БАКАЛАВРСЬКА РОБОТА

БР. ІП - 37.00.00.000 ПЗ

Група ІП-23-1К

Гараздюк Тетяна

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Гараздюк Тетяна Любомирівна

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка ВЕБ-базованого рішення ведення проектів

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Гараздюк Т.Л.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Романишин Юлія Любомирівна, д.п.н., професор

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Гараздюк Тетяні Любомирівні

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Розробка ВЕБ-базованого рішення ведення проектів ”

керівник проекту (роботи) Романишин Ю.Л.

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області розробки програмних рішень ведення і управління проектами
2. Системні вимоги для системи відстеження проектів
3. Проектування архітектури та макетів інтерфейсу системи відстеження та ведення проектів
4. Проектування бази даних системи
5. Програмна реалізація системи відстеження та ведення проектів

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

1. Концепція інтеграції веб та мобільних пристроїв (рис. 1.1)
2. Архітектура інтегрованої системи відстеження проектів (рис. 1.2)
3. Етапи водоспадної моделі розробки ПЗ (рис. 1.3)
4. Діаграма варіантів використання управління робочими процесами (рис. 1.4)
5. Діаграма варіантів використання управління проектами (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області розробки програмних рішень ведення і управління проектами	03.05.2025	виконано
2	Системні вимоги для системи відстеження проектів	15.05.2025	виконано
3	Проектування архітектури та макетів інтерфейсу системи відстеження та ведення проектів	25.05.2025	виконано
4	Проектування бази даних системи	01.06.2025	виконано
5	Програмна реалізація системи відстеження та ведення проектів	05.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 80 сторінок, 71 рисунок, список використаних джерел із 31 найменуванням.

Мета роботи: створення повнофункціонального веб-додатку для управління проєктами, що відповідає сучасним вимогам щодо гнучкості, продуктивності, безпеки та зручності використання.

Об'єктом дослідження є процеси планування, моніторингу та контролю за виконанням проєктних завдань, а предметом – методи, засоби і технології створення веб-базованих рішень для управління проєктами.

Предмет дослідження - методи, моделі, інструменти та програмні рішення для створення адаптивної системи відстеження та ведення проєктів.

В першому розділі проведено глибокий аналіз сучасних підходів до управління проєктами, розглянуто вимоги, архітектуру, адаптивність систем та обґрунтовано необхідність створення нового рішення

В другому розділі розроблено логічну архітектуру системи, базу даних і користувацький інтерфейс, що забезпечують ефективне управління проєктами на всіх етапах

В третьому розділі реалізовано повноцінну трирівневу програмну систему, проведено тестування функціоналу, що підтверджує працездатність та надійність рішення

Висновок: розроблена система дозволяє ефективно координувати проєкти в командній роботі, відстежувати прогрес виконання завдань і автоматизувати основні управлінські процеси.

КЛЮЧОВІ СЛОВА: СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ, ВЕБ-ДОДАТОК, АДАПТИВНІСТЬ, MVC, .NET MAUI, АРХІТЕКТУРА ПЗ, БАЗА ДАНИХ, ТЕСТУВАННЯ, МОБІЛЬНИЙ ДОДАТОК.

ANNOTATION

The bachelor's thesis consists of 80 pages, 71 figures, and a reference list with 31 sources.

Objective: To develop a fully functional web application for project management that meets modern requirements for flexibility, performance, security, and usability.

Research Object: The processes of planning, monitoring, and controlling the execution of project tasks.

Research Subject: Methods, tools, and technologies for creating web-based solutions for project management.

Subject of Study: Methods, models, tools, and software solutions for developing an adaptive system for project tracking and management.

Chapter 1: Conducted an in-depth analysis of modern project management approaches, reviewed requirements, system architecture, adaptability, and justified the need for a new solution.

Chapter 2: Developed the logical architecture of the system, database, and user interface to ensure effective project management at all stages.

Chapter 3: Implemented a complete three-tier software system and conducted functionality testing, confirming the solution's operability and reliability.

Conclusion: The developed system enables efficient project coordination in teamwork, task progress tracking, and automation of key management processes.

KEYWORDS: PROJECT MANAGEMENT SYSTEM, WEB APPLICATION, ADAPTABILITY, MVC, .NET MAUI, SOFTWARE ARCHITECTURE, DATABASE, TESTING, MOBILE APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CRUD – Create, Read, Update, Delete (Створити, Прочитати, Оновити, Видалити)

DAO – Data Access Object (Об'єкт доступу до даних)

IDE – Integrated Development Environment (Інтегроване середовище розробки)

iOS – iPhone Operating System (Операційна система iPhone)

MAUI – Multi-platform App UI (Багатоплатформний інтерфейс застосунків)

MVC – Model-View-Controller (Модель-Представлення-Контролер)

SSMS – SQL Server Management Studio (Студія управління SQL Server)

SSO – Single Sign-On (Єдиний вхід)

UML – Unified Modeling Language (Уніфікована мова моделювання)

XAML – eXtensible Application Markup Language (Розширювана мова розмітки застосунків)

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій та глобалізації бізнес-процесів ефективне управління проєктами є одним із ключових чинників успішної діяльності підприємств та організацій. Водночас традиційні методи контролю за виконанням завдань втрачають актуальність через зростаючу складність проєктів, динамічність змін і потребу в гнучких адаптивних інструментах. Веб-базовані системи управління проєктами дають змогу централізовано координувати завдання, відстежувати прогрес і забезпечувати прозорість комунікацій у командах незалежно від їхнього фізичного розташування.

Особливої важливості набувають рішення, що поєднують адаптивність, масштабованість і кросплатформеність, а також мають зручний інтерфейс і підтримують інтеграцію з іншими сервісами. Саме така система пропонується у цій роботі, що обумовлює її практичну значущість і актуальність у контексті цифрової трансформації управлінських процесів.

Актуальність роботи

Проблематика управління проєктами з кожним роком набуває все більшої актуальності, особливо в умовах гібридної роботи, дистанційного менеджменту та глобальної конкуренції. Багато існуючих систем не враховують усі індивідуальні потреби організацій, особливо середнього та малого бізнесу. Дослідження та розробка адаптивної системи, що дозволяє гнучко моделювати процеси, легко адаптується під користувача та має інтуїтивно зрозумілий інтерфейс, є відповіддю на виклики часу. Саме такі інструменти здатні суттєво підвищити ефективність виконання проєктів, зменшити кількість помилок і забезпечити проактивний контроль.

Управління проєктами в умовах сучасного ІТ-середовища вимагає використання інноваційних підходів і цифрових інструментів, що дозволяють ефективно координувати завдання, контролювати хід їх виконання,

забезпечувати комунікацію між учасниками команди та оперативно реагувати на зміни. З огляду на це, розробка веб-базованої адаптивної системи для ведення проєктів є актуальною задачею, яка поєднує в собі практичні й наукові аспекти.

Метою даної дипломної роботи є створення повнофункціонального веб-додатку для управління проєктами, що відповідає сучасним вимогам щодо гнучкості, продуктивності, безпеки та зручності використання.

Об'єктом дослідження є процеси планування, моніторингу та контролю за виконанням проєктних завдань, а предметом – методи, засоби і технології створення веб-базованих рішень для управління проєктами.

Предмет дослідження - методи, моделі, інструменти та програмні рішення для створення адаптивної системи відстеження та ведення проєктів.

Завдання дослідження

1. Провести аналіз предметної області управління проєктами та існуючих програмних рішень.

2. Визначити вимоги до функціональності, архітектури та інтерфейсу майбутньої системи.

3. Розробити архітектуру програмного рішення та бази даних.

4. Створити макети користувацького інтерфейсу для веб- та мобільної версії.

5. Реалізувати програмне рішення з використанням сучасних технологій (MVC, .NET MAUI).

6. Провести тестування системи для перевірки її функціональності та надійності.

Методи дослідження

В роботі використано метод аналізу та синтезу, функціональне та об'єктно-орієнтоване моделювання, метод порівняльного аналізу аналогів, проєктування баз даних, методи програмної реалізації (MVC, Web API, .NET MAUI), методи тестування (модульне, інтеграційне).

Наукова новизна

Запропоновано адаптивну модульну архітектуру системи відстеження проєктів, яка поєднує веб- та мобільні технології, підтримує масштабування, а також дозволяє ефективно керувати робочими процесами завдяки візуалізації даних, ролям користувачів та зручному інтерфейсу.

Практичне застосування

Система може бути впроваджена у будь-якій ІТ-компанії, освітньому закладі чи стартапі для автоматизації управління проєктами, планування, відстеження завдань та взаємодії між членами команди.

У рамках роботи проведено аналіз предметної області, сформульовано технічні та функціональні вимоги до системи, спроектовано архітектуру, інтерфейси та базу даних, реалізовано як веб-, так і мобільну версію продукту із застосуванням сучасних фреймворків і технологій. Завершальним етапом стала перевірка працездатності системи шляхом тестування її окремих компонентів і функціоналу загалом.

Бакалаврська робота містить 80 сторінок, 71 рисунок, 3 розділи список використаних джерел із 31 найменуванням.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ПРОГРАМНИХ РІШЕНЬ ВЕДЕННЯ І УПРАВЛІННЯ ПРОЕКТАМИ

1.1. Особливості розроблюваного додатку управління проектами

Пропонований веб-додаток, що функціонує на основі браузера, є комплексним рішенням для ефективного управління проектами. Він забезпечує всебічний доступ до інформації про проект, підтримує сучасні засоби комунікації та надає інструменти для оптимізації робочих процесів з будь-якого пристрою.

Завдяки інтуїтивно зрозумілому інтерфейсу, система сприяє співпраці в реальному часі, дозволяючи користувачам делегувати завдання, відстежувати прогрес та координувати діяльність незалежно від їхнього місцезнаходження. Це значно спрощує управління проектами, підвищуючи їхню гнучкість та оперативність.

Ключовою особливістю цього рішення є наявність спеціалізованого програмного інтерфейсу (API), призначеного для мобільних пристроїв. Це забезпечує безперебійну інтеграцію та двосторонню синхронізацію даних між основним веб-додатком та нативними мобільними додатками, розробленими для різних платформ. Така архітектура гарантує узгоджений та інтегрований користувацький досвід, дозволяючи здійснювати оперативне оновлення статусу завдань та миттєвий обмін інформацією на всіх підключених пристроях.

Приділяється особливу увагу безпеці даних, використовуючи надійні протоколи шифрування та ефективні механізми аутентифікації. Це гарантує захист конфіденційної інформації та приватності користувачів, мінімізуючи ризики несанкціонованого доступу та витоку даних.

Ця архітектура, що поєднує гнучкість веб-платформи з перевагами мобільної інтеграції та високим рівнем безпеки, робить даний додаток потужним інструментом для сучасного управління проектами в умовах розподілених команд та динамічного бізнес-середовища.

1.2. Оптимізація управління проектами за допомогою адаптивної системи відстеження

Ефективна система відстеження проектів є критично важливою для успішного функціонування будь-якої організації, оскільки вона забезпечує оновлення інформації в реальному часі щодо прогресу, розподілу ресурсів та доступності команд. Це значно підвищує як операційну ефективність, так і рівень підзвітності. Така система також сприяє стратегічному прийняттю рішень, надаючи керівництву повну картину всіх поточних проектів та залучених ресурсів. Це, своєю чергою, дозволяє оптимізувати розподіл ресурсів, встановлювати пріоритети завдань та оперативно реагувати на потенційні затримки або перевищення бюджету. Таким чином, сучасна система відстеження проектів є фундаментальним інструментом для підтримання високих стандартів безпеки, ефективності та підзвітності у роботі.

1.2.1. Модульні робочі процеси та їх переваги

Пропонована система управління проектами відрізняється від інших відомих рішень, таких як Jira, Basecamp та Zoho Projects, своєю унікальною здатністю інтегрувати модульні робочі процеси. Ці робочі процеси можуть бути спеціально налаштовані для потреб різних відділів, забезпечуючи високий рівень адаптивності. Робочий процес визначається як серія послідовних етапів, що представляють сукупність завдань, необхідних для завершення проекту. Він слугує дорожньою картою, що відображає прогрес

роботи від її ініціації до завершення, гарантуючи виконання всіх завдань у правильній послідовності.

Наприклад, для відділу інформаційних технологій (IT) робочий процес команди розробки програмного забезпечення може включати такі етапи, як "Очікування", "Дизайн", "Розробка", "Тестування" та "Виробництво". Натомість, робочий процес команди управління серверами може складатися з етапів "Очікування", "Надання", "Розгортання" та "Моніторинг". Такі гнучкі робочі процеси забезпечують чітку візуалізацію прогресу проекту, гарантуючи, що кожен етап належним чином завершений перед переходом до наступного.

1.2.2. Адаптивність у порівнянні з аналогами

На відміну від цього, більшість популярних програмних засобів управління проектами, включаючи Jira, Basecamp та Zoho Projects, зазвичай не пропонують такого рівня деталізованого, налаштовуваного управління робочими процесами для різних типів проектів "з коробки". Рівень адаптивності, притаманний цій системі управління проектами, дозволяє кожній команді в організації – включаючи IT, мережеві відділи, диспетчерські центри та інші – працювати з системою, що оптимізована під їхні специфічні потреби, при цьому залишаючись ефективно інтегрованою з іншими підрозділами.

1.2.3. Мета системи

Основною метою даної системи є моніторинг прогресу проекту, оптимізація розподілу ресурсів, визначення та відстеження термінів виконання, а також забезпечення відповідності цілей проекту стратегічним цілям організації. Система сприяє ефективній комунікації між членами команди, надає оновлення щодо прогресу проекту в реальному часі та допомагає виявляти потенційні проблеми або ризики, що можуть спричинити

затримки. Зосереджуючись на підвищенні відповідальності та покращенні процесу прийняття рішень шляхом централізації всієї інформації, пов'язаної з проектом, дана система відстеження проектів зрештою спрямована на успішне та своєчасне завершення проектів.

1.3. Архітектура рішення для відстеження ведення проектів

Запропонована система відстеження проектів є гібридною програмною платформою, що об'єднує функціональність веб- та мобільних додатків. Її розробка передбачає використання технології .NET MAUI для мобільної компоненти, а також ASP.NET MVC та Web API для веб-додатку. Такий технологічний стек забезпечить миттєвий обмін даними та гарантуватиме крос-платформну сумісність з різними типами пристроїв, включаючи смартфони, планшети та персональні комп'ютери.

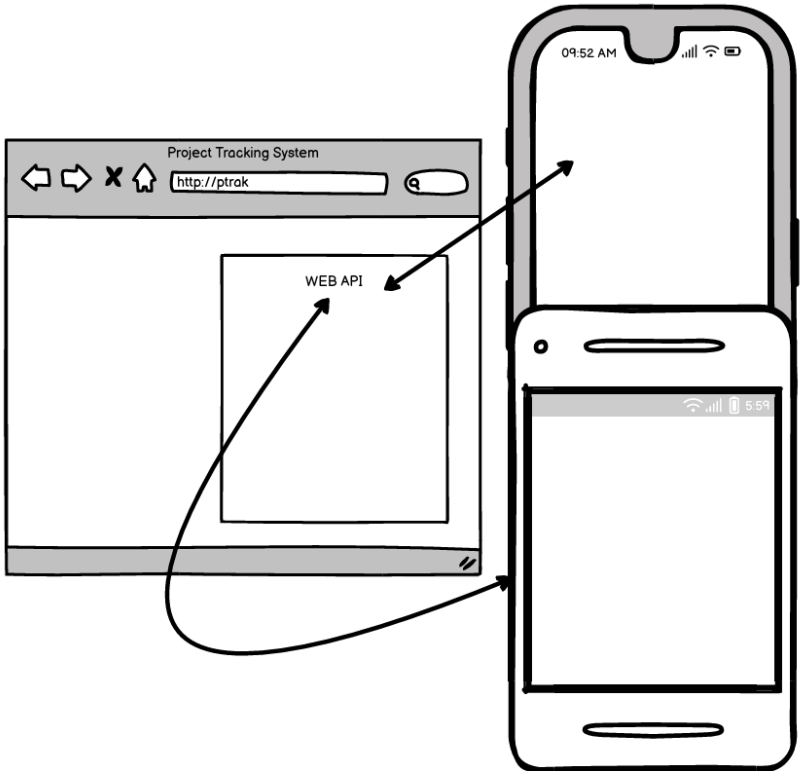

Рисунок 1.1 - Концепція інтеграції веб та мобільних пристроїв

Рисунок 1.1 ілюструє концептуальну модель системи відстеження проектів, де веб- та мобільні додатки взаємодіють через Web API для забезпечення безперебійного обміну даними.

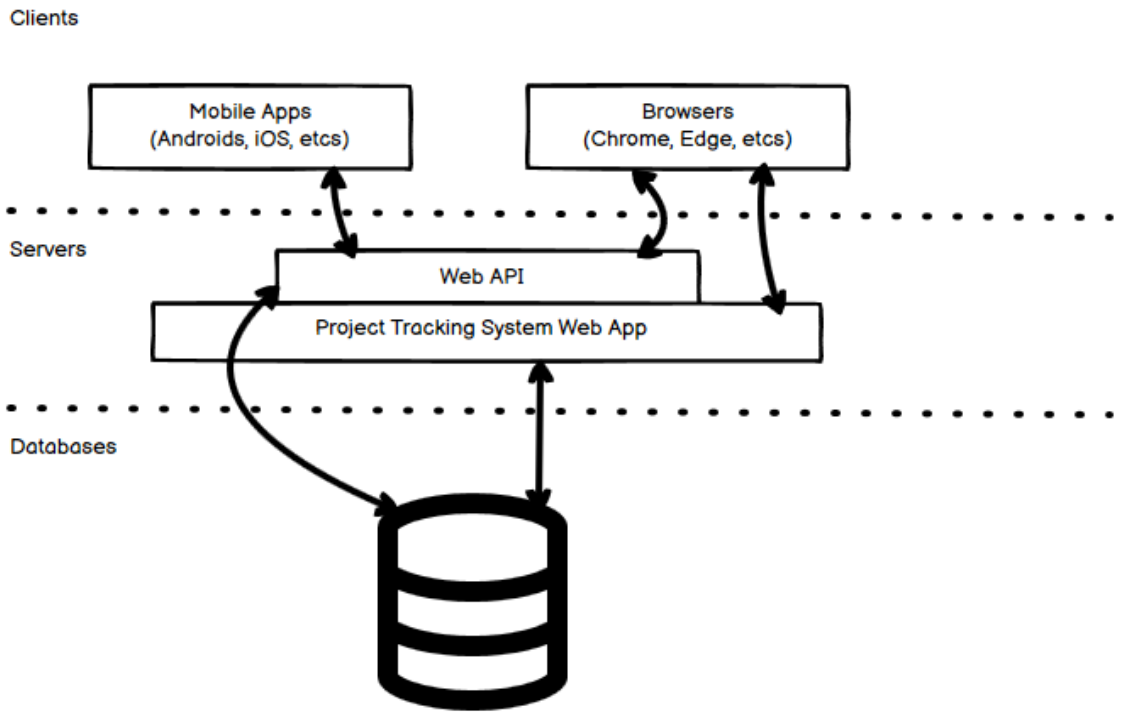

Рисунок 1.2 - Архітектура інтегрованої системи відстеження проектів

Рисунок 1.2 деталізує цю архітектуру, показуючи, як клієнтські додатки – зокрема додатки для Android та iOS (iPhone Operating System), а також веб-браузери Chrome та Edge – використовують Web API для взаємодії з центральним сервером. На сервері розташована база даних, яка зберігає всі дані проектів, забезпечуючи єдину точку доступу до інформації. Така структура дозволяє системі синхронізувати дані між різними типами клієнтів у реальному часі, роблячи будь-які зміни в проектах миттєво доступними для всіх користувачів. Цей дизайн забезпечує гнучкі та масштабовані міжплатформні взаємодії, що є критично важливим для ефективного та сучасного управління проектами.

1.4. Технічні вимоги до розробки системи відстеження ведення проектів

Для реалізації запропонованої гібридної платформи відстеження проектів визначено наступні вимоги до програмного та апаратного забезпечення.

