ДИПЛОМНА РОБОТА

БАКАЛАВРА

ДРБ. ІП - 34.00.00.000 ПЗ

Група ІП-22-1К

Полек Богдан-Августин

2024

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: "Розробка програмного забезпечення системи управління проектними ресурсами"

(назва згідно з наказом ректора)

спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 34.00.00.000 ПЗ

(позначення)

Консультанти:

з питань нормоконтролю

асист. /М.М. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

Студент

ІП-22-1К /Б-А.В.Полек/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /Р.Б.Вовк/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

2024
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2024 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Полеку Богдану-Августину Васильовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “Розробка програмного забезпечення системи управління проектними ресурсами”

керівник проекту (роботи) Вовк Роман Богданович, доцент, к.т.н.

затверджені наказом закладу вищої освіти від “ 7 ” червня 2024 р. № 347/7

2. Строк подання студентом проекту (роботи) 17 червня 2024 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 та інформаційних технологій управління проектами
2. Моделювання процесів та алгоритмічного забезпечення системи управління проектами
3. Розробка моделі бази даних та її опис
4. Програмна реалізація системи управління проектними ресурсами
5. Розробка інтерфейсу користувача системи управління проектними ресурсами

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

1. Діаграма варіантів використання (рис. 2.1)
2. Схема бізнес-процесу реєстрації (рис. 2.2)
3. Схема бізнес-процесу авторизації (рис. 2.3)
4. Схема бізнес-процесу створення проекту (рис. 2.4)
5. Схема бізнес-процесу призначення члена команди (рис. 2.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **22 травня 2024 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 та інформаційних технологій управління проектами	25.05.2024	виконано
2	Моделювання процесів та алгоритмічного забезпечення системи управління проектами	29.05.2024	виконано
3	Розробка моделі бази даних та її опис	30.05.2024	виконано
4	Програмна реалізація системи управління проектними ресурсами	04.06.2024	виконано
5	Розробка інтерфейсу користувача системи управління проектними ресурсами	10.06.2024	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2024	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В даній роботі розглядається процес розробки програмного забезпечення для системи управління проектами та проектними ресурсами. Метою розробки є створення інструменту, який забезпечить ефективне планування, організацію, контроль та аналіз проектної діяльності, а також управління ресурсами, залученими у проектах. Робота включає аналіз існуючих систем управління проектами, визначення їхніх сильних та слабких сторін, а також визначення вимог до нової системи. Описуються етапи проектування та розробки програмного забезпечення, включаючи архітектурні рішення, вибір технологій та інструментів, реалізацію функціональних модулів, а також тестування та впровадження системи. В результаті роботи була створена система, яка значно покращує процес управління проектами та ресурсами, підвищує продуктивність команди та забезпечує своєчасне виконання проектів з оптимальним використанням ресурсів.

КЛЮЧОВІ СЛОВА: УПРАВЛІННЯ ПРОЕКТАМИ, ВЕБ-ЗАСТОСУНОК, АВТОМАТИЗАЦІЯ, УПРАВЛІННЯ ЗАДАЧАМИ, МЕНЕДЖМЕНТ, БАЗА ДАНИХ, УПРАВЛІННЯ РЕСУРСАМИ

ANNOTATION

This work examines the process of software development of project management systems and project resources. The development method is the creation of a tool that provides effective planning, organization, control and analysis of project activities, as well as management of resources involved in projects. The work includes analysis of existing project management systems, determination of their strengths and weaknesses, as well as determination of requirements for a new system. The stages of software design and development are described, including architectural decisions, selection of technologies and tools, implementation of functional modules, as well as testing and implementation of systems. As a result of the work, a system was created that significantly improves the process of project and resource management, team productivity and ensures timely implementation of projects with optimal use of resources.

KEYWORDS: PROJECT MANAGEMENT, WEB APPLICATION, AUTOMATION, TASK MANAGEMENT, MANAGEMENT, DATABASE, RESOURCE MANAGEMENT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ ПРОЕКТАМИ 11
1.1. Особливості інформаційної системи управління проектами 11
1.2. Аналіз існуючих рішень 13
1.2.1. Менеджер проектів Microsoft Project 13
1.2.3. Менеджер завдань Trello 17
1.2.4. Інструмент Kanbanize 18
1.2.5. Менеджер проектів Asana 20
1.3. Загальні формулювання проблеми управління проектами, дерево завдань, дерево проблем, дерево результатів 21
1.4. Концепція побудови автоматизованої інформаційної системи управління проектами 24

2. МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА РОЗРОБКА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ ПРОЕКТНИМИ РЕСУРСАМИ 29
2.1. Розробка та опис діаграм використання програмної системи 29
2.2. Моделювання основних процесів розроблюваної системи 36
2.3. Вибір архітектури програмного забезпечення 44
2.4. Розробка ER діаграми сутностей 51
2.5. Розробка моделі бази даних та її опис 53
2.6. Вибір інструментів для розробки 59

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПРОЕКТНИМИ РЕСУРСАМИ 68
3.1. Розробка структурної схеми класів 68
3.2. Розробка інтерфейсу для реєстрації та авторизації користувача 70
3.3. Розробка інтерфейсу користувача системи управління проектними ресурсами 73
3.4. Реалізація процесів щодо забезпечення безпеки даних 78

ВИСНОВКИ 80
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 81
ДОДАТКИ
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI – Artificial intelligence, штучний інтелект

ML – Machine learning, машинне навчання

PWA – Progressive web app, поступовий веб-застосунок

CPM – Critical path method, метод критичного шляху

PERT – Program (Project) evaluation and review technique, техніка оцінювання та аналізу програм

ВСТУП

Управління проектом є невід'ємною складовою успішності бізнес процесів. Застосування різноманітних прийомів та інструментів дозволяє досягти поставлених проектних цілей і задовільнити потреби клієнтів. Важко уявити сучасну організацію, котра не використовує проектне планування та менеджмент задач на проекті.

Більшість проектних менеджерів зосередженні на максимальній підтримці конкретного проекту і надають його учасникам широкий набір функцій для організації проектної роботи. У свою чергу менеджери проектів отримують певний перелік необхідних інструментів для організації розподілення завдань та делегування ролей серед учасників команди.

Проте все частіше у зв'язку з розширенням ринку і значним зростанням кількості проектів які виконують лідируючі світові компанії, все частіше виникає потреба у груповому управлінні проектами.

Окрім зовсім очевидної функції підвищення продуктивності керівників проекту, а також створення комфортних умов задля забезпечення максимального управління часом; групове управління також сприяє більш ефективному проектному планування, а також дозволяє заощадити значну кількість різноманітних ресурсів, як людських, так і матеріальних

У сучасному швидкоплинному бізнес-середовищі успішне управління проектами має вирішальне значення для досягнення цілей організації. Однак управління великими та складними проектами є складним завданням, яке вимагає ефективного відстеження завдань, термінів виконання та витрат. Багато організацій досі покладаються на застарілі ручні методи, такі як електронні таблиці, для керування своїми проектами, що часто призводить до помилок і непорозумінь і має значні наслідки для успіху проекту.

Ручним методам управління проектами бракує видимості в реальному часі, що ускладнює моніторинг прогресу, визначення потенційних перешкод

і прийняття обґрунтованих рішень. Обмін оновленнями проекту із зацікавленими сторонами стає проблематичним без централізованої системи відстеження та обміну інформацією. Крім того, спілкування між співробітниками та розподіл ресурсів є ключовими факторами, які необхідно оптимізувати для успішного управління проектом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ УПРАВЛІННЯ ПРОЕКТАМИ

1.1. Особливості інформаційної системи управління проектами

Інформаційна система керування проектами – це важливий компонент корпоративної системи керування проектами, істотно підвищує ефективність проектного менеджменту в організації. Вона може включати різні модулі в залежності від завдань, що вирішуються в рамках корпоративної системи керування проектами. Інформація – це зібрані, оброблені і розподілені дані, які є корисними для прийняття рішень, лише коли інформація буде надана своєчасно, за призначенням і в зручній формі. Це вирішується за допомогою використання сучасних інформаційних технологій в рамках системи керування проектами.

Інформаційна система управління проектом-організаційно технологічний комплекс методичних, технічних, програмних та інформаційних засобів, спрямований на підтримку і підвищення ефективності процесів планування та керування проектами в організації, налаштований з урахуванням корпоративних стандартів УП та потреб учасників.

Програмне забезпечення для керування проектами - клас комп'ютерних програм, розроблених спеціально для підтримки основних процесів управління, включаючи планування і контроль термінів, витрат, комунікації та ін Очевидно, що більшість управлінських процесів у рамках проекту мають на увазі використання комп'ютерів і засобів зв'язку. Більше того, можна стверджувати, що від моменту зародження і до наших днів розвиток методів УП та їх практичне застосування в чому визначалися розвитком інформаційних технологій.

Проект має певні властивості:

1. Проект завжди має чітко визначену мету, яка виражається в отриманні певного результату. Досягнення цього результату означає успішне завершення і закінчення проекту.

2. Проект має чітко окреслене початок, що збігається з початком першої роботи, спрямованої на досягнення поставленої мети. Початок може задаватися директивно, або розраховуватися в результаті складання плану робіт за проектом.

3. Проект має чітко окреслений кінець, який збігається з кінцем останньої роботи, спрямованої на отримання заданого результату. Як і початок, кінець проекту може задаватися директивно, або розраховуватися при складанні плану робіт.

4. Проект виконується командою, до складу якої входить керівник проекту, менеджери, виконавці. Крім основної команди в ньому можуть брати участь допоміжні виконавці, команди і організації, які залучаються на тимчасовій основі для виконання окремих робіт.

5. При реалізації проекту використовуються матеріальні ресурси, їх номенклатура та кількість визначаються характером проекту і роботами, що входять у нього.

6. Проект має бюджет. Вартість проекту складається з вартості витрачених матеріальних ресурсів, витрат по оплаті праці команди, що реалізує його й інших витрат, пов'язаних з особливостями конкретних видів робіт.

Життєвий цикл проекту – це проміжок часу між моментами його початку і завершення. Він ділиться на чотири фази:

1) концептуальна фаза – формулювання цілей, аналіз інвестиційних можливостей, техніко-економічне обґрунтування і планування проекту.

2) Фаза розробки – включає визначення структури робіт і виконавців, побудова календарних графіків робіт, бюджету проекту, розробку проектно-

кошторисної документації, переговори й укладання контрактів з підрядниками і постачальниками.

3) Фаза виконання – роботи з реалізації проекту, у тому числі будівництво, маркетинг, навчання персоналу і т.п.

4) Фаза завершення – приймальні випробування, дослідна експлуатація і здача проекту в експлуатацію.

Результат проекту – це деяка продукція або корисний ефект, створювані в ході реалізації проекту. В залежності від мети проекту, в якості результату можуть виступати: наукова розробка, новий технологічний процес, програмний засіб, будівельний об'єкт, реалізована навчальна програма, реструктурована компанія, сертифікована система якості і т.д.

Про успішність проекту судять по тому, наскільки його результат відповідає за своїми витратним, дохідним, інноваційним, якісним, тимчасовим, соціальним, екологічним та іншим характеристикам запланованому рівню.

1.2. Аналіз існуючих рішень

1.2.1. Менеджер проектів Microsoft Project

Microsoft Project (MS Project) — це комплексне програмне забезпечення — система управління проектами і спосіб оптимізації управління портфелями проектів, який дозволяє планувати і контролювати проектну діяльність організацій. Для цього застосовуються вбудовані шаблони, інструменти для різних рівнів аналітики і статистики, засоби управління робочим часом. MS Project найчастіше використовується як програмне рішення для середнього та великого бізнесу.

Microsoft пропонує своїм користувачам декілька варіантів ліцензії для отримання цього продукту. Варто також зазначити, що до стандартної

підписки Office 365 (від недавно Microsoft 365) MS Project не входить навіть на рівні Business преміум, і корпорації змушені купувати ліцензію окремо.

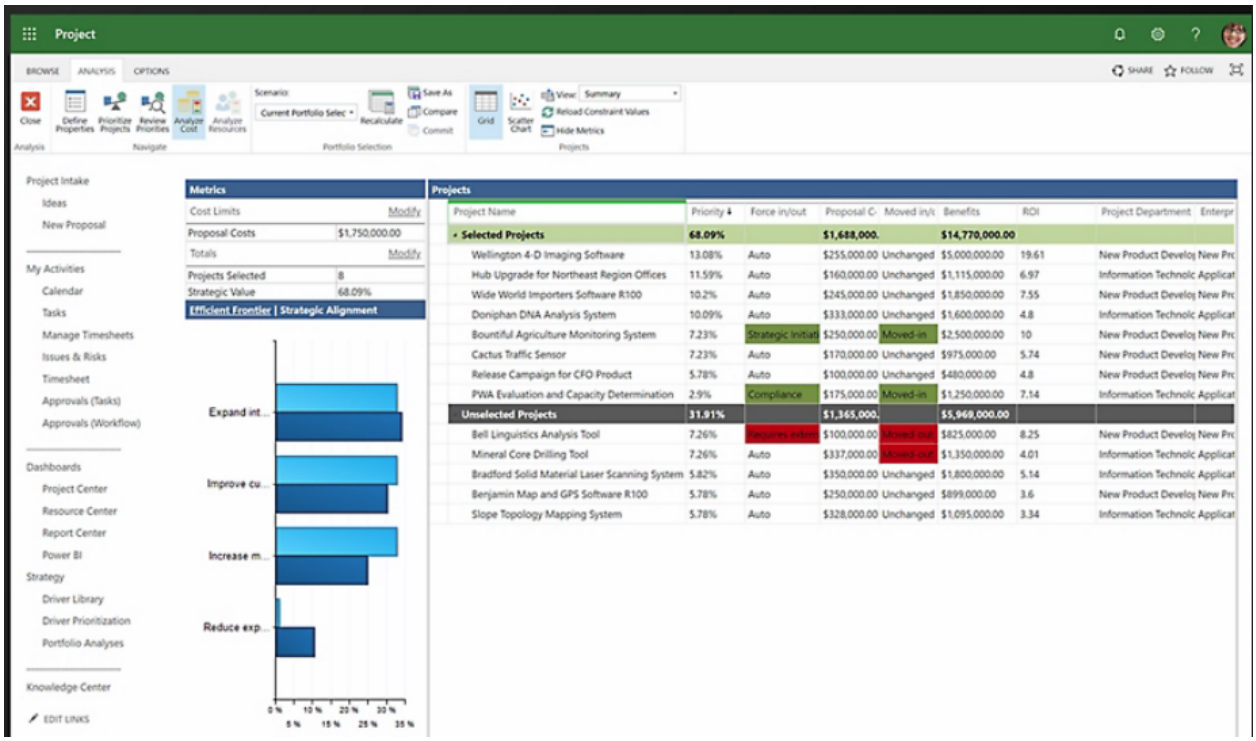


Рисунок 1.1 – Інтерфейс Microsoft Project

Варіанти ліцензування:

— Project Plan 3 — $30.00 з користувача за місяць;

— Project Plan 5 (розширені функції) — $55.00 з користувача за місяць;

— Project Standard 2019 — постійна ліцензія за $620.00 з користувача;

— Project Professional 2019 — постійна розширена ліцензія за $1030.00 з користувача;

Наразі MS Project не має свого програмного рішення для мобільних пристроїв на базі операційної системи Android, проте користувачі можуть скористатися веб-версією системи з обмеженим функціоналом.

1.2.2 Atlassian Jira

Jira - ще один відомий гнучкий інструмент розробки програмного забезпечення. Дозволяє керувати всіма етапами розробки ПЗ, починаючи з

самого початку, коли планується проєкт, і під час роботи над ним, що включає кілька менших завдань, і до моменту випуску продукту та передачі його клієнту. Jira застосовується до різних гнучких методів управління проєктами, незалежно від того, віддаєте перевагу Скрам чи Канбан.

Такі гнучкі інструменти планування, як Jira, дозволяють мати повну карту процесу розвитку. Все, що вам потрібно зробити, або те, що вже виконується, можна додавати та контролювати в системі. Кожне завдання відокремлене і ним може керувати працівник, який виконує його сам.

На рисунку 1.2 продемонстровано робочий стіл веб-додатку.

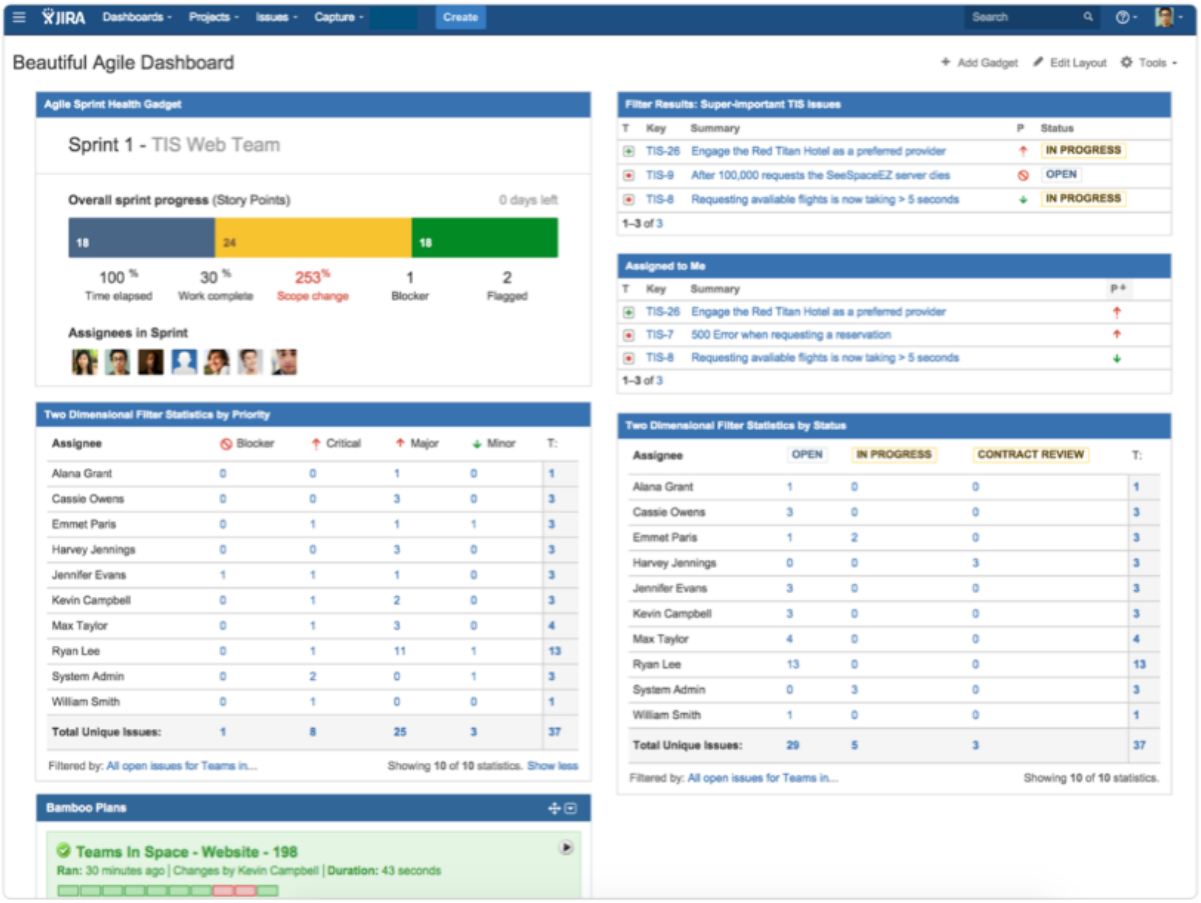


Рисунок 1.2 – Інтерфейс Atlassian Jira

Створюючи план проєкту, ви можете додавати історії користувачів, а також розбивати основне завдання на менші. Усі проблеми, що виникають у

процесі, також відстежуються, що дозволяє звертати увагу та знаходити рішення вчасно.

Спілкування між членами команди в Jira гарно організоване. Можна звернутися до цілої групи робітників, але також зв’язатися з певним членом команди.

Додаткові інструменти, які пропонує Jira, включають інформацію з різних бізнес-додатків. Вони можуть бути використані для підвищення ефективності роботи, але також надають розробникам більше ідей та деталей щодо сфери, в якій вони працюють. Користувачі Jira також мають можливість звернутися до команди підтримки, а також допоміжних матеріалів.

Немає конкретного гаджета, який потрібен, якщо ви хочете використовувати Jira. Існує варіант програмного забезпечення, який зручніше використовувати з персонального комп’ютера. Але якщо у вас його немає під рукою, ви можете увійти в систему за допомогою мобільного додатка та відстежувати весь проєкт незалежно від того, де ви знаходитесь.

Хоча це виглядає надзвичайно привабливим гнучким програмним забезпеченням для проєкту, у Jira також є кілька недоліків:

• Вам доведеться витратити досить багато часу, щоб звикнути як правильно з цим ПЗ працювати. Jira також не найпростіший інструмент з точки зору налаштування і може вимагати спеціалістів для встановлення;

• Jira дорога. Незважаючи на те, що великі компанії можуть дозволити собі платити щомісяця запитувану плату, для невеликих підприємств та проєктів це можуть бути витрати, які не варті результату.

Але в цілому виконуються інші вимоги, Jira ідеально підходить для колективної роботи, оскільки вона одночасно покращує командне спілкування та дозволяє визначати пріоритети та розподіляти завдання для кожного учасника.

Також доступне виявлення проблем, що є надзвичайно важливою функцією, оскільки дозволяє своєчасно їх виправляти та підвищує ефективність.

1.2.3. Менеджер завдань Trello

Якщо ж говорити про системи для мобільних пристроїв та планшетів, варто розглянути Trello — менеджер завдань для організації власного часу, а також створення колективних задач. Даний програмний продукт розробила компанія Fog Creek Software, позиціонувавши його як багатоплатформне рішення для управління проектами і задачами. Trello доступний для Android, iOS і Windows, також можна скористатися веб-версією.

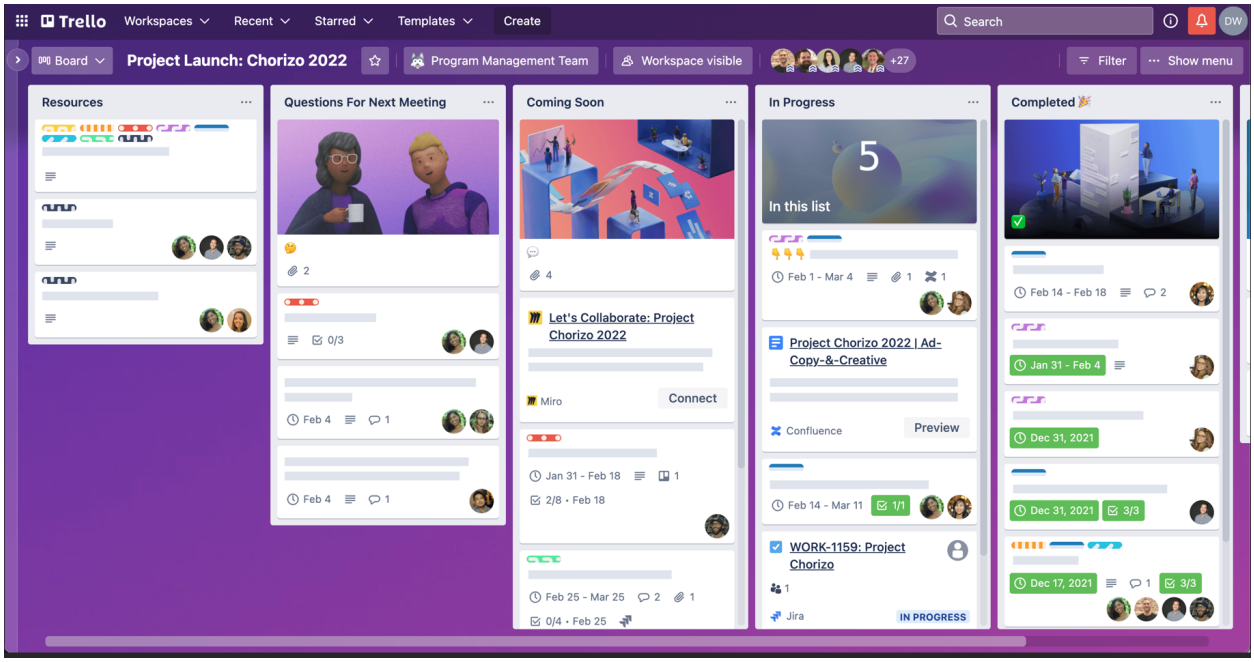


Рисунок 1.3 – Інтерфейс Trello

Більшість функцій спрямована саме на особисте користування, тоді як робота в проектах навіть для платних версій має досить обмежений функціонал. Окрім того відсутні рівні задач і розподілення ролей, менеджери проектів не мають окремого рівня доступу, а робота в декількох проектах одночасно не передбачена.

