

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)  
на тему:

**«Вебдодаток системи управління проектами»**

Виконав студент групи ІПЗк-22-1  
ЗИМА Дмитро Андрійович

Керівник роботи:  
САВІЦЬКИЙ Роман Святославович

Рецензент:  
ЛОКІТКОВА Тамара Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**

Завідувач кафедри інженерії  
програмного забезпечення  
Тетяна ВАКАЛЮК  
«26» лютого 2024 р.

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **ЗИМА Дмитро Андрійович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Вебдодаток системи управління проектами»**,  
затверджена Наказом закладу вищої освіти від «23» лютого 2024 р. №74с

Вихідні дані для роботи: **проектування та розробка вебдодатку системи управління проектами, включаючи їх структуру, функціонал, користувацький інтерфейс**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант   | Завдання видав | Завдання прийняв |
|--------|---------------|----------------|------------------|
| 1      | Савіцький Р.С | 27.02.2024р.   | 27.02.2024р.     |
| 2      | Савіцький Р.С | 27.02.2024р.   | 27.02.2024р.     |
| 3      | Савіцький Р.С | 27.02.2024р.   | 27.02.2024р.     |

## РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку системи управління проєктами та пояснювальної записки. Пояснювальна записка містить 67 сторінки, 25 ілюстрацію, 7 таблиці та 30 джерел.

Метою даної розробки є створення програмних засобів для автоматизації вирішення задачі «Вебдодаток системи управління проєктами» в рамках розробки системи «TeamWorker». Ця система спрямована на підвищення ефективності управління проєктами за допомогою інноваційних технологій та інструментів.

В результаті аналізу вхідних і вихідних даних, були розроблені логічна і фізичні моделі даних та описана організація інформаційної бази. Пояснювальна записка включає в себе результати розробки вебдодатку.

Система працює та взаємодіє зі спільною базою даних, завдяки створеним програмним процедурам.

**КЛЮЧОВІ СЛОВА:** ВЕБДОДАТОК, СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ, КЛІЄНТ-СЕРВЕР, БАЗА ДАНИХ

|           |      |                 |        |      |                                         |  |  |                                          |      |         |    |
|-----------|------|-----------------|--------|------|-----------------------------------------|--|--|------------------------------------------|------|---------|----|
|           |      |                 |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ                |  |  |                                          |      |         |    |
|           |      |                 |        |      |                                         |  |  |                                          |      |         |    |
| Змн.      | Арк. | № докум.        | Підпис | Дата |                                         |  |  |                                          |      |         |    |
| Розроб.   |      | Зима Д.А.       |        |      | Вебдодаток системи управління проектами |  |  | Літ.                                     | Арк. | Аркушів |    |
| Керівник. |      | Савіцький Р. С. |        |      |                                         |  |  |                                          |      | 3       | 67 |
|           |      |                 |        |      |                                         |  |  | Житомирська політехніка, група ІПЗк-22-1 |      |         |    |
| Рецензент |      | Савіцький Р. С. |        |      |                                         |  |  |                                          |      |         |    |
| Зав. каф. |      | Вакалюк Т. А.   |        |      |                                         |  |  |                                          |      |         |    |

## ABSTRACT

The Bachelor's qualification thesis consists of a web application for project management and an explanatory note. The explanatory note contains: 67 pages, 25 illustrations, 7 tables, and 30 references.

The purpose of this development is to create software tools to automate the task of «Web Application for Project Management» within the framework of the «TeamWorker» system development. This system aims to enhance the efficiency of project management through innovative technologies and tools.

The explanatory note of the thesis project contains the results of the development of an automated system. It analyzes the sets of input and output data for the task, describes the organization of the information database, and develops the logical and physical data models.

Software solutions for the task were developed in IntelliJ IDEA and WebStorm. The developed software procedures ensure the overall operation of the system and its interaction with the general system database.