1.4.1. Вимоги до програмного забезпечення

Середовища Розробки та Технології:

- .NET MAUI: Призначений для розробки крос-платформних мобільних додатків. Використовуватиметься .NET 8 з підтримкою платформ iOS, Android, macOS та Windows.

- ASP.NET MVC 5: Застосовуватиметься для обробки веб-API та реалізації бекенд-логіки на основі .NET Framework.

- Visual Studio 2022: Інтегроване середовище розробки (IDE), що є основним інструментом для розробки .NET додатків. Воно забезпечує повну підтримку .NET MAUI та MVC 5, включаючи вбудовані шаблони та інструменти налагодження.

- SQL Server: Використовуватиметься як сервер баз даних для управління даними додатка. Рекомендована версія – SQL Server 2022.

- SQL Server Management Studio (SSMS): Графічний інтерфейс для проектування, адміністрування та управління базами даних SQL Server. Рекомендується остання версія.

Тепер розглянемо стек програмного забезпечення.

Мови програмування:

- C#: Основна мова для розробки як .NET MAUI, так і MVC 5.

- HTML, CSS, JavaScript: Використовуватимуться для розробки фронтенду веб-додатку.

Фреймворки та Бібліотеки:

- .NET MAUI: Для розробки мобільного клієнта.

- jQuery та Bootstrap: Для фронтенд-скриптів та стилізації веб-інтерфейсу.

API та Сервіси:

- RESTful API: Реалізований за допомогою ASP.NET Web API для взаємодії між клієнтськими та серверними компонентами.

Безпека:

- HTTPS: Усі комунікації між мобільним клієнтом, веб-клієнтом та сервером будуть захищені за допомогою HTTPS для запобігання несанкціонованому перехопленню даних.

- TLS (Transport Layer Security): HTTPS забезпечує шифрування всіх даних, що передаються через мережу, гарантуючи їх конфіденційність та цілісність.

Захист даних. Токени та конфіденційні дані шифруватимуться під час передачі та, за необхідності, у стані спокою, з використанням стандартних криптографічних протоколів.

1.4.2. Вимоги до апаратного забезпечення

Розробка буде переважно здійснюватися на платформі Windows, з використанням Mac Mini для запуску емулятора iOS.

Апаратне забезпечення для розробки.

Персональний комп'ютер на базі Windows:

- Операційна система: Windows 10 або новіша.

- Процесор: Intel Core i5 або еквівалентний/кращий.

- Оперативна пам'ять: мінімум 8 ГБ.

- Накопичувач: SSD з об'ємом щонайменше 256 ГБ.

Повинен мати достатню продуктивність для одночасного запуску Visual Studio та кількох емуляторів/симуляторів.

Mac Mini (для розробки iOS):

- Процесор: Intel-процесор або Apple Silicon (M1/M2) або кращий.

- Оперативна пам'ять: мінімум 8 ГБ.

- Версія macOS: macOS Monterey або новіша.

- Необхідний для запуску симуляторів iOS через Visual Studio.

Опційні тестові пристрої:

- Веб-додаток: Будь-яка операційна система, що підтримує сучасні веб-браузери.

- iOS: iPhone 12 та пізніші моделі можуть використовуватися для додаткового тестування.

Серверне апаратне забезпечення

Сервер (віртуальна машина або виділений сервер):

- Процесор: Чотириядерний Xeon або еквівалентний/кращий.

- Оперативна пам'ять: мінімум 16 ГБ.

- Накопичувач: 256 ГБ SSD для основного сховища.

- Мережа: високошвидкісне, надійне інтернет-з'єднання для хостингу сервісів.

- Операційна система: Windows Server 2019 або новіша.

1.5. Методологія розробки та системні вимоги для системи відстеження проектів

У контексті розробки даного проекту одним розробником, застосування методології водоспаду є оптимальним підходом, що забезпечує послідовний та структурований процес. Перевага цієї стратегії полягає в її здатності сприяти ретельному плануванню та належному виконанню на кожному етапі, що мінімізує ймовірність дублювання завдань та необхідність переробки. Крім того, водоспадна модель дозволяє сконцентруватися на одному етапі розробки за раз, що є вкрай важливим для ефективного

управління різними аспектами проекту. Процес включатиме етапи збору вимог, аналізу та дизайну, а також розробки прототипу.

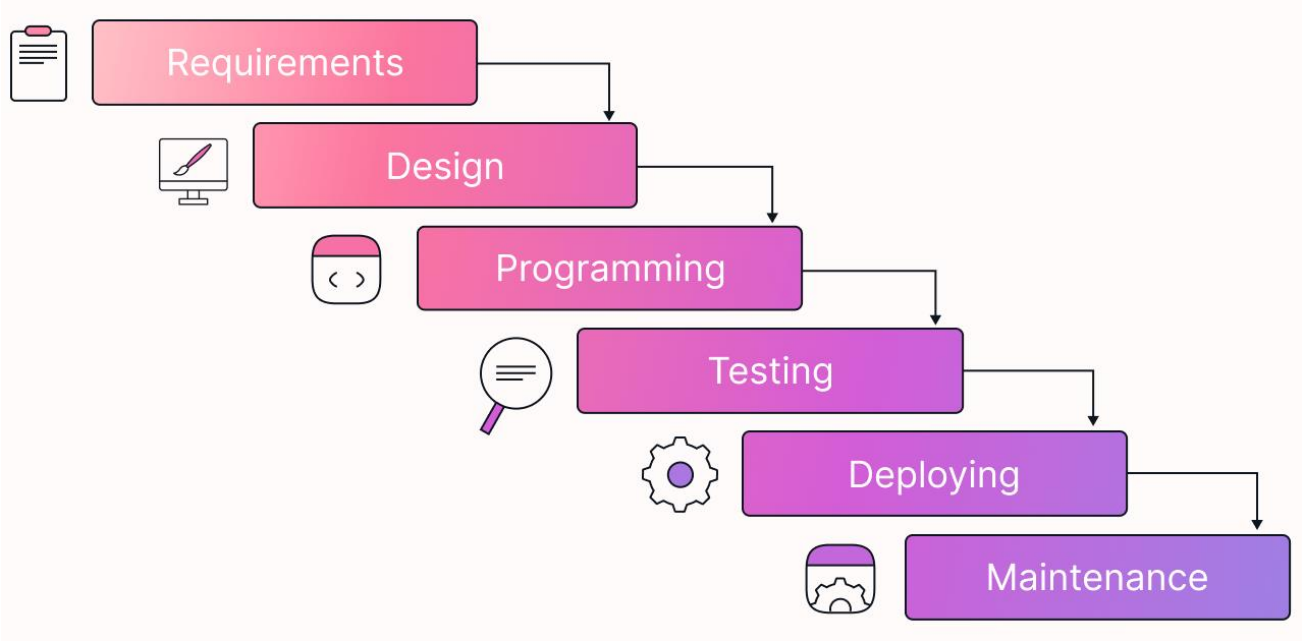

Рисунок 1.3 – Етапи водоспадної моделі розробки ПЗ

1.5.1. Збір вимог

Етап збору вимог є критично важливим для успішної розробки програмного забезпечення. Початкова версія системи відстеження проектів буде орієнтована на потреби команди розробки програмного забезпечення, з робочим процесом, спеціально адаптованим до процесів життєвого циклу розробки програмного забезпечення.

Система буде організовувати завдання в межах проектів, об'єднаних у робочі процеси, кожен з яких матиме власну ієрархію та структуру для підвищення ефективності управління проектами.

Передбачено підтримку кількох ролей користувачів:

-адміністратор,

- менеджер,

- член команди,

- зацікавлена сторона,

- системний користувач.

Під зацікавленою стороною розуміється будь-яка особа, що має інтерес до проекту або здійснює моніторинг його прогресу та результатів. Кожна роль матиме специфічні права доступу, що забезпечуватимуть відповідні взаємодії з системою. Ключові функціональні можливості включатимуть управління завданнями та проектами (специфічне для ролі), встановлення пріоритетів, оновлення статусу прогресу та функцію коментування для інтерактивної комунікації щодо завдань та проектів.

Система буде доступна як через веб-інтерфейс браузера, так і через мобільні додатки, що гарантує безперебійну інтеграцію та швидку синхронізацію даних за допомогою надійного Web API.

1.5.2. Аналіз та дизайн системи. Ролі користувачів

Основна функціональність системи базується на концепції робочих процесів. Кожен робочий процес є набором проектів, а кожен проект, своєю чергою, містить кілька завдань. Ця ієрархічна структура дозволяє організовано керувати проектами та завданнями, забезпечуючи чіткий нагляд та ефективне виконання.

Система включатиме наступні ролі користувачів, кожна з яких має унікальні обов'язки та рівні доступу:

- Адміністратор: Має повний контроль над системою. Адміністратори можуть створювати та керувати всіма робочими процесами, проектами та завданнями. Вони також мають право призначати ролі та керувати доступом для всіх інших користувачів.

- Менеджер: Відповідає за нагляд за всіма проектами. Менеджери можуть створювати проекти, призначати завдання членам команди та відстежувати прогрес. Вони також мають право змінювати деталі та пріоритети завдань.

- Член команди: Виконує призначені йому проектні завдання. Може оновлювати статус та деталі своїх завдань.

- Зацікавлена сторона: Має доступ лише для читання до визначених проектів та робочих процесів. Зацікавлені сторони можуть переглядати прогрес та отримувати оновлення, але не можуть вносити зміни до деталей проекту або завдання.

- Системний користувач: Неактивний обліковий запис користувача, що використовується програмно для автоматизації. Він не пов'язаний з реальними особами і виконує системні завдання, такі як пакетні завдання та обслуговування, без стандартних можливостей входу, забезпечуючи безпечну роботу.

1.5.3. Функціональні вимоги

- Управління робочими процесами: Адміністратори можуть створювати нові та змінювати існуючі робочі процеси. Робочі процеси служать великими контейнерами для проектів. Усі користувачі можуть переглядати робочі процеси, до яких вони мають доступ, відповідно до їхньої ролі та встановлених адміністратором прав.

- Управління проектами: Менеджери можуть створювати проекти, встановлювати їхні пріоритети та призначати членів команди. Проекти можуть бути оновлені менеджерами та членами команди з відповідними ролями.

- Управління завданнями: У межах кожного проекту менеджери та члени команди з достатніми привілеями можуть створювати або оновлювати завдання. Члени команди, призначені до завдання, можуть оновлювати його статус (наприклад, "До виконання", "Виконується", "Виконано") та надавати детальні звіти про прогрес.

- Управління ролями: Адміністратори визначають обов'язки користувачів та надають їм відповідні ролі.

- Система коментарів: Користувачі можуть додавати коментарі як до проектів, так і до завдань, що сприяє спілкуванню та співпраці між членами команди та зацікавленими сторонами в реальному часі.

1.5.4. Нефункціональні вимоги

- Зручність використання (Usability): Інтерфейс веб-додатку буде розроблений інтуїтивно зрозумілим та простим у використанні, що дозволить користувачам будь-якого рівня технічної підготовки ефективно взаємодіяти з системою.

- Продуктивність (Performance): Система буде налаштована для забезпечення високої продуктивності, з швидкими часами завантаження та ефективною обробкою даних. Це дозволить підтримувати роботу багатьох користувачів та великих обсягів даних без затримок.

- Безпека (Security): Захист конфіденційних даних та запобігання несанкціонованому доступу є пріоритетом. Це включатиме надійну аутентифікацію користувачів, шифрування даних та безпечні з'єднання API.

- Масштабованість (Scalability): Система буде розроблена з урахуванням можливості масштабування, що дозволить їй адаптуватися до зростаючої кількості користувачів, проектів та завдань без зниження продуктивності.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА МАКЕТІВ ІНТЕРФЕЙСУ СИСТЕМИ ВІДСТЕЖЕННЯ ТА ВЕДЕННЯ ПРОЕКТІВ

2.1. Моделювання архітектури системи за допомогою діаграм варіантів використання

Для візуального представлення, оцінки та вдосконалення запропонованих процесів у системі використовується метод моделювання. Цей підхід полегшує розуміння функціонування системи та її взаємодій, сприяючи чіткій комунікації між розробниками, дизайнерами та зацікавленими сторонами. У даному проекті для моделювання системи застосовується Уніфікована мова моделювання (UML), зокрема діаграми варіантів використання та діаграми послідовностей.

У цій системі традиційні функції видалення даних здебільшого замінені на використання прапора "активний" для всіх сутностей. Замість безповоротного видалення даних, відбувається зміна їхнього статусу активності, що запобігає незворотній втраті інформації та усуває потребу в процесах відновлення даних. Такий підхід забезпечує збереження повного історичного запису, що є критично важливим для комплексного звітування, аналізу та забезпечення відповідності юридичним нормам.

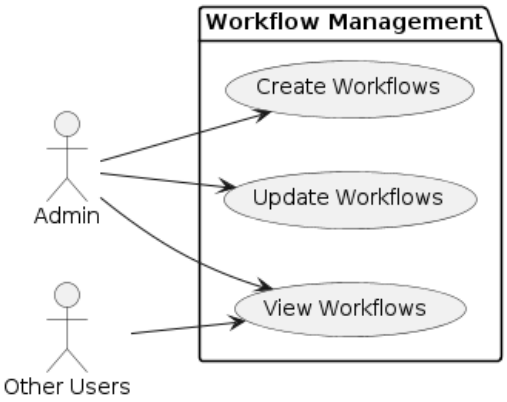

Рисунок 2.1 - Діаграма варіантів використання управління робочими процесами

Рисунок 2.1 ілюструє ролі адміністраторів та інших користувачів (менеджерів, членів команди та зацікавлених сторін) у контексті управління робочими процесами. Адміністратор має повноваження створювати, оновлювати та переглядати робочі процеси, тоді як інші користувачі обмежені лише функцією перегляду.

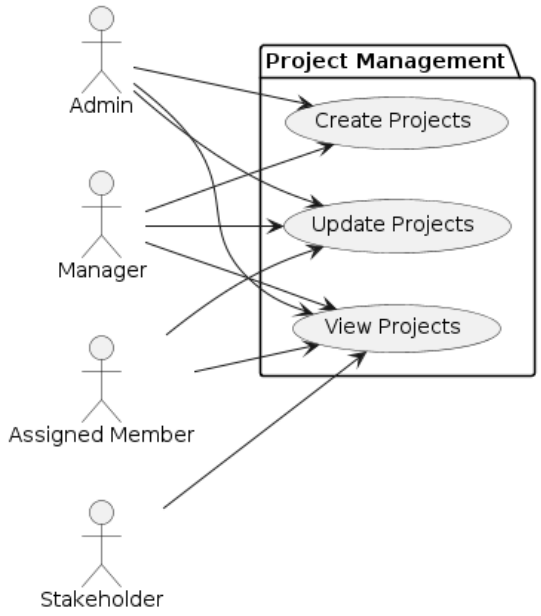

Рисунок 2.2 - Діаграма варіантів використання управління проектами

Рисунок 2.2 демонструє управління проектами. Адміністратор та менеджер можуть створювати, оновлювати та переглядати проекти, тоді як призначені члени команди мають можливість оновлювати та переглядати проекти. Зацікавлені сторони можуть лише переглядати проектну інформацію.

Рисунок 2.3 відображає функціональність адміністратора щодо управління пріоритетами проектів, включаючи їх створення, перегляд та оновлення. Приклади рівнів пріоритету включають "Низький", "Середній", "Високий" та інші класифікації.

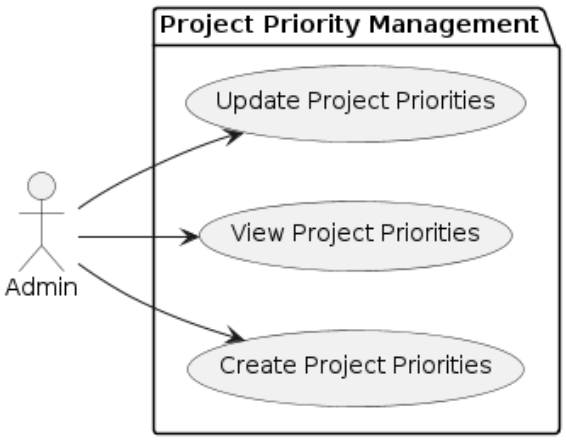

Рисунок 2.3 - Діаграма варіантів використання пріоритету проекту

Рисунок 2.4 ілюструє, що як адміністратори, так і менеджери, а також призначені члени команди, можуть створювати, оновлювати та переглядати завдання. Інші зацікавлені сторони обмежені лише переглядом завдань.

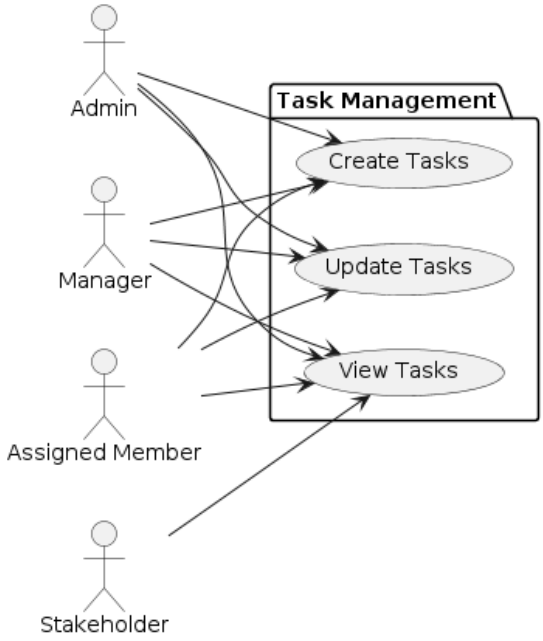

Рисунок 2.4 - Діаграма варіантів використання управління завданнями

Рисунок 2.5 показує, що адміністратор керує статусами завдань, створюючи, переглядаючи та оновлюючи їх. Приклади статусів завдань включають "До виконання", "В процесі", "Завершено" та інші стани.

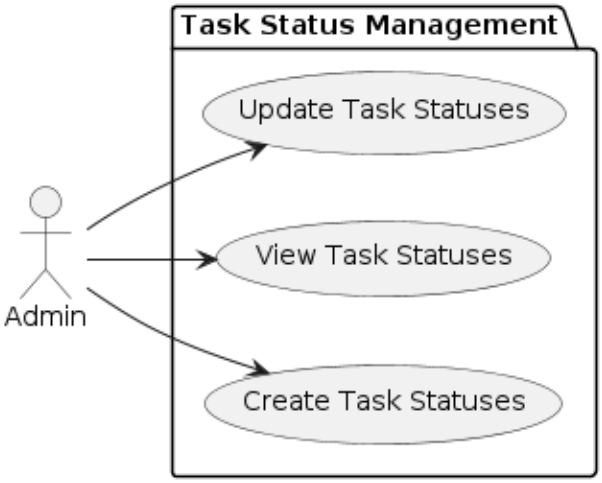

Рисунок 2.5 - Діаграма варіантів використання статусу завдання

Рисунок 2.6 ілюструє можливості адміністратора, які охоплюють призначення ролей користувачам, оновлення описів ролей та перегляд описів ролей кожного користувача.

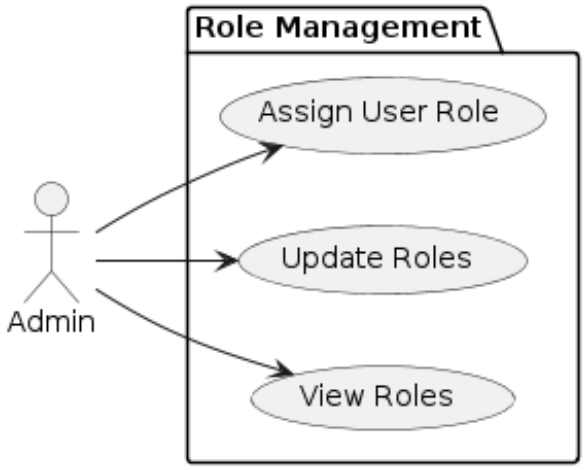

Рисунок 2.6 - Діаграма варіантів використання управління ролями

2.2. Проектування діаграм послідовностей

Рисунок 2.7 відображає послідовність взаємодій під час входу користувача до веб-додатку через його веб-API.

