МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Онлайн-система управління проєктами

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 42
Білінська К. В.
(прізвище та ініціали)
Керівник: Федусенко О. В.
(прізвище та ініціали)
кандидат технічних наук, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 13 від 5.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Білінській Катерині Василівній

(прізвище, ім'я, по батькові)

1. Тема роботи онлайн-система управління проєктами, затверджена протоколом засідання кафедри від «11» листопада 2022 р. №4
2. Термін здачі студентом закінченого проекту 31 травня 2023 року
3. Вихідні дані до проекту

Дані про проєкти та завдання, інформація про користувачів та групи, в які вони об'єднані.

4. Зміст розрахунково-пояснювальної записки
   4.1. Аналітичний огляд досліджень та публікацій. Аналіз існуючих систем управління проєктами. Постановка задачі та функціональний аналіз.
   4.2 Аналіз основних функцій та проектування архітектури застосунку.
   4.3 Реалізація програмного забезпечення для управління проєктами та перевірка його працездатності.
   4.4 Формування висновків.
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
   5.1 Аналітичний огляд: актуальність поставленої задач та проблематика.
   5.2 Постановка задачі: функціональне моделювання та розробка вимог до системи.
   5.3 Проектні рішення: основні функції систему управління проєктами та архітектура застосунку.
   5.4 Програмна реалізація та огляд процесу тестування: інформаційне забезпечення, структурна схема підсистем, результати тестування.
   5.5 Висновки

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 13 лютого 2023 року

Керівник _____ / Федусенко О.В. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Білінська К.В. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Вивчення навчальної, довідкової, спеціальної літератури, стану питання.	13.02-22.02	
2	Підготовка матеріалів першого розділу. Аналіз існуючих рішень, формування постановки задачі та концепції роботи.	22.02-1.03	
3	Підготовка матеріалів другого розділу. Визначення функцій системи та розробка архітектури програмного застосунку.	1.03-15.03	
4	Підготовка матеріалів другого розділу. Інформаційний аналіз системи.	15.03-1.04	
5	Підготовка матеріалів третього розділу. Програмна реалізація модулів системи та їх тестування.	1.04-7.05	
6	Аналіз отриманих результатів. Написання висновків.	7.05-15.05	
7	Завершення роботи, її оформлення та подання на рецензію.	15.05-20.05	
8	Створення мультимедійної презентації захисту.	21.05	

Студент-дипломник _____ / Білінська К.В. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Федусенко О.В. /
(підпис) (ПІБ)

АНОТАЦІЯ

Білінська Катерина Василівна виконала випускну кваліфікаційну роботу на тему «Онлайн-система управління проєктами» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено детальний аналіз процесу управління проєктами, включаючи визначення завдань та їх розподіл, досліджено сучасні системи управління проєктами, розглянуто архітектуру системи, а також розроблено інформаційне та програмне забезпечення, яке надає ефективні механізми для роботи з проєктами.

Ключові слова: проєкт, завдання, управління.

ANNOTATION

The degree project: «Online Project Management System» has been completed by Kateryna Bilinska specialty 122 – «Computer Sciences».

The final qualification work carried out a detailed analysis of the project management process, including the definition of tasks and their distribution. Modern project management systems we researched, and system architecture was considered. Additionally, information and software solutions were developed, providing efficient mechanisms for project management.

Keywords: project, tasks, management.

ЗМІСТ

ВСТУП

Випускна кваліфікаційна робота присвячена реалізації системи управління проєктами. Застосунок повинен надати можливість для здійснення різноманітних операції із проєктами та завданнями. Управління проєктами вже давно є невід'ємною частиною будь-якого бізнесу, тому питання розробки системи, яка полегшить цей процес, є надзвичайно важливою задачею.

Об'єкт дослідження: основні процеси управління проєктами.

Предмет дослідження: способи автоматизації процесів управління проєктами.

Метою дипломної роботи є реалізація онлайн-системи управління проєктами. Для досягнення поставленої мети необхідно вирішити наступні задачі:

- провести аналіз сучасних проблем управління проєктами і дослідити дійсні системи;
- виокремити основні бізнес-процеси управління проєктами без використання системи;
- розробити загальну постановку задачі;
- сформувати основні функціональні та нефункціональні вимоги до інформаційної системи;
- описати основні процеси управління проєктами з використанням автоматизованої системи;
- провести функціональний аналіз системи за допомогою карти процесів;
- побудувати архітектуру системи управління проєктами, а також її інформаційне забезпечення;

- реалізувати програмний продукт, визначити специфікацію програмного забезпечення та описати графічний інтерфейс з користувачем;
- провести тестування інформаційної системи та провести докладний аналіз результатів тестових прикладів;
- зробити висновки на основі проведеної роботи та визначити можливості подальшого використання.

РОЗДІЛ 1. АНАЛІЗ ПРОЦЕСУ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Аналітичний огляд літератури за темою управління проєктів

Перед тим, як почати ознайомлення з системами управління проєктів, потрібно розібрати: а що ж собою являє проєкт? Інститут управління проєктами(PMI) дає наступне визначення: проєкт - це тимчасові зусилля взяті на себе для розробки продукту або надання сервісу[1]. Це означає що проект може бути виконано лише один раз, тобто він не може бути повторюваним. Бажано, що проєкт мав часові обмеження, визначений бюджет, зрозумілий обсяг роботи до виконання і чіткі вимоги, які повинні бути дотримані. Тут застосовано саме слово “бажано”, адже дуже рідко буває так, що реальний проєкт повністю відповідає зазначеному визначенню. Відомий гуру якості Джозеф Джуран у своїй роботі[2] описує проєкт, як проблему заплановану для вирішення, що є влучною дефініцією, адже кожен проєкт проводиться для того, щоб вирішити певний вид проблеми для бізнесу.

З проєктами все зрозуміло, але що означає управляти ними і чому це так важливо? Управління проєктами - це застосування знань, навичок, інструментів і технік для проєктування дій, щоб досягти вимог проєкту. Управління проєктами здійснюється через застосування і інтеграцію наступних процесів: ініціювання, планування, виконання, відстеження і контролювання та закриття.[3] Важливість управління проєктами ніколи не може бути переоціненою, адже коли все зроблено правильно, то це допомагає кожній частині бізнесу працювати більш гладко. Наведемо основні переваги управління проєктами:

1. Заощадження часу, грошей та інших ресурсів. З правильним плануванням можна бути певним, що робота виконана вчасно і в межах бюджету. Використовуючи техніки управління проєктами, можна дуже легко відслідковувати подорож проєкту з самого початку і знати наперед, де дедлайни або витрати проєкту можуть бути перевищені для того, щоб

краще розподілити свої ресурси, уникаючи при цьому запізнень та надто великих витрат.

2. Покращення внутрішніх комунікацій всередині компанії. Робота в команді може виявитись складною, проте з ефективно налагодженими процесами планування можна зменшити складність колаборації, збільнити прозорість та забезпечити звітність, навіть працюючи з багатьма командами або підрозділами.

3. Прийняття кращих бізнес-рішень. З якісною візуалізацією процесів проєкту, можна отримати краще розуміння, куди витрачаються ресурси, що потрібно пріорітизувати і найголовніше коли, якщо є ризик зійти з курсу. Якісне управління проєктами означає, що можна передбачити проблеми, ще до того, як вони стануть проблемами, запобігти уповільненню прогресу та приймати більш виважені рішення.

4. Ітерування на успіхах. Управління проєктами допомагає масштабувати високу працездатність та вибудувати кращі практики роботи команди. Використовуючи дані і уроки з попередніх проєктів, можна точно визначити, де команда працює відмінно, а де є місця для покращення.

При реалізації кожної з методик управління проєктами зазвичай не обійтися без певного комплексу технологічного та організаційного інструментарію, тобто, без системи управління проектами. У загальному розумінні це певна сукупність методів, які можуть впливати на об'єкт управління з метою реалізації всіх поставлених завдань. Але найчастіше це поняття використовується в більш вузькому сенсі — як позначення конкретної програми. Зазвичай системи автоматизації управління проєктами містять наступні структурні елементи:

- засоби для календарно-сіткового планування;
- засоби для рішення окремих задач (допроєктний аналіз, розробка бізнес-планів, аналіз ризиків, управління контрактами, часом, бюджетом);

- засоби для організації комунікацій між виконавцями проекту.

У загальному вигляді управління проектами розглянуто у роботах Дж. Джурана[2], Б. Годфрі[2], Дж. Хігні[3], Р. Хосенлопа[6] та П. Робертса[7]. Значний внесок у теорію та практику управління проектами здійснено численними проектними інститутами, серед яких Американський національний інститут стандартів (ANSI), якому належить розробка провідного міжнародного стандарту управління проектами (PMBOK). Дослідженню, розробленню та впровадженню сучасних методологій управління проєктами присвячено багато робіт, серед найбільш ґрунтовних можна виокремити праці С. Макконел[8], Е. Харін[9], Дж. Хайсміт[10] та Дж. Ротман[11]. У вищенаведених наукових роботах висвітлюються професійні проблеми управління проєктами та пропонуються певні шляхи їх вирішення. У більшості з них вказується на необхідність використання ітеративного підходу при управлінні проєктами.

1.2 Аналіз існуючих інформаційних систем для управління проєктами

Управління проєктами є чи не найбільш актуальною проблемою сьогодення, тому існує велика кількість систем для вирішення цієї задачі. Найвідомішими планувальниками проєктів є Jira, Epicflow та Github Project. Для особистого управління завданнями найчастіше використовують Trello та Google Tasks. Нижче наведено опис до кожної із згаданих систем.

Jira - це частина сімейства продуктів, розроблених компанією Atlassian, щоб допомогти командам усіх типів керувати роботою. Спочатку програмне забезпечення Jira було розроблено як засіб відстеження помилок і проблем. Але сьогодні Jira перетворилася на потужний інструмент керування роботою для всіх типів випадків використання, від керування вимогами та тестами до гнучкої розробки програмного забезпечення.[12]

Головною перевагою Jira є широкий функціонал, тобто дане програмне забезпечення можна адаптувати до роботи з проєктами

різноманітної складності. Функціонал Jira також підлягає розширенню за допомогою використання різноманітних плагінів. Jira також має великий інтеграційний потенціал і вона приязно співпрацює з такими застосунками, як Github, Gmail та Slack. Ну і звісно, основним плюсом, або ж навіть головною ідеєю Jira є можливість роботи за методологіями Scrum або Kanban. Така велика кількість функцій Jira спричиняє один недолік - це складний інтерфейс, і через це налаштування конкретних робочих процесів вимагає багато часу. Проте цей мінус легко перекривається вивченням інструменту та постійною практикою.

Epicflow - це програмне забезпечення нового покоління, яке використовує машинне навчання та прогнозну аналітику для забезпечення успіху будь-якого бізнесу. Epicflow надає унікальну гнучкість для боротьби з невизначеністю та обмеженнями, які виникають у процесі управління проєктами.[13]

Застосунок Epicflow має наступні переваги:

- алгоритм Flow. Flow — це алгоритм, який зосереджується на загальному прогресі кожного завдання та проєкту. Цей алгоритм перевіряє загальний статус проєкту, скільки завдань завершено та хто над яким завданням працює. Потім він обчислює та визначає можливі області затримки в системі. Він також може сповістити вас про ці області затримки та повідомити, який тип затримки буде спричинено.
- управління завданнями та проєктами. Одна з головних цілей програми — допомогти в плануванні та розкладі завдань. Вона також дозволяє переглядати майбутні і прострочені завдання або фільтрувати окремі проєкти за компанією та кінцевим терміном, призначеними працівниками та обсягом виконаних робіт.
- візуалізація даних. Система звітності Epicflow дозволяє ефективно аналізувати будь-які набори даних. Будь-який показник або

статистику можна записати та порівняти за допомогою цього інструменту.

- інтеграція з іншими застосунками. У систему можна інтегрувати кілька програмних рішень сторонніх виробників. Можна додати різні платформи керування проєктами, наприклад, Topdesk, Jira та MS Project.

Недоліком Epicflow можна назвати те, що даний інструмент не надає механізми для індивідуального планування, а також те, що він є досить складним у користуванні і вимагає багато часу для налаштування усіх робочих процесів.

Github Projects - це адаптивний, гнучкий інструмент для планування та відстеження роботи на GitHub.[14] Взагалі GitHub полегшує роботу з системою управління версіями Git, а Github Projects дозволяє розпланувати створення проекту та пов’язує пул-реквести, які створюють розробники, із завданнями, які представлено за допомогою методології Kanban.

Плюсами Github Projects можна назвати простоту у використанні, зручний інтерфейс та інтеграцією з GitHub. Проте головним мінусом є те, що цей інструмент підходить лише для планування IT-проєктів.

Trello - це візуальний інструмент, що дає змогу команді керувати різноманітними проєктами й робочими процесами та відстежувати виконання завдань. Trello, як і Jira, було розроблено компанією Atlassian. Туди можна додавати файли, контрольні списки або правила для автоматизації: налаштування Trello допомагають оптимізувати індивідуальну та командну роботу.[15]

Основними перевагами Trello є:

- простий у використанні інтерфейс, який не вимагає багато часу для налаштування.
- управління проєктом та командою. Це, мабуть, найпопулярніше використання Trello, яке дозволяє менеджерам контролювати проєкт

і створювати дошки для конкретних учасників, які містять їхні завдання.

- візуалізація даних. Хоча Trello за замовчуванням дані відображаються в Kanban-дошці, існують додаткові способи налаштування системи за допомогою платної версії.

Проте у застосунку є і недоліки:

- управління невеликими проєктами. Trello чудово підходить, коли, наприклад, кілька користувачів маніпулюють кількома картками щодня чи тижня. Але чим більше завдань на одній дошці, чим більше карток у кожному списку та чим більше користувачів додається в проєкт, тим важче все стає.
- обмежене сховище даних. Trello дозволяє вкладення, але користувачі безкоштовної версії мають лише до 10 Мб на завантаження, що досить мало.

Google Tasks - це програма для керування завданнями, розроблена Google і включена в Google Workspace. Завдання Google, які спочатку були включені в Gmail і календар Google, були запущені як основний продукт із окремою програмою в 2018 році.[16]

Google Tasks надає простий інтерфейс для управління завданнями, має чудові механізми сповіщень, підходить як для індивідуального, так і для колективного планування, є частиною екосистеми Google, і через це співпрацює з великою частиною їхніх сервісів, і також є цілком безкоштовним. Проте через досить обмежений функціонал даний інструмент не підійде для планування великих та складних проєктів.

В таблиці 1.1 можна побачити порівняльний аналіз усіх вищезгаданих застосунків за наступними ознаками:

- Механізми для індивідуального планування
- Механізми колективного планування
- Наявність підсистеми сповіщень

- Зрозумілий інтерфейс
- Декілька способів впорядкування інформації
- Безкоштовність
- Можливість управління декількома проєктами
- Формування звітів
- Автоматичне обчислення пріоритетності завдань
- Візуалізація статистики

Таблиця 1.1 Порівняльний аналіз популярних систем планування

№	Ознака	Jira	Github Projects	Trello	Epicflow	Google Tasks
1	Механізми для індивідуального планування	-	-	+	-	+
2	Механізми колективного планування	+	+	-	+	+
3	Наявність підсистеми сповішень	-	-	-	-	+
4	Зрозумілий інтерфейс	-	+	+	+	+
5	Декілька способів впорядкування інформації	+	+	+	+	-
6	Безкоштовність	-	+/-	-	-	+
7	Можливість управління декількома проєктами	+	+	+	+	-

8	Формування звітів	+	-	-	+	-
9	Автоматичне обчислення пріоритетності завдань	-	-	-	+	-
10	Візуалізація статистики	+	-	+	+	-

Як можна побачити з таблиці 1.1 у кожної з систем є певні недоліки, які все-таки не є приємним доповненням для кінцевих користувачів. Тому створення системи, яка задовольнятиме всім вищенаведеним характеристикам, не є простою задачею, проте вона зробить процес управління проєктами набагато легшим та приємнішим.

1.3 Аналіз основних процесів планування проєктів

Для моделювання основних процесів управління проєктами в нотаціях IDEF0 будемо використовувати онлайн-інструмент для побудови діаграм draw.io.

Розглянемо основні межі обстеження:

1. Територіальні межі – дослідження проводиться онлайн
2. Часові межі – період обстеження кожного робочого дня
3. Функціональні межі – для них визначимо основні функції процесу «Управління проєктом»:

- Ініціювання
- Планування
- Виконання
- Відстеження
- Закриття

Розгянемо контекстну діаграму процесу управління проєктом ЯК Є, яку представлено на рисунку 1.1.

