Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«РОЗРОБКА ВЕБ-РЕСУРСУ «ТИХЕ ПОЛЮВАННЯ»

зі спеціальності 122 Комп’ютерні науки
освітня програма «Комп’ютерні науки»
ступеня бакалавра

Виконавець роботи Троневський Владислав Валерійович
_____ «____»_____ 2026 р.
(підпис)

Науковий керівник к.ф.-м. н., доцент, Чілікіна Тетяна Василівна
_____ «____»_____ 2026 р.
(підпис)

Рецензент

ПОЛТАВА 2026 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка веб-ресурсу «Тихе полювання»»
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Троневський Владислав Валерійович
Затверджена наказом ректора № 213-Н від «01» жовтня 2025 р.
Термін подання студентом роботи «___»_____ 2026 р.
Вихідні дані до кваліфікаційної роботи: статті та документації з теми розробки веб застосунків та сайтів.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ
2.1. Аналіз існуючих вебресурсів аналогічного призначення
2.2. Порівняльна характеристика сучасних підходів і технічних реалізацій
2.3. Переваги та недоліки існуючих програмних рішень
2.4. Сучасні підходи до розробки вебресурсів для спільнот любителів активного відпочинку та риболовлі
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Характеристика використаних технологій і засобів розробки
3.2. Проєктування структури та основних модулів вебзастосунку для риболовлі
3.3. Опис архітектури системи та взаємодії її компонентів
3.4. Проєктування бази даних та моделі зберігання інформації
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка структури даних та основних сутностей вебзастосунку
4.2. Реалізація клієнтської частини та динамічної поведінки вебзастосунку
4.3. Розробка користувацького інтерфейсу та основних функціональних можливостей
4.4. Практичне використання та особливості функціонування вебзастосунку
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Перелік графічного матеріалу: 2 аркуші блок-схем, 19 ілюстрацій.
Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Чілікіна Т.В.		
Інформаційний огляд	Чілікіна Т.В.		
Теоретична частина	Чілікіна Т.В.		
Практична частина	Чілікіна Т.В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
Вступ		
Вивчення методичних рекомендацій та стандартів та звіт керівнику		
Постановка задачі		
Інформаційний огляд джерел бібліотек та інтернету		
Теоретична частина		
Практична частина		
Закінчення оформлення		
Доповідь студента на кафедрі		
Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 2025 р.

Здобувач вищої освіти _____ Троневський Владислав Валерійович
(підпис)

Науковий керівник _____ к.ф.-м.н., доц. Чілікіна Т.В.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2026 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 2025 р.

Погоджено
Науковий керівник _____
к.пед.н., Тетяна ЧІЛІКІНА
«____» _____ 2025 р.

План

дипломного проекту з фаху
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
на тему «Розробка веб-ресурсу «Тихе полювання»»
Прізвище, ім'я, по батькові Троневський Владислав Валерійович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ
2.1. Аналіз існуючих вебресурсів аналогічного призначення
2.2. Порівняльна характеристика сучасних підходів і технічних реалізацій
2.3. Переваги та недоліки існуючих програмних рішень
2.4. Сучасні підходи до розробки вебресурсів для спільнот любителів активного відпочинку та риболовлі
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Характеристика використаних технологій і засобів розробки
3.2. Проєктування структури та основних модулів вебзастосунку для риболовлі
3.3. Опис архітектури системи та взаємодії її компонентів
3.4. Проєктування бази даних та моделі зберігання інформації
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. Розробка структури даних та основних сутностей вебзастосунку
4.2. Реалізація клієнтської частини та динамічної поведінки вебзастосунку
4.3. Розробка користувацького інтерфейсу та основних функціональних можливостей
4.4. Практичне використання та особливості функціонування вебзастосунку
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

Здобувач вищої освіти _____ В.В. Троневський
(підпис)
«____» _____ 2025 р.

РЕФЕРАТ

ВЕБЗАСТОСУНОК, РИБОЛОВЛЯ, ASP.NET CORE MVC, ENTITY FRAMEWORK CORE, SQL SERVER, БЛОГ, КОРИСТУВАЧ, КОМЕНТУВАННЯ, ВЕБІНТЕРФЕЙС, ІНФОРМАЦІЙНИЙ РЕСУРС.

Записка: 103 сторінки, 19 рисунків, 1 додаток, 27 літературних джерела.

Дипломна робота присвячена розробці вебзастосунку для любителів риболовлі, призначеного для публікації інформаційних матеріалів, обміну досвідом між користувачами та організації тематичної онлайн-спільноти. Актуальність роботи зумовлена зростанням популярності веборієнтованих інформаційних ресурсів, які забезпечують оперативне поширення тематичного контенту, підтримують комунікацію між користувачами та створюють умови для формування спільнот за інтересами.

Метою роботи є розробка вебзастосунку для любителів риболовлі, призначеного для публікації інформаційних матеріалів, обміну досвідом та взаємодії користувачів.

Об’єктом дослідження є процес створення та використання веборієнтованих інформаційних систем для організації тематичних онлайн-спільнот.

Предметом дослідження є методи, засоби та програмні технології розробки вебзастосунків для публікації та обміну інформацією між користувачами.

У ході виконання дипломної роботи проведено аналіз існуючих вебресурсів для любителів риболовлі та активного відпочинку, досліджено особливості їх функціонування, інтерфейсні рішення та технологічні підходи до реалізації. На основі проведеного аналізу визначено функціональні вимоги до розроблюваного програмного забезпечення та сформовано структуру майбутнього вебзастосунку.

У роботі спроєктовано архітектуру вебзастосунку, структуру бази даних, функціональні модулі та механізми взаємодії користувачів із системою.

Розроблено клієнтську та серверну частини програмного забезпечення, реалізовано механізми керування контентом, навігації, публікації матеріалів, коментування та роботи з графічними файлами.

Для реалізації проєкту використано платформу ASP.NET Core MVC, мову програмування C#, технологію Entity Framework Core, систему керування базами даних Microsoft SQL Server, а також HTML5, CSS3, Bootstrap та JavaScript. Обраний стек технологій забезпечив створення сучасного вебзастосунку з багаторівневою архітектурою та можливістю подальшого розширення функціональних можливостей.

У системі реалізовано модуль блогу для публікації інформаційних матеріалів, механізми категоризації та тегування контенту, систему коментування, форму зворотного зв’язку, адміністрування інформаційного наповнення ресурсу та засоби роботи із мультимедійним контентом. Передбачено можливість централізованого керування публікаціями, категоріями, тегами та іншими елементами вебресурсу через адміністративну панель.

У процесі виконання роботи було реалізовано адаптивний користувацький інтерфейс, який забезпечує коректне відображення сторінок на різних типах пристроїв. Розроблена структура даних забезпечує ефективне зберігання інформації та підтримує взаємозв’язки між основними сутностями предметної області.

Отже, розроблений вебзастосунок може бути використаний як практичний інструмент для організації тематичної онлайн-спільноти любителів риболовлі, публікації інформаційних матеріалів та забезпечення взаємодії між користувачами.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 12
РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ 15
2.1. Аналіз існуючих вебресурсів аналогічного призначення 15
2.2. Порівняльна характеристика сучасних підходів і технічних реалізацій 21
2.3. Переваги та недоліки існуючих програмних рішень 25
2.4. Сучасні підходи до розробки вебресурсів для спільнот любителів активного відпочинку та риболовлі 29
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА 34
3.1. Характеристика використаних технологій і засобів розробки 34
3.2. Проєктування структури та основних модулів вебзастосунку для риболовлі 38
3.3. Опис архітектури системи та взаємодії її компонентів 43
3.4. Проєктування бази даних та моделі зберігання інформації 50
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА 56
4.1. Розробка структури даних та основних сутностей вебзастосунку 56
4.2. Реалізація серверної частини інформаційної системи 60
4.3. Розробка користувацького інтерфейсу та основних функціональних можливостей 65
4.4. Практичне використання та особливості функціонування вебзастосунку 70
ВИСНОВКИ 77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТОК А 82

ВСТУП

Інформаційні технології відіграють важливу роль у розвитку сучасних онлайн-спільнот та сервісів, орієнтованих на обмін інформацією між користувачами. Активне поширення веборієнтованих платформ сприяє створенню спеціалізованих інформаційних ресурсів для представників різних сфер діяльності та захоплень. Одним із таких напрямів є риболовля, яка об’єднує значну кількість людей, зацікавлених в обміні досвідом, публікації власних досягнень, обговоренні спорядження та пошуку інформації про місця для риболовлі. У зв’язку зі зростанням популярності тематичних інтернет-спільнот актуальним завданням є розробка вебзастосунків, що забезпечують ефективну взаємодію користувачів, централізоване зберігання інформації та зручний доступ до контенту.

Сучасні вебресурси для любителів риболовлі виконують не лише інформаційну функцію, а й виступають засобом комунікації між користувачами. Такі системи дають змогу створювати публікації, додавати фотографії уловів, залишати коментарі, обмінюватися рекомендаціями та накопичувати корисні матеріали. Проте значна частина існуючих рішень має обмежений функціонал, недостатньо зручний інтерфейс або не забезпечує належного рівня структуризації інформації. Це зумовлює необхідність створення спеціалізованих вебзастосунків, орієнтованих на потреби користувачів рибальської спільноти [8, 19].

Актуальність теми визначається потребою у створенні сучасного вебзастосунку для любителів риболовлі, який забезпечуватиме можливість публікації матеріалів, обміну досвідом між користувачами та централізованого зберігання інформації. Використання вебтехнологій дозволяє реалізувати доступ до системи незалежно від місця перебування користувача та типу пристрою, а також забезпечити зручне адміністрування контенту та підтримку взаємодії між учасниками спільноти.

Метою роботи є розробка вебзастосунку для любителів риболовлі, призначеного для публікації інформаційних матеріалів, обміну досвідом та взаємодії користувачів.

Об’єктом дослідження є процес створення та використання веборієнтованих інформаційних систем для організації тематичних онлайн-спільнот.

Предметом дослідження є методи, засоби та програмні технології розробки вебзастосунків для публікації та обміну інформацією між користувачами.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз існуючих вебресурсів для любителів риболовлі та активного відпочинку;

– визначити функціональні та нефункціональні вимоги до програмного продукту;

– обґрунтувати вибір технологій і засобів розробки;

– спроєктувати архітектуру вебзастосунку та структуру бази даних;

– реалізувати серверну частину програмного забезпечення; розробити клієнтський вебінтерфейс;

– реалізувати механізми реєстрації та автентифікації користувачів;

– забезпечити створення, редагування та перегляд публікацій; реалізувати механізми коментування та взаємодії між користувачами;

– забезпечити можливість завантаження та зберігання зображень;

– реалізувати адміністративні функції керування контентом.

Під час виконання роботи використано методи аналізу предметної області, методи проєктування інформаційних систем, принципи побудови багаторівневих вебзастосунків, технології клієнт-серверної взаємодії, а також підходи до розробки програмного забезпечення на основі архітектурного шаблону MVC. Архітектура розробленої системи базується на розподілі

функціональності між рівнем представлення, рівнем обробки даних та рівнем зберігання інформації.

Серверна частина системи реалізована із використанням платформи ASP.NET Core MVC, яка забезпечує маршрутизацію запитів, взаємодію між контролерами та представленнями, а також підтримку механізмів обробки даних. Для роботи з базою даних використано Entity Framework Core, що реалізує об’єктно-реляційне відображення та забезпечує виконання операцій створення, читання, оновлення і видалення даних. Для формування запитів до бази даних використано технологію LINQ. Реалізація доступу до даних здійснюється із застосуванням шаблону Repository Pattern.

Для зберігання інформації використано систему керування базами даних Microsoft SQL Server. Основними сутностями системи є користувачі, публікації, коментарі та допоміжні об’єкти, необхідні для функціонування вебзастосунку. Реалізована структура даних забезпечує зберігання інформації про користувачів, контент і взаємодію між учасниками спільноти.

Клієнтська частина застосунку розроблена із використанням технології Razor-представлення (.cshtml), мов HTML5, CSS3 та JavaScript. Для побудови адаптивного інтерфейсу використано фреймворк Bootstrap. Реалізований користувацький інтерфейс забезпечує перегляд публікацій, створення нового контенту, коментування матеріалів та виконання адміністративних операцій. Для завантаження та зберігання графічних матеріалів використано спеціалізований сервіс роботи з файлами.

Практичне значення роботи полягає у створенні вебзастосунку, який забезпечує організацію інформаційного простору для любителів риболовлі, спрощує обмін досвідом між користувачами, надає можливість публікації матеріалів та сприяє формуванню тематичної онлайн-спільноти.

Пояснювальна записка складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі сформульовано постановку задачі та визначено основні вимоги до програмного

забезпечення. У другому розділі проведено аналіз існуючих вебресурсів аналогічного призначення та розглянуто сучасні підходи до розробки вебзастосунків. У третьому розділі виконано проєктування системи, розроблено її архітектуру, структуру бази даних та основні функціональні модулі. У четвертому розділі описано реалізацію вебзастосунку, особливості побудови користувацького інтерфейсу, реалізований функціонал та практичне використання програмного продукту.

.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Розвиток інформаційних технологій та широке поширення мережі Інтернет сприяли появі великої кількості веборієнтованих ресурсів, призначених для обміну інформацією між користувачами. Особливого значення набувають спеціалізовані вебзастосунки, орієнтовані на певні групи користувачів та конкретні предметні області. Однією з таких сфер є риболовля, яка поєднує значну кількість людей, зацікавлених у пошуку корисної інформації, обговоренні спорядження, обміні досвідом та демонстрації власних досягнень.

Сучасні користувачі активно використовують цифрові засоби комунікації для поширення інформації про результати риболовлі, особливості використання рибальського спорядження, місця відпочинку та способи лову риби. Найчастіше для цього застосовуються соціальні мережі, тематичні форуми та різноманітні інформаційні портали. Проте такі ресурси не завжди забезпечують достатній рівень структурованості інформації, ефективні механізми пошуку та зручне адміністрування контенту.

Предметна область дослідження охоплює процеси створення, зберігання, обробки та поширення інформації, пов'язаної з риболовлею. Основними учасниками цих процесів є користувачі вебзастосунку, які мають можливість переглядати матеріали, створювати власні публікації, завантажувати зображення, залишати коментарі та взаємодіяти з іншими учасниками спільноти. Для забезпечення ефективної роботи системи необхідно реалізувати централізоване зберігання інформації та механізми керування контентом.

Аналіз особливостей предметної області свідчить про необхідність створення спеціалізованого програмного забезпечення, яке дозволить об'єднати функції інформаційного порталу та платформи для спілкування користувачів. Такий підхід забезпечує зручний доступ до тематичних матеріалів, спрощує процес публікації інформації та підвищує ефективність взаємодії між учасниками спільноти.

Розроблюваний вебзастосунок повинен забезпечувати можливість реєстрації та автентифікації користувачів, створення та редагування публікацій, перегляду інформаційних матеріалів, додавання коментарів і завантаження графічного контенту. Крім того, система повинна підтримувати функції адміністрування, що дозволяють контролювати розміщені матеріали та підтримувати належний рівень інформаційної безпеки [23, 24, 26, 27].

З урахуванням особливостей предметної області до програмного забезпечення висуваються функціональні та нефункціональні вимоги. До функціональних вимог належать реєстрація користувачів, авторизація в системі, створення та редагування публікацій, зберігання фотографій, коментування записів, перегляд інформаційних матеріалів і керування контентом. До нефункціональних вимог належать зручність використання, надійність роботи, забезпечення цілісності даних, масштабованість, безпека та підтримка роботи у сучасних веббраузерах.

Для реалізації поставлених вимог обрано архітектурний підхід MVC, який забезпечує розподіл функціональності між моделями, представленнями та контролерами. Серверна частина системи реалізується засобами платформи ASP.NET Core MVC із використанням технології Entity Framework Core для роботи з базою даних. Для побудови користувацького інтерфейсу використовуються Razor-представлення, HTML5, CSS3, JavaScript та Bootstrap. Зберігання інформації здійснюється у системі керування базами даних Microsoft SQL Server.

Для реалізації поставленої задачі дослідження необхідно виконати аналіз предметної області та існуючих програмних рішень, визначити вимоги до програмного забезпечення, спроєктувати архітектуру системи та структуру бази даних, реалізувати серверну і клієнтську частини вебзастосунку, розробити механізми взаємодії користувачів із системою, забезпечити підтримку роботи з публікаціями та мультимедійними матеріалами, а також реалізувати функції адміністрування контенту.

Таким чином, розробка вебзастосунку для любителів риболовлі є актуальним завданням, спрямованим на створення сучасної інформаційної системи для обміну досвідом, публікації тематичних матеріалів та організації ефективної взаємодії між користувачами в межах єдиної вебплатформи.

.

РОЗДІЛ 2. ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

2.1. Аналіз існуючих вебресурсів аналогічного призначення

Стрімкий розвиток вебтехнологій та широке поширення мережі Інтернет сприяли появі значної кількості спеціалізованих інформаційних ресурсів, призначених для об'єднання користувачів за спільними інтересами. Одним із таких напрямів є риболовля, яка залишається популярним видом активного відпочинку та формує великі спільноти користувачів, зацікавлених в обміні досвідом, поширенні інформації про місця риболовлі, використанні спеціалізованого спорядження та обговоренні різноманітних технік лову риби. Для забезпечення ефективної взаємодії між учасниками таких спільнот створюються тематичні вебресурси, які реалізують функції форумів, блогів, соціальних мереж або комплексних інформаційних платформ.

Дослідження існуючих вебресурсів аналогічного призначення дозволяє визначити основні тенденції розвитку даного класу програмних систем, виявити найбільш затребувані функціональні можливості, а також встановити недоліки існуючих рішень, які можуть бути враховані під час проєктування нового програмного продукту.

Одним із найбільш відомих українських тематичних ресурсів є «Український форум рибалок» (https://uafisher.com.ua). Даний вебресурс функціонує як класичний інтернет-форум та містить значну кількість тематичних розділів, присвячених різним аспектам риболовлі. Структура ресурсу передбачає поділ інформації за категоріями, серед яких окреме місце займають обговорення рибальського спорядження, особливостей лову різних видів риби, опис водойм та звіти про риболовлю.

Основною перевагою ресурсу є велика база накопиченої інформації та активна спільнота користувачів. Завдяки тривалому періоду функціонування форуму сформовано значний архів тематичних матеріалів, який може

використовуватися як довідкова база для рибалок. Разом із тим організація взаємодії між користувачами базується на форумному принципі, який сьогодні поступово втрачає популярність через поширення більш сучасних соціальних платформ. Крім того, інтерфейс ресурсу має ознаки застарілого дизайну та не забезпечує достатнього рівня персоналізації контенту.

Наступним прикладом є ресурс «Рибалка в Україні» (https://on-fishing.com/forum/), який також реалізований у вигляді форумної системи. Головною особливістю даного вебресурсу є значна кількість тематичних розділів, присвячених водоймам різних регіонів України. Користувачі мають можливість створювати власні теми, обговорювати результати риболовлі та ділитися практичними рекомендаціями.

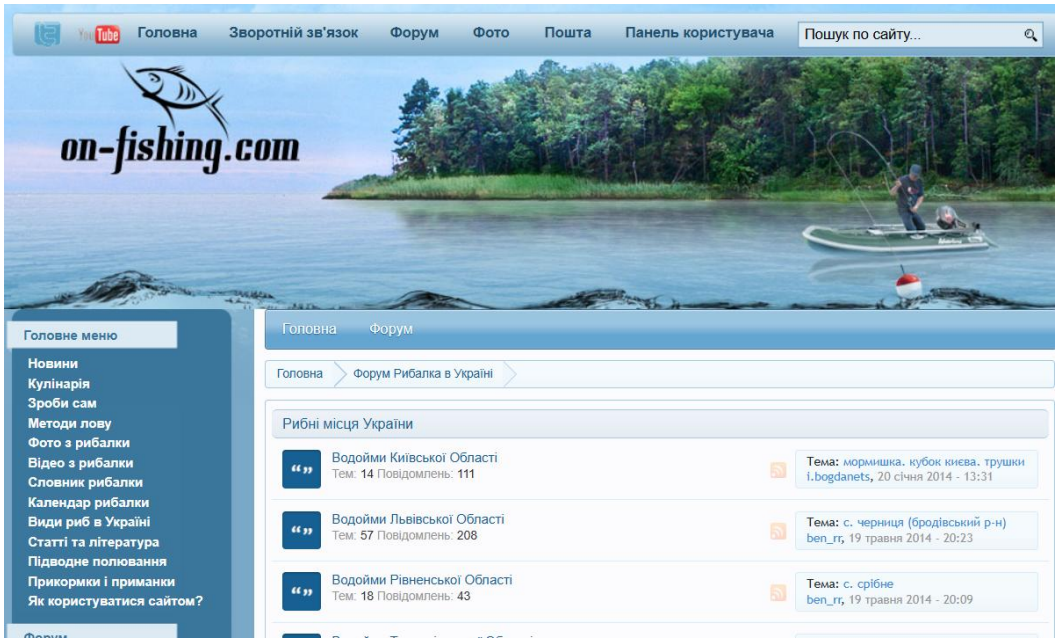


Рисунок 2.1. Сайт on-fishing.com

Перевагою цього ресурсу є високий рівень деталізації інформації про місця риболовлі та наявність великої кількості користувацьких матеріалів. Разом із тим ресурс не підтримує сучасні механізми соціальної взаємодії, характерні для сучасних вебзастосунків. Відсутність персоналізованих профілів із розширеними можливостями взаємодії та обмежена підтримка мультимедійного контенту знижують зручність використання системи.

Ще одним українським ресурсом є форум туристичного порталу Rukzak.ua (https://forum.rukzak.ua/viewforum.php?f=94), який містить окремий розділ, присвячений риболовлі. Даний ресурс дозволяє користувачам обговорювати різні аспекти рибальської діяльності та ділитися власним досвідом. Важливою перевагою ресурсу є його інтеграція із тематикою активного відпочинку та туризму, що розширює коло потенційних користувачів.