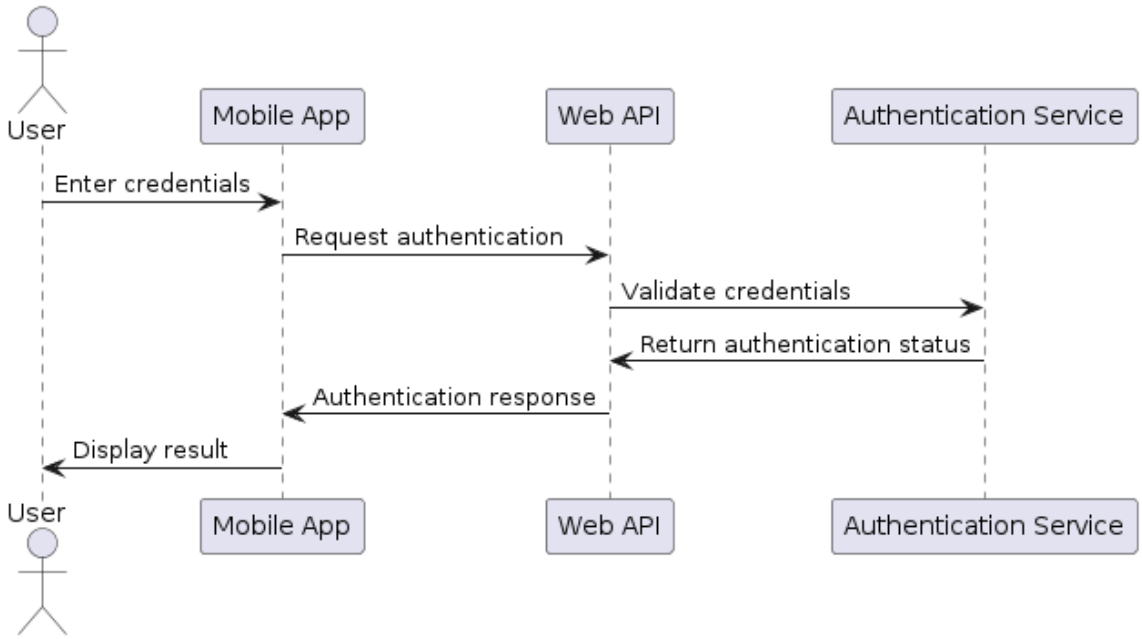

Рисунок 2.7 - Діаграма послідовності входу користувача в мобільний додаток

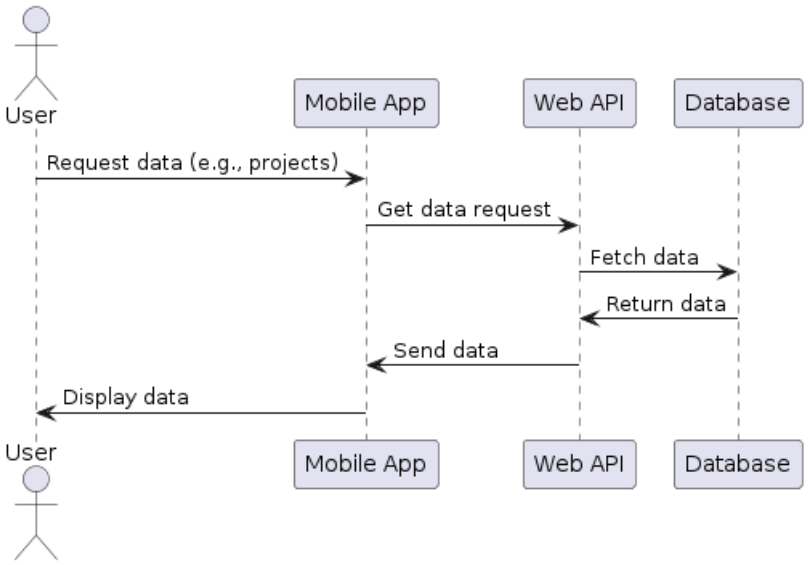

Рисунок 2.8 - Діаграма послідовності запиту даних

Рисунок 2.8 ілюструє покроковий процес отримання даних із сервера за допомогою веб-API додатку та подальшого їх відображення у веб-додатку.

Рисунок 2.9 демонструє процес надсилання коментаря до проекту користувачем через інтерфейс мобільного додатку.

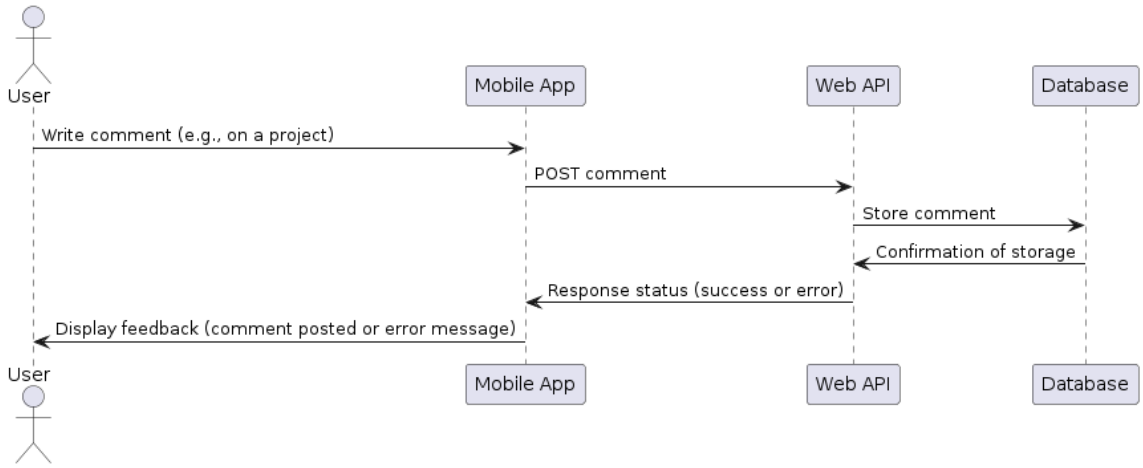

Рисунок 2.9 - Діаграма послідовності відправки даних

Ці діаграми UML забезпечують чітке та стандартизоване представлення функціональних можливостей системи та взаємодії між її компонентами, що є важливим для подальшої розробки та впровадження.

2.3. Проектування бази даних системи відстеження та ведення проектів

Якісне проектування бази даних є фундаментальним для системи відстеження проектів, оскільки воно забезпечує цілісність даних, оптимізує продуктивність системи та спрощує виконання складних запитів. Добре структурована база даних підтримує гнучке масштабування, прискорює пошук даних та полегшує інтеграцію з іншими системами. Це значно покращує управління та моніторинг ресурсів, завдань і проектів. Для цього проекту використовуються Microsoft SQL Server 2022 та інструмент SQL Server Management Studio (SSMS) для створення та адміністрування структури бази даних.

2.3.1. Модель сутностей та відношень

Нижче представлено опис ключових сутностей та їхніх атрибутів, що формують основу архітектури бази даних.

Сутність Workflow

Workflow (робочі процеси) визначають послідовність етапів, через які проходить проект від початку до завершення. Ці етапи можуть відрізнятися для різних відділів організації. Наприклад, для проекту розробки програмного забезпечення робочий процес може включати такі етапи: Очікування (проекти очікують на затвердження та розподіл ресурсів), Дизайн (створення специфічних дизайнів та користувацьких інтерфейсів), Розробка (безпосереднє кодування), Тестування (варіанти тестування для забезпечення відповідності вимогам) та Виробництво (розгортання програмного забезпечення в робочому середовищі).

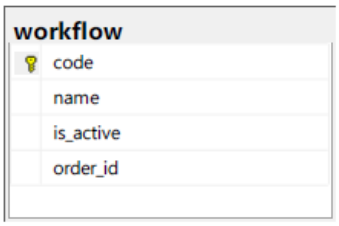

Рисунок 2.10 - Діаграма даних Workflow

Рисунок 2.10 ілюструє структуру таблиці workflow, яка складається з чотирьох стовпців, призначених для моніторингу та категоризації робочих процесів у системі.

Таблиця 2.1 - Опис сутності Workflow

Назва стовпця	Тип даних	Опис
code	Текст	Унікальний код, що представляє робочий процес (до 50 символів).
name	Текст	Назва робочого процесу (до 255 символів).
is_active	Булевий	Булеве значення, що вказує, чи активний робочий процес (1) чи неактивний (0).
order_id	Число	Числове значення, що використовується для визначення порядку робочих процесів.

Сутність Project

Таблиця project зберігає всю ключову інформацію, пов'язану з різними проектами в системі, охоплюючи їхні основні деталі.

Рисунок 2.11 ілюструє взаємозв'язки бази даних між таблицею project та трьома іншими таблицями: priority, workflow та user.

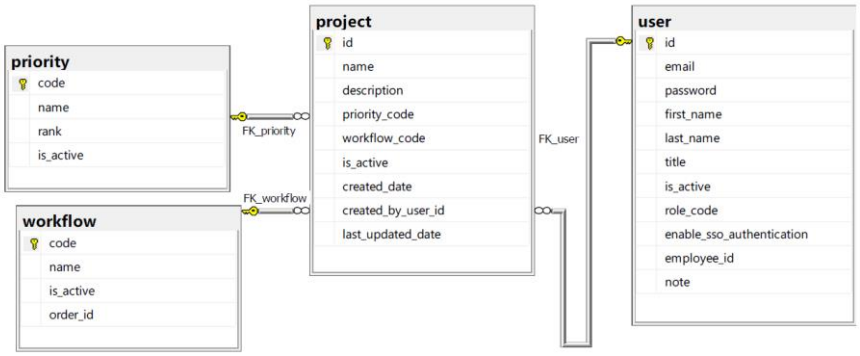

Рисунок 2.11 - Діаграма даних Project

Таблиця 2.2 - Опис сутності Project

Назва стовпця	Тип даних	Опис
id	Число	Унікальний ідентифікатор для кожного проекту (автоінкремент).
name	Текст	Назва проекту.
description	Текст	Детальний опис проекту.
priority_code	Текст	Код, що вказує на пріоритет проекту, пов'язаний з таблицею пріоритетів.
workflow_code	Текст	Код, що представляє статус робочого процесу проекту, пов'язаний з таблицею робочих процесів.
is_active	Булевий	Проект активний (1) чи неактивний (0).
created_date	Дата та час	Дата та час створення проекту.
created_by_user_id	Число	Ідентифікатор користувача, який створив проект.
last_updated_date	Дата та час	Дата та час останнього оновлення проекту.

Сутність Project Comment

Таблиця project_comment призначена для обробки та зберігання коментарів, пов'язаних з конкретними проектами. Вона є важливою для

комунікації та співпраці, дозволяючи членам команди та зацікавленим сторонам залишати нотатки безпосередньо до проектів.

Рисунок 2.12 показує взаємозв'язки бази даних між таблицями project та project_comment.

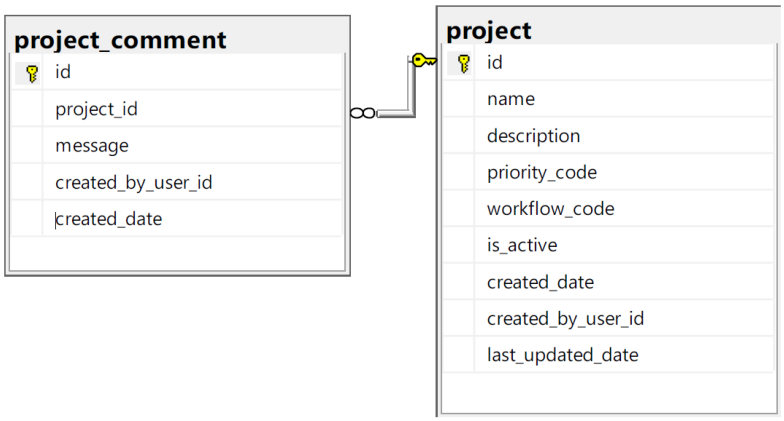

Рисунок 2.12 - Діаграма даних Project Comment

Таблиця 2.3 - Опис сутності Project Comment

Назва стовпця	Тип даних	Опис
Id	Число	Унікальний ідентифікатор для кожного коментаря.
project_id	Число	Ідентифікатор проекту, до якого належить коментар.
message	Текст	Текст коментаря.
created_by_user_id	Число	Ідентифікатор користувача, який створив коментар.
created_date	Дата та час	Дата та час створення коментаря.

Сутність Task

Таблиця task розроблена для ефективного управління та відстеження окремих завдань, пов'язаних з різними проектами.

Рисунок 2.13 показує взаємозв'язки бази даних між таблицями project, task та task_status.

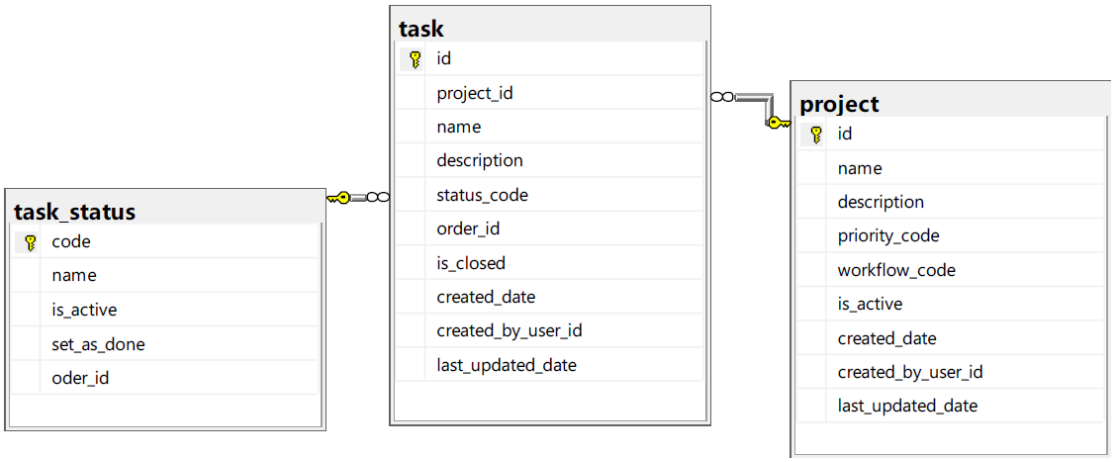

Рисунок 2.13 - Діаграма даних сутності Task

Таблиця 2.4 - Опис сутності Task

Назва стовпця	Тип даних	Опис
id	Число	Унікальний ідентифікатор для кожного завдання.
project_id	Число	Ідентифікатор проекту, до якого належить завдання.
name	Текст	Назва завдання (до 4000 символів).
description	Текст	Детальний опис завдання.
status_code	Текст	Код поточного статусу завдання, пов'язаний з таблицею статусів.
order_id	Число	Числове значення для впорядкування або встановлення пріоритетів завдань.
is_closed	Булевий	Булеве поле, що вказує, чи закрите завдання.
created_date	Дата та час	Дата та час створення завдання.
created_by_user_id	Число	Ідентифікатор користувача, який створив завдання.
last_updated_date	Дата та час	Дата та час останнього оновлення завдання.

Сутність Task Comment

Таблиця task_comment розроблена для ефективного управління та зберігання коментарів, пов'язаних з окремими завданнями. Вона дозволяє зацікавленим сторонам та членам команди залишати коментарі, сприяючи регулярному обміну думками.

Рисунок 2.14 показує взаємозв'язки бази даних між таблицями task та task_comment.

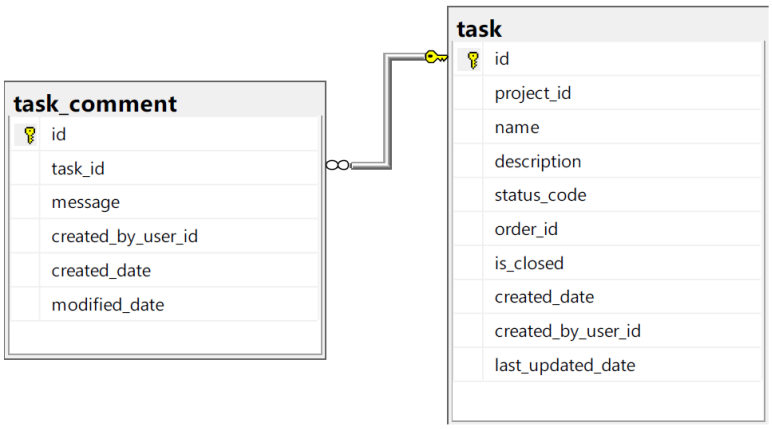

Рисунок 2.14 - Діаграма даних Task Comment

Таблиця 2.5 - Опис сутності Task Comment

Назва стовпця	Тип даних	Опис
id	Число	Унікальний ідентифікатор для кожного коментаря.
Task_id	Число	Пов'язування коментарів з відповідними завданнями.
Message	Текст	Текст коментаря.
Created_by_user_id	Число	Ідентифікатор користувача, який створив коментар.
Created_date	Дата та час	Дата та час створення коментаря.
Modified_date	Дата та час	Дата та час останньої зміни коментаря.

Сутність Task Status

Таблиця task_status налаштована для опису та відстеження різних станів, в яких можуть перебувати завдання, полегшуючи моніторинг їхнього прогресу.

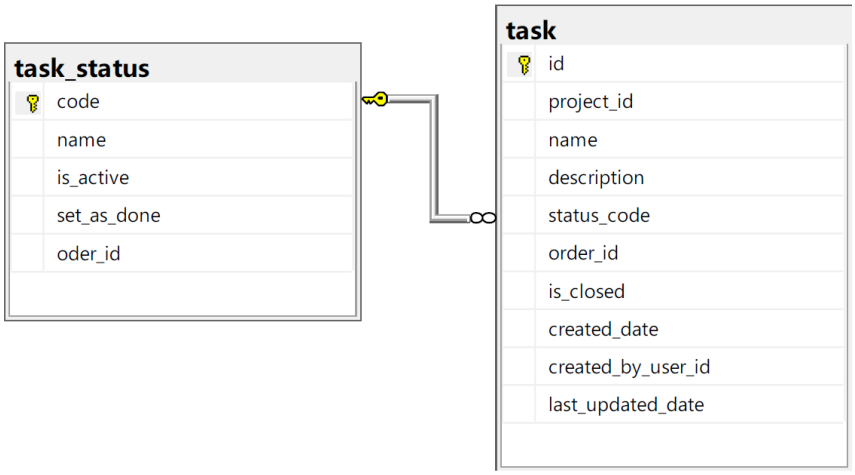

Рисунок 2.15 - Діаграма даних Task Status

Рисунок 2.15 показує взаємозв'язки бази даних між таблицями task та task_status.

Таблиця 2.6 - Опис сутності Task Status

Назва стовпця	Тип даних	Опис
code	Текст	Унікальний код для кожного статусу (до 255 символів).
name	Текст	Назва статусу (до 255 символів).
is_active	Булевий	Булеве поле, що вказує, чи активний статус.
set_as_done	Булевий	Булеве поле, що вказує, чи слід вважати завдання з цим статусом завершеним.
order_id	Число	Числове значення для визначення порядку або пріоритету статусів.

Сутність User

Таблиця user є критично важливою для управління користувачами та забезпечення безпеки системи, зберігаючи важливу інформацію про кожного користувача.

Рисунок 2.16 показує взаємозв'язки бази даних між таблицями user та user_role.