**KEYWORDS:** WEB APPLICATION, PROJECT MANAGEMENT SYSTEM, CLIENT-SERVER, DATABASE.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 4    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ЗМІСТ

|                                                                                              |           |
|----------------------------------------------------------------------------------------------|-----------|
| <b>ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ .....</b>                                                       | <b>6</b>  |
| <b>ВСТУП.....</b>                                                                            | <b>7</b>  |
| <b>РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ .....</b> | <b>9</b>  |
| 1.1 Постановка задачі .....                                                                  | 9         |
| 1.2 Аналіз аналогів програмного продукту.....                                                | 10        |
| 1.3 Вибір архітектури Вебдодатку системи управління проєктами .....                          | 16        |
| 1.4 Обґрунтування вибору інструментальних засобів .....                                      | 18        |
| <b>Висновки до розділу 1.....</b>                                                            | <b>21</b> |
| <b>РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ ДЛЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ .....</b>              | <b>23</b> |
| 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи .....       | 23        |
| 2.2 Розробка бази даних системи.....                                                         | 30        |
| 2.3 Проєктування та реалізація алгоритмів роботи системи .....                               | 33        |
| 2.4 Реалізація вебдодатку .....                                                              | 35        |
| <b>Висновки до розділу 2.....</b>                                                            | <b>40</b> |
| <b>РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ .....</b>   | <b>41</b> |
| 3.1 Порядок встановлення та налаштування параметрів системи .....                            | 41        |
| 3.2 Структура інтерфейсу вебдодатку системи управління проєктами .....                       | 42        |
| 3.3 Тестування роботи програмного комплексу .....                                            | 52        |
| <b>Висновки до розділу 3.....</b>                                                            | <b>54</b> |
| <b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....</b>                                                      | <b>57</b> |
| <b>ДОДАТКИ.....</b>                                                                          | <b>60</b> |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTTP – протокол передачі гіпертексту

SQL – мова структурованих запитів

АС – автоматизована система

БД – база даних

ОС – операційна система

ПК – персональний комп'ютер

СУБД – система управління базою даних

ІСУП – інформаційна система управління проектами

JSON – JavaScript Object Notation

MVC – Model-view-controller

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 6    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ВСТУП

**Актуальність.** Сучасний світ характеризується високою динамікою та швидким розвитком технологій, що значно змінює підходи до управління проєктами. Успішне виконання проєктів вимагає не лише ефективного планування, але й гнучкого управління ресурсами та своєчасного коригування дій. Вебдодатки для управління проєктами стають незамінними інструментами, що дозволяють забезпечити оперативний доступ до даних, підтримку комунікації між членами команди та моніторинг виконання завдань у режимі реального часу. Актуальність даної теми полягає в необхідності створення інтегрованих рішень, які сприяють підвищенню ефективності командної роботи та якості управління проєктами в умовах постійних змін і високої конкуренції.

**Основною метою** даної роботи є розробка інноваційного вебдодатка для управління проєктами, який об'єднає в собі найкращі практики сучасного менеджменту та інформаційних технологій. Додаток має забезпечити користувачам можливість ефективно планувати та контролювати проєкти, розподіляти завдання, відстежувати їх виконання.

Для вдалого завершення данної мети потрібно вирішити ряд завдань:

1. Аналіз поточних конкурентних продуктів;
2. Встановлення вимог користувачів до функціоналу додатку;
3. Проектування архітектури та користувацького інтерфейсу;
4. Розробка та тестування програмного забезпечення;
5. Оцінювання ефективності додатку в реальних умовах.

**Об'єктом** даного дослідження є процеси та інструменти управління проєктами, які охоплюють етапи планування, реалізації, контролю та завершення проєктів, а також взаємодію між членами проєктної команди.

**Предметом** дослідження виступає процес розробки вебдодатка для управління проєктами, який включає аналіз вимог користувачів, проектування архітектури додатка, реалізацію функціональних можливостей та тестування ефективності його використання в реальних умовах.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 7    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Ця робота спрямована на створення високоякісного та зручного у використанні вебдодатка, який стане ефективним інструментом для менеджерів проєктів та їх команд, допомагаючи їм досягати поставлених цілей та забезпечувати успішне виконання проєктів в умовах сучасного динамічного середовища.

Тези до даної кваліфікаційної роботи [1].