Trello передбачає як безкоштовний варіант для особистого користування і користування в командах до 10 користувачів, так і платні версії, а саме:

— Business Class — $ 9.99 з користувача за місяць;

— Enterprise — $ 20.83 з користувача за місяць;

Trello ідеально підходить для одного користувача, або ж для розробки певних продуктів у невеликих колективах для оптимізації часу.

1.2.4. Інструмент Kanbanize

Kanbanize - це гнучкий інструмент розробки програмного забезпечення, який підвищує продуктивність праці шляхом створення візуальної карти робочого процесу. Це більше, ніж просто дошка в стилі Канбан, оскільки вона також включає автоматизацію бізнес-процесів і створює середовище для продуктивної та ефективної роботи розробників.

За винятком дошки Канбан, яка може бути встановлена в точності до потреб вашого проєкту та інструментів автоматизації бізнесу, Kanbanize також пропонує аналітичну систему, яка полегшує роботу менеджера проєкту, а також інструменти планування, що оптимізують терміни виконання роботи. Відстежуються різні сторони завершення проєкту та пропонуються дані, які згодом оптимізуються та впроваджуються в робочий процес.

Це також особливо корисно для між функціональних команд, оскільки дозволяє створювати кілька одночасних робочих процесів. Щоденні та довгострокові заходи можна запланувати за допомогою Kanbanize. Кожен з них може бути представлений простим завданням з чіткими вимогами, а також позначеним як найвищий пріоритет і призначеним конкретному працівникові. Залежності між завданнями також керовані, що дозволяє розробникам звертати увагу на завдання в необхідному порядку.

На рисунку 1.4 продемонстровано головний екран додатку.

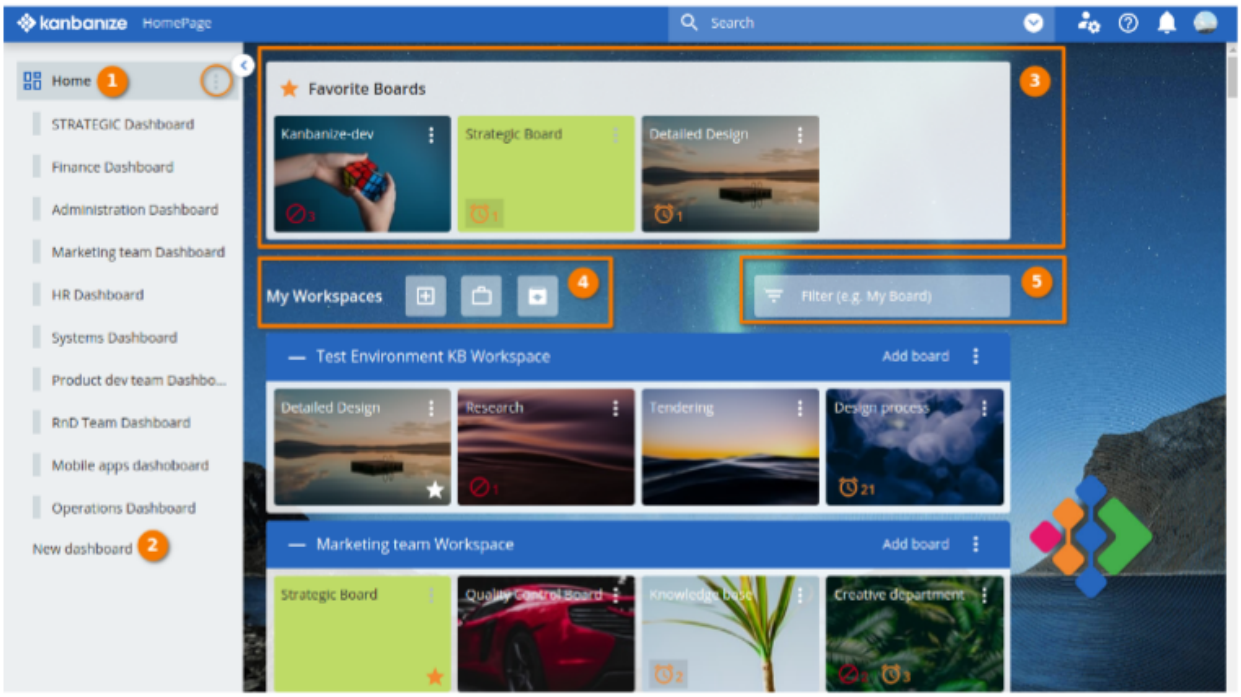


Рисунок 1.4 - Веб-додаток Kanbanize

Kanbanize має обмеження. Це означає, що працівники не перевантажені роботою і не спокушаються виконувати кілька завдань одночасно. Також їм набагато простіше приділяти достатньо уваги кожному завданню і виконувати його без помилок.

У той же час, система забезпечує велику прозорість серед команди, а також у компанії в цілому. Користувачам дуже просто змінити робоче поле, налаштувати кількість карток та обмеження роботи в процесі. Ще однією важливою особливістю Kanbanize є те, що він показує роботу та прогрес у реальному часі. Це означає, що більше не потрібно оновлювати статус задач від членів команди, і тому вони можуть повністю зосередитися на завданні. Це також означає, що набагато легше передбачити реалізацію проєкту, а також спланувати наступні етапи завершення розробки продукту. Розробники виконують лише цінну роботу, яка впливає на кінцевий результат.

За необхідності Kanbanize можна інтегрувати з іншими інструментами, з якими працює ваша компанія. До них належать Google Drive, Jira, Dropbox

та інші. Це особливо зручно для великих компаній з декількома відділами, які можуть використовувати різне програмне забезпечення.

1.2.5. Менеджер проектів Asana

Ще один з програмних додатків для управління проектами на базі операційних систем Android та iOS це Asana від однойменної компанії-розробника. Система була створена колишніми працівниками компанії Facebook Дастіном Московіцем і Джастіном Розенштейном.

Asana позиціонується як розширений планувальник завдань для їх виконання в командах. На відміну від Trello, тут присутні певний функціонал для забезпечення зв'язку між користувачами. Окрім того додаток пропонує і особисті списки завдань.

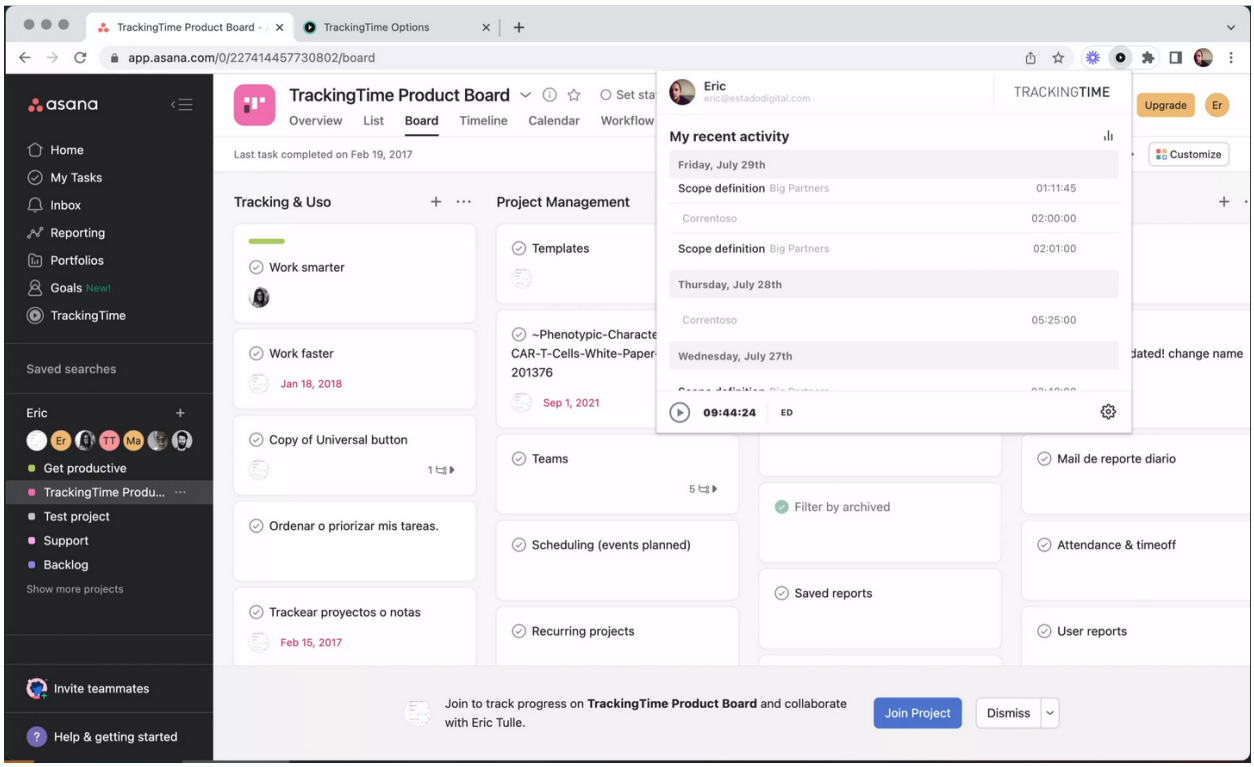


Рисунок 1.5 – Інтерфейс Asana

Попри все рівні доступу все ще погано організовані, тоді як багаторівневі завдання недоступні. Asana присутня як у безкоштовному

варіанті до 15 користувачів, так і в розширених варіантах, зокрема із функцією пріоритетів завдань і проекту.

Ліцензії Asana:

— Premium Plan — $ 9.99 з користувача за місяць;

— Business Plan — $ 19.99 з користувача за місяць;

— Enterprise Plan — $ 29.99 з користувача за місяць;

Система не передбачає групового управління проектами, але надає певні функції для менеджерів проектів, за допомогою яких вони можуть управляти діяльністю одного проекту з делегуванням зобов'язань і розподіленням робочого часу, до того ж модель Kanban (Канбан) тут реалізована на значному рівні, що надає змогу ефективно організувати дошки завдань і розроблювати проектні цілі та розставляти пріоритети.

1.3. Загальні формулювання проблеми управління проектами, дерево завдань, дерево проблем, дерево результатів

Проблема, яку проєкт прагне вирішити, полягає в тому, що багато команд мають труднощі з координацією роботи, обміном інформацією та контролем над ходом виконання завдань під час управління проєктами. Ця проблема посилюється зі збільшенням розміру команди та складності проєкту. Для більш наочного представлення проблеми, завдань і результатів можна побудувати дерева завдань, проблем і результатів.

Дерево завдань (рисунок 1.6) включає такі основні елементи: управління завданнями, управління ресурсами, управління комунікаціями та управління ризиками. Кожне з цих завдань включає підзадачі, які необхідно вирішити для досягнення спільної мети. Дерево завдань для нашого проєкту:

1) управління проєктами:

− створення нового проєкту;

− редагування інформації про проєкт;

− видалення проєкту;

− перегляд списку активних проєктів;

− перегляд списку завершених проєктів.



Рисунок 1.6 - Дерево завдань

2) планування проєкту:

− визначення цілей і завдань проєкту;

− розподіл ресурсів на завдання;

− створення графіка виконання проєкту;

− управління залежностями між завданнями.

3) управління завданнями:

− створення нового завдання;

− редагування інформації про завдання;

− призначення відповідальних виконавців;

− відстеження прогресу виконання завдання;

− управління пріоритетами та термінами завдань.

4) управління ресурсами:

− облік доступних ресурсів (людських, матеріальних і фінансових);

− розподіл ресурсів на завдання та проєкти;

− моніторинг використання ресурсів та їх доступності.

5) комунікація та спільна робота:

− обмін повідомленнями та коментарями щодо проєкту та завдань;

− спільне редагування документів і файлів;

− повідомлення про зміни та події в проєкті;

− можливість проведення онлайн-зустрічей і нарад;

− моніторинг та звітність;

− відстеження прогресу проєкту та виконання завдань;

− створення звітів про статус проєкту;

− аналіз витрат часу і ресурсів на завдання;

− візуалізація даних про проєкт у вигляді графіків і діаграм.

6) управління ризиками та змінами;

− ідентифікація та оцінка ризиків проєкту;

− облік та управління змінами в проєкті.

7) аналіз та оцінка впливу змін на проєкт.

Дерево проблем відображає проблеми, які виникають при управлінні проєктами, включаючи відсутність координації між членами команди, труднощі з відстеженням прогресу завдань, труднощі з управлінням ресурсами та ризиками. Дерево проблем проєкту:

1) відсутність централізованої системи управління проєктами;

2) недостатня прозорість і контроль завдань;

3) недостатня комунікація та співпраця між учасниками проєкту;

4) низька ефективність планування та розподілу ресурсів;

5) неефективне управління ризиками та змінами в проєктах.

Дерево результатів показує очікувані результати від реалізації проєкту, включаючи поліпшене управління завданнями, ефективне управління ресурсами, поліпшену комунікацію в команді і зниження ризиків. Дерево результатів проєкту:

1) покращена ефективність управління проєктами;

2) підвищення прозорості та контролю завдань;

3) покращена комунікація та співпраця між учасниками проєкту;

4) оптимізація планування та розподілу ресурсів;

5) ефективне управління ризиками та змінами в проєктах.

Управління проєктами включає планування, організацію, управління та контроль над ресурсами для досягнення конкретних цілей у рамках проєкту. Це включає такі аспекти, як управління завданнями, ресурсами, комунікаціями та ризиками. У даній області важливо забезпечити ефективне планування, організацію та виконання проєктів, щоб досягти поставлених цілей в задані терміни і з обмеженими ресурсами. Автоматизація управління проєктами на малих підприємствах допомагає оптимізувати процеси, підвищити прозорість і контроль, скоротити час і ресурси, а також поліпшити комунікацію і співпрацю між учасниками проєктів.

1.4. Концепція побудови автоматизованої інформаційної системи управління проектами

Ефективність діяльності компанії, а саме реалізація ІТ-проектів, багато в чому залежить від системи, яка використовується для управління розробкою ІТ-проектів. Мати сучасну та багатофункціональну систему управління проектами є необхідним критерієм для проведення ефективної діяльності ІТ-компанії.

До основних процесів управління здійснення ІТ-проектів входять:

- Планування обсягу робіт, необхідних для виконання проекту і досягнення його цілей.

- Організацію ресурсів для їх виконання в межах встановленого бюджету і строків.

- Впровадження розробленої програми дій.

- Контроль за виконанням плану або його коригування у разі необхідності.

На сьогоднішній день існує велика кількість ІТ-компаній, які надають різноманітні послуги, починаючи з підтримки користувачів та закінчуючи

розробками високоефективних та масштабних систем. Всі ці компанії, в процесі розробки проектів, використовують інформаційні системи для управління розробки проектами. Для того, щоб забезпечити конкурентноспроможність компанії ця система повинна бути сучасною та багатофункціональною.

Згідно з структурою типового компанії та процесом розробки проектів для візуального представлена взаємин між особами, які відбуваються в IT-компанії під час розробки проектів використаємо модель кооперації, яка представлена у вигляді схеми на рисунку 1.7.

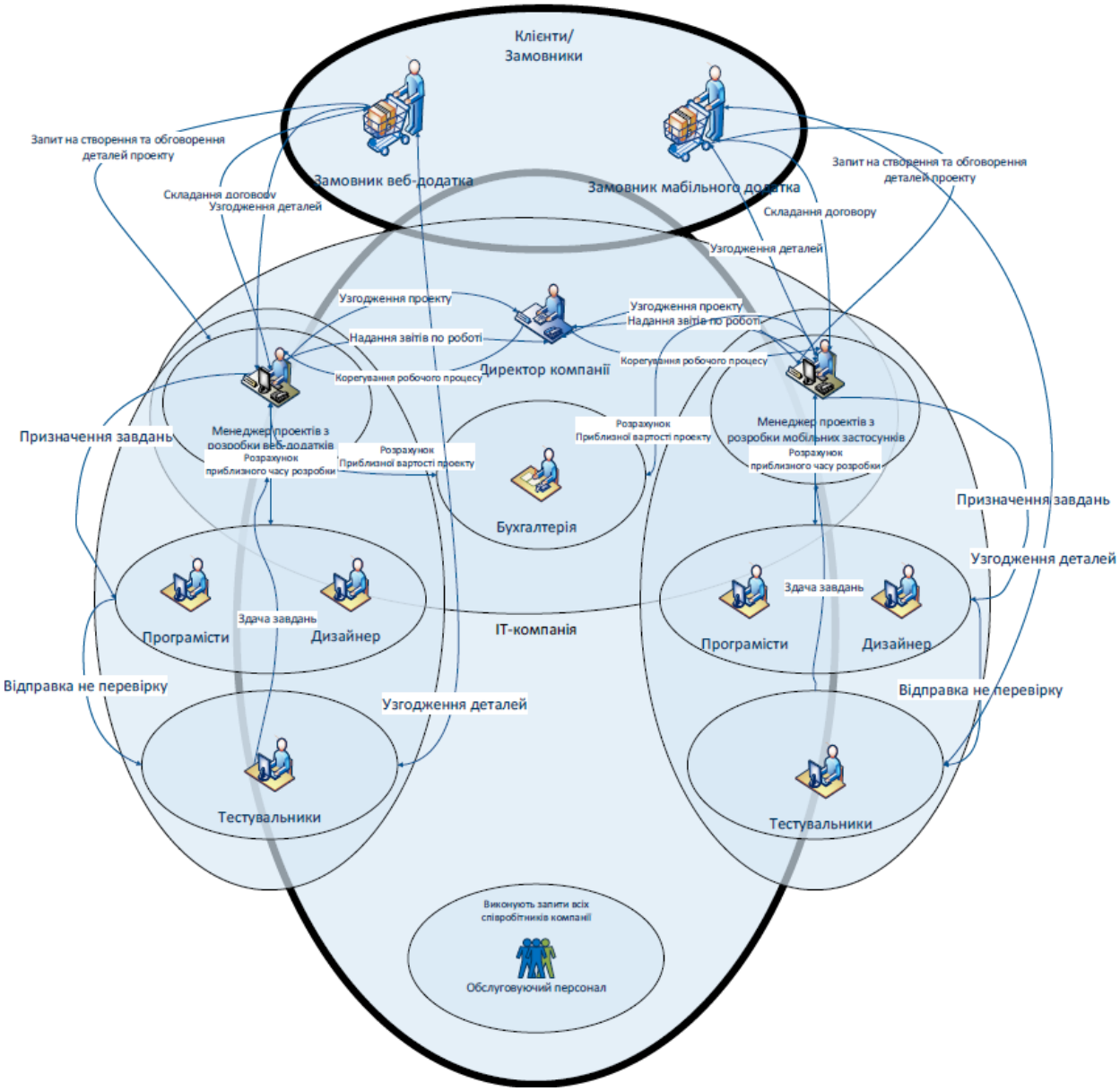


Рисунок 1.7 – Схема моделі кооперації

На основі концепції інформаційної системи з метою розробки проектних рішень автоматизованої інформаційної системи управління проектами ІТ-компанії здійснимо постановку задачі.

Необхідно створити систему управління проектами ІТ-компанії, яка складається з двох частин: серверної та користувацького веб-інтерфейсу. Всі необхідні таблиці повинні створюватися міграціями для того, щоб в подальшому можна буде легко і швидко розширювати функціонал системи та здійснювати розгортання. Для даних, які заповнюються в базу даних при інсталяції повинні бути створені сіди. Ієрархія проекту має складатися з пакетів, кожен з який містить свій функціонал. Вся робота з сайтом повинна відбуватися без перезавантаження сторінок. Але у випадках, коли в цьому є необхідність перезавантаження дозволяється.

Розробка концепції є рекомендованою фазою створення автоматизованої інформаційної системи, що стоїть між фазами "постановка вимог до інформаційної системи" і "розробкою програмних рішень". Концепція покликана досягти певного рівня структурування ідеї автоматизованої інформаційної системи управління проектами та визначити процеси і їх учасників.

Головно метою автоматизованої інформаційної системи управління є підвищення ефективності процесу управління проектами ІТ-компанії. Автоматизована інформаційна система дозволить більш швидко та економічно вигідно створювати нові та удосконалювати існуючі ІТ-продукти.

На основі опису діяльності, організаційної структури компанії та поставлених вимог до інформаційної системи визначимо основні процеси та їх учасників.

Концепція автоматизованої інформаційної системи управління проектами ІТ-компанії представлена на рисунку 1.8.

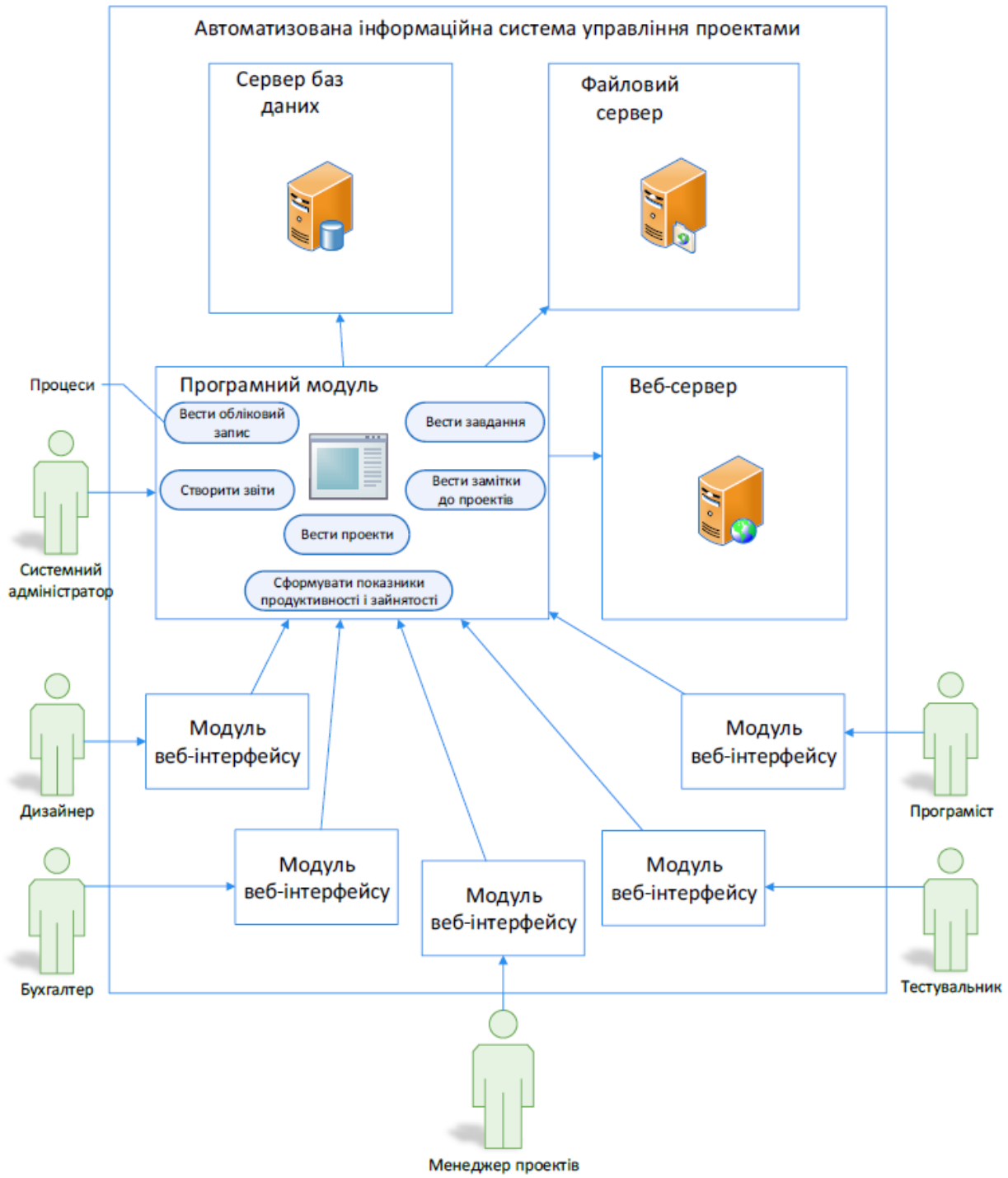


Рисунок 1.8 – Концепція автоматизованої інформаційної системи управління проектами IT-компанії

В результаті було розроблено концепцію інформаційної системи та представлено постановку задачі. На основі концепції та постановки задачі

будуть розроблені програмні рішення автоматизованої інформаційної системи управління проектами ІТ-компанії.