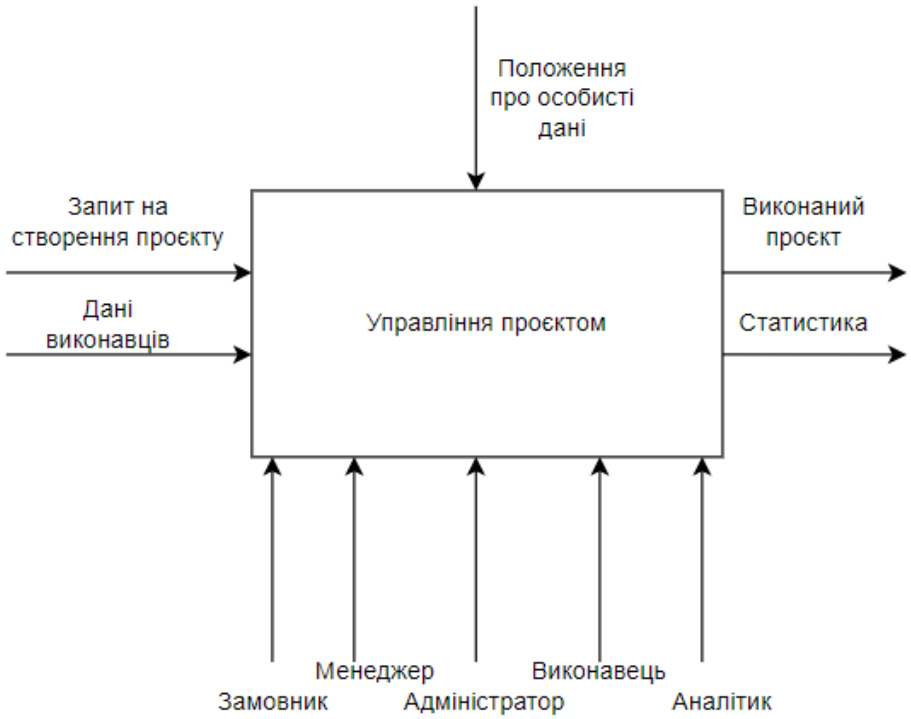

Рисунок 1.1 - Контекстна діаграма ЯК Є процесу «Управління проєктами» в нотації IDEF0

На рисунку 1.2 зображено декомпозицію процесу «Управління проєктом». Вхідними даними процесу є запит на створення проєкту та дані виконавців, а вихідними - виконаний проєкт та статистика. Процес керується положенням про особисті дані та до нього залучені такі особи, як замовник, менеджер, адміністратор, виконавець та аналітик. Бізнес-процес “Управління проєктом” складається з наступних підпроцесів:

1. Ініціювання

   Вхідні дані:

   - Запит на створення проєкту

   Вихідні дані:

   - Схвалені вимоги до проєкту

2. Планування

   Вхідні дані:

   - Схвалені вимоги до проєкту
   - Дані виконавців

Вихідні дані:

- Чіткий план

3. Виконання

Вхідні дані:

- Чіткий план

Вихідні дані:

- Виконані завдання

4. Відстеження

Вхідні дані:

- Виконані завдання

Вихідні дані:

- Звіти

5. Закриття

Вхідні дані:

- Звіти

Вихідні дані:

- Виконаний проєкт
- Статистика

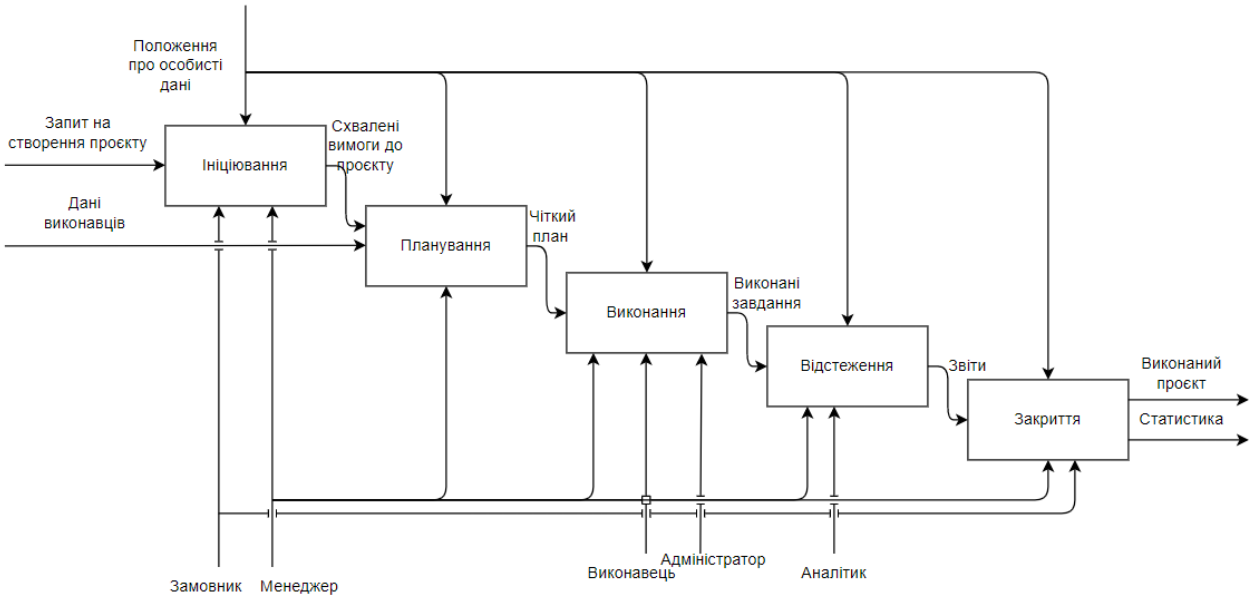

Рисунок 1.2 - Декомпозиція контекстної діаграми ЯК Є процесу «Управління проєктами» в нотації IDEF0

Процес “Ініціювання” позначає початок створення проєкту. Замовник звертається до менеджера з ідеєю для створення власного проєкту. Вони вдвох займаються обговоренням запиту проєкту, на основі цього менеджер описує конкретні вимоги, які потім має схвалити замовник. Даний бізнес-процес зображений на рисунку 1.3.

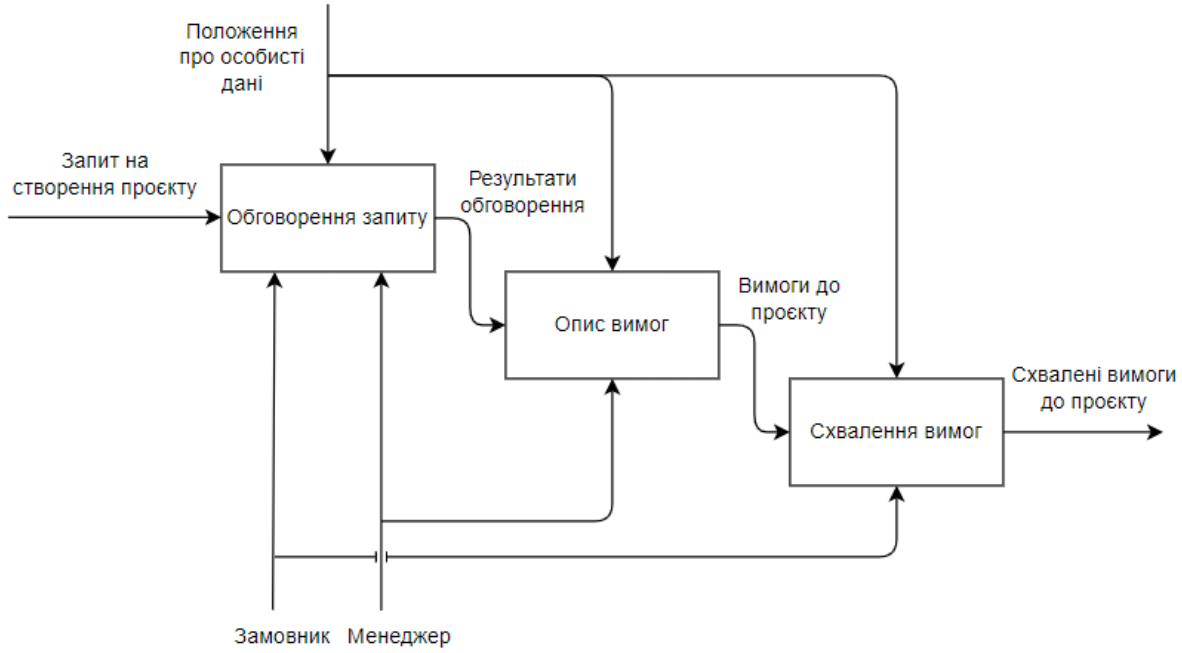

Рисунок 1.3 - Декомпозиція діаграми ЯК Є процесу «Ініціювання» в нотації IDEF0

Процес “Планування”, який представлено на рисунку 1.4, є одним з ключових бізнес-процесів в управлінні проєктами, від якого дуже багато чого залежить. Плануванням займається менеджер, який аналізує вимоги до проєкту і на основі цього визначає його терміни виконання та формує завдання. Потім, використовуючи дані виконавців, менеджер призначає на кожне завдання конкретного виконавця.складається з наступних підпроцесів:

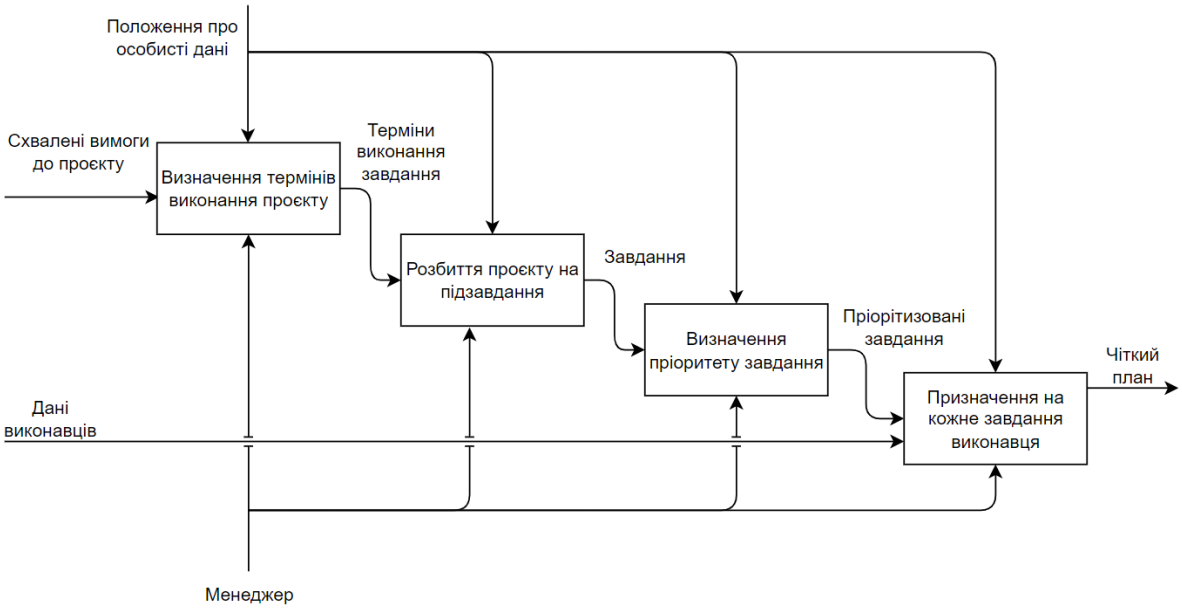

Рисунок 1.4 - Декомпозиція діаграми ЯК Є процесу «Планування» в нотації IDEF0

Наступним етапом є процес “Виконання”, який зображено на рисунку 1.5. До нього відповідно залучений менеджер, адже він зобов’язаний сповістити виконавців про їхні завдання. Далі виконавці працюють над цими завданням, та повідомляють адміністратора про виконане завдення. Адміністратор у свою чергу заносить відмітки про завдання в сховище з даними.

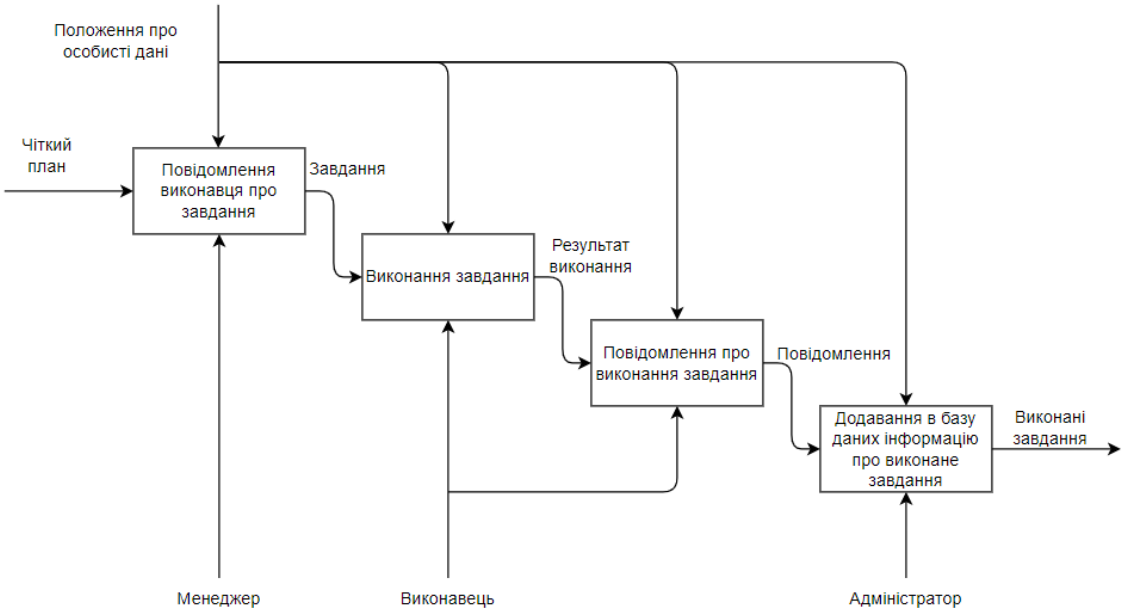

Рисунок 1.5 - Декомпозиція ЯК Є діаграми процесу «Виконання» в нотації IDEF0

Процес “Відстеження” також є частиною процесу “Управління проєктом. Тут менеджер перевіряє виконані завдання на правильність, а далі аналітик займається збиранням даних, і на основі цього робить звітність по проєкту, що і є представлено на рисунку 6.

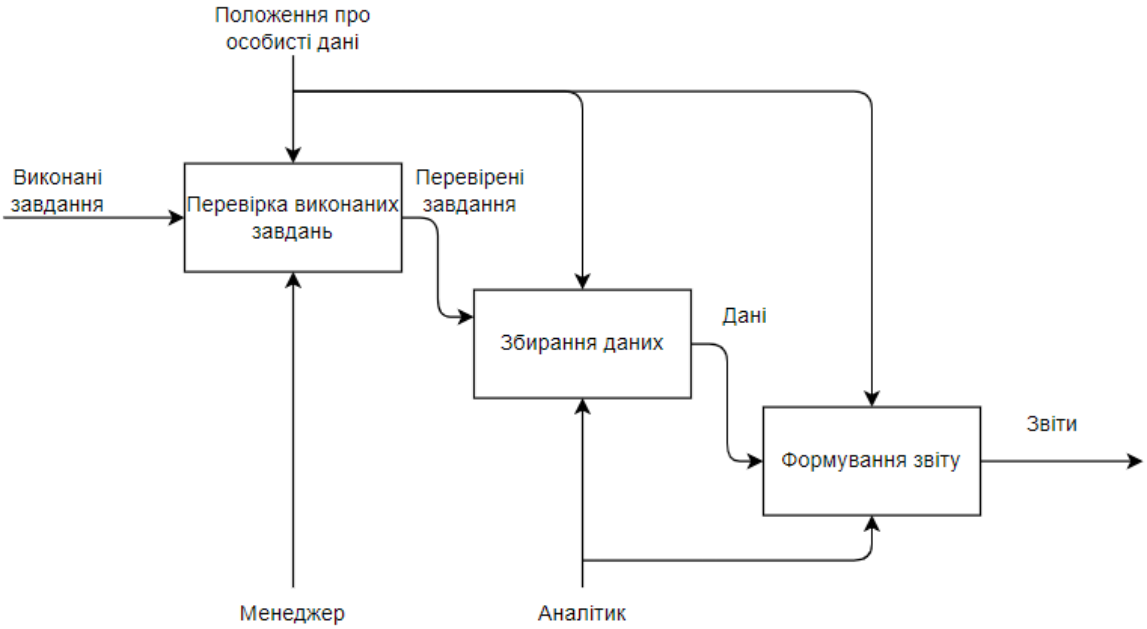

Рисунок 1.6 - Декомпозиція діаграми ЯК Є процесу «Відстеження» в нотації IDEF0

Фінальним бізнес-процесом є “Закриття”, який зображено на рисунку 1.7. До нього залучені менеджер та замовник, які ознайомлюються зі звітами та обговорюють ситуацію, що склалась. Прийняття рішення про закриття проєкту лежить на замовникові, а менеджер в свою чергу має сповістити усіх співробітників про це. Таким чином, проєкт переживає останню фазу - це запинення або закриття.