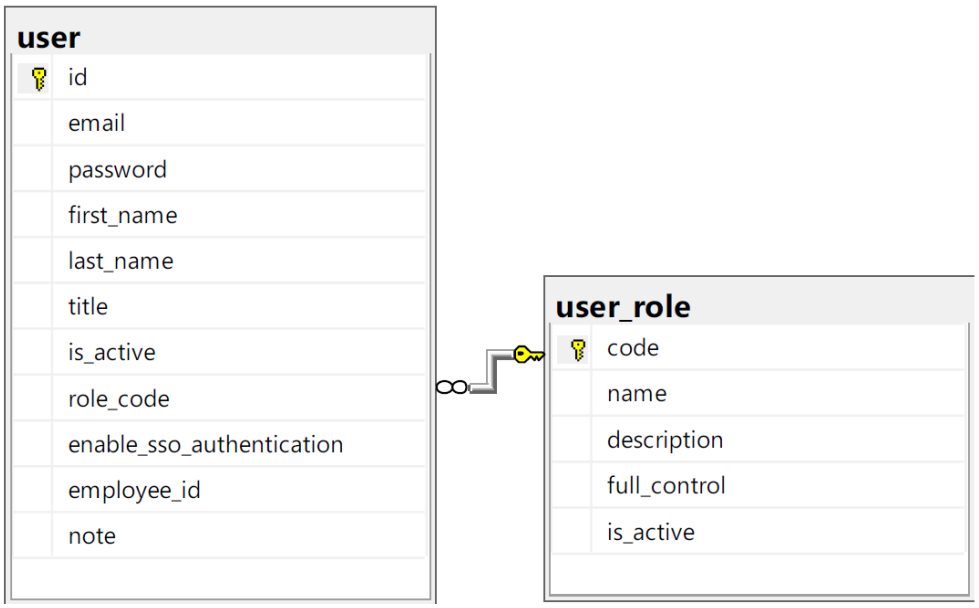

Рисунок 2.16 - Діаграма даних User

Таблиця 2.7 - Опис сутності User

Назва стовпця	Тип даних	Опис
id	Число	Унікальний ідентифікатор для кожного користувача.
email	Текст	Електронна адреса користувача (логін).
password	Текст	Пароль користувача (збережений у хеш-форматі).
first_name	Текст	Ім'я користувача (до 255 символів).
last_name	Текст	Прізвище користувача (до 255 символів).
title	Текст	Професійна або посадова назва користувача (до 255 символів).
is_active	Булевий	Булеве поле, що вказує, чи активний обліковий запис користувача.
role_code	Текст	Код, що ідентифікує роль користувача, пов'язаний з таблицею ролей.
enable_sso_authentication	Булевий	Булеве поле, що вказує, чи може користувач аутентифікуватися через єдиний вхід (SSO).
employee_id	Текст	Унікальний ідентифікатор співробітника в організації.
note	Текст	Додаткова інформація або нотатки про користувача.

Сутність User Role

Таблиця user_role є ключовим елементом у системі управління доступом, визначаючи ролі, які можуть бути призначені користувачам, разом з їхніми правами та обов'язками.

Рисунок 2.17 ілюструє таблицю ролей користувачів у базі даних.

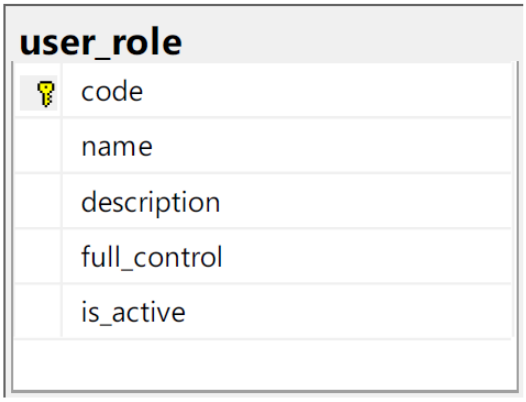

Рисунок 2.17 - Діаграма даних User Role

Таблиця 2.8 - Опис сутності User Role

Назва стовпця	Тип даних	Опис
code	Текст	Унікальний код для кожної ролі (до 255 символів).
Name	Текст	Назва ролі (до 255 символів).
Description	Текст	Детальний опис ролі.
Full_control	Булевий	Булеве поле, що вказує, чи має роль повний контроль над системою.
Is_active	Булевий	Булеве поле, що вказує, чи активна роль.

Сутність User Token

Таблиця user_token створена для управління токенами аутентифікації користувачів, які є критично важливими для встановлення безпечних з'єднань мобільних додатків з веб-API.

Рисунок 2.18 показує взаємозв'язки бази даних між таблицями user_token та user.

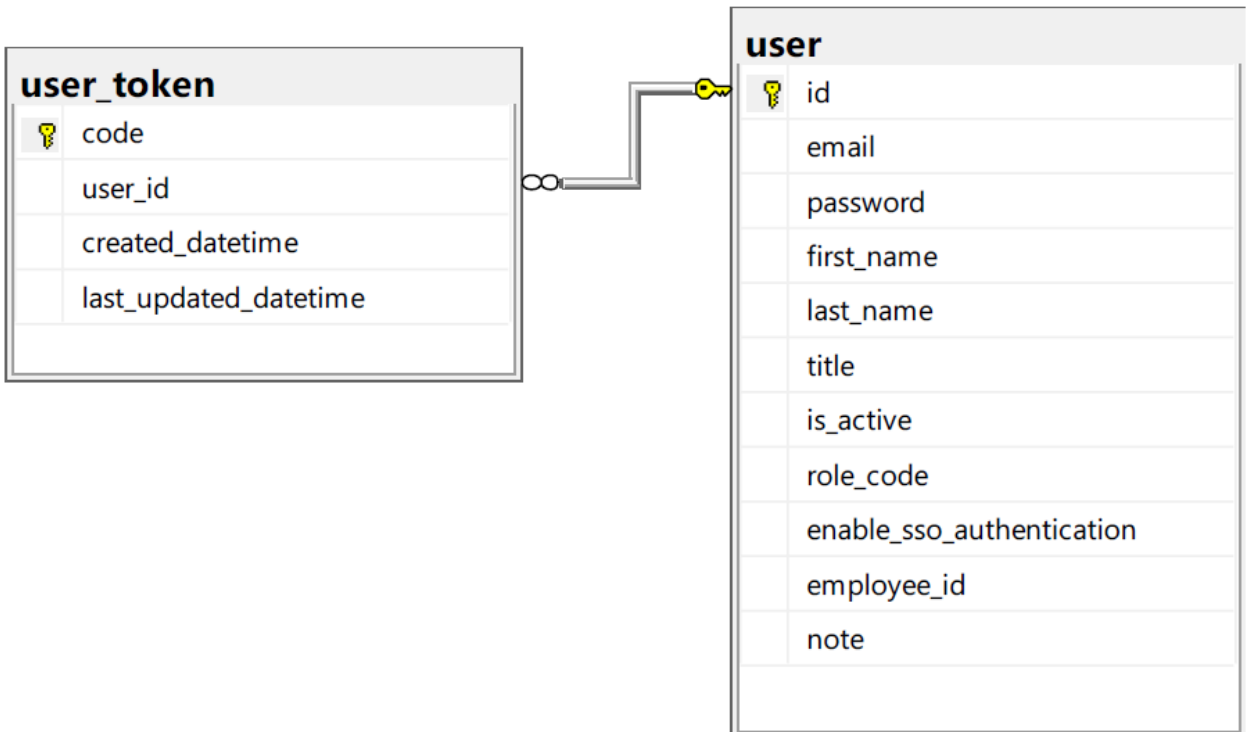

Рисунок 2.18 - Діаграма даних User Token

Таблиця 2.9 - Опис сутності User Token

Назва стовпця	Тип даних	Опис
code	Текст	Унікальний токен, згенерований для сесії користувача (до 255 символів).
user_id	Число	Ідентифікатор користувача, якому присвоєно токен.
created_datetime	Дата та час	Дата та час створення токена.
last_updated_datetime	Дата та час	Дата та час останнього оновлення токена.

Сутність Priority

Призначення таблиці priority полягає у встановленні та управлінні рівнями пріоритету, призначеними проектам, що впливає на розподіл ресурсів та уваги.

Рисунок 2.19 ілюструє таблицю пріоритетів проектів у базі даних.

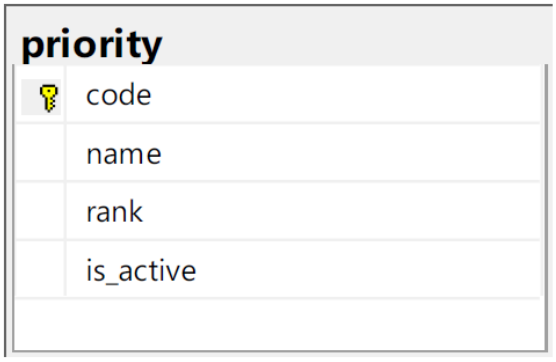

Рисунок 2.19 - Діаграма даних Priority

Таблиця 2.10 - Опис сутності Priority

Назва стовпця	Тип даних	Опис
code	Текст	Унікальний код для кожного рівня пріоритету (до 50 символів).
name	Текст	Назва рівня пріоритету (до 1000 символів).
rank	Число	Числове значення, що представляє ранг або порядок пріоритету.
is_active	Булевий	Рівень пріоритету активний чи ні.

2.3.2. Зв'язуючі таблиці (відношення багато-до-багатьох)

Таблиця project_assignee розроблена для ефективного управління відносинами "багато-до-багатьох" між проектами та виконавцями, дозволяючи призначати кількох користувачів (виконавців) до кількох проектів.

Рисунок 2.20 показує взаємозв'язки бази даних між таблицями project, project_assignee та user.

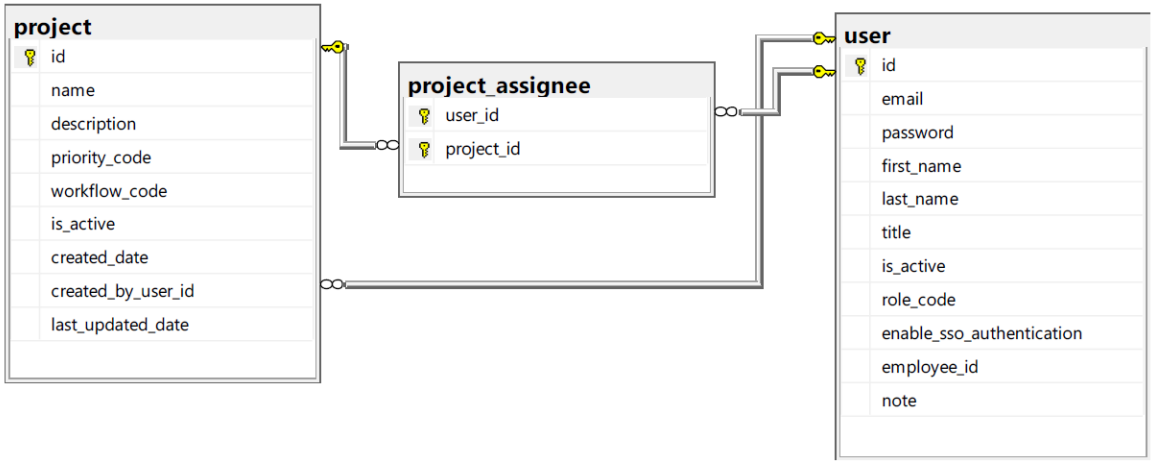

Рисунок 2.20 - Діаграма даних Project Assignee

Таблиця 2.11 - Опис сутності Project Assignee

Назва стовпця	Тип даних	Опис
user_id	Число	Ідентифікатор користувача, призначеного до проекту.
project_id	Число	Ідентифікатор проекту, до якого призначений користувач.

Таблиця project_watcher призначена для обробки відносин "багато-до-багатьох" між проектами та спостерігачами, дозволяючи кільком користувачам ("спостерігачам") стежити за кількома проектами.

Рисунок 2.21 показує взаємозв'язки бази даних між таблицями project, project_watcher та user.

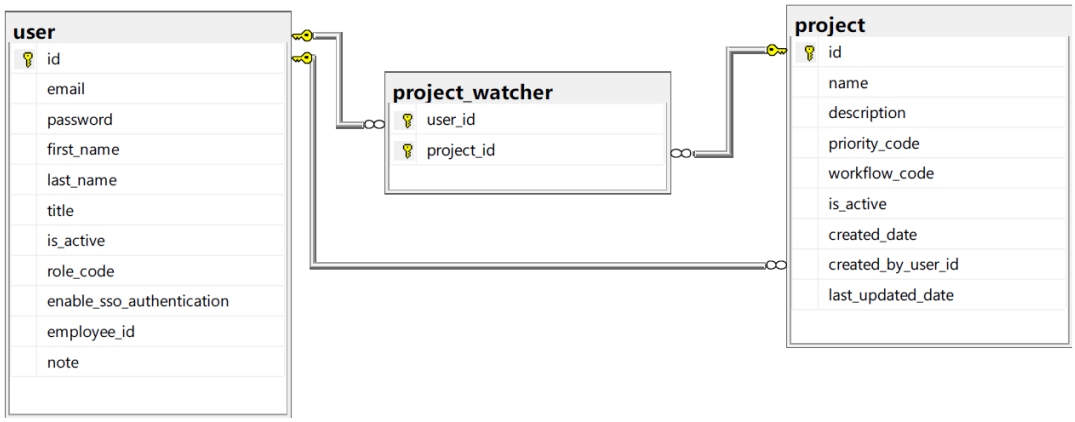

Рисунок 2.21 - Діаграма даних Project Watcher

Таблиця 2.12 - Опис сутності Project Watcher

Назва стовпця	Тип даних	Опис
`user_id`	Число	Ідентифікатор користувача, який спостерігає за проектом.
`project_id`	Число	Ідентифікатор проекту, за яким спостерігає користувач.

Ця детальна модель бази даних забезпечує надійну основу для зберігання та управління даними в системі відстеження проектів, підтримуючи її функціональні та нефункціональні вимоги.

2.4. Проектування інтерфейсу користувача засобами створення макетів інтерфейсу

Створення макетів інтерфейсу є невід'ємним етапом у розробці програмного забезпечення, оскільки вони надають візуальне представлення інтерфейсу користувача (UI) до початку фактичної розробки. Ця рання візуалізація допомагає визначити структуру, макет та взаємодії різних компонентів, забезпечуючи, що всі зацікавлені сторони мають чітке уявлення про очікувані функції та потоки користувачів. У контексті методології водоспаду макети інтерфейсу відіграють ключову роль у ранньому виявленні вимог, що мінімізує необхідність у дорогих змінах на пізніших етапах

розробки. Вони сприяють чіткій комунікації між дизайнерами, розробниками та клієнтами, усуваючи неоднозначності та забезпечуючи узгодженість очікувань усіх сторін. Це призводить до більш ефективного процесу розробки та сприяє загальному успіху проекту.

Даний проект передбачає розробку двох окремих наборів макетів інтерфейсу: один для веб-додатку та інший для мобільного додатку. Кожен макет інтерфейсу матиме унікальний дизайн, причому мобільна версія пропонуватиме дещо менше функціональних можливостей порівняно з веб-версією.

2.4.1. Макети інтерфейсу веб-додатку

Рисунок 2.22 відображає форму входу для веб-додатку системи відстеження проектів, де користувачі повинні ввести свою електронну пошту та пароль для безпечної автентифікації своїх облікових записів.

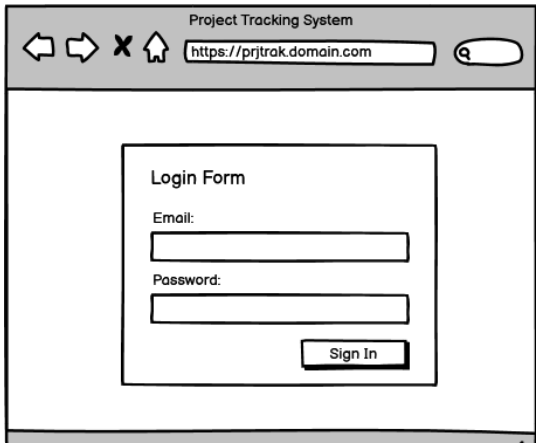

Рисунок 2.22 – Сторінка входу в веб-додаток

Рисунок 2.23 представляє робочий процес проекту, який дозволяє користувачам легко переміщатися між проектами, організованими за різними станами: Очікування, Дизайн, Розробка, Тестування та Виробництво. Цей інтерфейс забезпечує організований вигляд, що полегшує моніторинг та оновлення статусу.

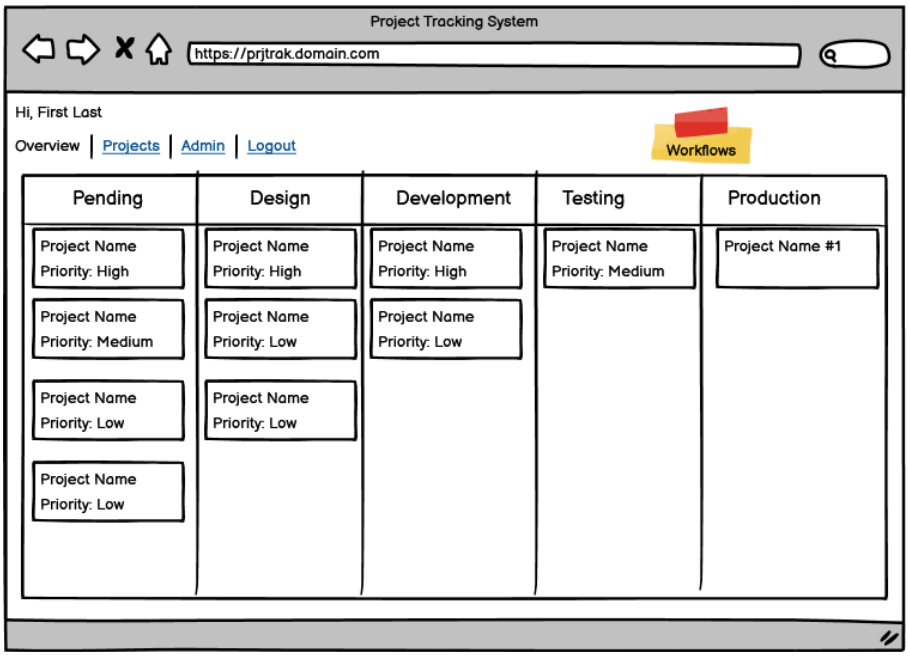

Рисунок 2.23 – Макет панелі огляду веб-додатку

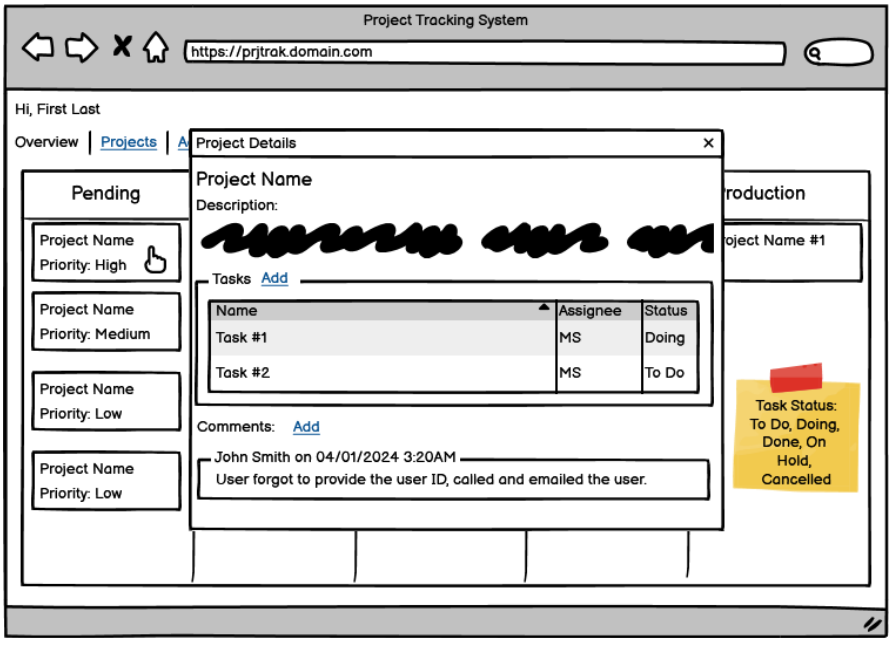

Рисунок 2.24 – Макет сторінки деталей проекту веб-додатку

Рисунок 2.24 є оглядом проекту, що включає розділи для опису проекту, коментарів та завдань. Він дозволяє користувачам легко додавати завдання та відстежувати їхній прогрес на різних етапах, сприяючи співпраці та ефективному управлінню проектами.