Отже, було надано опис предметної області, який включає в себе різні аспекти управління проєктами. Завдання розробки програмного забезпечення було чітко сформульовано, і воно полягає у створенні інструменту, який допоможе командам ефективніше керувати своїми проєктами, забезпечуючи гнучкість і адаптивність під різні типи проєктів і команд.

2. МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА РОЗРОБКА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ ПРОЕКТНИМИ РЕСУРСАМИ

2.1. Розробка та опис діаграм використання програмної системи

Діаграми варіантів використання (use case diagrams) використовуються в програмній інженерії для візуалізації взаємодії між користувачами (акторами) і системою. Основні цілі та використання таких діаграм включають:

1. Зрозуміння вимог до системи: Діаграми варіантів використання допомагають зрозуміти, які функціональні можливості повинні бути реалізовані в системі. Вони чітко показують, як користувачі взаємодіють із системою, і які конкретні завдання вони виконують.

2. Комунікація з зацікавленими сторонами: Такі діаграми є зручним інструментом для обговорення вимог з користувачами, замовниками та іншими зацікавленими сторонами. Вони дозволяють легко пояснити, як система буде використовуватись у реальних сценаріях.

3. Документування системних вимог: Діаграми варіантів використання слугують офіційною документацією вимог до системи. Вони описують різні сценарії використання системи, які необхідно реалізувати.

6. Планування тестування: Варіанти використання часто стають основою для розробки тестових сценаріїв. Це допомагає забезпечити, що всі вимоги до системи були правильно реалізовані та перевірені.

Розробка веб-додатку для управління проектами передбачає реалізацію кількох ключових функціональних можливостей. Ці функції призначені для автоматизації процесів управління проектами, покращення зв’язку між членами команди, оптимізації розподілу ресурсів і оцінки ефективності запропонованого рішення.

Головною функцією програмного забезпечення є оптимізація ресурсів за допомогою нейронної мережі, більше функцій можна побачити на рисунку 1.4.

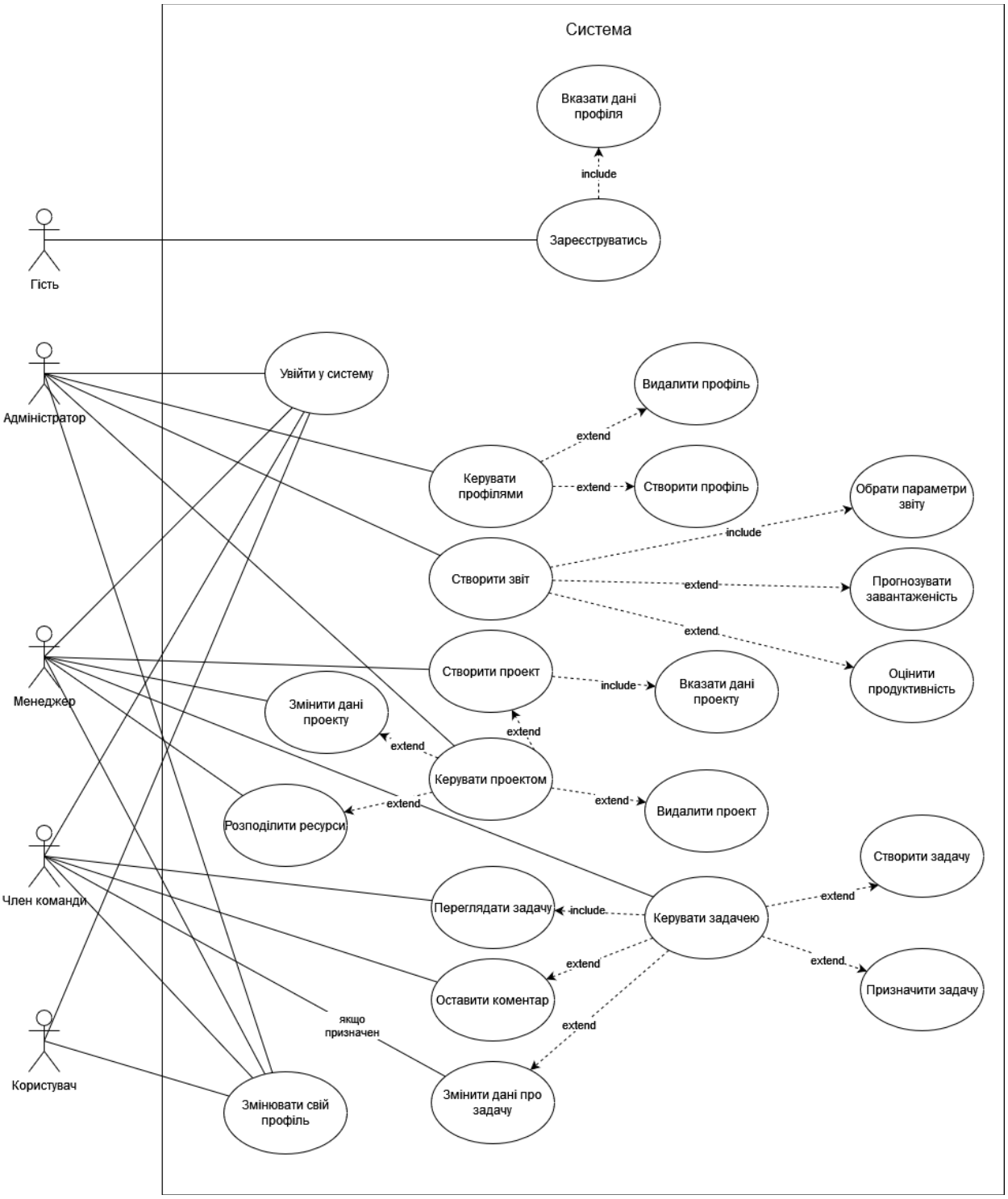


Рисунок 2.1 – Діаграма варіантів використання

Детальний опис варіантів використання, що були наведені на діаграмі, показано в таблицях 21 - 2.8. У них включено ціль, якої треба досягти, опис ситуації, в якої варіант може бути потрібен, опис подій, які відбуватимуться при спрацьовуванні певного варіанта використання, та отриманий результат після завершення варіанту використання.

Таблиця 2.1 – Варіант використання “Створення проекту”

Мета	Створити новий проект та назначити членів команди на нього
Роль	Менеджер проекту
Дії користувача	1. Менеджер проекту натискає на кнопку “Створити проект”. 2. Система пропонує менеджеру проекту ввести деталі проекту (такі як, ім’я, опис, дата початку, дата кінця). 3. Менеджер проекту вводить деталі. 4. Система створює проект і назначає членів команди на нього.
Результат	Проект створено та назначено членів команди

Таблиця 2.2 – Варіант використання “Призначення задачі”

Мета	Призначити задачу членам команди та відстежувати їх прогрес
Роль	Менеджер проекту, члени команди
Дії користувача	1. Менеджер проекту обирає проект. 2. Система показує список членів команди назначених на проект. 3. Менеджер проекту обирає члена команди. 4. Система показує список завдань які мають бути виконані.

	5. Менеджер проекту призначає завдання члену команди. 6. Член команди позначає призначене завдання “у виконанні” коли починає роботу. 7. Член команди позначає призначене завдання “виконано” коли робота виконана.
Результат	Завдання призначене члену команди і прогрес відстежується

Таблиця 2.3 – Варіант використання “Оцінка продуктивності”

Мета	Оцінити ефективність рішень управління проектом
Роль	Менеджер проекту
Дії користувача	1. Менеджер проекту обирає виконаний проект. 2. Система показує різні оцінки продуктивності(час виконання проекту, використання ресурсів). 3. Менеджер проекту оглядає оцінки продуктивності. 4. Менеджер проекту оцінює ефективність рішень управління проектом на основі оцінок продуктивності.
Результат	Менеджер проекту оцінив ефективність рішень управління проектом

Таблиця 2.4 – Варіант використання “Розподілення ресурсів”

Мета	Оптимізація розподілу ресурсів
Роль	Менеджер проекту
Дії користувача	1. Менеджер обирає проект. 2. Система показує список членів команди назначених на проект. 3. Менеджер проекту обирає члена команди. 4. Система показує завантаженість та доступність члена

	команди. 5. Менеджер проекту перерозподіляє завдання та балансує завантаженість.
Результат	Розподілення ресурсів збалансовано

Таблиця 2.5 – Варіант використання “Обговорення оновлень”

Мета	Покращення комунікації між членами команди
Роль	Менеджер проекту, члени команди
Дії користувача	1. Менеджер проекту обирає проект. 2. Система показує список задач. 3. Менеджер проекту обирає задачу. 4. Менеджер проекту залишає коментар відмічаючи членів команди. 5. Члени команди отримують повідомлення.
Результат	Менеджер проекту обговорив оновлення проекту з членами команди

Таблиця 2.6 – Варіант використання “Створення звітів”

Мета	Створення звітів про прогрес виконання проекту та розподілу ресурсів
Роль	Менеджер проекту
Дії користувача	1. Менеджер проекту обирає проект. 2. Система показує різні опції для звіту (звіт з прогресу, звіт з розподілу ресурсів). 3. Менеджер проекту обирає опції звіту. 4. Система створює звіт.
Результат	Створено звіт прогресу та розподілу ресурсів по визначеному проекту

Таблиця 2.7 – Варіант використання “Відстеження прогресу”

Мета	Відстежити прогрес проекту в реальному часі
Роль	Менеджер проекту, члени команди
Дії користувача	1. Менеджер проекту або член команди обирає проект. 2. Система показує панель з прогресом проекту в реальному часі (завдання закінчене, завдання які треба виконати, терміни виконання завдань). 3. Менеджер проекту або член команди оновлює панель коли статус завдання або інші подробиці змінюються.
Результат	Прогрес проекту відслідковано або оновлений у реальному часі

Таблиця 2.8 – Варіант використання “Прогнозування завантаженості”

Мета	Спрогнозувати завантаженість та попит на ресурси на майбутні проекти.
Роль	Менеджер проекту
Дії користувача	1. Менеджер проекту обирає виконаний проект. 2. Система показує різні оцінки завантаженості та попиту на ресурси (кількість виконаних завдань, кількість годин роботи членів команди). 3. Менеджер проекту переглядає оцінки. 4. Система прогнозує завантаженість та попит на ресурси для майбутніх проектів базуючись на оцінках.
Результат	Прогноз завантаженості та попиту ресурсів для майбутніх проектів

Щоб розробити веб-застосунок для управління проектними рішеннями, важливо ідентифікувати та визначити функціональні вимоги, яким має відповідати застосунок. Ці вимоги є специфічними функціями, які система

повинна мати можливість виконувати, щоб задовольнити потреби своїх користувачів. Були визначені функціональні вимоги.

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. Модель вимог наведено в таблиці 2.9.

Таблиця 2.9 – Пріоритетність та ризики вимог

Вимога	**Пріоритет**	**Ризик**
1. Керування користувачами	Високий	Високий
2. Створення та управління проектом	Високий	Низький
3. Керування задачами	Високий	Низький
4. Розподіл ресурсів	Середній	Середній
5. Оновлення та сповіщення в реальному часі	Середній	Середній
6. Безпека та контроль доступу	Високий	Високий
7. Інструменти спілкування	Низький	Низький
8. Параметри налаштування	Низький	Низький
9. Звітність та аналітика	Високий	Високий

Згідно вимоги “Керування користувачами” система повинна дозволяти створення та керування обліковими записами користувачів з різними рівнями доступу (наприклад, адміністратор, менеджер, член команди). З міркувань безпеки кожен обліковий запис користувача повинен мати унікальне ім’я користувача та пароль.

Система повинна дозволяти створювати нові проекти та керувати існуючими проектами. Це включає в себе призначення членів команди для проектів, встановлення кінцевих термінів і етапів проекту, а також відстеження прогресу проекту.

Система повинна дозволяти створювати та керувати завданнями в рамках проекту. Задачі повинні бути призначені конкретним членам команди, а для кожного завдання повинні бути встановлені терміни. Члени

команди повинні мати можливість позначати завдання як виконані та оновлювати їх прогрес.

Щодо вимоги “Звітність та аналітика”, то система повинна забезпечувати функціональні можливості звітності та аналітики, щоб допомогти користувачам контролювати хід проекту, визначати потенційні перешкоди та приймати обґрунтовані рішення. Це включає створення звітів про статус проекту, розподіл ресурсів і показники виконання завдань.

Щодо безпеки та контролю доступу, то система повинна забезпечувати надійні функції безпеки та контролю доступу для забезпечення конфіденційності та цілісності даних проекту. Це включає шифрування конфіденційних даних, багатофакторну автентифікацію та контроль доступу на основі ролей і дозволів користувачів.

Таким чином, функціональні вимоги, визначені вище, надають вичерпний перелік ключових функцій, якими має володіти веб-застосунок для керування задачами проекту. Задовольнивши ці вимоги, система зможе автоматизувати керування проектними рішеннями, покращити зв’язок і співпрацю між членами команди, а також надавати оновлення та сповіщення в режимі реального часу.

2.2. Моделювання основних процесів розроблюваної системи

BPMN (Business Process Model and Notation) модель призначена для візуалізації, моделювання, аналізу та оптимізації бізнес-процесів. Основними елементами BPMN є:

- Події (Events): Вказують на те, що щось відбувається, можуть бути стартові, проміжні та кінцеві.

- Дії (Activities): Описують завдання або роботу, що виконується в рамках процесу.

- Потоки (Flows): Вказують на послідовність виконання дій.

- Гейти (Gateways): Використовуються для визначення точок прийняття рішень.

- Артефакти (Artifacts): Додаткові елементи для уточнення інформації, такі як дані або примітки.

- Пули та дорожки (Pools and Lanes): Використовуються для визначення меж процесів та ролей учасників.

BPMN забезпечує стандартизований спосіб моделювання, що робить його ефективним інструментом для управління та покращення бізнес-процесів.

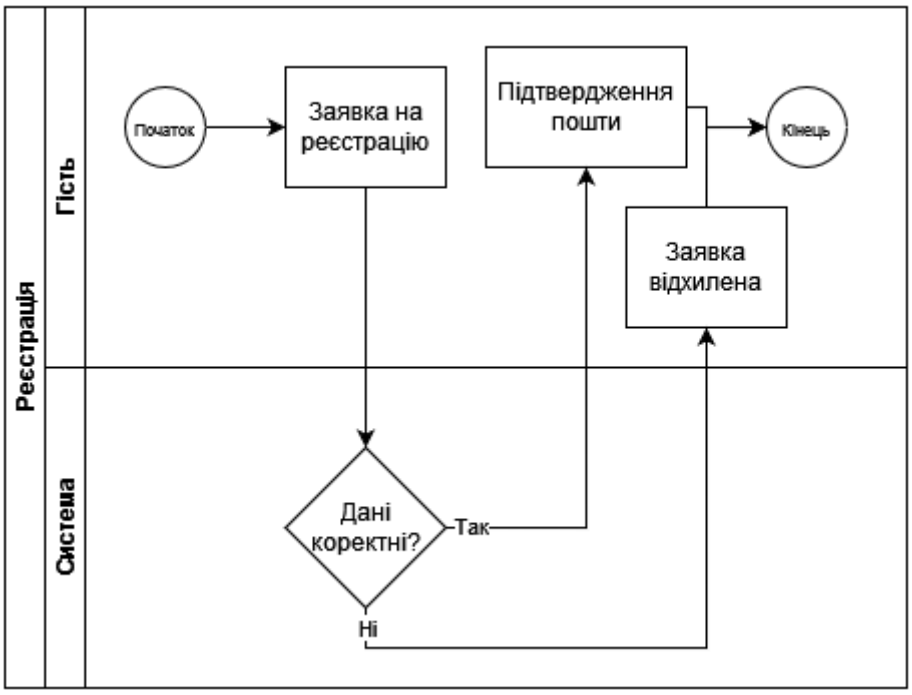


Рисунок 2.2 – Схема бізнес-процесу реєстрації

Опис послідовності створення облікового запису користувача:

− користувач переходить на сторінку реєстрації;

− користувач заповнює поля реєстрації;

− якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою.

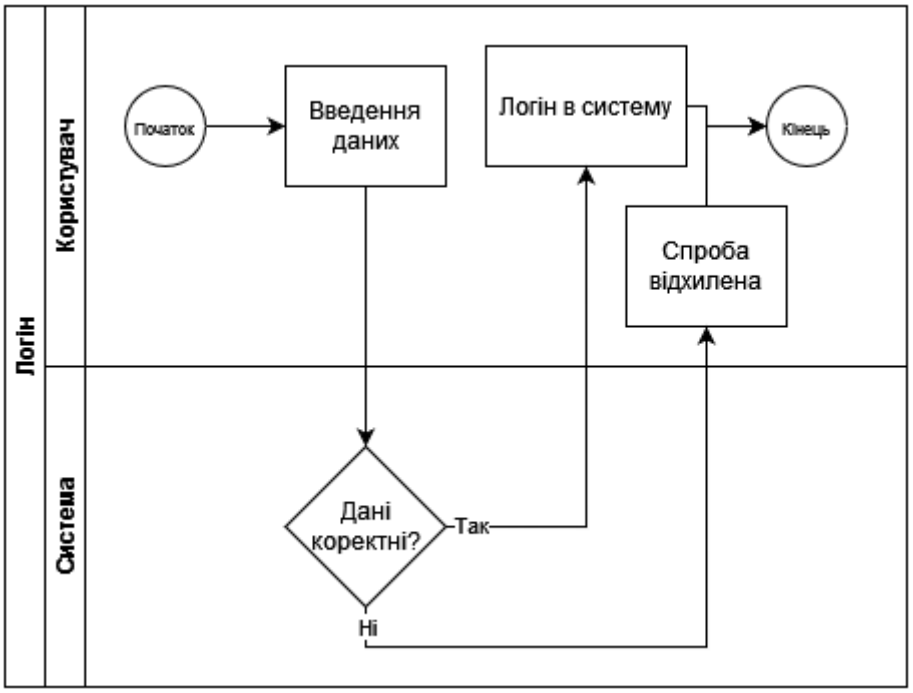


Рисунок 2.3 – Схема бізнес-процесу авторизації

Опис послідовності авторизації користувача:

− користувач переходить на сторінку авторизації;

− користувач заповнює поля логіна;

− якщо введені поля не відповідають жодному обліковому запису, на клієнтській стороні виникає помилка.

На рисунку 2.4 показана схема бізнес-процесу створення проекту в системі.

Опис послідовності створення проекту:

− користувач переходить на сторінку створення проекту;

− користувач заповнює інформацію про проект;

− якщо введені поля не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою;

− в іншому випадку користувач запрошує інших користувачів до проекту;

− інші користувачі можуть підтвердити свою участь чи ні;

− якщо користувач підтверджує участь, то він додається до команди проекту.

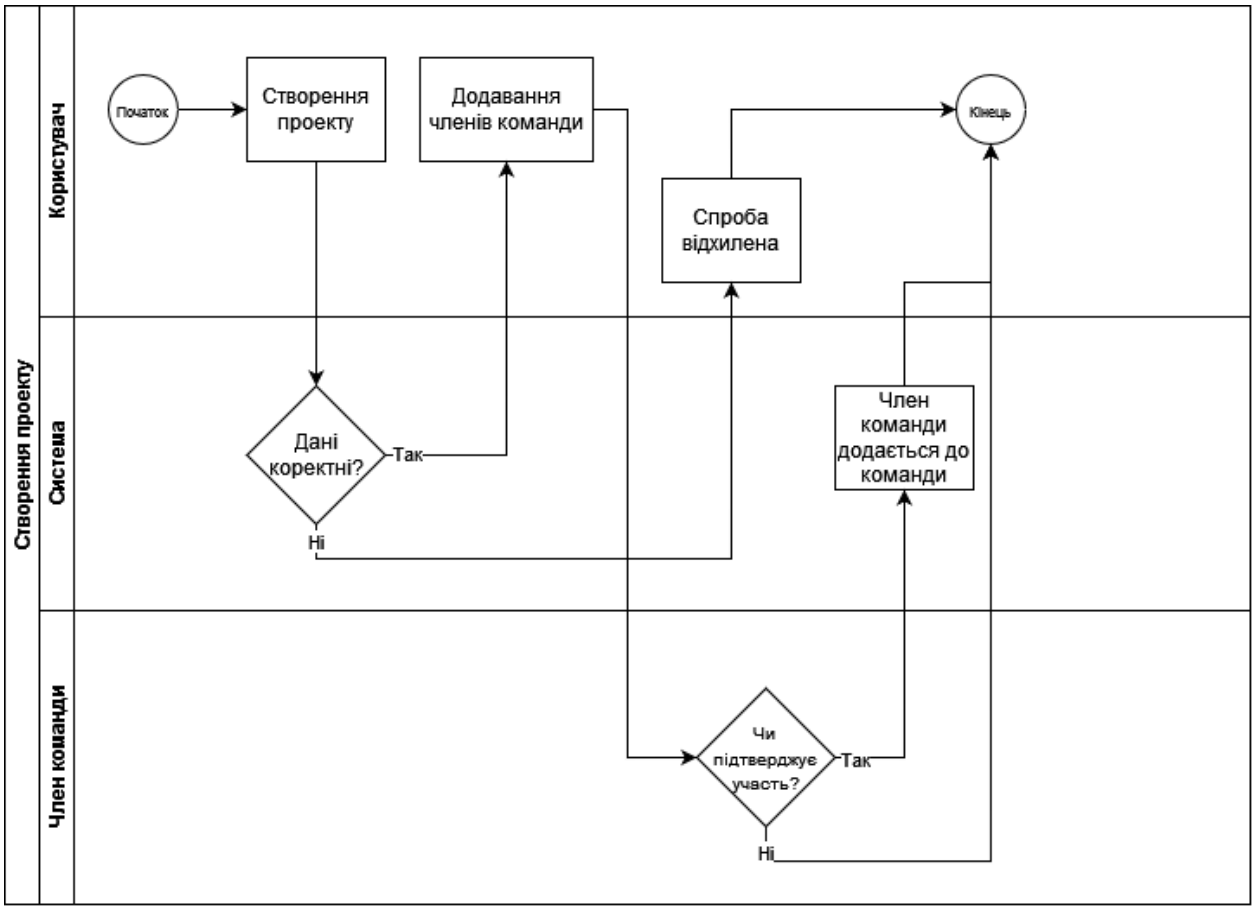


Рисунок 2.4 – Схема бізнес-процесу створення проекту

На рисунку 2.5 представлена схема бізнес-процесу призначення члена команди.

Опис послідовності призначення члена команди до задачі в розроблюваній системі наступний:

− менеджер обирає проект та задачу та призначає інших членів команди до неї;

− система відправляє повідомлення всім призначеним членам команди проекту;

− члени команди отримують повідомлення.