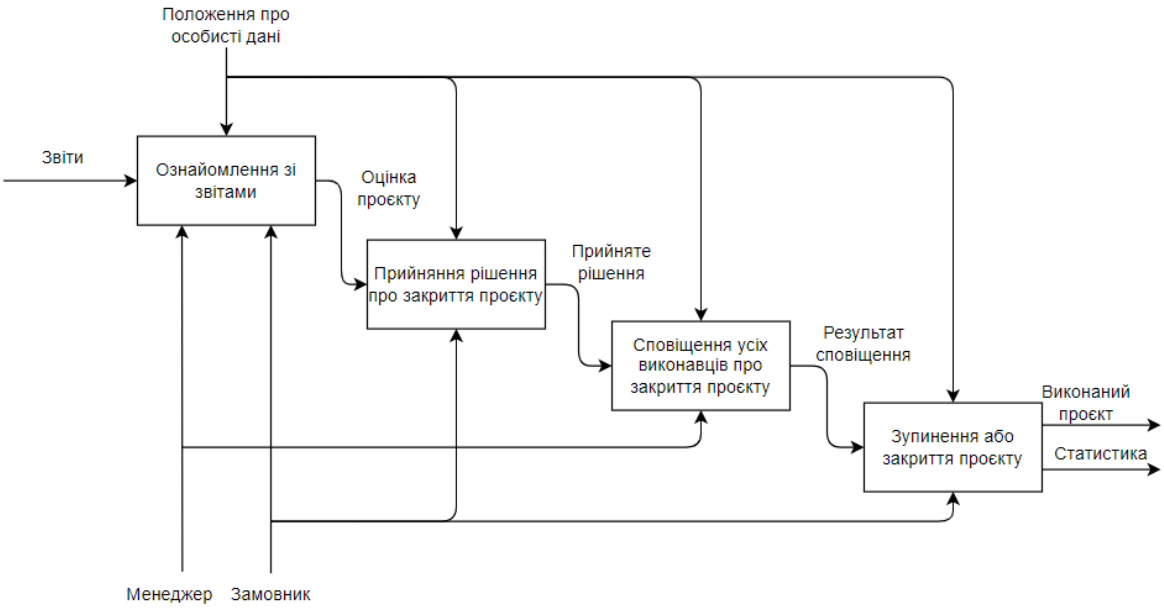

Рисунок 1.7 - Декомпозиція діаграми ЯК Є процесу «Закриття» в нотації IDEF0

Основним недоліком даних діаграм ЯК-Є, як видно з рисунків 1.2-1.7, є велика кількісь зовнішніх об’єктів і всі їхні бізнес-процеси виконуються вручну, що збільшує ймовірність механічних помилок. Такими процесами є повідомлення кожного виконавця про завдання, занесення відміток про виконанні завдання в сховище, збирання даних та формування звітів. Для того, щоб уникнути небажаних помилок та затримок, дані процеси необхідно автоматизувати.

1.4 Постановка задачі на розробку системи для управління проєктами

Застосунок повинен надати можливості для створення, редагування та видалення проєктів та завдань, а також користувачів та груп. Візуалізація даних в системі має бути представлена декількома способами, а також необхідне зручне відображення статистиски проєктів для відстеження прогресу. Усі завдання з проєктів повинні сортуватись за пріоритетністю, яка має обчислюватись автоматично. Також потрібно налаштувати в системі механізми аутентифікації та авторизації, і впровадити різні режими

доступу для користувачів, наприклад, адміністратор, менеджер проєкту та звичайний користувач.

Система управління проєктами повинна відповідати наступним функціональним вимогам:

1. Можливість додавання, редагування та видалення проєктів і завдань.
2. Автоматичне обчислення пріоритетності завдань.
3. Аутентифікація та авторизація користувачів, а також наявність різних ролей, у який буде різний режим доступу в системі: адміністратор та користувач.
4. Представлення завдань різними способами: календар, kanban-дошка, таблиця.
5. Сортування проєктів та завдань за їхнім пріоритетом.
6. Додавання, редагування та видалення міток до завдань.
7. Можливість створення і управління командами користувачів.
8. Наявність системи сповіщень користувачів, яка нагадуватиме користувачам про терміни виконання завдань.
9. Формування звітів по проєктам.
10. Візуалізація статистичних даних для відстеження прогресу.
11. Можливість фільтрації та сортування за різними параметрами.

При реалізації програмного продукту повинні бути дотримані наступні нефункціональні вимоги та обмеження:

1. Легкий у використанні інтерфейс.
2. Кросплатформність та кросбраузерність.
3. Унікальний дизайн, що легко впізнати.
4. Кольорова гама, що повинна відповідати кольорам, зображеним на рисунку 1.8.

Рисунок 1.8 - Кольорова гама застосунку

5. Усі дії в програмі мають виконуватись швидко та плавно.
6. Валідація даних.
7. Клієнт-серверна архітектура застосунку.
8. Наявність документації для серверної частини.

1.5 Висновки до аналітичного розділу

В аналітичному розділі було визначено сучасні проблеми управління проєктами, а також було описано значення термінів проєкт, управління проєктами та системи управління проєктами. На сьогоднішній момент існує багато інформаційних систем, які вирішують проблему управління проєктами, тому було проведено дослідження програмного ринку на наявність систем, які займаються управлінням проєктами, та проаналізовано їхні позитивні і негативні сторони. Для кращого розуміння предметної області було проведено моделювання процесу “Управління проєктом” в нотації IDEF0 та визначено основі недоліки діаграми ЯК-Є.

В результаті вищезазначених пунктів було сформовано постановку задачі на розробку та проведено функціональний та нефункціональний аналіз для кращого розуміння вимог до програмного застосунку.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

2.1 Розробка архітектури системи управління проєктами

2.1.1 Функціональний аналіз управління проєктами

Процес управління проєктами складається з наступних підпроцесів:

- робота з проєктами;
- робота із завданнями;
- управління ресурсами;
- робота з документами;
- робота із статистикою.

Кожен із вищезазначених процесів можна розділити ще на декілька функцій:

1. Робота із проєктами:
    - створення проєктів;
    - редагування проєктів;
    - визначення пріоритету проєкту;
    - видалення проєкту.
2. Робота із завданнями:
    - створення завдання;
    - редагування;
    - призначення користувача для виконання завдання;
    - визначення пріоритету;
    - зміна статусу завдання;
    - видалення завдання.
3. Управління ресурсами:
    - управління групами:
        - створення груп;
        - редагування груп;
        - видалення груп.

- управління користувачами:
    - створення користувачів;
    - редагування користувачів;
    - авторизація та аутентифікація;
    - видалення користувачів.

4. Робота із документами:
    - формування звітів;
    - завантаження звітів.
5. Робота із статистикою:
    - збирання даних;
    - візуалізація за допомогою графіків.

Функціональний аналіз управління проєктами представлений на карті процесів, яку було побудовано за допомогою програмного забезпечення ARIS Express (рис. 2.1).

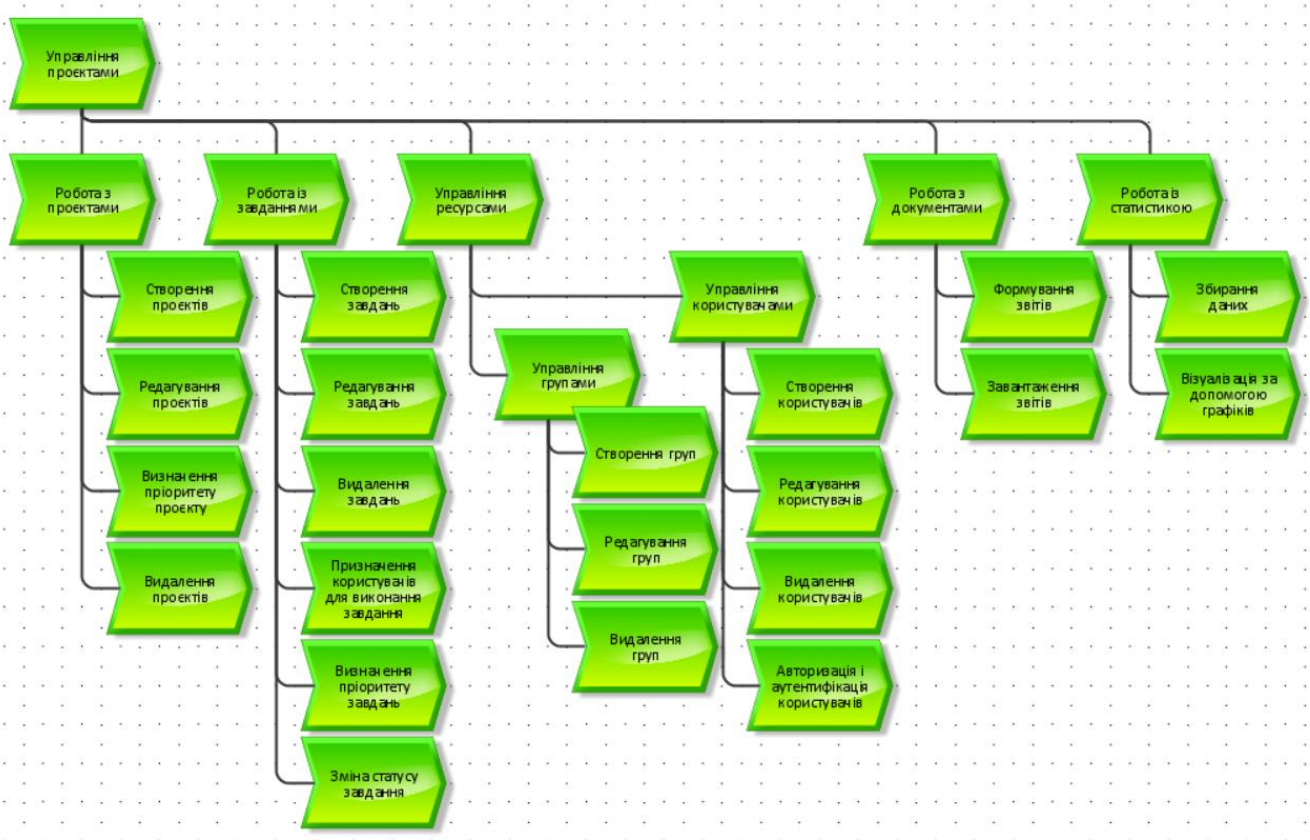

Рисунок 2.1 - Карта процесів для системи управління проєктів

2.1.2 Аналіз автоматизованого процесу управління проєктами

У попередньому розділі було розглянуто діаграми процесу управління проєктами та визначено основні недоліки такого підходу. Тепер розглянемо контекстну діаграму процесу управління проєктом ЯК БУДЕ. На рисунку 2.2 можна побачити, що ролі адміністратора та аналітика відсутні, а їхнє місце займає система.

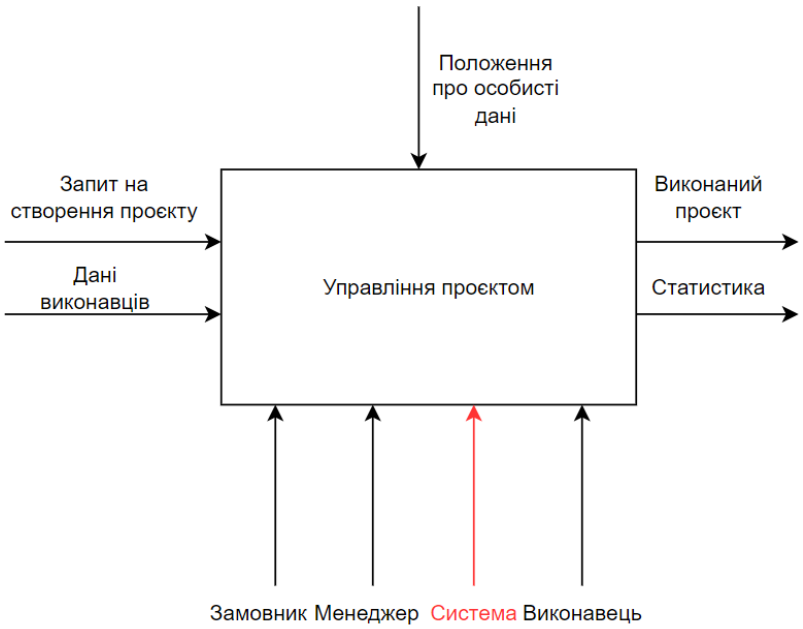

Рисунок 2.2 - Контекстна діаграма ЯК БУДЕ процесу «Управління проєктами» в нотації IDEF0

Декомпозицію контекстної діграми зображено на рисунку 2.3. Автоматизована система буде залучена до кожного етапу управління проєктами.

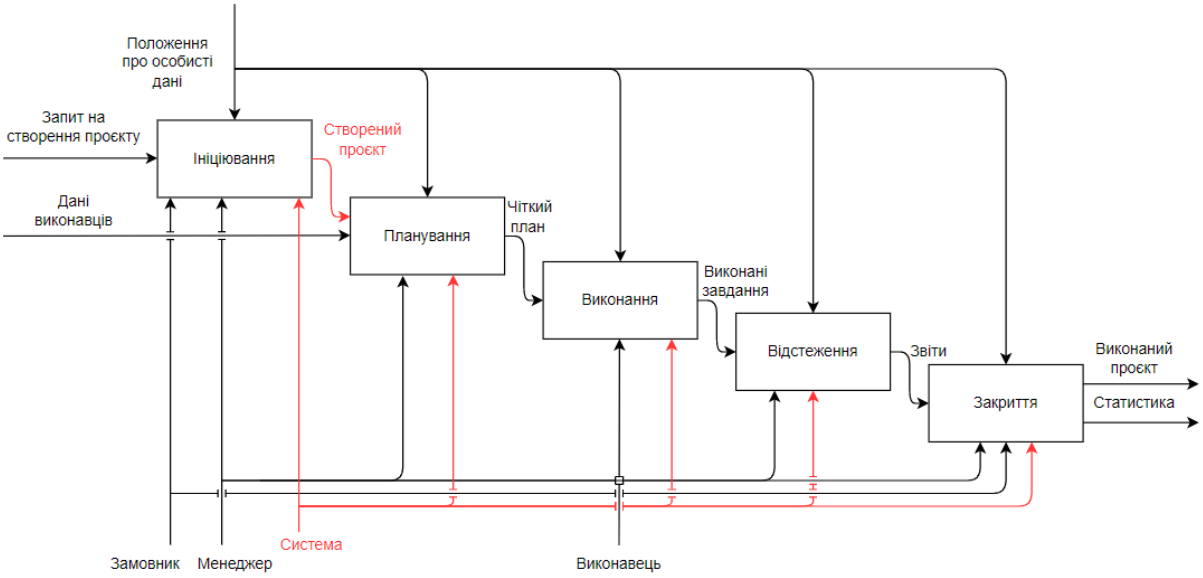

Рисунок 2.3 - Декомпозиція контекстної діаграми ЯК БУДЕ процесу «Управління проєктами» в нотації IDEF0

До процесу “Ініціювання” було додано ще один бізнес-процес(рис. 2.4), а саме створення проєкту, яким займатиметься менеджер, і звісно ж буде залучена сама система.

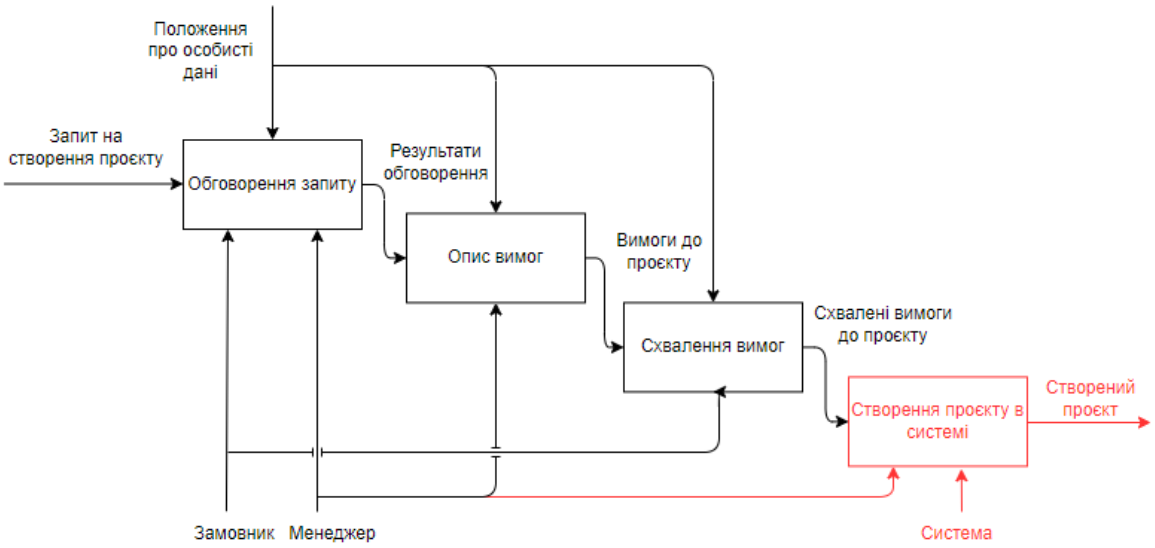

Рисунок 2.4 - Декомпозиція діаграми ЯК БУДЕ процесу «Ініціювання» в нотації IDEF0

Бізнес-процес “Планування”, який зображено на рисунку 2.5, також зміниться на краще з додаванням системи. Менеджеру більше не потрібно

буде самому обчислювати пріоритет якого-небудь завдання, цим займатиметься система.

