спрямований на створення та перегляд блогів, відкриває перед користувачами цілий ряд можливостей та переваг.

Блог дозволяє користувачам виражати свої думки, ідеї, творчість та досвід. Це сприяє створенню віртуальних спільнот та обміну інформацією між різними авторами.

Дизайн, спрямований на зручність користування, робить процес створення блогів легким та доступним, що може заохочити більше людей висловлювати свої думки та ідеї в онлайн-середовищі.

Можливість створювати та взаємодіяти з вмістом блогів через відгуки, коментарі та реакції дозволяє встановлювати взаємодію між авторами та читачами, роблячи платформу більш соціальною.

Блог стає ефективним засобом для збору та поширення інформації. Користувачі можуть отримувати актуальні новини, та ділитися досвідом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Microsoft .NET Documentation 2023. URL: https://learn.microsoft.com/en-us/dotnet/ (дата звернення: 06.07.2023).
2. Entity Framework Core Documentation. 2023. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 08.07.2023).
3. ASP.NET Core Documentation 2023. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0 (дата звернення: 14.07.2023).
4. C# Programming Guide 2023. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 20.07.2023).
5. Introduction to ASP.NET Identity. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio (дата звернення: 22.07.2023).
6. Як вчити .NET: докладна інструкція для новачків і кілька порад для досвідчених 2023. URL: https://dou.ua/lenta/articles/net-for-beginners/ (дата звернення 21.07.2023).
7. Entity Framework Core in Action, Second Edition. 2021. URL: https://www.manning.com/books/entity-framework-core-in-action-second-edition (дата звернення: 28.09.2023).
8. *C# 10 і .NET* Сучасна крос-платформна розробка 2022. URL: https://habr.com/ru/companies/piter/articles/714396/ (дата звернення: 01.10.2023).
9. Троелсен Э. Язык программирования C# и платформа .NET 4.5. Київ : Вильямс, 2014. 1300 с.
10. Visual Studio. URL: https://visualstudio.microsoft.com/ru/ (дата звернення: 05.10.2023).
11. .NET Framework Guide. URL: https://docs.microsoft.com/en-us/dotnet/framework/ (дата звернення: 02.10.2023).

ДОДАТОК А

Код запуску програми

```
using System.Collections.Generic;
using System.Net.Http;
using System.Reflection.Metadata;
using System.Threading.Tasks;
using global::MySite.Net.Models;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;

namespace MySite.Net.Services
{
   public class JsonFileBlogServices
   {
      private readonly IWebHostEnvironment _webHostEnvironment;
      private readonly IConfiguration _configuration;
      private readonly HttpClient _httpClient;

      public JsonFileBlogServices(IWebHostEnvironment webHostEnvironment,
IConfiguration configuration, HttpClient httpClient)
      {
         _webHostEnvironment = webHostEnvironment;
         _configuration = configuration;
         _httpClient = httpClient;
      }

      private string ApiUrl => _configuration["ApiUrl"]; // Make sure to set this in
your appsettings.json
```

```
    public async Task<IEnumerable<Blog>> GetBlogs()
    {
        try
        {
            HttpResponseMessage response = await _httpClient.GetAsync(ApiUrl);

            if (response.IsSuccessStatusCode)
            {
                return await
response.Content.ReadFromJsonAsync<IEnumerable<Blog>>();
            }
            else
            {
                Console.WriteLine($"Error: {response.StatusCode} -
{response.ReasonPhrase}");
                return null;
            }
        }
        catch (HttpRequestException ex)
        {
            Console.WriteLine($"Request error: {ex.Message}");
            return null;
        }
    }

    public async Task<Blog> GetBlog(int id)
    {
        try
        {
            HttpResponseMessage response = await
```

```
_httpClient.GetAsync($"{ApiUrl}/{id}");

            if (response.IsSuccessStatusCode)
            {
                return await response.Content.ReadFromJsonAsync<Blog>();
            }
            else
            {
                Console.WriteLine($"Error: {response.StatusCode} -
{response.ReasonPhrase}");
                return null;
            }
        }
        catch (HttpRequestException ex)
        {
            Console.WriteLine($"Request error: {ex.Message}");
            return null;
        }
    }
  }
}
```

