МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

ДИПЛОМНА РОБОТА

на тему:

“СТВОРЕННЯ САЙТУ ДЛЯ ВЕДЕННЯ БЛОГІВ”

Виконав: студент групи ПМІ-45
спеціальності
122 «Комп’ютерні науки»
(шифр і назва спеціальності)

Бурда Б. Т.
(підпис) (прізвище та ініціали)

Керівники проф. Притула М. М.

конс.ас. Прядко О. Я.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Львів – 2023

РЕФЕРАТ

Робота складається із вступу, чотирьох розділів, висновків, джерел та додатків. Обсяг дипломної роботи: 32 сторінки, 1 таблиця та 11 рисунків. Список використаних джерел складається з 5 найменувань.

Дипломна робота на тему "Створення сайту для ведення блогів" присвячена розробці повноцінного рішення для ведення блогів з використанням технологій Angular та .NET Web API. Метою цієї роботи є створення функціонального та привабливого вебсайту, який надає користувачам зручні та ефективні інструменти для ведення свого власного блогу. Для досягнення цієї мети було проведено аналіз потреб та вимог користувачів. Зрозумівши, що користувачі очікують від такого вебсайту, було розроблено дизайн та інтерфейс. Застосування Angular, дозволило реалізувати функціональність вебсайту з точки зору користувача.

Одним із ключових аспектів розробки було забезпечення безпеки та захисту даних користувачів. З цією метою були реалізовані можливості реєстрації та авторизації користувачів з використанням захищених протоколів та шифрування. Це дозволяє забезпечити доступ до блогів тільки авторизованим користувачам та гарантувати конфіденційність їх особистої інформації. Окрім цього, розробка функціоналу для створення, редагування та видалення блогів була важливим етапом роботи. Користувачі можуть створювати свої власні блоги, додавати нові записи, редагувати та видаляти існуючі записи, надавати коментарі та обмінюватися думками з іншим. Це створює сприятливу та інтерактивну спільноту для обміну ідеями та думками.

Результатом дипломної роботи є функціональний вебсайт для ведення блогів. Він забезпечує користувачам зручність, надійність та ефективність у веденні їхнього власного блогу. Завдяки інтуїтивному інтерфейсу та доступним функціям, користувачі можуть легко створювати та редагувати свої блоги, спілкуватися з іншими користувачами та зберігати свої дані у безпечному середовищі.

ЗМІСТ

ВСТУП

У наш час інтернет займає дуже велику частину нашого життя. Багато людей у ньому шукають місце, де вони зможуть поділитись своїми думками, ідеями, або можливість допомогти іншим стати кращими версіями себе, публікуючи різні навчальні та мотиваційні матеріали. У той самий час є й люди, які шукають, де можуть знайти цю інформацію та, беручи приклад з людей, які досягли тих чи інших вершин, досягти ще більшого, завдяки їхньому досвіду. Сайт для вебблогів – це місце, яке може об’єднати ці дві категорії людей.

У зв’язку з цим, метою моєї дипломної роботи є створення повноцінного рішення для ведення блогів. Для Frontend-частини сайту буде використана технологія Angular, яка є однією з найпопулярніших технологій для розробки вебдодатків. Для розробки Backend частини - технологія .NET, що надає широкі можливості для створення багатофункціонального API.

Завданням дипломної роботи є:

- Аналіз потреб і вимог користувачів для вебблогу.
- Розробка дизайну і інтерфейсу вебсайту для зручного ведення блогів.
- Використання технології Angular для розробки фронтенду.
- Реалізація можливостей реєстрації і авторизації користувачів.
- Розробка функціоналу для створення, редагування і видалення блогів.
- Розробка системи взаємодії між користувачами, включаючи коментарі та вподобання дописів.
- Використання технології .NET для розробки бекенду та API.
- Забезпечення безперебійної роботи системи шляхом обробки помилок та винятків.

- Взаємодія з базою даних для збереження і відображення даних користувачів та їх блогів.
- Тестування розробленого рішення для перевірки його функціональності та безпеки.

Після успішної реалізації проекту і проведення всіх необхідних тестів, цей вебсайт може стати цінним ресурсом для всіх, хто зацікавлений у розвитку та спілкуванні зі спільнотою однодумців. Він надасть можливість поділитись знаннями та досвідом, що сприятиме взаємному зростанню та розвитку.

Ця дипломна робота є важливим кроком у напрямку розробки сучасних вебдодатків, адже використовує найактуальніші технології для цього, та надає рішення проблемам сучасного світу. Результати роботи можуть бути корисні для розробників вебсайтів, які працюють у галузі веброзробки та хочуть збільшити свої знання та навички.

1. ТЕОРЕТИЧНІ ЗАСАДИ РЕАЛІЗАЦІЇ ВЕБСАЙТУ

1.1 Дослідження аналогів, які надають можливості ведення блогів.

Існує безліч аналогів, на яких кожен може створити власний блог на ту чи іншу тематику. Розглянемо найпопулярніші з них, та порівняємо їх з рішенням, яке буде реалізовано у результаті цієї дипломної роботи.

1.1.1 Вебсайти для блогів, як окремих вебсторінок

Це вебсайти, які надають можливість створити окрему сторінку, на якій користувачі можуть вести свій блог та ділитись ним з іншими. Основною відмінністю їх від рішення, яке пропонується для реалізації дипломної роботи, є те, що для того, щоб інші люди мали можливість слідкувати за цим блогом, його потрібно поширювати за допомогою сторонніх сервісів або ж соціальних мереж. Це рішення підходить для людей, які хочуть розширити вже існуючий блог на інших платформах, створивши окремий вебсайт та поділившись посиланням на нього. Прикладами таких платформ є Blogger.com, WordPress та багато інших.

1.1.2 Вебсайти для блогів та слідкування за ними

Окрім вебсайтів, які надають можливість створити окрему сторінку для блогу, існують також платформи, що спеціалізуються на створенні та розповсюдженні вмісту. Такі платформи, як, наприклад, Medium, дозволяють користувачам створювати свої блоги та публікувати на них статті, а також слідкувати за іншими авторами та їх вмістом. Це рішення підходить для тих, хто хоче мати свій власний блог, але не хоче витрачати час та зусилля на створення та просування вебсайту. Medium та інші подібні платформи надають велику аудиторію, що дозволяє збільшити кількість читачів та отримати більшу кількість

відгуків на свої публікації. Однак в основному на таких платформах є платні підписки, або ж перенасиченість зайвого функціоналу [1]. Метою ж даної дипломної роботи є створення повністю безкоштовного простору з найважливішими функціями, з якими зможе розібратись кожен користувач за кілька хвилин, проведених на вебсайті.

Отже, дослідження аналогів показало, що існує безліч платформ, на яких можна створити свій блог. Вони мають як свої переваги, так і недоліки.