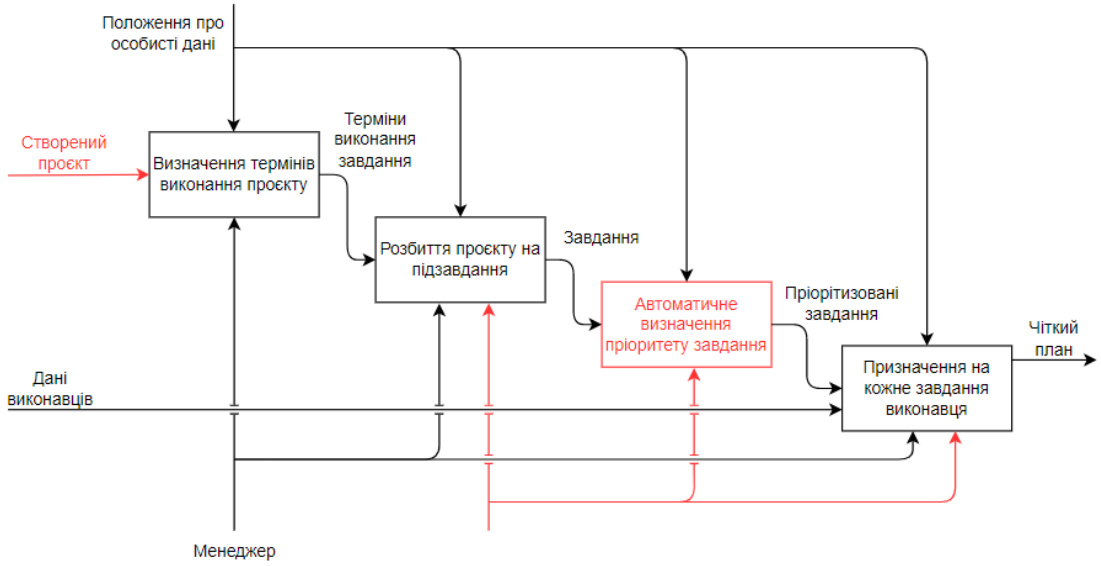

Рисунок 2.5 - Декомпозиція діаграми ЯК БУДЕ процесу «Планування» в нотації IDEF0

На рисунку 2.6 показано, як система витіснила ролі менеджера та адміністратора для бізнес-процесу “Виконання”.

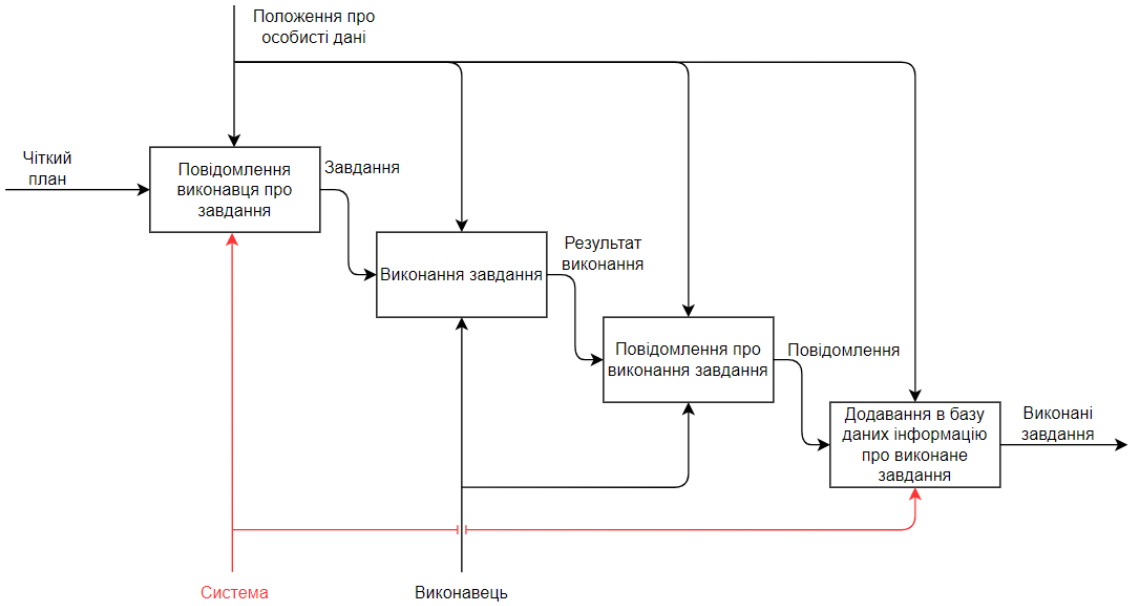

Рисунок 2.6 - Декомпозиція діаграми ЯК БУДЕ процесу «Виконання» в нотації IDEF0

Наступним етапом є бізнес-процес “Відстеження”. На рисунку 2.7 видно, що з системою не потрібно буде займатись збиранням даних та ручним формуванням звітів, оскільки система буде робити це автоматично.

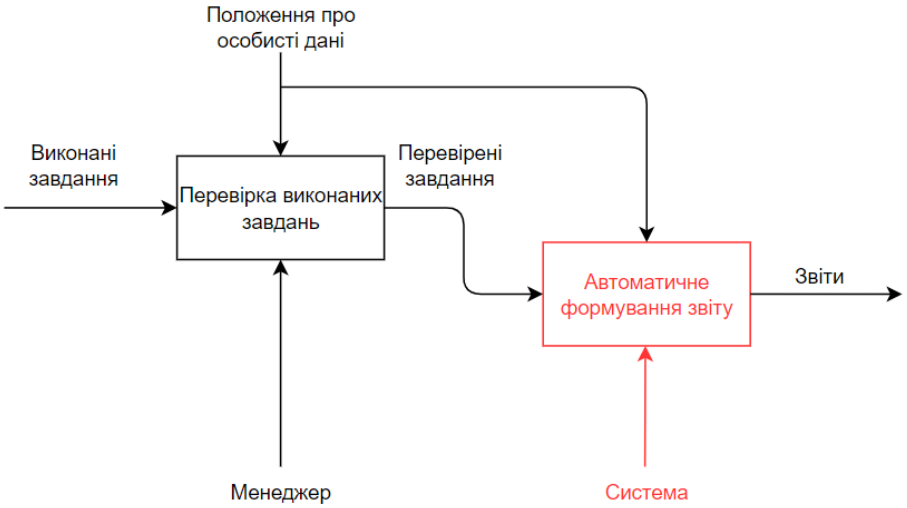

Рисунок 2.7 - Декомпозиція діаграми ЯК БУДЕ процесу «Відстеження» в нотації IDEF0

Завершальний етап “Закриття” також не відбуватиметься без втручання системи. В даному випадку вона займатиметься сповіщенням виконавців про завершення проєкту та буде причетна до його зупинення або закриття (рис. 2.8).

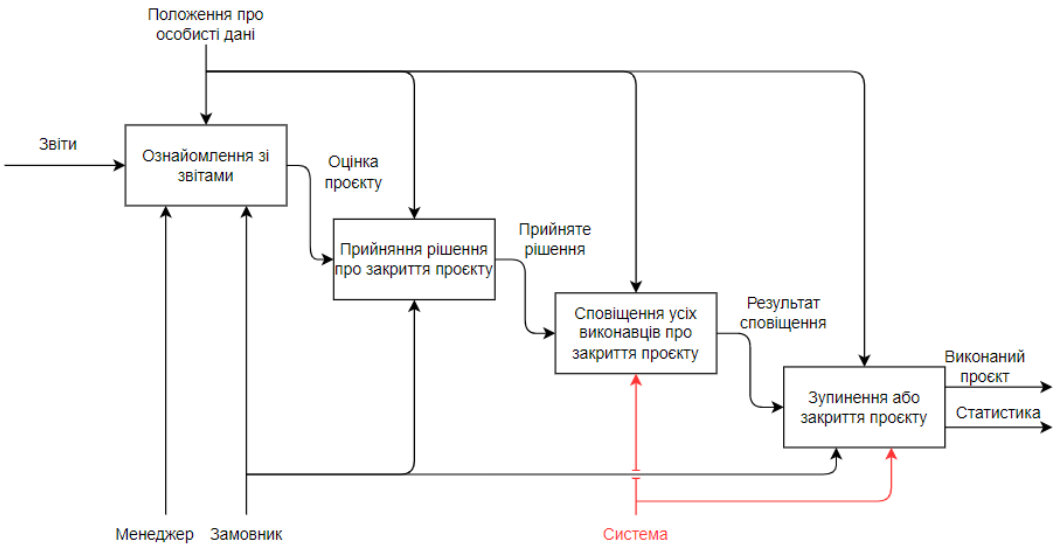

Рисунок 2.8 - Декомпозиція діаграми ЯК БУДЕ процесу «Закриття» в нотації IDEF0

Проаналізувавши вищенаведені діаграми ЯК БУДЕ можна зробити висновок, що система значно покращить бізнес-процес управління проєктами. Вона дозволить уникнути великої кількості помилок, виконуючи значну частину процесів автоматизовано. Система також надасть можливість усім сутностям більш уважно зосередитись на дійсно важливій частині роботи, забезпечивши їх зручними автоматизованими механізмами роботи з даними та гарною візуалізацією цих даних.

2.1.3 Архітектура системи управління проєктами

Для системи управління проєктами було обрано клієнт-серверну архітектуру веб-застосунку, діаграму якої зображено на рисунку 2.9.

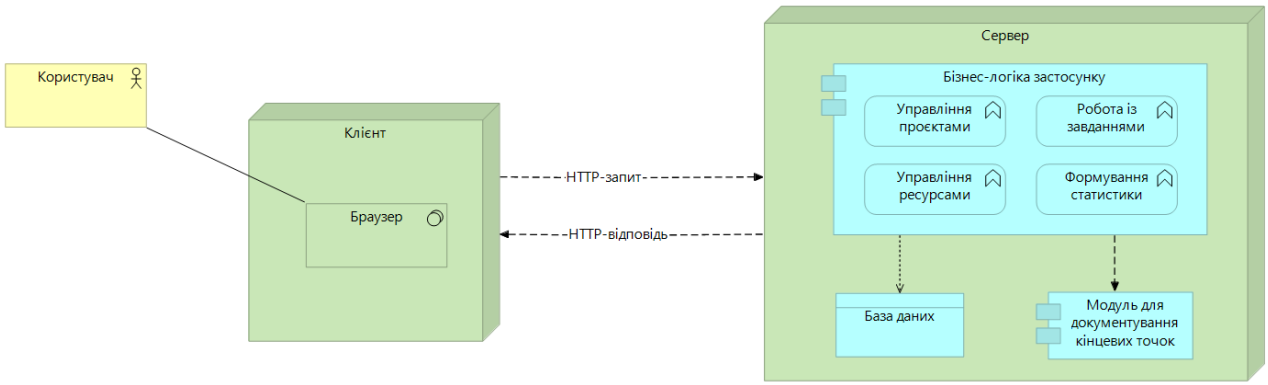

Рисунок 2.9 - Діаграма архітектури прикладної програми для планування в середовищі ArchiMate

Як видно з діаграми (рис. 2.9), кінцевий користувач взаємодіє лише з клієнтською частиною застосунку: вводить дані, натискає на кнопки і так далі. Клієнт або ж frontend-частина відслідковує дії користувача, і відправляє HTTP-запит на сервер або backend-частину з метою отримання потрібних даних, які потрібно представити користувачу. Серверна частина застосунку містить бізнес-логіку застосунку, яка описує правила обробки бізнес-даних і взаємодію з усіма іншими модулями. У випадку розробленого застосунку саме тут описоно операції для роботи з проєктами, завданнями, ресурсами - групами та користувачами, а також формуванням

статистистики. Після того, як усі необхідні операції над даними було здійснено, backend-частина формує HTTP-відповідь для клієнта, ці дані відправляються на frontend, який вже і представляє їх у зручному для користувача вигляді. Варто зазначити, що усі дані, які є вхідними або вихідними для клієнтської і серверної частини, а також для модулю прогнозування, представляються у форматі JSON.

На сервері також присутній модуль для документування кінцевих точок, який зручно представляє усі приклади запитів і відповідей, які необхідні для взаємодії клієнта і сервера.

2.2 Інформаційне забезпечення системи управління проєктами

Спочатку для визначення основних сутностей, що формуватимуть інформаційне забезпечення системи, було побудовано діаграму Чена, яку зображено на рисунку 2.10.

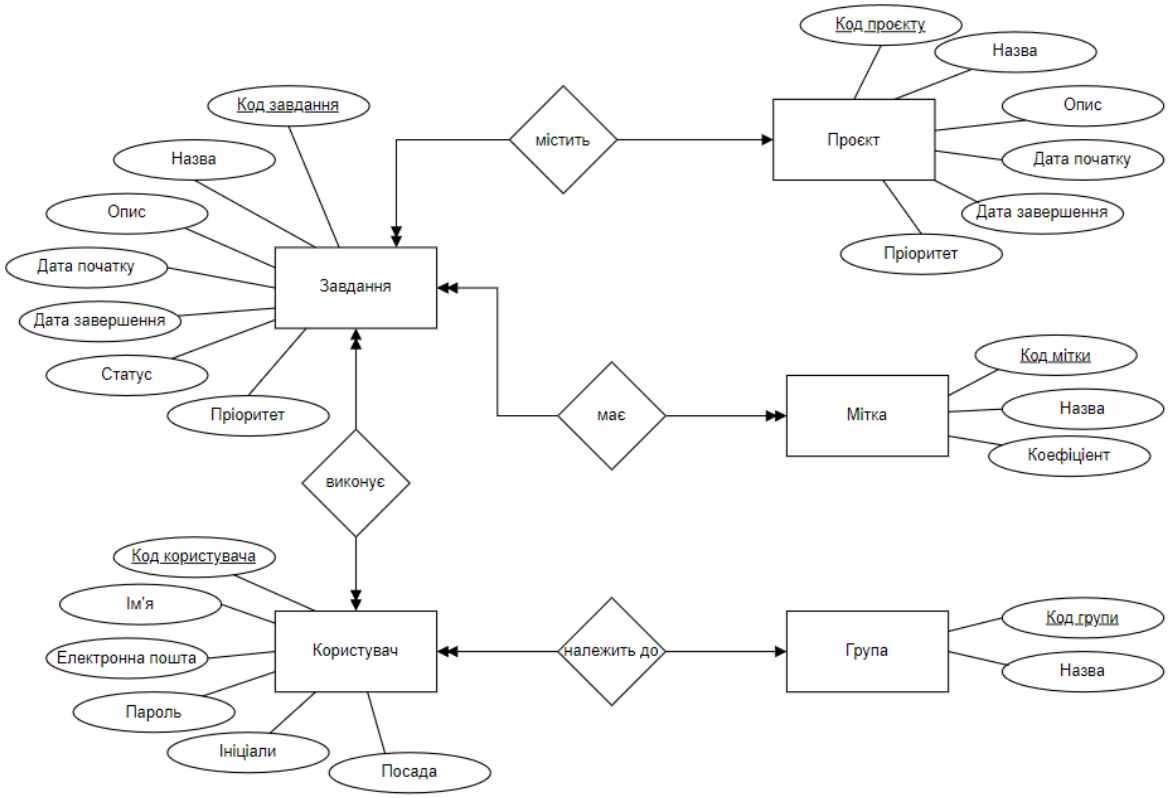

Рисунок 2.10 - Діаграма Чена

Основними сутностяєми є Завдання, Проєкт, Користувач, Група та Мітка. Опис даних сутностей та зв’язків між ними наведено в таблицях 2.1-2.2.

Таблиця 2.1 - Опис сутностей інфологічної моделі

№ п/п	Назва атрибуту	Ідентифікатор	Характеристики атрибуту		
			Тип значення (довжина поля)	Обмеження на довжину/ значення	Признак ключового атрибуту
Сутність «Проєкт»					
1	Код проєкту	Id	Ціле	1..n	PK
2	Назва	Name	Символьний	1…100	-
3	Опис	Description	Символьний	1…10000	-
4	Пріоритет	Priority	Дійсне	0..n	-
5	Дата початку	StartDate	Дата	10 символів	-
6	Дата завершення	EndDate	Дата	10 символів	-
Сутність «Завдання»					
1	Код завдання	Id	Ціле	1…n	PK
2	Назва	Name	Символьний	1…200	-

3	Опис	Description	Символьний	1…10000	-
4	Дата початку	StartDate	Дата	10 символів	-
5	Дата завершення	EndDate	Дата	10 символів	-
6	Статус	Status	Ціле	0..2	-
7	Пріоритет	Priority	Дійсне	0..n	-
8	Код проєкту	ProjectId	Ціле	1..n	FK
Сутність «Користувач»					
1	Код користувача	Id	Ціле	1…n	PK
2	Ім’я	Name	Символьний	1...100	-
3	Електронна пошта	Email	Символьний	1..100	-
4	Пароль	Password	Символьний	1…60	-
5	Код групи	GroupId	Ціле	1..n	FK
6	Ініціали	Initials	Символьний	1..3	-
7	Посада	Title	Символьний	1..100	-
Сутність «Група»					

1	Код групи	Id	Ціле	1…n	PK
2	Назва	Name	Символьний	1…100	-
Сутність «Мітка»					
1	Код мітки	Id	Ціле	1..n	PK
2	Назва	Name	Символьний	1..20	-
3	Коефіцієнт	Coefficient	Дійсне	0..n	-

Таблиця 2.2 - Опис зв’язків між сутностями інфологічної моделі

№ п/п	Сутності, що утворюють зв’язок	Тип зв’язку	Пояснення
1	Проєкт – Завдання	1..1:1..n	Проєкти складаються з одного або декількох завдань, проте завдання може належати лише до одного проєкту.
2	Завдання – Користувач	1..n:1..n	Кожен користувач може займатись декількома завданнями, і над кожним завданням може працювати декілька користувачів
3	Користувач – Група	1..n:1..1	Кожний користувач може належати лише до 1 групи, а група може містити більше ніж 1 користувача
4	Завдання – Мітка	1..n:1..n	Кожне завдання може мати декілька міток, а у однієї мітки може бути багато завдань

Оскільки на рисунку 2.10 можна побачити декілька зв’язків багато до багатьох, на логічній моделі бази даних їх було прибрано, як представлено на рисунку 2.11. Щоб вирішити зв’язок багато до багатьох між таблицями Користувач (User) - Завдання (Issue), було введено нову таблицю Призначення (Assignment), яка буде містити два поля: код користувача та код завдання. За таким же принципом було прибрано зв’язок багато до багатьох для таблиць Завдання(Issue) та Мітка(Tag), та введено таблицю IssueTag.