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 8    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

# РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

## 1.1 Постановка задачі

Компанії, що займаються проєктною діяльністю, потребують ефективного управління. Якщо проєкти не вимагають значних фінансових вкладень та активної командної роботи, управління може бути розрізненим та безсистемним, що негативно впливає на бізнес. Практика показує, що зазвичай компанії реалізують кілька проєктів одночасно, і при цьому задіяні ті самі люди. Автоматизація управління проєктами значно полегшує роботу працівників та підвищує ефективність діяльності.

Система управління проєктами включає ряд методик та інструментів технологічного й організаційного характеру, спрямованих на управління проєктною діяльністю підприємства та підвищення ефективності їх реалізації. Досить часто це поняття розглядається вузько, маючи на увазі під системою управління проєктами певну програму, тобто інформаційну чи автоматизовану систему.

Основи майбутньої моделі управління проєктами закладаються вже сьогодні. Вони складаються з дій щодо запровадження та розвитку інформаційних систем управління проєктами (ІСУП), які включають технічні, програмні, інформаційні та комунікаційні засоби. На сучасному етапі розвитку проєктної діяльності, коли далеко не кожен задум має успішний фінал, автоматизація управління проєктами є досить затратною та не вигідною справою. Однак постійно ведеться робота над покращенням загального процесу, розробляються оригінальні рішення, тому можливість здійснення віртуального управління стає все більш реальною.

Сучасні компанії активно застосовують спеціалізовані програми для автоматизації управління проєктами, що інтегрують усю необхідну інформацію в один центр управління. Це сприяє підвищенню продуктивності працівників та

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 9    |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

поліпшенню якості керівництва проєктами, забезпечуючи ефективніше використання ресурсів і автоматизоване введення та аналіз даних.

Аналіз показав, що автоматизована СУП повинна виконувати наступні функції:

1. Створення проєктів.
2. Призначення працівників на проєкт.
3. Створення завдань для працівників.
4. Задання пріоритетів та дедлайнів для завдань.
5. Вибір завдання працівником зі списку завдань для нього.
6. Контроль процесу виконання завдань.
7. Формування звітів по різних параметрах.
8. Перегляд статистики по користувачам.
9. Забезпечення необхідних рівнів доступу.

На основі вивчення предметної області вбачається, що завдання, яке буде автоматизоване, відноситься до типових аналітично-розрахункових задач і може бути вирішено з використанням програмних та інструментальних засобів, передбачених технічним завданням, включаючи елементи файл-серверних баз даних. Після впровадження проєкту буде автоматизовано процес управління проєктами та персоналом.

## **1.2 Аналіз аналогів програмного продукту**

Аналіз аналогів програмного продукту є важливою складовою процесу вибору оптимального рішення для вирішення конкретної задачі. Він дозволяє з'ясувати переваги та недоліки кожного з існуючих аналогів, що має вирішальне значення для прийняття обґрунтованого рішення щодо подальшого розвитку проєкту чи впровадження програмного продукту.

При аналізі аналогів програмного продукту важливо розглянути різноманітні спеціалізовані програми, які використовуються для контролю виконання робочих процесів. Наприклад, «Airtable» (рис. 1.1) відомий своєю

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 10   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

гнучкістю та можливістю організації даних у вигляді таблиць, баз даних або канбан-дошок [2].

Рисунок 1.1 – Airtable

Airtable — популярна система управління базами даних, яка поєднує в собі функціональність електронних таблиць і традиційних баз даних, що робить її дуже корисною для організації та управління різними проєктами та процесами. На відміну від класичних баз даних, Airtable має інтуїтивно зрозумілий інтерфейс, схожий на електронні таблиці, що дозволяє користувачам легко вводити та обробляти дані без необхідності глибоких технічних знань чи спеціального навчання.

Разом з тим, на відміну від звичайних електронних таблиць, Airtable пропонує розширені можливості управління даними, включаючи зв'язування таблиць між собою, створення складних структур баз даних та використання різних типів полів (наприклад, текст, числа, дати, вкладення файлів). Це забезпечує високу гнучкість і адаптивність для різних потреб.