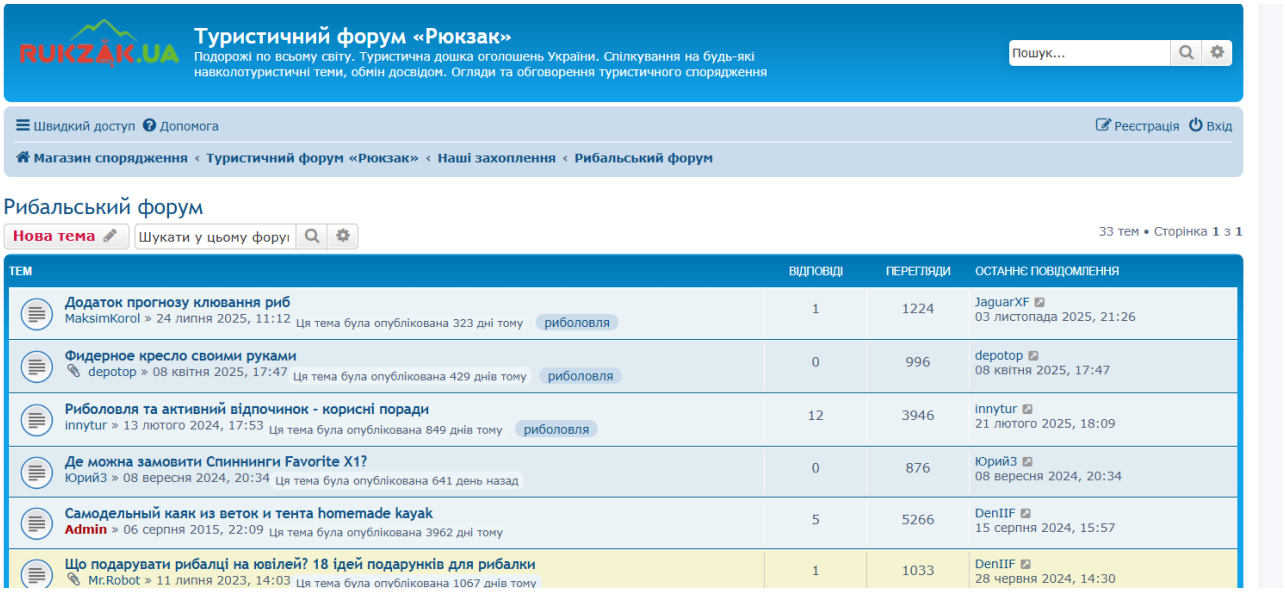


Рисунок 2.2. Туристичний форум Рюкзак

Недоліком системи є відсутність спеціалізованих функцій, орієнтованих безпосередньо на рибальську спільноту. Зокрема, користувачі не мають можливості вести персональні журнали риболовлі, створювати структуровані публікації або використовувати механізми централізованого зберігання мультимедійних матеріалів.

Окремої уваги заслуговує вебресурс громадської організації «Клуб рибалок України» (https://uafishing.club). Основне призначення цього ресурсу полягає в інформаційній підтримці діяльності організації, популяризації спортивної риболовлі та висвітленні тематичних заходів. Сайт містить новини, анонси змагань, інформацію про діяльність клубу та матеріали, присвячені рибальській тематиці.

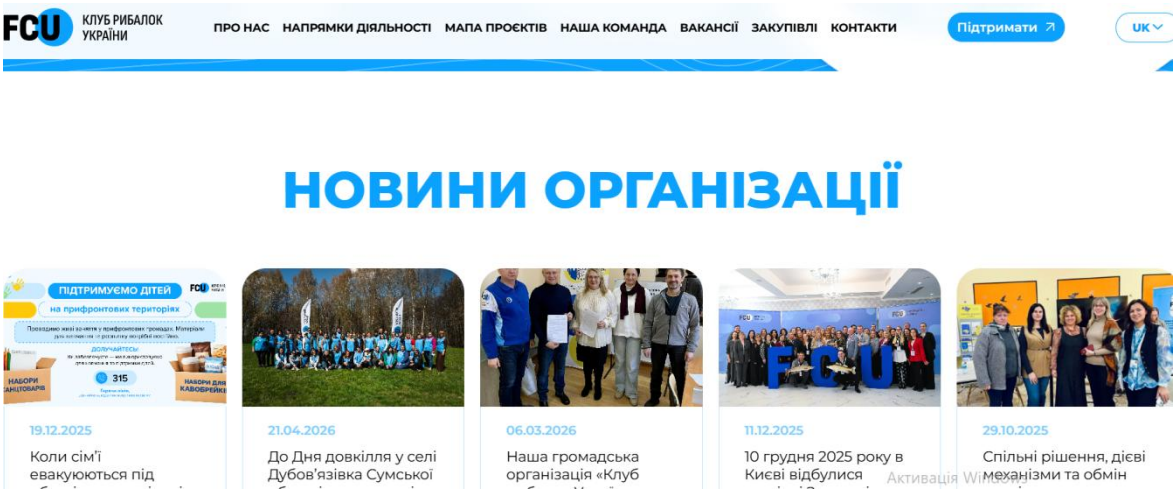


Рисунок 2.3. Вебресурс громадської організації «Клуб рибалок України»

Перевагою ресурсу є якісний інформаційний контент та орієнтація на розвиток рибальської культури. Проте його функціональні можливості не передбачають створення повноцінної соціальної платформи для взаємодії користувачів. Відсутність механізмів публікації користувацького контенту та коментування матеріалів обмежує можливості формування активної онлайн-спільноти.

Серед міжнародних вебресурсів одним із найбільш популярних є Fishbrain (https://fishbrain.com). Дана система є багатофункціональною соціальною платформою для рибалок та об'єднує мільйони користувачів із різних країн світу. Основний функціонал включає ведення електронного журналу риболовлі, публікацію фотографій уловів, визначення місць риболовлі за допомогою картографічних сервісів, аналіз статистичних даних та прогнозування активності риби.

Головною перевагою Fishbrain є комплексний підхід до організації інформації та використання сучасних технологій аналізу даних. Система забезпечує високий рівень інтерактивності та персоналізації. До недоліків належать складність окремих функціональних модулів для нових користувачів, а також обмеження доступу до частини можливостей у безкоштовній версії.

Іншим міжнародним ресурсом є FishFriender

(https://www.fishfriender.com). Платформа поєднує функціональні можливості соціальної мережі та сервісу для ведення особистої статистики риболовлі. Користувачі можуть зберігати інформацію про власні улови, аналізувати погодні умови під час риболовлі та обмінюватися матеріалами з іншими учасниками спільноти.

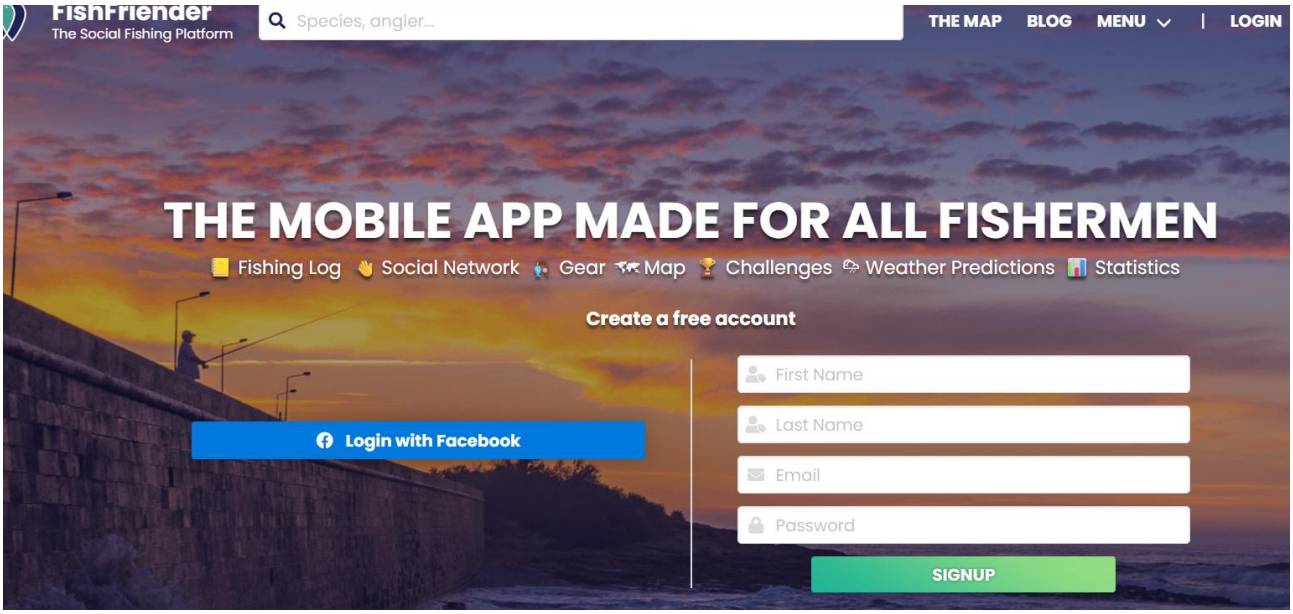


Рисунок 2.4. Платформа FishFriender

Перевагою системи є орієнтація на персоналізацію користувацького досвіду та підтримка мобільних платформ. Разом із тим вебверсія ресурсу має меншу функціональність порівняно з мобільними застосунками, що може створювати певні незручності для окремих категорій користувачів.

Значний інтерес також становить платформа Fishsurfing (https://www.fishsurfing.com), яка поєднує можливості соціальної мережі, геоінформаційної системи та тематичного інформаційного порталу. Сервіс дозволяє знаходити водойми, переглядати відгуки користувачів, публікувати власні звіти та використовувати інтерактивні карти для пошуку місць риболовлі.

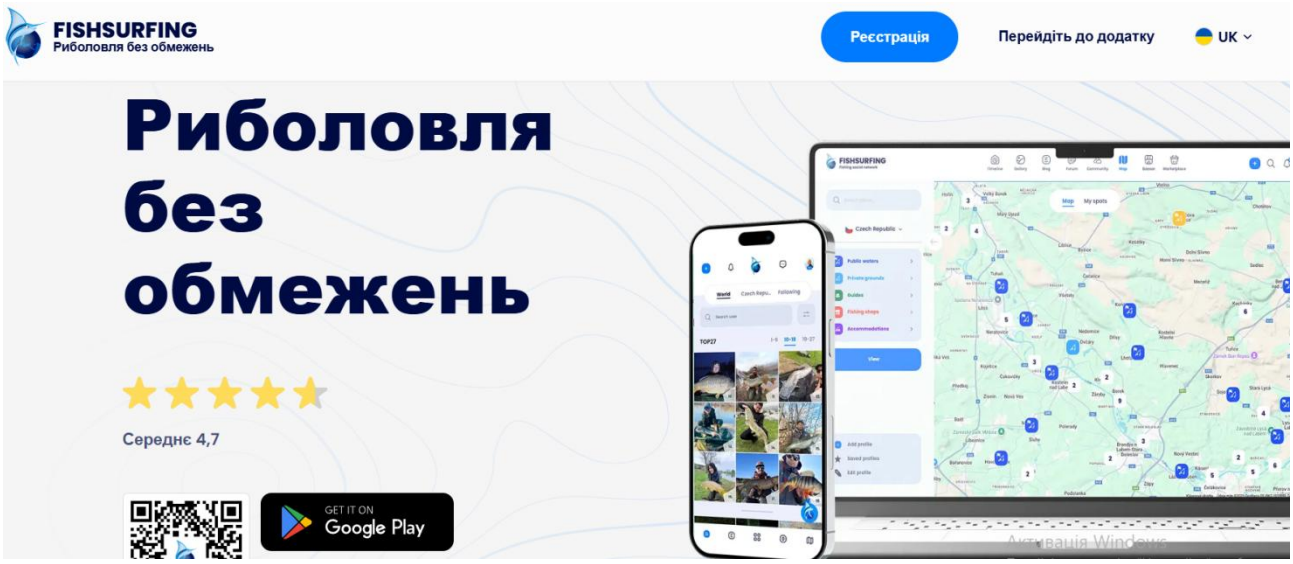


Рисунок 2.5. Платформа Fishsurfing

Основною перевагою даного ресурсу є інтеграція великої кількості функціональних можливостей у межах єдиної системи. Водночас велика кількість сервісів ускладнює інтерфейс та збільшує час освоєння системи новими користувачами. Частина функціоналу може бути надлишковою для користувачів, які зацікавлені лише в обміні інформацією та публікації матеріалів.

Проведений аналіз дозволяє зробити висновок, що більшість існуючих вебресурсів для любителів риболовлі можна умовно поділити на три групи: класичні форуми, інформаційні портали та соціальні платформи. Форумні системи характеризуються високим рівнем структурованості інформації, проте мають застарілі механізми взаємодії користувачів. Інформаційні портали забезпечують якісний контент, але зазвичай не підтримують активне створення матеріалів користувачами. Соціальні платформи пропонують широкий набір функціональних можливостей, однак часто характеризуються складністю інтерфейсу та надлишковістю окремих сервісів.

Отже, результати аналізу існуючих програмних рішень свідчать про доцільність розробки вебзастосунку, який поєднуватиме переваги сучасних соціальних платформ із простотою використання та зручністю керування

контентом. Така система повинна забезпечувати реєстрацію користувачів, створення та редагування публікацій, завантаження зображень, коментування матеріалів, централізоване зберігання інформації та ефективні засоби адміністрування. Саме зазначені вимоги покладено в основу розроблюваного вебзастосунку для любителів риболовлі.

2.2. Порівняльна характеристика сучасних підходів і технічних реалізацій

Розвиток вебтехнологій сприяв появі різноманітних підходів до створення інформаційних ресурсів для любителів риболовлі. Аналіз сучасних програмних рішень свідчить, що сьогодні можна виділити декілька основних напрямів реалізації таких систем. До них належать класичні форумні платформи, інформаційні портали, корпоративні вебсайти рибальських господарств і баз відпочинку, а також спеціалізовані соціальні мережі для рибалок. Кожен із зазначених підходів характеризується власними особливостями інтерфейсу користувача, архітектури програмного забезпечення та набору технологій, які використовуються під час розробки.

Найстарішим і досі поширеним підходом є створення тематичних форумів. До цієї категорії належать українські ресурси UAFisher та On-Fishing. Основною особливістю таких систем є ієрархічна структура представлення інформації, яка складається з категорій, розділів, тем та повідомлень користувачів. Візуальне оформлення форумів зазвичай базується на табличній структурі сторінок із мінімальним використанням мультимедійних елементів. Основна увага приділяється текстовому контенту, тоді як фотографії та відеоматеріали виконують допоміжну функцію.

З точки зору користувацького інтерфейсу форумні системи забезпечують

високу структурованість інформації та зручність тематичного пошуку. Проте їх дизайн часто не відповідає сучасним тенденціям веброзробки. Інтерфейс характеризується перевантаженістю текстовими елементами, невеликою кількістю візуальних компонентів та недостатньою адаптацією до мобільних пристроїв. Значна частина форумів була створена на основі платформ phpBB, vBulletin або їх аналогів, які розроблялися ще на початку поширення вебтехнологій другого покоління. У результаті користувачі отримують функціонально потужний, але візуально застарілий програмний продукт.

Іншим підходом є створення інформаційних порталів. Такі ресурси орієнтовані переважно на публікацію новин, аналітичних матеріалів, оглядів спорядження та рекомендацій для рибалок. Для них характерний сучасніший дизайн, використання адаптивної верстки та значна кількість графічного контенту. У більшості випадків такі системи реалізуються на базі популярних систем керування контентом, зокрема WordPress, Joomla або Drupal.

Важливою особливістю інформаційних порталів є використання сучасних принципів UX/UI-дизайну. На головній сторінці зазвичай розміщуються великі графічні банери, блоки популярних публікацій, стрічки новин та інтерактивні елементи навігації. Такий підхід покращує сприйняття інформації та забезпечує швидкий доступ до основного контенту. Водночас можливості взаємодії між користувачами залишаються обмеженими, оскільки основна модель роботи таких ресурсів передбачає створення контенту адміністраторами або редакторами.

Окрему категорію становлять вебсайти рибальських господарств, баз відпочинку та платних водойм. Для Полтавської області такі ресурси є особливо актуальними, оскільки регіон характеризується значною кількістю водойм та об'єктів рибальського туризму.

Серед регіональних ресурсів можна виділити сайт рибальської бази «Івашки», яка розташована в Полтавській області. Вебресурс містить інформацію про умови риболовлі, наявні види риби, інфраструктуру та послуги

для відвідувачів. Інтерфейс сайту побудований за принципом інформаційного каталогу та використовує фотографії водойм як основний засіб візуального представлення інформації. Основний акцент зроблено на рекламуванні послуг і залученні клієнтів, а не на формуванні онлайн-спільноти рибалок.

Подібний підхід реалізовано й на сайтах платних водойм та баз відпочинку Кременчуцького району. Аналіз таких ресурсів показує, що вони орієнтовані на представлення послуг, бронювання відпочинку та ознайомлення користувачів з умовами риболовлі. Дизайн подібних систем характеризується використанням великих фотографій, мінімалістичного інтерфейсу та адаптивної структури сторінок. Проте можливості створення користувацького контенту, ведення блогів, коментування матеріалів та соціальної взаємодії практично відсутні.

З точки зору технічної реалізації регіональні ресурси найчастіше побудовані на основі систем керування контентом WordPress. Такий підхід забезпечує швидке створення вебсайту та спрощує його супровід. Водночас використання готових шаблонів обмежує можливості індивідуалізації інтерфейсу та розширення функціоналу.

Найбільш сучасним напрямом розвитку рибальських вебресурсів є створення спеціалізованих соціальних платформ. До цієї категорії належать Fishbrain, FishFriender, Fishingstock та Fishsurfing. На відміну від форумів та інформаційних порталів, такі системи орієнтовані на активну участь користувачів у формуванні контенту [8].

Однією з ключових особливостей сучасних соціальних платформ є використання принципу Mobile First. Це означає, що інтерфейс спочатку проєктується для мобільних пристроїв, а вже потім адаптується до великих екранів. Такий підхід дозволяє забезпечити комфортне використання системи на смартфонах і планшетах, які сьогодні є основними засобами доступу до вебресурсів.

Fishbrain демонструє високий рівень опрацювання користувацького

інтерфейсу. У системі активно використовуються інтерактивні карти, динамічні стрічки контенту, персоналізовані рекомендації та інструменти аналітики. Значна увага приділяється візуалізації інформації. Кожен користувач може вести власний профіль, завантажувати фотографії уловів та взаємодіяти з іншими учасниками спільноти. Інтерфейс нагадує сучасні соціальні мережі та забезпечує високий рівень залучення користувачів.

Платформа FishFriender використовує подібний підхід, проте робить більший акцент на персональній статистиці та аналізі результатів риболовлі. Інтерфейс побудований на основі карток, графічних елементів та інтерактивних панелей. Завдяки цьому користувач отримує швидкий доступ до інформації про власні улови, погодні умови та історію активності.

Система Fishsurfing поєднує функціональність соціальної мережі, блогу, картографічного сервісу та інформаційного порталу. Інтерфейс побудований із використанням великої кількості інтерактивних компонентів. Водночас значна кількість функціональних можливостей ускладнює навігацію та підвищує поріг входження для нових користувачів.

Аналіз технічних реалізацій сучасних вебресурсів свідчить про поступову відмову від монолітних архітектур на користь багаторівневих систем. Для серверної частини все частіше використовуються ASP.NET Core, Node.js, Django, Laravel та Spring Boot. Такі технології забезпечують високу продуктивність, підтримку REST API та можливість масштабування програмного забезпечення.

Для реалізації клієнтської частини активно застосовуються JavaScript-фреймворки React, Angular та Vue.js. Їх використання дозволяє створювати інтерактивні користувацькі інтерфейси без необхідності повного перезавантаження сторінок. Проте для проєктів середнього рівня складності ефективним рішенням залишається використання серверного рендерингу на основі MVC-підходу [1-3, 11,16-18, 22].

Порівняння сучасних підходів показує, що форумні системи

забезпечують високий рівень структурованості інформації, але поступаються за якістю інтерфейсу. Інформаційні портали характеризуються сучасним дизайном, проте не підтримують активну взаємодію між користувачами. Регіональні сайти рибальських господарств Полтавської області орієнтовані на представлення послуг і не реалізують механізмів формування спільноти. Соціальні платформи забезпечують найкращі можливості взаємодії користувачів, однак характеризуються складною структурою та надлишковим функціоналом.

Таким чином, аналіз сучасних підходів і технічних реалізацій дозволяє зробити висновок про доцільність створення вебзастосунку, який поєднуватиме переваги тематичного блогу та соціальної платформи. Розроблювана система повинна забезпечувати сучасний адаптивний інтерфейс, підтримку мультимедійного контенту, можливість створення та коментування публікацій, а також простоту використання. Для реалізації таких вимог доцільним є застосування архітектури MVC та технологій ASP.NET Core MVC, Entity Framework Core, Microsoft SQL Server, HTML5, CSS3, JavaScript і Bootstrap, які забезпечують оптимальне співвідношення функціональності, продуктивності та зручності супроводу програмного забезпечення .

2.3. Переваги та недоліки існуючих програмних рішень

Аналіз сучасних вебресурсів для любителів риболовлі показує, що існуючі програмні рішення відрізняються не лише функціональними можливостями, але й технологічними підходами до їх реалізації. Вибір архітектури програмного забезпечення, засобів розробки, технологій побудови інтерфейсу користувача та інструментів роботи з даними безпосередньо впливає на продуктивність системи, її масштабованість, зручність використання та подальший супровід. У зв'язку з цим доцільно виконати аналіз переваг і недоліків існуючих програмних рішень саме з технічної точки зору.

Одним із найпоширеніших типів програмних рішень залишаються форумні системи. Більшість таких ресурсів реалізовано на основі готових платформ phpBB, MyBB, XenForo або vBulletin. Головною перевагою подібного підходу є швидкість розробки та відносно невисока вартість впровадження. Використання готової платформи дозволяє отримати функціональну систему без необхідності створення власної архітектури програмного забезпечення.

Важливою перевагою форумних систем є висока структурованість даних. Інформація зберігається у вигляді категорій, розділів, тем та повідомлень, що спрощує її пошук і систематизацію. Крім того, більшість форумних платформ підтримує механізми авторизації користувачів, розмежування прав доступу та адміністрування контенту.

Разом із тим форумні рішення мають низку технічних недоліків. Архітектура багатьох із них була сформована ще на початку розвитку вебтехнологій і не враховує сучасних вимог до побудови користувацького інтерфейсу. Більшість таких систем використовує серверну генерацію сторінок із повним перезавантаженням браузера після виконання кожної дії користувача. У результаті знижується швидкість взаємодії із системою та погіршується користувацький досвід.