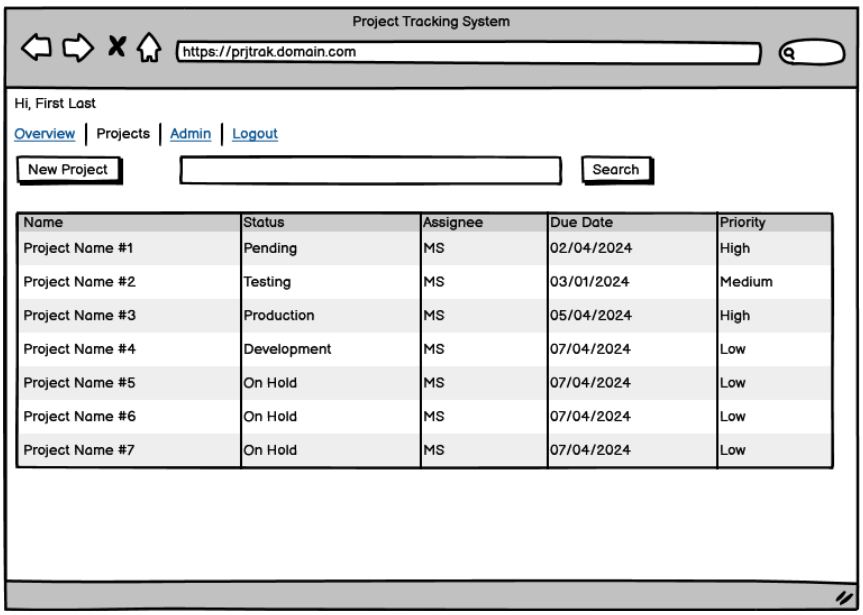

Рисунок 2.25 – Макет сторінки проектів веб-додатку

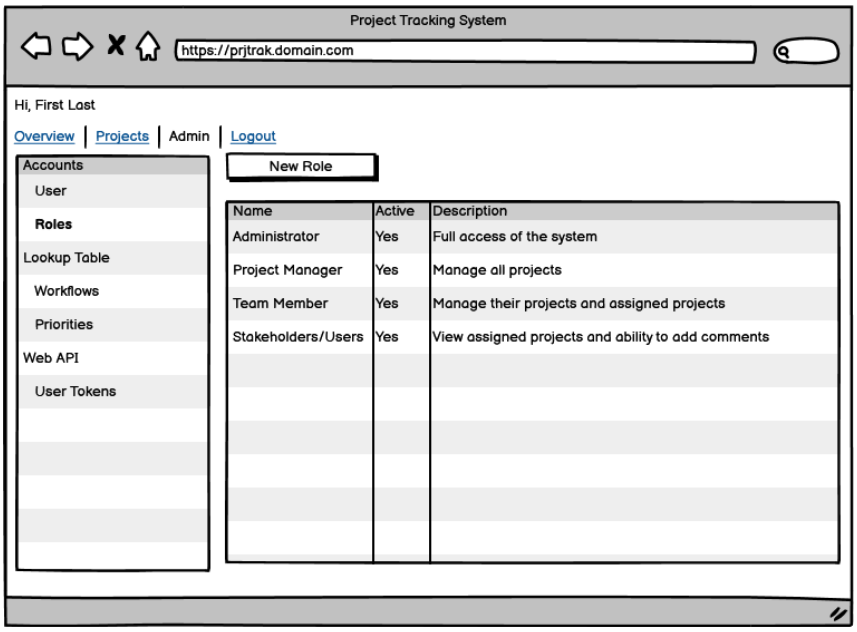

Рисунок 2.26 – Макет сторінки управляння ролями користувача

Рисунок 2.25 надає короткий огляд проектів, включаючи таку інформацію, як назва проекту, статус, виконавець, активний статус тощо. Мета цього списку — надати менеджерам зручний спосіб оцінки прогресу та терміновості проектів.

Рисунок 2.26 відображає інтерфейс для управління ролями, надаючи архітектуру для визначення та модифікації ролей користувачів. Це включає

можливість встановлення дозволів та описів для кожної ролі, що є критично важливим для забезпечення належного доступу до функцій системи.

Рисунок 2.27 ілюструє макет інтерфейсу користувача веб-додатку.

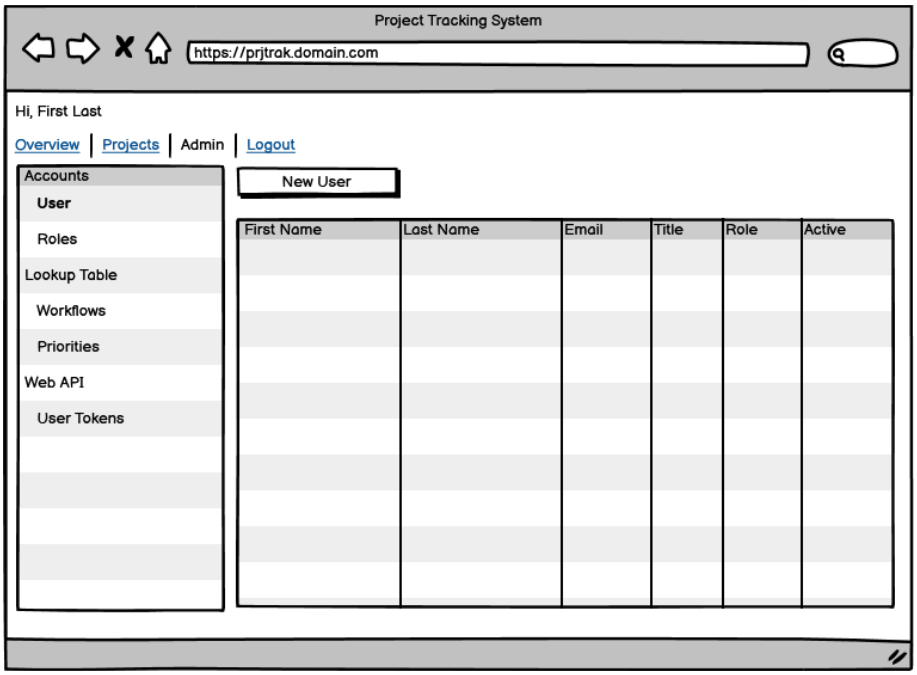

Рисунок 2.27 – Макет сторінки інтерфейсу користувача

2.4.2. Макети інтерфейсу мобільного додатку

Рисунок 2.28 ілюструє макети інтерфейсу входу в мобільний додаток.

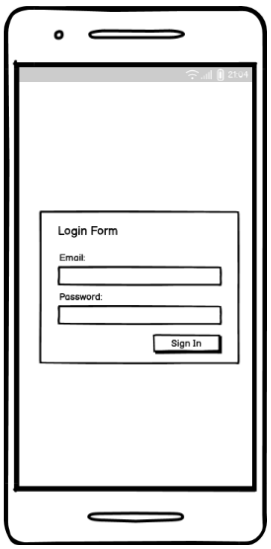

Рисунок 2.28 – Макет входу в мобільний додаток

Рисунок 2.29 ілюструє макет інтерфейсу першого робочого процесу мобільного додатку.

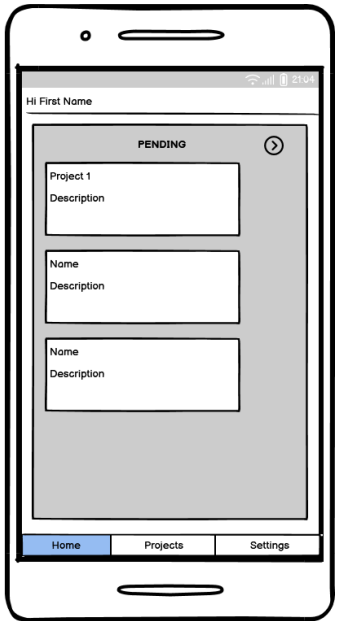

Рисунок 2.29 - Макет інтерфейсу першого робочого процесу

Рисунок 2.30 ілюструє етап робочого процесу, на якому відображається перелік проектів із зазначенням їхніх назв та коротких описів. Користувач має можливість навігації до попереднього або наступного етапу робочого процесу за допомогою іконок "<" та ">".

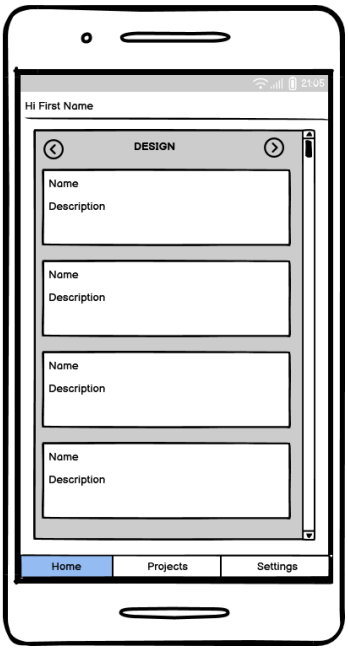

Рисунок 2.30 - Макет інтерфейсу середнього робочого процесу

Рисунок 2.31 ілюструє етап робочого процесу, на якому відображається перелік проектів із зазначенням їхніх назв та коротких описів. Оскільки це останній етап робочого процесу, користувач може здійснити навігацію лише до попереднього етапу за допомогою іконки "<".

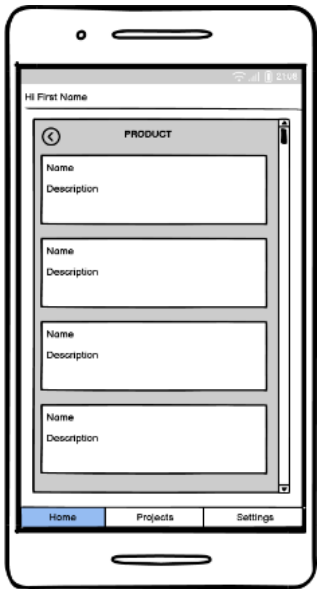

Рисунок 2.31 - Макет інтерфейсу останнього робочого процесу

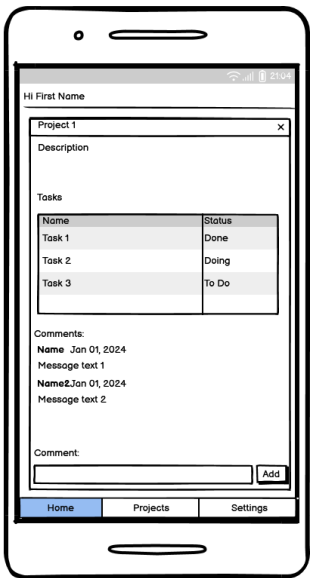

Рисунок 2.32 - Макет інтерфейсу “Деталі проекту”

Рисунок 2.32 ілюструє випадок, коли користувач натискає на проект, щоб переглянути його деталі та пов'язану інформацію. До цієї інформації входять коментарі до проекту та перелік пов'язаних завдань.

Рисунок 2.33 ілюструє, що при натисканні користувачем на вкладку "Проекти" відображається повний список його проектів, що включає як активні, так і неактивні проекти. Користувач може скористатися кнопкою "Додати", щоб додати нові проекти до списку, а також має можливість редагувати проект, натиснувши на нього.

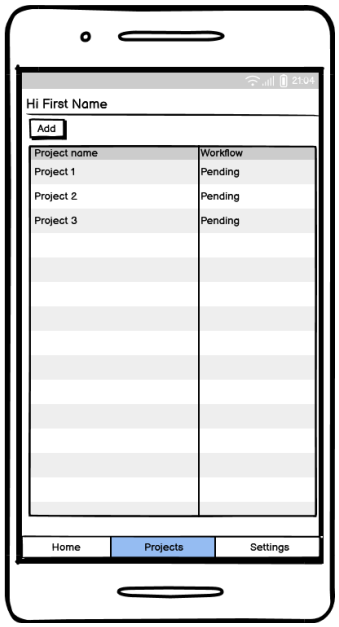

Рисунок 2.33 - Макет інтерфейсу мобільного додатку “Проекти”

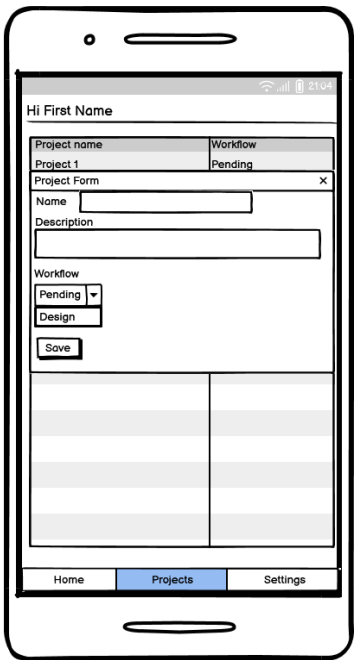

Рисунок 2.34 - Макет інтерфейсу мобільного додатку “Форма проекту”

Рисунок 2.34 ілюструє можливість користувача додавати або редагувати проект. При натисканні на кнопку "Додати", користувач отримує змогу внести новий проект до списку. З'являється нова форма проекту, що дозволяє користувачеві ввести необхідну інформацію та натиснути "Зберегти" для збереження.

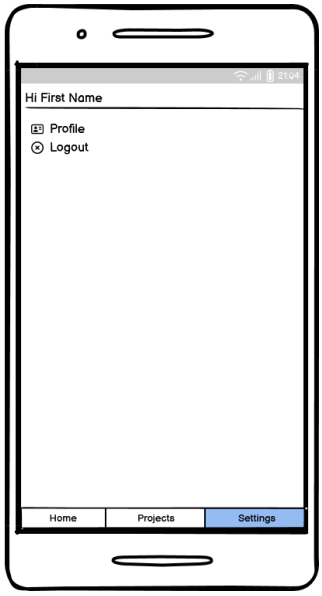

Рисунок 2.35 - Макет інтерфейсу мобільного додатку “Налаштування”

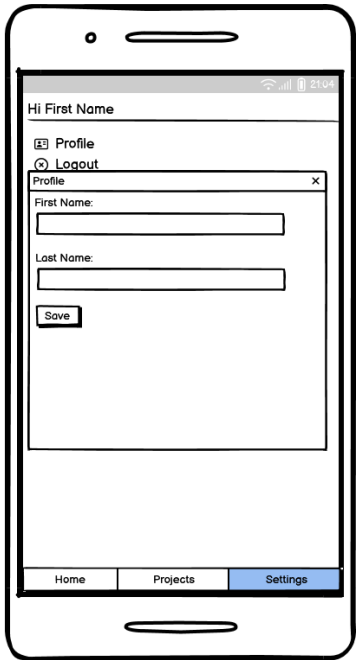

Рисунок 2.36 - Макет інтерфейсу мобільного додатку “Профіль”

Рисунок 2.35 ілюструє, що користувач може або редагувати свій профіль, або вийти з системи. При натисканні на "Профіль" з'явиться форма профілю. Для виходу з системи потрібно натиснути "Вийти", після чого користувач повернеться до форми входу.

Рисунок 2.36 ілюструє, що користувач може змінити своє ім'я та прізвище. Після натискання кнопки "Зберегти" інформація його профілю оновлюється.

Ці макети інтерфейсу слугують основою для розробки інтерфейсу користувача, забезпечуючи послідовність та відповідність вимогам на етапах реалізації.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВІДСТЕЖЕННЯ ТА ВЕДЕННЯ ПРОЕКТІВ

У цьому розділі описано практичну реалізацію системи відстеження проектів, яка використовує як веб-, так і мобільні платформи. Ця система розробляється з застосуванням технологій MVC 5 та .NET MAUI. Архітектура системи базується на трирівневій моделі [13], що забезпечує чітке розділення між рівнем представлення, рівнем бізнес-логіки та рівнем доступу до даних. Такий шаруватий підхід значно покращує підтримку та масштабованість, а також спрощує процес розробки.

3.1. Архітектура трирівневої системи та переваги спільних бібліотек

Трирівнева архітектура є фундаментальною концепцією в розробці програмного забезпечення, що передбачає логічне розділення системи на три основні рівні: рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень доступу до даних (Data Access Layer). Ця архітектура сприяє створенню модульних, масштабованих та підтримуваних програмних рішень, забезпечуючи чітке розмежування обов'язків та покращуючи загальну організацію та ефективність системи.

Рівень представлення, відомий також як рівень інтерфейсу користувача, є верхнім рівнем трирівневої архітектури. Він відповідає за взаємодію з користувачем, обробляючи вхідні дані та відображаючи вихідні. Цей рівень зазвичай складається з інтерфейсів користувача, таких як веб-сторінки, мобільні екрани або настільні додатки, що дозволяють користувачам взаємодіяти з системою. Його основна мета полягає у наданні користувачам зручного та інтуїтивно зрозумілого способу доступу до функцій та даних програми.

Рівень бізнес-логіки, або середній рівень, є проміжним компонентом трирівневої архітектури. Він містить основну логіку програми та бізнес-правила, що визначають реакцію системи на вхідні дані та послідовність виконуваних операцій. Цей рівень обробляє дані, отримані з рівня представлення, застосовує необхідні обчислення та правила, а потім передає результати назад до рівня представлення або до рівня доступу до даних для зберігання. Основне призначення цього рівня полягає у забезпеченні централізованого та організованого управління логікою програми, що значно спрощує підтримку та оновлення системи.

Рівень доступу до даних, відомий також як рівень зберігання даних, є нижнім рівнем трирівневої архітектури. Він відповідає за взаємодію з базою даних або іншими системами зберігання даних. Цей рівень обробляє всі операції, пов'язані з даними, такі як читання, запис, оновлення та видалення даних з бази даних. Він також забезпечує абстракцію даних, що дозволяє рівню бізнес-логіки взаємодіяти з даними без необхідності знати деталі реалізації бази даних. Основна мета цього рівня — забезпечити надійний та ефективний доступ до даних, а також підтримувати цілісність та безпеку інформації.

Спільна бібліотека в контексті розробки програмного забезпечення – це набір попередньо написаного коду, функцій, класів та ресурсів, які можуть бути використані в різних частинах програми або навіть в різних програмах. Використання спільної бібліотеки надає кілька значних переваг, що покращують процес розробки та якість кінцевого продукту:

- спільна бібліотека дозволяє розробникам перевикористовувати код, який вже був написаний, протестований та оптимізований. Це зменшує необхідність писати новий код з нуля, що значно економить час та зусилля, а також зменшує кількість помилок та багів.

- використання спільної бібліотеки допомагає забезпечити консистентність у всьому проекті. Оскільки всі частини програми

використовують один і той же набір функцій та ресурсів, це допомагає уникнути розбіжностей та забезпечити однаковий вигляд та поведінку в різних модулях програми.

- спільна бібліотека спрощує підтримку та оновлення коду. Всі зміни та оновлення вносяться в одному централізованому місці, що усуває необхідність модифікувати кожну частину програми окремо. Це також гарантує, що всі компоненти програми використовують останню версію коду та ресурсів.

- спільна бібліотека сприяє масштабованості програми. Оскільки код та ресурси організовані централізовано, це дозволяє легко додавати нові функції та ресурси, а також розширювати існуючі. Це забезпечує легку адаптацію програми до змін у вимогах та умовах.

- використання спільної бібліотеки може підвищити ефективність програми. Оскільки код та ресурси оптимізовані та протестовані, це допомагає зменшити кількість помилок та покращити продуктивність.

3.2. Огляд Фреймворків MVC 5 та .NET MAUI у розробці

MVC 5 (Model-View-Controller) та .NET MAUI (Multi-platform App UI) є двома провідними фреймворками, що застосовуються для розробки веб-додатків та крос-платформних мобільних додатків відповідно. Кожен з них володіє унікальними особливостями та перевагами, що зумовлюють їхню популярність серед розробників.

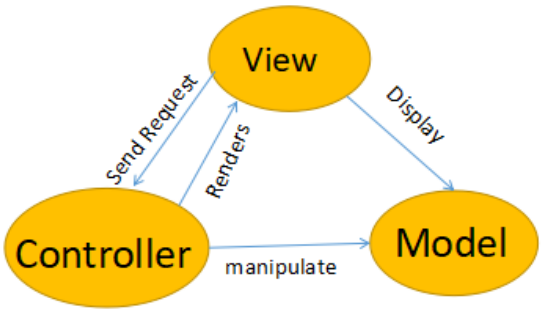

Рисунок 3.1 – MVC фреймворк

MVC 5 – це фреймворк для розробки веб-додатків, заснований на архітектурному шаблоні Model-View-Controller. Цей шаблон розділяє додаток на три взаємопов'язані компоненти:

Модель (Model) - відповідає за управління даними та бізнес-логікою програми. Вона взаємодіє з базою даних або іншими системами зберігання даних для отримання та оновлення інформації.