Переваги вебсайтів для блогів як окремих вебсторінок:

- Можливість налаштування дизайну та велика кількість функціоналу

Недоліки вебсайтів для блогу як окремих вебсторінок:

- Не мають можливості слідкування за блогом без поширення через сторонні сервіси або соціальні мережі
- Не надають функціоналу для фільтрації та пошуку інформації

Переваги вебсайтів для блогів та слідкування за ними:

- Можливість слідкувати за багатьма блогами на одному сайті;
- Зручний інтерфейс для читача
- Широкий вибір тематик

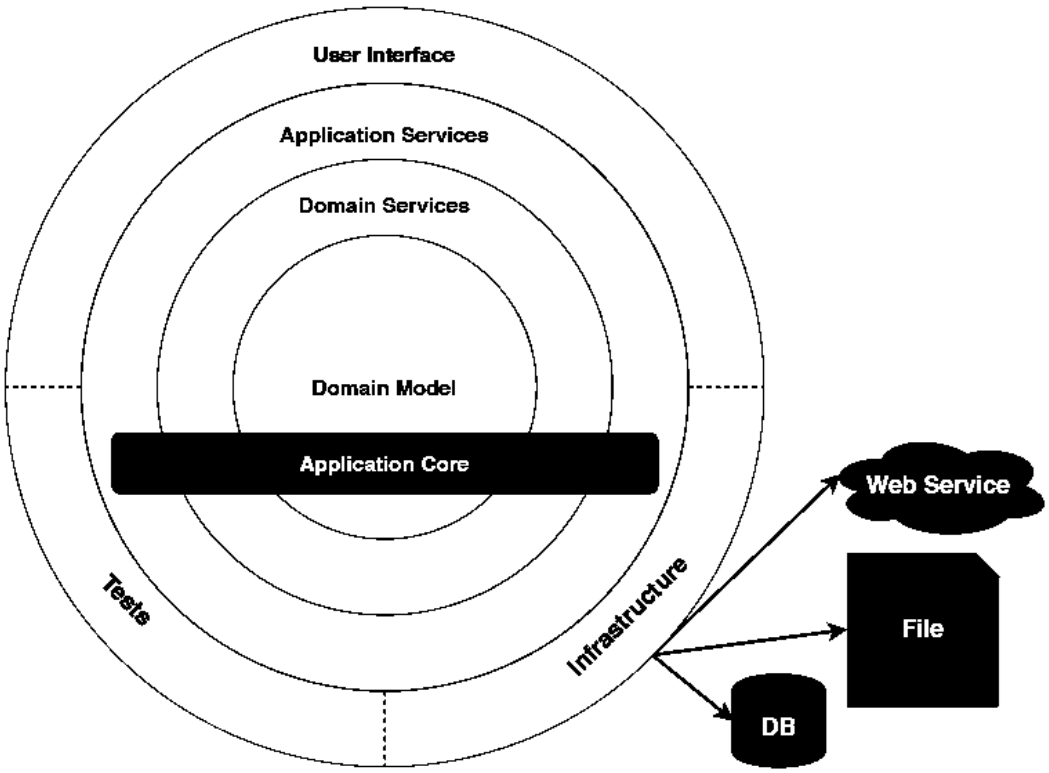
Недоліки вебсайтів для блогу та слідкування за ними:

- Обмежена можливість налаштування дизайну
- Платні підписки
- Перенасиченість зайвим функціоналом

Ці платформи можуть бути як окремими вебсторінками, так і вебсайтами для блогів та слідкування за ними. Окрім цього, платформи для блогів можуть мати різні функціональні можливості та платні підписки. Виходячи з цієї інформації, основною метою роботи є створення безкоштовного вебсайту з найважливішими

функціями, що буде доступний для кожного користувача та дозволить легко створювати та публікувати свій вміст, слідкувати за діяльністю інших учасників з можливістю фільтрації інформації за тим, що цікаво користувачам.

1.2 Clean Architecture



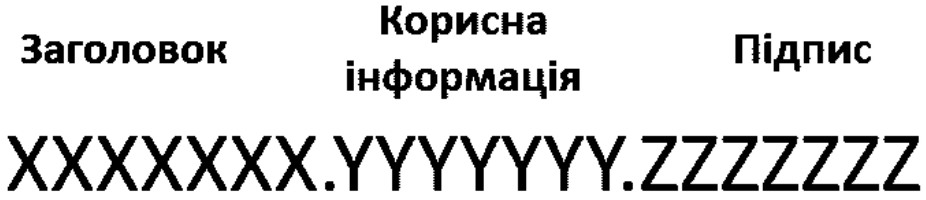
Малюнок 1 Діаграма Clean Architecture

Вебсайт повинен притримуватись принципів Clean Architecture. Цей принцип поєднує в собі ідеї декількох інших архітектурних підходів, що акцентують увагу на тестуванні, незалежності від UI та БД, зовнішніх фреймворків та бібліотек. Це досягається через розділення на різні шари та дотримання правила залежностей, яке стверджує, що внутрішні шари не повинні залежати від зовнішніх (Малюнок 1). Це правило допомагає забезпечити легку підтримку системи, тому що будь-які зміни в зовнішніх шарах не впливатимуть на внутрішні шари [2] .

1.3 JWT Token Authentication

.NET Web API, який використовується для реалізації цієї дипломної роботи, має глибоке захищення даних користувачів, яке забезпечується використанням JWT токену. JWT (JSON Web Token) - це стандарт, який використовується для передачі інформації між сторонами у форматі JSON. В основі JWT лежить три частини: заголовок, корисна інформація та підпис (Малюнок 2).

- Заголовок містить тип токена та алгоритм шифрування, який використовується для підпису.
- Корисна інформація містить дані про користувача або будь-яку іншу інформацію, яка повинна бути передана.
- Підпис гарантує цілісність даних та перевіряє, що дані не були змінені після створення токена. JWT Token може бути використаний для автентифікації користувача на сервері.



Малюнок 2 Шаблон JWT токена

При кожному запиті до сервера, клієнт повинен передавати JWT Token у заголовку запиту. Сервер перевіряє підпис токена та використовує дані з корисної інформації для авторизації запиту. Це дозволяє переконатись у тому, чи має даний користувач доступ до того чи іншого ресурсу.

Також, існує таке поняття, як refresh token. Основна ідея полягає в тому, що після успішної аутентифікації користувача, йому виділяються два типи токенів: доступу (access token) і оновлення (refresh token). Access token має обмежений

термін життя і використовується для авторизації запитів користувача до захищених ресурсів. Refresh token має довший термін життя і використовується для отримання нового access token після того, як попередній access token закінчився.

В процесі використання refresh token, коли access token стає недійсним (наприклад, через закінчення його терміну дії або вимушене видалення), користувач може використовувати refresh token для запиту нового access token без необхідності повторно вводити свої облікові дані. Це полегшує процес аутентифікації для користувача і дозволяє тривалий час тримати його у системі.

2. ТЕХНІЧНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ ВЕБСАЙТУ

Обрати правильний стек технологій при розробці нового проекту – це один з найважливіших етапів проектування програмного забезпечення. Обираючи те, що буде використано у цій роботі, була звернута увага на тенденції у сучасному світі програмування. Проект, який було реалізовано, складається з двох частин – клієнтської та серверної.

Було використано наступні технології для реалізації функціоналу проекту:

- FrontEnd частина
    - Angular
    - TypeScript
    - HTML & CSS
    - Angular Material UI
- BackEnd частина
    - ASP.NET 5 WEB API
    - MS SQL
    - Entity Framework

Також були використані наступні програмні засоби для реалізації проекту та тестування готового функціоналу:

- Visual Studio 2022
- Visual Studio Code
- Swagger UI
- Postman
- MS SQL Server Management Studio

2.1 Angular

Angular - це платформа для розробки вебдодатків, яка базується на мові програмування TypeScript. Angular надає компонентно-орієнтований фреймворк для створення масштабованих вебдодатків, а також колекцію інтегрованих бібліотек, які охоплюють широкий спектр функціональних можливостей, включаючи маршрутизацію, управління формами, комунікацію клієнта та сервера і ще багато іншого.

Angular також надає набір інструментів для розробки, збирання, тестування та оновлення коду. Ця платформа призначена для розробки як окремих проектів з одним розробником, так і корпоративних додатків на ряди великих проектних команд.