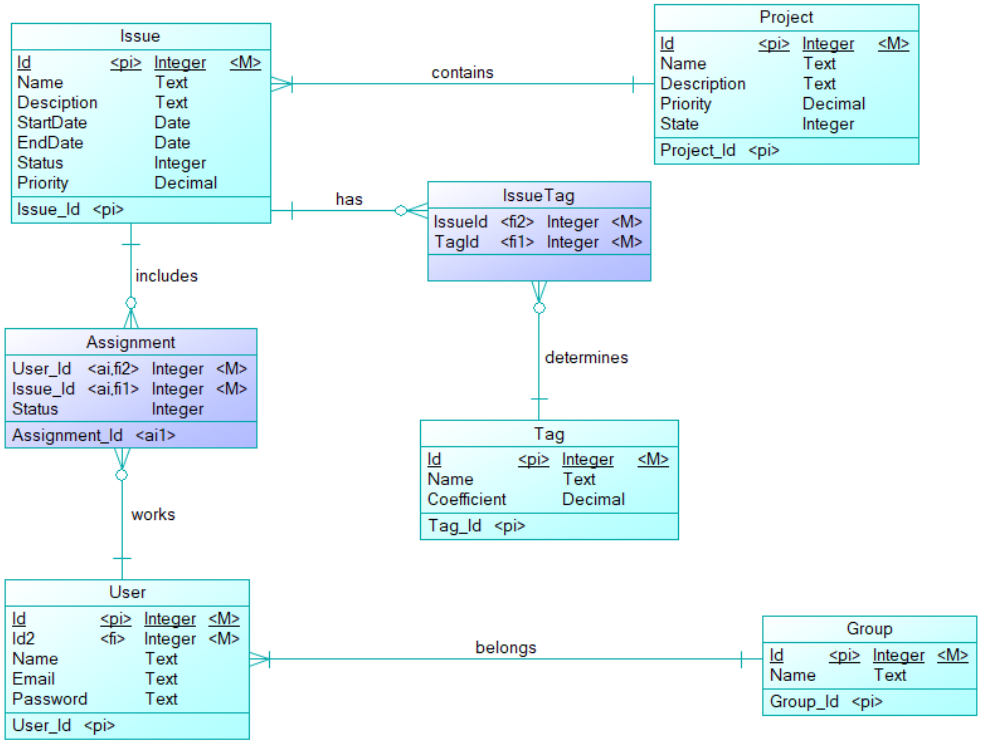

Рисунок 2.11 - Логічна модель бази даних

Також за допомогою програмного забезпечення Power Designer було побудовано фізичну модель бази даних (рис. 2.12).

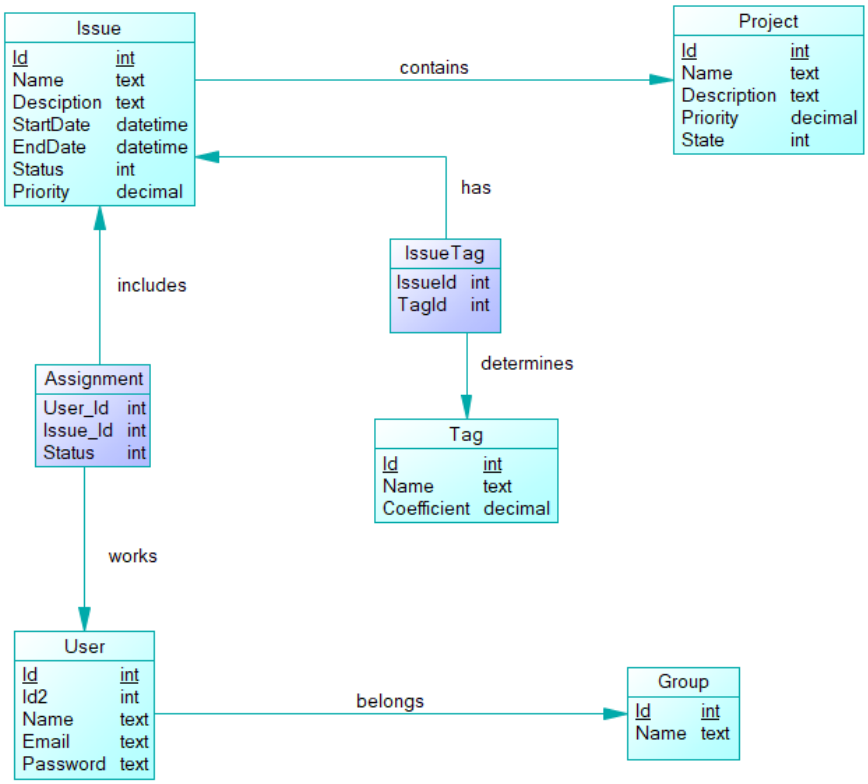

Рисунок 2.12 - Фізична модель бази даних

2.3 Висновки стосовно архітектури проєкту

В розділі розробки архітектури системи управління проєктами було описано функціональний аналіз управління проєктами за допомогою карти процесів в середовищі ARIS Express. Наступним етапом був аналіз автоматизованого процесу управління проєктами на основі діаграм ЯК-Є з аналітичного розділу та було побудовано діаграми ЯК-БУДЕ в нотації IDEF0. В результаті проведеннч попередніх кроків було обрано клієнт-серверну архітектуру веб-застосунку та побудовано її діаграму в застосунку ArchiMate.

Завершальним етапом було проведення аналізу інформаційного забезпечення системи управління проєктами. Для цього було визначено та описано основні сутності в системі, а також сформовано інфологічну модель в нотації Чена. Далі було побудовано концептуальну, логічну та фізичну моделі бази даних в середовищі Power Designer, а також усунено зв’язки багато до багатьох між сутностями.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

3.1 Обґрунтування вибору інструментальних засобів для програмної реалізації системи управління проєктами

Для створення системи управління проєктами було обрано платформу .NET, яка надає зручні механізми опису веб-додатків. Взагалі .NET - це безкоштовний, кросплатформний і відкритий ресурс, розроблений компанією Microsoft, для створення різних типів застосунків. При роботі з даною платформою використовувалась мова C#, проте .NET надає можливості роботи і з іншими мовами, наприклад F# та Visual Basic [17].

C# - це сучасна об’єктно-орієнтована мова програмування зі статичною типізацією, що відноситься до широко відомого сімейства мов C. Перевагою C# є те, що він вважається високорівневою мовою програмування із легким для розуміння синтаксисом, або якщо говорити іншими словами він є високорівневою абстракцією над машинним кодом, проте C# також надає і прямі способи доступу до пам’яті з використанням покажчиків. Важливим плюсом використання C# є те, що розробникові не потрібно займатись звільненням пам’яті, адже для цього в платформу .NET вбудовано збирач сміття, який відслідковує об’єкти, що вже не використовуються та звільняє пам’ять.

Оскільки розробка цілої системи з початку лише при використанні мови C# є складним і трудомістким процесом, платформа .NET містить в собі величезну кількість бібліотек для того, щоб максимально полегшити його. При реалізації програмного застосунку для планування було використано наступні фреймфорки та бібліотеки: ASP.NET, Entity Framework Core, Blazor.

ASP.NET - це фреймвок для створення веб-застосунків та веб-сервісів з використанням мови C# та платформи .NET, що підтримується на різних операційних системах. Фактично, ASP.NET - це розширення .NET

інструментами та бібліотеками спеціально для створення веб-додатків. ASP.NET надає зручний синтаксис шаблонів веб-сторінок, який називається Razor, бібліотеки для загальновідомих патернів, таких як MVC та WebAPI, систему аутентифікації та зручні розширення для редагування коду. Серверна частина системи для планування була написана за допомогою ASP.NET та шаблону додатку WebAPI.

Blazor є частиною ASP.NET та призначений для створення клієнтської частини в веб-застосунку. Його головною особливістю є те, що він дозволяє використовувати при розробці користувацького інтерфейсу C# замість усім звичного JavaScript. Застосунки написані з допомогою Blazor базуються на використанні компонентів. Компонентом в Blazor вважається елемент інтерфейсу, наприклад сторінка або ж форма для введення даних. За допомогою даної технології і було описано клієнтську частину застосунку.

Entity Framework Core представляє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Порівняно з традиційними методами засобами доступу до даних, Entity Framework Core являє собою більш високий рівень абстракції, який незалежний від самої бази даних і дозволяє працювати з даними, незважаючи на тип сховища. Якщо на фізичному рівні проводяться операції з таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який називається Entity Framework Core, вже відбувається взаємодія з об’єктами. Доступ до даних здійснювався лише з використанням Entity Framework Core в системі для управління завданнями.

3.2 Структура програмного забезпечення

На рисунку 3.1 представлено структуру клієнської частини програмного забезпечення.

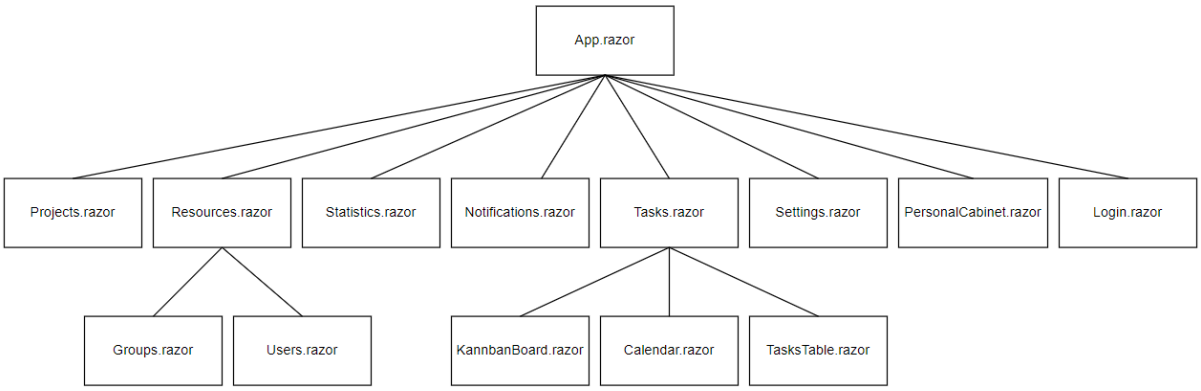

Рисунок 3.1 - Структура клієнської частини програмного забезпечення

В таблиці 3.1 описано усі програмні модулі, які було визначено на рисунку 3.1 вище.

Таблиця 3.1 - Специфікація програмних модулів

Модуль	Опис
App.razor	Початковий модуль, який усі інші сторінки і компоненти. Вхідна інформація: - Вихідна інформація: вміст сайту.
Projects.razor	Модуль для відображення проєктів, а також для різноманітних операцій над ними. Вхідна інформація: токен користувача. Вихідна інформація: список проєктів.
Resources.razor	Модуль для відображення вкладок з користувачами та групами. Вхідна інформація: токен користувача. Вихідна інформація: групи та користувачі, доступні для авторизованого клієнта.

Groups.razor	Модуль для відображення груп, а також для різноманітних операцій над ними. Вхідна інформація: токен користувача. Вихідна інформація: список груп.
Users.razor	Модуль для відображення: користувачі, а також для різноманітних операцій над ними. Вхідна інформація: токен користувача. Вихідна інформація: список користувачів.
Tasks.razor	Модуль для відображення завдань і для різноманітних операцій над ними. Надає мехінізми навігації між різними способами візуалізації завдань. Вхідна інформація: токен користувача. Вихідна інформація: вкладки із завданнями.
KanbanBoard.razor	Модуль для візуалізації завдань за методологією kanban. Вхідна інформація: токен користувача. Вихідна інформація: kanban-дошка із завданнями.
Calendar.razor	Модуль для візуалізації завдань за допомогою календаря. Вхідна інформація: токен користувача. Вихідна інформація: календар із завданнями.
TasksTable.razor	Модуль для візуалізації завдань за допомогою таблиці. Вхідна інформація: токен користувача.

	Вихідна інформація: таблиця із завданнями.
Statistics.razor	Модуль для візуалізації статистики ща проєктами та завданнями. Вхідна інформація: токен користувача. Вихідна інформація: статистичні дані.
Settings.razor	Модуль для редагування налаштувань застосунку. Вхідна інформація: токен користувача. Вихідна інформація: налаштування.
Notifications.razor	Модуль для перегляду сповіщень про завдання. Вхідна інформація: токен користувача. Вихідна інформація: сповіщення про нові завдання та дедлайни.
Login.razor	Модуль для входу в систему. Вхідна інформація: - Вихідна інформація: доступ до вмісту сайту.
PersonalAccount.razor	Модуль для перегляду та редагування особистої інформації користувача. Вхідна інформація: токен користувача. Вихідна інформація: персональні дані користувача.

Серверна частина застосунку зображена у вигляді діаграми класів на рисунку 3.2.

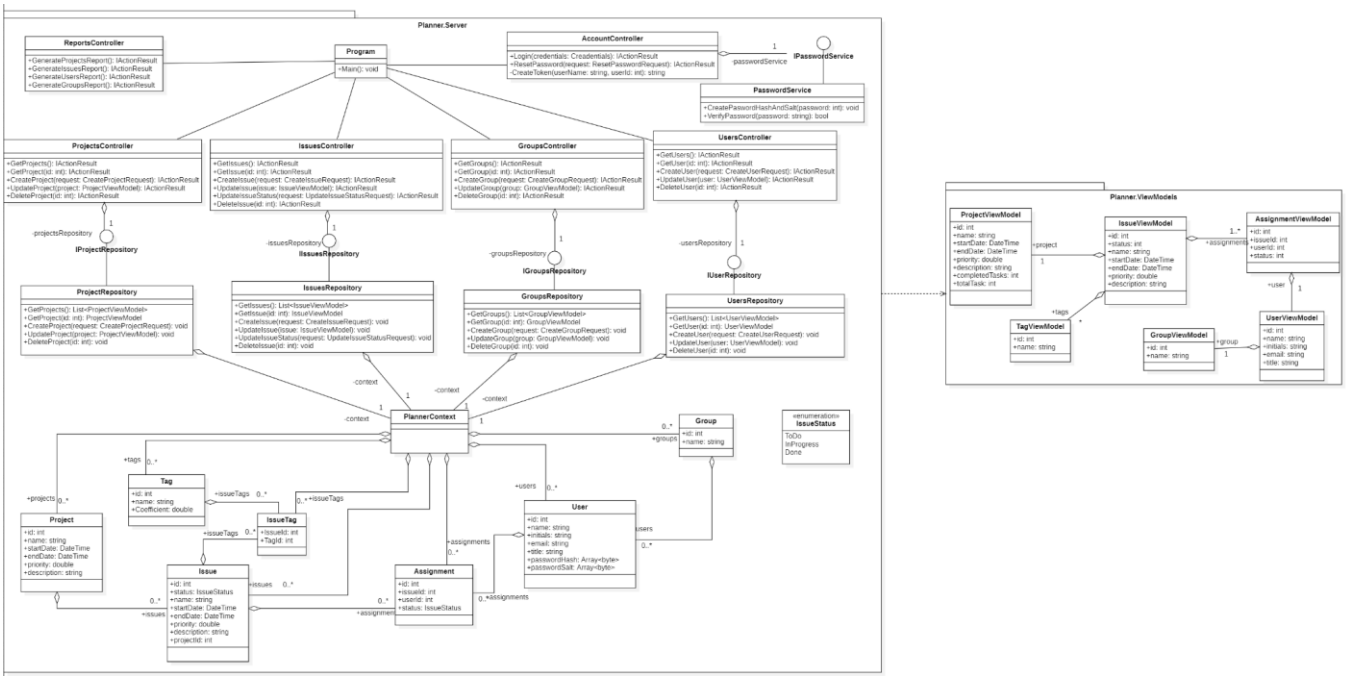

Рисунок 3.2 - Діаграма класів серверної частини застосунку

Опис кожного компоненту з діаграми наведено нижче:

Клас Program - головний клас, який запускає увесь застосунок, містить єдиний метод Main().

Клас ProjectsController - клас-контроллер, який формує відповіді, що стосуються операцій над проєктами, та відправляє їх на клієнтську частину.

Клас IssuesController - клас-контроллер, який формує відповіді, що стосуються операцій над завданнями, та відправляє їх на клієнтську частину.