Представлення (View) - відповідає за відображення даних користувачеві. Вона взаємодіє з моделлю для отримання даних та їх відображення у зручному для користувача форматі.

Контролер (Controller) - обробляє вхідні дані від користувача та взаємодіє з моделлю та представленням для виконання необхідних дій. Він приймає вхідні дані, обробляє їх, а потім оновлює модель та представлення відповідно.

Переваги MVC 5 включають:

- Чітке розмежування компонентів покращує організацію та підтримку коду.

- Розділення компонентів дозволяє легко тестувати кожен з них ізольовано, забезпечуючи коректне функціонування.

- Фреймворк дозволяє легко додавати нові функції та розширювати існуючі, забезпечуючи адаптацію програми до змінних вимог.

.NET MAUI – це фреймворк для розробки мобільних додатків, що дозволяє створювати нативні додатки для iOS, Android, macOS та Windows з використанням єдиної кодової бази. Він використовує XAML (eXtensible Application Markup Language) для визначення інтерфейсу користувача та C# для реалізації логіки програми.

Ключові переваги .NET MAUI:

- Можливість розробки для декількох платформ з одним кодом значно економить час та зменшує кількість помилок.

- Тісна інтеграція з Visual Studio спрощує процеси створення, тестування та налагодження додатків.

- Дозволяє легко додавати нові функції та масштабувати додатки, забезпечуючи їх адаптивність до змінних умов.

Visual Studio 2022 є потужним інтегрованим середовищем розробки (IDE) від Microsoft, призначеним для розробки програмного забезпечення на різних платформах, включаючи Windows, macOS, iOS та Android. Вона пропонує широкий спектр інструментів та функцій для створення, тестування та налагодження програм.

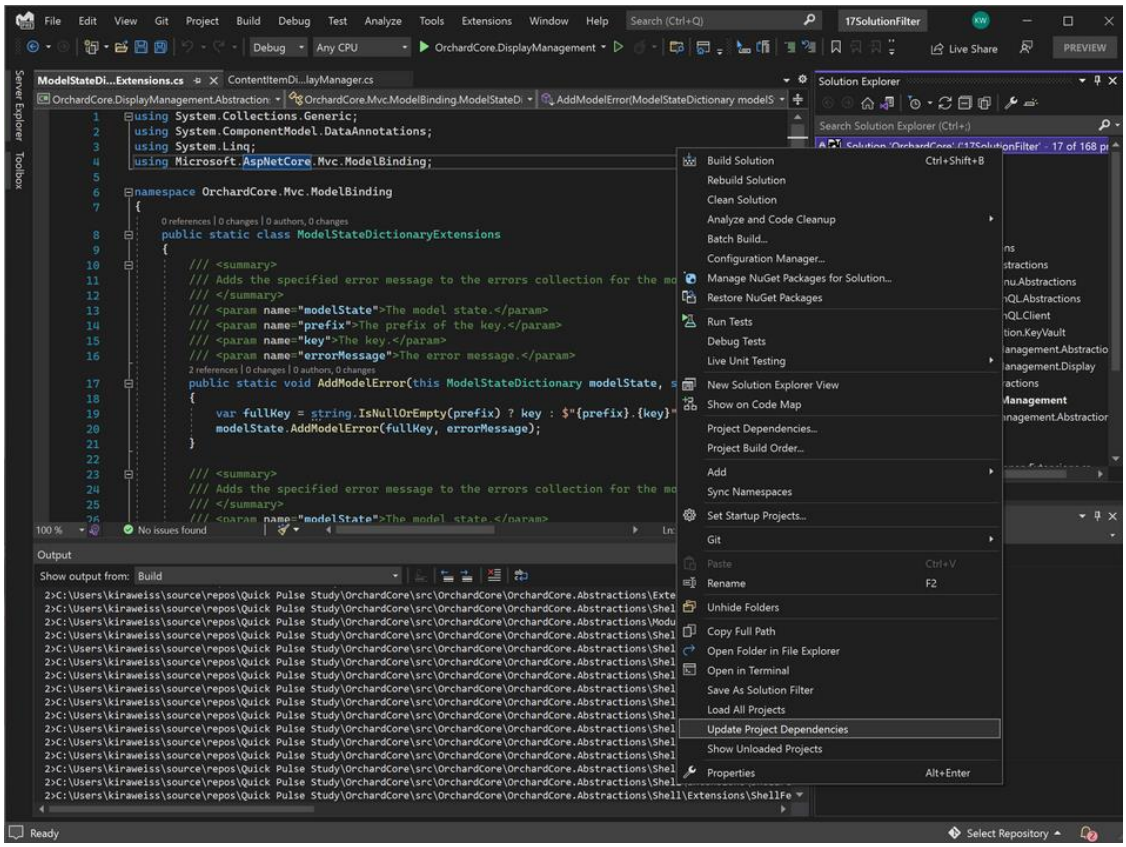

Рисунок 3.2 – Інтерфейс IDE Visual Studio 2022

Основні особливості Visual Studio 2022:

- Охоплює C#, C++, Python, JavaScript та інші мови, дозволяючи розробникам вибирати оптимальну мову для проекту.

- Підтримує популярні системи, такі як Git та Azure DevOps, для ефективного управління кодом та співпраці.

- Надає потужні інструменти для модульного тестування, інтеграційного тестування та налагодження коду.

- Дозволяє створювати додатки, які працюють на різних пристроях.

- Забезпечує легке розгортання та управління додатками в хмарі, зокрема з Azure.

- Включає інструменти для ASP.NET, HTML, CSS та JavaScript для створення сучасних веб-додатків.

- Сумісність з Docker, Kubernetes та Jenkins.

Visual Studio 2022 є універсальним інструментом, що надає все необхідне для створення високоякісного та ефективного програмного забезпечення.

3.3. Структура рішення проекту

Структура рішення проекту в контексті розробки програмного забезпечення відноситься до організації та управління різними компонентами та ресурсами, які входять до складу проекту. Вона включає в себе планування, проектування, розробку, тестування та розгортання програмного забезпечення, а також управління командою, бюджетом та термінами.

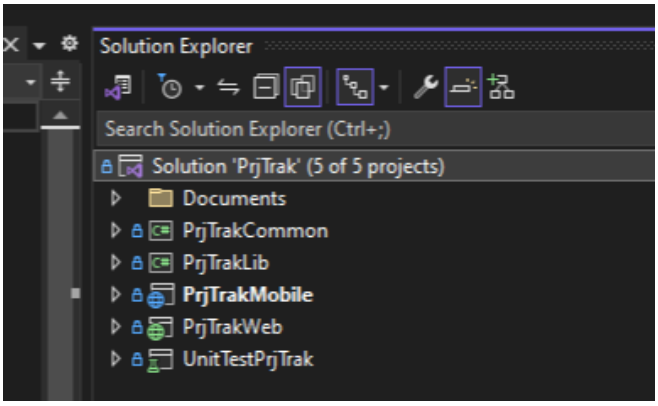

Рисунок 3.3 - Структура програмного рішення

Рисунок 3.3 представляє детальний аналіз кожного елементу в структурі програмного рішення системи відстеження проектів.

Основні компоненти структури рішення проекту включають:

1. Вимоги. Визначення та документування вимог до програмного забезпечення, включаючи функціональні та нефункціональні вимоги, а також вимоги до безпеки та продуктивності.

2. Архітектура. Проектування архітектури програмного забезпечення, включаючи вибір технологій, фреймворків та інструментів, а також визначення структури та компонентів системи.

3. Розробка. Реалізація програмного забезпечення відповідно до вимог та архітектури, включаючи написання коду, створення інтерфейсів користувача та інтеграцію з іншими системами та сервісами.

4. Тестування. Перевірка програмного забезпечення на відповідність вимогам та виявлення помилок та багів, включаючи модульне тестування, інтеграційне тестування та системне тестування.

5. Розгортання. Встановлення та налаштування програмного забезпечення на цільових системах та пристроях, включаючи планування розгортання, тестування розгортання та моніторинг продуктивності.

6. Управління проектами: Планування, організація та контроль за розробкою програмного забезпечення, включаючи управління командою, бюджетом та термінами, а також комунікацію з зацікавленими сторонами.

Структура рішення проекту є важливою для успішної реалізації програмного забезпечення, оскільки вона допомагає забезпечити, що всі компоненти та ресурси проекту організовані та керовані ефективно, а також що проект відповідає вимогам та очікуванням зацікавлених сторін.

PriTrakCommon - це спільна бібліотека, яка використовується в проекті для забезпечення перевикористання коду та ресурсів між різними компонентами системи. Вона містить загальні функції, класи та ресурси, які можуть бути використані в різних частинах програми, що допомагає

забезпечити консистентність та ефективність коду. PriTrakCommon є важливою частиною проекту, оскільки вона допомагає забезпечити перевикористання коду та ресурсів, а також консистентність та ефективність коду в усьому проекті.

PriTrakLib - це бібліотека, яка містить основну логіку та функції для системи відстеження проектів. Вона надає функції для управління проектами, завданнями, користувачами та іншими компонентами системи, а також забезпечує інтеграцію з базою даних та іншими системами.

PriTrakMobile - це мобільний додаток, який дозволяє користувачам взаємодіяти з системою відстеження проектів з їх мобільних пристроїв. Він надає можливості для перегляду та управління проектами, завданнями та іншими компонентами системи, а також для отримання сповіщень та оновлень в реальному часі.

PriTrakWeb - це веб-додаток, який дозволяє користувачам взаємодіяти з системою відстеження проектів з їх веб-браузерів. Він надає можливості для перегляду та управління проектами, завданнями та іншими компонентами системи, а також для отримання сповіщень та оновлень в реальному часі.

UnitTestPriTrak - це набір модульних тестів для системи відстеження проектів, який використовується для перевірки окремих компонентів та функцій системи на відповідність вимогам та виявлення помилок та багів. Модульні тести допомагають забезпечити, що кожен компонент системи працює правильно та відповідає вимогам, а також допомагають виявити та виправити помилки та баги на ранніх етапах розробки.

3.4. Реалізація веб-додатку на основі MVC архітектури

Ефективна реалізація веб-додатку значною мірою залежить від стратегічного використання архітектури MVC (Model-View-Controller). Ця архітектура чітко визначає ролі контролерів, представлень та моделей, що

значно спрощує розробку та подальшу підтримку. Контролери виступають у ролі посередників між користувацьким інтерфейсом та моделлю даних, обробляючи вхідні дані від користувача, виконуючи бізнес-логіку та обираючи відповідні представлення для рендерингу відповідей.

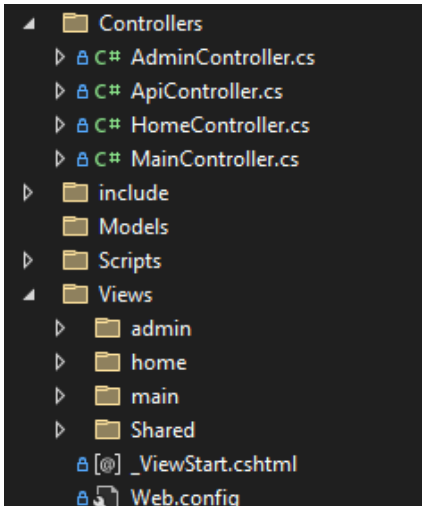

Рисунок 3.4 - Структура MVC додатку

Рисунок 3.4 ілюструє структуру MVC веб-додатку. Проектна структура є добре організованою, з контролерами, розподіленими за різними областями взаємодії з користувачем:

- AdminController відповідає за управління адміністративними функціями та забезпечення безпечного середовища для нагляду за системою.

- ApiController відіграє ключову роль у зв'язуванні мобільного додатку з бекендом, забезпечуючи безперебійний обмін даними для мобільних користувачів.

- HomeController відповідає за обробку специфічних для користувача представлень та взаємодій, забезпечуючи персоналізований користувацький досвід.

MainController доступний для всіх користувачів, обробляє загальні запити та слугує відправною точкою для неаутентифікованих користувачів, відображаючи загальнодоступні компоненти.

3.4.1. Рівні бізнес-логіки та доступу до даних

Рисунок 3.5 демонструє надійну трирівневу архітектуру, яка ефективно розділяє обов'язки на рівні об'єктів доступу до даних (DAO) та сервісні рівні для підтримання чіткого розподілу шарів.

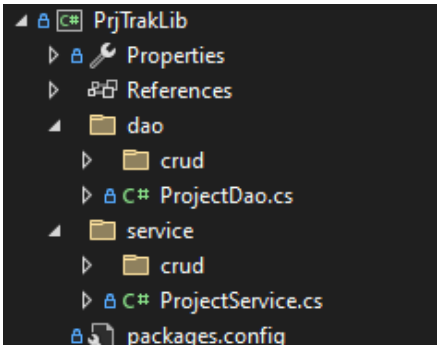

Рисунок 3.5 - Рівень бізнес-логіки та доступу до даних

Через ProjectDao.cs шар DAO ефективно обробляє всі прямі взаємодії з даними, зосереджуючись на CRUD (створення, читання, оновлення, видалення) операціях для оптимізації управління базою даних та підвищення заходів безпеки. Крім того, сервісний рівень, як показано в ProjectService.cs, ефективно інкапсулює бізнес-логіку, забезпечуючи її незалежність від рівня доступу до даних. Цей модульний дизайн дозволяє легко повторно використовувати сервісний рівень у різних додатках або безперешкодно адаптувати його до нових технологій, що значно підвищує підтримку та масштабованість системи.

3.5. Проектування бази даних

Проектування бази даних для цієї системи відстеження проектів було ретельно виконано з використанням SQL Server Management Studio v19.1 для забезпечення ефективної продуктивності та надійного управління даними.

Рисунок 3.6 ілюструє використання дванадцяти таблиць:

- dbo.project для деталізації проектів

- dbo.task для специфікації завдань
- dbo.user для інформації про користувачів
- dbo.project_assignee для зв'язування користувачів з проектами
- dbo.task_comment для обговорень, пов'язаних із завданнями
- dbo.user_role для визначення дозволів користувачів
- dbo.task_status для відстеження прогресу завдань
- dbo.project_comment для обговорень проектів
- dbo.project_watcher для сповіщення користувачів про оновлення
- dbo.user_token для безпечних сесій
- dbo.workflow для визначення етапів процесу
- dbo.priority для встановлення терміновості завдань.

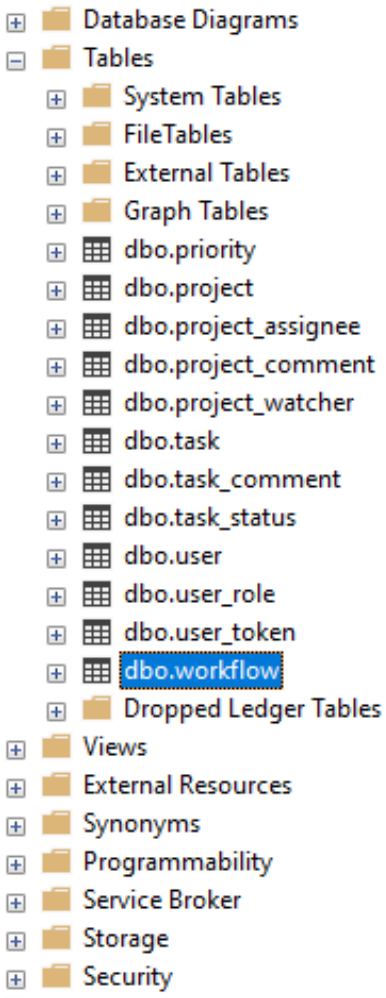

Рисунок 3.6 - Таблиці бази даних

Кожна таблиця ретельно структурована для захоплення та обробки всіх відповідних даних.

3.6. Реалізація Web API та мобільного додатку

Цей проект підкреслює важливість Web API, реалізуючи його через ApiController, який обробляє всі HTTP POST запити від мобільних пристроїв.

```
public class ApiController : Controller
{

    [HttpPost]
    0 references | Mike, 2 days ago
    public JsonResult Mobile(string data)
    {
        var wsResponse = new WResponse();
        wsResponse.setStatus(WStatus.Error);
        try
        {
```

Рисунок 3.7 - Контролер Web API

Функціональність, зображена на рисунку 3.7, є необхідною для безпечної та ефективної передачі даних між мобільним додатком та сервером. Це забезпечує динамічну та чуйну взаємодію мобільного додатку. Наступний розділ пояснить, як ApiController полегшує ці дії та його ключову роль в архітектурі системи.

Рисунок 3.8 ілюструє сучасний підхід до використання .NET MAUI Mobile — фреймворку, який дозволяє розробникам створювати інтерактивні користувацькі інтерфейси для крос-платформних мобільних додатків, використовуючи C#.

Мобільний додаток обробляє рівень представлення, який взаємодіє з бекенд-сервісами через Web API. Web API виступає як міст між мобільним додатком та сервером, обробляючи всю бізнес-логіку, операції з базою даних та зберігання даних.

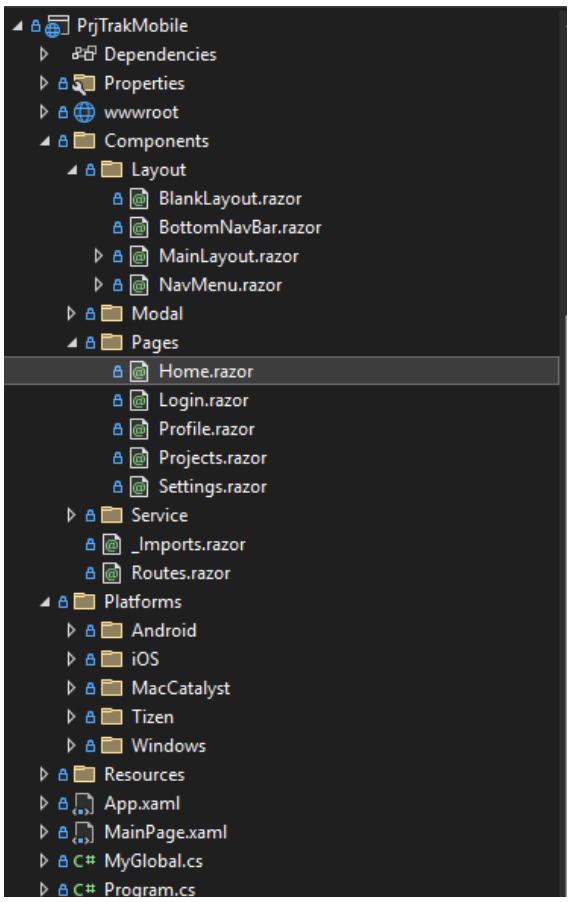

Рисунок 3.8 - Структура .NET MAUI Mobile

3.7. Реалізація інтерфейсу користувача

3.7.1. Представлення веб-додатку системи відстеження та ведення проектів

Рисунок 3.9 - Вхід у веб-додаток

Рисунок 3.10 відображає загальний інтерфейс користувача та ключові елементи навігації веб-додатку.

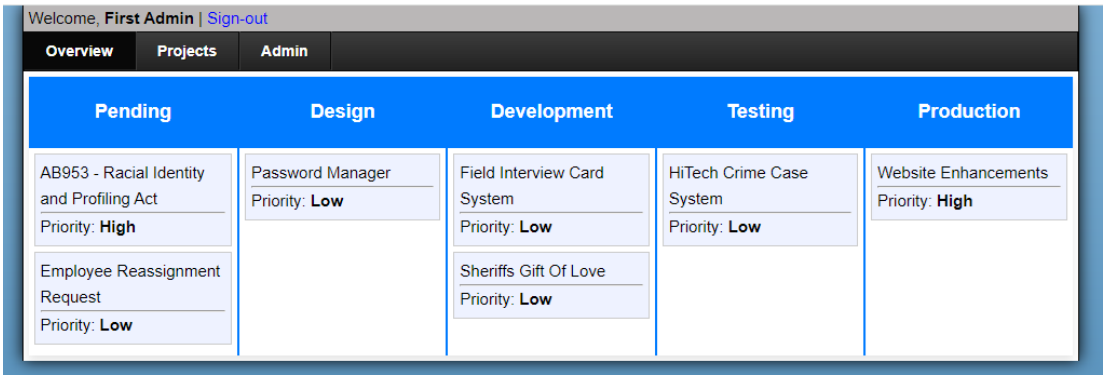

Рисунок 3.10 - Огляд веб-додатку

Рисунок 3.11 представляє форму для введення інформації та параметрів нового завдання.