Основна ідея Angular полягає у створенні компонентного дерева з окремих компонентів. Кожен компонент має свою логіку та представлення (відображення на сторінці). Компоненти можуть взаємодіяти між собою та з іншими компонентами, що дозволяє створювати складні вебдодатки з багатьма функціональними можливостями [3].

2.2 TypeScript

TypeScript – мова програмування, що компілюється у мову JavaScript. Основними перевагами над JavaScript є:

- Статична типізація (можливість явного визначення типів)
- Можливість використання повноцінних класів
- Підтримка модулів

Ці переваги значно полегшують процес розробки та рефакторингу коду, тому що основні помилки випливають ще на етапі компіляції. Основний принцип мови — будь-який код на JavaScript сумісний з TypeScript.

2.3 HTML & CSS

HTML – мова розмітки, яка є основою будь-якого гіпертекстового документу у мережі інтернет. Теги розмітки визначають ті чи інші елементи сторінки, такі як заголовки, абзаци, таблиці, фото та інші. Завдяки тегам, браузер правильно інтерпретує сторінки.

CSS – мова для опису стилів HTML документа. Її використовують для задання кольору, шрифту, розміщення та розміру блоків. Основною метою CSS є розділення структури HTML сторінки та її зовнішнього вигляду.

2.4 Angular Material UI

Angular Material UI - це бібліотека для Angular, яка дозволяє створювати вебдодатки з використанням Material Design. Ця бібліотека містить широкий спектр компонентів, таких як кнопки, форми, таблиці та інші, які можна використовувати для створення вебдодатків з красивим і сучасним дизайном. Завдяки ним можна з легкістю дотримуватись одного дизайн – коду [4].

2.5 ASP.NET 5 WEB API

ASP.NET 5 WEB API - це фреймворк для розробки вебдодатків, що дозволяє створювати RESTful API для забезпечення обміну даними між клієнтом та сервером. Його можна використовувати для розробки вебдодатків будь-якої складності, від простих сайтів до великих корпоративних застосунків.

Фреймворк базується на мові програмування C# та дозволяє легко та швидко створювати вебсервіси, які можуть обробляти запити від клієнтів та повертати відповіді на них у форматі JSON або XML. Він також надає можливості, які допомагають розробнику встановити зв'язок з базою даних, налаштувати права доступу до ресурсів та обробляти помилки, які виникають під час роботи додатку.

2.6 MS SQL

MS SQL (Microsoft SQL Server) - це реляційна база даних, розроблена компанією Microsoft, яку використовують для зберігання та керування даними. MS SQL підтримує декілька мов запитів, зокрема SQL (Structured Query Language), T-SQL (Transact-SQL) та PL/SQL (Procedural Language/Structured Query Language). Це дає можливість здійснювати операції з даними, такі як створення таблиць, індексів, збережених процедур, функцій та тригерів. Окрім цього, MS SQL має безліч корисних функцій та інструментів для підтримки даних, такі як механізм резервного копіювання, реплікація даних та управління транзакціями. Вона також має можливість підключення до інших додатків та інтеграцію з різними інструментами розробки.

2.7 Entity Framework

Entity Framework – це фреймворк, що дозволяє працювати з базами даних як з об'єктами. Він дає можливість легко взаємодіяти з базами даних, без необхідності написання складних SQL запитів.

Entity Framework забезпечує мапінг об'єктів на таблиці бази даних та автоматично генерує SQL-запити, дозволяє зменшити кількість написаного коду, що підвищує його читабельність, та збільшує продуктивність. Також він підтримує багато різних типів баз даних, таких як Microsoft SQL Server, Oracle, MySQL та ін.

Цей фреймворк значно спрощує процес взаємодії з базою даних та зменшує кількість написаного коду. Також, використання Entity Framework дозволяє зосередитися на розробці функціональності додатку, замість того, щоб витрачати багато часу на написання SQL-запитів та обробку відповідей від бази даних.

2.8 Visual Studio 2022 & Visual Studio Code

Visual Studio - це інтегроване середовище розробки (IDE), яке дозволяє розробникам створювати різноманітні програми та додатки для різних платформ.

В IDE включено різноманітні інструменти для дебагінгу, підтримки версій, тестування, створення інтерфейсів користувача та велика кількість інших корисних функцій.

2.9 Swagger UI & Postman

Swagger UI та Postman - це два різних інструменти, що використовуються для розробки та тестування API. Postman дозволяє розробникам створювати колекції запитів, які можна виконувати із збереженням різних параметрів. Крім того, Postman надає можливість автоматизувати тестування та моніторинг API. У той же час, Swagger UI надає можливість швидко і просто протестувати endpoint прямо з вебінтерфейсу одразу після запуску API сервера.

2.10 MS SQL Server Management Studio

MS SQL Server Management Studio - це інтегроване середовище розробки, яке дозволяє розробникам та адміністраторам баз даних працювати з MS SQL Server. Воно надає засоби для створення, зміни та видалення об'єктів баз даних, таких як таблиці, процедури, функції, індекси та інші.

3. РЕАЛІЗАЦІЯ

Реалізацію вебсайту було розділено на дві окремі частини – клієнтську та серверну. Так, для серверної частини було створено проект ASP.NET Web API, у якому закладена реалізація авторизації/аутентифікації, роботи з базою даних та, відповідно, усього головного функціоналу, який забезпечений від стороннього доступу. Для клієнтської частини ж було обрано Angular, який отримує дані за допомогою запитів на сервер, обробляє їх та за допомогою компонентів виводить на екран користувача.

3.1 BackEnd

3.1.1 Startup

Реалізація backend частини починається з класу startup.cs. Він визначає конфігурацію та поведінку додатка при запуску [5].

У методі ConfigureServices (Додаток А) відбувається налаштування сервісів, які будуть використовуватись в додатку:

- AddControllers: додає можливість використовувати контролери для обробки HTTP-запитів.
- AddDbContext: налаштовує контекст бази даних з використанням підключення, отриманого з конфігурації.
- AddSingleton: додає Dependency injection налаштування для сервісу, який буде використовуватись для створення токенів оновлення.

Також у цьому методі використовується конфігурація JWT (JSON Web Token) для аутентифікації. Встановлюється схема аутентифікації JWT Bearer, налаштовуються параметри перевірки токена (ключ шифрування та інші).

У методі Configure відбувається налаштування конвеєра обробки HTTP-запитів (HTTP request pipeline). Він складається з різних компонентів Деякі з них включають:

- UseDeveloperExceptionPage: встановлює сторінку винятків для відображення винятків при розробці.
- UseSwagger та UseSwaggerUI: додають підтримку Swagger для створення документації API.
- UseHttpsRedirection: перенаправляє HTTP-запити на HTTPS.
- UseCors: налаштовує політику Cross-Origin Resource Sharing (CORS), що дозволяє обмінюватись ресурсами між різними доменами.
- UseAuthentication та UseAuthorization: аутентифікація та авторизація у додатку.
- UseEndpoints: налаштовує маршрутизацію для контролерів.

3.1.2 RefreshTokenGenerator

Цей клас використовується для генерації та збереження refresh токенів, які використовуються для продовження аутентифікації користувачів у системі (Додаток Б).

Метод GenerateToken генерує випадковий refresh токен для заданого користувача. Він створює випадкове число байтів, перетворює його у рядок Base64 і повертає цей рядок як refresh токен. Після цього метод перевіряє, чи існує в базі даних запис про користувача з заданим ідентифікатором (іменем користувача). Якщо такий запис існує, то він оновлює токен для цього користувача і зберігає зміни в базі даних. Якщо відсутній, то створюється новий запис з випадковим ідентифікатором та зберігається новий токен для цього користувача.