Клас GroupsController - клас-контроллер, який формує відповіді, що стосуються операцій над групами, та відправляє їх на клієнтську частину.

Клас UsersController - клас-контроллер, який формує відповіді, що стосуються операцій над користувачами, та відправляє їх на клієнтську частину.

Клас AccountController - клас-контроллер, який обробляє запити користувача, що стосуються аутентифікації.

Клас ReportsController - клас-контроллер, який формує звіти та заванажує їх в файли.

Інтефейс IProjectRepository - інтерфейс, який описує основі операції над проєктами.

Клас ProjectRepository - реалізація інтерфейсу IProjectRepository, який витягує проєкти з класу PlannerContext і виконує над ними бажані операції.

Інтерфейс IIssuesRepository - інтерфейс, який описує основі операції над завданнями та мітками.

Клас IssuesRepository - реалізація інтерфейсу IIssuesRepository, даний клас виконує над завданнями, ассайнментами і мітками потрібні операції.

Інтерфейс IGroupsRepository - інтерфейс, який описує основі операції над групами.

Клас GroupsRepository - реалізація інтерфейсу IGroupsRepository , який витягує групи з класу PlannerContext і виконує над ними бажані операції.

Інтерфейс IUserRepository - інтерфейс, який описує основі операції над користувачами.

Клас UsersRepository - реалізація інтерфейсу IProjectRepository, даний клас проводить необхідні операції з користувачами.

Інтерфейс IPasswordService - інтерфейс, що дозволяє створити хеш паролю, та здійснити перевірку паролю.

Клас PasswordService - реалізація класу IPasswordService для роботи з паролями.

Клас PlannerContext - клас, що надає доступ до даних з бази даних та зручне їх представлення.

Клас Project - клас, що відображає сутність “Проект” в базі даних.

Клас Issue - клас, що відображає сутність “Завдання” в базі даних.

Клас Assignment - клас, який дозволяє вирішити зв’язок багато до багатьох між завданнями та користувачами.

Клас Tag - клас, що відображає сутність “Мітка” в базі даних.

Клас IssueTag - клас, який дозволяє вирішити зв’язок багато до багатьох між завданнями і мітками.

Клас User - клас, що відображає сутність “Користувач” в базі даних.

Клас Group - клас, що відображає сутність “Група” в базі даних.

Перелічення IssueStatus - перелічення, що містить три стани завдання: ToDo, InProgress, Done.

Клас ProjectViewModel - клас, для обміну даними про проєкти між клієнською та серверною частинами.

Клас IssueViewModel - клас, для обміну інформації про завдання між клієнською та серверною частинами.

Клас AssignmentViewModel - клас, для обміну даними про ассайнменти між клієнською та серверною частинами.

Клас TagViewModel - клас, для обміну інформації про мітки між клієнською та серверною частинами.

Клас UserViewModel - клас, для предствлення користувачів клієнській частині.

Клас GroupViewModel - клас, для обміну даними про групи між клієнською та серверною частинами.

3.3 Керівництво користувача для системи управління проєктами

При першому відкритті застосунку користувачу буде запропоновано ввести свій логін та пароль і після успішної авторизації, йому буде надано доступ до системи.

Сторінка із проєктами, яка забражена на рисунку 3.3 надає можливості для операцій створення, редагування, перегляду та видалення проєктів. Тут можна також відсортувати та відфільтрувати проєкти за бажаними параметрами. Для отримання звітів по проєктам достатньо натиснути на кнопку завантаження, яка знаходиться у верхньому правому куточку.

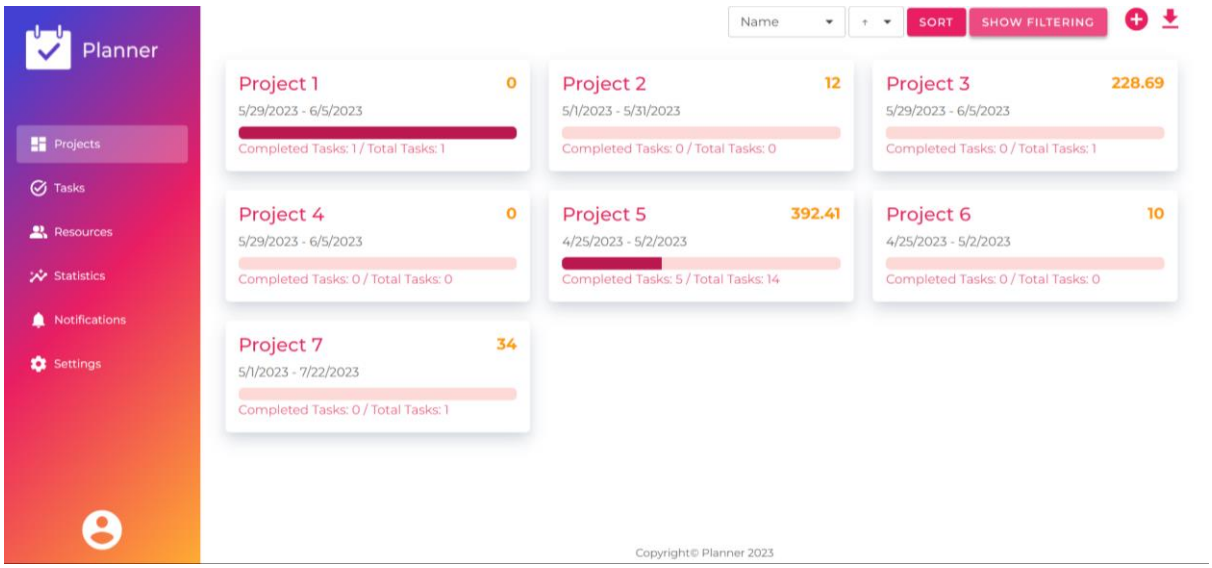

Рисунок 3.3 - Сторінка з проєктами

Для того, щоб створити новий проєкт достатньо натиснути на кнопку “+” у правому верхньому куточку і на екрані з’явиться форма створення проєкту зображена на рисунку 3.4.

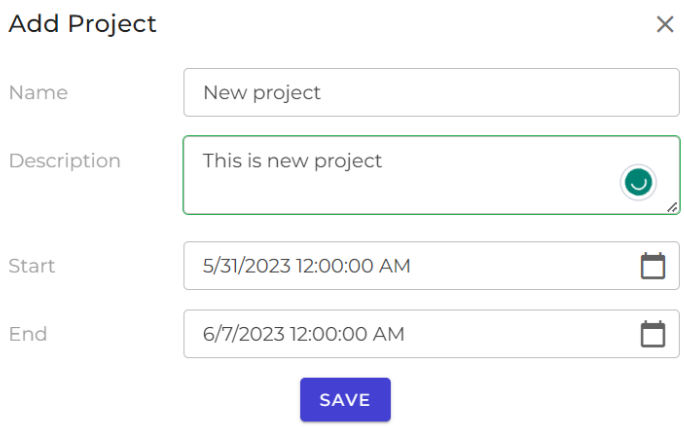

Рисунок 3.4 - Форма створення проєкту

Після натискання на кнопку “Save” проєкт буде збережено та відображено у загальному списку (рис. 3.5).

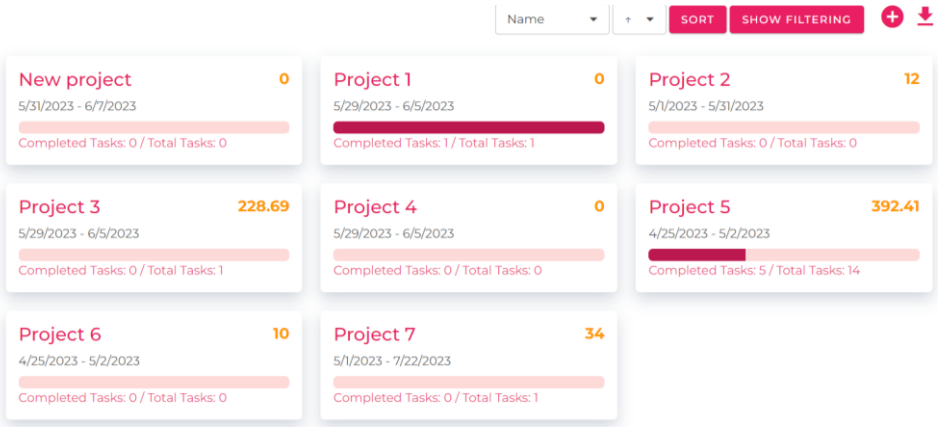

Рисунок 3.5 - Список проєктів після створення

Якщо виникає потреба відредагувати проєкт, достатньо клікнути на картку бажаного проєкту і на екрані з’явиться форма редагування (рис. 3.6). Після оновлення бажаних параметрів потрібно натиснути на кнопку “Save” і усі зміни будуть збережені і відображені, як показано на рисунку 3.7.

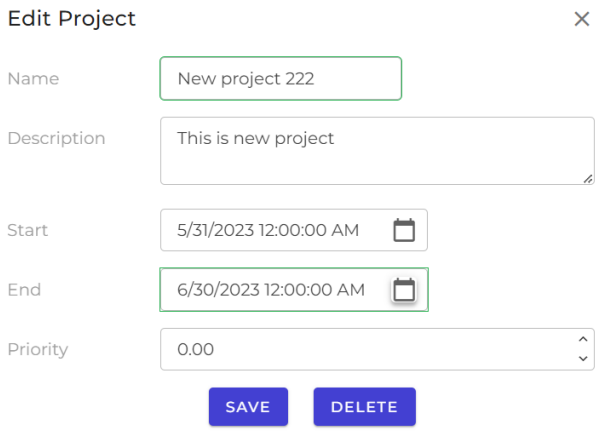

Рисунок 3.6 - Форма редагування проєкту

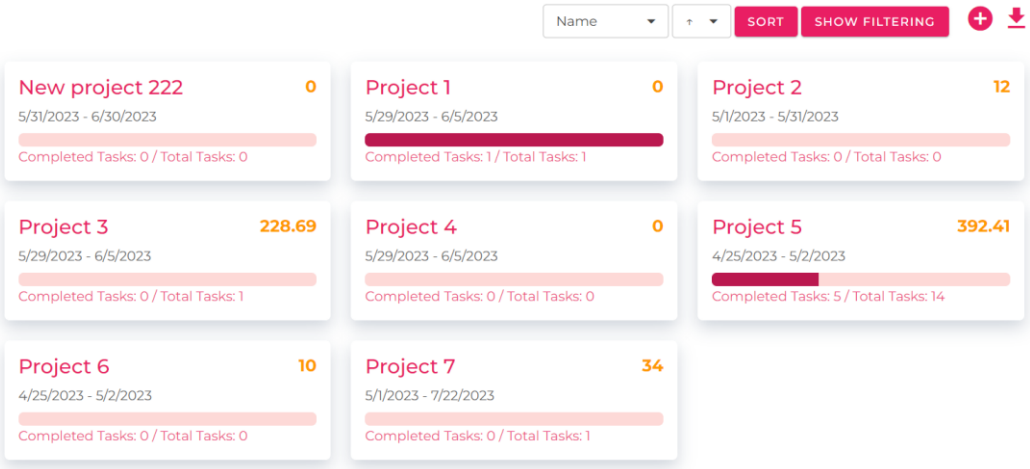

Рисунок 3.7 – Список проєктів після редагування

Для видалення проєкту необхідно знову відкрити форму редагування проєкту, що показана на рисунку 3.6, і натиснути на кнопку “Delete”. Після цього форму буде закрито, а і проєкт буде прибрано із загального списку, як зображено на рисунку 3.8.

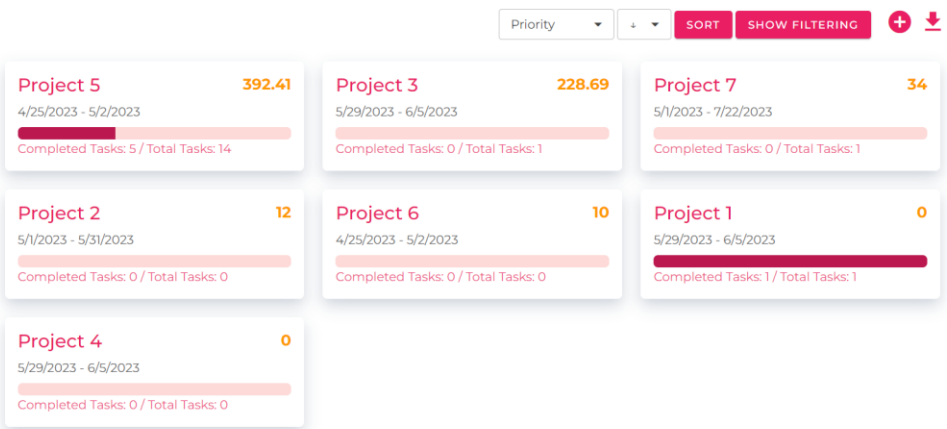

Рисунок 3.8 - Список проєктів після видалення

Сторінка із завдання дозволяє проводити різноманітні маніпуляції над завданнями, проте авторизований користувач може лише переглядати свої завдання та змінювати їх статус, а додавати, редагувати та переглядати усі завдання може адміністратор. Сторінка із завданнями містить 3 вкладення, які візуалізують завдання різними способами:

- календар
- kanban-дошка
- таблиця.

На вкладці з календарем (рис. 3.9) авторизований користувач може переглядати свої завдання за день, місяць та тиждень. Для того, щоб додати нове завдання прямо тут достатньо клікнути на пустому слоті і тоді з’явиться форма створення завдання. Якщо є бажання оновити завдання можна клікнути на нього і тоді на екрані з’явиться форма для редагування.

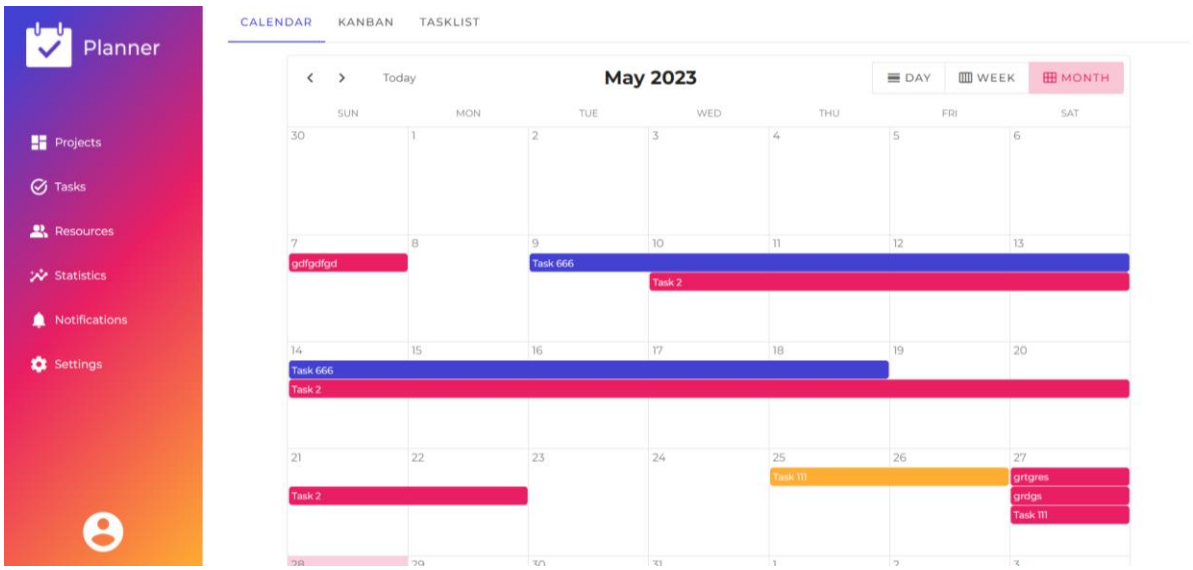

Рисунок 3.9 - Вкладка календар на сторінці завдань

Наступною є вкладка, що предсталяє завдання у вигляді kanban-дошки, на якій є три категорії або ж стани завдань: ToDo, InProgress і Done (рис. 3.10). Дана вкладка також надає можливості перегляду створення, редагування та видалення завдань. Стосовно редагування, то при перетягування завдання з однієї дошки на іншу його статус змінюється. Усі

завдання, відображені на kanban-дошці, відсортовані в порядку спадання їх пріоритету.