Ще одним недоліком є складність адаптації форумних платформ до потреб конкретного проєкту. Незважаючи на наявність великої кількості модулів і розширень, внесення значних змін до функціональності часто потребує модифікації ядра системи. Це ускладнює подальше оновлення програмного забезпечення та підвищує ризик виникнення помилок.

Наступною категорією є вебресурси, побудовані на основі систем керування контентом. Найбільш поширеними інструментами цієї категорії є WordPress, Joomla та Drupal. Вони широко використовуються для створення інформаційних порталів, корпоративних сайтів та вебресурсів рибальських господарств.

Основною перевагою CMS-рішень є швидкість створення програмного

продукту. Завдяки використанню готових шаблонів дизайну та великої кількості плагінів можна реалізувати більшість типових функцій без написання значного обсягу програмного коду. Крім того, системи керування контентом мають розвинені засоби адміністрування та забезпечують можливість керування матеріалами без залучення розробників.

Важливою перевагою сучасних CMS є підтримка адаптивного дизайну. Більшість шаблонів побудована на основі Bootstrap або аналогічних CSS-фреймворків, що забезпечує коректне відображення сторінок на мобільних пристроях. Також значною перевагою є велика кількість готових модулів для роботи з мультимедійним контентом.

Недоліком такого підходу є залежність від сторонніх компонентів. Використання великої кількості плагінів може негативно впливати на продуктивність системи та створювати додаткові ризики безпеки. Крім того, значна частина функціональності реалізується за рахунок сторонніх модулів, які можуть втрачати підтримку або ставати несумісними з новими версіями платформи.

Ще одним недоліком CMS-рішень є складність реалізації специфічної бізнес-логіки. Якщо проєкт передбачає нестандартні сценарії роботи або складну структуру даних, використання готової системи керування контентом може призвести до надмірного ускладнення програмного коду та зниження продуктивності.

Найбільш сучасними є програмні рішення, побудовані за принципом соціальних платформ. До цієї категорії належать Fishbrain, FishFriender та Fishsurfing. Такі системи реалізуються із використанням сучасних технологій веброзробки та характеризуються високим рівнем інтерактивності.

Перевагою подібних платформ є використання сучасних архітектурних підходів. Найчастіше серверна частина будується на основі REST API або GraphQL API, а клієнтська реалізується як односторінковий застосунок. Такий підхід дозволяє суттєво підвищити швидкість взаємодії користувача із

системою та мінімізувати кількість повних перезавантажень сторінок.

Для реалізації клієнтської частини подібних систем використовуються сучасні JavaScript-фреймворки React, Angular або Vue.js. Їх застосування забезпечує створення динамічного інтерфейсу користувача, підтримку компонентного підходу до розробки та високу масштабованість програмного забезпечення.

Серверна частина найчастіше реалізується із використанням Node.js, ASP.NET Core, Spring Boot або Django. Дані технології забезпечують високу продуктивність, підтримку асинхронної обробки запитів та ефективну інтеграцію із сучасними системами зберігання даних [1-4, 11, 13, 16, 22].

Перевагою сучасних соціальних платформ є також активне використання хмарних технологій. Для зберігання фотографій, відео та інших мультимедійних матеріалів застосовуються сервіси Amazon S3, Google Cloud Storage та Azure Blob Storage. Це дозволяє забезпечити високу швидкодію та масштабованість системи навіть за значних навантажень.

Разом із тим подібні рішення мають і певні недоліки. Насамперед це висока складність розробки та супроводу. Побудова розподіленої системи із використанням окремого серверного API, клієнтського застосунку та хмарної інфраструктури потребує значних ресурсів і високої кваліфікації розробників.

Додатковою проблемою є складність тестування та налагодження таких систем. Велика кількість компонентів, взаємодія між різними сервісами та використання сторонніх API збільшують ймовірність виникнення помилок і ускладнюють процес супроводу програмного забезпечення.

Порівняльний аналіз існуючих рішень показує, що форумні системи забезпечують простоту реалізації та ефективну структуризацію інформації, проте характеризуються застарілими підходами до побудови інтерфейсу. Системи керування контентом дозволяють швидко створювати вебресурси, але мають обмеження щодо реалізації складної бізнес-логіки та часто залежать від сторонніх модулів. Сучасні соціальні платформи забезпечують найвищий

рівень функціональності та користувацького досвіду, однак потребують значних ресурсів для розробки та супроводу.

З огляду на результати проведеного аналізу можна зробити висновок, що для реалізації вебзастосунку для любителів риболовлі доцільним є використання підходу, який поєднує переваги класичних вебсайтів та сучасних соціальних платформ. Архітектура MVC, реалізована засобами ASP.NET Core MVC, дозволяє забезпечити чітке розділення функціональності системи, спрощує супровід програмного забезпечення та створює умови для його подальшого розвитку. Використання Entity Framework Core забезпечує ефективну взаємодію з базою даних, а застосування HTML5, CSS3, JavaScript і Bootstrap дозволяє створити сучасний адаптивний інтерфейс користувача. Такий підхід забезпечує оптимальний баланс між складністю реалізації, продуктивністю, масштабованістю та зручністю використання програмного продукту [1-4, 7, 11, 13, 16, 22].

2.4. Сучасні підходи до розробки вебресурсів для спільнот любителів активного відпочинку та риболовлі

Стрімкий розвиток інформаційних технологій та постійне зростання кількості користувачів мережі Інтернет суттєво вплинули на підходи до розробки вебресурсів, призначених для об'єднання людей за спільними інтересами. Особливо це стосується спільнот любителів активного відпочинку та риболовлі, діяльність яких значною мірою базується на обміні інформацією, досвідом, фотографіями, рекомендаціями та результатами власної діяльності. У зв'язку з цим сучасні вебресурси вже не обмежуються функціями простого інформування користувачів, а перетворюються на комплексні інтерактивні системи, що забезпечують ефективну взаємодію між учасниками спільноти.

Однією з основних тенденцій сучасної веброзробки є орієнтація на користувача та його потреби. Якщо раніше більшість тематичних ресурсів будувалися навколо інформаційного контенту, то сьогодні центральним елементом системи стає користувач і його взаємодія з іншими учасниками спільноти. Саме тому сучасні вебресурси активно використовують елементи соціальних мереж, що дозволяють створювати персональні профілі, публікувати власний контент, залишати коментарі, обмінюватися повідомленнями та формувати власні інформаційні стрічки.

Важливим напрямом розвитку вебресурсів є використання адаптивного дизайну. Сучасний користувач отримує доступ до інформаційних систем не лише за допомогою персональних комп'ютерів, але й через смартфони, планшети та інші мобільні пристрої. За статистичними даними, частка мобільного трафіку у багатьох вебпроєктах перевищує половину загальної кількості відвідувань. У зв'язку з цим під час розробки сучасних вебресурсів використовується концепція Responsive Web Design, яка забезпечує автоматичне пристосування інтерфейсу до різних розмірів екранів.

Для реалізації адаптивного інтерфейсу широко використовуються CSS-фреймворки Bootstrap, Tailwind CSS та Foundation. Вони забезпечують наявність готових компонентів інтерфейсу, адаптивних сіток та механізмів масштабування елементів сторінки. Використання таких інструментів дозволяє значно скоротити час розробки та забезпечити однакову якість відображення вебресурсу на різних пристроях.

Ще однією характерною особливістю сучасних вебресурсів є активне використання мультимедійного контенту. Для спільнот любителів активного відпочинку та риболовлі фотографії та відеоматеріали часто мають не менше значення, ніж текстова інформація. Саме тому сучасні платформи передбачають інтеграцію механізмів завантаження, зберігання та обробки графічних матеріалів.

Реалізація таких можливостей потребує використання спеціалізованих сервісів роботи з файлами та оптимізації мультимедійного контенту. У сучасних вебпроєктах широко використовуються хмарні сховища даних, зокрема Amazon S3, Azure Blob Storage та Google Cloud Storage. Їх застосування дозволяє зменшити навантаження на сервер застосунку та забезпечити високу швидкість доступу до мультимедійних матеріалів.

Важливим напрямом розвитку вебресурсів для рибальських спільнот є використання геоінформаційних технологій. Значна частина інформації, яка цікавить користувачів, безпосередньо пов'язана з конкретними водоймами, місцями риболовлі та туристичними маршрутами. Саме тому сучасні платформи все частіше інтегрують картографічні сервіси.

Для реалізації таких можливостей використовуються Google Maps API, OpenStreetMap та Leaflet. Завдяки цьому користувачі можуть переглядати місця риболовлі на інтерактивних картах, додавати власні позначки, формувати маршрути та отримувати додаткову інформацію про водойми. Використання геоінформаційних сервісів суттєво підвищує функціональність системи та покращує зручність її використання.

Окремим напрямом розвитку є застосування принципів соціальної взаємодії. У сучасних вебресурсах активно використовуються механізми коментування матеріалів, оцінювання публікацій, підписки на інших користувачів та формування персоналізованих стрічок новин. Подібні функції дозволяють підвищити активність користувачів та сприяють формуванню стабільної онлайн-спільноти.

З технічної точки зору реалізація таких можливостей потребує використання сучасних клієнтських і серверних технологій. Для створення інтерактивних інтерфейсів широко застосовуються JavaScript-фреймворки React, Angular та Vue.js. Використання компонентного підходу дозволяє реалізувати складний функціонал із мінімальними витратами ресурсів та забезпечити високу швидкодію вебзастосунку.

Для серверної частини сучасних вебресурсів використовуються різноманітні платформи та фреймворки. Найбільш поширеними є ASP.NET Core, Node.js, Django, Laravel та Spring Boot. Вибір конкретної технології залежить від вимог до продуктивності, масштабованості та складності бізнес-логіки.

Особливого поширення набуває використання архітектури MVC. Даний підхід передбачає розділення системи на три основні компоненти: модель, представлення та контролер. Таке розділення забезпечує зручність супроводу програмного забезпечення, спрощує тестування та дозволяє незалежно модифікувати окремі частини системи. Саме тому MVC залишається одним із найпопулярніших архітектурних підходів під час створення вебзастосунків різного рівня складності.

Для організації доступу до даних сучасні системи активно використовують ORM-технології. Серед найбільш поширених рішень можна виділити Entity Framework Core, Hibernate, SQLAlchemy та Sequelize. Використання ORM дозволяє працювати з базою даних на рівні об'єктної моделі предметної області, що спрощує розробку та підвищує надійність програмного коду.

Важливим аспектом сучасної веброзробки є забезпечення інформаційної безпеки. Зростання кількості вебзастосунків призводить до збільшення кількості кіберзагроз, тому сучасні програмні рішення повинні передбачати механізми захисту даних користувачів. Для цього використовуються технології автентифікації та авторизації, захист від міжсайтового виконання сценаріїв, шифрування паролів та використання захищених протоколів передачі даних.

Сучасні вебресурси також характеризуються активним використанням принципів модульності та повторного використання програмного коду. Значна частина функціоналу реалізується у вигляді окремих сервісів, бібліотек та компонентів, що спрощує подальший розвиток системи та дозволяє масштабувати її функціональні можливості без суттєвої зміни архітектури.

Ще одним важливим напрямом розвитку є застосування хмарних технологій та контейнеризації. Для розгортання сучасних вебресурсів широко використовуються Docker, Kubernetes та різноманітні хмарні платформи. Використання таких технологій дозволяє забезпечити високу доступність системи, автоматичне масштабування та спрощення процесів супроводу програмного забезпечення.

Проведений аналіз показує, що сучасні підходи до розробки вебресурсів для спільнот любителів активного відпочинку та риболовлі орієнтовані на створення інтерактивних інформаційних систем із високим рівнем залучення користувачів. Основними тенденціями є використання адаптивного дизайну, соціальних механізмів взаємодії, мультимедійного контенту, геоінформаційних сервісів, сучасних клієнтських і серверних технологій, а також архітектурних підходів, що забезпечують масштабованість і зручність супроводу програмного забезпечення.

З урахуванням результатів проведеного аналізу можна зробити висновок, що для розробки вебзастосунку для любителів риболовлі доцільно використовувати сучасні підходи до побудови вебресурсів, зокрема архітектуру MVC, адаптивний інтерфейс користувача, механізми роботи з мультимедійним контентом та централізоване зберігання даних. Застосування платформи ASP.NET Core MVC, технології Entity Framework Core, системи керування базами даних Microsoft SQL Server, а також засобів HTML5, CSS3, JavaScript і Bootstrap дозволяє реалізувати зазначені вимоги та забезпечити створення сучасного вебзастосунку для рибальської спільноти [1-4, 7, 11, 13, 16, 22-27].

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Характеристика використаних технологій і засобів розробки

Під час розробки вебзастосунку для любителів риболовлі було використано комплекс сучасних програмних технологій та інструментальних засобів, які забезпечують створення надійного, масштабованого та зручного у використанні програмного продукту. Вибір технологічного стеку здійснювався з урахуванням функціональних вимог до системи, необхідності забезпечення зручного адміністрування контенту, підтримки взаємодії користувачів та ефективної роботи з базою даних.

Основою програмного забезпечення виступає платформа ASP.NET Core MVC. Дана технологія є сучасним фреймворком для створення вебзастосунків, розробленим компанією Microsoft. ASP.NET Core є кросплатформним рішенням, що підтримує роботу в операційних системах Windows, Linux та macOS. Основними перевагами платформи є висока продуктивність, модульна архітектура, підтримка механізмів впровадження залежностей та можливість побудови масштабованих вебсистем різного рівня складності.

У процесі розробки використано архітектурний шаблон Model-View-Controller (MVC), який забезпечує розділення програмного забезпечення на три логічні компоненти. Модель відповідає за представлення даних та бізнес-логіки системи, представлення забезпечує формування користувацького інтерфейсу, а контролер виконує обробку запитів користувачів та координує взаємодію між іншими компонентами. Використання архітектури MVC сприяє підвищенню якості програмного коду, спрощує його супровід та забезпечує можливість незалежного розвитку окремих частин системи.

Для реалізації доступу до даних використано технологію Entity Framework Core. Вона являє собою ORM-фреймворк, який забезпечує об'єктно-

реляційне відображення між класами предметної області та таблицями бази даних. Застосування Entity Framework Core дозволяє виконувати операції створення, читання, оновлення та видалення даних без необхідності безпосереднього написання SQL-запитів для більшості операцій.

Однією з важливих переваг Entity Framework Core є підтримка механізму міграцій. Завдяки цьому структура бази даних може автоматично змінюватися відповідно до модифікацій моделей даних у програмному коді. Це спрощує процес супроводу програмного забезпечення та дозволяє підтримувати узгодженість між програмною моделлю та фізичною структурою бази даних.

Для формування запитів до бази даних використовується технологія LINQ (Language Integrated Query). Вона забезпечує можливість створення запитів до колекцій об'єктів та баз даних із використанням синтаксису мови програмування C#. Застосування LINQ дозволяє підвищити читабельність програмного коду, зменшити кількість помилок та забезпечити типобезпечність під час роботи з даними [1-4, 7, 11, 13, 16, 22].

Зберігання інформації у системі здійснюється за допомогою системи керування базами даних Microsoft SQL Server. Дана СКБД належить до класу реляційних систем керування базами даних та забезпечує надійне зберігання інформації, підтримку транзакцій, контроль цілісності даних і високий рівень продуктивності. Використання Microsoft SQL Server дозволяє ефективно працювати з великими обсягами інформації та забезпечує стабільне функціонування вебзастосунку [1-4, 7, 11, 13, 16, 22].

У структурі бази даних зберігається інформація про користувачів системи, публікації, коментарі та інші сутності предметної області. Взаємодія між таблицями реалізована за допомогою зовнішніх ключів, що забезпечує підтримку реляційних зв'язків та цілісність інформації.

Для побудови користувацького інтерфейсу використано Razor-представлення. Дана технологія дозволяє поєднувати HTML-розмітку та серверний код мовою C# у межах одного файлу. Завдяки цьому забезпечується

динамічне формування вебсторінок на основі даних, отриманих із серверної частини застосунку.

Перевагою Razor є простота інтеграції із архітектурою ASP.NET Core MVC та можливість створення гнучких інтерфейсів користувача. Крім того, використання серверного рендерингу дозволяє покращити швидкість початкового завантаження сторінок та позитивно впливає на індексацію вебресурсу пошуковими системами [1-4, 7, 11, 13, 16, 22].

Для створення структури вебсторінок використовується мова розмітки HTML5. Вона забезпечує формування логічної структури документа та підтримує сучасні елементи інтерфейсу користувача. HTML5 є базовою технологією для створення вебзастосунків та забезпечує сумісність із усіма сучасними браузерами.

Оформлення користувацького інтерфейсу реалізовано за допомогою каскадних таблиць стилів CSS3. Використання CSS3 дозволяє налаштовувати зовнішній вигляд елементів сторінок, керувати розташуванням компонентів інтерфейсу та реалізовувати адаптивний дизайн. Завдяки підтримці сучасних можливостей CSS3 забезпечується коректне відображення вебресурсу на різних пристроях незалежно від розміру екрана.

Для реалізації динамічної поведінки інтерфейсу використовується мова програмування JavaScript. Вона забезпечує можливість реагування на дії користувача без необхідності повного перезавантаження сторінки. JavaScript використовується для роботи з елементами інтерфейсу, перевірки коректності введених даних та реалізації інтерактивних компонентів вебзастосунку.

Важливу роль у розробці інтерфейсу відіграє фреймворк Bootstrap. Він містить набір готових компонентів інтерфейсу, адаптивну систему сіток та набір стилів, які дозволяють створювати сучасні вебсторінки із мінімальними витратами часу. Використання Bootstrap забезпечує єдиний стиль оформлення всіх сторінок застосунку та покращує зручність користування системою.

Для організації доступу до даних у проєкті використовується шаблон

Repository Pattern. Даний підхід передбачає створення окремого рівня доступу до даних, який приховує особливості роботи з базою даних від інших компонентів системи. Застосування шаблону Repository дозволяє зменшити зв'язність між компонентами програмного забезпечення та підвищити гнучкість архітектури.

У системі також використовується механізм ViewModel. Його призначення полягає у передачі даних між контролерами та представленнями. Використання ViewModel дозволяє формувати окремі моделі для інтерфейсу користувача та уникати безпосередньої передачі сутностей бази даних до рівня представлення.

Для роботи із графічними файлами та іншими мультимедійними матеріалами використовується спеціалізований сервіс FileService. Його основним завданням є завантаження, збереження та обробка файлів, які додаються користувачами під час створення публікацій. Застосування окремого сервісу для роботи з файлами забезпечує централізацію відповідного функціоналу та спрощує супровід програмного забезпечення.

Під час розробки програмного забезпечення використовувалося інтегроване середовище розробки Microsoft Visual Studio. Дане середовище забезпечує підтримку створення проєктів ASP.NET Core, засоби налагодження програмного коду, інтеграцію із системами контролю версій та інструменти автоматизованого керування залежностями проєкту. Використання Visual Studio дозволяє підвищити продуктивність розробки та забезпечує ефективний супровід програмного забезпечення на всіх етапах його життєвого циклу.

Таким чином, для реалізації вебзастосунку використано сучасний технологічний стек, основу якого становлять ASP.NET Core MVC, Entity Framework Core, LINQ, Microsoft SQL Server, Razor-представлення, HTML5, CSS3, JavaScript та Bootstrap. Поєднання зазначених технологій забезпечує створення надійного, масштабованого та зручного у використанні програмного продукту, який відповідає сучасним вимогам до веборієнтованих

інформаційних систем [1-4, 7, 11, 13, 16, 22].

3.2. Проєктування структури та основних модулів вебзастосунку для риболовлі

Проєктування структури програмного забезпечення є одним із ключових етапів створення вебзастосунку, оскільки саме на цьому етапі визначаються основні компоненти системи, їх взаємозв'язки, способи обробки інформації та принципи організації функціональних можливостей. Грамотно спроєктована структура програмного забезпечення забезпечує його масштабованість, спрощує супровід та дозволяє реалізувати необхідний функціонал із мінімальними витратами ресурсів.

Під час розробки вебзастосунку для любителів риболовлі було використано архітектурний шаблон Model-View-Controller, який реалізований засобами платформи ASP.NET Core MVC. Використання даного підходу дозволяє розділити програмне забезпечення на окремі логічні компоненти та забезпечити незалежний розвиток окремих частин системи.

Загальна архітектура програмного забезпечення побудована за багаторівневим принципом і включає рівень представлення, рівень бізнес-логіки та рівень зберігання даних. Користувач взаємодіє із системою через вебінтерфейс, сформований за допомогою Razor-представлень. Контролери здійснюють обробку запитів та координують взаємодію між інтерфейсом користувача, сервісами та репозиторіями. Доступ до даних реалізується за допомогою Entity Framework Core та системи керування базами даних Microsoft SQL Server. Загальну архітектуру розробленого вебзастосунку наведено на рисунку 3.1.

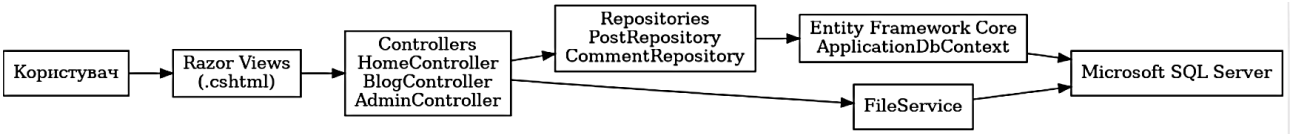


Рисунок 3.1. Загальна архітектура системи

Під час аналізу предметної області було встановлено, що основними користувачами системи є відвідувачі вебресурсу, зареєстровані користувачі та адміністратор. Відвідувачі мають можливість переглядати інформаційні матеріали, зареєстровані користувачі можуть створювати власні публікації та взаємодіяти з іншими учасниками спільноти, а адміністратор здійснює керування інформаційним наповненням та контроль роботи вебресурсу.

Для забезпечення необхідної функціональності систему було поділено на окремі програмні модулі. Такий підхід дозволяє ізолювати окремі функціональні можливості та забезпечує зручність супроводу програмного забезпечення. Основними модулями розробленого вебзастосунку є модуль керування користувачами, модуль публікацій, модуль коментарів, модуль роботи із зображеннями, адміністративний модуль та модуль доступу до даних.