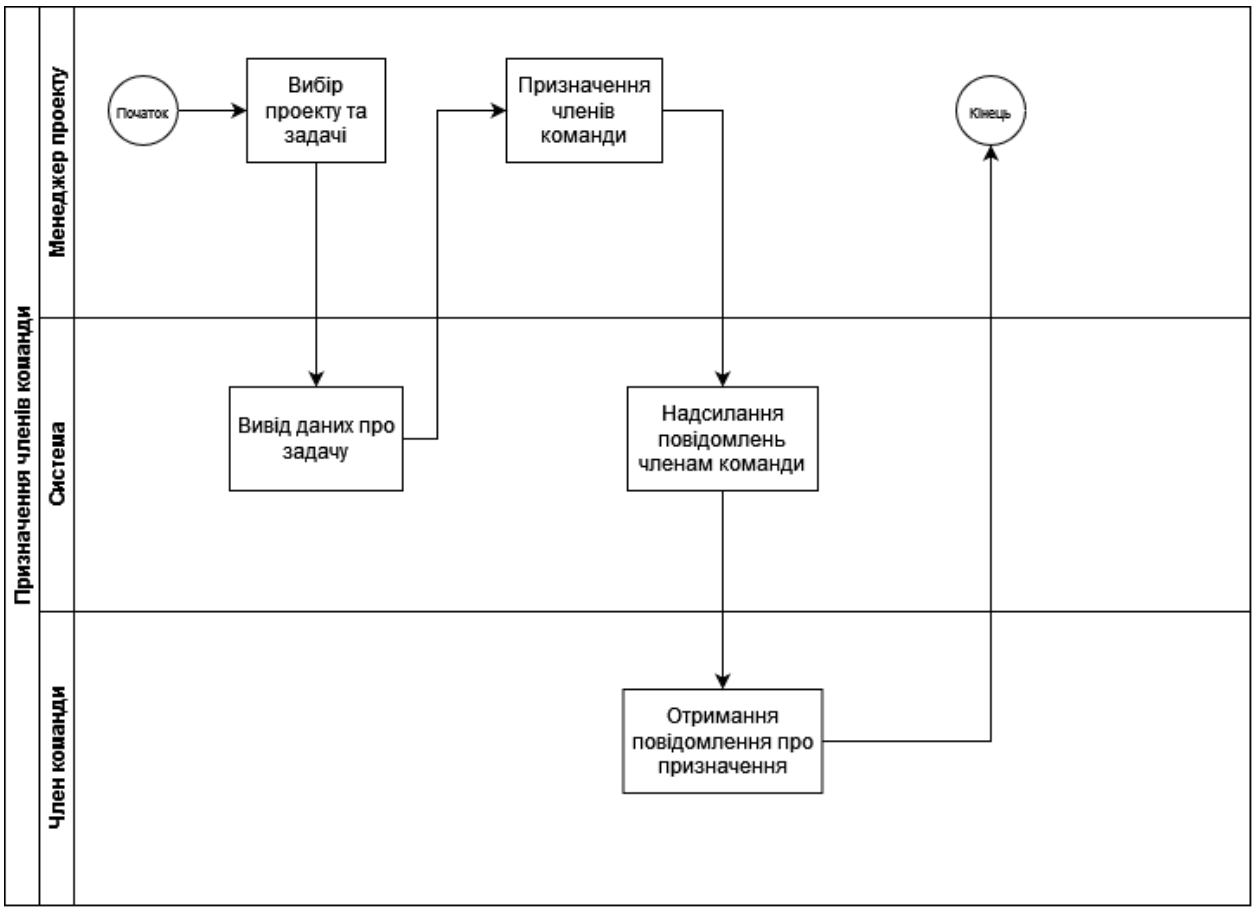


Рисунок 2.5 – Схема бізнес-процесу призначення члена команди

Опис послідовності створення звіту з оцінки продуктивності:

− менеджер обирає проект, для якого потрібно здійснити оцінку;

− якщо проект не завершено, то спроба відхиляється;

− в іншому випадку система виводить значення продуктивності зі сторони різних метрик;

− менеджер обирає лише потрібні йому метрики;

− система здійснює експорт згенерованого файлу;

− якщо у процесі є якісь помилки, то система повідомляє про це менеджера;

− в іншому випадку менеджер отримує файл з оцінкою продуктивності команди.

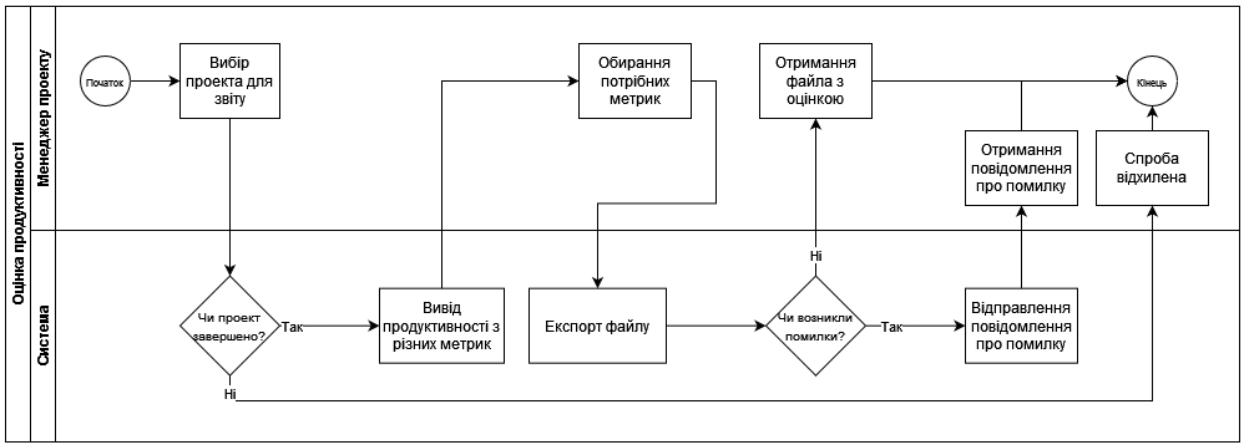


Рисунок 2.6 – Схема бізнес-процесу оцінки продуктивності

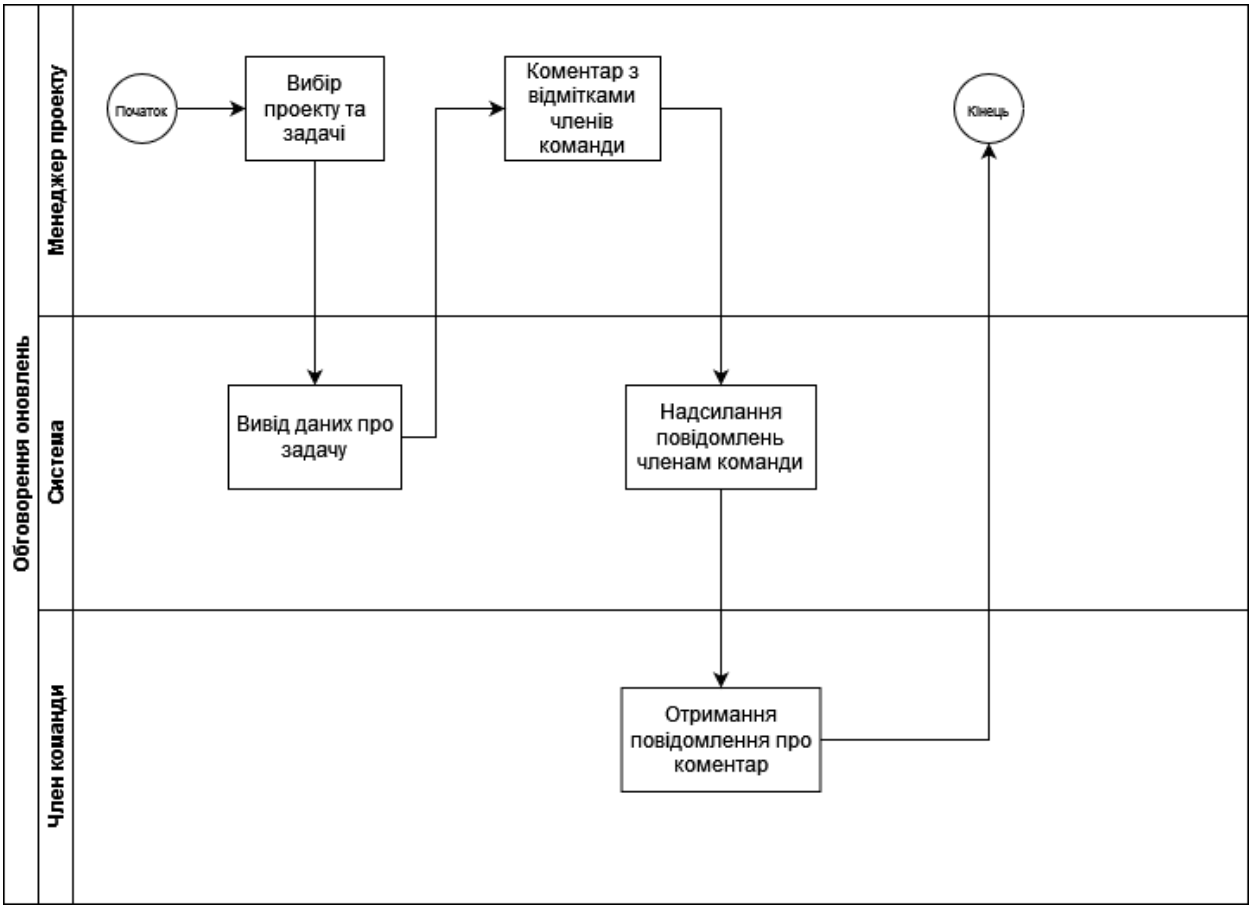


Рисунок 2.7 – Схема бізнес-процесу обговорення оновлень

Опис послідовності обговорення оновлень (рис. 2.7):

− менеджер (або член команди) обирає проект та задачу;

− система переадресує його на сторінку задачі;

− менеджер (або член команди) залишає коментар та відмічає інших членів команди;

− система відправляє усім відміченим членам команди повідомлення;

− члени команди отримують це повідомлення.

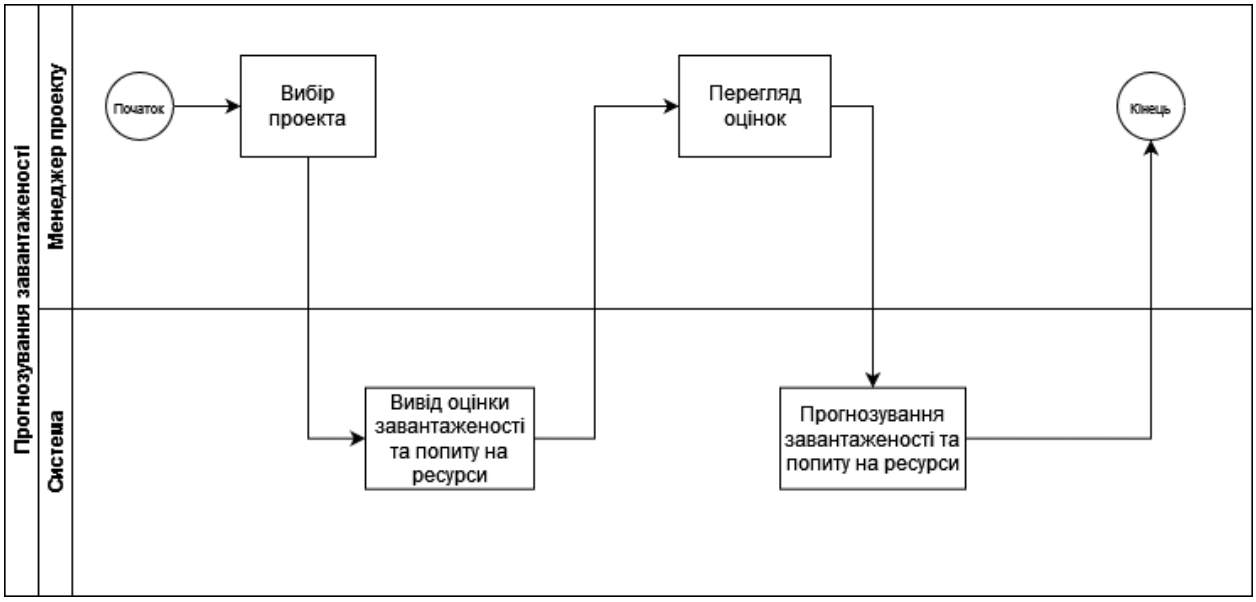


Рисунок 2.8 – Схема бізнес-процесу прогнозування завантаженості

Опис послідовності прогнозування завантаженості (рис. 2.8):

− менеджер обирає проект для оцінки завантаженості;

− система виводить оцінку завантаженості проекту;

− менеджер переглядає оцінку системи та дає запит на прогнозування;

− система прогнозує завантаженість;

Опис послідовності відстеження прогресу (рис. 2.9):

− менеджер (або член команди) обирає проект, за яким хоче слідкувати;

− система виводить подробиці про задачі та їх статуси;

− менеджер (або член команди) обирає задачу на проекті;

− якщо в нього є доступ до редагування задачі, то він може відредагувати будь-які дані про задачу;

− система оновляє вигляд задачі після її зміни;

– якщо доступу нема, то користувач бачить лише основну інформацію про задачу.

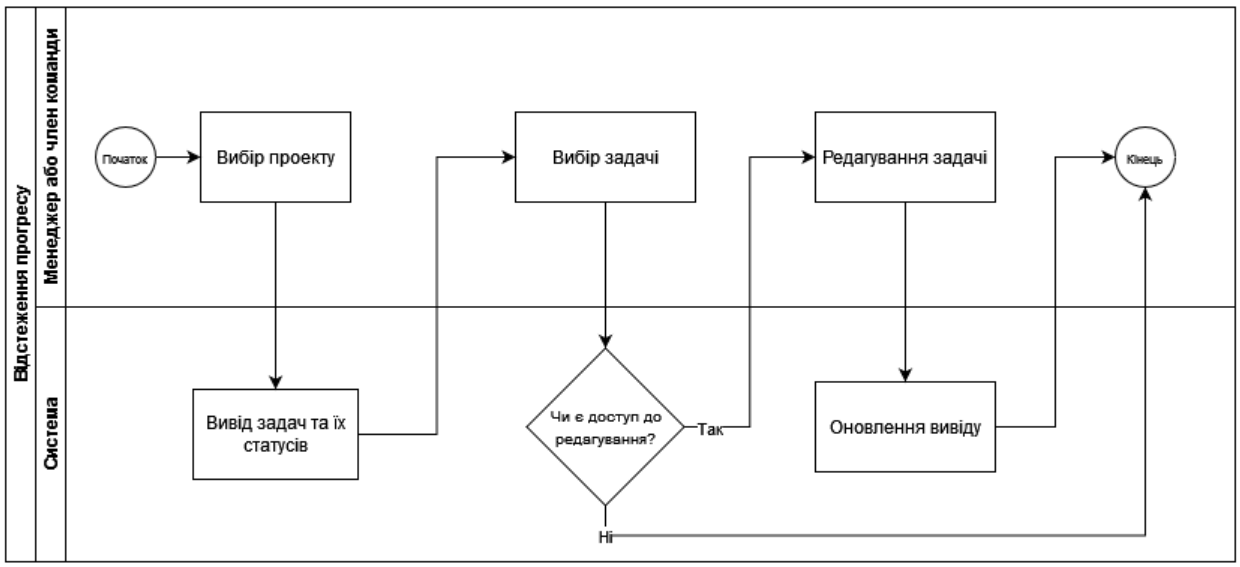


Рисунок 2.9 – Схема бізнес-процесу відстеження прогресу

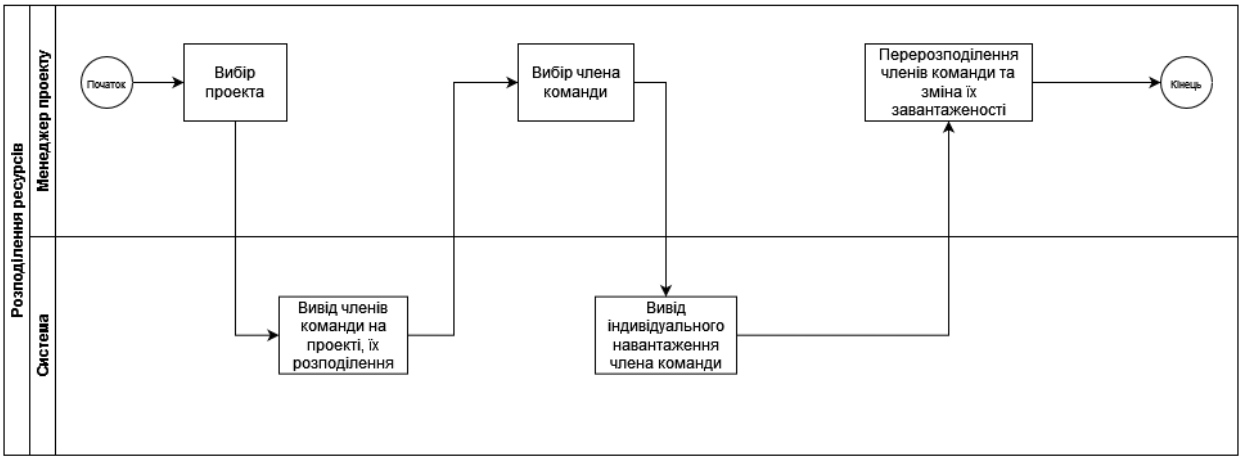


Рисунок 2.10 – Схема бізнес-процесу розподілу ресурсів

Опис послідовності розподілу ресурсів (рис. 2.10):

– менеджер обирає проект, щоб отримати розподілення ресурсів на ньому;

– система виводить дані про розподілення ресурсів;

– менеджер обирає члена команди;

– система виводить його індивідуальне навантаження;

− менеджер має можливість перерозподілити ресурси цього члена команди або іншого.

2.3. Вибір архітектури програмного забезпечення

Проблема вибору архітектури для розроблюваного додатку є часто обговорюваною в середовищі розробників та системних архітекторів. Різні джерела по різному оцінюють ситуацію, приводячи як позитивні, так і негативні сторони використання монолітної та мікросервісної архітектури.

Майбутня архітектура програмного забезпечення, повинна бути такою, щоб процес розробки і супроводження був швидким, простим та саме головне ефективним. Потрібно розуміти, що добре побудована архітектура на сам перед, це одна із запорук успіху. Адже, якщо потрібно буде розширювати, додавати новий функціонал, тестувати, налагоджувати або щось змінювати, то в подальшій розробці, це не складе жодних труднощів.

В процесі розробки нового додатку, потрібно розуміти, який з даних архітектурних шаблонів підходить для конкретної предметної області, бізнес-задач тощо. В кожній з цих архітектур є свої переваги та недоліки, адже поки не існує одного гарного рішення для всіх задач.

Одними із популярних шаблонів проєктування в програмуванні є монолітна та мікросервісна архітектури. Окрім аналізу і формулювання вимог, дані архітектури надають міцний фундамент для розробки програмного забезпечення.

Монолітна архітектура (рис. 2.11) — це традиційна уніфікована модель розробки програмного забезпечення. Монолітний у цьому контексті означає складений з одного шматка. Згідно з Кембриджським словником, прикметник «монолітний» також означає «занадто великий і не піддається зміні» [5]. Він являється деяким додаток, розгортання якого, здійснюється наприклад в

вигляді JAR-файлу або Node з єдиною точкою входу. При побудові додатка на основі моноліта, всі процеси будуть управлятися всередині одного модуля.

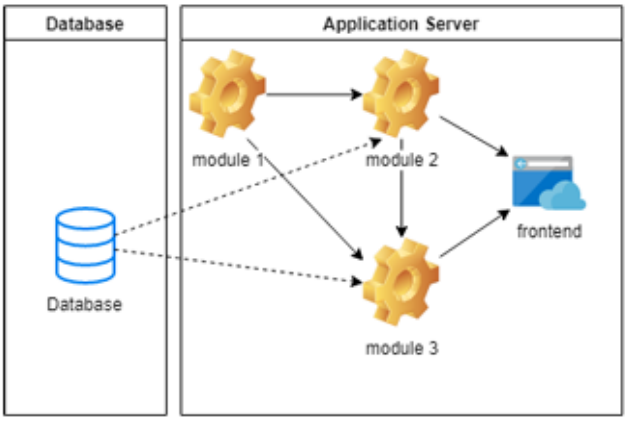


Рисунок 2.11 – Приклад монолітної архітектури

Монолітна архітектура будується на основі клієнт-сервер, в якій функції презентації, обробки програм та управління даними фізично розділені. Рівень презентації відображає різну інформацію, на прикладі застосунку доставки, це може бути різні партнери, послуги, продукти, які доступні на цьому сайті. Рівень обробки програм займається бізнес-логікою, яка в свою чергу призначена для контролю функціональності програми, щоб виконувати детальні обробки. Рівень управління даними призначений для розміщення серверів баз даних, де в свою чергу зберігається інформація. Монолітний підхід є стандартною моделлю побудови та створення програмного забезпечення. Але, його тенденція йде на спад, оскільки створення «моноліту» надає певну кількість проблем, пов’язаних із обробкою величезної кодової бази, впровадженням нових технологій, розгортанням, додаванням нових змін тощо.

Переваги:

1. Легко реалізувати. Даний підхід є стандартним способом створення застосунків, тому, будь-яка команда може мати необхідні знання та можливості для розробки застосунків на основі монолітної архітектури.

2. Простий в розгортанні. В кінці розробки монолітної системи не потрібно працювати з багатьма розгортаннями — лише один файл або каталог.

3. Високий рівень безпеки.

4. Розробникам не потрібно вивчати різні програми, адже можуть зосередитися лише на одній програмі.

5. Проблеми затримки мережі та безпеки відносно менші.

6. Простіше тестування та налагодження. На відміну від мікросервісної архітектури, монолітну — набагато легше налагоджувати та тестувати.

Оскільки вона є єдиною та неподільною і може запускати наскрізне тестування набагато швидше.

7. Конфігурованість.

8. Менша кількість наскрізних проблем. Наскрізні проблеми — це проблеми, які впливають на все програмне забезпечення, наприклад введення журналу, кешування, обробка, моніторинг продуктивності, тощо. Дана область функціональності стосується лише поточного одного програмного застосунку, тому і легше працювати з цією системою.

Мікросервісна архітектура — це архітектурний шаблон, в якому застосунок будується за рахунок невеликих сервісів, кожен з яких працює у своєму власному процесі та спілкується з іншими, використовуючи прості та швидкі протоколи передачі даних HTTP (рис. 2.12). Ці послуги будуються з урахуванням бізнес-можливостей і можуть бути незалежно розгорнуті за допомогою повністю автоматизованого механізму розгортання. Існує мінімум централізованого управління цими службами, які можуть бути написані різними мовами програмування та використовувати різні технології зберігання даних [6]. Ключові концепції, які потрібно використовувати при розробці мікросервісної архітектури — це декомпозиція та розгрупування.

Функціональність додатків має бути повністю незалежною одна від одної.

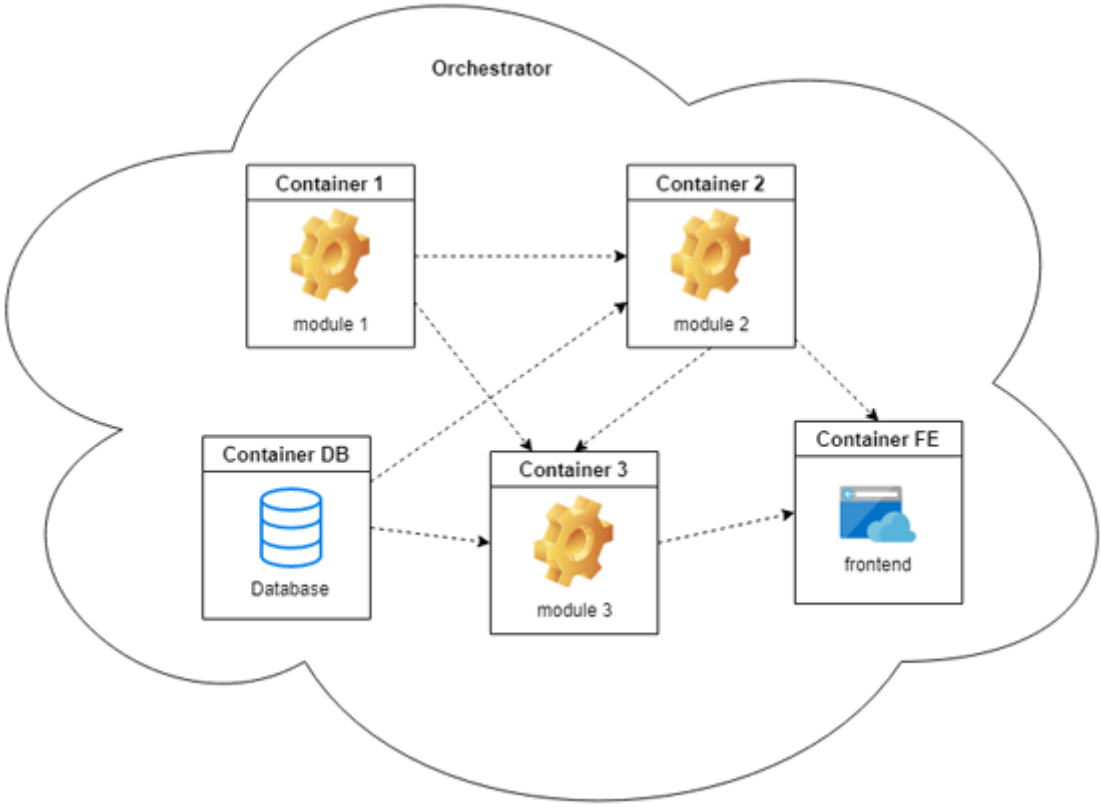


Рисунок 2.12 – Приклад мікросервісної архітектури

Мікросервісна архітектура має суттєвий вплив на зв'язок між застосунком і базою даних. Замість використання спільної бази даних з іншими мікросервісами, кожен з них має свою власну. Часто це може призвести до дублювання даних. Але будування бази даних на кожний мікросервіс є важливим аспектом, якщо потрібно мати вигоду від даної архітектури, та, щоб вона забезпечувала слабкий зв'язок. Ще одна перевага наявності окремої бази даних, полягає в тому, що кожен сервіс може використовувати тип бази даних, який найкраще підходить для його потреб.