3.1.3 UserController

Це контролер, який використовується для обробки HTTP-запитів до шляху "/api/user". Він має функціонал для автентифікації користувачів за допомогою токенів та для реєстрації нових користувачів.

Конструктор класу приймає об'єкт Blog_DBContext для звернення до бази даних, об'єкт JWTSetting для отримання конфігурації JWT та об'єкт IRefreshTokenGenerator для генерації refresh токену.

Метод Authenticate (Додаток В) приймає ім'я користувача та масив з правами доступу. Він генерує JWT-токен з заданими правами та терміном дії. Він також генерує refresh токен за допомогою переданого імені користувача та повертає об'єкт TokenResponse, який містить обидва токени. Цей метод має атрибут [NonAction] та не використовується у запитах до API. Він був написаний для роботи з перевантаженим методом, який приймає Usercred об’єкт.

Метод Authenticate HTTP POST-запита на шляху /api/user/Authenticate. Цей метод приймає об'єкт Usercred, що містить ім'я користувача та пароль. Він перевіряє, чи існує користувач з таким іменем та паролем в базі даних. Якщо так, то він генерує JWT-токен та refresh токен за допомогою методу Authenticate. Якщо користувача не знайдено, повертається помилка 401 Unauthorized.

Метод Refresh приймає об'єкт TokenResponse, який містить JWT-токен та refresh токен. Він перевіряє, чи токен є дійсним та чи існує відповідний йому refresh токен у базі даних. Якщо токен не є дійсним або не знайдено відповідний refresh токен, повертається помилка 401 Unauthorized. Якщо все в порядку, метод Authenticate використовується для генерації нових токенів та повертає об'єкт TokenResponse.

Метод Register приймає об'єкт TblUser для реєстрації нового користувача. Він перевіряє, чи вже існує користувач з таким ідентифікатором Userid в базі даних. Якщо користувач вже існує, повертається порожній рядок. В іншому випадку, створюється новий об'єкт TblUser з переданими даними (id, ім'я, електронна пошта,

роль, пароль), додається до бази даних і зберігається. Після успішної реєстрації повертається об'єкт APIResponse зі статусом "pass".

Метод GetAllRole є HTTP GET-запитом на /api/user/GetAllRole. Він повертає список всіх ролей TblRole з бази даних у відповідь.

3.1.4 UserMasterController

Для доступу до цього контролера потрібна авторизація користувача. Це перевіряється за допомогою атрибута [Authorize] (Додаток Г).

Клас має наступні методи:

- Get() - HTTP GET-запит для отримання списку всіх користувачів TblUser з бази даних.
- Get(string id) - HTTP GET-запит для отримання конкретного користувача з бази даних за заданим ідентифікатором.
- Save([FromBody] TblUser value) - HTTP POST-запит для збереження (оновлення або створення нового) користувача в базі даних. В залежності від того, чи існує користувач з заданим ідентифікатором, виконується оновлення або створення нового запису TblUser.
- ActivateUser([FromBody] TblUser value) - HTTP POST-запит для активації або деактивації користувача. Змінює статус активності користувача, що визначає можливість створювати дописи.
- Delete(string id) - HTTP DELETE-запит для видалення користувача з бази даних за заданим id.

3.2 FrontEnd

Структура фронтенд частини подана у додатку Ґ.

3.2.1 App Component

Це основний компонент усього проекту. Він включає у себе наступні файли:

- app.module

  Це модуль, який містить декларації компонентів, які використовуються в додатку та імпорт інших модулів, які потрібні для розробки додатку. Наприклад, BrowserModule забезпечує необхідні ресурси для роботи з браузером, FormsModule і ReactiveFormsModule додають підтримку для роботи з формами, HttpClientModule дозволяє здійснювати HTTP-запити та інші. Також у модулі визначаються провайдери сервісів, які надають доступ до певних функцій або даних в додатку. AppModule також визначає головну точку входу bootstrap додатку, якою є компонент AppComponent. Це означає, що AppComponent буде першим компонентом, який відображатиметься при запуску додатку.

- app.component.ts

  Отримує роль користувача, та передає її на фронтенд, щоб правильно відобразити доступний функціонал.

- app-routing.module.ts

  Налаштовує шляхи для усіх компонентів програми (Додаток Д).

3.2.2 Status Component

Компонент для відображення сторінки 404.

3.2.3 Services

У цій директорії зберігаються усі сервіси, які є на стороні FrontEnd.

3.2.3.1 TokenInterceptorService

Цей сервіс використовується для додавання токену авторизації до заголовків кожного HTTP-запиту.

3.2.3.2 UserAdminService & UserService

Ці два сервіси використовуються для зв’язку з UserMasterController та UserController.

3.2.4 Login Component

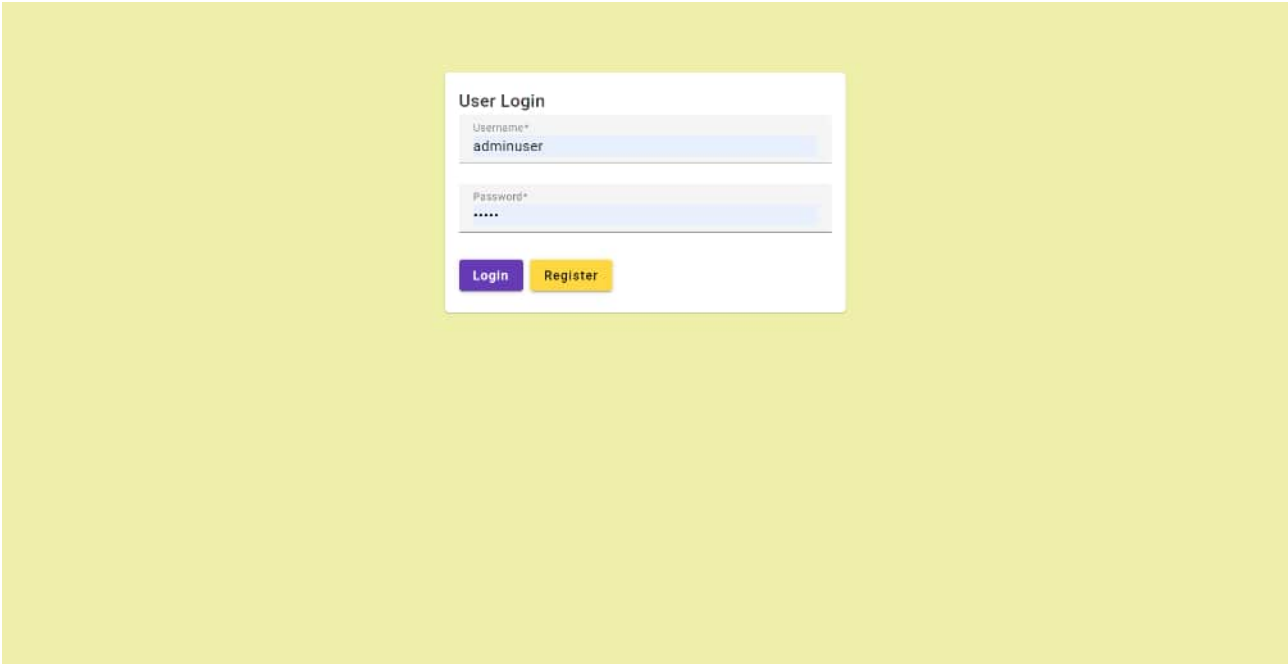
Цей компонент відповідає за логін користувача та збереження jwt токена у localStorage. Також він перенаправляє на реєстрацію та очищує localStorage при ініціалізації класу.