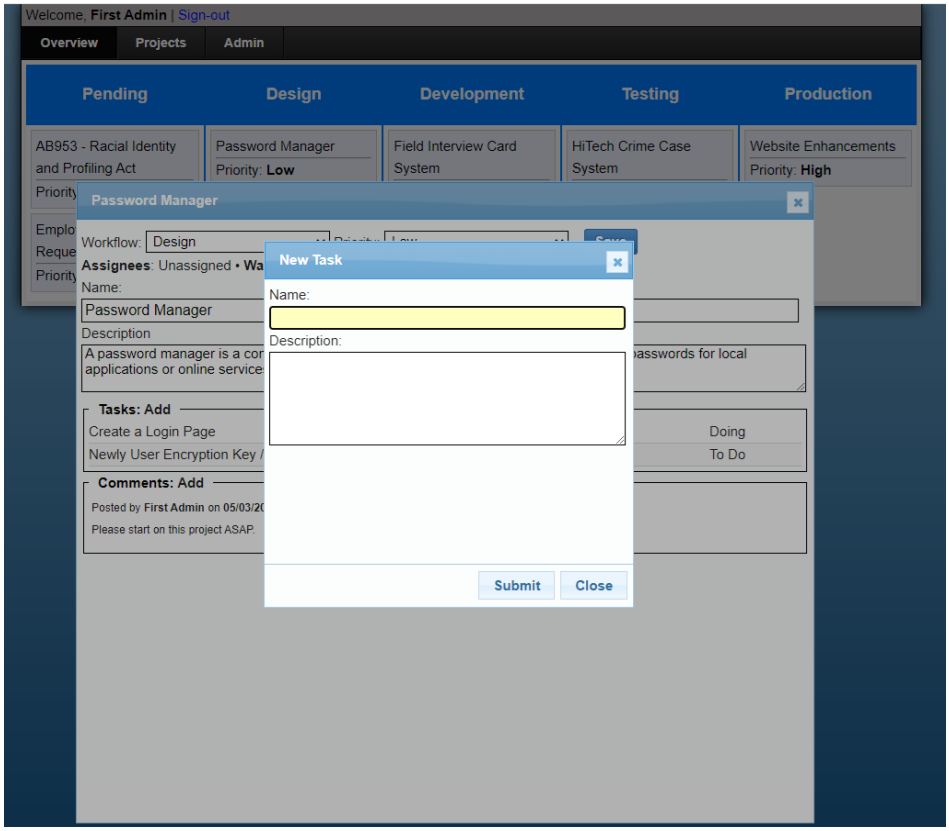

Рисунок 3.11 - Створення нового завдання у веб-додатку

Рисунок 3.12 ілюструє інтерфейс для ініціювання та конфігурації нового проекту в системі.

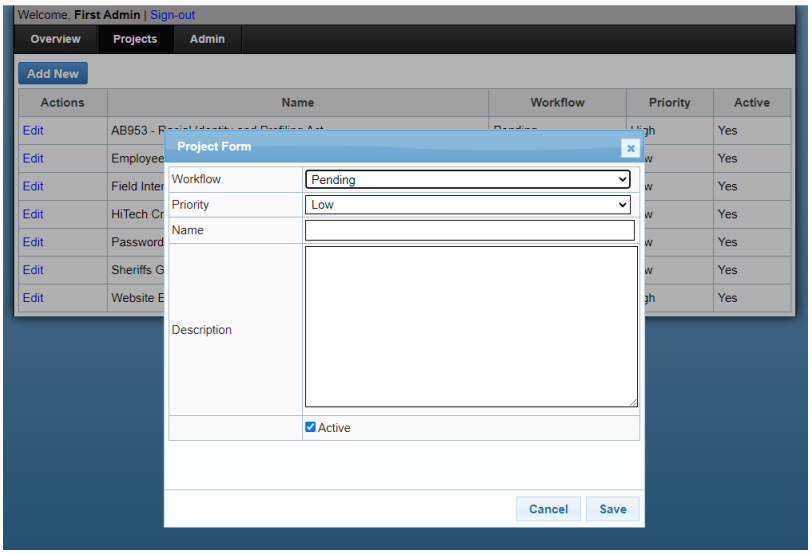

Рисунок 3.12 - Додавання проекту у веб-додатку

Рисунок 3.13 відображає екран управління обліковими записами користувачів, включаючи їхні ролі та права доступу.

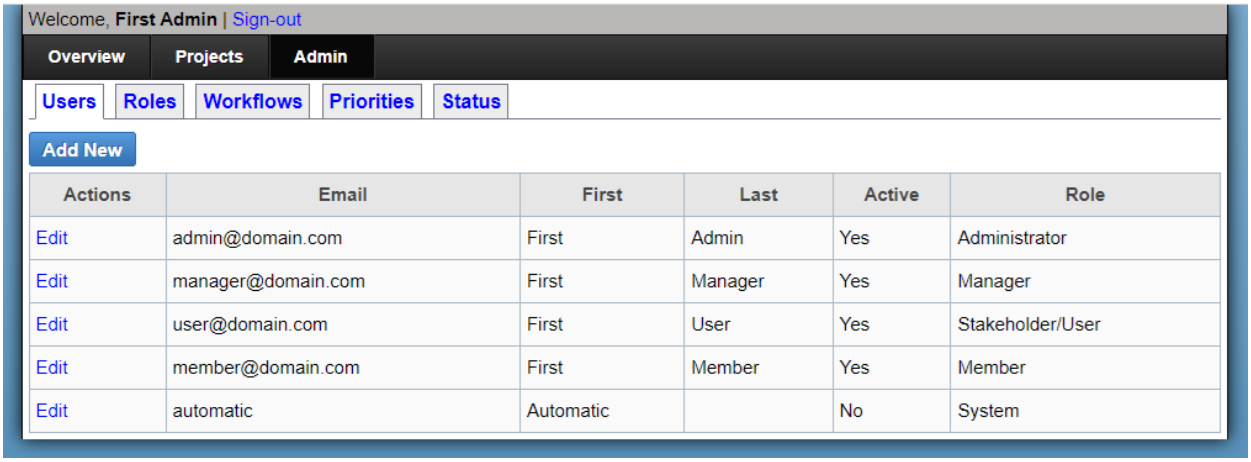

Рисунок 3.13 - Адміністрування користувачів у веб-додатку

Рисунок 3.14 представляє інтерфейс для налаштування та модифікації визначених робочих процесів.

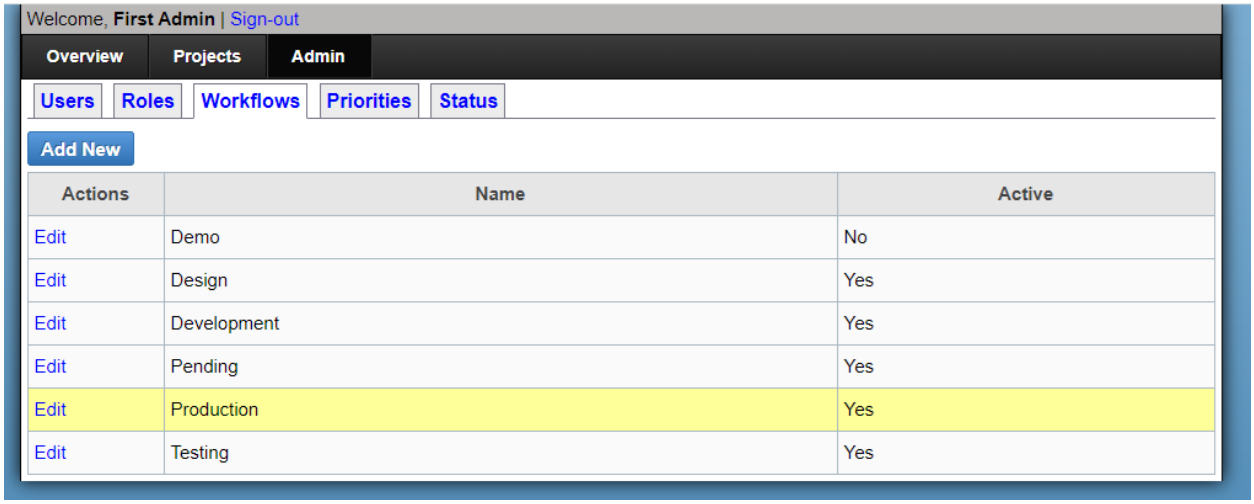

Рисунок 3.14 - Адміністрування робочих процесів у веб-додатку

Рисунок 3.15 ілюструє екран для визначення, перегляду та редагування ролей користувачів та їхніх привілеїв.

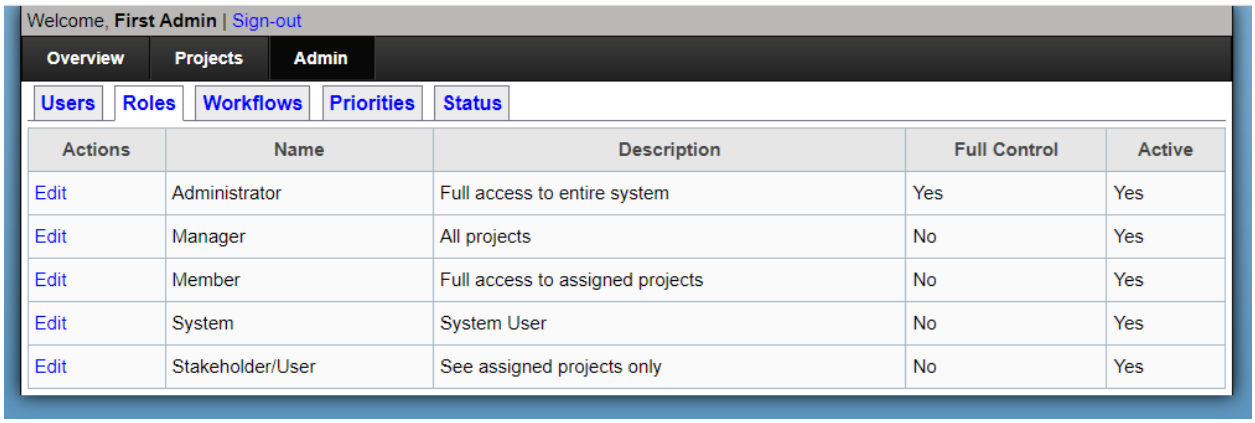

Рисунок 3.15 - Адміністрування ролей у веб-додатку

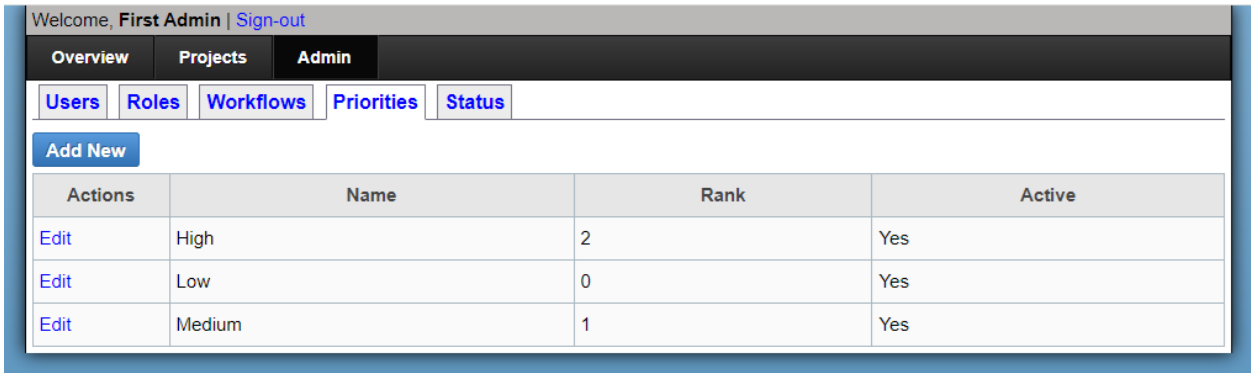

Рисунок 3.16 - Адміністрування пріоритетів у веб-додатку

Рисунок 3.16 відображає інтерфейс для встановлення та управління рівнями пріоритетності проектів.

Рисунок 3.17 представляє екран для конфігурації та моніторингу різних статусів виконання завдань.

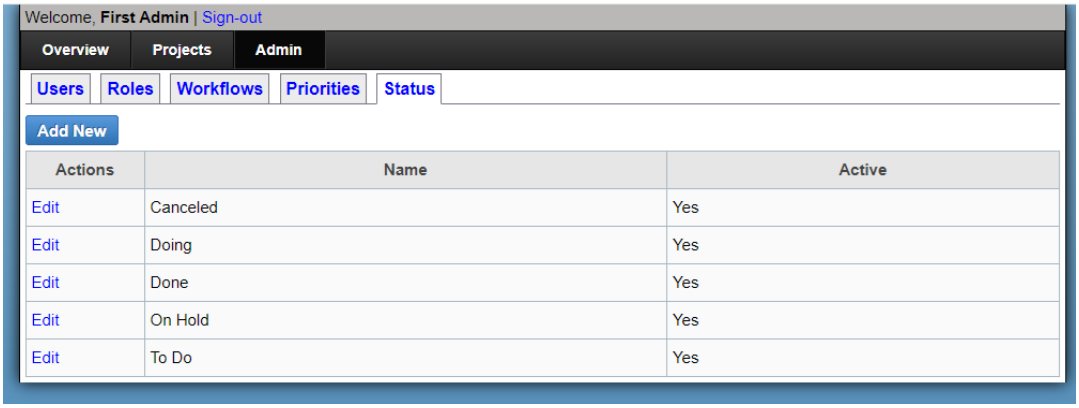

Рисунок 3.17 - Адміністрування статусу завдань у веб-додатку

3.7.2. Представлення інтерфейсу мобільної версії системи відстеження та ведення проектів

Рисунок 3.18 ілюструє інтерфейс автентифікації користувачів у мобільному додатку на платформі iOS.

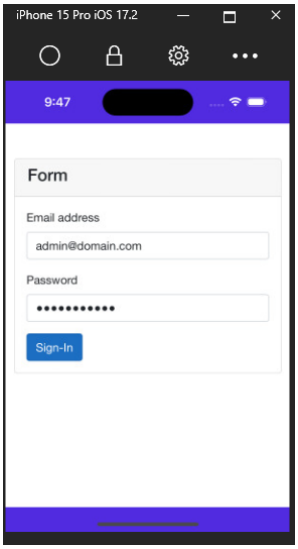

Рисунок 3.18 - Форма входу в iOS додаток

Рисунок 3.19 відображає основний інтерфейс мобільного додатку після успішного входу на платформі iOS.

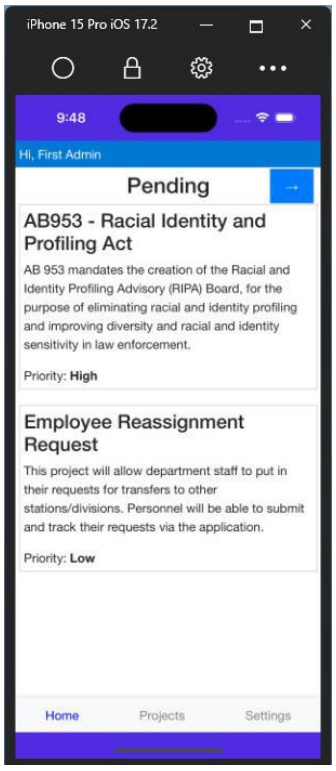

Рисунок 3.19 - Головний екран iOS

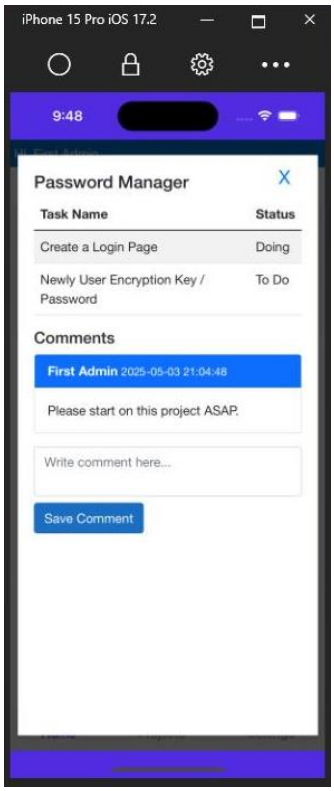

Рисунок 3.20 - Перегляд проекту менеджера паролів iOS

Рисунок 3.20 представляє екран перегляду деталей проекту з функціональністю управління паролями на iOS.

Рисунок 3.21 ілюструє перелік доступних проектів у мобільному додатку на платформі iOS.

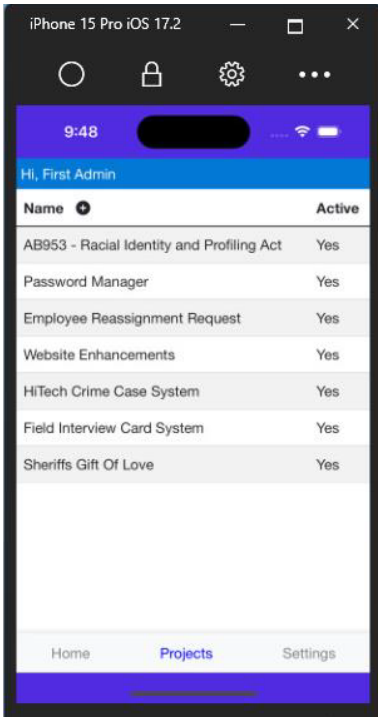

Рисунок 3.21 - Список проектів iOS

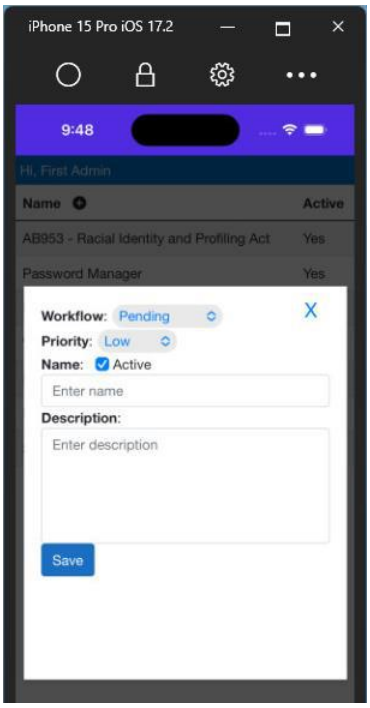

Рисунок 3.22 - Додавання проекту iOS

Рисунок 3.22 відображає форму для створення нового проекту у мобільному додатку на платформі iOS.

Рисунок 3.23 представляє інтерфейс для конфігурації параметрів та доступу до функцій профілю в мобільному додатку на iOS.

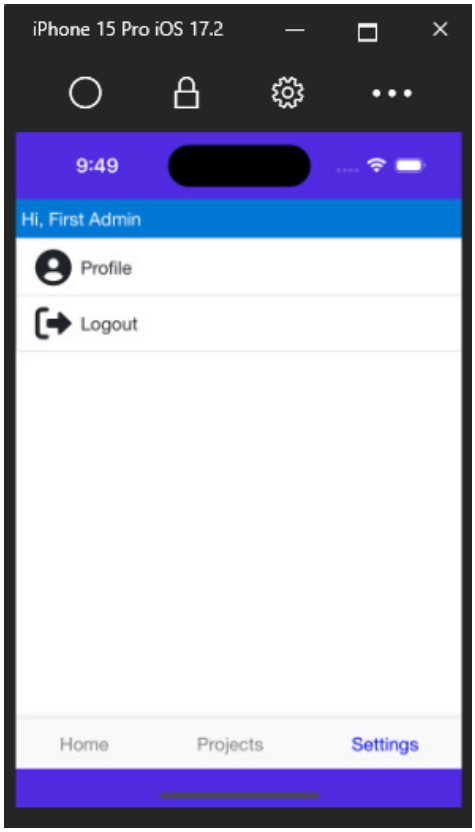

Рисунок 3.23 - Екран налаштувань iOS

3.8. Тестування системи

Як і в будь-якому проекті з розробки програмного забезпечення, тестування та випуск є дуже важливими етапами. Основна увага приділяється тому, щоб переконатися, що кожен компонент працює правильно перед тим, як система буде запущена. Ефективні методи тестування виявляють та вирішують проблеми до випуску програмного забезпечення, підвищуючи надійність користувацького досвіду.