Кожен сервіс пропонує безпечну границю якогось модуля, щоб інші мікросервіси, могли бути написані на різних мовах програмування.

Архітектура мікросервісу містить багато шаблонів, наприклад виявлення та реєстрація служб, кешування, API-Gateway, безпека тощо.

Архітектура програмного забезпечення на основі подій (Event-Driven Architecture, EDA) — це підхід до розробки систем, у яких взаємодія між

компонентами здійснюється через передачу та обробку подій. В основі цієї архітектури лежить принцип реагування на події, що дозволяє створювати гнучкі та масштабовані системи.

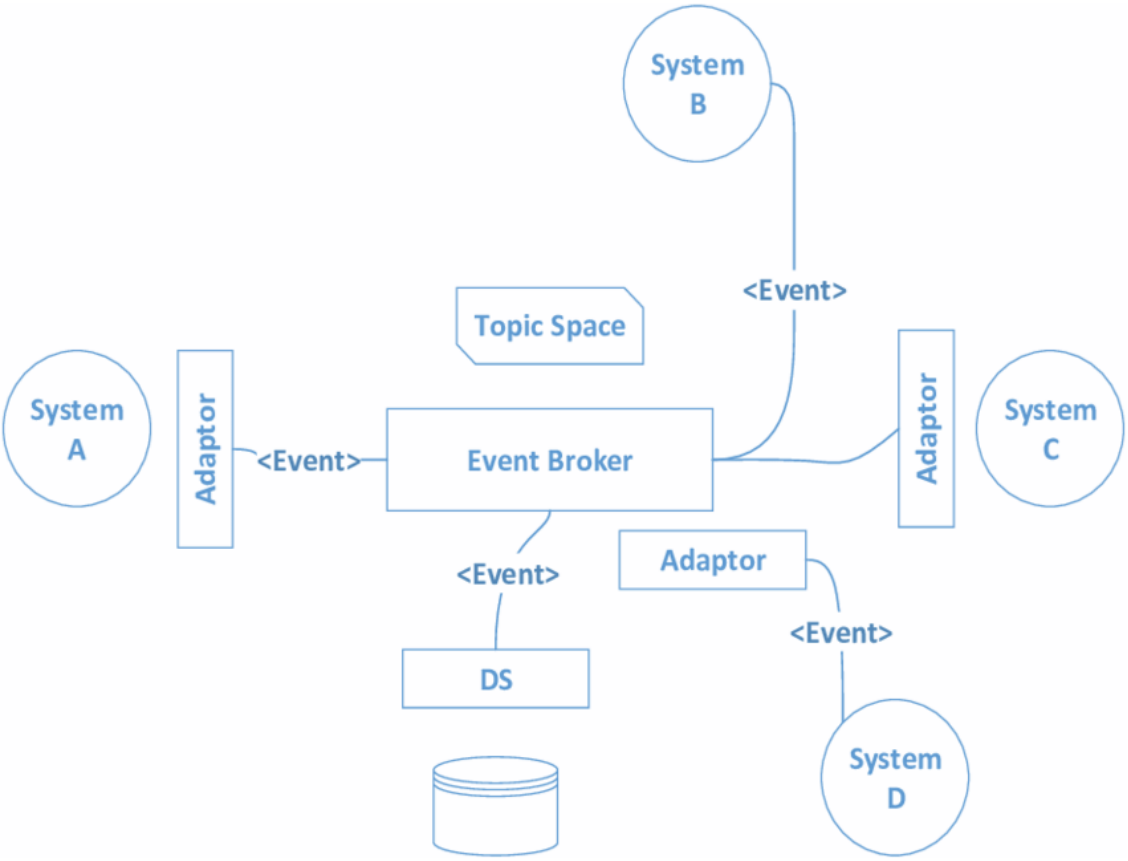


Рисунок 2.13 – Приклад Event-Driven Architecture (EDA)

Основні компоненти EDA:

1. Відправники подій (Event Emitters/Producers):

- Компоненти, які створюють і відправляють події.

- Відправниками можуть бути різні частини системи, такі як користувацькі інтерфейси, датчики або інші програми.

2. Споживачі подій (Event Consumers):

- Компоненти, які підписуються на події і реагують на них.

- Споживачі можуть виконувати різні дії, такі як обробка даних, оновлення інтерфейсу або ініціація інших процесів.

3. Шини подій (Event Bus):

- Механізм, який забезпечує доставку подій від відправників до споживачів.

- Може бути реалізований за допомогою різних технологій, таких як черги повідомлень, брокери повідомлень або системи публікації-підписки (publish-subscribe).

4. Події (Events):

- Повідомлення, які містять інформацію про зміни стану або певні дії в системі.

- Можуть містити дані про те, що сталося, коли сталося і яка інформація пов'язана з подією.

Переваги EDA

1. Масштабованість:

- Легка інтеграція нових компонентів без впливу на існуючу систему.

- Система може масштабуватись як горизонтально (додавання нових вузлів), так і вертикально (збільшення ресурсів на вузлах).

2. Гнучкість:

- Компоненти можуть працювати незалежно один від одного, що спрощує їх розробку та тестування.

- Легка зміна бізнес-логіки шляхом додавання або зміни обробників подій.

3. Відмовостійкість:

- Децентралізована структура дозволяє системі продовжувати роботу навіть при збої окремих компонентів.

4. Асинхронність:

- Події можуть оброблятись асинхронно, що покращує продуктивність і знижує затримки в системі.

EDA дозволяє створювати динамічні та реактивні системи, які можуть ефективно обробляти події в реальному часі, забезпечуючи високу продуктивність і гнучкість.

Враховуючи ці особливості проаналізованих архітектур, було обрано моноліт, оскільки він найбільш зручний у використанні при розробці невеликих веб-застосунків.

Система є незалежною системою, побудованою на монолітній архітектурі з використанням тришарового архітектурного шаблону (presentation, business logic та data access шари) (рис. 2.14).

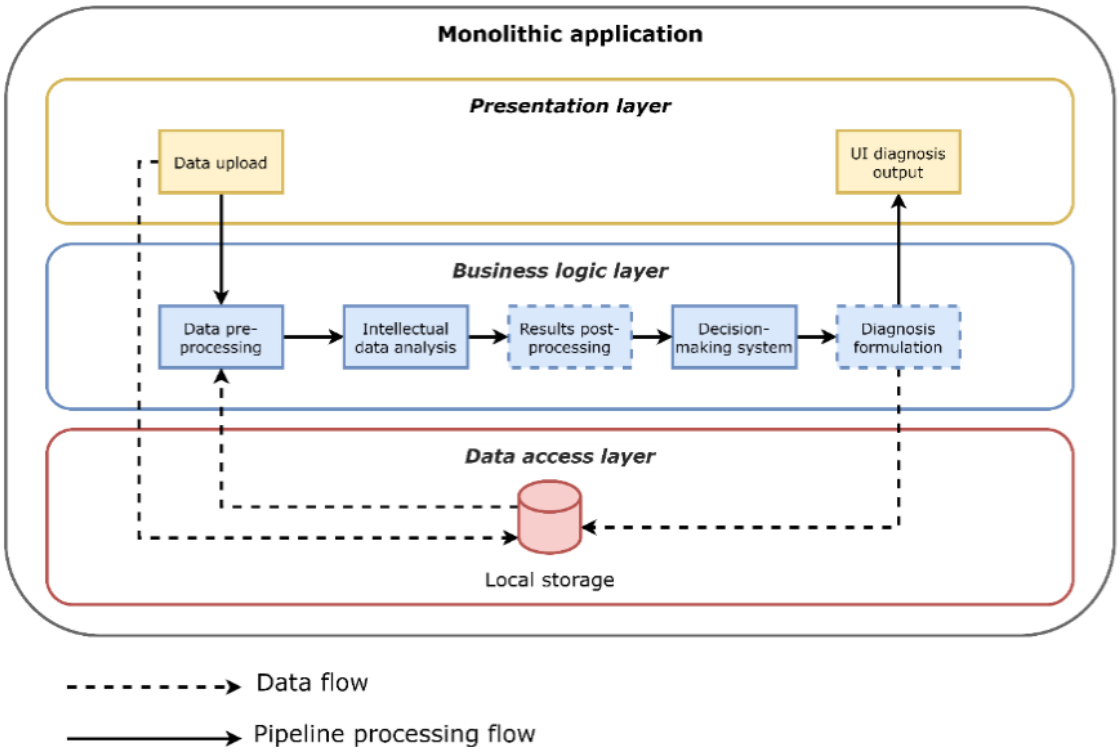


Рисунок 2.14 – Монолітна архітектура системи управління проектними ресурсами

Такий підхід має такі переваги:

1) простота розгортання програми на комп'ютерах;

2) мінімальні накладні витрати на передачу даних;

3) повна незалежність від зовнішніх серверів, мережевих компонентів, все відбувається тільки на одному комп'ютері;

4) можливість підстроювання (кастомізації) системи під необхідні потреби;

5) найменша вартість;

6) швидша розробка;

7) легкість і простота впровадження.

Проте з класичним підходом обробка буде синхронною і користувач повинен чекати, доки вона не закінчиться, тому ефективним рішенням для переведення обчислень в асинхронний режим (щоб можна було продовжити роботу з системою під час обчислень) буде використання message-driven архітектури на рівні бізнес логіки (рис. 2.15).

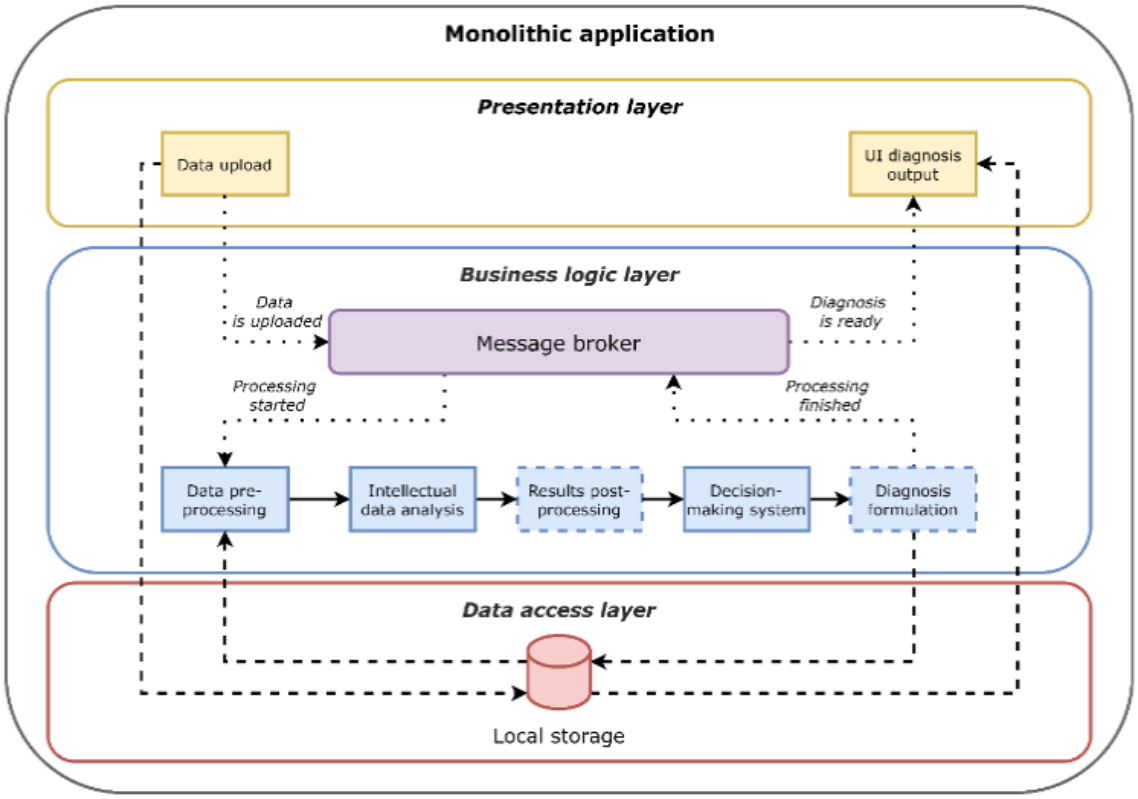


Рисунок 2.15 – Монолітна архітектура системи с message-driven бізнес логікою

2.4. Розробка ER діаграми сутностей

Система розробляється, втілюючи ООП підхід. На рисунку 2.16 можна побачити основні сутності, що будуть задіяні в проекті. За ними буде проектуватися база даних, а також, моделі в коді.

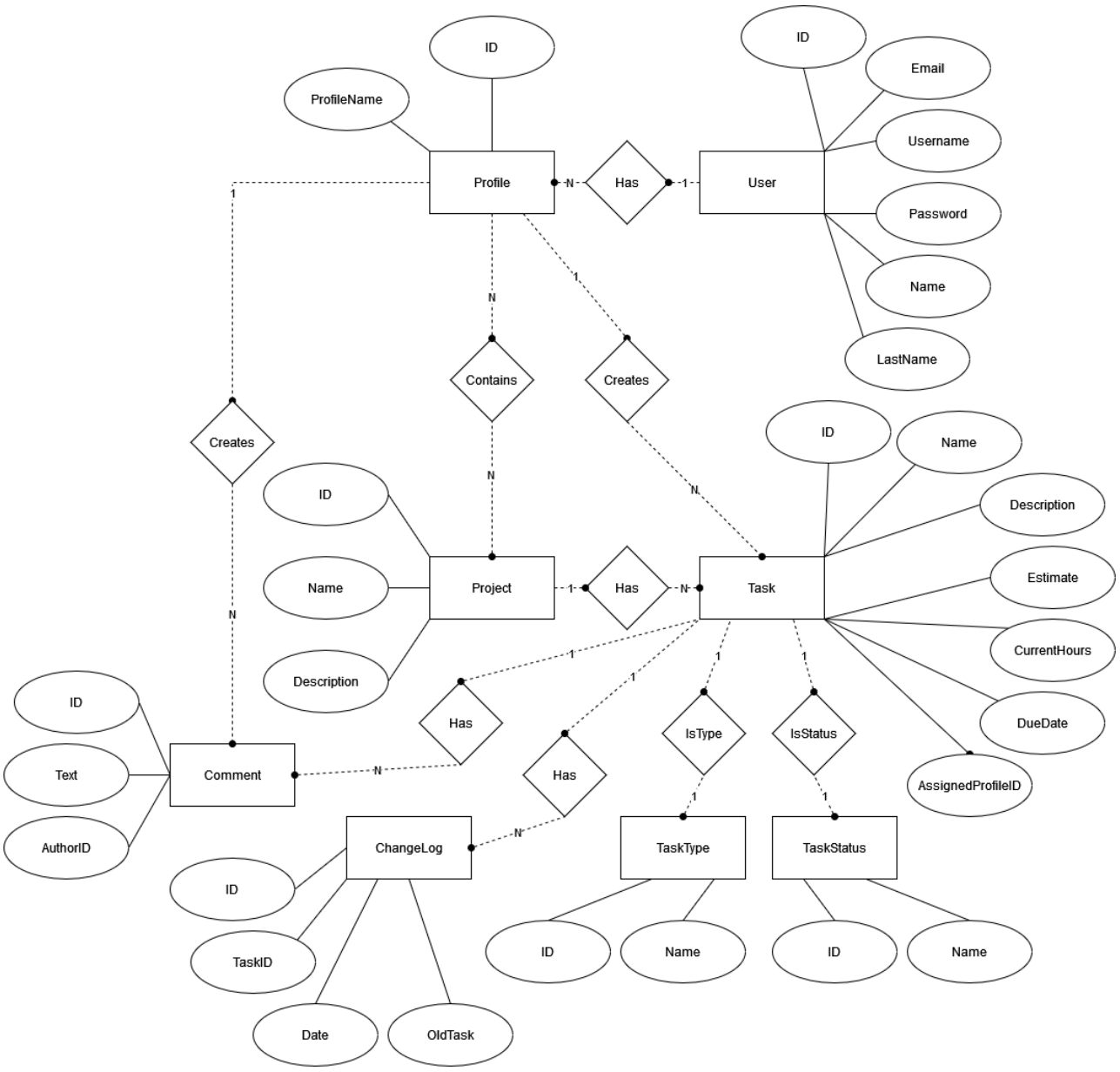


Рисунок 2.16 – ER діаграма сутностей

На рисунку 2.16 показано діаграму сутностей User, Profile, Task, Project, Comment, ChangeLog, TaskType, TaskStatus.

У будь-якого користувача може бути багато профілів, тому що один й той ж користувач може мати декілька ролей в різних командах та проектах. Профіль має роль, яка може бути адміністратором, менеджером, членом команди або звичайним користувачем. Адміністратор має повний контроль над системи, у той час як менеджер має контроль лише над створеними їм проектами.

Проекти мають задачі та члени команди можуть редагувати лише ті задачі, на які вони назначені. Звичайні користувачі можуть лише переглядати проекти та задачі, до яких в них є доступ. Вони не можуть редагувати їх чи бути назначені до них.

Менеджери можуть генерувати звіти та бачити оцінку проектів з різних критеріїв. Вони також можуть перерозподіляти ресурси та членів команди між різними задачами та проектами.

Система втілює механізми для отримання повідомлень та оновлень у режимі реального часу. Таким чином, будь-який користувач буде мати саму нову інформацію.

Задачі можуть змінюватися за статусом, а, також, мають тип. Це може бути задача, група задач, баг тощо.

Щоб зберегти інформацію для отримання оцінки та прогнозів, в усіх задачах ведеться лог змін. Тобто, будь-яка зміна в задачі буде зберігатися в системі. Такий підхід до проектування програмного забезпечення легше всього реалізувати, використовуючи багатошарову архітектуру, яка включає в себе шар представлення, шар бізнес-логіки, шар доступу к БД та інші можливі шари. Зокрема, шар представлення буде здійснювати комунікацію з бізнес-логікою за допомогою сервісів, які будуть мати доступ до інформації завдяки репозиторіям, що відсилають запити до самої БД.

Це популярний підхід до проектування програмного забезпечення, що довів свою ефективність та по цей день широко використовується в різних проектних рішеннях.

2.5. Розробка моделі бази даних та її опис

В якості системи управління базами даних використовується Microsoft SQL Server. База даних серверу призначена для зберігання користувачів, проектів, задач, а також інших даних про систему.

Microsoft SQL Server — це реляційна система управління базами даних (RDBMS), розроблена компанією Microsoft. Вона використовується для зберігання та керування даними, підтримує різноманітні типи даних і забезпечує високий рівень продуктивності, надійності та безпеки. Основні аспекти Microsoft SQL Server включають:

1. Архітектура

- Інстанси: SQL Server може працювати з декількома інстансами на одному сервері, дозволяючи використовувати одну фізичну машину для обслуговування кількох баз даних.

- Сервери і бази даних: Основними компонентами є сервер (інстанс SQL Server) і база даних, яка містить дані та структури для їх зберігання і керування.

2. Мови запитів

- Transact-SQL (T-SQL): Розширення SQL, яке включає програмні конструкції, такі як змінні, умовні оператори, цикли, а також можливості для керування транзакціями та викликів процедур.

3. Безпека

- Аутентифікація: Підтримка Windows-аутентифікації та SQL Server-аутентифікації.

- Шифрування: Шифрування даних на рівні колонок та Transparent Data Encryption (TDE) для шифрування всіх даних у базі.

4. Продуктивність

- Індекси: Підтримка кластеризованих та некластеризованих індексів для підвищення швидкості запитів.

- Кешування: Використання буферного пулу для зберігання часто використовуваних даних у пам'яті, що зменшує кількість звернень до диску.

5. Висока доступність та відновлення після збоїв

- Always On Availability Groups: Забезпечує високу доступність баз даних та можливість автоматичного переключення на резервні сервери.

- Журнали транзакцій: Використання журналів транзакцій для забезпечення відновлення даних після збоїв.

6. Реплікація

- Видана-Абонентна (Publisher-Subscriber): Механізм для розподілення даних між різними серверами та базами даних.

- Міррорінг баз даних: Створення копій баз даних на різних серверах для підвищення надійності.

7. Бізнес-аналітика

- SQL Server Analysis Services (SSAS): Інструмент для аналітичної обробки даних та створення багатовимірних моделей.

- SQL Server Reporting Services (SSRS): Платформа для створення та розповсюдження звітів.

8. Інтеграція даних

- SQL Server Integration Services (SSIS): Інструмент для екстракції, трансформації та завантаження даних (ETL).

Використання та застосування

- Корпоративні додатки: Використовується для керування великими обсягами даних в бізнес-додатках.

- Аналітика і звітність: Використовується для створення звітів та аналітичних систем, що дозволяють бізнесу приймати обґрунтовані рішення на основі даних.

- Веб-додатки: Зберігання і управління даними для різноманітних веб-додатків.

- Інтернет-магазини: Обробка транзакцій та управління каталогами товарів.

Випуски та ліцензування

Microsoft SQL Server пропонується в різних версіях, кожна з яких підходить для певних потреб:

- SQL Server Enterprise: Підходить для великих підприємств з високими вимогами до продуктивності, доступності та безпеки.

- SQL Server Standard: Призначений для середніх підприємств і забезпечує базові можливості управління даними та бізнес-аналітики.

- SQL Server Express: Безкоштовна версія з обмеженими можливостями, підходить для невеликих додатків та навчання.

Інтеграція та підтримка

SQL Server легко інтегрується з іншими продуктами Microsoft, такими як Azure, Power BI, Visual Studio та Active Directory. Це робить його потужним рішенням для розробки комплексних бізнес-додатків та аналітичних систем.

Microsoft SQL Server є однією з найпопулярніших систем управління базами даних завдяки своїй надійності, масштабованості та широким можливостям для управління даними і є однією з найбільш використовуваних СКБД.

Опис таблиць бази даних наведено у таблицях 2.10 - 2.15. Окрім цих таблиць також присутня таблиця користувачів, що створюється фреймворком Blazor за замовченням.

Таблиці мають перелік усіх полів в таблиці, їх тип даних та опис, який дає більше інформації про те, навіщо поле потрібно в цій таблиці та як воно пов’язано з іншими таблицями. Модель бази даних повністю співпадає з сутностями моделей, що наведені на рисунку 2.17.