3.2.5 User Components

Усі інші компоненти можна узагальнити назвою User Components. Вони використовують сервіси для отримання даних з API та виводять після обробки отриману інформацію на екран користувача.

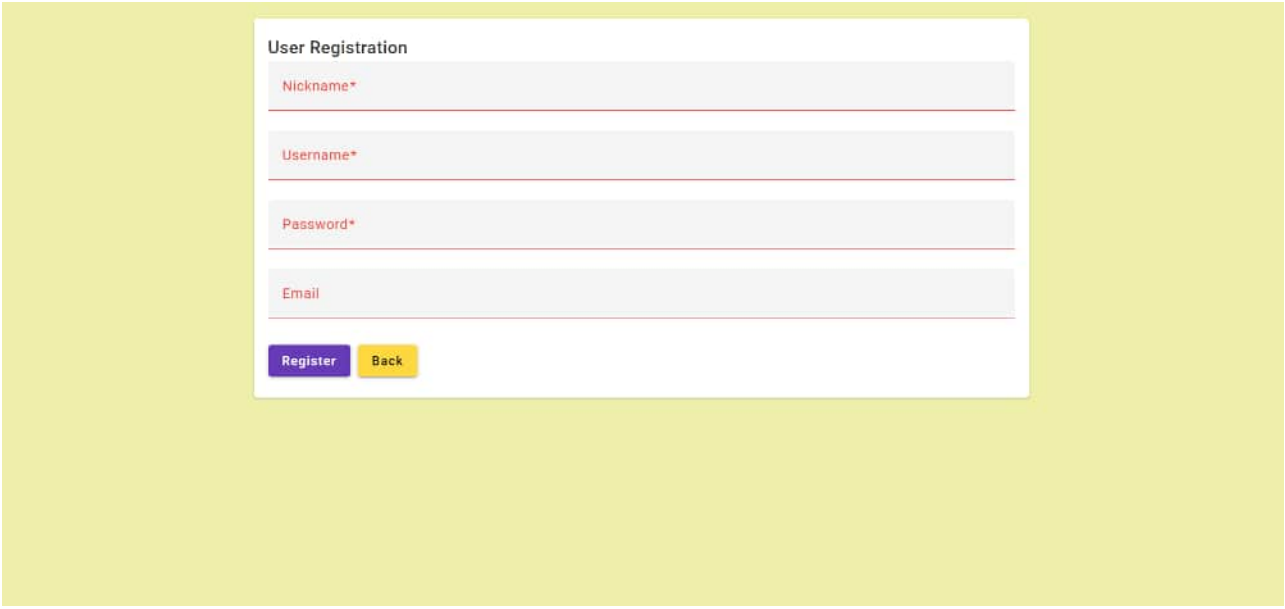
4. ДЕМОНСТРАЦІЯ ВЕБСАЙТУ

Коли користувач заходить на платформу, він бачить сторінку логіну (Малюнок 3). Інші сторінки йому не доступні до моменту, поки він не увійде на сайт або не зареєструється.



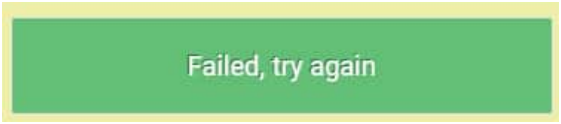
Малюнок 3 Екран логіну

При переході на сторінку реєстрації, користувача чекає екран з наступними обов’язковими полями (Малюнок 4). Валідація перевіряє правильність введених даних як на фронтенді, так і на бекенді.

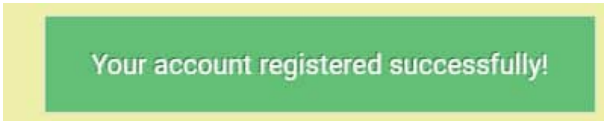


Малюнок 4 Екран реєстрації

Будь-яка дія користувача, яка передбачає зміну даних, реєстрацію або інше, супроводжується відповідним повідомленням про не успішність (Малюнок 5), або успішність (Малюнок 6) здійсненої операції справого низу вікна.