Модуль керування користувачами забезпечує реєстрацію, автентифікацію та авторизацію користувачів. Його основним призначенням є контроль доступу до функціональних можливостей системи та зберігання інформації про користувачів. Даний модуль забезпечує формування персоналізованого середовища роботи та дозволяє розмежовувати права доступу залежно від ролі користувача.

Центральним елементом системи виступає модуль публікацій. Саме через нього реалізується основна функціональність вебресурсу, пов'язана зі створенням, редагуванням та переглядом матеріалів. Кожна публікація містить текстовий опис, інформацію про автора, дату створення та пов'язані графічні матеріали. Завдяки цьому користувачі можуть ділитися власним досвідом риболовлі, описувати особливості водойм та публікувати результати своїх поїздок.

Для забезпечення взаємодії між учасниками спільноти використовується модуль коментарів. Його функціональність дозволяє залишати відгуки до опублікованих матеріалів та організовувати обговорення між користувачами. Наявність такого модуля сприяє формуванню активної спільноти та підвищує рівень залученості користувачів до роботи із системою.

Окремим компонентом виступає модуль роботи із зображеннями. Його реалізація здійснюється за допомогою сервісу FileService, який забезпечує завантаження, збереження та відображення графічних матеріалів. Використання окремого сервісу дозволяє централізувати обробку файлів та спростити реалізацію інших компонентів системи.

Адміністративний модуль призначений для підтримки працездатності вебресурсу та керування його інформаційним наповненням. За допомогою даного модуля адміністратор може здійснювати модерацію контенту, редагувати публікації, контролювати коментарі та забезпечувати актуальність інформації, що зберігається у системі.

Для забезпечення взаємодії між програмними компонентами використовується модуль доступу до даних, реалізований із використанням шаблону Repository Pattern. Такий підхід дозволяє відокремити логіку роботи з базою даних від бізнес-логіки системи та підвищити гнучкість програмного забезпечення. Структуру основних модулів вебзастосунку наведено на рисунку 3.2.

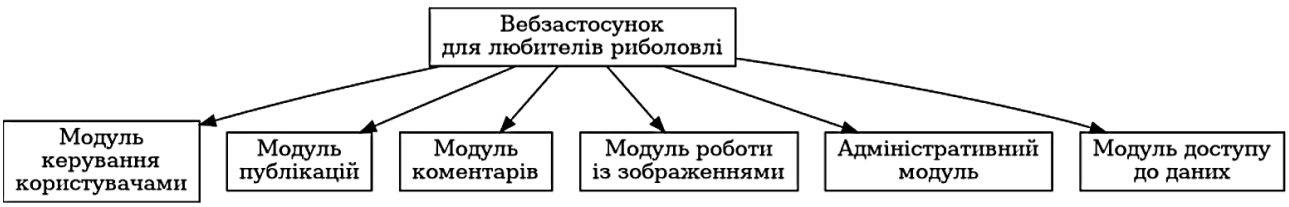


Рисунок 3.2. Структура основних модулів вебзастосунку

На етапі проєктування було також визначено основні сценарії взаємодії користувачів із системою. Користувач може переглядати публікації, проходити

процедуру реєстрації та авторизації, створювати власні матеріали, редагувати їх, завантажувати зображення та залишати коментарі. Адміністратор додатково отримує можливість керування контентом і виконання функцій модерації.

Для формалізації функціональних можливостей системи було побудовано UML-діаграму варіантів використання. Вона відображає основних учасників системи та їх взаємодію з функціональними модулями програмного забезпечення. Відповідну діаграму наведено на рисунку 3.3.

Основу програмної моделі системи становлять сутності предметної області. Центральною сутністю є публікація, яка використовується для зберігання інформації про матеріали, створені користувачами. Модель публікації містить ідентифікатор запису, заголовок, текстовий опис та інформацію про зображення.

Модель Post

```
public class Post
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public string Image { get; set; }
}
```

Використання окремої моделі для представлення публікацій дозволяє забезпечити структуроване зберігання інформації та підтримку взаємозв'язків із іншими сутностями системи. На основі цієї моделі реалізуються операції створення, редагування, перегляду та видалення записів.

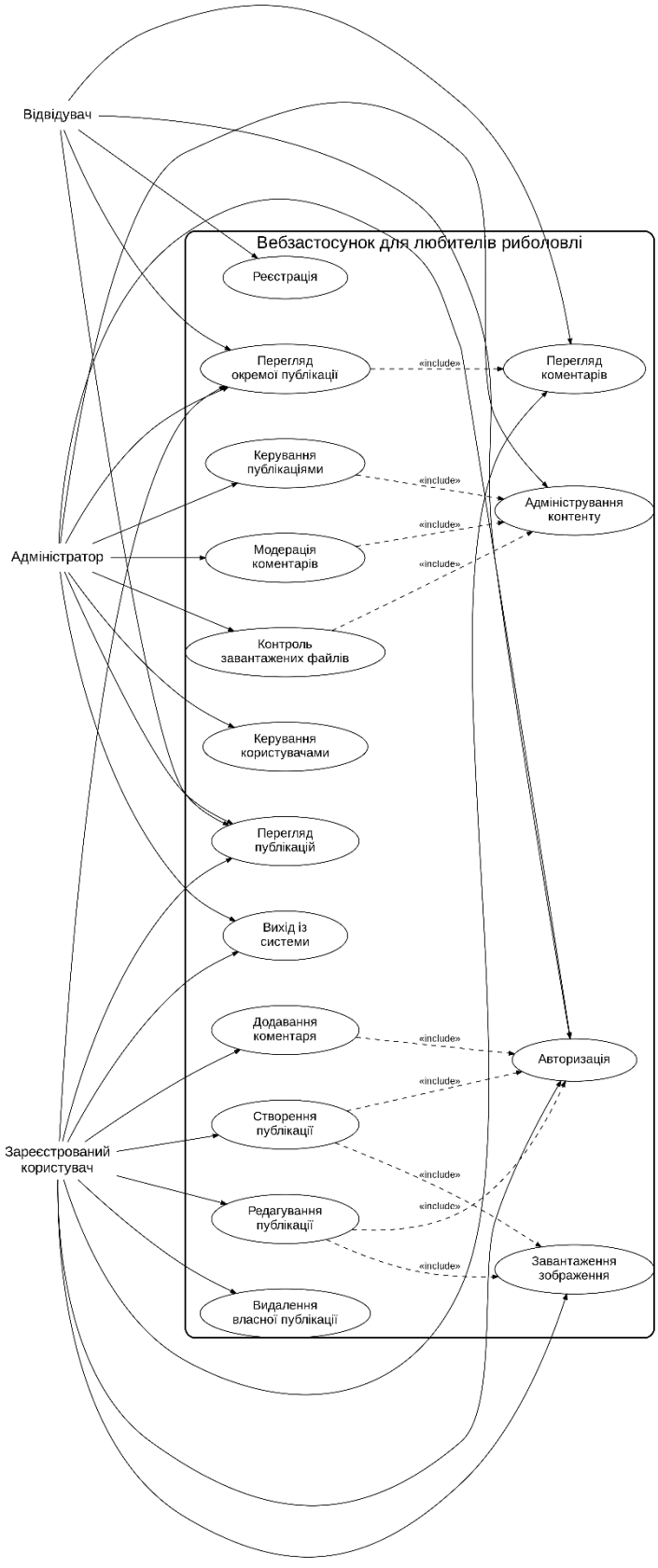


Рисунок 3.3. UML Use Case діаграма

Для взаємодії між рівнем представлення та рівнем даних використовуються контролери та репозиторії. Контролери здійснюють обробку HTTP-запитів користувачів, а репозиторії забезпечують доступ до інформації, що зберігається у базі даних. Такий підхід дозволяє реалізувати чітке розмежування відповідальності між компонентами програмного забезпечення та відповідає принципам архітектури MVC.

Під час проєктування інтерфейсу користувача було враховано необхідність забезпечення адаптивності вебресурсу. Для цього використано можливості Bootstrap та сучасні засоби HTML5 і CSS3. У результаті інтерфейс коректно відображається як на персональних комп'ютерах, так і на мобільних пристроях, що є важливим фактором для сучасних вебзастосунків.

Таким чином, у процесі проєктування було сформовано структуру вебзастосунку, яка забезпечує реалізацію всіх необхідних функціональних можливостей для роботи тематичної спільноти любителів риболовлі. Запропонована архітектура забезпечує ефективну організацію програмних компонентів, підтримує розширення функціональності системи та створює основу для подальшої реалізації програмного забезпечення.

3.3. Опис архітектури системи та взаємодії її компонентів

Архітектура програмного забезпечення визначає принципи організації його компонентів, способи обміну даними між ними та механізми реалізації функціональних можливостей системи. Для вебзастосунку для любителів риболовлі було використано багаторівневу архітектуру, побудовану на основі шаблону Model-View-Controller. Такий підхід дозволяє забезпечити розділення відповідальності між окремими компонентами системи, підвищити якість програмного коду та спростити процес подальшого супроводу програмного

забезпечення.

У розробленій системі можна виділити три основні рівні. Перший рівень представлений інтерфейсом користувача, який реалізований за допомогою Razor Views, HTML5, CSS3, JavaScript та Bootstrap. Другий рівень утворюють контролери та бізнес-логіка застосунку, які відповідають за обробку запитів користувачів та виконання необхідних операцій. Третій рівень представлений компонентами доступу до даних, що взаємодіють із базою даних Microsoft SQL Server через Entity Framework Core та репозиторії. Загальну структуру взаємодії компонентів системи подано у вигляді UML Component Diagram.

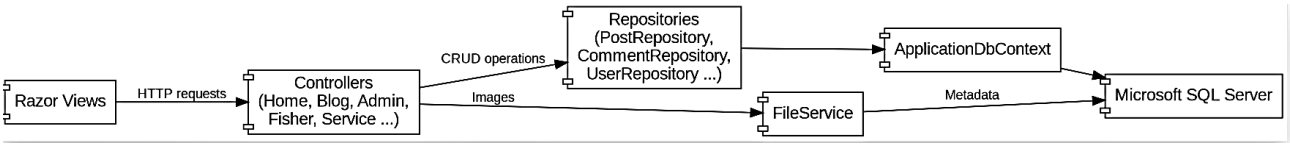


Рисунок 3.4. UML Component Diagram вебзастосунку

Основу серверної частини становить набір контролерів, кожен із яких відповідає за реалізацію окремої функціональної підсистеми. Аналіз структури проєкту показує наявність контролерів HomeController, BlogController, AdminController, FisherController, ProjectController, ServiceController, SubscribeController та AboutController. Такий підхід забезпечує логічне розділення функціональності між окремими компонентами та спрощує підтримку програмного забезпечення.

Особливе значення для функціонування вебресурсу має контролер BlogController, який реалізує механізми перегляду публікацій, фільтрації матеріалів за категоріями та тегами, а також додавання коментарів до публікацій. У конструкторі контролера використовується впровадження залежностей для отримання екземплярів репозиторіїв PostRepository та TagRepository, що відповідає сучасним принципам побудови ASP.NET Core-застосунків.

Фрагмент контролера BlogController

```
public class BlogController : Controller
{
   private PostRepository _postReposytory;
   private TagRepository _tagRepository;
   public BlogController(PostRepository postRepository, TagRepository tagRepository)
   {
      _postReposytory = postRepository;
      _tagRepository = tagRepository;
   }
}
```

Після отримання HTTP-запиту контролер виконує необхідні операції над даними за допомогою відповідних репозиторіїв та передає результати до представлення. Наприклад, під час перегляду списку публікацій система виконує запит до репозиторію, здійснює сортування матеріалів за датою публікації та формує модель даних для відображення у вебінтерфейсі. Така схема роботи забезпечує незалежність представлення від механізмів зберігання даних і відповідає принципам архітектури MVC.

Для організації доступу до інформації використовується шаблон Repository Pattern. У структурі проєкту реалізовано декілька спеціалізованих репозиторіїв, серед яких PostRepository, CommentRepository, UserRepository, TagRepository, NavigationRepository, OptionsRepository та інші. Кожен репозиторій відповідає за роботу із певною групою сутностей предметної області.

Використання репозиторіїв дозволяє ізолювати логіку роботи з базою даних від інших частин програмного забезпечення. У результаті контролери взаємодіють не безпосередньо з Entity Framework Core, а через відповідний рівень доступу до даних. Це забезпечує зниження зв'язності між компонентами системи та підвищує можливість подальшої модифікації програмного

забезпечення.

Фрагмент реалізації UserRepository

```
public class UserRepository
{
   private ApplicationDbContext _dbContext;
   public UserRepository(ApplicationDbContext clinicDbContext)
   {
      _dbContext = clinicDbContext;
   }
   public User? GetUserByEmail (string email)
   {
      return _dbContext.Users.FirstOrDefault(u => u.Email.ToLower() == email.ToLower());
   }
   public User? GetUserByLogin(string login)
   {
      return _dbContext.Users.FirstOrDefault(u => u.Login == login);
   }
}
```

З наведеного фрагмента видно, що взаємодія з таблицями бази даних здійснюється через контекст ApplicationDbContext. Для формування запитів використовується технологія LINQ, яка забезпечує об'єктно-орієнтований підхід до роботи з даними та підвищує читабельність програмного коду. Використання LINQ також дозволяє зменшити кількість потенційних помилок під час формування SQL-запитів.

Центральним елементом підсистеми доступу до даних виступає ApplicationDbContext, який забезпечує взаємодію між моделями предметної області та таблицями бази даних. Через цей компонент здійснюється виконання операцій створення, читання, оновлення та видалення інформації. Саме ApplicationDbContext використовується всіма репозиторіями для доступу до даних, що забезпечує єдину точку взаємодії із системою керування базами даних.

Важливим компонентом архітектури є механізм роботи із графічними файлами. Оскільки користувачі можуть додавати до публікацій фотографії улову, водойм або туристичних маршрутів, у системі реалізовано окремий сервіс для роботи із файлами. Його функціональність забезпечує збереження та видалення зображень на сервері.

Реалізація FileService

```
public class FileService
{
    public static void SaveImage(IFormFile file, string fullFilePath)
    {
        string? directory = Path.GetDirectoryName(fullFilePath);
        if (directory == null)
            throw new Exception("Invalid path");
        Directory.CreateDirectory(directory);
        using (var stream = new FileStream(fullFilePath, FileMode.Create))
        {
            file.CopyTo(stream);
        }
    }
}
```

Під час створення або редагування публікації контролер викликає метод SaveImage, який виконує перевірку каталогу збереження файлів та записує отримане зображення на сервер. Такий підхід дозволяє централізувати логіку роботи із файлами та уникнути дублювання програмного коду в інших компонентах системи.

Для забезпечення безпечного завантаження зображень використовується додатковий механізм перевірки файлів. У проєкті реалізовано клас ImageValidator, який контролює допустимі розширення файлів, перевіряє MIME-типи та обмежує максимальний розмір завантажуваних зображень. Це дозволяє підвищити рівень безпеки вебресурсу та зменшити ризик завантаження небезпечних файлів.

Особливу роль у взаємодії компонентів відіграє механізм маршрутизації запитів. Після надходження HTTP-запиту від браузера користувача система визначає відповідний контролер та метод обробки, виконує необхідні операції над даними, після чого формує відповідне представлення та повертає результат клієнту. Даний процес є типовим для архітектури MVC та забезпечує ефективну організацію роботи вебзастосунку.

Послідовність взаємодії компонентів під час перегляду окремої публікації представлена у вигляді UML Sequence Diagram.

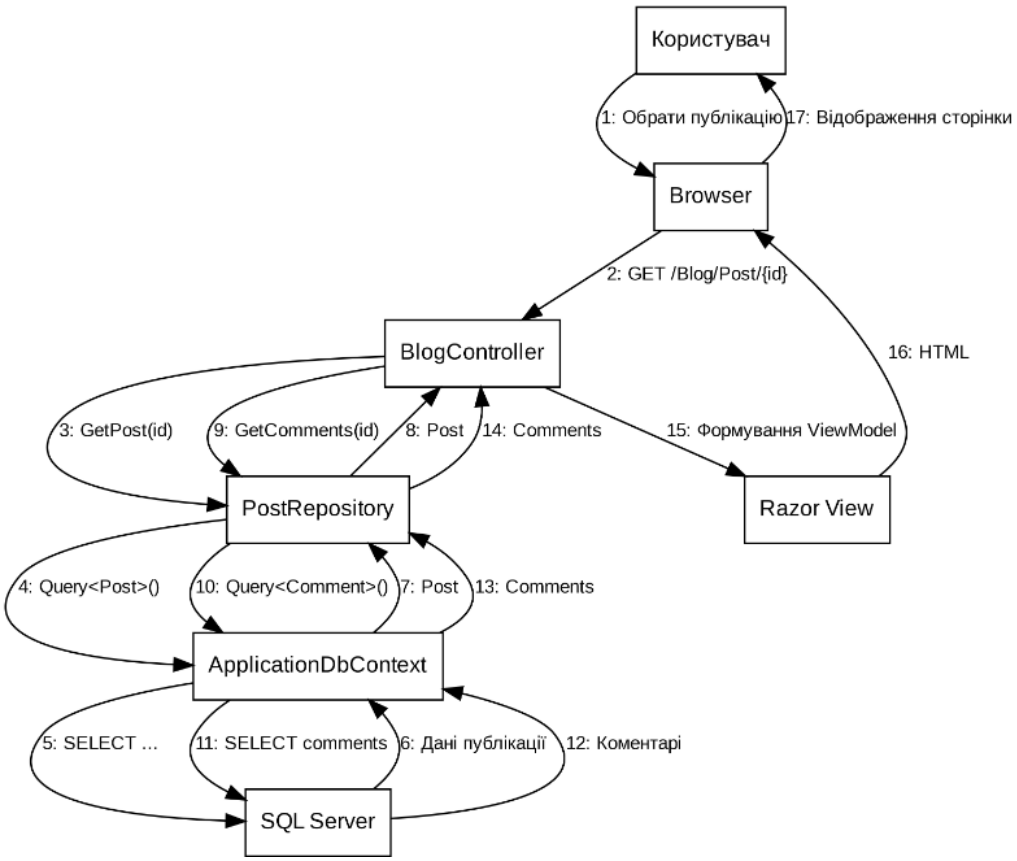


Рисунок 3.5. Діаграма взаємодії компонентів під час перегляду публікації

На рисунку 3.5 наведено послідовність взаємодії компонентів системи під час перегляду користувачем окремої публікації. Після вибору матеріалу браузер формує HTTP-запит до контролера BlogController, який через репозиторій PostRepository отримує необхідні дані з бази даних. Для виконання операцій доступу до даних використовується ApplicationDbContext та технологія Entity

Framework Core. Після отримання інформації про публікацію та пов'язані коментарі контролер формує модель представлення і передає її до Razor View для генерації HTML-сторінки. Така схема взаємодії забезпечує розділення функціональності між компонентами системи та відповідає принципам архітектури MVC.

Під час запуску застосунку виконується конфігурація всіх необхідних компонентів через контейнер залежностей ASP.NET Core. У файлі Program.cs здійснюється реєстрація контексту бази даних, налаштування автентифікації та підключення сервісів застосунку. Для взаємодії із базою даних використовується Entity Framework Core та провайдер SQL Server.

Значну роль у побудові інтерфейсу користувача відіграють View Components. У структурі проєкту реалізовано окремі компоненти для формування заголовка сайту, відображення випадкових проєктів, сервісів та інших інформаційних блоків. Такий підхід дозволяє повторно використовувати елементи інтерфейсу та зменшує дублювання програмного коду.

Для відображення складних сторінок використовуються ViewModels, які забезпечують передачу структурованих даних між контролерами та представленнями. Використання ViewModels дозволяє уникнути безпосередньої передачі сутностей бази даних до рівня інтерфейсу користувача та сприяє кращій організації програмного коду.

Таким чином, архітектура розробленого вебзастосунку базується на використанні шаблону Model-View-Controller, технології Entity Framework Core та шаблону Repository Pattern. Взаємодія компонентів системи реалізована за багаторівневим принципом, що забезпечує розділення функціональності між окремими модулями, підвищує масштабованість програмного забезпечення та спрощує його подальший супровід. Використання спеціалізованих сервісів, репозиторіїв, View Components та механізмів валідації дозволяє забезпечити стабільну роботу вебресурсу та реалізувати всі необхідні функціональні

можливості для підтримки діяльності спільноти любителів риболовлі.

3.4. Проєктування бази даних та моделі зберігання інформації

Одним із ключових етапів розробки вебзастосунку для любителів риболовлі є проєктування бази даних, яка забезпечує централізоване зберігання інформації, підтримку взаємозв’язків між сутностями предметної області та ефективний доступ до даних. Якість проєктування структури бази даних безпосередньо впливає на продуктивність програмного забезпечення, цілісність інформації, масштабованість системи та зручність її подальшого супроводу.

Для зберігання даних у розробленому вебзастосунку використовується реляційна система керування базами даних Microsoft SQL Server. Вибір даної СКБД обумовлений її високою продуктивністю, підтримкою складних зв’язків між таблицями, механізмами забезпечення цілісності даних та повною інтеграцією з платформою ASP.NET Core і технологією Entity Framework Core.

Структура бази даних формувалася відповідно до функціональних вимог системи та особливостей предметної області. Основне призначення бази даних полягає у зберіганні інформації про користувачів вебресурсу, публікації, коментарі, категорії, теги, навігаційні елементи, контактні повідомлення та інші об’єкти, необхідні для забезпечення роботи інформаційної системи.

При проєктуванні структури даних використовувався принцип нормалізації, що дозволяє уникнути дублювання інформації та забезпечити логічну організацію даних. Кожна сутність предметної області представлена окремою таблицею, а взаємозв’язки між ними реалізовані за допомогою первинних та зовнішніх ключів.

Центральне місце у структурі бази даних займає сутність користувача. Вона використовується для зберігання інформації про зареєстрованих учасників спільноти. Для кожного користувача зберігаються облікові дані, контактна інформація та інші атрибути, необхідні для реалізації механізмів автентифікації та авторизації. Наявність окремої таблиці користувачів забезпечує можливість персоналізації контенту та організації взаємодії між учасниками вебресурсу.

Однією з основних сутностей предметної області є публікація. Саме вона використовується для зберігання матеріалів, які створюються користувачами або адміністраторами системи. Кожна публікація містить заголовок, текстовий опис, дату створення, інформацію про автора та інші характеристики. Зберігання матеріалів у вигляді окремої сутності забезпечує ефективну організацію інформаційного контенту та спрощує його подальший пошук і відображення.