Таблиця 2.10 – Опис таблиці Project

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер проекту
name	varchar	ім'я проекту

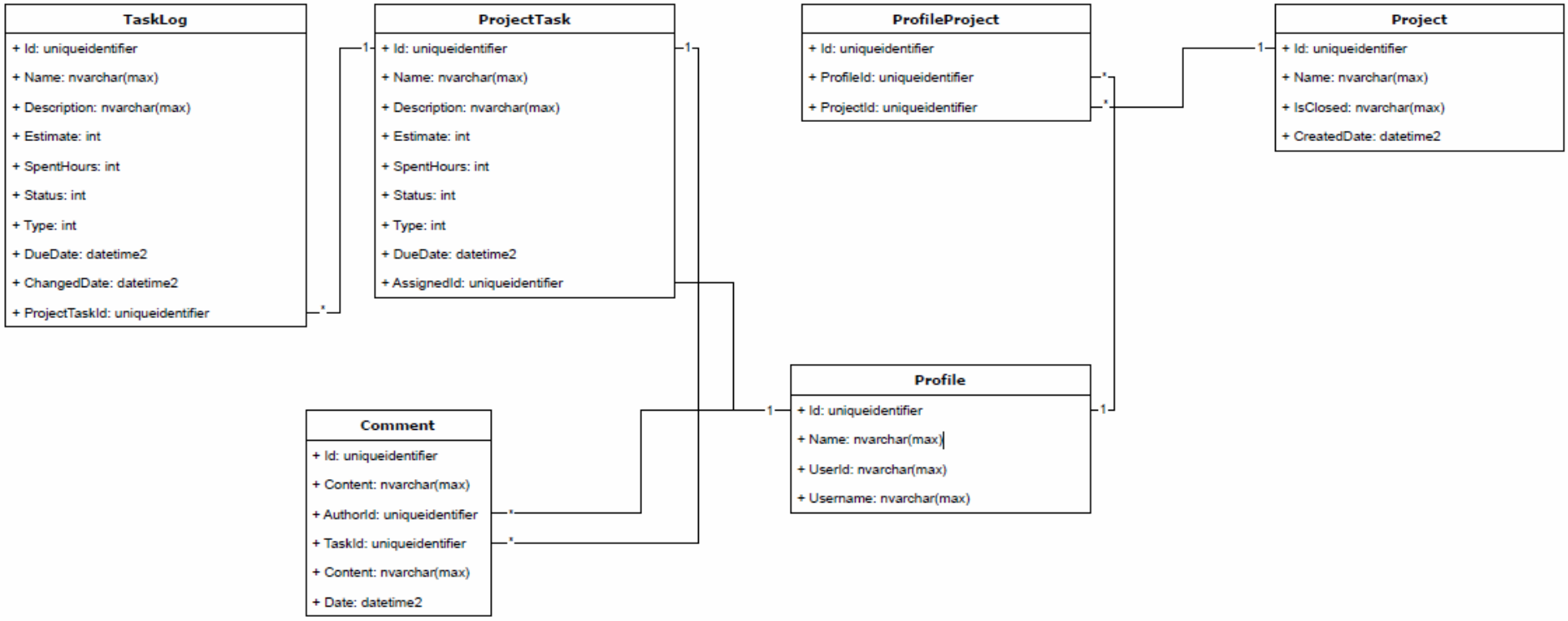


Рисунок 2.17 – Модель бази даних

Таблиця 2.11 – Опис таблиці Profile

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер профіля
name	varchar	ім'я профіля

Таблиця 2.12 – Опис таблиці ProjectTask

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер задачі
description	varchar	опис задачі
name	varchar	ім'я задачі
assigned_id	Guid	ідентифікаційний номер профіля, що назначений на задачу
due_date	DateTime	дата завершення задачі
type_id	Guid	ідентифікаційний номер типу задачі
status_id	Guid	ідентифікаційний номер статусу задачі
spent_hours	int	кількість годин, витрачених на задачу
Estimate	int	кількість годин, необхідних для виконання задачі

Таблиця 2.13 – Опис таблиці Comment

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер коментаря
task_id	Guid	ідентифікаційний номер задачі
author_id	Guid	ідентифікаційний номер профіля, що створив коментар
text	varchar	вміст коментарю

Таблиця 2.14 – Опис таблиці Profile_Project

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер запису в таблиці
profile_id	Guid	ідентифікаційний номер профіля
project_id	Guid	ідентифікаційний номер проекту

Таблиця 2.15 – Опис таблиці Profile_Project

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер статусу
name	varchar	ім'я статусу

Таблиця 2.15 – Опис таблиці Type

Назва поля	Тип даних	Опис
id	Guid	ідентифікаційний номер типу
name	varchar	ім'я типу

2.6. Вибір інструментів для розробки

Для розробки програмного забезпечення серверної частини дипломної роботи використовується Visual Studio.

Visual Studio — це інтегроване середовище розробки (IDE), розроблене компанією Microsoft, яке використовується для створення різноманітних типів додатків, включаючи веб-додатки, настільні додатки, мобільні додатки та хмарні сервіси. Visual Studio підтримує багато мов програмування та надає широкий спектр інструментів для розробки, тестування та розгортання програмного забезпечення.

Основні характеристики Visual Studio:

1. Підтримка різних мов програмування:

- C#, VB.NET, C++, F#, Python, JavaScript, TypeScript, і багато інших.

- Можливість додавання підтримки інших мов через розширення.

2. Редактор коду:

- Підсвітка синтаксису та автодоповнення коду (IntelliSense).

- Підтримка рефакторингу коду.

- Навігація по коду, включаючи перехід до визначення, перегляд посилань і використання функцій.

3. Інструменти для відлагодження:

- Вбудований відлагоджувач з підтримкою відладження на рівні коду, інтерфейсу користувача та бази даних.

- Можливість відладження як локально, так і віддалено.

4. Інструменти для тестування:

- Підтримка створення та виконання модульних тестів, інтеграційних тестів і тестів користувацького інтерфейсу.

- Інструменти для аналізу покриття коду тестами.

5. Інтеграція з системами контролю версій:

- Підтримка Git, Azure Repos, SVN та інших систем.

- Інструменти для управління гілками, злиттям і вирішенням конфліктів.

6. Розширення та налаштування:

- Можливість встановлення розширень для додавання нових функцій, мов програмування та інструментів.

- Магазин Visual Studio Marketplace, де можна знайти численні розширення.

7. Розробка для різних платформ:

- Підтримка створення додатків для Windows, macOS, Linux, Android, iOS і веб-додатків.

- Інтеграція з Xamarin для розробки мобільних додатків.

- Інтеграція з Unity для розробки ігор.

8. Хмарні сервіси та DevOps:

- Інтеграція з Azure для розробки, тестування та розгортання хмарних додатків.

- Підтримка інструментів DevOps для автоматизації процесів CI/CD (continuous integration/continuous deployment).

9. Інструменти для баз даних:

- Інтеграція з Microsoft SQL Server, Azure SQL, та іншими базами даних.

- Інструменти для створення, редагування та відладження баз даних.

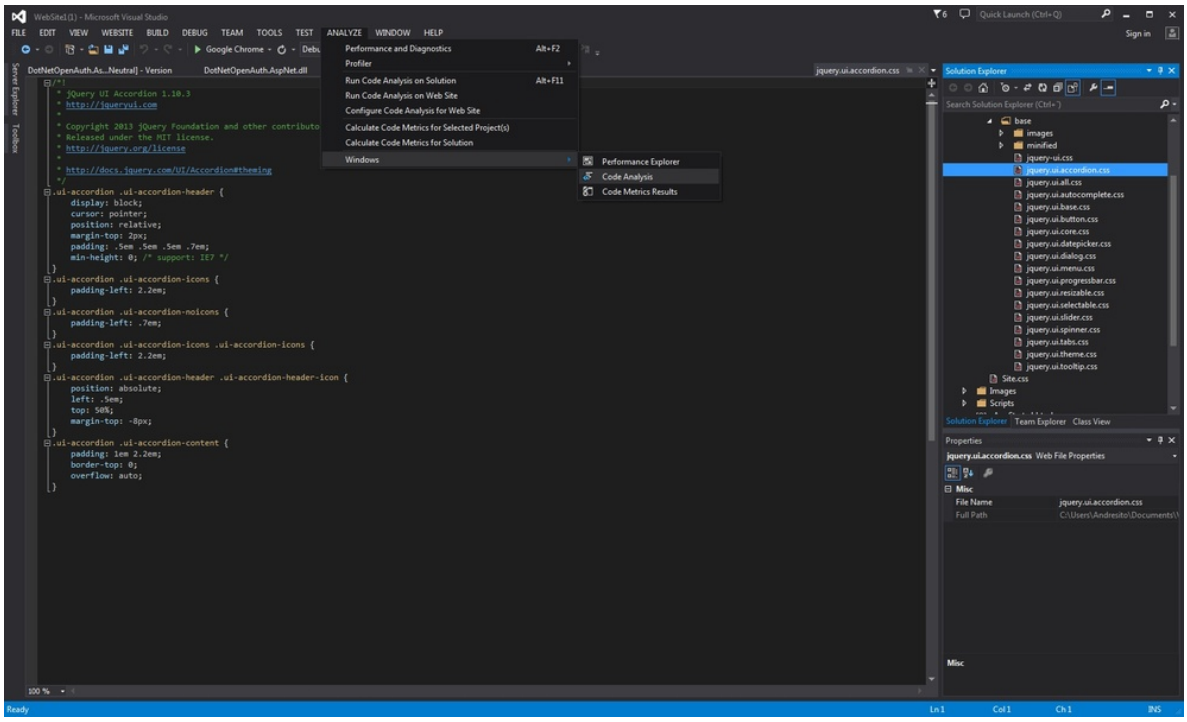


Рисунок 2.18 – Інтерфейс Visual Studio

Випуски Visual Studio:

1. Visual Studio Community:

- Безкоштовна версія для індивідуальних розробників, студентів і невеликих команд з відкритим кодом.

- Містить більшість функцій, доступних у професійній версії.

2. Visual Studio Professional:

- Платна версія з розширеними можливостями для професійних розробників і невеликих команд.

- Додаткові інструменти для співпраці та управління проєктами.

3. Visual Studio Enterprise:

- Платна версія з максимальним набором функцій для великих команд і підприємств.

- Включає розширені інструменти для архітектурного аналізу, тестування, управління вимогами та DevOps.

Переваги Visual Studio:

- Інтуїтивний інтерфейс: Призначений для спрощення розробки та підвищення продуктивності.

- Широка підтримка: Підтримка різноманітних мов програмування, платформ і технологій.

- Потужні інструменти для розробки: Включає потужні інструменти для редагування коду, відлагодження, тестування та розгортання.

- Інтеграція з іншими продуктами Microsoft: Легка інтеграція з Azure, GitHub, Microsoft SQL Server та іншими продуктами.

Visual Studio є потужним інструментом для розробників, який надає всі необхідні функції для створення, тестування та розгортання сучасних додатків. Завдяки своїй гнучкості та широкій підтримці різних технологій, Visual Studio підходить для розробників з різним рівнем досвіду та для проєктів будь-якої складності.

Для доступу до бази даних яке надає графічний інтерфейс використано SQL Server Management Studio.

SQL Server Management Studio (SSMS) — це інтегроване середовище управління для роботи з Microsoft SQL Server і Azure SQL Database. SSMS надає користувачам інструменти для адміністрування баз даних, написання та виконання SQL-запитів, розробки та налагодження процедур, а також для моніторингу продуктивності серверів.

Основні характеристики SSMS:

1. Адміністрування баз даних:

- Управління інстансами SQL Server, базами даних, таблицями, індексами та іншими об'єктами.

- Налаштування безпеки, управління користувачами та ролями, встановлення дозволів.

2. Інтерфейс користувача:

- Графічний інтерфейс, який дозволяє легко виконувати адміністрування баз даних без необхідності використовувати командний рядок.

- Інтуїтивно зрозумілі вікна та панелі для навігації по об'єктах бази даних і виконання різних завдань.

3. Написання та виконання SQL-запитів:

- Вбудований редактор коду з підсвіткою синтаксису, автодоповненням та підказками IntelliSense.

- Можливість запуску, налагодження та оптимізації SQL-запитів, збережених процедур, функцій та тригерів.

4. Моніторинг продуктивності:

- Інструменти для моніторингу стану серверів і баз даних, відстеження активності та виявлення проблем з продуктивністю.

- Звіти про продуктивність, які дозволяють аналізувати використання ресурсів та виявляти вузькі місця.

5. Розширене управління SQL Server Agent:

- Налаштування та управління завданнями автоматизації, такими як резервне копіювання, відновлення баз даних, та інші завдання обслуговування.

- Створення та управління алертами, операторами та робочими процесами.

6. Інструменти для роботи з даними:

- Виконання операцій імпорту та експорту даних, підключення до різних джерел даних, включаючи файли CSV, Excel та інші бази даних.

- Використання SQL Server Integration Services (SSIS) для створення та управління ETL-процесами (екстракція, трансформація, завантаження).

7. Безпека:

- Управління логінами та користувачами, встановлення та налаштування дозволів на рівні серверів та баз даних.

- Використання функцій шифрування та аудиту для забезпечення безпеки даних.

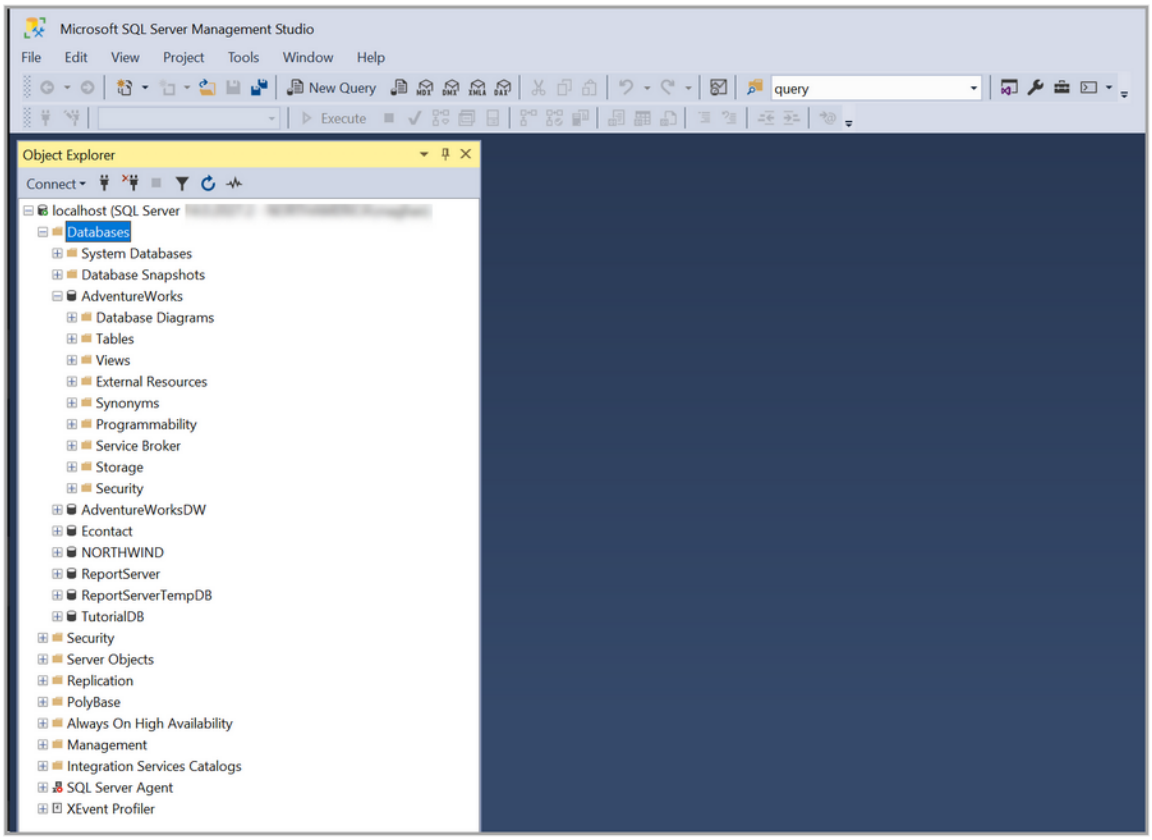


Рисунок 2.19 – Інтерфейс SQL Server Management Studio

Переваги SSMS

1. Зручність використання:

- Графічний інтерфейс, який полегшує виконання складних завдань без необхідності глибоких знань SQL.

- Інтуїтивно зрозуміле середовище, яке дозволяє швидко навчитися та ефективно використовувати інструменти SSMS.

2. Потужні інструменти для розробки:

- Редактор коду з розширеними функціями для написання, редагування та відлагодження SQL-запитів та збережених процедур.

- Підтримка інтеграції з системами контролю версій для управління змінами коду.

3. Моніторинг та оптимізація:

- Інструменти для детального моніторингу продуктивності серверів та баз даних, що дозволяють виявляти та вирішувати проблеми з продуктивністю.

- Звіти та інструменти для аналізу продуктивності запитів та оптимізації роботи баз даних.

4. Адміністративні функції:

- Можливості для повного адміністрування серверів та баз даних, включаючи резервне копіювання, відновлення, налаштування безпеки та автоматизацію завдань.

- Інтеграція з іншими інструментами та службами Microsoft, такими як Azure, для розширених можливостей управління.

Використання та застосування:

1. Адміністратори баз даних (DBA):

- Використовують SSMS для повсякденного управління базами даних, моніторингу продуктивності та виконання завдань обслуговування.

- Налаштовують безпеку, керують резервними копіями та відновленням даних, автоматизують завдання обслуговування.

2. Розробники:

- Використовують SSMS для написання, тестування та оптимізації SQL-запитів, збережених процедур і тригерів.

- Використовують інтегровані інструменти для налагодження та профілювання коду.

3. Аналітики даних:

- Використовують SSMS для виконання складних запитів, аналізу даних та створення звітів.

- Імпортують та експортують дані з різних джерел, інтегрують дані для подальшого аналізу.

SSMS є потужним інструментом для управління SQL Server та Azure SQL Database, що надає широкий спектр функцій для адміністрування, розробки та аналізу даних. Завдяки своїм можливостям, SSMS є незамінним інструментом для адміністраторів баз даних, розробників та аналітиків.

Node.js – це середовище виконання JavaScript із відкритим вихідним кодом, яке дозволяє розробникам створювати програми на стороні сервера за допомогою JavaScript. Він пропонує велику кількість сторонніх бібліотек і пакетів, що робить його ідеальним вибором для розробки масштабованих і ефективних веб-застосунків. Node.js відомий своєю високою продуктивністю і керованою подіями архітектурою, що робить його придатним для програм реального часу.

Blazor – це платформа для створення веб-застосунків на стороні клієнта з відкритим кодом для створення односторінкових застосунків (SPA) та великих за обсягом рішень за допомогою .NET і C# [1]. Це дозволяє розробникам створювати інтерактивні та адаптивні веб-інтерфейси користувача за допомогою компонентного підходу. Blazor дозволяє розробникам писати код на C# і надає такі функції, як зв’язування даних, впровадження залежностей і маршрутизація.

ASP.NET – це надійна структура веб-застосунків, яка надає розробникам широкий спектр інструментів і функцій. Це популярний вибір для програм корпоративного рівня завдяки своїй надійності, безпеці та

масштабованості. Однак це технологія, яка може не підійти розробникам, які віддають перевагу рішенням з відкритим кодом.

Blazor є відносно новим фреймворком, який підтримує створення багаторазових компонентів. Blazor можна використовувати для створення SPA, прогресивних веб-застосунків (PWA) і гібридних мобільних застосунків. Через його сучасність і зручний компонентний підхід перевагу віддано йому.

Отже, було виконано моделювання програмного забезпечення та створено діаграми і таблиці, що описують архітектуру веб-додатку. Були створені BPMN моделі для опису процесу взаємодії клієнтів із програмним продуктом. Був обраний підхід для розробки програмного забезпечення та створена його архітектура. Також описано основні бібліотеки, що будуть використовуватись під час створення та сформульовано таблиці БД, описано їх зв’язок між собою. Обрано програмні інструменти, що будуть використовуватись під час роботи зі створенням програмного забезпечення.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПРОЕКТНИМИ РЕСУРСАМИ

3.1. Розробка структурної схеми класів

На рисунку 3.1 відображена структурна схема класів, за якою було створено основний функціонал системи.

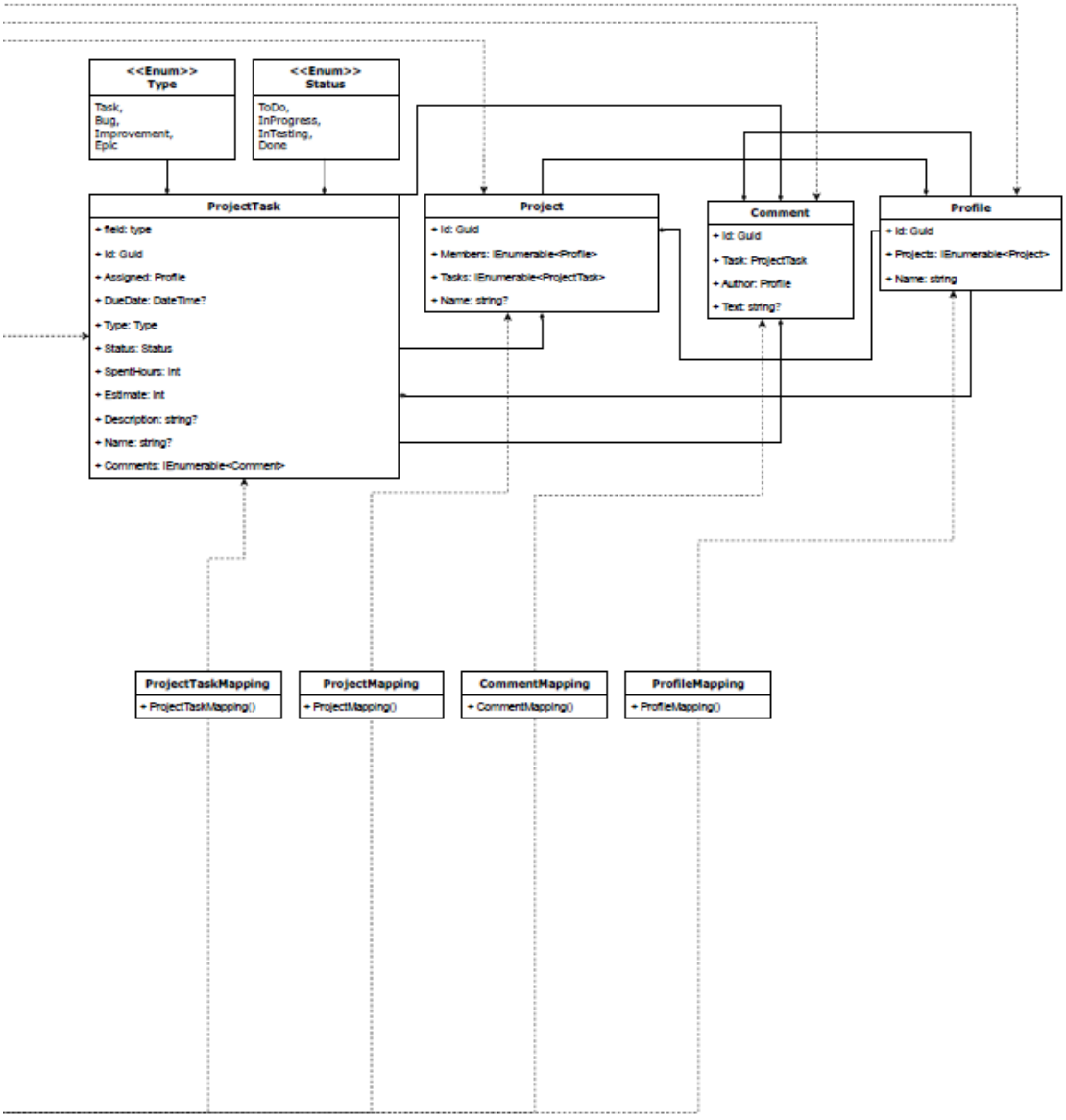


Рисунок 3.1 - Структурна схема класів (права частина)

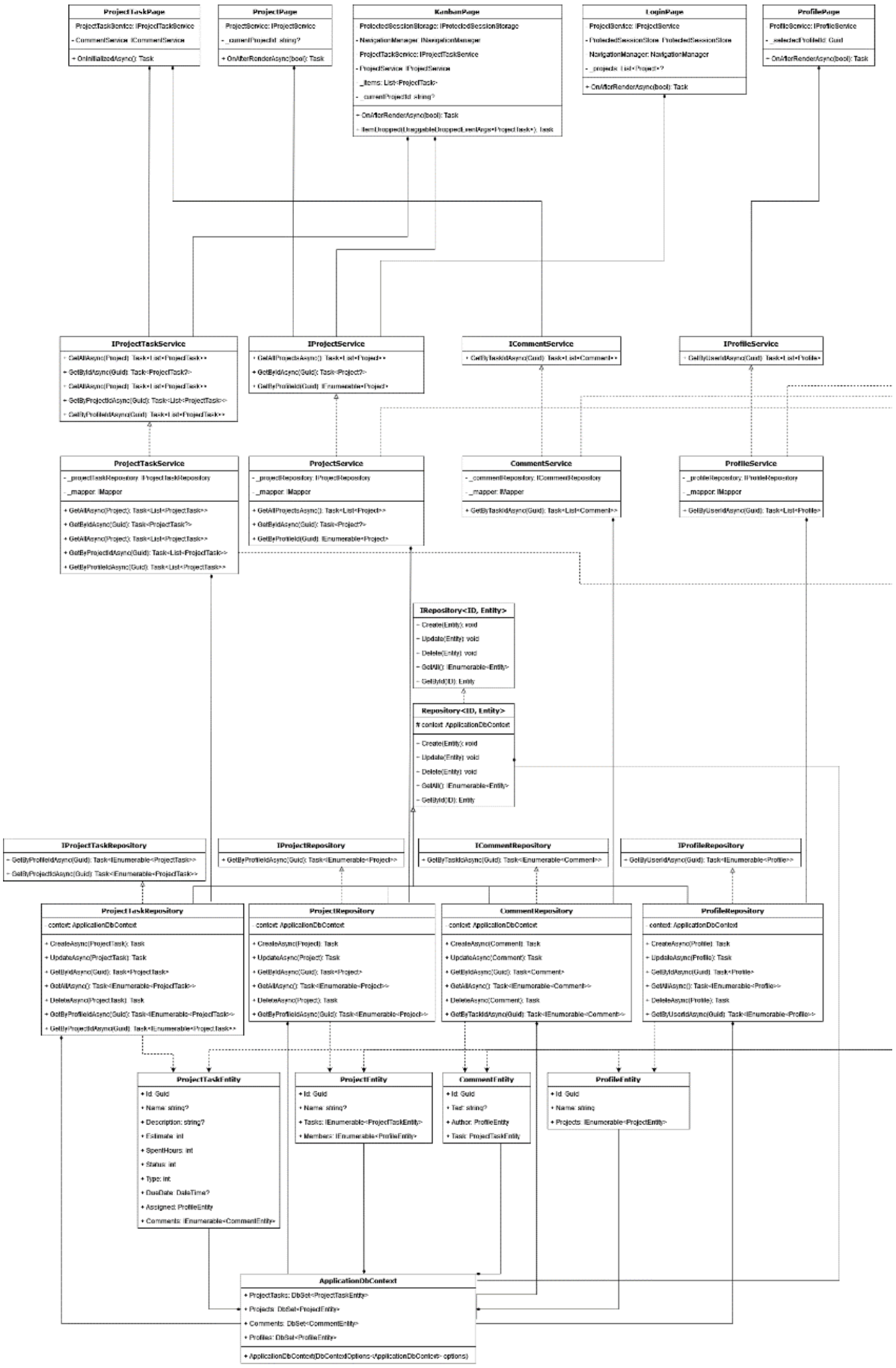


Рисунок 3.1 - Структурна схема класів (ліва частина)

Для взаємодії з БД використовується бібліотека EntityFramework, що вирішує проблеми одночасних запитів до БД та дозволяє використовувати веб-застосунок в асинхронному режимі. Також, вона надає зручний інтерфейс для того, щоб абстрагуватися від конкретної БД і, у разі потреби, замінити її.