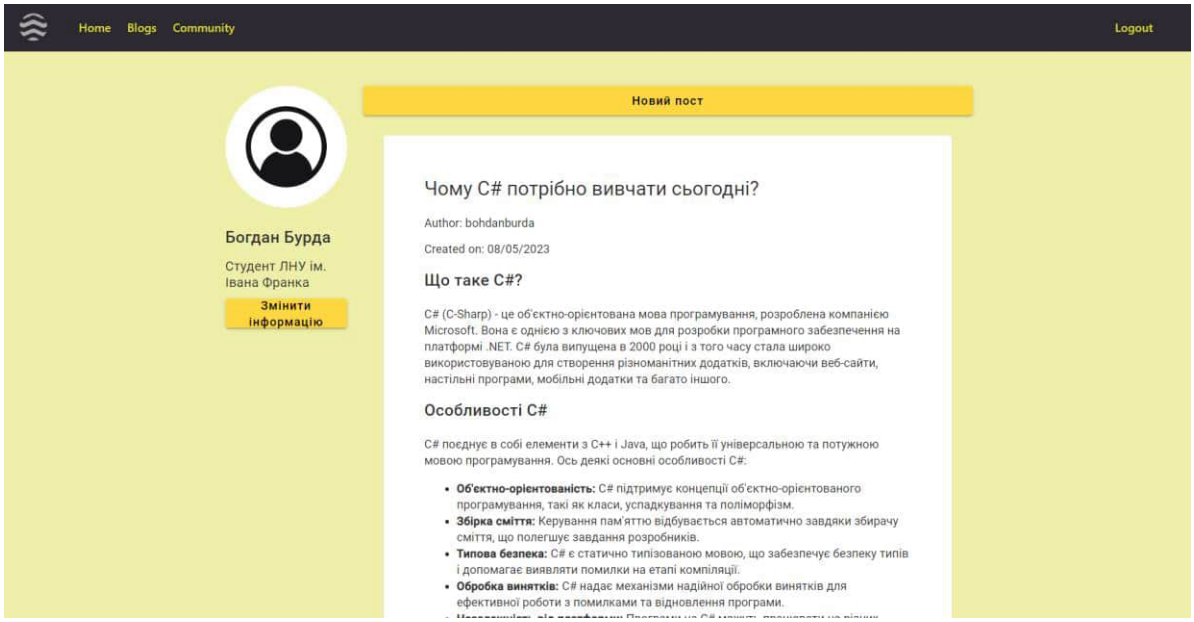


Малюнок 5 Невдалий логін



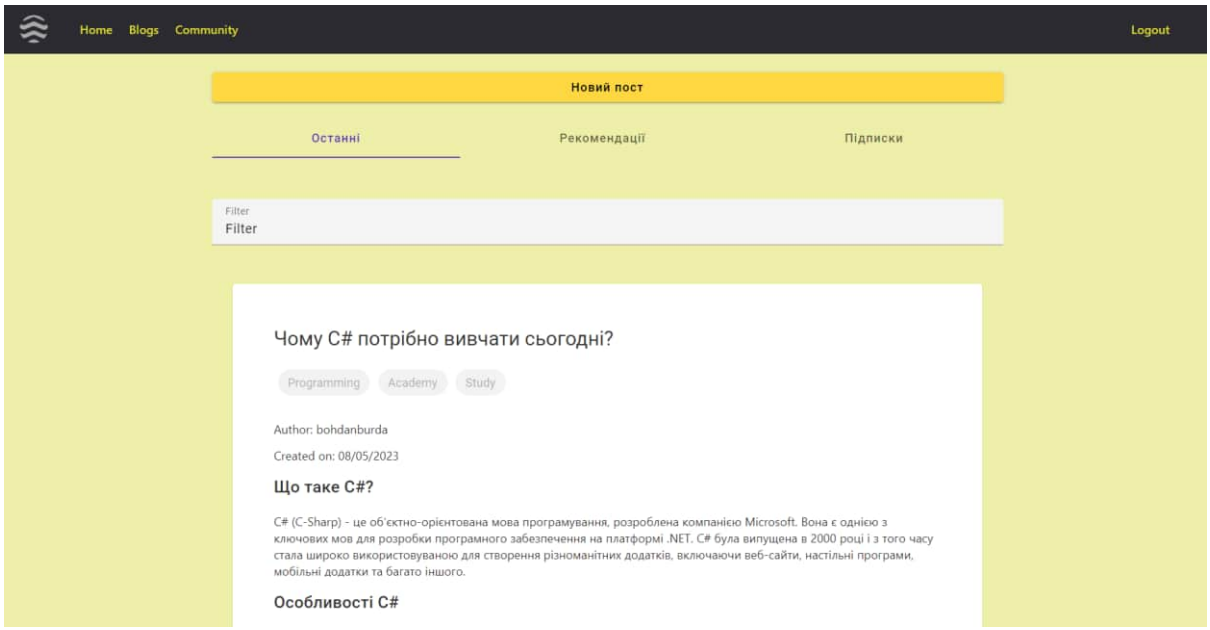
Малюнок 6 Успішна реєстрація

Після успішного логіну користувач потрапляє на свою сторінку блогу (Малюнок 7). Тут він може змінити інформацію про себе, додати новий пост у свій блог та переглянути вже створені пости.



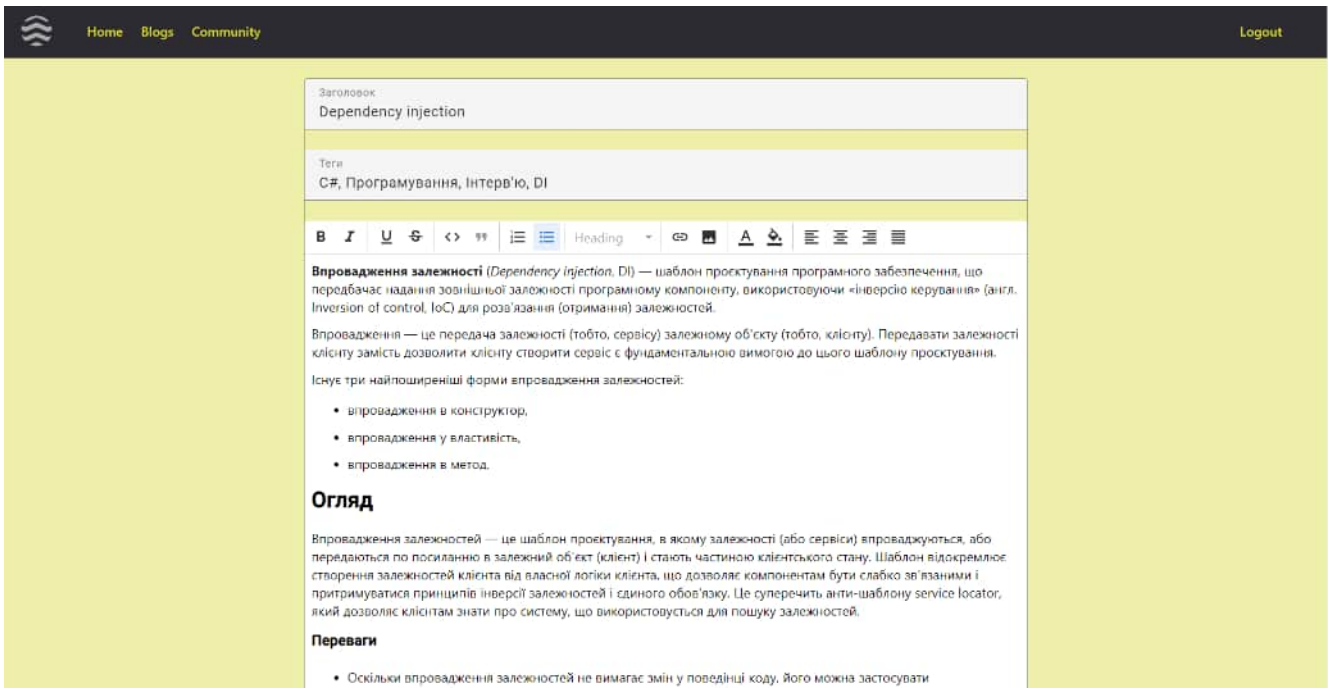
Малюнок 7 Сторінка користувача

На сторінці блогів користувач може переглянути останні додані блоги на платформу (Малюнок 8). Відфільтрувати їх за категоріями. Також переглянути рекомендації, у яких спочатку будуть ті дописи, які містять в собі теги, пости з якими користувач вподобував найчастіше. І остання вкладка це підписки. Тут можна знайти усі пости від людей, за якими користувач стежить.



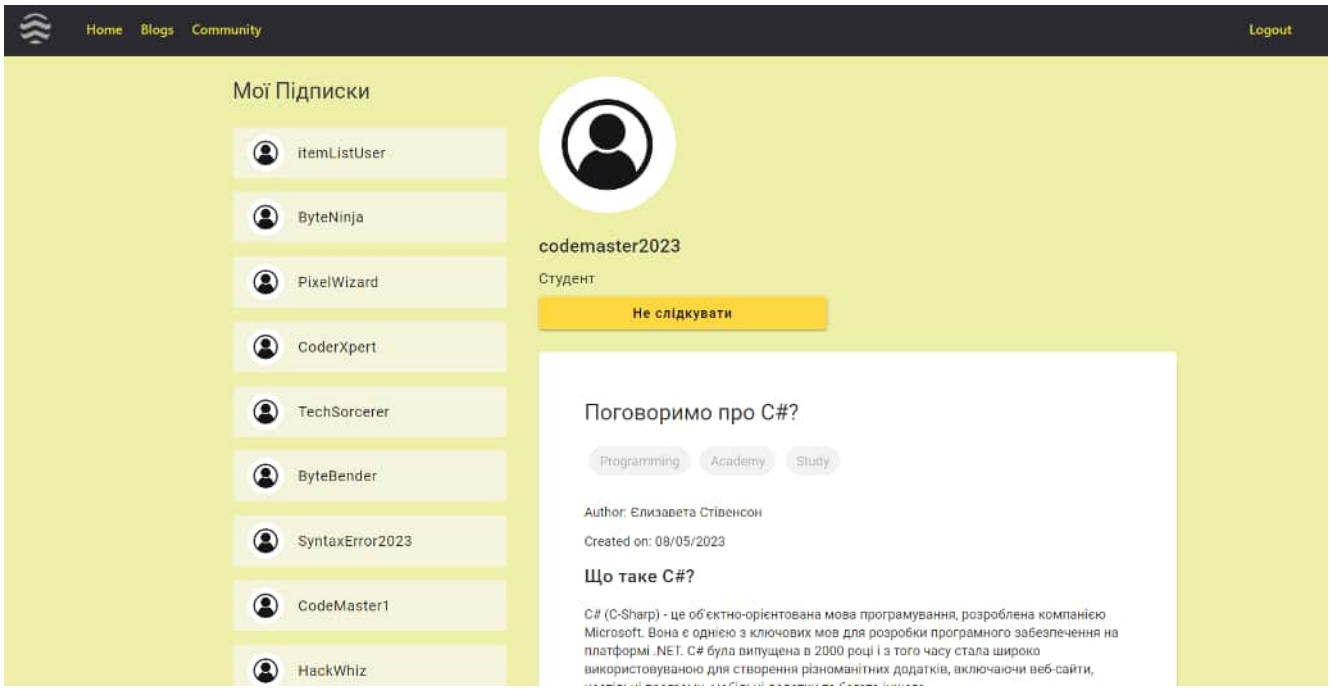
Малюнок 8 Сторінка блогів

Коли користувач хоче додати новий пост, то у нього відкривається вікно з наступним функціоналом (Малюнок 9). Тут він може додати заголовок, теги та у зручному редакторі відредагувати свій пост для подальшої публікації.