Для реалізації функцій обговорення матеріалів використовується сутність коментаря. Кожен коментар пов’язаний із конкретною публікацією та користувачем, який його створив. Такий підхід дозволяє організувати ієрархічну структуру даних та забезпечити коректне відображення коментарів під відповідними матеріалами. Зв’язок між публікаціями та коментарями реалізується за принципом «один до багатьох», оскільки одна публікація може містити необмежену кількість коментарів.

Для класифікації матеріалів у системі використовуються категорії. Наявність окремої таблиці категорій дозволяє групувати публікації за тематичними напрямами, що спрощує навігацію по вебресурсу та підвищує зручність пошуку інформації. Кожна категорія може бути пов’язана з багатьма публікаціями, тоді як кожна публікація належить до певної категорії.

Додатковим механізмом класифікації контенту є система тегів. На відміну від категорій, теги дозволяють більш гнучко описувати зміст матеріалів та забезпечують можливість тематичного пошуку. Оскільки одна публікація може містити декілька тегів, а один тег може використовуватися для багатьох

публікацій, між відповідними сутностями реалізується зв’язок типу «багато до багатьох».

У структурі бази даних також передбачено таблиці для зберігання навігаційних елементів вебресурсу. Вони використовуються для формування меню, структури сторінок та інших елементів інтерфейсу користувача. Завдяки цьому адміністратор отримує можливість централізованого керування структурою вебсайту без внесення змін до програмного коду.

Для забезпечення зворотного зв’язку із користувачами використовується сутність контактного повідомлення. Вона дозволяє зберігати інформацію, отриману через форми звернення на сайті. Такі дані можуть використовуватися адміністрацією ресурсу для обробки запитів користувачів та підтримки комунікації із відвідувачами вебресурсу.

Важливою особливістю розробленої моделі даних є використання об’єктно-реляційного відображення, яке реалізується засобами Entity Framework Core. У даному випадку кожній таблиці бази даних відповідає окремий клас моделі, а взаємозв’язки між таблицями описуються за допомогою навігаційних властивостей. Такий підхід дозволяє працювати з даними на рівні об’єктів мови програмування C# та суттєво спрощує реалізацію операцій доступу до інформації.

Для централізованого керування всіма сутностями використовується контекст бази даних ApplicationDbContext. Даний компонент виконує функції посередника між прикладною логікою та фізичною базою даних. Через нього здійснюється створення, читання, оновлення та видалення інформації. Крім того, контекст забезпечує контроль змін об’єктів та автоматичне формування SQL-запитів відповідно до операцій, які виконуються у програмному коді.

Під час проєктування моделі даних особлива увага приділялася забезпеченню цілісності інформації. Для цього використовуються первинні ключі, зовнішні ключі та механізми обмежень, що підтримуються системою керування базами даних. Наявність таких механізмів дозволяє запобігти появі

неузгоджених даних та забезпечує коректність зв’язків між сутностями.

Для підвищення продуктивності роботи системи використовуються індекси, які прискорюють виконання операцій пошуку та вибірки інформації. Особливо важливими є індекси для таблиць користувачів, публікацій та коментарів, оскільки саме вони найчастіше використовуються під час роботи вебзастосунку.

Суттєвою перевагою обраної моделі зберігання інформації є її масштабованість. Структура бази даних дозволяє без значних змін додавати нові сутності та функціональні можливості. Наприклад, у майбутньому до системи можуть бути інтегровані модулі рейтингу користувачів, оцінювання публікацій, бронювання місць риболовлі або ведення особистих журналів рибалок.

Центральною сутністю моделі є таблиця Post, яка призначена для зберігання інформації про публікації, що розміщуються на вебресурсі. Для кожної публікації зберігаються заголовок, текстовий опис, зображення, дата створення та посилання на автора і категорію. Саме ця сутність є основою інформаційного наповнення вебзастосунку.

Таблиця User містить інформацію про зареєстрованих користувачів системи. Між сутностями User та Post реалізовано зв’язок типу «один до багатьох», оскільки один користувач може створювати багато публікацій, тоді як кожна публікація належить лише одному автору. Аналогічний зв’язок існує між таблицями User та Comment, що дозволяє зберігати інформацію про авторів коментарів.

Для реалізації можливості обговорення матеріалів використовується таблиця Comment. Кожен коментар пов’язаний із конкретною публікацією та конкретним користувачем. Між сутностями Post і Comment реалізовано зв’язок типу «один до багатьох», що дозволяє одній публікації містити довільну кількість коментарів.

Класифікація матеріалів здійснюється за допомогою таблиці Category.

Кожна категорія може містити багато публікацій, тоді як кожна публікація належить лише одній категорії. Використання категорій дозволяє структурувати контент вебресурсу та спрощує навігацію між тематичними розділами.

Для більш гнучкого опису змісту матеріалів використовується система тегів. Таблиця Tag містить перелік тегів, які можуть бути прив’язані до публікацій. Оскільки одна публікація може мати декілька тегів, а один тег може використовуватися для багатьох публікацій, між сутностями Post і Tag реалізовано зв’язок типу «багато до багатьох». Для його реалізації використовується проміжна таблиця PostTag, яка містить зовнішні ключі на відповідні записи публікацій і тегів.

У структурі бази даних також передбачені допоміжні сутності Navigation та ContactRequest. Таблиця Navigation використовується для зберігання елементів навігаційного меню вебресурсу та забезпечує можливість централізованого керування структурою сайту. Таблиця ContactRequest призначена для зберігання повідомлень, отриманих через форму зворотного зв’язку, що дозволяє організувати комунікацію між користувачами та адміністрацією вебресурсу.

Представлена структура бази даних забезпечує цілісність інформації, підтримує необхідні взаємозв’язки між сутностями предметної області та створює основу для ефективної роботи вебзастосунку. Використання первинних і зовнішніх ключів дозволяє гарантувати коректність зв’язків між об’єктами системи та забезпечує надійне зберігання даних.

Таким чином, у процесі розробки вебзастосунку було спроєктовано реляційну базу даних, яка забезпечує централізоване зберігання інформації, підтримку взаємозв’язків між сутностями предметної області та ефективне виконання операцій доступу до даних. Використання Microsoft SQL Server та Entity Framework Core дозволило реалізувати надійну модель зберігання інформації, яка відповідає функціональним вимогам системи та забезпечує

можливість її подальшого розвитку.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Розробка структури даних та основних сутностей вебзастосунку

Розробка структури даних є одним із ключових етапів створення вебзастосунку, оскільки саме вона визначає принципи зберігання, обробки та взаємодії інформації в системі. Для вебзастосунку, призначеного для об'єднання любителів риболовлі, була сформована структура даних, яка забезпечує зберігання інформації про користувачів, публікації, коментарі, категорії, теги, елементи навігації та звернення користувачів. Розроблена модель даних орієнтована на підтримку інформаційного наповнення ресурсу, організацію взаємодії між учасниками спільноти та забезпечення можливості подальшого розширення функціональності програмного забезпечення.

У процесі реалізації проєкту використано реляційну базу даних Microsoft SQL Server та технологію Entity Framework Core, яка забезпечує механізм об'єктно-реляційного відображення. Завдяки цьому кожна сутність предметної області представлена окремим класом мови програмування C#, а її екземпляри автоматично відображаються у відповідні таблиці бази даних.

Усі основні сутності системи реєструються в контексті бази даних ApplicationDbContext. Для кожного типу даних створено окремий набір DbSet, який забезпечує взаємодію між програмним кодом та таблицями бази даних.

Фрагмент коду бази даних ApplicationDbContext

```
public DbSet<Category> Categories { get; set; }
public DbSet<Comment> Comments { get; set; }
public DbSet<Post> Posts { get; set; }
public DbSet<Tag> Tags { get; set; }
public DbSet<PostTags> PostTags { get; set; }
public DbSet<ContactRequest> ContactRequests { get; set; }
public DbSet<User> Users { get; set; }
```

Наведений фрагмент демонструє перелік основних сутностей, що

використовуються у вебзастосунку. Кожна з них відповідає окремій таблиці бази даних та містить інформацію, необхідну для реалізації відповідного функціонального модуля системи.

Центральне місце в структурі даних займає сутність Post, яка використовується для зберігання публікацій користувачів. Саме публікації формують основний контент вебресурсу та забезпечують можливість обміну досвідом між рибалками. Для кожної публікації зберігається заголовок, короткий опис, основний текст, дата створення, зображення, інформація про автора та категорію.

Кожна публікація пов'язана з конкретним користувачем, який є її автором. Для реалізації такого зв'язку використовується зовнішній ключ UserId та відповідна навігаційна властивість. Налаштування взаємозв'язку між сутностями виконується в методі OnModelCreating контексту бази даних.

Реалізація зв'язку між сутностями Post та User

```
modelBuilder.Entity<Post>()
   .HasOne(p => p.User)
   .WithMany(u => u.Posts)
   .HasForeignKey(p => p.UserId);
```

Представлений фрагмент програмного коду реалізує зв'язок типу «один до багатьох», відповідно до якого один користувач може створювати багато публікацій, тоді як кожна публікація належить лише одному автору.

Для представлення користувачів системи використовується сутність User. Вона містить інформацію про логін, адресу електронної пошти, хеш пароля та інші атрибути, необхідні для роботи механізмів реєстрації, авторизації та керування обліковими записами. Використання окремої моделі користувача забезпечує можливість персоналізації роботи вебресурсу та підтримку механізмів безпеки.

Значну роль у функціонуванні вебзастосунку відіграє сутність Comment, яка забезпечує можливість обговорення публікацій. Кожен коментар містить текст повідомлення, дату створення та посилання на користувача й публікацію.

Така структура дозволяє організувати взаємодію між учасниками спільноти та підтримувати активне обговорення тематичних матеріалів.

Для структуризації інформаційного контенту використовуються категорії. Сутність Category забезпечує групування матеріалів за тематичними напрямами, такими як риболовля на коропа, спінінгова риболовля, огляди спорядження, звіти про рибальські поїздки та інші теми. Наявність категорій суттєво покращує навігацію вебресурсом та спрощує пошук необхідної інформації.

Додатковим інструментом класифікації контенту є система тегів. На відміну від категорій, теги дозволяють більш детально описувати зміст матеріалів та використовуються для реалізації тематичного пошуку. Оскільки одна публікація може містити декілька тегів, а один тег може використовуватися в багатьох публікаціях, між цими сутностями реалізовано зв'язок типу «багато до багатьох».

Для реалізації такого зв'язку використовується проміжна таблиця PostTags.

Реалізація складеного ключа таблиці PostTags

```
modelBuilder.Entity<PostTags>()
   .HasKey(pt => new { pt.PostId, pt.TagId });
```

Використання складеного первинного ключа забезпечує унікальність кожної пари «публікація–тег» та запобігає дублюванню записів у таблиці зв'язків.

.

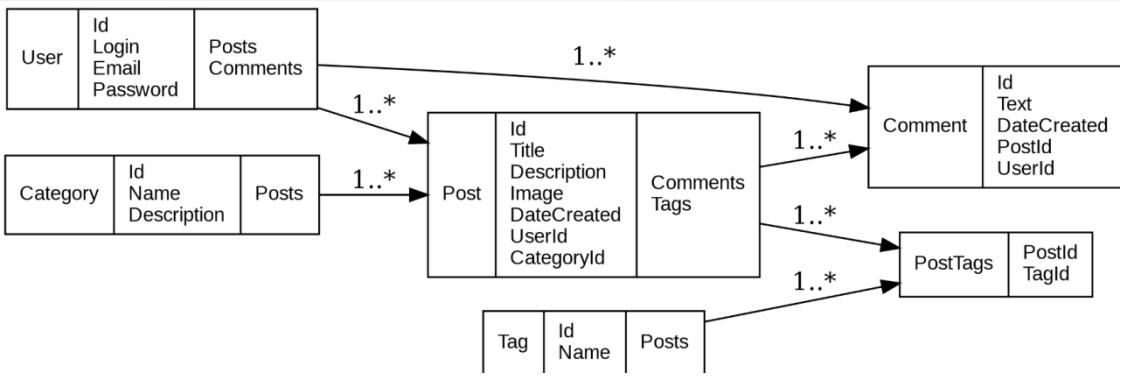


Рисунок 4.1. UML-діаграма основних сутностей вебзастосунку

На рисунку 4.1 відображено основні класи предметної області та взаємозв'язки між ними. Центральною сутністю виступає клас Post, який взаємодіє з класами User, Category, Comment та Tag. Така структура забезпечує логічну організацію даних та відповідає функціональним вимогам розробленого програмного забезпечення.

Окрім основних сутностей, у структурі системи використовуються допоміжні об'єкти. Зокрема, сутність Navigate забезпечує формування навігаційного меню вебресурсу, ContactRequest використовується для збереження повідомлень із форми зворотного зв'язку, Subscribe реалізує механізм підписки користувачів на новини ресурсу, а Option забезпечує централізоване зберігання параметрів конфігурації вебзастосунку.

Для роботи із сутностями використовується репозиторний підхід. Кожна група даних обслуговується окремим репозиторієм, який інкапсулює логіку доступу до бази даних та забезпечує виконання операцій створення, читання, оновлення і видалення інформації.

Приклад реалізації доступу до даних користувачів наведено нижче.

```
public User? GetUserByEmail (string email)
{
   return _dbContext.Users.FirstOrDefault(u => u.Email.ToLower() == email.ToLower());
}
public User? GetUserByLogin(string login)
{
   return _dbContext.Users.FirstOrDefault(u => u.Login == login);
}
```

Представлений фрагмент демонструє реалізацію пошуку користувачів за адресою електронної пошти та логіном із використанням Entity Framework Core. Застосування репозиторіїв дозволяє відокремити логіку роботи з даними від бізнес-логіки застосунку та підвищує підтримуваність програмного забезпечення.

Таким чином, у процесі розробки вебзастосунку була сформована структура даних, яка охоплює всі основні об'єкти предметної області та забезпечує їх ефективне зберігання і взаємодію. Використання Entity Framework Core, механізму об'єктно-реляційного відображення та репозиторного підходу дозволило реалізувати гнучку й масштабовану модель даних, що відповідає вимогам сучасних веборієнтованих інформаційних систем.

4.2. Реалізація серверної частини інформаційної системи

Клієнтська частина вебзастосунку є одним із найважливіших компонентів програмного забезпечення, оскільки саме вона забезпечує взаємодію користувачів із функціональними можливостями системи. Від якості реалізації інтерфейсу залежить зручність використання вебресурсу, швидкість отримання інформації та ефективність роботи користувачів із контентом. Під час розробки вебзастосунку для любителів риболовлі особлива увага приділялася створенню сучасного інтерфейсу користувача, який забезпечує зручну навігацію, адаптивне відображення сторінок та підтримку динамічної взаємодії з даними.

Клієнтська частина проєкту реалізована засобами технології ASP.NET Core MVC із використанням Razor Views. Для формування структури сторінок застосовуються мова розмітки HTML5 та шаблони Razor, які дозволяють поєднувати HTML-код із серверною логікою. Такий підхід забезпечує формування динамічного вмісту сторінок на основі даних, отриманих із бази даних.

Для оформлення інтерфейсу використовується каскадна таблиця стилів CSS3. Вона забезпечує єдиний стиль відображення елементів вебресурсу, підтримує адаптивне компонування сторінок та дозволяє коректно відображати інформацію на різних типах пристроїв. Завдяки використанню сучасних засобів

CSS реалізовано гнучке позиціонування елементів, адаптивні блоки контенту та стилізовані форми введення даних.

Важливу роль у реалізації інтерфейсу відіграє фреймворк Bootstrap. Його використання дозволило значно прискорити процес розробки вебзастосунку та забезпечити адаптивність інтерфейсу без необхідності створення великої кількості власних стилів. Bootstrap використовується для побудови сіткової структури сторінок, оформлення меню навігації, кнопок, карток публікацій, форм введення даних та інших компонентів інтерфейсу.

Для реалізації динамічної поведінки сторінок використовується мова програмування JavaScript. Завдяки її застосуванню забезпечується інтерактивна взаємодія користувача з вебресурсом без необхідності повного перезавантаження сторінки. JavaScript використовується для обробки подій користувача, керування елементами інтерфейсу та реалізації додаткових функціональних можливостей.

Однією з основних сторінок вебзастосунку є головна сторінка, яка виконує функцію точки входу до системи. На ній розміщується загальна інформація про ресурс, відображаються актуальні матеріали та забезпечується швидкий доступ до основних розділів сайту. Формування вмісту головної сторінки здійснюється динамічно на основі інформації, отриманої з бази даних.

Для перегляду тематичних матеріалів реалізовано окремий модуль блогу. Його основним призначенням є відображення публікацій, створених користувачами або адміністраторами ресурсу. Сторінка блогу забезпечує виведення списку матеріалів із можливістю переходу до повного тексту окремої публікації.

Обробка запитів до сторінок блогу здійснюється контролером BlogController.

```
public class BlogController : Controller
{
    private PostRepository _postReposytory;
    private TagRepository _tagRepository;
```

```
    public BlogController(PostRepository postRepository, TagRepository tagRepository)
    {
        _postReposytory = postRepository;
        _tagRepository = tagRepository;
    }
}
```

Наведений фрагмент демонструє використання механізму впровадження залежностей для отримання доступу до репозиторіїв публікацій та тегів. Завдяки цьому контролер може формувати необхідні дані для відображення на сторінках блогу.

Для відображення окремих інформаційних блоків у проєкті використовуються View Components. Їх застосування дозволяє створювати незалежні компоненти інтерфейсу, які можуть багаторазово використовуватися на різних сторінках вебресурсу. Такий підхід підвищує модульність програмного забезпечення та спрощує подальшу підтримку інтерфейсу.

На рисунку 4.2 представлено структуру сторінки публікації вебзастосунку для любителів риболовлі. Сторінка публікації є одним із ключових елементів користувацького інтерфейсу, оскільки саме через неї здійснюється ознайомлення користувачів із тематичними матеріалами, новинами, звітами про риболовлю та іншою інформацією, що публікується на вебресурсі.

У верхній частині сторінки розташовано блок Header, який містить логотип вебресурсу та навігаційне меню. Даний елемент забезпечує швидкий доступ до основних розділів системи та використовується на всіх сторінках вебзастосунку. Нижче розташовується навігаційний блок, який відображає поточне місце користувача у структурі сайту та спрощує переміщення між категоріями контенту.

Основним інформаційним елементом сторінки є заголовок публікації, який відображає тему матеріалу. Після заголовка розміщується головне зображення, що використовується для візуального представлення матеріалу та

привернення уваги користувача. Наступним елементом є блок основного тексту, який містить зміст публікації та формує основну частину інформаційного наповнення сторінки.

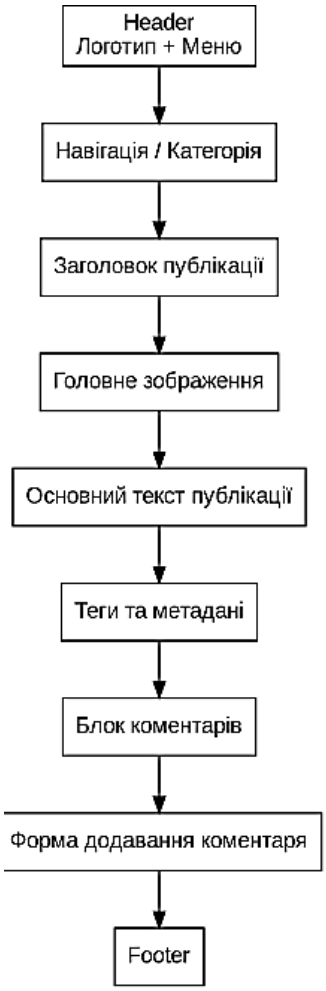


Рисунок 4.2. Структура клієнтської частини вебзастосунку

Після текстового матеріалу відображаються теги та додаткові метадані публікації. Використання тегів забезпечує тематичну класифікацію контенту та спрощує пошук пов'язаних матеріалів. Нижче розташовується блок коментарів, який забезпечує можливість обговорення публікації користувачами вебресурсу. Для додавання нових повідомлень передбачено окрему форму введення коментарів.

У нижній частині сторінки знаходиться блок Footer, який містить

службову інформацію, додаткові навігаційні посилання та контактні відомості. Така структура сторінки забезпечує логічне розташування інформації, покращує сприйняття контенту користувачами та відповідає сучасним принципам проєктування вебінтерфейсів.

Для покращення користувацького досвіду у вебзастосунку реалізовано механізм роботи із зображеннями. Під час створення або редагування публікацій користувач може додавати фотографії, які відображаються разом із текстовим матеріалом. Це дозволяє підвищити інформативність контенту та зробити вебресурс більш привабливим для відвідувачів.

Завантаження файлів здійснюється за допомогою спеціалізованого сервісу FileService.

```
public static void SaveImage(IFormFile file, string fullFilePath)
{
    string? directory = Path.GetDirectoryName(fullFilePath);
    if (directory == null)
        throw new Exception("Invalid path");
    Directory.CreateDirectory(directory);
    using (var stream = new FileStream(fullFilePath, FileMode.Create))
    {
        file.CopyTo(stream);
    }
}
```

Представлений метод забезпечує збереження графічних файлів на сервері та використовується під час роботи із публікаціями. Завдяки централізації механізму завантаження файлів спрощується підтримка системи та забезпечується повторне використання програмного коду.

Однією з важливих особливостей реалізованого вебзастосунку є адаптивність інтерфейсу. Усі сторінки коректно відображаються як на персональних комп'ютерах, так і на планшетах та смартфонах. Це забезпечується використанням адаптивної сітки Bootstrap та застосуванням

медіазапитів CSS.

Для відображення складних сторінок використовуються моделі представлення ViewModel. Вони забезпечують передачу структурованих даних від контролерів до представлень та дозволяють уникнути прямого використання сутностей бази даних у рівні інтерфейсу. Такий підхід відповідає принципам архітектури MVC та підвищує якість програмного коду.

Під час взаємодії користувача із системою реалізовано механізми динамічного оновлення вмісту сторінок. Залежно від вибраної категорії, тегу або окремої публікації користувач отримує відповідний набір даних, який формується контролерами на основі інформації, що зберігається в базі даних. У результаті забезпечується швидка навігація між розділами вебресурсу та зручний доступ до необхідної інформації.