Для мапінга сутностей в моделі використовується AutoMapper, що спрощує процес створення профілів мапінгу та дозволяє зручно відокремити алгоритм від його застосування.

3.2. Розробка інтерфейсу для реєстрації та авторизації користувача

Сторінка реєстрації є ключовим компонентом будь-якої системи, що забезпечує користувачам можливість створити обліковий запис і отримати доступ до послуг та функцій системи. Вона допомагає забезпечити безпеку, персоналізувати досвід користувачів, а також надати адміністраторам інструменти для управління користувачами і моніторингу активності.

Сторінка реєстрації має наступні поля:

– пошта;

– псевдонім;

– ім’я;

– пароль;

– підтвердження паролю.

Якщо користувач вводить некоректні дані, то поле підсвічується червоним красним кольором. Система не повинна дати змогу зареєструватись, поки хоча б одно з полів має некоректні дані. Система не дає реєструвати декілька акаунтів з одною поштою або псевдонімом. Сторінка реєстрації показана на рисунку 3.2.

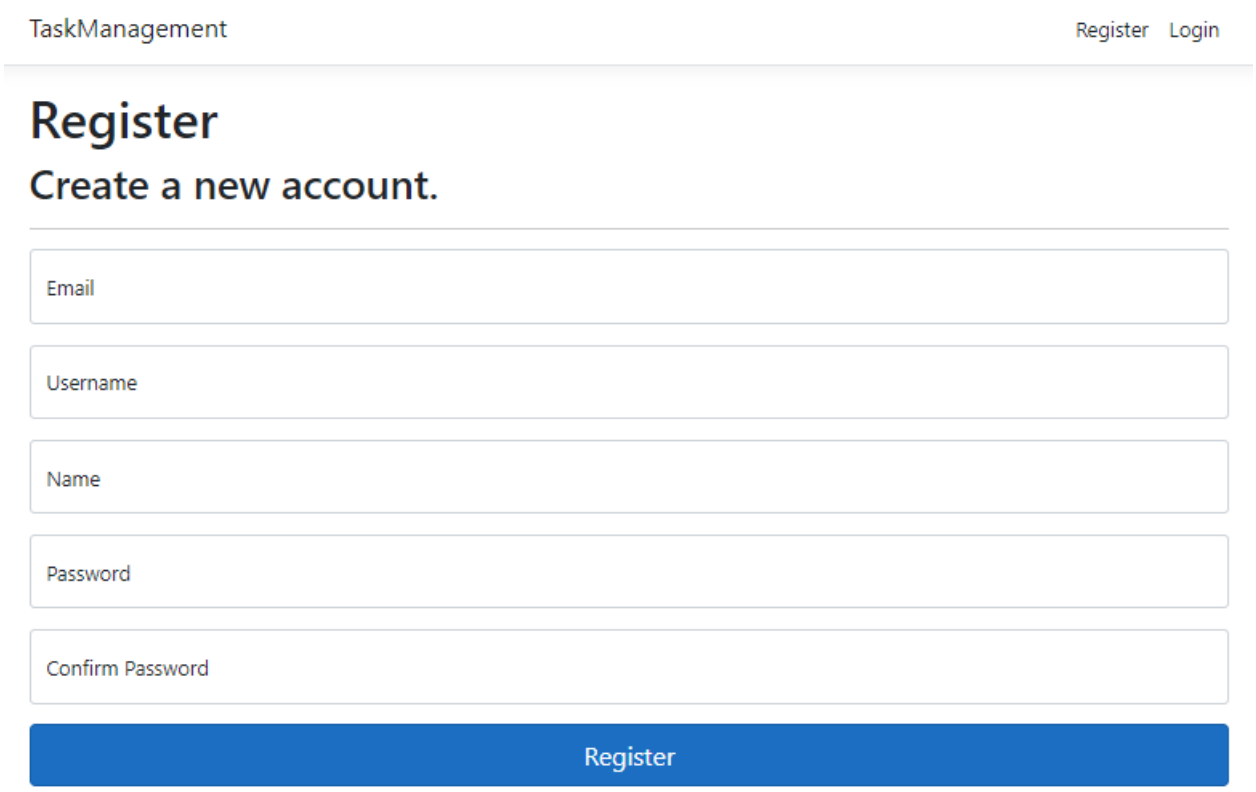


Рисунок 3.2 – Сторінка реєстрації

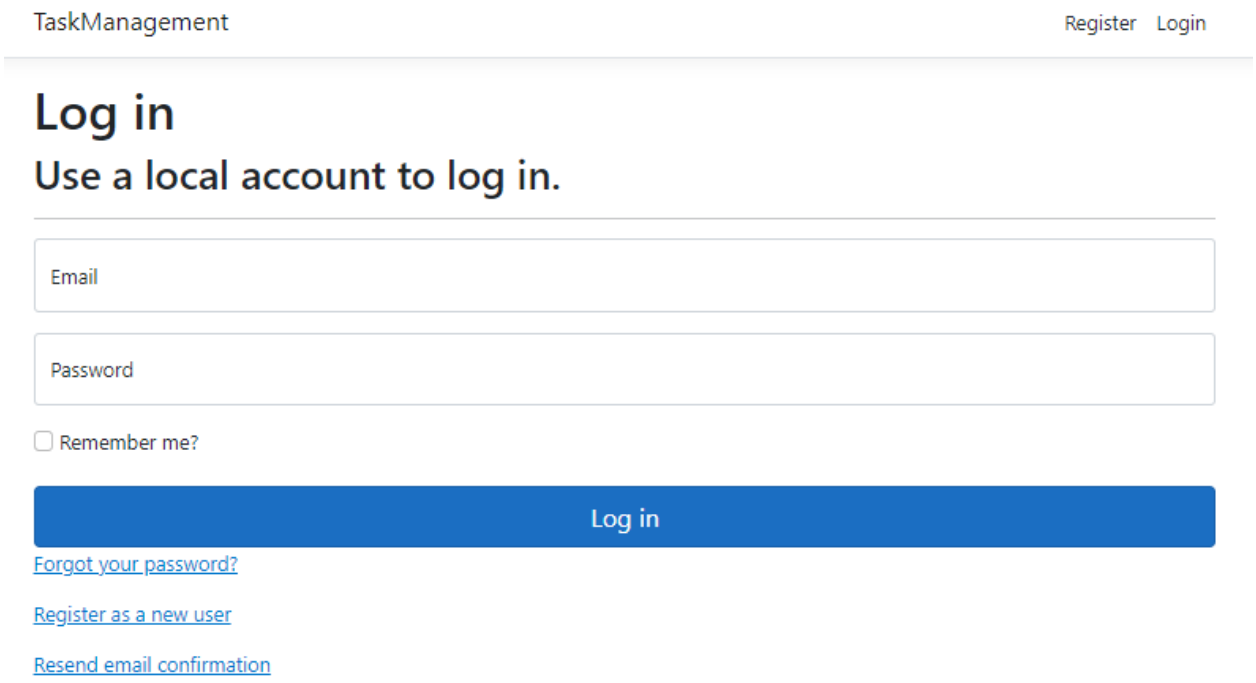


Рисунок 3.3 – Сторінка авторизації

Сторінка авторизації має наступні поля:

– пошта;

– пароль.

Якщо користувач вводить некоректні дані, то система повинна видати попередження та попросити користувача ввести дані ще раз. Сторінка авторизації відображено на рисунку 3.3.

Для реєстрації користувач вводить коректні дані: пошту, псевдонім, ім’я, пароль та підтвердження паролю.

Система не видає ніяких попереджень, тому що дані правильні та можуть бути оброблені серверною стороною без помилок (рисунок 3.4).

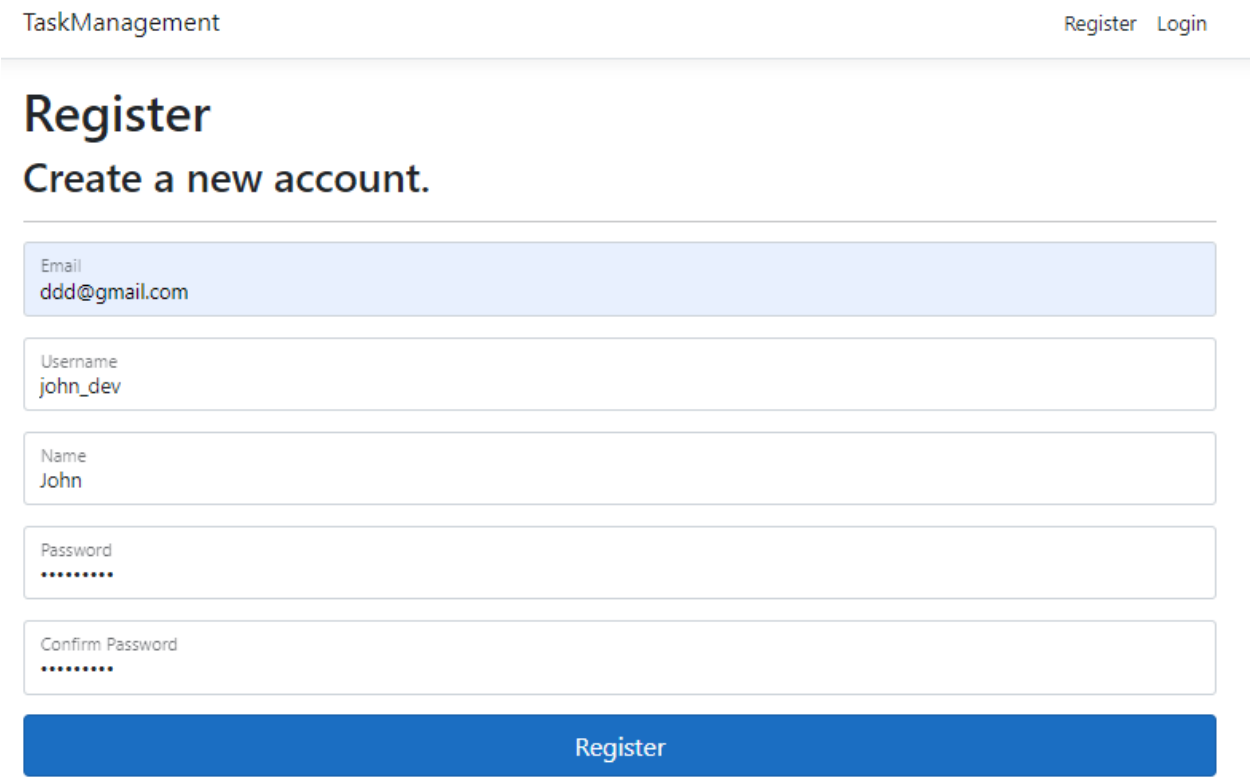


Рисунок 3.4 – Заповнена форма реєстрації

Якщо ввести некоректні дані, то форма повинна видати попередження та не дати можливості гостю зареєструватися, поки той не введе коректні дані (рисунок 3.5).

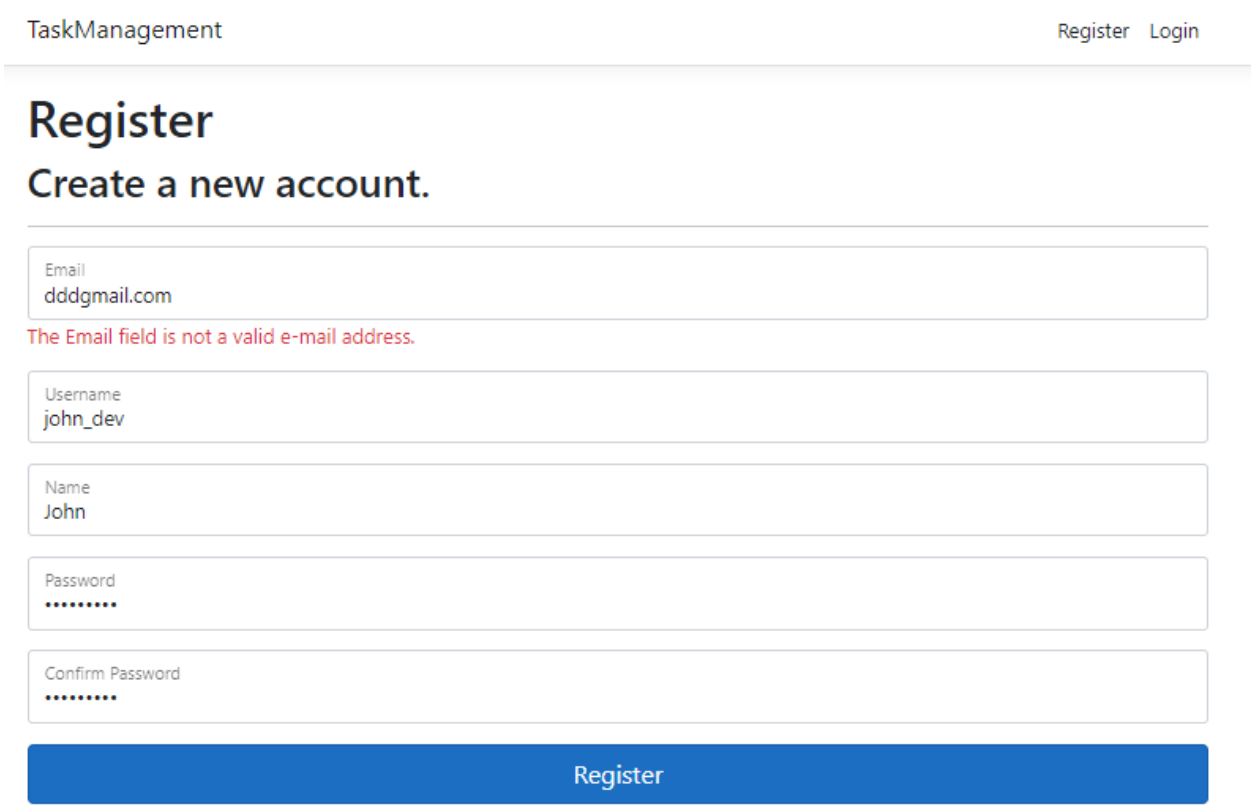


Рисунок 3.5 – Форма реєстрації з некоректними даними

3.3. Розробка інтерфейсу користувача системи управління проектними ресурсами

Після реєстрації система створює начальний проект та пропонує створити ще один. Користувач вводить ім’я проекту та запрошує до нього інших користувачів за їх псевдонімами (рисунок 3.6).

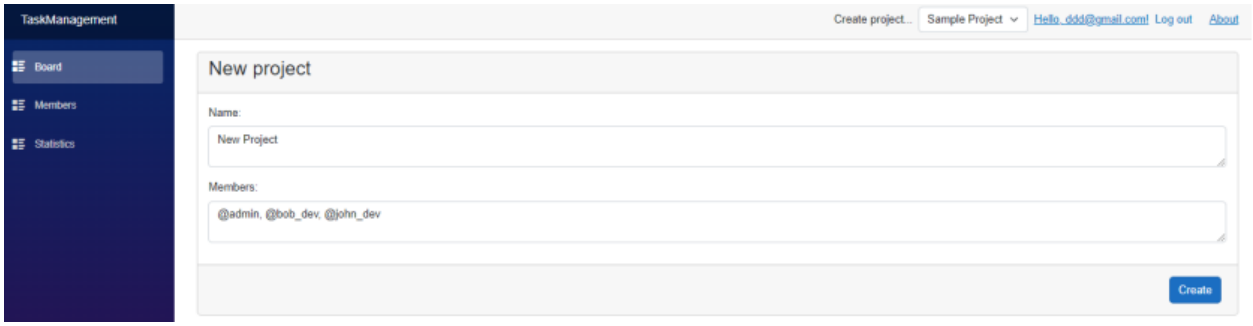


Рисунок 3.6 – Заповнена форма створення проекту

Після створення проекту система пропонує створити задачі. Користувач вводить ім’я, опис, оцінку, затрачений час, дату завершення задачі, виконавця, статус та тип (рисунок 3.7).

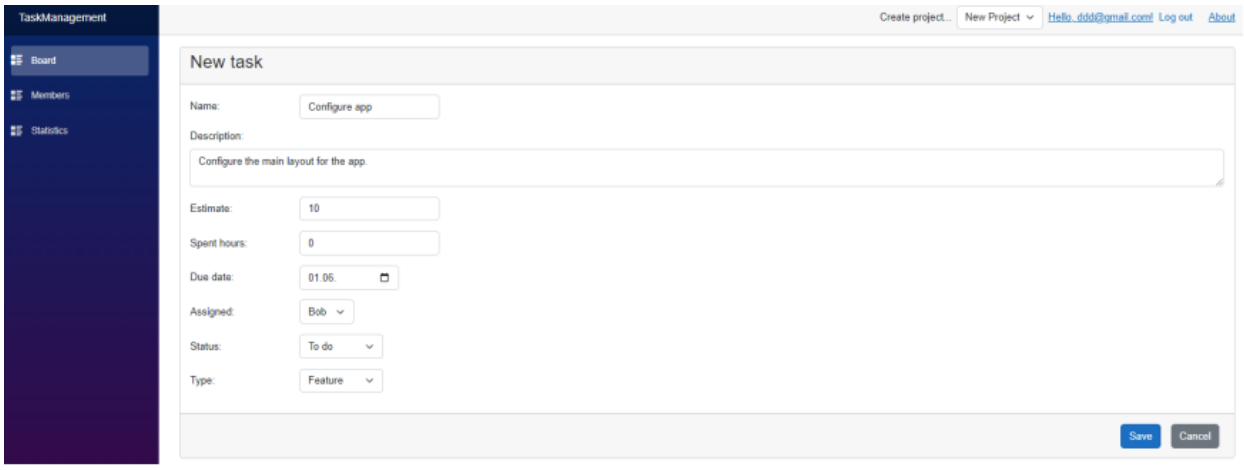


Рисунок 3.7 – Заповнена форма створення задачі

Таким чином користувач створює декілька задач та обирає різних виконавців, назначає їм різні статуси.

Дошка з задачами відображає список задач у форматі kanban. Кожна картка задачі повинна мати назву та опис. Кожна задача відображається в колонці, що відповідає її статусу. Усього є чотири колонки: to do, in progress, in testing, done. Окрім цього, зверху відображається шкала прогресу, що відображає кількість завершених задач поділену на кількість задач в цілому (рисунок 3.8).

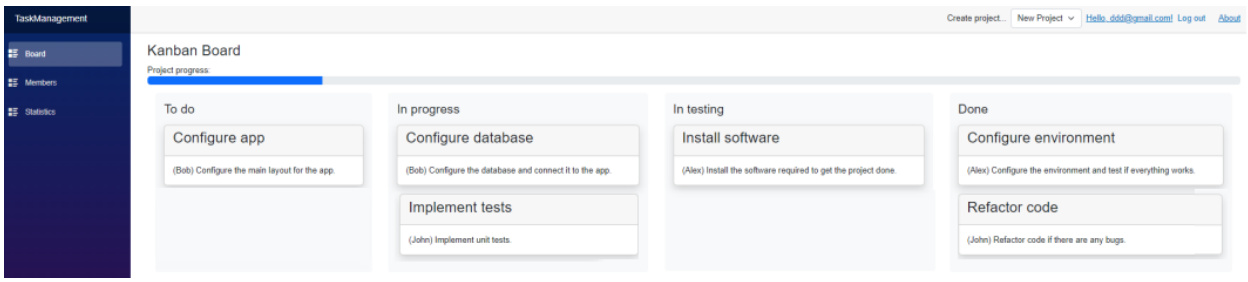


Рисунок 3.9 – Kanban дошка проекту

Сторінка з розподіленням ресурсів відображає всіх учасників проекту та їх завантаженість. Шкала прогресу рахує дані виходячи з того, що кожен розробник має 100 годин роботи. Кожна задача, над якою ведеться робота, займає кількість годин, що вказано в самій задачі.

Сторінка також відображає всі задачі, якими займаються або займались розробники на обраному проекті. Синім кольором відображаються задачі, що ще в прогресі, а сірим – що вже були виконані (рисунок 3.10).