Тестування є дуже важливим, оскільки воно виявляє проблеми, які можуть вплинути на роботу системи перед її запуском. Кілька методів тестування зосереджуються на різних компонентах системи.

3.8.1. Модульне тестування

Це вимагає тестування кожного компонента або модуля додатку окремо, щоб перевірити, чи працюють вони правильно. Модульне тестування має повністю охоплювати всі операції CRUD (Створення, Читання, Оновлення, Видалення) у межах моделей даних для системи відстеження проектів. Це важливо для забезпечення цілісності даних та точного виконання бізнес-логіки.

Рисунок 3.24 демонструє процес тестування шляхом натискання правою кнопкою миші на "UnitTestPjTrak", відображається контекстне меню. Вибір "Run Tests" запускає модульні тести проекту. Цей інструмент дозволяє розробникам легко запускати всі модульні тести в проекті для миттєвого зворотного зв'язку щодо цілісності та функціональності коду в середовищах розробки.

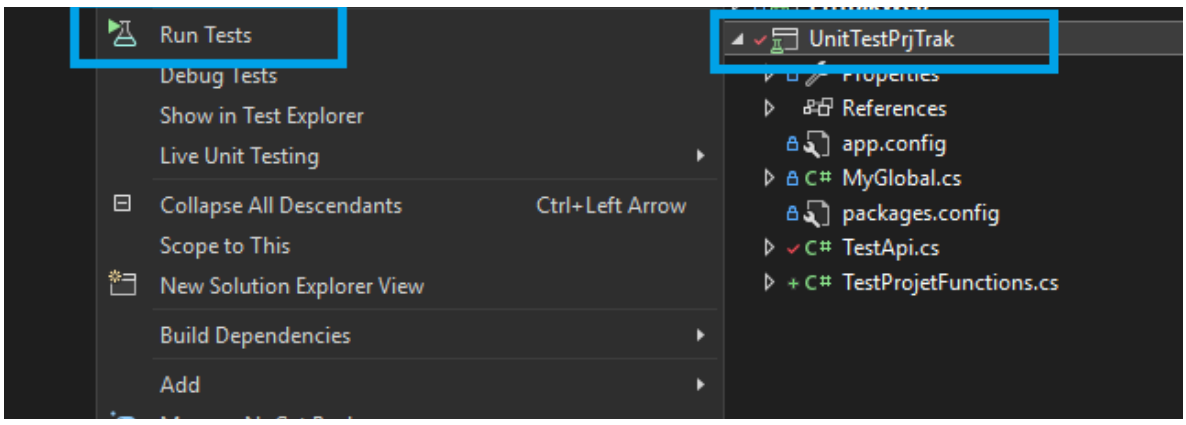

Рисунок 3.24 – Початок процесу тестування

Рисунок 3.25 ілюструє модульні тести для "Project Entity" з використанням операцій CRUD, показані на зображенні. Виконувались вони у порядку, заданому їх числовими префіксами (01, 02, 03, 04), забезпечуючи

створення перед читанням, читання перед оновленням та оновлення перед видаленням. Послідовність дій така: Test01CreateProject, Test02ReadProject, Test03UpdateProject та Test04DeleteProject. Test Explorer показує, що всі чотири тести пройшли без проблем та були виконані, вказуючи на те, що операції CRUD сутності Project, ймовірно, реалізовані правильно та добре працюють в умовах тестування.

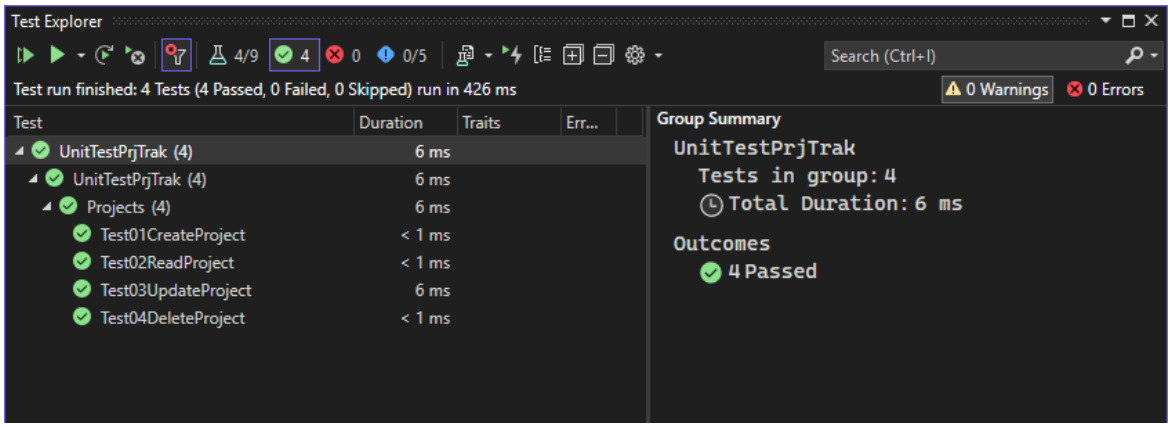

Рисунок 3.25 - Результат модульного тестування

Необхідно виконувати цю дію для кожної сутності у вашій системі, щоб гарантувати успішне завершення всіх тестів. Кожного разу, коли вноситься зміна до будь-якої сутності, модульні тести мають бути повторені для перевірки функціональності нової реалізації. Тут не буде описано детально про кожну сутність у системі відстеження проектів, але буде наведено, як створити модульний тест для сутності Project.

```
private Project _project = null;
[TestMethod]
(1) 0 references | M., Less than 5 minutes ago | 1 author, 1 change
public void ProjectCRUD()
{
    CreateProject();
    ReadProject();
    UpdateProject();
    DeleteProject();
}
```

Рисунок 3.26 - Модульне тестування сутності за операціями CRUD

Це залежить від того, як ви пишете свої тестові випадки. Немає правильного чи неправильного. Був намір знищити все, що було створено під час моїх модульних тестів.

Рисунок 3.26 ілюструє, що було оголошено null об'єкт, який використовується. У моєму випадку, _project дорівнює null. В даному випадку, буде використано цей об'єкт для операцій CRUD.

Рисунок 3.27 демонструє можливість створення об'єкта проекту. Тут пов'язано перших активних користувачів з створеним користувачем ID. Код пріоритету проекту з першою активною сутністю пріоритету та код робочого процесу з першою активною сутністю робочого процесу.

```
private void CreateProject()
{
    var newEntry = true;
    var project = new Project();
    project.setName("name");
    project.setDescription("description");
    project.setCreatedByUserId(MyGlobal.getProjectService().getAllActiveUsers()[0].getId());
    project.setPriorityCode(MyGlobal.getProjectService().getAllPriorities()[0].getCode());
    project.setWorkflowCode(MyGlobal.getProjectService().getActiveWorkflows()[0].getCode());
    project.setIsActive(true);
    project.setCreatedDate(DateTime.Now);
    project.setLastUpdatedDate(DateTime.Now);
    _project = MyGlobal.getProjectService().saveProject(project, newEntry);
    Assert.IsNotNull(_project);
}
```

Рисунок 3.27 - Модульне тестування створення сутності "Project"

Assert.IsNotNull(_project): Це перевіряє, що змінна _project не порожня після збереження проекту. Це допомагає переконатися, що процес збереження працював правильно.

Рисунок 3.28 ілюструє Assert у кожному методі, щоб переконатися, що він працює під час модульного тестування. Якщо якийсь із методів не вдається під час тесту, весь модульний тест ProjectCRUD не вдається.

```
1 reference
private void ReadProject()
{
    Assert.IsNotNull(_project);
    var project = MyGlobal.getProjectService().getProjectById(_project.getId());
    Assert.IsNotNull(project);
}

1 reference
private void UpdateProject()
{
    Assert.IsNotNull(_project);
    Random r = new Random(); int n = r.Next();
    var project = MyGlobal.getProjectService().getProjectById(_project.getId());
    project.setName(n.ToString());
    Assert.AreNotEqual<Project>(project, _project);
}
1 reference
private void DeleteProject()
{
    Assert.IsNotNull(_project);
    var projectId = _project.getId();
    MyGlobal.getProjectService().deleteProjectById(projectId);
    var project = MyGlobal.getProjectService().getProjectById(projectId);
    Assert.IsNull(project);
}
```

Рисунок 3.28 - Модульне тестування читання, оновлення та видалення сутності " Project ".

Рисунок 3.29 ілюструє результат перегляду з модульного тесту, який показує результати тестів та час виконання для кожного тесту та групи тестів.

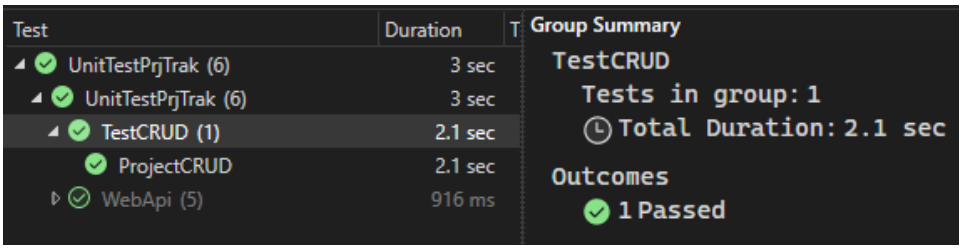

Рисунок 3.29 - Результат модульного тестування CRUD операцій для сутності " Project "

3.8.2. Тестування веб-API

Аналогічно модульному тестуванню, тестування Web API має вирішальне значення для верифікації безпеки, надійності та продуктивності веб-сервісів, особливо з огляду на їхню взаємодію з CRUD-операціями (Create, Read, Update, Delete), які становлять основу функціональності більшості веб-додатків.

Необхідно ретельно перевіряти тип та вміст усіх вхідних даних. Важливо також оцінювати, як API обробляє неприпустимі, неочікувані або відсутні дані, перевіряючи, чи генеруються при цьому коректні повідомлення про помилки. Окрім того, слід тестувати реакцію API на системні помилки, спричинені проблемами з мережею або базою даних, забезпечуючи повернення відповідних статусних кодів та повідомлень.

Рисунок 3.30 ілюструє результат перегляду з тесту веб-API, який показує результати тестів та час виконання.

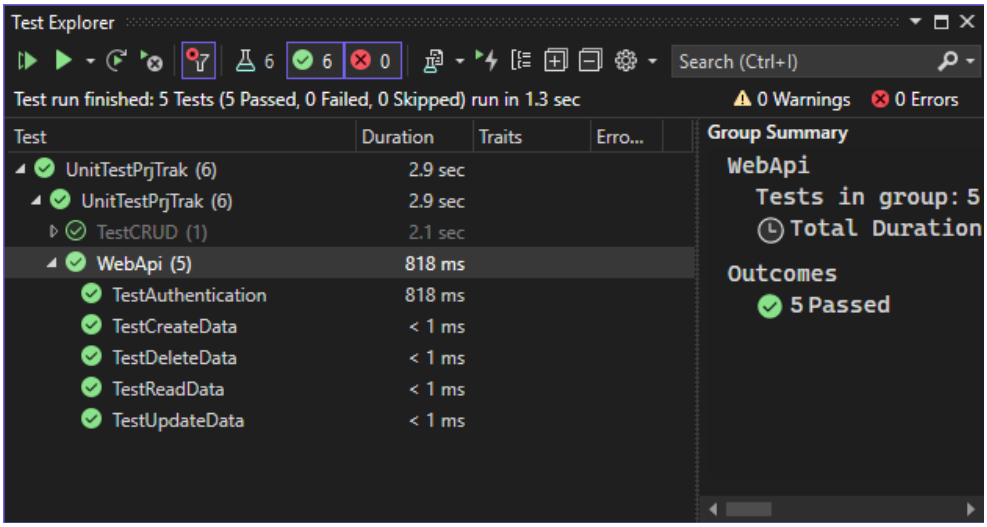

Рисунок 3.30 - Результат перегляду веб-API тестування

Завершення розробки истеми стало цінним досвідом, що підкреслив важливість застосування галузевих стандартів та сучасних технологій у сфері розробки програмного забезпечення. У рамках цього проекту успішно створено та розгорнуто повністю функціональну систему, яка інтегрує веб-додатки, мобільні додатки та веб-API в єдину архітектуру. Це досягнення демонструє здатність розробника створювати крос-платформні рішення та ефективно управляти складними архітектурами проектів, дотримуючись найкращих практик у програмній інженерії.

Використання фреймворку MVC 5 забезпечило ефективну структуру веб-додатку. Його архітектура "модель-представлення-контролер" сприяла

чіткому розділенню шарів, що значно покращило підтримуваність та масштабованість системи. Паралельно, .NET MAUI суттєво спростив процес розробки мобільних додатків, дозволивши створювати крос-платформні додатки з єдиної кодової бази. Цей підхід оптимізував розробку та управління додатками для платформ iOS та Android, забезпечивши узгодженість функціональності на різних пристроях та прискоривши загальний процес розробки.

ВИСНОВКИ

В результаті виконання дипломної роботи було здійснено повний цикл розробки ВЕБ-базованого рішення для ведення та управління проєктами, що охоплює аналіз предметної області, проектування архітектури системи та бази даних, реалізацію програмного продукту, а також його тестування.

На етапі аналізу предметної області було встановлено актуальність створення адаптивної системи управління проєктами, здатної ефективно реагувати на зміни вимог та умов реалізації проєктів. Було проведено порівняльний аналіз аналогічних систем, що дозволило виділити ключові функціональні та нефункціональні вимоги до розроблюваного рішення, а також визначити його цільову аудиторію, архітектурні особливості та технічні умови використання.

У рамках проектування архітектури системи було використано сучасні методи моделювання, зокрема діаграми варіантів використання та діаграми послідовностей. Це дозволило формалізувати логіку взаємодії користувачів із системою. Ретельно спроектовано базу даних із урахуванням відношень "багато-до-багатьох", що забезпечує гнучкість у збереженні та обробці інформації. Окрему увагу приділено проектуванню інтерфейсу користувача для веб- та мобільної версій, що сприяє підвищенню зручності та інтуїтивності використання.

На етапі програмної реалізації було застосовано архітектурний підхід трирівневої моделі, що забезпечує модульність, масштабованість та повторне використання компонентів. Для створення веб-додатку використано фреймворк MVC 5, а для мобільного додатку – платформу .NET MAUI. Реалізовано Web API для забезпечення взаємодії між клієнтськими додатками та серверною частиною. Усі компоненти системи інтегровані в єдине рішення з чітким розмежуванням бізнес-логіки, доступу до даних та представлення інформації.

У ході тестування системи проведено модульне тестування окремих компонентів та функціональне тестування Web API. Результати тестування підтвердили відповідність розробленого рішення поставленим вимогам щодо надійності, продуктивності та функціональності.

Таким чином, у межах дипломної роботи було розроблено повнофункціональну, масштабовану та адаптивну систему ведення проектів, яка може бути використана як у малих, так і у великих організаціях для підвищення ефективності управління проєктною діяльністю.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ahmad, S., & Khan, M. A. (2018). Design and Implementation of a Three-Tier Web Application using ASP.NET MVC. International Journal of Computer Applications, 179(4), 1-5.
2. Al-Ahmari, A. M. A., & Al-Amri, S. M. (2019). Developing a Mobile Application for Project Management using Xamarin.Forms and Web API. International Journal of Advanced Computer Science and Applications, 10(4), 112-118.
3. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley Professional.
4. Beck, K. (2000). Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.
5. Choudhary, A., & Gupta, P. (2020). A Comparative Study of Cross-Platform Mobile Application Development Frameworks: Xamarin vs. React Native vs. Flutter. International Journal of Engineering and Advanced Technology, 9(4), 2162-2168.
6. Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley Professional.
7. Conway, C. (2015). Designing RESTful APIs: A Comprehensive Guide. Journal of Web Engineering, 14(3), 193-210.
8. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
9. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 11. Gemino, A., & Wand, Y. (2005). Complexity and clarity in conceptual modeling: an empirical study. Data & Knowledge Engineering, 55(3), 301-322.

10. Gomaa, H. (2011). Software Modeling and Design: UML, Use Cases, Architectures, and Processes. Cambridge University Press.
11. Grossman, D., & Moshovos, A. (2006). An Introduction to Database Systems (8th ed.). Pearson Education.
12. Guha, R., & Gupta, A. (2019). Enhancing Software Quality through Unit Testing: A Comprehensive Review. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 5(2), 195-200.
13. IBM. (n.d.). What Is Three-Tier Architecture? Retrieved from https://www.ibm.com/think/topics/three-tier-architecture
14. Jawale, S., & Agrawal, H. (2017). Performance Evaluation of ASP.NET MVC 5 and ASP.NET Core. International Journal of Computer Applications, 175(8), 1-4.
15. Kalidindi, A. (2015). SQL Server 2014 Design and Development for Experts. Apress.
16. Khurana, H., Singh, J., & Kaur, R. (2016). Role of UML Diagrams in Software Development Life Cycle. International Journal of Computer Science and Mobile Computing, 5(4), 164-169.
17. Kirova, D., & Ilieva, S. (2019). Database Design Principles and Best Practices. Journal of Engineering Science and Technology Review, 12(1), 125-130.
18. Kumar, A., & Garg, R. (2018). Design and Implementation of a Mobile Application for Project Management. International Journal of Research in Engineering and Technology, 7(5), 1-6.
19. Lee, J., & Kim, Y. (2020). A Study on the Best Practices for RESTful API Design. Journal of Information Science and Engineering, 36(3), 603-614.
20. McConnell, S. (2004). Code Complete (2nd ed.). Microsoft Press.

21. Mesbah, A., & van Deursen, A. (2009). A survey on web API testing. Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, 391-396.
22. Microsoft. (n.d.). .NET MAUI Documentation. Retrieved from https://learn.microsoft.com/en-us/dotnet/maui/
23. Microsoft. (n.d.). ASP.NET MVC. Retrieved from https://learn.microsoft.com/en-us/aspnet/mvc/
24. Mitchell, S. (2007). ASP.NET 3.5 Website Programming: Problem - Design - Solution. Wrox Press.
25. Moser, M., & Wirsing, M. (2014). A systematic review of RESTful API design guidelines. Journal of Systems and Software, 96, 203-216.
26. Ralph, P., & Wand, Y. (2009). A Proposal for a Formal Definition of the Design Concept in Information Systems. Proceedings of the 2009 ACM SIGMIS Conference on Computers and People Research, 17-27.
27. Sharma, P., & Gupta, M. (2017). Importance of Unit Testing in Software Development. International Journal of Computer Science and Engineering, 5(4), 160-162.
28. Somiari, S. N., & Oruwari, T. A. (2021). Three-Tier Architecture for Web-Based Applications: A Review. International Journal of Computer Science and Information Technology Research, 9(1), 1-8.
29. Stroustrup, B. (2013). The C++ Programming Language (4th ed.). Addison-Wesley Professional.
30. Trusov, I., & Demchenko, Y. (2019). Analysis of .NET MAUI as a Cross-Platform Framework for Mobile Application Development. International Journal of Computer Engineering and Technology (IJCET), 10(3), 101-108.
31. Vinod, V. (2020). Relational Database Design Principles and Best Practices for Scalable Applications. International Journal of Engineering Research & Technology, 9(7), 51-54.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Розробка ВЕБ-базованого рішення ведення проектів ”

Обсяг пояснювальної записки: 80 аркушів.

Дата закінчення роботи: 10 червня 2025 р.

Підпис студента _____