Для підвищення ефективності роботи користувачів значна увага приділялася зручності інтерфейсу. Усі основні функції вебресурсу доступні через навігаційне меню, а структура сторінок побудована таким чином, щоб користувач міг швидко знаходити необхідні матеріали. Використання сучасних засобів веброзробки дозволило реалізувати інтуїтивно зрозумілий інтерфейс, який відповідає вимогам сучасних інформаційних ресурсів.

Таким чином, у процесі реалізації клієнтської частини вебзастосунку було створено адаптивний користувацький інтерфейс, який забезпечує зручну роботу із контентом та підтримує динамічну взаємодію користувача із системою. Використання Razor Views, HTML5, CSS3, JavaScript, Bootstrap та View Components дозволило реалізувати сучасний вебінтерфейс, який забезпечує ефективне використання функціональних можливостей розробленого програмного забезпечення.

4.3. Розробка користувацького інтерфейсу та основних функціональних можливостей

Розробка користувацького інтерфейсу є важливим етапом створення вебзастосунку, оскільки саме через інтерфейс здійснюється взаємодія користувачів із функціональними можливостями системи. Від якості проєктування інтерфейсу залежить зручність використання програмного забезпечення, швидкість доступу до інформації та ефективність виконання користувачами необхідних операцій. Під час розробки вебзастосунку для любителів риболовлі основна увага приділялася створенню інтуїтивно зрозумілого середовища, яке забезпечує зручну навігацію між розділами сайту та швидкий доступ до інформаційного контенту.

Інтерфейс вебзастосунку реалізовано засобами HTML5, CSS3, Bootstrap та Razor Views. Використання даних технологій дозволило створити сучасний адаптивний дизайн, який забезпечує коректне відображення сторінок на різних типах пристроїв. Структура інтерфейсу побудована таким чином, щоб користувач міг швидко знаходити необхідну інформацію незалежно від рівня досвіду роботи з вебресурсом.

Основу навігації вебзастосунку становить головне меню, яке забезпечує доступ до основних функціональних розділів системи. Через меню користувач може перейти до головної сторінки, переглянути публікації блогу, ознайомитися з інформацією про проєкт, скористатися додатковими сервісами або надіслати повідомлення через форму зворотного зв’язку. Для авторизованих користувачів із відповідними правами доступу додатково надається можливість роботи з адміністративною панеллю.

Особливу увагу під час розробки інтерфейсу було приділено представленню інформаційного контенту. Основним функціональним елементом системи є публікації, які відображаються у вигляді окремих інформаційних блоків із заголовком, коротким описом, датою створення та графічним зображенням. Такий підхід забезпечує зручне ознайомлення користувачів із доступними матеріалами та дозволяє швидко переходити до перегляду повної інформації.

Для відображення окремої публікації реалізовано спеціалізовану сторінку, яка містить повний текст матеріалу, графічні елементи, тематичні теги та блок коментарів. Використання структурованого представлення інформації дозволяє підвищити читабельність матеріалів та забезпечити зручне сприйняття контенту користувачами.

Рисунок 4.3. Головна сторінка вебзастосунку

На рисунку 4.3 представлено інтерфейс головної сторінки вебресурсу із відображенням навігаційного меню, інформаційних блоків та основних елементів керування.

Однією з основних функціональних можливостей системи є перегляд та публікація інформаційних матеріалів. Для реалізації даної функції використовується модуль блогу, який забезпечує створення, редагування, видалення та відображення публікацій. Усі операції виконуються через відповідні контролери та репозиторії, що забезпечують взаємодію із базою даних.

Для забезпечення пошуку та тематичного групування інформації використовується механізм категорій і тегів. Категорії дозволяють розподіляти публікації за основними напрямами, тоді як теги забезпечують додаткову деталізацію змісту матеріалів. Використання даних механізмів значно підвищує

зручність роботи із великими обсягами контенту.

Важливою складовою функціональності вебзастосунку є система коментування. Вона забезпечує можливість обговорення публікацій та взаємодії між учасниками спільноти. Користувачі можуть залишати власні повідомлення, ділитися досвідом та брати участь у тематичних дискусіях.

Для роботи із коментарями використовуються відповідні моделі даних та механізми доступу до інформації. У результаті забезпечується збереження повідомлень у базі даних та їх подальше відображення на сторінках публікацій.

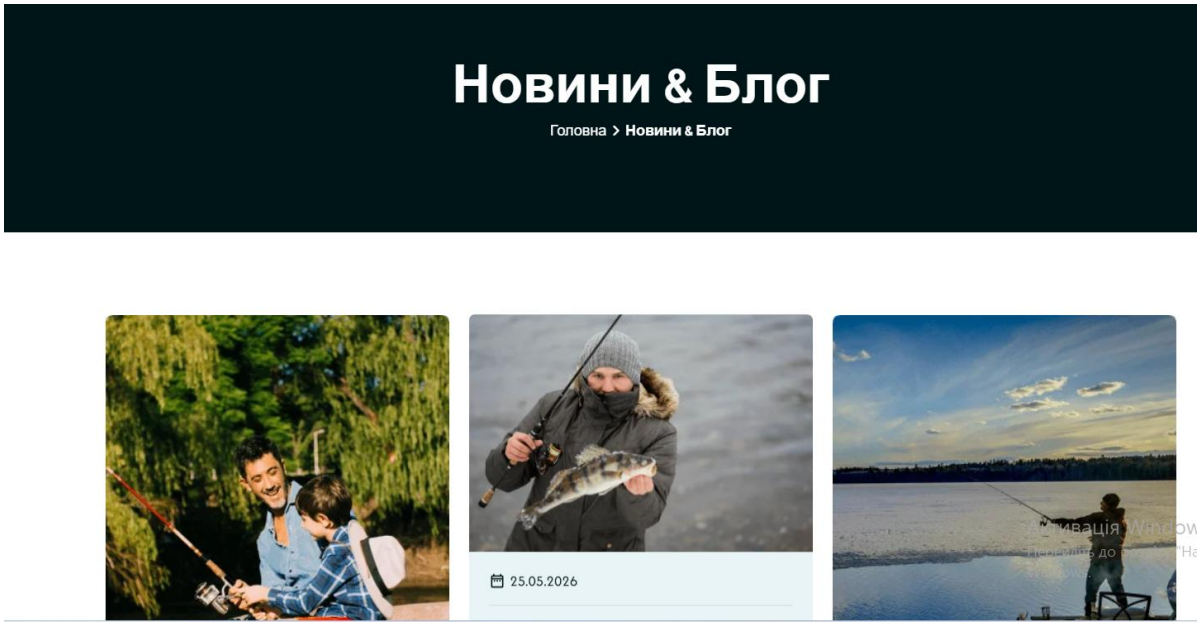


Рисунок 4.4. Сторінка перегляду публікації з блоком коментарів

На рисунку 4.4 представлено інтерфейс сторінки окремої публікації із відображенням основного матеріалу, тегів, коментарів та форми додавання нового повідомлення.

Значну роль у функціонуванні системи відіграє адміністративна панель. Вона призначена для керування інформаційним наповненням вебресурсу та забезпечує доступ до функцій редагування матеріалів, категорій, тегів, навігаційних елементів і користувацьких даних. Завдяки використанню адміністративної панелі спрощується процес супроводу вебресурсу та централізованого керування контентом.

Для забезпечення роботи адміністративної частини застосунку використовується механізм авторизації користувачів. Доступ до адміністративних функцій надається лише користувачам із відповідними правами доступу. Це дозволяє захистити систему від несанкціонованого внесення змін та забезпечити безпеку даних.

Приклад реалізації пошуку користувачів наведено нижче.

Фрагмент роботи з користувачами

```
public User? GetUserByEmail (string email)
{
   return _dbContext.Users.FirstOrDefault(u => u.Email.ToLower() == email.ToLower());
}

public User? GetUserByLogin(string login)
{
   return _dbContext.Users.FirstOrDefault(u => u.Login == login);
}
```

Наведений фрагмент програмного коду демонструє реалізацію доступу до інформації про користувачів через репозиторій та використання технології Entity Framework Core для виконання операцій пошуку.

Важливою функціональною можливістю вебзастосунку є підтримка роботи із графічним контентом. Користувачі можуть додавати зображення до публікацій, що дозволяє зробити матеріали більш інформативними та привабливими для відвідувачів ресурсу. Для реалізації даної функції використовується спеціалізований сервіс роботи із файлами.

Фрагмент сервісу збереження зображень

```
public static void SaveImage(IFormFile file, string fullFilePath)
{
   string? directory = Path.GetDirectoryName(fullFilePath);
```

```
    if (directory == null)
        throw new Exception("Invalid path");

    Directory.CreateDirectory(directory);

    using (var stream = new FileStream(fullFilePath, FileMode.Create))
    {
        file.CopyTo(stream);
    }
}
```

Завдяки використанню даного механізму забезпечується централізоване збереження графічних файлів та їх подальше відображення в інтерфейсі вебресурсу.

Окрему увагу під час розробки інтерфейсу було приділено адаптивності. Усі сторінки коректно відображаються на пристроях із різними розмірами екранів, включаючи персональні комп’ютери, планшети та смартфони. Це забезпечується використанням адаптивної сітки Bootstrap та сучасних засобів CSS.

Таким чином, у процесі розробки користувацького інтерфейсу було реалізовано зручне та функціональне середовище для роботи користувачів із вебзастосунком. Створений інтерфейс забезпечує доступ до всіх функціональних можливостей системи, підтримує адаптивне відображення сторінок, спрощує навігацію між розділами та забезпечує ефективну взаємодію користувачів із інформаційним контентом вебресурсу.

4.4. Практичне використання та особливості функціонування вебзастосунку

Практичне використання розробленого вебзастосунку спрямоване на створення єдиного інформаційного середовища для любителів риболовлі, яке забезпечує можливість публікації тематичних матеріалів, обміну досвідом між користувачами, пошуку корисної інформації та взаємодії учасників спільноти. Реалізований вебресурс може використовуватися як інформаційний портал, тематичний блог або платформа для формування онлайн-спільноти рибалок.

Особливістю функціонування вебзастосунку є використання клієнт-серверної архітектури, за якої обробка запитів здійснюється на серверній стороні із подальшим формуванням динамічного вебконтенту для користувача. Такий підхід дозволяє забезпечити централізоване зберігання інформації, спростити адміністрування ресурсу та підтримувати актуальність усіх даних незалежно від пристрою, з якого здійснюється доступ до системи.

Робота користувача із вебресурсом починається із переходу на головну сторінку. Після відкриття сайту відвідувач отримує доступ до основних розділів системи, навігаційного меню та актуальних публікацій. Завдяки використанню адаптивного інтерфейсу всі елементи сторінки коректно відображаються як на персональних комп'ютерах, так і на мобільних пристроях.

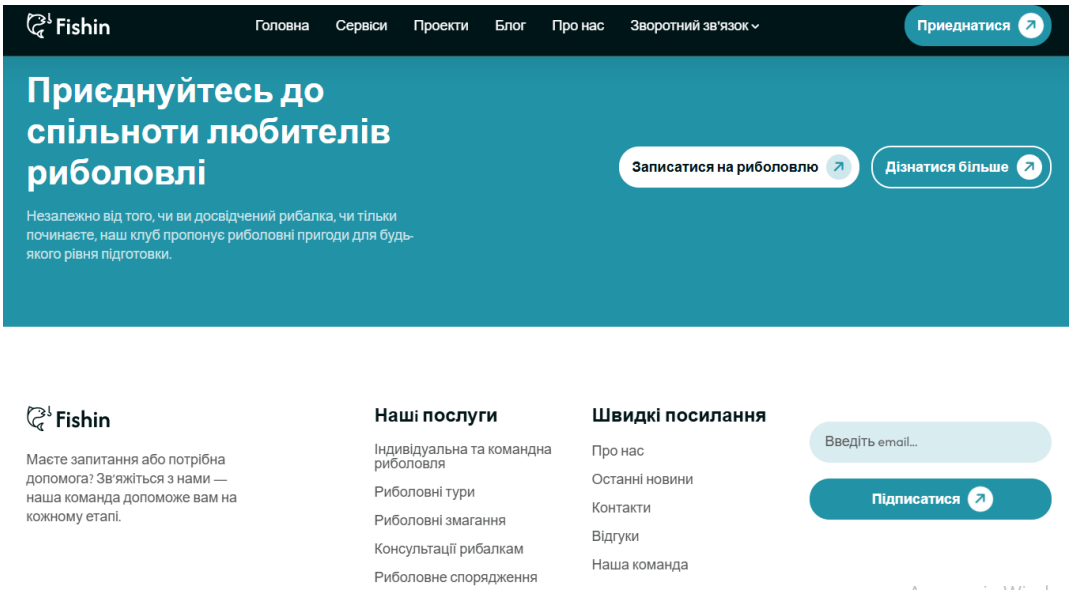


Рисунок 4.5. Функціонал сайту

Однією з основних функціональних можливостей системи є перегляд інформаційних матеріалів. Користувач може ознайомлюватися з публікаціями, переглядати детальну інформацію про окремі матеріали, аналізувати фотографії та використовувати навігаційні механізми для пошуку пов'язаного контенту. Завдяки реалізованій структурі категорій та тегів користувачі можуть швидко знаходити інформацію за тематичними напрямами.

Для забезпечення інтерактивності вебресурсу реалізовано механізм коментування публікацій. Після ознайомлення із матеріалом користувач може залишити власний коментар, поставити запитання або поділитися практичним досвідом. У результаті формується активна взаємодія між учасниками спільноти та створюються додаткові можливості для обміну інформацією.

Важливою особливістю роботи системи є підтримка мультимедійного контенту. Під час створення публікацій користувачі або адміністратори можуть додавати графічні матеріали, які зберігаються на сервері та відображаються безпосередньо на сторінках вебресурсу. Використання фотографій дозволяє зробити матеріали більш наочними та підвищує якість сприйняття інформації.

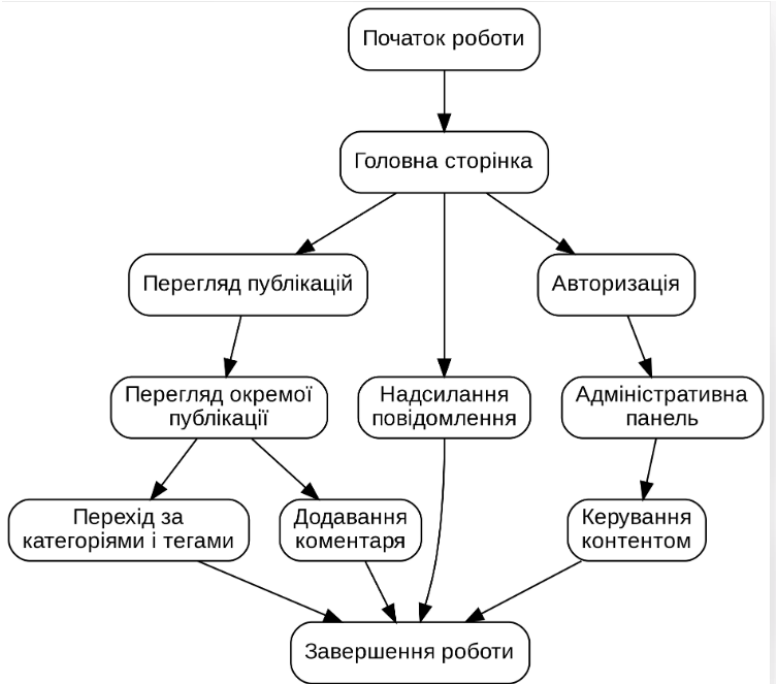


Рисунок 4.6. Сценарій роботи користувача з вебзастосунком

На рисунку 4.6 наведено узагальнений сценарій роботи користувача з вебзастосунком для любителів риболовлі. Взаємодія із системою починається з відкриття головної сторінки, яка надає доступ до основних функціональних можливостей ресурсу. Після переходу на сайт користувач може переглядати доступні публікації, ознайомлюватися з окремими матеріалами та використовувати механізми навігації за категоріями і тегами.

Під час перегляду публікацій користувач отримує можливість ознайомитися з повним змістом матеріалу, переглянути графічні ілюстрації та перейти до пов'язаних записів. Для забезпечення взаємодії між учасниками спільноти реалізовано механізм коментування, який дозволяє залишати повідомлення та брати участь в обговоренні матеріалів.

Окремим напрямом взаємодії із системою є використання форми зворотного зв'язку, через яку користувач може надсилати повідомлення адміністрації вебресурсу. Для користувачів із відповідними правами доступу передбачено процедуру авторизації, після проходження якої відкривається доступ до адміністративної панелі.

Через адміністративний інтерфейс здійснюється керування інформаційним наповненням ресурсу, включаючи створення, редагування та видалення публікацій, категорій, тегів та інших елементів системи. Представлений сценарій демонструє основні варіанти використання вебзастосунку та відображає послідовність взаємодії користувачів із його функціональними можливостями.

Практична експлуатація вебзастосунку передбачає використання різних ролей користувачів. Незареєстровані відвідувачі мають можливість переглядати інформаційний контент, знайомитися з матеріалами та використовувати основні навігаційні можливості сайту. Для користувачів із розширеними правами доступу передбачені додаткові функції керування інформаційним наповненням ресурсу.

Особливе місце в роботі системи займає адміністративна панель. Вона

забезпечує централізоване керування контентом вебресурсу та дозволяє виконувати операції створення, редагування й видалення інформаційних матеріалів. Через адміністративний інтерфейс здійснюється керування категоріями, тегами, навігаційними елементами та іншими компонентами системи.

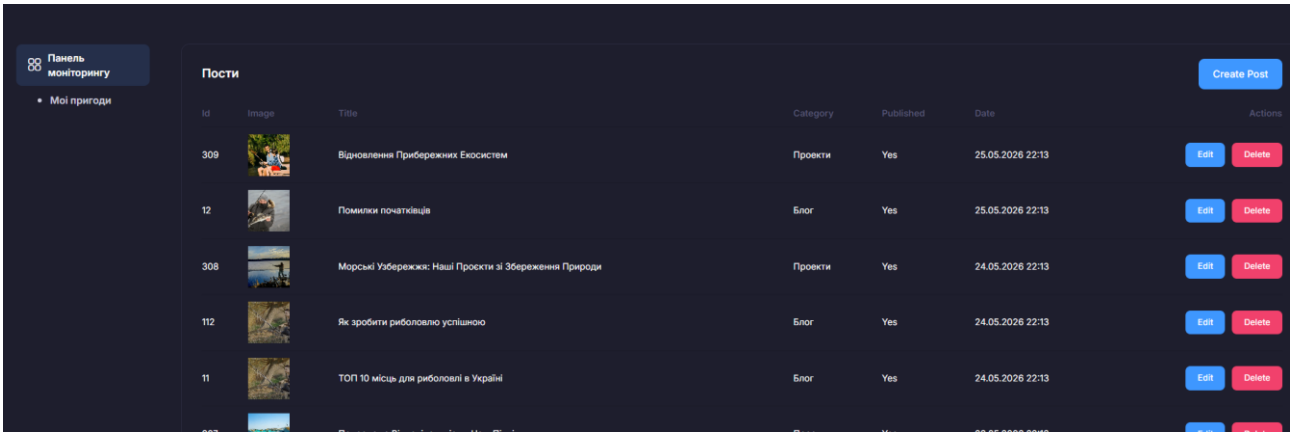
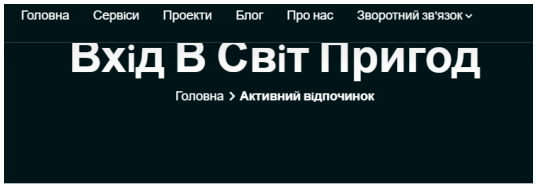


Рисунок 4.7. Адмін панель

Для реалізації функцій адміністрування використовується механізм авторизації користувачів. Після введення облікових даних система виконує перевірку інформації та надає доступ до відповідних функцій згідно з правами конкретного користувача. Такий підхід дозволяє забезпечити захист даних та запобігти несанкціонованому внесенню змін до інформаційного наповнення вебресурсу.



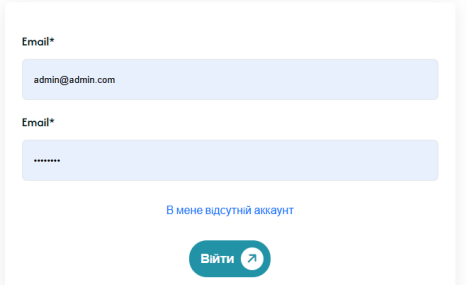


Рисунок 4.8. Авторизація користувача

Практичне використання системи також передбачає застосування механізмів пошуку та фільтрації інформації. Користувачі можуть знаходити матеріали за категоріями, тегами або іншими параметрами, що суттєво спрощує роботу з великими обсягами контенту. Особливо актуальною дана можливість є для ресурсів, які містять значну кількість тематичних публікацій та архівних матеріалів.

Рисунок 4.9. Частина функціоналу сайту

Для підтримки працездатності вебзастосунку використовується база даних Microsoft SQL Server, у якій зберігаються всі основні дані системи. Взаємодія між прикладною логікою та базою даних реалізується засобами Entity Framework Core, що дозволяє автоматизувати виконання операцій створення, читання, оновлення та видалення інформації.

Важливою особливістю функціонування розробленого програмного забезпечення є можливість його подальшого розвитку. Архітектура системи дозволяє додавати нові функціональні модулі без необхідності суттєвої перебудови існуючої структури. У перспективі вебресурс може бути доповнений сервісами бронювання місць для риболовлі, інтерактивними

картами водойм, рейтингами користувачів, особистими кабінетами рибалок та іншими інформаційними сервісами.

Практичне застосування вебзастосунку підтверджує ефективність використання сучасних вебтехнологій для створення інформаційних ресурсів тематичного спрямування. Реалізований програмний продукт забезпечує централізоване зберігання інформації, підтримує взаємодію між користувачами, спрощує керування контентом та створює зручне середовище для обміну досвідом між учасниками спільноти любителів риболовлі.

Отримані результати свідчать про те, що розроблений вебзастосунок повністю виконує поставлені функціональні завдання, забезпечує стабільну роботу основних модулів та може використовуватися як основа для подальшого розвитку інформаційного ресурсу відповідно до потреб його користувачів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено вебзастосунок для любителів риболовлі, призначений для публікації тематичних матеріалів, організації взаємодії між користувачами та формування єдиного інформаційного середовища для обміну досвідом у сфері активного відпочинку та риболовлі.