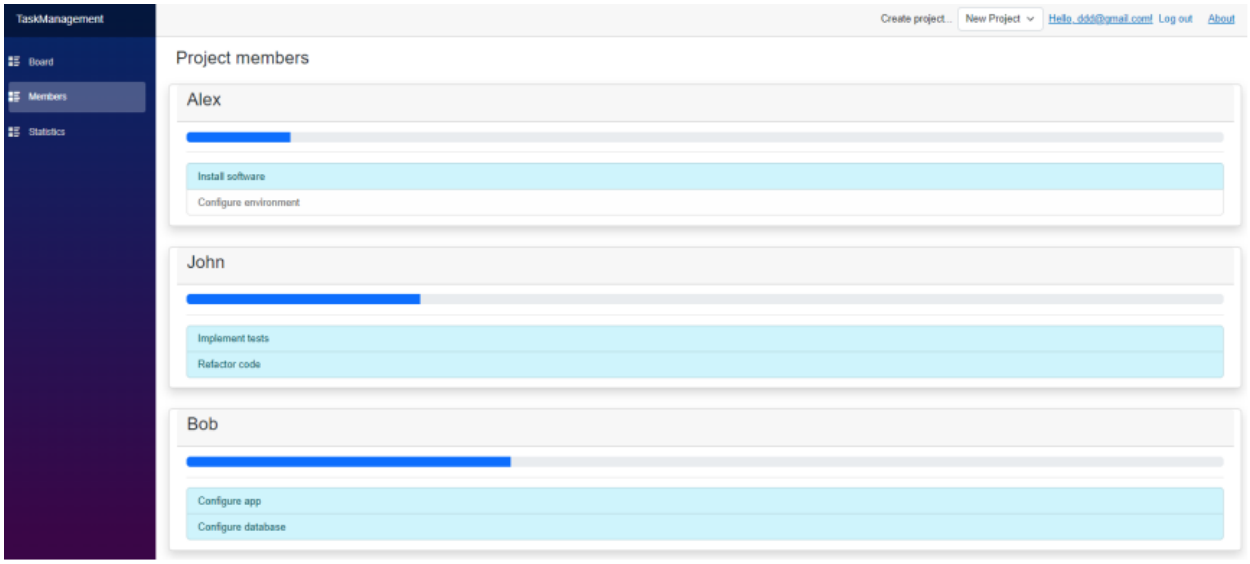


Рисунок 3.10 – Сторінка розподілу завантаженості учасників команди

Сторінка зі статистикою відображає ту ж саму шкалу прогресу, що й головна проектна дошка. Окрім цього, вона відображає статус проекту, дату його створення та кількість днів, за котрі проект було завершено. Також, на сторінці можна побачити індивідуальну статистику для кожного з розробників. Вона повинна включати в себе кількість виконаних задач, кількість прогнозованих годин роботи та реальну кількість потрачених годин. З цих параметрів розраховується ефективність кожного учасника команди (рисунок 3.11).

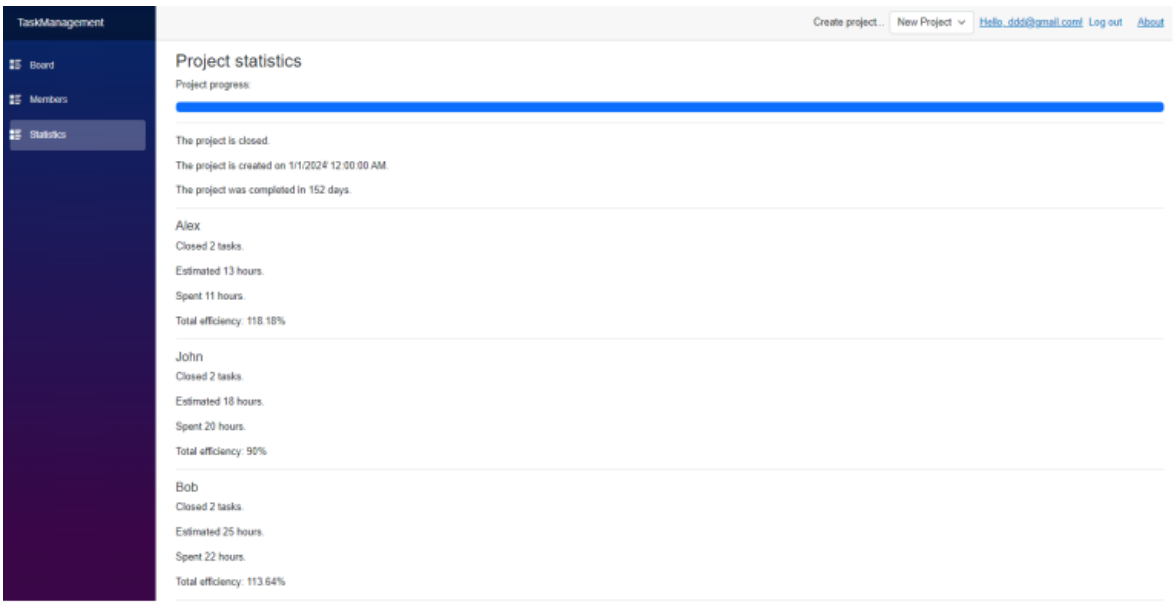


Рисунок 3.11 - Сторінка розподілу завантаженості

Всього веб-додаток має дві сторінки звітів: звіт з прогресу та звіт з розподілення ресурсів. Вони повинні показувати ту ж саму статистику, що й сторінки розподілення ресурсів та статистики, але мати інший формат, що більш пригідний для експорту (рисунок 3.12).

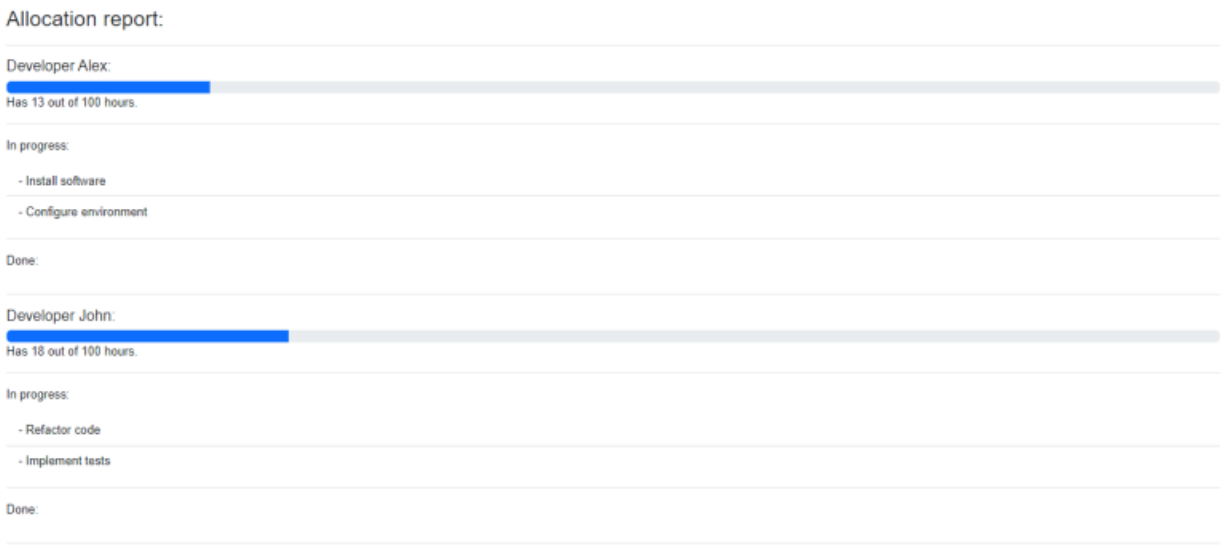


Рисунок 3.12 – Прототип звіту з розподілу ресурсів

Користувач закриває всі задачі та закриває проект. Після цього отримує повну статистику ефективності розподілу ресурсів на проекті. Ця статистика

містить детальну інформацію про проект та ефективність окремих членів команди.

Інформація про членів команди містить кількість виконаних задач, їх оцінки та реальний витрачений час. Рахується їх ефективність (рисунок 3.11).

Також можна отримати прогноз щодо наступного проекту. Сторінка прогнозування повинна мати той ж вигляд, що й сторінка статистики, але дані, що виводяться на ній, повинні бути прогнозом для наступного проекту. За основу для прогнозування треба брати дані, отримані на обраному проекті. Для цього користувач вказує дані про наступний проект та отримує прогноз щодо його тривалості та ефективності його членів (рисунок 3.13).

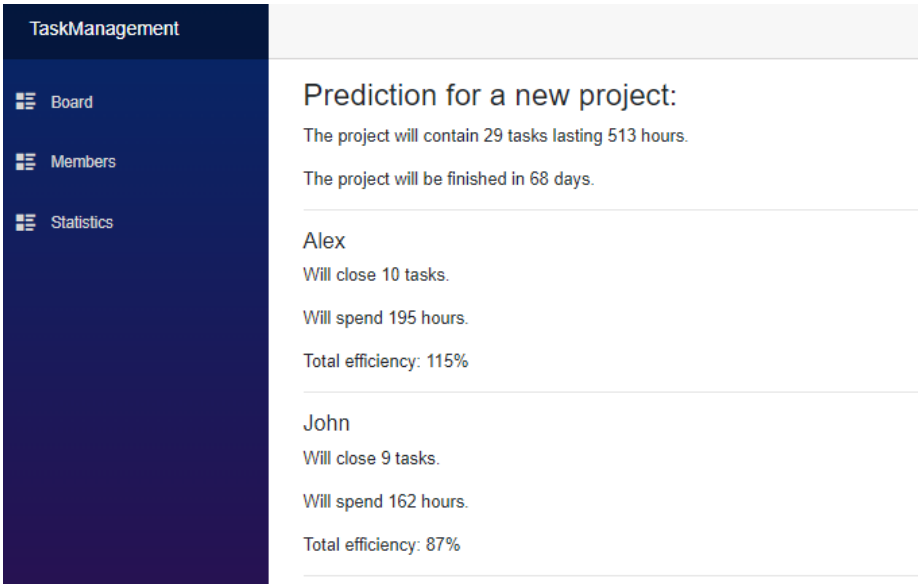


Рисунок 3.13 – Сторінка прогнозу для майбутнього проекту

Функціональні можливості для користувача:

– приймати участь в проектах;

– коментувати задачі в проекті;

– відмічати інших користувачів;

– отримувати повідомлення;

– змінювати дані про свою задачу;

– змінювати свій профіль (ім’я, пароль).

Функціональні можливості для менеджера проекту:

– увесь функціонал, що й користувача;

– створення проектів;

– створення задач;

– додавання членів команди в проект;

– призначення задачі;

– зміна будь-яких даних про проект та задачі;

– оцінка продуктивності;

– розподіл ресурсів між задачами (членів команди);

– прогнозування завантаженості для майбутнього проекту;

– отримання звітів з прогресу та розподіленню ресурсів.

Адміністратор має увесь функціонал, що й менеджер проекту, але з можливість видаляти акаунти інших користувачів.

3.4. Реалізація процесів щодо забезпечення безпеки даних

Одним із ключових аспектів безпеки даних є забезпечення доступу до програми та пов’язаних із нею даних лише авторизованих осіб. Поширені вразливості включають слабкі паролі, неправильне керування сеансами та неадекватний контроль доступу на основі ролей користувача. Щоб пом’якшити ці ризики, цей додаток застосовує політику надійних паролів, безпечні методи керування сеансами та використовує механізми контролю доступу для обмеження несанкціонованого доступу до конфіденційних даних.

Дані, що передаються між веб-застосунком та її користувачами, захищені, щоб запобігти перехопленню або несанкціонованому доступу. Для шифрування даних під час передачі використовується HTTPS. Крім того, конфіденційні дані, що зберігаються в базах даних або інших системах зберігання, зашифровані.

Запропонований веб-застосунок реалізує методи перевірки введених даних і використовує безпечні методи кодування для дезінфекції даних, наданих користувачем, перед їх обробкою або зберіганням.

В цьому додатку підтримується в актуальному стані базовий стек програмного забезпечення, регулярно застосовується виправлення безпеки та оновлення.

Також, реалізовано регулярне резервне копіювання даних і створено план аварійного відновлення для забезпечення доступності та цілісності даних.

ВИСНОВКИ

В дипломній роботі здійснена розробка програмного забезпечення системи управління проектними ресурсами. Основною метою представленої роботи є процес автоматизації управління проектними рішеннями за допомогою нейронної мережі, а саме: здійнення обчислення ефективності працівників, побудова звітності щодо виконання графіка тощо.

Для виконання поставленого завдання було здійснено аналіз вимог до програмного забезпечення, розглянуто існуючі технології та проекти, що вирішують задачі, поставлені в роботі. Проводиться аналіз та опис обраної предметної області, аналізуються вимоги до програмного забезпечення та створюється постановка задачі. Виконано моделювання програмного забезпечення, створено архітектуру та її конструюванню. Окремо проводиться аналіз безпеки даних. Виконано вибір методології тестування, проведено аналіз якості та описано процеси тестування, які були використані при створенні програмного забезпечення системи управління проектними ресурсами. Створений програмний продукт реалізує основні функції для групового управління проектами, а також надає доступ до проектних задач різних рівнів. Користувач має змогу переглядати та змінювати свої особисті дані, переглядати всі свої поточні проекти, а також списки всіх своїх проектних завдань і статус їх виконання. Менеджери проектів за рахунок розподілення ролей в системі, мають змогу переглядати всі завдання на тих проектах, на яких вони мають управлінські функції, а також додавати нові завдання і видаляти вже існуючі. Розробка призначена для спрощення процесу управління задачами на проектах, покращення ефективності, зменшення помилок та покращення спілкування між членами команди та зацікавленими сторонами. Використовуючи технології для оптимізації управління проектами, організації можуть виконувати успішні проекти вчасно та в рамках бюджету.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бушуєва Н. С. Управління проектами та програмами організаційного розвитку : навч. посібн. / Н.С. Бушуєва, Ю.Ф. Ярошенко, Р.Ф. Ярошенко. – К. : Саммит-Книга, 2010. – 200 с
2. Frame Davidson J. The New Project Management. San Francisco: Jossey-Bass Publishers, 1994. — 328 p.
3. Kerzner,H. Project Management: a systems approach to planning, scheduling, and controlling. 4th ed., New York, Van Nostrand Reinhold, 1992. — 1028 p.
4. Project Management Body of Knowledge (PM BOK), Project Management Institute, Drexel Hill, Pennsylvania, 1987. — 756 p
5. Kochenderfer M. J. Algorithms for Optimization / Mykel J. Kochenderfer, Tim A. Wheeler – The MIT Press, 2019. – 520 ст.
6. Engelbrecht A. P. Computational Intelligence: An Introduction: 2nd Edition / Andries P. Engelbrecht – Wiley, 2007. – 640 ст.
7. How Can Artificial Intelligence Improve Resource Optimization. URL : https://www.aiplusinfo.com/blog/how-can-artificial-intelligence-improve-resource-optimization/
8. Optimization Methods for Resource Allocation and Machine Learning Applications. – URL : https://escholarship.org/uc/item/7wt8n5wk
9. Joseph Phillips - Project Management for Small Business: A Streamlined Approach from Planning to Completion. 2012.
10. Event Driven Architecture (EDA) | Download Scientific Diagram - https://www.researchgate.net/figure/Event-Driven-Architecture-EDA_fig3_324219695
11. Wright, T. B. Blazor WebAssembly by Example: A project-based guide to building web apps with .NET, Blazor WebAssembly, and C#: 1st Edition / Toi B. Wright – Packt Publishing, 2021. – 266 ст.

12. Sagar R. Jira Quick Start Guide: Manage your projects efficiently using the all-new Jira / Ravi Sagar - Packt Publishing, 2019. – 162 ст.
13. Складнощі тестування мікросервісів та що з ними робити | DOU - https://dou.ua/lenta/articles/difficulties-in-microservices-testing/
14. Документація Deepsource. URL : https://docs.deepsource.com/docs/introduction
15. Документація JetBrains Rider. URL: https://www.jetbrains.com/help/rider/Introduction.html
16. Thomas E. Cloud Computing: Concepts, Technology & Architecture: 1st Edition / Erl Thomas, Puttini Ricardo, Mahmood Zaigham – Pearson, 2013. – 747 ст.
17. Freeman, A. Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages: 8th Edition / Adam Freeman. – Apress, 2020. – 2151 ст.
18. Turnbull J. Cloud Computing: Concepts, Technology & Architecture: 1st Edition / James Turnbull 2014. – 388
19. Heller P. Automating Workflows with GitHub Actions: Automate software development workflows and seamlessly deploy your applications using GitHub Actions / Priscila Heller – Packt Publishing, 2021. – 216 ст.
20. Fowler M. Patterns of Enterprise Application Architecture 1st Edition / Martin Fowler – Addison-Wesley Professional, 2002. – 560 ст.
21. Harold Kerzner - Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 2017.
22. Lim G. Beginning Node.js, Express & MongoDB Development / Greg Lim – Опубліковано самостійно, 2019. – 155 ст.
23. William P. Athayde - Project Management Essentials: A Quick and Easy Guide to the Most Important Concepts and Best Practices for Managing Your Projects Right. 2009.

24. Wolff D. Get Things Done with Trello: Your Quick Access to Productivity and Success includes a Step-by-Step Guide to Set Up and Implement Trello / Dominic Wolff - Organized Living Press, 2014. – 110 ст.
25. Charles G. Cobb - The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach. 2015.
26. Cynthia Snyder Dionisio - A Project Manager's Book of Forms: A Companion to the PMBOK Guide. 2022.

ДОДАТКИ

Додаток А

Програмні лістинги

ProjectService.cs

```
using AutoMapper;
using TaskManagement.DA;
using TaskManagement.Models;
namespace TaskManagement.BL;
public class ProjectService
{
private readonly IProjectRepository _repository;
private readonly IMapper _mapper;
public ProjectService(IProjectRepository repository, IMapper
mapper)
{
_repository = repository;
_mapper = mapper;
}
public async Task<List<Project>> GetAllProjectsAsync()
{
return _mapper.Map<List<Project>>(await
_repository.GetAllAsync());
}
public async Task<Project?> GetByIdAsync(Guid id)
{
var result = await _repository.GetByIdAsync(id);
if (result == null) return null;
return _mapper.Map<Project>(result);
}
public async Task CreateAsync(Project project)
{
var entity = _mapper.Map<Data.Project>(project);
var profileProjects = new List<Data.ProfileProject>();
foreach (var member in project.Members!)
```

```
{
profileProjects.Add(new Data.ProfileProject
{
ProfileId = member.Id,
ProjectId = project.Id
});
}
entity.ProfileProjects = profileProjects;
await _repository.CreateAsync(entity);
}
public async Task<List<Project>> GetByProfileIdAsync(Guid id)
{
var result = await _repository.GetByProfileIdAsync(id);
return _mapper.Map<IEnumerable<Project>>(result)
.OrderBy(p => p.CreatedDate)
.ToList();
}
}
```

ProjectTaskService.cs

```
using AutoMapper;
using TaskManagement.DA;
using TaskManagement.Models;
namespace TaskManagement.BL;
public class ProjectTaskService
{
private readonly IProjectTaskRepository _repository;
private readonly IMapper _mapper;
public ProjectTaskService(IProjectTaskRepository repository,
IMapper mapper)
{
_repository = repository;
_mapper = mapper;
}
```

```
public async ValueTask<List<ProjectTask>>
GetByProjectIdAsync(Guid id)
{
return _mapper.Map<List<ProjectTask>>(await
_repository.GetByProjectIdAsync(id));
}
public async ValueTask<ProjectTask?> GetByIdAsync(Guid id)
{
return _mapper.Map<ProjectTask>(await
_repository.GetByIdAsync(id));
}
public async Task UpdateAsync(ProjectTask task)
{
await
_repository.UpdateAsync(_mapper.Map<Data.ProjectTask>(task));
}
}
```

ApplicationDbContext.cs

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TaskManagement.Data;
namespace TaskManagement.DA;
public class ApplicationDbContext : IdentityDbContext
{
public DbSet<Comment> Comments { get; set; }
public DbSet<Profile> Profiles { get; set; }
public DbSet<Project> Projects { get; set; }
public DbSet<ProjectTask> ProjectTasks { get; set; }
public
ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
: base(options)
{
}
```

```
protected override void OnModelCreating(ModelBuilder
modelBuilder)
{
modelBuilder.Entity<Profile>()
.HasMany(e => e.ProfileProjects)
.WithOne(e => e.Profile);
modelBuilder.Entity<Project>()
.HasMany(e => e.ProfileProjects)
.WithOne(e => e.Project);
base.OnModelCreating(modelBuilder);
}
}
```

CommentRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public class CommentRepository : Repository<Guid, Comment>,
ICommentRepository
{
public CommentRepository(ApplicationDbContext context) :
base(context)
{
}
public async Task<IEnumerable<Comment>> GetByTaskIdAsync(Guid
id)
{
return await Task.Run(() => _context.Set<Comment>().Where(c =>
c.Task != null && c.Task.Id == id));
}
}
```

ICommentRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public interface ICommentRepository : IRepository<Guid, Comment>
```

```
{
Task<IEnumerable<Comment>> GetByTaskIdAsync(Guid id);
}
```

Файл IProfileRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public interface IProfileRepository : IRepository<Guid, Profile>
{
Task<IEnumerable<Profile>> GetByProjectIdAsync(Guid id);
Task<IEnumerable<Profile>> GetByUserIdAsync(string id);
Task<IEnumerable<Profile>> GetByUsernameAsync(string username);
}
```

ProjectRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public interface IProjectRepository : IRepository<Guid, Project>
{
Task<IEnumerable<Project>> GetByProfileIdAsync(Guid id);
}
```

ProjectTaskRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public interface IProjectTaskRepository : IRepository<Guid,
ProjectTask>
{
Task<IEnumerable<ProjectTask>> GetByProfileIdAsync(Guid id);
Task<IEnumerable<ProjectTask>> GetByProjectIdAsync(Guid id);
}
```

IRepository.cs

```
namespace TaskManagement.DA;
public interface IRepository<ID, Entity>
```

```
{
Task CreateAsync(Entity entity);
Task UpdateAsync(Entity entity);
Task DeleteAsync(Entity entity);
Task<IEnumerable<Entity>> GetAllAsync();
Task<Entity?> GetByIdAsync(ID id);
}
```

ProjectTaskRepository.cs

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public class ProjectTaskRepository : Repository<Guid,
ProjectTask>, IProjectTaskRepository
{
public ProjectTaskRepository(ApplicationDbContext context) :
base(context)
{
}
public async Task<IEnumerable<ProjectTask>>
GetByProfileIdAsync(Guid id)
{
return await Task.FromResult(_context
.Set<ProjectTask>()
.Where(c => c.Assigned != null && c.Assigned.Id == id));
}
public async Task<IEnumerable<ProjectTask>>
GetByProjectIdAsync(Guid id)
{
return await Task.FromResult(_context
.Set<ProjectTask>()
.Where(c => c.Project!.Id == id));
}
```

Repository.cs

```
using Microsoft.EntityFrameworkCore;
```

```
using TaskManagement.Data;
namespace TaskManagement.DA;
public class Repository<ID, Entity> : IRepository<ID, Entity>
where ID : IEquatable<ID>
where Entity : BaseEntity<ID>
{
protected ApplicationDbContext _context;
public Repository(ApplicationDbContext context)
{
_context = context;
}
public virtual async Task CreateAsync(Entity entity)
{
_context.Set<Entity>().Attach(entity!);
await _context.SaveChangesAsync();
}
public async Task DeleteAsync(Entity entity)
{
_context.Remove(entity!);
await _context.SaveChangesAsync();
}
public async Task<IEnumerable<Entity>> GetAllAsync()
{
return await Task.FromResult(_context.Set<Entity>());
}
public async Task<Entity?> GetByIdAsync(ID id)
{
return await Task.FromResult(_context
.Set<Entity>()
.FirstOrDefault(e => e.Id!.Equals(id)));
}
public async Task UpdateAsync(Entity entity)
{
var localEntry = _context.Set<Entity>()
.Local
```

```
.FirstOrDefault(e => e.Id!.Equals(entity.Id));
if (localEntry != null)
_context.Entry(localEntry).State = EntityState.Detached;
_context.Entry(entity).State = EntityState.Modified;
await _context.SaveChangesAsync();
}
}
```

BaseEntity.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace TaskManagement.Data;
public class BaseEntity<ID>
{
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
[Key]
public ID? Id { get; set; }
}
```

Comment.cs

```
using System.ComponentModel.DataAnnotations.Schema;
namespace TaskManagement.Data;
public class Comment : BaseEntity<Guid>
{
public string? Content { get; set; }
public DateTime Date { get; set; }
public Guid AuthorId { get; set; }
public Guid TaskId { get; set; }
[ForeignKey(nameof(AuthorId))]
public virtual Profile? Author { get; set; }
[ForeignKey(nameof(TaskId))]
public virtual ProjectTask? Task { get; set; }
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Розробка програмного забезпечення системи управління проектними ресурсами”

Обсяг пояснювальної записки: 83 аркуші.

Дата закінчення роботи: 17 червня 2024 р.

Підпис студента _____