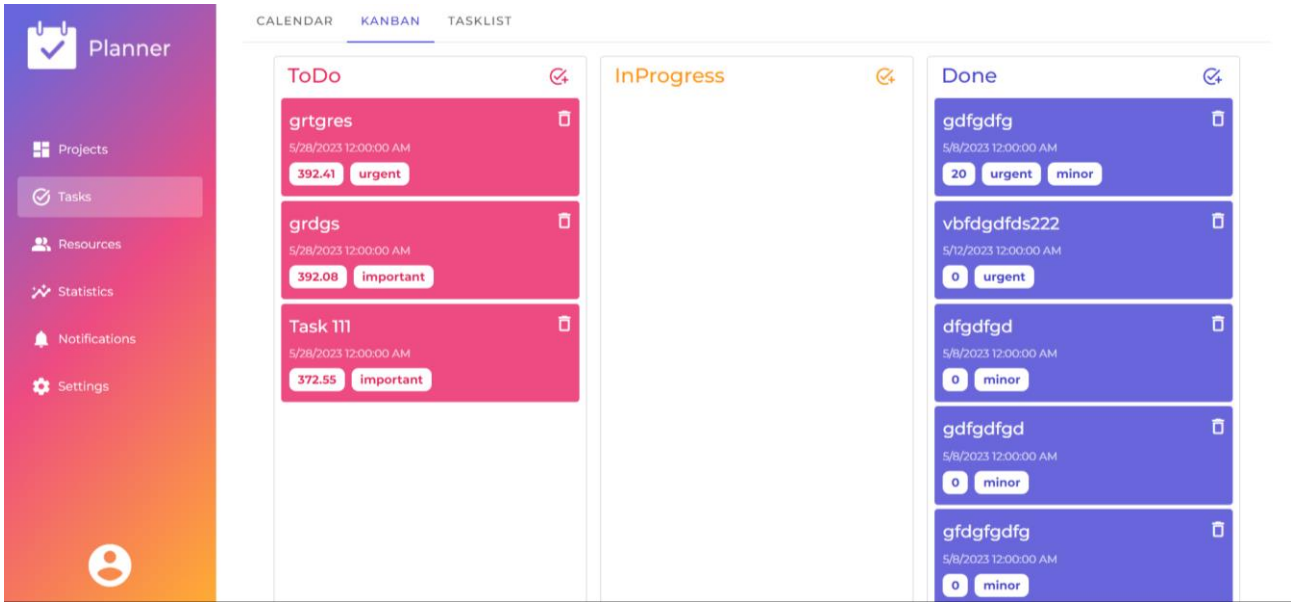

Рисунок 3.10 - Вкладка kanban-дошки на сторінці завдань

Для того, щоб створити нове завдання на даній сторінкці(рис. 3.10) потрібно натиснути на кнопку “+”. Після цього на екрані користувача з’явиться модальне вікно. В даному прикладі завдання додається в категорію InProgress. Також вартьо зазначити, що завдання неможливо створити, не призначивши як мінімум одного користувача для виконання. Оскільки форма для додавання є пустою, потрібно внести дані, наприклад, як на рисунку 3.11.

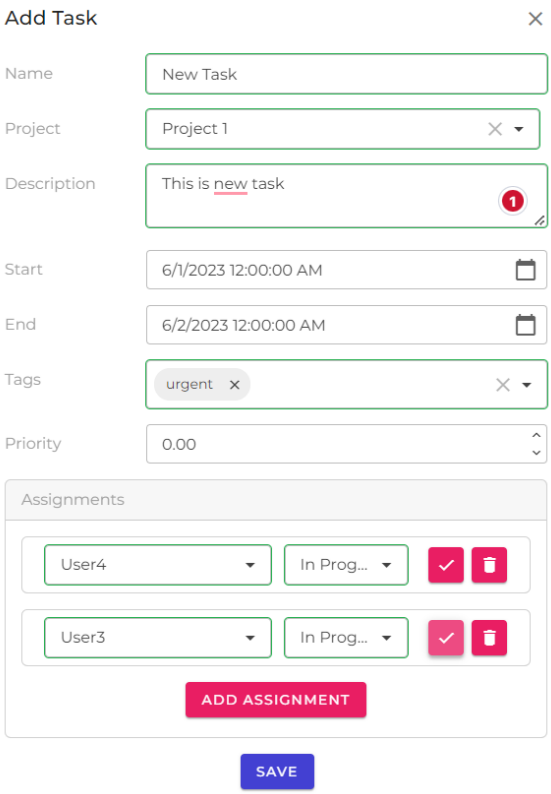

Рисунок 3.11 - Заповнена форма для додавання завдань

При натисканні на кнопку “Save”, модальне вікно буде закрито, завдання буде додано в категорію InProgress (рис. 3.12).

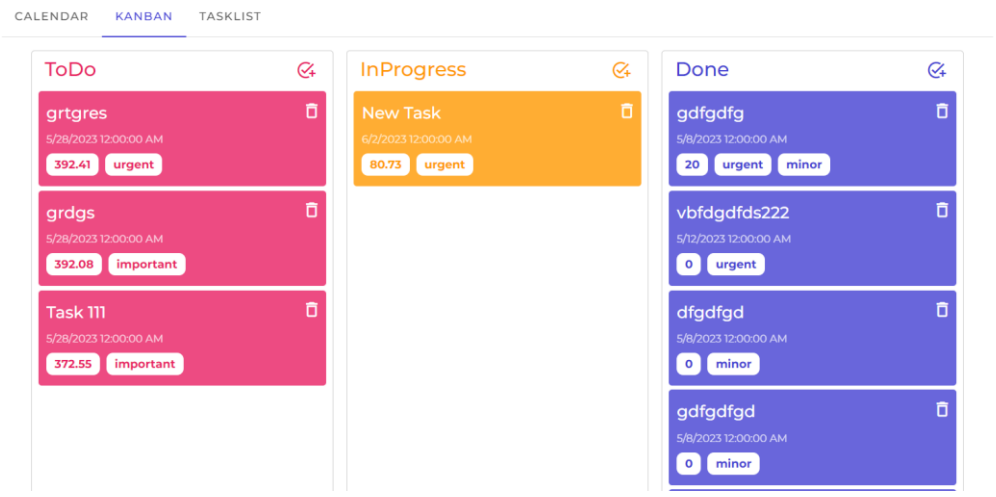

Рисунок 3.12 - Результат додавання нового завдання

Щоб обрати інши терміни виконання завдання, користувач може клікнути на назву завдання, на його екрані відкриється форма для редагування, де він може змінити бажані параметри. На рисунку 3.13 зображено модальне вікно з виправленою кінцевою датою виконання.

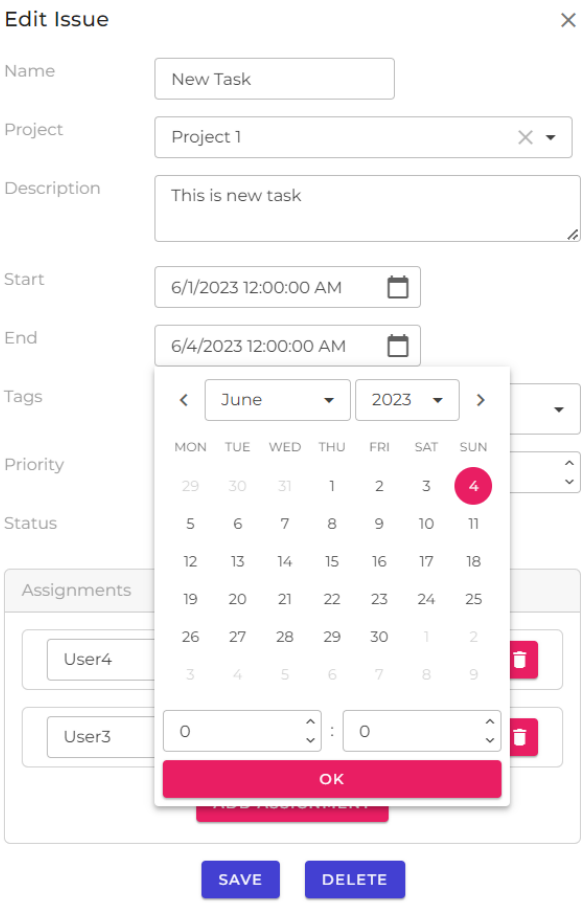

Рисунок 3.13 - Форма для редагування завдання

Для того, щоб змінити статус завдання, користувачеві потрібно перетягнути завдання на іншу дошку. Наприклад на рисунку 3.14 зображено процес зміни статусу для завдання “New task” з InProgress в Done, а на рисунку 3.15 можна побачити результат перетягування. Проте оскільки на дане завдання призначені два користувачі “User3” і “User4”, а перетягував завдання “User4”, то статус завдання буде змінено лише його, для “User3” статус залишиться незмінним, як показано на рисунку 3.16 .

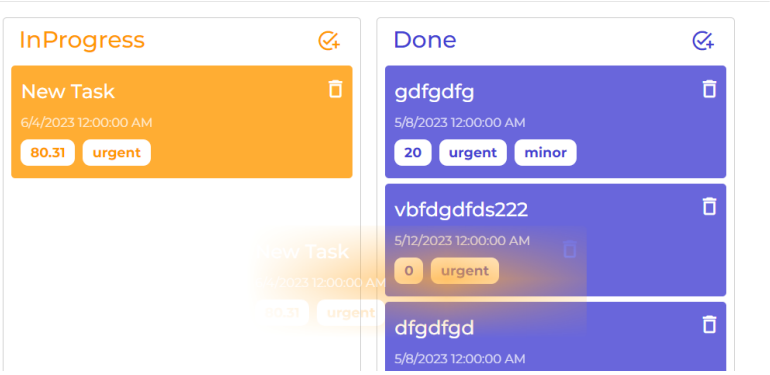

Рисунок 3.14 - Процес перетягування завдання

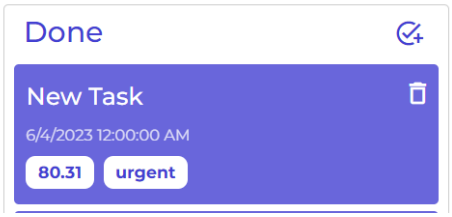

Рисунок 3.15 - Результат перетягування завдання для користувача “User4”

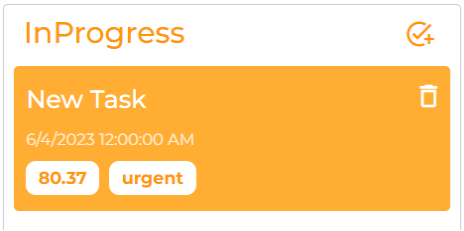

Рисунок 3.16 - Результат перетягування завдання для користувача “User3”

Якщо користувач забажає прибрати якесь завдання, то він може натиснути на кнопку видалення, що є на картці кожного завдання. На рисунку 3.17 можна побачити результат видалення завдання з назвою “New task”.

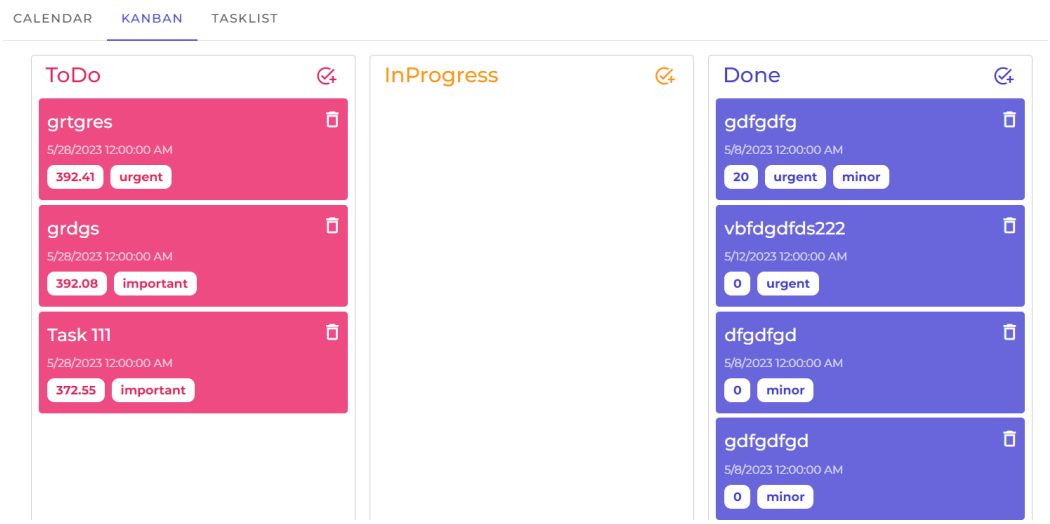

Рисунок 3.17 - Результат видалення завдання

Завдання також відображаються у вигляді таблиці, як представлено на рисунку 3.18. Саме на цій сторінці доступна фільтрації та сортування завдань.

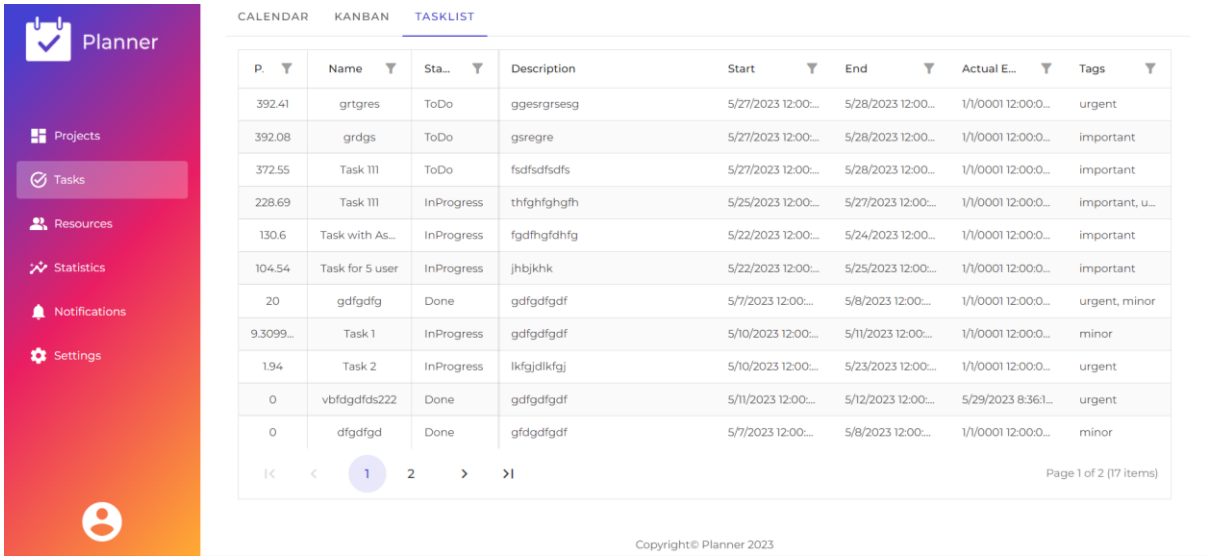

Рисунок 3.18 - Вкладка таблиці на сторінці завдань

На сторінці з ресурсами представлено дві вкладки. Одна з вкладок відповідає за відображення груп та надає механізми роботи над ними, як можна побачити на рисунку 3.19. Інша надає інформацію про користувачів та дозволяє працювати з ними(рис. 3.20).

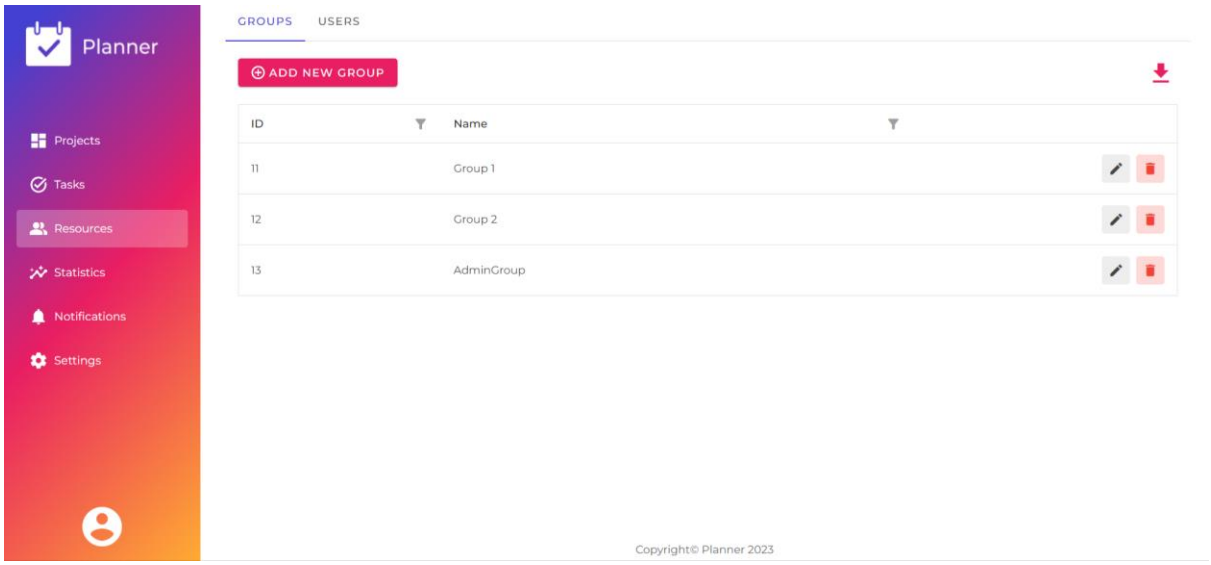

Рисунок 3.19 - Вкладка зі списком груп

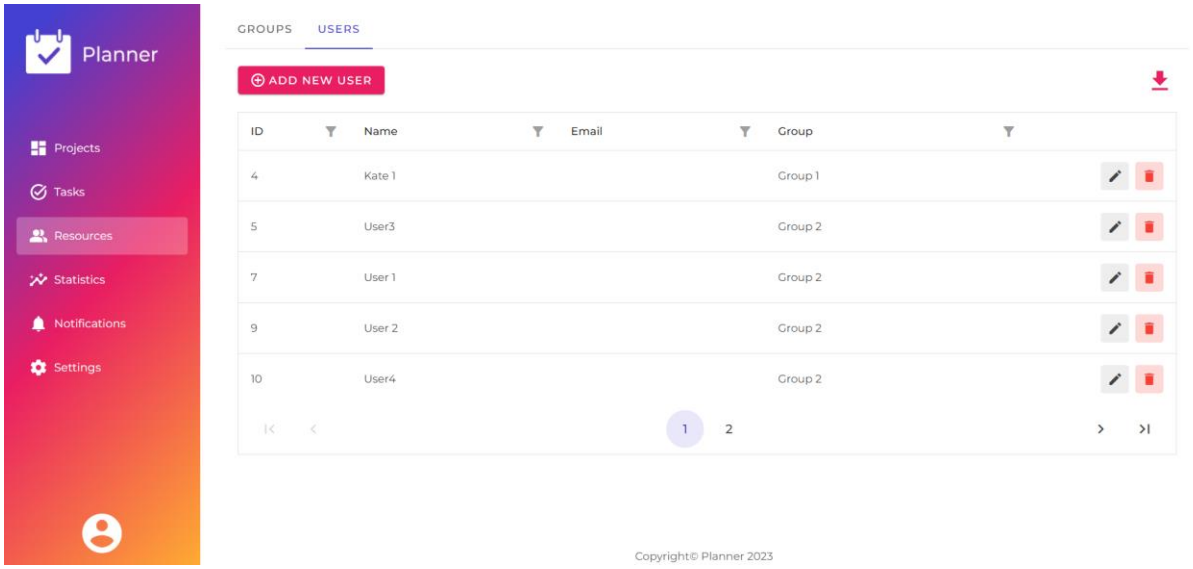

Рисунок 3.20 - Вкладка із списком користувачів

Сповіщення про нові завдання та близькі дедлайни можна побачити на сторінці з нотифікаціями(рис. 3.21).