У процесі виконання роботи проведено аналіз сучасних вебресурсів аналогічного призначення, досліджено особливості їх функціонування, структуру інтерфейсів та технологічні підходи до реалізації. Виконаний аналіз дозволив визначити основні функціональні вимоги до майбутнього програмного забезпечення та сформувати перелік функцій, необхідних для забезпечення ефективної роботи користувачів із вебресурсом.

У роботі обґрунтовано вибір технологій розробки, серед яких платформа ASP.NET Core MVC, мова програмування C#, технологія Entity Framework Core та система керування базами даних Microsoft SQL Server. Використання зазначених засобів дозволило реалізувати сучасний вебзастосунок із багаторівневою архітектурою, що забезпечує розділення функціональності між окремими компонентами системи та спрощує подальший супровід програмного забезпечення.

У ході проєктування розроблено архітектуру вебзастосунку, визначено структуру функціональних модулів, побудовано модель даних та реалізовано взаємозв’язки між основними сутностями предметної області. Створена структура бази даних забезпечує централізоване зберігання інформації про користувачів, публікації, коментарі, категорії, теги та інші об’єкти системи.

Практичним результатом роботи стало створення вебзастосунку, який реалізує механізми публікації та перегляду інформаційних матеріалів, підтримує роботу із категоріями та тегами, забезпечує функціонування системи коментування, навігації та зворотного зв’язку. Також реалізовано

адміністративну панель, призначену для керування інформаційним наповненням вебресурсу та підтримки його актуального стану.

Під час розробки користувацького інтерфейсу застосовано сучасні підходи до побудови вебінтерфейсів із використанням HTML5, CSS3, Bootstrap та Razor Views. Реалізований інтерфейс забезпечує адаптивне відображення сторінок на різних типах пристроїв, підтримує зручну навігацію та створює комфортні умови для роботи користувачів із контентом вебресурсу.

Отримані результати підтверджують досягнення поставленої мети роботи. Розроблений вебзастосунок забезпечує ефективне розміщення та поширення тематичної інформації, підтримує взаємодію між учасниками спільноти та створює основу для подальшого розвитку функціональних можливостей ресурсу. Архітектура програмного забезпечення дозволяє у майбутньому розширювати систему шляхом додавання нових сервісів, пов’язаних із риболовлею, активним відпочинком та інформаційною підтримкою користувачів.

За результатами виконання дипломної роботи опубліковано тези на XLIX Міжнародній науковій конференції студентів та аспірантів «Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті» (м. Полтава, 23 квітня 2026 р.) [27].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anglers' Net. Режим доступу URL: https://www.anglersnet.co.uk/ – Назва з екрану.

2. ASP.NET Core fundamentals overview. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-10.0 – Назва з екрану.

3. ASP.NET Core MVC with EF Core. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/?view=aspnetcore-10.0 – Назва з екрану.

4. ASP.NET documentation. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-10.0 – Назва з екрану.

5. Bassmaster. Режим доступу URL: https://www.bassmaster.com/ – Назва з екрану.

6. Common web application architectures. Режим доступу URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures – Назва з екрану.

7. CSS: Cascading Style Sheets. Режим доступу URL: https://developer.mozilla.org/en-US/docs/Web/CSS – Назва з екрану.

8. Fishingstock. Режим доступу URL: https://fishingstock.ua/ – Назва з екрану.

9. Forms Bootstrap v5.3. Режим доступу URL: https://getbootstrap.com/docs/5.3/forms/overview/ – Назва з екрану.

10. HTML: HyperText Markup Language. Режим доступу URL: https://developer.mozilla.org/en-US/docs/Web/HTML – Назва з екрану.

11. Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-

4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application – Назва з екрану.

12. Introduction Bootstrap v5.3. Режим доступу URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ – Назва з екрану.

13. JavaScript Guide. Режим доступу URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide – Назва з екрану.

14. Microsoft Learn. Introduction to ASP.NET Core MVC. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview – Назва з екрану.

15. Microsoft Learn. Working with Data in ASP.NET Core Applications. Режим доступу URL: https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app – Назва з екрану.

16. Model validation in ASP.NET Core MVC and Razor Pages. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-10.0 – Назва з екрану.

17. Overview of Entity Framework Core – EF Core. Режим доступу URL: https://learn.microsoft.com/en-us/ef/core/ – Назва з екрану.

18. Razor syntax reference for ASP.NET Core. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-10.0 – Назва з екрану.

19. Flagman.ua. Режим доступу URL: https://flagman.ua / – Назва з екрану.

20. SQL Server technical documentation. Режим доступу URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17 – Назва з екрану.

21. Tables Bootstrap v5.3. Режим доступу URL: https://getbootstrap.com/docs/5.3/content/tables/ – Назва з екрану.

22. Upload files in ASP.NET Core. Режим доступу URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/models/file-uploads?view=aspnetcore-10.0 – Назва з екрану.

23. Кошова О. П., Ольховська О. В., Тацій Д. С., Олексійчук Ю. Ф., Черненко О. О. Розробка веб-додатків та сервісів на платформі Node.js // Таврійський науковий вісник. Серія: Технічні науки. 2023. Вип. 2. С. 78–89. Режим доступу URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/358/331

24. Кошова О. П., Черненко О. О., Чілікіна Т. В., Комар І. І. Особливості розробки web-застосунків для системи дистанційного навчання з допомогою бібліотеки React // Системи та технології. 2023. Т. 65, № 1. С. 20–31. Режим доступу URL: https://doi.org/10.32782/2521-6643-2023.1-65.3

25. Ольховська О. В., Черненко О. О. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра. Полтава : ПУЕТ, 2024. 67 с. Режим доступу URL: http://dspace.puet.edu.ua/bitstream/123456789/14768/1/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0%20%D0%91%D0%A0%20%D0%B4%D0%BB%D1%8F%20%D0%9A%D0%9D%202024_%D0%B1%D0%B0%D0%BA%D0%B0%D0%BB%D0%B0%D0%B2%D1%80.pdf – Назва з екрану.

26. Пех П. А., Янковський Б. О. Особливості створення веб-додатків з використанням C# ASP.NET Core MVC // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2025. № 59. С. 253–257. Режим доступу URL: https://doi.org/10.36910/6775-2524-0560-2024-57-32

27. Троневський В. В., Чілікіна Т. В. Проектування веб ресурсу «Тихе полювання» // Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті : тези доповідей XLIX Міжнародної наукової конференції студентів та аспірантів (м. Полтава, 23 квітня 2026 р.). Полтава : ПУЕТ, 2026. С. 464–467. Режим доступу URL: https://puet.edu.ua/wp-content/uploads/2026/06/zb_tez-2026-qual-osv-xxi.pdf

ДОДАТОК А
Код програми

WebApplicationFishingApp/WebApplicationFishingApp/Program.cs

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.EntityFrameworkCore;
using WebApplicationFishingApp.Data;
using WebApplicationFishingApp.Models;
namespace WebApplicationFishingApp
{
   public class Program
   {
      public static void Main(string[] args)
      {
         var builder = WebApplication.CreateBuilder(args);

         var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new
InvalidOperationException("Connection string 'DefaultConnection' not found.");
         builder.Services.AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(connectionString));
         builder.Services.AddDatabaseDeveloperPageExceptionFilter();

         builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme).AddCookie(
            option =>
            {
               option.LoginPath = new PathString("/Fisher/LoginIn");
               option.AccessDeniedPath = new PathString("/Error/AcssessDenied");
            }
         );

         builder.Services.AddScoped<OptionsRepository>();
         builder.Services.AddScoped<NavigationRepository>();
         builder.Services.AddScoped<CategoryReposytory>();
         builder.Services.AddScoped<CommentRepository>();
         builder.Services.AddScoped<NavigationRepository>();
         builder.Services.AddScoped<OptionsRepository>();
         builder.Services.AddScoped<PostRepository>();
         builder.Services.AddScoped<TagRepository>();
         builder.Services.AddScoped<ContactRequestRepository>();

         builder.Services.AddControllersWithViews();
         builder.Services.AddHttpContextAccessor();

         var app = builder.Build();

         using (var scope = app.Services.CreateScope())
         {
            var context = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();
            DbInitializer.Initialize(context);
         }

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

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

         app.UseRouting();

         StaticHttpContextExtensions.UseStaticHttpContext(app);

         app.UseAuthentication();
         app.UseAuthorization();


         app.UseMiddleware<CustomMiddleWare>();

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

```
        app.Run();
      }
   }
}
```

==========
WebApplicationFishingApp/WebApplicationFishingApp/Data/ApplicationDbContext.cs
==========

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using WebApplicationFishingApp.Entities;

namespace WebApplicationFishingApp.Data
{
   public class ApplicationDbContext : IdentityDbContext<IdentityUser>
   {
      public DbSet<Option> Options { get; set; }
      public DbSet<Navigate> Navigations { get; set; }
      public DbSet<Subscribe> Subscribes { get; set; }

      public DbSet<Category> Categories { get; set; }
      public DbSet<Comment> Comments { get; set; }
      public DbSet<Post> Posts { get; set; }
      public DbSet<Tag> Tags { get; set; }
      public DbSet<PostTags> PostTags { get; set; }
      public DbSet<ContactRequest> ContactRequests { get; set; }
      public DbSet<User> Users { get; set; }
      public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
         : base(options)
      {
         Database.EnsureCreated();
      }

      protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
         base.OnModelCreating(modelBuilder);


         modelBuilder.Entity<Option>()
            .HasIndex(o => o.Name)
            .IsUnique();

         modelBuilder.Entity<Subscribe>()
            .HasIndex(o => o.Email)
            .IsUnique();

         modelBuilder.Entity<Category>()
           .HasIndex(c => c.Slug)
           .IsUnique();

         modelBuilder.Entity<Post>()
            .HasIndex(p => p.UrlSlug)
            .IsUnique();


         modelBuilder.Entity<Tag>()
            .HasIndex(t => t.Slug)
            .IsUnique();

         modelBuilder.Entity<PostTags>()
            .HasKey(pt => new { pt.PostId, pt.TagId });

         modelBuilder.Entity<Post>()
            .HasOne(p => p.User)
            .WithMany(u => u.Posts)
            .HasForeignKey(p => p.UserId);
      }
   }
}
```

==========
WebApplicationFishingApp/WebApplicationFishingApp/Controllers/HomeController.cs

```
================================================================================================
using Microsoft.AspNetCore.Mvc;

namespace WebApplicationFishingApp.Controllers
{
   public class HomeController : Controller
   {

      public HomeController()
      {

      }

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



================================================================================================
WebApplicationFishingApp/WebApplicationFishingApp/Controllers/BlogController.cs
================================================================================================
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Hosting;
using WebApplicationFishingApp.Entities;
using WebApplicationFishingApp.Models;
using WebApplicationFishingApp.ViewModels;

namespace WebApplicationFishingApp.Controllers
{
   public class BlogController : Controller
   {

      private PostRepository _postReposytory;
      private TagRepository _tagRepository;

      public BlogController(PostRepository postRepository, TagRepository tagRepository)
      {
         _postReposytory = postRepository;
         _tagRepository = tagRepository;
      }

      public IActionResult Index(string? category, string? tag, int page = 1, int pageSize = 6)
      {
         if (page < 1)
            page = 1;

         var blogViewModel = new BlogViewModel
         {
            CurrentCategory = category,
            CurrentTag = tag,
            CurrentPage = page,
            PageSize = pageSize
         };

         IEnumerable<Post> posts;

         if (!string.IsNullOrEmpty(category))
         {
            posts = _postReposytory.GetPostsByCategorySlug(category);
         }
         else if (!string.IsNullOrEmpty(tag))
         {
            posts = _postReposytory.GetPostsByTagSlug(tag);
         }
         else
         {
            posts = _postReposytory.GetPosts();
         }

         posts = posts.OrderByDescending(p => p.DateOfPublished);

         blogViewModel.TotalItems = posts.Count();
```

```
            blogViewModel.Posts = posts
                .Skip((page - 1) * pageSize)
                .Take(pageSize)
                .ToList();

            return View("AllPosts", blogViewModel);
        }

        public IActionResult PostNotFound()
        {
            return View("PostNotFound");
        }

        [HttpGet]
        public IActionResult SinglePost(string? slug)
        {
            if (slug == null)
            {
                return RedirectToAction("Index", "Error");
            }
            Post? onePost = _postReposytory.GetPostBySlug(slug);
            if (onePost == null)
            {
                return RedirectToAction("Index", "Error");
            }
            return View(onePost);
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult AddComment(Comment comment, string slug)
        {
            if (!ModelState.IsValid)
            {
                return RedirectToAction("SinglePost", new { slug });
            }

            _postReposytory.AddComment(comment);

            return RedirectToAction("SinglePost", new { slug });
        }
    }
}
```

==========
WebApplicationFishingApp/WebApplicationFishingApp/Controllers/AdminController.cs
==========

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using WebApplicationFishingApp.Data;
using WebApplicationFishingApp.Dto.Requests;
using WebApplicationFishingApp.Entities;
using WebApplicationFishingApp.Helpers;
using WebApplicationFishingApp.Models;

namespace WebApplicationVetClinic.Controllers
{
    [Authorize]
    public class AdminController : Controller
    {
        private ApplicationDbContext _dbContext;

        private OptionsRepository _optionModel;
        private TagRepository _tagRepository;
        private CategoryReposytory _categoryRepository;
        private CommentRepository _commentRepository;
        private NavigationRepository _navigationRepository;
        private PostRepository _postRepository;
        private UserRepository _userRepository;
        public AdminController(ApplicationDbContext dbContext)
        {
            _dbContext = dbContext;
            _postRepository = new PostRepository(dbContext);
            _userRepository = new UserRepository(dbContext);
            _categoryRepository = new CategoryReposytory(dbContext);
            _tagRepository = new TagRepository(dbContext);
```

```
}

[HttpGet]
public IActionResult Index()
{
    return RedirectToAction("Posts");
}

[HttpGet]
public IActionResult Posts()
{
    string login = User.Identity.Name;

    var posts = _postRepository.GetPostsByUser(login);

    return View(posts);
}

[HttpGet]
public IActionResult CreatePost()
{
    ViewBag.Categories = _categoryRepository.GetAllCategories();
    ViewBag.Tags = _tagRepository.GetAllTags();

    return View(new PostCreateDto
    {
        DateOfPublished = DateTime.Now
    });
}

[HttpPost]
public IActionResult SaveNewPost(PostCreateDto dto)
{
    ViewBag.Categories = _categoryRepository.GetAllCategories();
    ViewBag.Tags = _tagRepository.GetAllTags();

    if (!ModelState.IsValid)
    {
        return View("CreatePost", dto);
    }

    if (dto.Avatar == null || dto.Avatar.Length == 0)
    {
        ModelState.AddModelError("Avatar", "Avatar is required.");
        return View("CreatePost", dto);
    }

    if (!ImageValidator.IsValidImage(dto.Avatar, out string error))
    {
        ModelState.AddModelError("Avatar", error);
        return View("CreatePost", dto);
    }

    string? login = User.Identity?.Name;

    if (string.IsNullOrEmpty(login))
    {
        return RedirectToAction("LoginIn", "Fisher");
    }

    User? currentUser = _userRepository.GetUserByLogin(login);

    if (currentUser == null)
    {
        return RedirectToAction("LoginIn", "Fisher");
    }

    var origExt = Path.GetExtension(dto.Avatar.FileName);

    string uploadFolder = Path.Combine(
        Directory.GetCurrentDirectory(),
        "wwwroot",
        "images",
        "posts"
    );

    if (!Directory.Exists(uploadFolder))
```

```
    {
        Directory.CreateDirectory(uploadFolder);
    }

    string safeFileName = Guid.NewGuid().ToString();
    string fileName = $"{safeFileName}{origExt}";
    string physicalPath = Path.Combine(uploadFolder, fileName);

    FileService.SaveImage(dto.Avatar, physicalPath);

    Category? blogCategory = _categoryRepository.GetCategoryBySlug("blog");

    var post = new Post
    {
        Title = dto.Title ?? string.Empty,
        Slogan = dto.Slogan ?? string.Empty,
        ImgAlt = dto.ImgAlt ?? string.Empty,
        ImgSrc = $"/images/posts/{fileName}",
        UrlSlug = dto.UrlSlug ?? string.Empty,
        CategoryId = blogCategory.Id,
        Content = dto.Content ?? string.Empty,
        PostedOn = dto.PostedOn,
        DateOfPublished = dto.DateOfPublished ?? DateTime.Now,

        UserId = currentUser.Id
    };

    if (_postRepository.SavePost(post, dto.TagIds))
    {
        return RedirectToAction("Posts", "Admin");
    }

    ModelState.AddModelError("", "Post was not saved.");
    return View("CreatePost", dto);
}

[HttpGet]
public IActionResult EditPost(int id)
{
    string? login = User.Identity?.Name;

    if (string.IsNullOrEmpty(login))
    {
        return RedirectToAction("LoginIn", "Fisher");
    }

    var post = _postRepository.GetUserPostById(id, login);

    if (post == null)
    {
        return Forbid();
    }

    ViewBag.Categories = _categoryRepository.GetAllCategories();
    ViewBag.Tags = _tagRepository.GetAllTags();

    var dto = new PostEditDto
    {
        Id = post.Id,
        Title = post.Title,
        Slogan = post.Slogan,
        ImgAlt = post.ImgAlt,
        CurrentImgSrc = post.ImgSrc,
        UrlSlug = post.UrlSlug,
        CategoryId = post.CategoryId,
        Content = post.Content,
        PostedOn = post.PostedOn,
        DateOfPublished = post.DateOfPublished,
        TagIds = post.PostTags.Select(x => x.TagId).ToList()
    };

    return View(dto);
}
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult SaveEditedPost(PostEditDto dto)
{
```

```
    ViewBag.Categories = _categoryRepository.GetAllCategories();
    ViewBag.Tags = _tagRepository.GetAllTags();

    if (!ModelState.IsValid)
    {
        return View("EditPost", dto);
    }

    string? login = User.Identity?.Name;

    if (string.IsNullOrEmpty(login))
    {
        return RedirectToAction("LoginIn", "Fisher");
    }

    var currentPost = _postRepository.GetUserPostById(dto.Id, login);

    if (currentPost == null)
    {
        return Forbid();
    }

    string imagePath = currentPost.ImgSrc;

    if (dto.Avatar != null && dto.Avatar.Length > 0)
    {
        if (!ImageValidator.IsValidImage(dto.Avatar, out string error))
        {
            ModelState.AddModelError("Avatar", error);
            return View("EditPost", dto);
        }

        var origExt = Path.GetExtension(dto.Avatar.FileName);

        string uploadFolder = Path.Combine(
            Directory.GetCurrentDirectory(),
            "wwwroot",
            "images",
            "posts"
        );

        if (!Directory.Exists(uploadFolder))
        {
            Directory.CreateDirectory(uploadFolder);
        }

        string safeFileName = Guid.NewGuid().ToString();
        string fileName = $"{safeFileName}{origExt}";
        string physicalPath = Path.Combine(uploadFolder, fileName);

        FileService.SaveImage(dto.Avatar, physicalPath);

        imagePath = $"/images/posts/{fileName}";
    }
    Category? blogCategory = _categoryRepository.GetCategoryBySlug("blog");
    currentPost.Title = dto.Title ?? string.Empty;
    currentPost.Slogan = dto.Slogan ?? string.Empty;
    currentPost.ImgAlt = dto.ImgAlt ?? string.Empty;
    currentPost.ImgSrc = imagePath;
    currentPost.UrlSlug = dto.UrlSlug ?? string.Empty;
    currentPost.CategoryId = blogCategory.Id;
    currentPost.Content = dto.Content ?? string.Empty;
    currentPost.PostedOn = dto.PostedOn;
    currentPost.DateOfPublished = dto.DateOfPublished ?? DateTime.Now;

    if (_postRepository.UpdatePost(currentPost, dto.TagIds))
    {
        return RedirectToAction("Posts", "Admin");
    }

    ModelState.AddModelError("", "Post update failed.");
    return View("EditPost", dto);
}

[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult DeletePost(int id)
```

```
        {
            string? login = User.Identity?.Name;

            if (string.IsNullOrEmpty(login))
            {
                return RedirectToAction("LoginIn", "Fisher");
            }

            var post = _postRepository.GetUserPostById(id, login);

            if (post == null)
            {
                return Forbid();
            }

            if (_postRepository.RemovePost(post))
            {
                return RedirectToAction("Posts", "Admin");
            }

            return RedirectToAction("Error");
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult TogglePostPublished(int id)
        {
            string? login = User.Identity?.Name;

            if (string.IsNullOrEmpty(login))
            {
                return RedirectToAction("LoginIn", "Fisher");
            }

            var post = _postRepository.GetUserPostById(id, login);

            if (post == null)
            {
                return Forbid();
            }

            post.PostedOn = !post.PostedOn;

            _postRepository.UpdatePost(
                post,
                post.PostTags.Select(x => x.TagId).ToList()
            );

            return RedirectToAction("Posts", "Admin");
        }
    }
}
```

```
================================================================================
WebApplicationFishingApp/WebApplicationFishingApp/Models/PostRepository.cs
================================================================================
using Microsoft.EntityFrameworkCore;
using WebApplicationFishingApp.Data;
using WebApplicationFishingApp.Entities;

namespace WebApplicationFishingApp.Models
{
    public class PostRepository
    {
        private readonly ApplicationDbContext _dBContext;
        public PostRepository(ApplicationDbContext clinicDBContext)
        {
            _dBContext = clinicDBContext;
        }

        public IEnumerable<Post> GetPostsByCategorySlug(string categorySlug)
        {
            return _dBContext.Posts
                .AsNoTracking()
                .Include(p => p.Category)
```

```
        .Include(p => p.Comments)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .Where(p => p.PostedOn && p.Category != null && p.Category.Slug == categorySlug)
        .OrderByDescending(p => p.DateOfPublished)
        .ToList();
}
public List<Post> GetPostsByUser(string login)
{
    return _dBContext.Posts
        .Include(p => p.User)
        .Include(p => p.Category)
        .Include(p => p.Comments)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .Where(p => p.User.Login == login)
        .OrderByDescending(p => p.DateOfPublished)
        .ThenByDescending(p => p.Id)
        .ToList();
}
public Post? GetUserPostById(int postId, string login)
{
    return _dBContext.Posts
        .Include(p => p.User)
        .FirstOrDefault(p =>
            p.Id == postId &&
            p.User.Login == login);
}
public IEnumerable<Post> GetPostsByTagSlug(string tagSlug)
{
    return _dBContext.Posts
        .AsNoTracking()
        .Include(p => p.Category)
        .Include(p => p.Comments)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .Where(p => p.PostedOn && p.PostTags.Any(pt => pt.Tag.Slug == tagSlug))
        .OrderByDescending(p => p.DateOfPublished)
        .ToList();
}

public IEnumerable<Post> GetPosts()
{
    return _dBContext.Posts
        .AsNoTracking()
        .Include(p => p.Category)
        .Include(p => p.Comments)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .Where(p => p.PostedOn)
        .OrderByDescending(p => p.DateOfPublished)
        .ToList();
}

public List<Post> GetAllPostsForAdmin()
{
    return _dBContext.Posts
        .Include(p => p.Category)
        .Include(p => p.Comments)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .OrderByDescending(p => p.DateOfPublished)
        .ThenByDescending(p => p.Id)
        .ToList();
}

public Post? GetOnePost(string slug)
{
    return _dBContext.Posts
        .Include(p => p.Category)
        .Include(p => p.PostTags)
            .ThenInclude(pt => pt.Tag)
        .FirstOrDefault(x => x.UrlSlug.Contains(slug));
}

public List<Post> GetLatestPosts(int count)
{
```

```
    return _dBContext.Posts
        .AsNoTracking()
        .Where(p => p.PostedOn)
        .OrderByDescending(p => p.DateOfPublished)
        .Include(p => p.Comments)
        .Take(count)
        .ToList();
}

public Post? GetPostBySlug(string slug)
{
    return _dBContext.Posts
        .Include(x => x.Category)
        .Include(x => x.Comments)
        .Include(x => x.PostTags)
            .ThenInclude(x => x.Tag)
        .FirstOrDefault(x => x.UrlSlug == slug);
}

public Post? GetPostById(int postId)
{
    return _dBContext.Posts
        .Include(p => p.Category)
        .Include(p => p.PostTags)
        .FirstOrDefault(p => p.Id == postId);
}

public bool SavePost(Post post, List<int> tagIds)
{
    try
    {
        _dBContext.Posts.Add(post);
        _dBContext.SaveChanges();

        if (tagIds.Any())
        {
            var postTags = tagIds
                .Distinct()
                .Select(tagId => new PostTags
                {
                    PostId = post.Id,
                    TagId = tagId
                })
                .ToList();

            _dBContext.PostTags.AddRange(postTags);
            _dBContext.SaveChanges();
        }

        return true;
    }
    catch
    {
        return false;
    }
}

public bool UpdatePost(Post post, List<int> tagIds)
{
    try
    {
        var currentPost = _dBContext.Posts
            .Include(x => x.PostTags)
            .FirstOrDefault(x => x.Id == post.Id);

        if (currentPost == null)
            return false;

        currentPost.Title = post.Title ?? string.Empty;
        currentPost.Slogan = post.Slogan ?? string.Empty;
        currentPost.ImgSrc = post.ImgSrc ?? string.Empty;
        currentPost.ImgAlt = post.ImgAlt ?? string.Empty;
        currentPost.UrlSlug = post.UrlSlug ?? string.Empty;
        currentPost.CategoryId = post.CategoryId;
        currentPost.Content = post.Content ?? string.Empty;
        currentPost.PostedOn = post.PostedOn;
        currentPost.DateOfPublished = post.DateOfPublished;
```

```
            var oldPostTags = _dBContext.PostTags.Where(x => x.PostId == post.Id).ToList();
            if (oldPostTags.Any())
            {
                _dBContext.PostTags.RemoveRange(oldPostTags);
            }

            if (tagIds.Any())
            {
                var newPostTags = tagIds
                    .Distinct()
                    .Select(tagId => new PostTags
                    {
                        PostId = post.Id,
                        TagId = tagId
                    })
                    .ToList();

                _dBContext.PostTags.AddRange(newPostTags);
            }

            _dBContext.SaveChanges();
            return true;
        }
        catch
        {
            return false;
        }
    }

    public bool RemovePost(Post remove)
    {
        try
        {
            var postTags = _dBContext.PostTags
                .Where(pt => pt.PostId == remove.Id)
                .ToList();

            _dBContext.PostTags.RemoveRange(postTags);
            _dBContext.Posts.Remove(remove);
            _dBContext.SaveChanges();

            return true;
        }
        catch
        {
            return false;
        }
    }
    public void AddComment(Comment comment)
    {
        _dBContext.Comments.Add(comment);
        _dBContext.SaveChanges();
    }
  }
}
```