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using MySite.Net.Models;
using MySite.Net.Services;
using Microsoft.Extensions.Logging;

namespace MySite.Net.Pages
```

```
{
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;
        public JsonFileBlogServices BlogService;
        public IEnumerable<Blog> Blogs { get; private set; }

        public IndexModel(ILogger<IndexModel> logger,
            JsonFileBlogServices blogService)
        {
            _logger = logger;
            BlogService = blogService;
        }

        public async Task OnGet()
        {
            Blogs = await BlogService.GetBlogs();
        }
    }
}
```

ДОДАТОК Б

Код активації кнопок та функції

```
const databaseURL = "http://localhost:8000/blogs";

let selectedId = null;

let title = $("#titleInput").val();
let description = $("#descriptionInput").val();
let img = $("#imageInput").val();
let url = $("#urlInput").val();
let maker = $("#makerInput").val();

const ShowForm = (id = "", title = "", description = "", img = "", url = "", writer = "",
isEditing =false) => {
   $('#inputForm').addClass('form-on');

   selectedId = +id;

   if (isEditing) {
      $('#textOnForm').text("Input edited data in your blog");
      $('#titleInput').val(title);
   } else {
      $('#titleInput').val(title.name);
   }
   $('#descriptionInput').val(description);
   $('#imageInput').val(img);
   $('#urlInput').val(url);
   $('#makerInput').val(writer);
```

```
}

const CloseForm = () => {
    $('#inputForm').removeClass('form-on');
}

const getCurrentInput = () => {
     title = $("#titleInput").val();
     description = $("#descriptionInput").val();
     img = $("#imageInput").val();
     url = $("#urlInput").val();
     maker = $("#makerInput").val();
}

const checkCreateOrUpdate = () => {
    getCurrentInput();
    if (selectedId !== selectedId || selectedId === null) {
        console.log("postNewBlog");
        postNewBlog(title, description, img, url, maker);
    } else {
        console.log("updatePostedBlog");
        updatePostedBlog(title, description, img, url, maker);
    }
}

const postNewBlog = async (title, description, img, url, maker) => {

    const response = await fetch(databaseURL);
    if (!response.ok) {
        throw new Error("Could not fetch data from that resource");
```

```
    }
    const existingBlogs = await response.json();
    const maxId = Math.max(...existingBlogs.map(blog => blog.id));
    let id = maxId + 1;
    const newBlog = { id, title, description, img, url, maker };
    console.log(newBlog);
    fetch(databaseURL, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(newBlog),
    }).then((res) => {
        if (!res.ok) {
            throw Error("could not fetch data from that resource");
        }
        console.log("Blog added");
        location.reload()
    });
};

const updatePostedBlog = (title, description, img, url, maker) => {
    const blog = {selectedId, title, description, img, url, maker};

    console.log(blog);

    fetch(databaseURL + /${selectedId}, {
        method: "PATCH",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(blog),
    }).then((res) => {
        if (!res.ok) {
```

```
            throw Error("could not fetch data from that resource");
        }
        console.log("Blog updated");
        location.reload()
    });
};

const deleteBlog = (idToDelete) => {
    let confirmMenu = window.confirm("Are you sure you want to delete this blog?");

    if (confirmMenu) {
        fetch(databaseURL + /${idToDelete}, {
            method: "DELETE",
        }).then((res) => {
            if (!res.ok) {
                throw Error("could not redact data");
            }
            console.log("Blog deleted");
            location.reload()
        });
    }
}


const trimText = (text) => {
    let maxLength = 300;

    if (text.length > maxLength) {
        // Trim the text and add "..."
        console.log('text trimed');
```

```
        let trimedText = text.substring(0, maxLength - 3) + '...';
        return trimedText;
    }
    return text;

}


$('#showEmptyForm').on("click", ShowForm);
$('#cancelButton').on("click", CloseForm);
$('#pushButton').on("click", checkCreateOrUpdate);

window.onload = () => {
    $('.description-text').each(function () {
        $(this).text(trimText($(this).text()));
    });
    //createNewBlog(title = "adsasd", description = "adasdasdasd", img = "", url = "",
writer = "DDDD");
}
```

ДОДАТОК В

Program.cs код сервісів

```
using MySite.Net.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddHttpClient<JsonFileBlogServices>();
builder.Services.AddTransient<JsonFileBlogServices>();

var app = builder.Build();

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

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

app.UseRouting();

app.UseAuthorization();
app.MapRazorPages();


app.Run();
```