Airtable підходить для широкого спектру використань: від управління проєктами та завданнями, відстеження інвентарю та ресурсів, до ведення CRM-системи для управління відносинами з клієнтами. Вона дозволяє користувачам створювати налаштовані рішення, які ідеально підходять для конкретних бізнес-процесів чи особистих проєктів.

Завдяки цим характеристикам, Airtable здобула популярність серед користувачів, які шукають інструмент, що поєднує простоту використання електронних таблиць із потужністю баз даних, забезпечуючи водночас гнучкість та багатофункціональність у роботі з даними.

Однак, як і будь-яка програма, Airtable має свої недоліки. Давайте розглянемо деякі з них. Залежність від інтернету є одним з головних мінусів Airtable, адже це хмарний сервіс, і для доступу до даних необхідно мати стабільне інтернет-з'єднання. Відсутність офлайн-режиму може створювати труднощі для деяких користувачів. Обмежений функціонал CRM також можна віднести до недоліків, оскільки хоча Airtable можна використовувати як CRM, вона може не мати всіх спеціалізованих функцій, що пропонуються іншими CRM-системами, таких як автоматизація процесів продажу чи глибока аналітика.

Також варто зазначити обмеження у масштабуванні. Airtable може бути недостатньо потужною для дуже великих організацій або складних баз даних, адже є обмеження на кількість записів та обсяг вкладень у безкоштовних та навіть деяких платних планах. Вартість є ще одним фактором, що може стати проблемою. Для великих команд або організацій з великими обсягами даних Airtable може стати досить дорогою через платні тарифи.

Таким чином, незважаючи на свої численні переваги, Airtable має певні обмеження, які варто враховувати при виборі цієї платформи для роботи.

Давайте розглянемо програму «Asana» (рис. 1.2). Asana – популярний інструмент для управління проєктами, що дозволяє ефективно контролювати виконання завдань. Завдяки Asana можна створювати проєкти, додавати нових користувачів, створювати завдання та розподіляти їх між учасниками команди.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 12   |