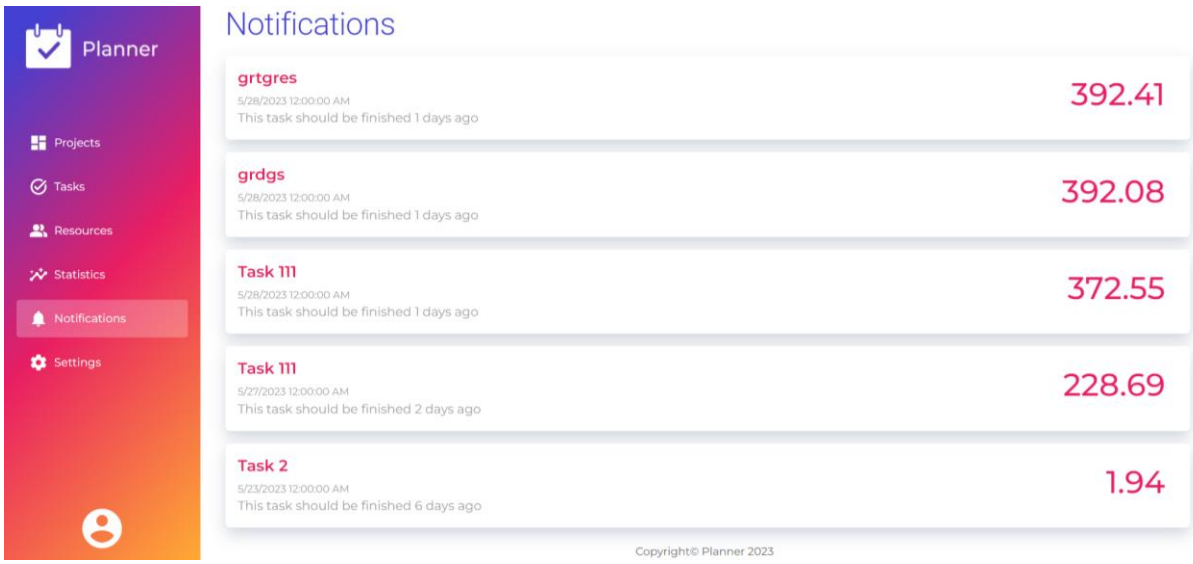

Рисунок 3.21 - Сторінка зі сповіщеннями

Сторінка з особистим кабінетом користувача представлено на рисунку 3.22. Саме тут клієнт може замінити свою особисту інформацію, оновити пароль чи вийти із системи.

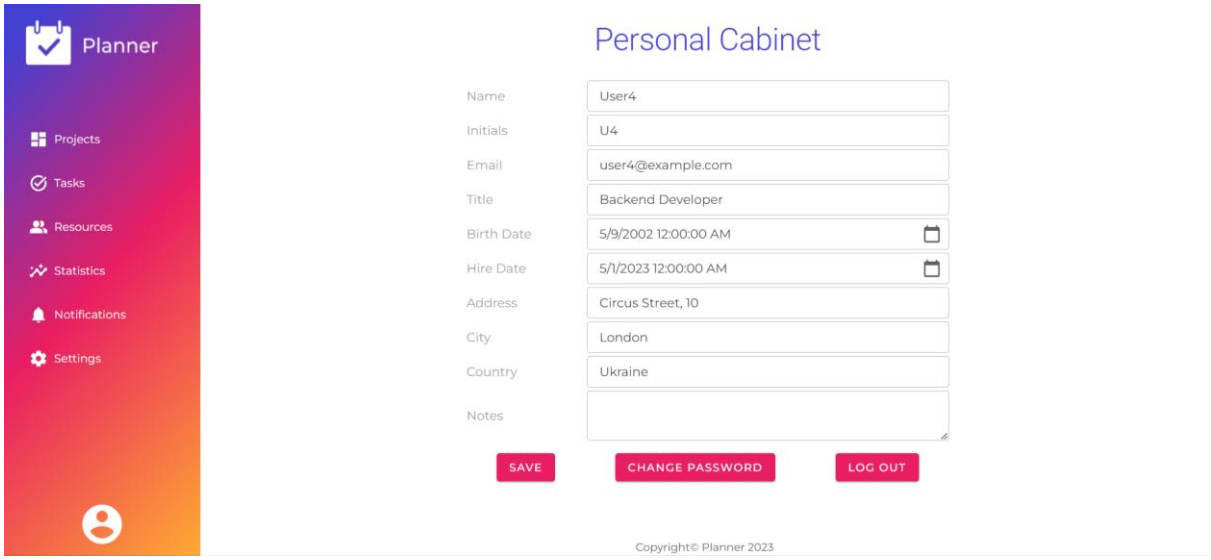

Рисунок 3.22 - Особистий кабінет користувача

3.4 Огляд процесу тестування системи управління проєктами

Невід'ємним етапом реалізації системи є тестування, тому нижче було описано тест-кейси для перевірки працездатності системи управління проєктами.

1. Створення проєкту

Передумови тест-кейсу: користувач має доступ до системи і права адміністратора.

Кроки тест-кейсу:

1. Зайти на сторінку з проєктами.
2. Натиснути на кнопку "+".
3. Ввести необхідні значення в модальне вікно(ім'я, опис, початкова дата та бажаний термін завершення).
4. Натиснути на кнопку "Save".

Очікуваний результат: проєкт було успішно створено з значенням пріоритету 0 та відобрабражено поруч з іншими проєктами на екрані. Процес відтворення тест-кейсу і його результати зображені на рисунках 3.3-3.5.

2. Редагування проєкту

Передумови тест-кейсу: користувач має доступ до системи і права адміністратора, а також в системі є хоча б один проєкт.

Кроки тест-кейсу:

1. Зайти на сторінку з проєктами.
2. Обрати проєкт для редагування.
3. Змінити необхідні значення в модальому вікні (ім’я, опис, початкова дата та бажаний термін завершення).
4. Натиснути на кнопку “Save”.

Очікуваний результат: дані будуть успішно збережені. Процес здійснення тест-кейсу і його результати зображені на рисунках 3.6-3.7.

3. Видалення проєкту

Передумови тест-кейсу: користувач має доступ до системи і права адміністратора, а також в системі є хоча б один проєкт.

Кроки тест-кейсу:

1. Зайти на сторінку з проєктами.
2. Обрати проєкт для видалення.
3. В модальому вікні натиснути на кнопку “Delete”.

Очікуваний результат: проєкт було видалено зі сторінки, а також завдання, що стосуються даного проєкту, були видалені. Результат видаленяя проєкту зображений на рисунках 3.7-3.8.

4. Створення нових завдань на сторінці з kanban-дошкою.

Передумови тест-кейсу: користувач має доступ до програмного модулю системи та права адміністратора, а також в системі раніше було створено деякі проєкти та користувачі.

Кроки тест-кейсу:

1. Зайти на сторінку з kanban-дошкою.
2. Натиснути на кнопку “+” в будь-якій з категорії.
3. Ввести необхідні значення в модальне вікно(ім’я, опис, пріоритет, мітки, початкова дата та бажаний термін виконання).

4. Обрати проєкт для даного завдання.
5. Призначити користувачів для виконання.
6. Натиснути на кнопку “Save”.

Очікуваний результат: завдання буде збережено в категорії, яка була обрана на 2 кроці. При оновленні сторінки завдання на kanban-дошці все одно коректно відображатиметься, а також воно має бути додано на сторінки календаря і таблиці завдань. Процес здійснення тест-кейсу і його результати зображені на рисунках 3.10-3.12.

5. Створення нових завдань на сторінці з календарем.

Передумови тест-кейсу: користувач має доступ до програмного модулю системи та права адміністратора, а також в системі раніше було створено деякі проєкти та користувачі.

Кроки тест-кейсу:

1. Зайти на сторінку з календарем.
2. Обрати бажану початкову дату виконання завдання.
3. Натиснути на будь-яке порожнє місце в клітинці даної дати.
4. Ввести необхідні значення в модальне вікно(ім’я, опис, пріоритет, мітки, початкова дата та бажаний термін виконання).
5. Обрати проєкт для даного завдання.
6. Призначити користувачів для виконання.
7. Натиснути на кнопку “Save”.

Очікуваний результат: створене завдання повинно бути коректно відображено на сторінці з календарем, міра його розтягнення має відповідати початковій і кінцевій датам. Також усі зміни має бути додані на сторінки kanban-дошки і таблиці завдань.

6. Редагування завдань на сторінці з kanban-дошкою.

Передумови тест-кейсу: користувач має доступ до програмного модулю системи з правами адміністратора і вже створив якісь завдання.

Кроки тест-кейсу:

1. Зайти на сторінку з kanban-дошкою.
2. Обрати завдання, яке потрібно відредагувати.
3. Натиснути на ім’я завдання.
4. В модальному вікні відредагувати значення (ім’я, опис, пріоритет, мітки, початкова дата, бажаний термін виконання, проект або призначений користувач).
5. Натиснути на кнопку “Save”.

Очікуваний результат: відредаговані дані повинні бути коректно збережені і мають бути оновлені на всіх сторінках.

7. Редагування завдань на сторінці з календарем.

Передумови тест-кейсу: користувач має доступ до програмного модулю системи з правами адміністратора і вже створив якісь завдання.

Кроки тест-кейсу:

1. Зайти на сторінку з календарем.
2. Обрати завдання, яке потрібно відредагувати.
3. Натиснути на слот завдання.
4. В модальному вікні відредагувати значення (ім’я, опис, пріоритет, мітки, початкова дата, бажаний термін виконання, проект або призначений користувач).
5. Натиснути на кнопку “Save”.

Очікуваний результат: відредаговані дані повинні бути коректно збережені і оновлені на всіх сторінках.

8. Перетягування завдання з однієї дошки на іншу.

Передумови тест-кейсу: користувач має доступ до системи, а для нього є завдання.

Кроки тест-кейсу:

1. Зайти на сторінку з kanban-дошкою.
2. Обрати завдання, якому потрібно змінити статус.

3. Натиснути на слот завдання і перетягнути його в бажану категорію.

Очікуваний результат: якщо над завданням працює лише один користувач, то статус завдання має бути змінений, проте якщо над завданням працюють декілька користувачів, то статус завдання має змінитись лише для того, хто перетягував це завдання. Статуси інших користувачів мають бути незмінені. Процес відтворення тест-кейсу і його результати зображені на рисунках 3.14-3.16.

9. Видалення завдань з kanban-дошки.

Передумови тест-кейсу: користувач має доступ до програмного модулю системи і вже створив якісь завдання.

Кроки тест-кейсу:

1. Зайти на сторінку з kanban-дошкою.
2. Обрати завдання для видалення.
3. Натиснути на кнопку видалення.

Очікуваний результат: завдання відсутнє на усіх сторінках системи. Результат видалення завдання зображений на рисунку 3.17.

Після проведення тестування можна зробити висновок, що розроблене програмне забезпечення працює коректно при управлінні проєктами та завданнями. Застосунок надає можливості різноманітних маніпуляцій із завданнями та зручний інтерфейс, що дозволяє кінцевому користувачеві якісно і швидко планувати свої проєкти та завдання.

3.5 Висновки до розділу практичної реалізації

Першим кроком при практичній реалізації було обрано інструментальні засоби для розробки системи управління проєктами, ними виявилися платформа .NET і мова програмування C#, адже вони надають

зручні механізми для створення і підтримки додатків, а також середовище для розробки Visual Studio 2022.

Задля кращого розуміння було побудовано діаграму структури клієнської частини програмного забезпечення та описано взаємодію усіх модулів системи. Серверну частину застосунку було представлено за допомогою діаграми класів, кожен клас було детально досліджено та описано.

Після завершення розробки було описано ряд тест-кейсів та проведено детальне тестування програмного модулю з метою перевірки його працездатності. За результатами даного етапу можна свідчити, що розроблена система управління проєктами працює правильно.

ВИСНОВКИ

Під час дипломного проєктування була вирішена наступна задача – проектування та розробка системи управління проєктами.

Спочатку було визначено сучасні проблеми планування, з метою кращого розуміння предметної області планування було визначено основні бізнес-процеси та проведено аналіз позитивних і негативних сторін систем, що вже існують. Також було сформовано постановку задачі на розробку проведено функціональний та нефункціональний аналіз майбутньої системи.

Далі передбачалась реалізація самого програмного продукту, тому було визначено архітектуру програмного застосунку і описано структуру бази даних, оскільки постановка задачі передбачала її використання. На поточному етапі було розроблено графічний інтерфейс програми та серверну частину. Завершальним етапом був опис тест-кейсів та тестування програмного продукту задля перевірки його працездатності.

Для розробки програмного застосунку було обрано мову програмування C# та платформу .NET. Протягом дипломного проєктування було вивчено особливості побудови веб-додатків за допомогою фреймворку ASP.NET. Завдяки проектуванню даного програмного продукту були здобуті навички зі створення графічної оболонки для додатків, реалізації програмного функціоналу та поєднання цих компонентів в єдине ціле.

Розроблений програмний застосунок можна удосконалити у багатьох напрямках, наприклад, додати механізми захисту даних або можливість інтеграції системи управління проєктами з іншими популярними інструментами, такими як системи контролю версій, комунікаційні інструменти або електронні поштові сервіси. Це допоможе забезпечити гладку взаємодію та обмін даними з іншими системами. Також можна розробити мобільний додаток або адаптувати існуючу систему для використання на мобільних пристроях, що дозволить користувачам

здійснювати управління власними проєктами з будь-якого місця та в будь-який час.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Project management Institute [Електронний ресурс] – Режим доступу: https://www.pmi.org/
2. Joseph M. Juran, A. Blanton Godfrey. Juran's quality handbook, 1998 – 130-140 ст.
3. Joseph Heagney. Fundamentals of Project Management, 2012 – 1-17ст.
4. Why is project management important? [Електронний ресурс] – Режим доступу: https://www.teamwork.com/project-management-guide/why-is-project-management-important/
5. Загальна характеристика систем управління проектами [Електронний ресурс] – Режим доступу: https://buklib.net/books/28871/
6. Hossenlopp R. Organizational project management: Linking strategy and projects / R. Hossenlopp. – Management Projects Inc. , 2010.
7. Roberts P. Effective project management / P. Roberts. – London – Philadelphia – New Delhi: Kogan Pages Ltd., 2012.
8. Steve McConnell. More Effective Agile: A Roadmap for Software Leaders – Construx Press, 2019.
9. Elizabeth Harrin. Collaboration Tools for Project Managers: How to Choose, Get Started and Collaborate with Technology – Project Management Institute; 1st edition, 2016.
10. Jim Highsmith. Adaptive Leadership: Accelerating Enterprise Agility – Addison-Wesley Professional; 1st edition, 2013.
11. Johanna Rothman. Agile and Lean Program Management: Scaling Collaboration Across the Organization – Practical Ink, 2016.
12. Atlassian [Електронний ресурс] – Режим доступу: https://www.atlassian.com/
13. Epicflow [Електронний ресурс] – Режим доступу: https://www.epicflow.com/

14. GitHub Docs [Електронний ресурс] – Режим доступу: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects
15. Trello [Електронний ресурс] – Режим доступу: https://trello.com/
16. Google WorkSpace Updates, 2018 [Електронний ресурс] – Режим доступу: https://workspaceupdates.googleblog.com/2018/06/google-tasks-to-launch-as-g-suite-core.html
17. Документація Microsoft [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/