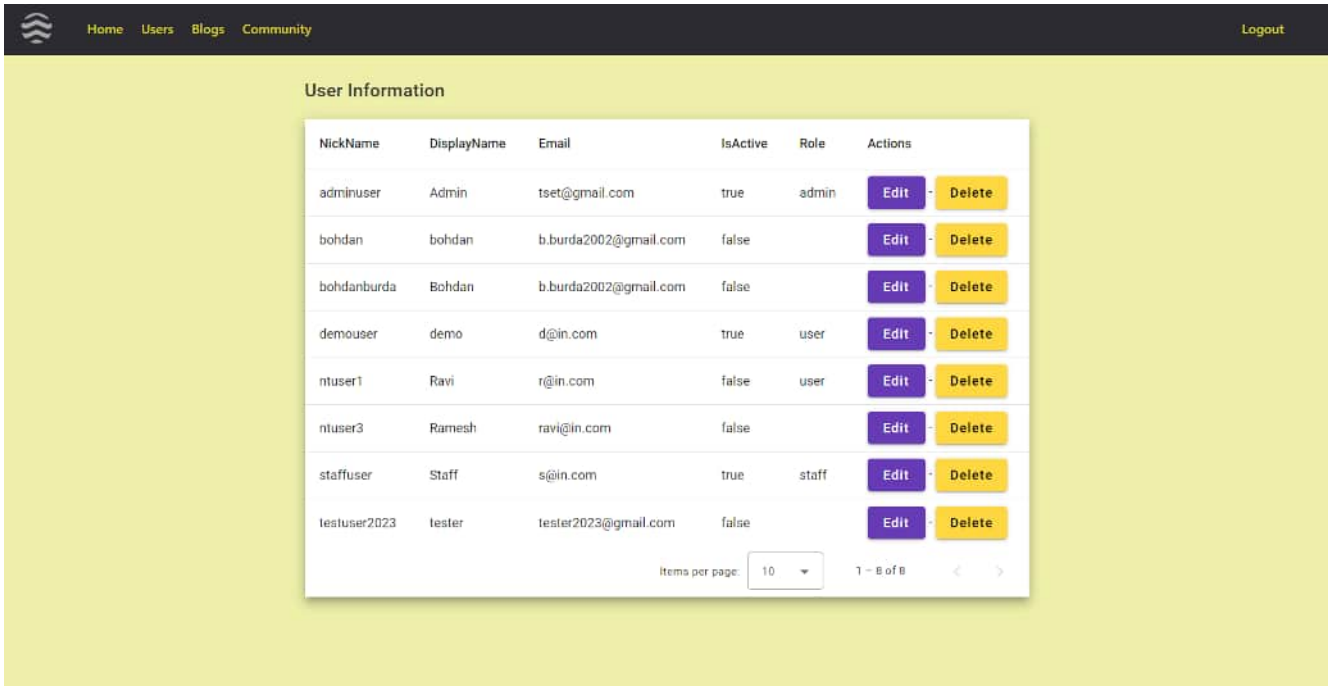
Малюнок 9 Сторінка створення нового посту

У вкладці спільнота користувач може знайти свої підписки та окремо переглянути блог кожного, за ким він слідкує (Малюнок 10).



Малюнок 10 Сторінка спільноти

Якщо увійти на сайт за логіном та паролем адміністратора, то з'явиться ще одна вкладка "Користувачі" (Малюнок 11). Тут адміністратор може переглянути кожного користувача, змінити його роль, дозволити/заборонити створювати нові записи та видалити користувача за потреби.



Малюнок 11 Сторінка адміністратора

ВИСНОВКИ

У результаті дипломної роботи був розроблений вебсайт для блогів з використанням технологій Angular та .NET Web API. Процес розробки включав у себе створення функціонального та привабливого вебінтерфейсу для користувачів, а також налаштування серверної частини для забезпечення зберігання та доступу до даних.

У дипломній роботі було використано фреймворк Angular для розробки клієнтської частини вебсайту. За допомогою Angular були створені різноманітні компоненти, які взаємодіють з користувачем, дозволяючи йому переглядати, додавати, редагувати та видаляти різну інформацію на сайті. Також були використані різні модулі та сервіси Angular для забезпечення ефективного управління даними та навігації по вебсторінках.

Для забезпечення зв'язку між клієнтською та серверною частинами було використано .NET Web API. Це дозволило реалізувати API для обміну даними між клієнтом та сервером. За допомогою .NET Web API було налаштовано маршрутизацію, контролери та моделі, що дозволило обробляти запити від клієнта та забезпечувати доступ до бази даних для зберігання блогів, користувачів та інших пов'язаних даних.

Під час розробки вебсайту були враховані принципи дизайну із використанням сучасних інструментів та бібліотек для створення функціонального інтерфейсу. Було реалізовано унікальний вебсайт, який дає змогу вести свій блог та зручно знаходити блоги інших авторів. Завдяки рекомендаціям та фільтрам, функціонал яких реалізований у ході виконання роботи, цим вебсайтом можуть користуватись усі категорії користувачів, від учнів у школі до викладачів в університеті. Було використано переваги сучасних фреймворків для розробки сайту, що зробило його легко підтримуваним, функціональним та простим у використанні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 10 найкращих безкоштовних сайтів для створення блогів [Електронний ресурс] - https://www.bloggersideas.com/uk/10-free-blogging-sites-list-to-create-blogs/
2. The Clean Code Blog by Robert C. Martin (Uncle Bob) [Електронний ресурс] - https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
3. Документація Angular [Електронний ресурс] - https://angular.io/guide/what-is-angular
4. Документація Angular Material UI [Електронний ресурс] - https://material.angular.io/guides
5. Mastering ASP.NET Web API: build powerful HTTP services and make the most of the ASP.NET Core Web API platform / Mithun Pattankar, Malendra Hurbuns – 2017

ДОДАТОК А

```
0 references
public void ConfigureServices(IServiceCollection services)
{

    services.AddControllers();
    services.AddDbContext<Blog_DBContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("constring")));

    var _dbcontext = services.BuildServiceProvider().GetService<Blog_DBContext>();

    services.AddSingleton<IRefreshTokenGenerator>(provider => new RefreshTokenGenerator(_dbcontext));

    var _jwtsetting = Configuration.GetSection("JWTSetting");
    services.Configure<JWTSetting>(_jwtsetting);

    var authkey = Configuration.GetValue<string>("JWTSetting:securityKey");

    services.AddAuthentication(item =>
    {
        item.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        item.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    }).AddJwtBearer(item =>
    {

        item.RequireHttpsMetadata = true;
        item.SaveToken = true;
        item.TokenValidationParameters = new TokenValidationParameters()
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(authkey)),
            ValidateIssuer = false,
            ValidateAudience = false
        };
    });

    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "API", Version = "v1" });
    });
}
```