==========
WebApplicationFishingApp/WebApplicationFishingApp/Models/CommentRepository.cs
==========

```
using WebApplicationFishingApp.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApplicationFishingApp.Data;
using Microsoft.EntityFrameworkCore;

namespace WebApplicationFishingApp.Models
{
  public class CommentRepository
  {
    private readonly ApplicationDbContext _dBContext;
    public CommentRepository(ApplicationDbContext clinicDBContext)
    {
```

```
        _dBContext = clinicDBContext;
    }
    public List<Comment> GetPostComments(int postId)
    {
        return _dBContext.Comments.Where(p=> p.PostId == postId ).OrderBy(x => x.CreatedAt).ToList();
    }
    public IQueryable<Comment> GetPostReviews(int postId)
    {
        return _dBContext.Comments.Where(p => p.PostId == postId ).OrderBy(x => x.CreatedAt);
    }

    public bool SaveComment(Comment comment)
    {
        _dBContext.Comments.Add(comment);
        return _dBContext.SaveChanges() == 1 ? true : false;
    }


    public Comment? GetCommentById(int id)
    {
        return _dBContext.Comments
            .Include(x => x.Post)
            .FirstOrDefault(x => x.Id == id);
    }

    public bool DeleteComment(Comment comment)
    {
        try
        {
            _dBContext.Comments.Remove(comment);
            _dBContext.SaveChanges();
            return true;
        }
        catch
        {
            return false;
        }
    }
  }
}
```

```
================================================================================================
WebApplicationFishingApp/WebApplicationFishingApp/ViewModels/BlogViewModel.cs
================================================================================================
using WebApplicationFishingApp.Entities;

namespace WebApplicationFishingApp.ViewModels
{
  public class BlogViewModel
  {
    public IEnumerable<Post> Posts { get; set; } = new List<Post>();


    public int CurrentPage { get; set; }
    public int PageSize { get; set; }
    public int TotalItems { get; set; }

    public int TotalPages => (int)Math.Ceiling((double)TotalItems / PageSize);

    public string? CurrentCategory { get; set; }
    public string? CurrentTag { get; set; }
  }
}
```

```
================================================================================================
WebApplicationFishingApp/WebApplicationFishingApp/Views/Home/Index.cshtml
================================================================================================
<div class="hero-area1">

  <div class="container">

    <div class="row">
```

```
<div class="col-lg-5">

   <div class="main-heading">

      <div class="image1 image-anime reveal">
         <img src="~/img/hero/hero1-image1.png" alt="">
      </div>

      <div class="image2 shape-animaiton4">
         <img src="~/img/shapes/hero-image1-shape1.png" alt="">
      </div>

      <div class="image3 animate2">
         <img src="~/img/shapes/hero-image1-shape2.png" alt="">
      </div>

      <div class="space30"></div>

      <div class="heading1-w">

         <p data-aos="zoom-in-up"
           data-aos-duration="700">

            Наш клуб об’єднує людей, які люблять
            природу, риболовлю та активний відпочинок.
            Ми створюємо незабутні пригоди,
            організовуємо риболовні тури,
            експедиції та подорожі для всіх,
            хто прагне відчути справжню свободу
            серед дикої природи.

         </p>

      </div>

      <div class="space30"></div>

      <div class="buttons"
          data-aos="zoom-in-up"
          data-aos-duration="900">

         <a class="theme-btn1"
           href="/About/ContactUs">

            Зв’язатися з нами

            <span class="arrow">
               <i class="fa-solid fa-arrow-right"></i>
            </span>

         </a>

         <a class="theme-btn3"
           href="/Service">

            Наші послуги

            <span class="arrow">
               <i class="fa-solid fa-arrow-right"></i>
            </span>

         </a>

      </div>

   </div>

</div>

<div class="col-lg-7">

   <div class="main-heading">

      <h1 class="text-anime-style-3">

         Відчуйте Азарт Пригод
         Та Насолоду Справжньої Риболовлі
```

```
                </h1>

                <div class="space60"></div>

                <div class="main-image">

                    <div class="reveal image-anime">
                        <img src="~/img/hero/hero1-image2.png" alt="">
                    </div>

                    <div class="shape animate4">
                        <img src="~/img/shapes/hero-image2-shape1.png" alt="">
                    </div>

                </div>

            </div>

        </div>

    </div>

    <img class="sec-shape1"
        src="~/img/shapes/hero-sec-shape1.png"
        alt="">

    <img class="sec-shape2"
        src="~/img/shapes/hero-sec-shape2.png"
        alt="">

    <img class="sec-shape3 animate1"
        src="~/img/shapes/hero-sec-shape3.png"
        alt="">

    <img class="sec-shape4 animate3"
        src="~/img/shapes/hero-sec-shape4.png"
        alt="">
</div>

<!--====== HERO AREA END ========-->
<!--====== SERVICE AREA START ========-->
@await Component.InvokeAsync("RandomProjects")

<!--====== SERVICE AREA END ========-->
<!--====== ABOUT AREA START ========-->

<div class="about1 sp bg1">
    <div class="container">
        <div class="row">

            <div class="col-lg-6">

                <div class="heading1">

                    <span class="span"
                        data-aos="zoom-in-left"
                        data-aos-duration="700">

                        <img src="~/img/icons/span-icon1.png" alt="">
                        Про Нас

                    </span>

                    <h2 class="text-anime-style-3">
                        Створюємо незабутні спогади серед дикої природи
                    </h2>

                    <div class="space16"></div>

                    <p data-aos="fade-left"
                        data-aos-duration="800">

                        Протягом багатьох років наш клуб об’єднує
                        людей, закоханих у риболовлю, активний
```

```
    відпочинок та пригоди на природі.
    Ми створюємо простір для спілкування,
    навчання та нових відкриттів серед
    мальовничих водойм і лісів.

</p>

<div class="row">

    <div class="col-md-6">

        <div class="about1-box"
            data-aos="zoom-in-up"
            data-aos-duration="700">

            <div class="progresbar">
                <div class="progressbar">

                    <div class="circle"
                        data-percent="95">

                        <div class="count">95%</div>

                    </div>

                </div>
            </div>

            <div class="space16"></div>

            <div class="heading1">

                <h5>
                    <a href="#">
                        Прихильники активного відпочинку
                    </a>
                </h5>

                <div class="space16"></div>

                <p>
                    Наш клуб створений для тих,
                    хто цінує свободу, пригоди,
                    свіже повітря та справжню
                    атмосферу дикої природи.
                </p>

            </div>

        </div>

    </div>

    <div class="col-md-6">

        <div class="about1-box"
            data-aos="zoom-in-up"
            data-aos-duration="900">

            <div class="progresbar">
                <div class="progressbar">

                    <div class="circle"
                        data-percent="99">

                        <div class="count">99%</div>

                    </div>

                </div>
            </div>

            <div class="space16"></div>

            <div class="heading1">

                <h5>
```

```
                        <a href="#">
                            Майстри риболовлі та пригод
                        </a>
                    </h5>

                    <div class="space16"></div>

                    <p>
                        Ми ділимося досвідом,
                        організовуємо експедиції,
                        навчаємо новачків та
                        допомагаємо створювати
                        незабутні враження.
                    </p>

                </div>

            </div>

        </div>

        <div class="space40"></div>

        <div class="row">

            <div class="col-12">

                <div class="buttons"
                    data-aos="zoom-in-up"
                    data-aos-duration="900">

                    <a class="theme-btn1"
                      href="/Services">

                        Дізнатися більше

                        <span class="arrow">
                            <i class="fa-solid fa-arrow-right"></i>
                        </span>

                    </a>

                    <div class="contact-btn">

                        <div class="icon">
                            <img src="~/img/icons/about1-phone.png" alt="">
                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

    <div class="col-lg-6">

        <div class="about-main-image">
            <img src="~/img/about/about1-main-image.png" alt="">
        </div>

    </div>

  </div>
</div>

<img src="~/img/shapes/about1-shape1.png"
    alt=""
    class="shape1 animate1">
```

```
    <img src="~/img/shapes/about1-shape2.png"
        alt=""
        class="shape2 animate3">

    <img src="~/img/shapes/about1-shape3.png"
        alt=""
        class="shape3 animate4">
</div>
<!--====== ABOUT AREA END ========-->
<!--====== CASE AREA STARTS ========-->
@await Component.InvokeAsync("PortfolioProjects")
<!--====== CASE AREA ENDS ========-->

<!--====== FAQ AREA START ========-->

<div class="faq sp">
    <div class="container">

        <div class="row">

            <div class="col-lg-6 m-auto text-center">

                <div class="heading1">

                    <span class="span"
                        data-aos="zoom-in-left"
                        data-aos-duration="700">

                        <img src="~/img/icons/span-icon1.png" alt="">
                        FAQ

                    </span>

                    <h2 class="text-anime-style-3">
                        Відповіді на Популярні Запитання
                    </h2>

                    <div class="space16"></div>

                    <p data-aos="fade-left"
                        data-aos-duration="700">

                        Ми зібрали відповіді на найпоширеніші
                        запитання про наш клуб, риболовні тури,
                        участь у заходах, оренду спорядження
                        та організацію активного відпочинку.

                    </p>

                </div>

            </div>

        </div>

        <div class="space30"></div>

        <div class="row">

            <div class="col-lg-4">

                <div class="space40"></div>

                <div class="faq-image image-anime reveal">
                    <img src="~/img/others/faq1-image.png" alt="">
                </div>

            </div>

            <div class="col-lg-8">

                <div class="space40"></div>

                <div class="accordion" id="accordionExample">

                    <div class="accordion-item"
                        data-aos="fade-up"
```

```
    data-aos-duration="600">

    <h2 class="accordion-header" id="headingOne">

        <button class="accordion-button"
                type="button"
                data-bs-toggle="collapse"
                data-bs-target="#collapseOne"
                aria-expanded="true"
                aria-controls="collapseOne">

            Чи надаєте ви спорядження для риболовлі?

        </button>

    </h2>

    <div id="collapseOne"
        class="accordion-collapse collapse show"
        aria-labelledby="headingOne"
        data-bs-parent="#accordionExample">

        <div class="accordion-body">

            Так, ми пропонуємо оренду сучасного
            риболовного спорядження та необхідного
            обладнання для комфортного та безпечного
            відпочинку на природі.

        </div>

    </div>

</div>

<div class="accordion-item"
    data-aos="fade-up"
    data-aos-duration="800">

    <h2 class="accordion-header" id="headingTwo">

        <button class="accordion-button collapsed"
                type="button"
                data-bs-toggle="collapse"
                data-bs-target="#collapseTwo"
                aria-expanded="false"
                aria-controls="collapseTwo">

            Які правила діють на території клубу?

        </button>

    </h2>

    <div id="collapseTwo"
        class="accordion-collapse collapse"
        aria-labelledby="headingTwo"
        data-bs-parent="#accordionExample">

        <div class="accordion-body">

            Ми дотримуємося принципів безпечного
            та відповідального відпочинку,
            поваги до природи та правил
            риболовлі, встановлених для наших локацій.

        </div>

    </div>

</div>

<div class="accordion-item"
    data-aos="fade-up"
    data-aos-duration="900">

    <h2 class="accordion-header" id="headingThree">
```

```
        <button class="accordion-button collapsed"
                type="button"
                data-bs-toggle="collapse"
                data-bs-target="#collapseThree"
                aria-expanded="false"
                aria-controls="collapseThree">

            Чи є сезонні обмеження для риболовлі?

        </button>

    </h2>

    <div id="collapseThree"
        class="accordion-collapse collapse"
        aria-labelledby="headingThree"
        data-bs-parent="#accordionExample">

        <div class="accordion-body">

            Так, деякі види риболовлі можуть
            мати сезонні обмеження відповідно
            до природоохоронних правил та
            періоду нересту.

        </div>

    </div>

</div>

<div class="accordion-item"
    data-aos="fade-up"
    data-aos-duration="1000">

    <h2 class="accordion-header" id="headingFour">

        <button class="accordion-button collapsed"
                type="button"
                data-bs-toggle="collapse"
                data-bs-target="#collapseFour"
                aria-expanded="false"
                aria-controls="collapseFour">

            Чи проводите ви навчання для новачків?

        </button>

    </h2>

    <div id="collapseFour"
        class="accordion-collapse collapse"
        aria-labelledby="headingFour"
        data-bs-parent="#accordionExample">

        <div class="accordion-body">

            Наш клуб регулярно організовує
            майстер-класи, тренування та
            практичні заняття для початківців
            і досвідчених рибалок.

        </div>

    </div>

</div>

<div class="accordion-item"
    data-aos="fade-up"
    data-aos-duration="1100">

    <h2 class="accordion-header" id="headingFive">

        <button class="accordion-button collapsed"
                type="button"
```

```
                        data-bs-toggle="collapse"
                        data-bs-target="#collapseFive"
                        aria-expanded="false"
                        aria-controls="collapseFive">

                        Чи можна брати участь усією родиною?

                    </button>

                </h2>

                <div id="collapseFive"
                    class="accordion-collapse collapse"
                    aria-labelledby="headingFive"
                    data-bs-parent="#accordionExample">

                    <div class="accordion-body">

                        Так, ми проводимо сімейні заходи,
                        дитячі активності та програми,
                        які підходять для відпочинку
                        разом із друзями та родиною.

                    </div>

                </div>

            </div>

        </div>

    </div>
</div>

<!--====== FAQ AREA END ========-->


================================================================================================
WebApplicationFishingApp/WebApplicationFishingApp/Views/Blog/SinglePost.cshtml
================================================================================================
@model WebApplicationFishingApp.Entities.Post

@{
    var post = Model;
}
<div class="inner-hero">
    <div class="container">
        <div class="rwo">
            <div class="col-lg-8 m-auto text-cetner">
                <div class="main-heading">
                    <h1>@post.Title</h1>
                </div>
            </div>
        </div>
    </div>
    <img src="~/img/shapes/inner-hero-shape1.png" alt="" class="shape1">
    <img src="~/img/shapes/inner-hero-shape2.png" alt="" class="shape2">
    <img src="~/img/shapes/inner-hero-shape3.png" alt="" class="shape3">
</div>

<div class="service-details sp">
    <div class="container">
        <div class="row">

            <div class="col-lg-8 m-auto">

                <article>
                    <div class="details-info-box">

                        <div class="image">
                            <img src="@post.ImgSrc" alt="@post.ImgAlt">
                        </div>
```

```
        <div class="space30"></div>

        <div class="autor-list">

            <a href="#" class="icons">
                <img src="~/img/icons/blog-details-icon1.svg" alt="">
                @post.DateOfPublished.ToString("dd.MM.yyyy")
            </a>

            @if (post.Category != null)
            {
                <a asp-controller="Blog"
                   asp-action="Index"
                   asp-route-category="@post.Category.Slug"
                   class="icons">
                    @post.Category.Title
                </a>
            }
        </div>

        <div class="space20"></div>
        <div class="heading1">
            <h3>@post.Title</h3>

            <div class="space16"></div>
            <p>@post.Content</p>
        </div>
    </div>
</article>
<div class="details-border"></div>

<div class="details-social-area">
    <div class="tags-area">
        <ul>
            <li class="text">Теги:</li>

            @foreach (var tag in post.PostTags.Select(x => x.Tag))
            {
                <li>
                    <a asp-controller="Blog"
                       asp-action="Index"
                       asp-route-tag="@tag.Slug">
                        @tag.Name
                    </a>
                </li>
            }
        </ul>
    </div>
</div>
<div class="details-border"></div>

<div class="heading1">
    <h3>Коментарі (@post.Comments.Count)</h3>
</div>
<div class="commnet-boxs">
    @foreach (var comment in post.Comments.OrderByDescending(x => x.CreatedAt))
    {
        <div class="commnet-box">
            <div class="top-area">
                <div class="author-area">
                    <div class="heading1">
                        <h5>
                            <a href="#">
                                @comment.FirstName @comment.LastName
                            </a>
                        </h5>
                        <p>@comment.CreatedAt.ToString("dd.MM.yyyy")</p>
                    </div>
                </div>
            </div>

            <div class="space20"></div>
            <div class="heading1">
                <p>@comment.Message</p>
            </div>
        </div>
    }
```

```
            </div>

            <div class="details-contact-area">
               <div class="heading">
                  <h4>Leave a Reply</h4>
                  <div class="space10"></div>
                  <p>Provide clear contact information, including phone number, email, and address.</p>
               </div>
               <form asp-controller="Blog"
                     asp-action="AddComment"
                     method="post">

                  @Html.AntiForgeryToken()

                  <input type="hidden" name="PostId" value="@post.Id" />
                  <input type="hidden" name="slug" value="@post.UrlSlug" />

                  <div class="row">
                     <div class="col-md-6">
                        <div class="single-input">
                           <input type="text" name="FirstName" placeholder="Ім’я" required>
                        </div>
                     </div>

                     <div class="col-md-6">
                        <div class="single-input">
                           <input type="text" name="LastName" placeholder="Прізвище">
                        </div>
                     </div>

                     <div class="col-md-6">
                        <div class="single-input">
                           <input type="email" name="Email" placeholder="Email" required>
                        </div>
                     </div>

                     <div class="col-md-6">
                        <div class="single-input">
                           <input type="text" name="Phone" placeholder="Телефон">
                        </div>
                     </div>

                     <div class="col-md-12">
                        <div class="single-input">
                           <input type="text" name="Subject" placeholder="Тема">
                        </div>
                     </div>

                     <div class="col-md-12">
                        <div class="single-input">
                           <textarea name="Message" rows="5" placeholder="Коментар" required></textarea>
                        </div>

                        <div class="space30"></div>

                        <div class="button">
                           <button type="submit" class="theme-btn1">
                              Надіслати
                              <span class="arrow">
                                 <i class="fa-solid fa-arrow-right"></i>
                              </span>
                           </button>
                        </div>
                     </div>
                  </div>
               </form>
            </div>

         </div>

      </div>
   </div>
</div>
```