ДОДАТОК Б

```
3 references
public string GenerateToken(string username)
{
    var randomnumber = new byte[32];
    using (var randomnumbergenerator = RandomNumberGenerator.Create())
    {
        randomnumbergenerator.GetBytes(randomnumber);
        string RefreshToken = Convert.ToBase64String(randomnumber);

        var _user = context.TblRefreshtoken.FirstOrDefault(o => o.UserId == username);
        if (_user != null)
        {
            _user.RefreshToken = RefreshToken;
            context.SaveChanges();
        }
        else
        {
            TblRefreshtoken tblRefreshtoken = new TblRefreshtoken()
            {
                UserId = username,
                TokenId = new Random().Next().ToString(),
                RefreshToken = RefreshToken,
                IsActive = true
            };
        }

        return RefreshToken;
    }
}
```

ДОДАТОК В

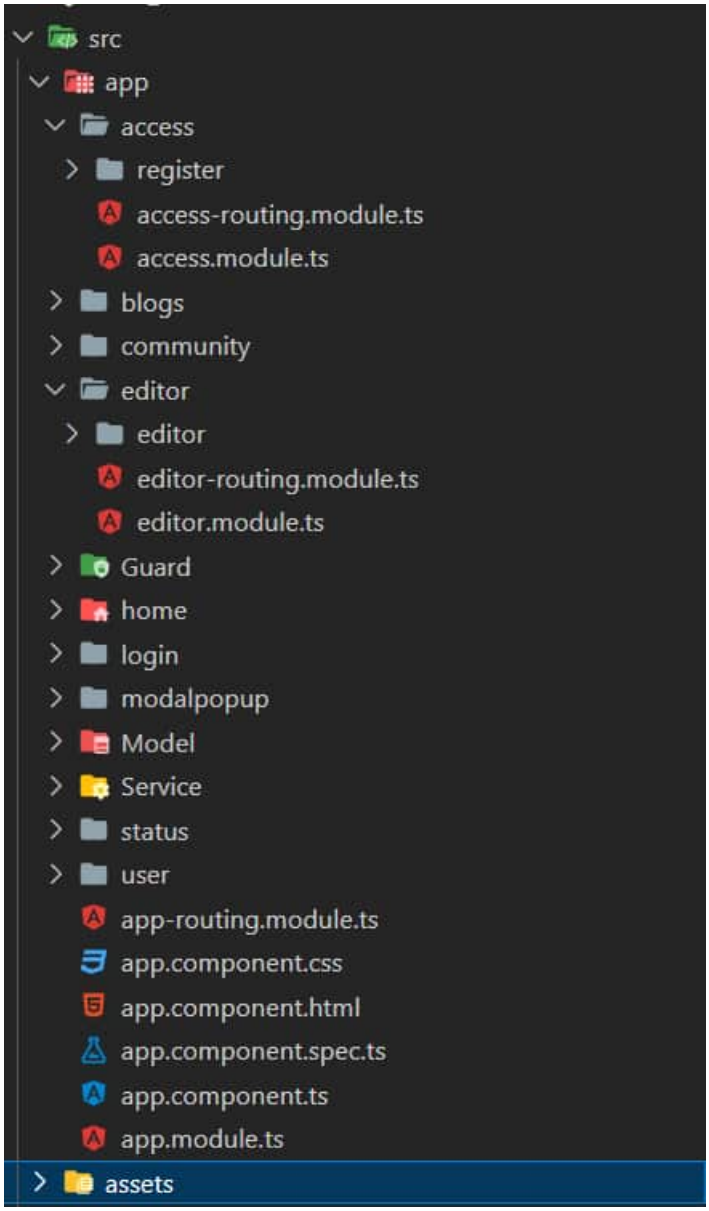
```
[NonAction]
1 reference
public TokenResponse Authenticate(string username, Claim[] claims)
{
    TokenResponse tokenResponse = new TokenResponse();

    var tokenkey = Encoding.UTF8.GetBytes(setting.securityKey);
    var tokenhandler = new JwtSecurityToken(
    claims: claims,
    expires: DateTime.Now.AddMinutes(60),
        signingCredentials: new SigningCredentials(new SymmetricSecurityKey(tokenkey), SecurityAlgorithms.HmacSha256)

    );
    tokenResponse.JWTToken = new JwtSecurityTokenHandler().WriteToken(tokenhandler);
    tokenResponse.RefreshToken = tokenGenerator.GenerateToken(username);

    return tokenResponse;
}
```

```
[Route("Authenticate")]
[HttpPost]
0 references
public IActionResult Authenticate([FromBody] Usercred user)
{
    TokenResponse tokenResponse = new TokenResponse();

    var _user = context.TblUser.FirstOrDefault(o => o.Userid == user.username && o.Password == user.password);
    if (_user == null)
        return Unauthorized();

    var tokenhandler = new JwtSecurityTokenHandler();
    var tokenkey = Encoding.UTF8.GetBytes(setting.securityKey);
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(
            new Claim[]
            {
                new Claim(ClaimTypes.Name, _user.Userid),
                new Claim(ClaimTypes.Role, _user.Role)

            }
        ),
        Expires = DateTime.Now.AddMinutes(20),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(tokenkey), SecurityAlgorithms.HmacSha256)
    };
    var token = tokenhandler.CreateToken(tokenDescriptor);
    string finaltoken = tokenhandler.WriteToken(token);

    tokenResponse.JWTToken = finaltoken;
    tokenResponse.RefreshToken = tokenGenerator.GenerateToken(user.username);

    return Ok(tokenResponse);
}
```

ДОДАТОК Г

```
[Authorize]
[Route("api/[controller]")]
[ApiController]
1 reference
public class UserMasterController : ControllerBase
{
    private readonly Blog_DBContext context;
    0 references
    public UserMasterController(Blog_DBContext learn_DB)
    {
        context = learn_DB;
    }
```

ДОДАТОК Ґ


src
app
access
register
access-routing.module.ts
access.module.ts
blogs
community
editor
editor
editor-routing.module.ts
editor.module.ts
Guard
home
login
modalpopup
Model
Service
status
user
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
assets

ДОДАТОК Д

```
const routes: Routes = [
  {path: "home", component:HomeComponent, canActivate:[AuthGuard]},
  {path: "user", component:UserComponent, canActivate:[AuthGuard]},
  {path: "blogs", component:BlogsComponent, canActivate:[AuthGuard]},
  {path: "community", component:CommunityComponent, canActivate:[AuthGuard]},
  {path: "editor", loadChildren:()=>import('./editor/editor.module').then(opt=>opt.EditorModule)},
  {path: "access", loadChildren:()=>import('./access/access.module').then(opt=>opt.AccessModule)},
  {path: "login", loadComponent:()=>import('./login/login.component').then(opt=>opt.LoginComponent)},
  {path:"**", component:StatusComponent, canActivate:[AuthGuard]}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
```

ДОДАТОК Е

```
export class LoginComponent implements OnInit {

  constructor(private service:UserService, private route: Router) {}

  ngOnInit(): void{
    localStorage.clear();
  }

  respdata:any;
  ProceedLogin(logindata:any){
    if(logindata.valid){
      this.service.ProceedLogin(logindata.value).subscribe(item=>{
        this.respdata=item;
        if(this.respdata!=null){
          localStorage.setItem('token', this.respdata.jwtToken);
          this.route.navigate(['home']);
        }else{
          alert("Login Failed");
        }
      });
    }
  }

  RedirectRegister(){
    this.route.navigate(['access/register']);
  }

}
```