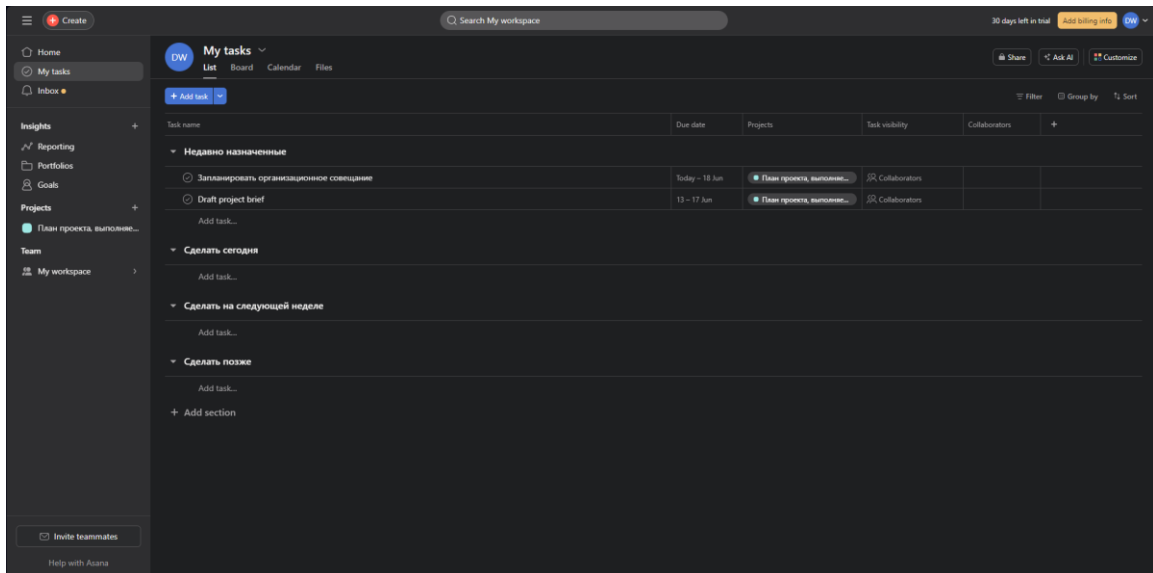


Рисунок 1.2 – Asana

Переваги Asana включають додавання співвиконавців та спостерігачів, що забезпечує кращу координацію роботи команди, дозволяючи залучати до завдань різних учасників [3]. Функція «Виконується» дозволяє стежити за перебігом виконання завдань у реальному часі, підвищуючи прозорість процесу. Розширена бізнес-звітність у Asana допомагає аналізувати продуктивність та ефективність команди, надаючи цінну інформацію для прийняття управлінських рішень завдяки вбудованим звітам.

Недоліки Asana включають відсутність ієрархії співробітників, що може створювати труднощі у великих командах, де необхідна чітка структура та розподіл ролей. Також важко розставляти пріоритети для проєктів, особливо у великих проєктах, оскільки інструменти для пріоритизації завдань реалізовані не найкращим чином. Завдання можуть забуватися і губитися, що стає серйозною проблемою у великих проєктах з великою кількістю завдань.

Таким чином, для великих підприємств може бути вигідніше розробка програмних продуктів, спеціально адаптованих під їхні потреби, замість використання готових рішень, таких як Asana.

Jira – потужний інструмент для управління проєктами, розроблений переважно для відділів розробки (рис. 1.3) [4]. Вона може бути використана

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 13   |

великими командами понад 100 осіб і часто використовується як баг-трекер, хоча її функціональність виходить далеко за ці межі.

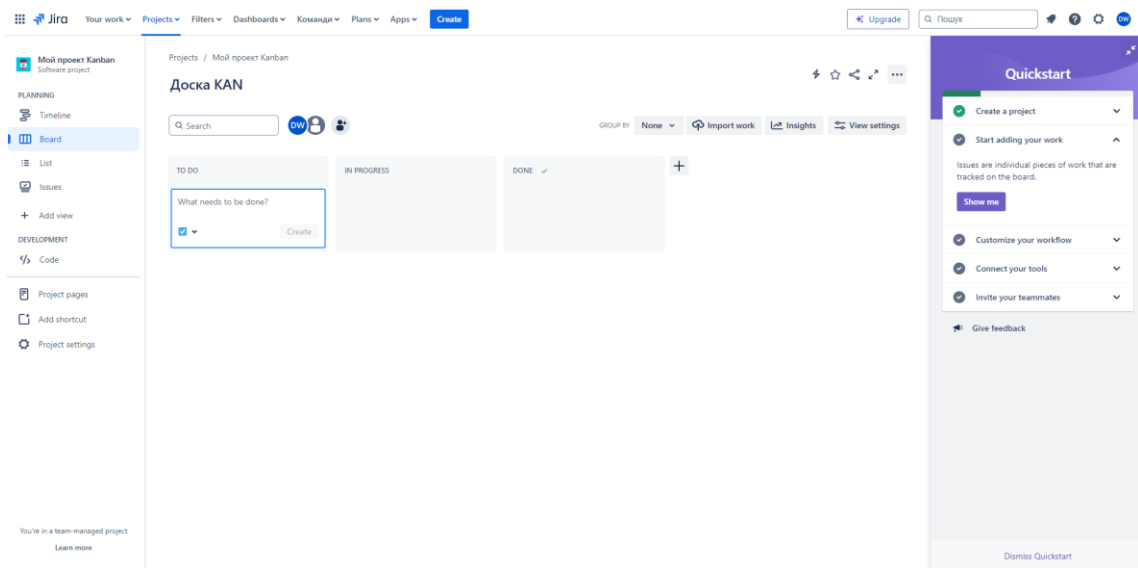


Рисунок 1.3 – Jira

Переваги Jira включають величезний набір функцій та можливостей для кастомізації, що дозволяє гнучко налаштовувати систему під потреби конкретного проєкту або команди. Потужний API дає можливість інтегрувати Jira з іншими системами та автоматизувати багато процесів. Інструменти для зручного планування спринтів та Agile-беклог особливо корисні для команд, що працюють за методологіями Agile. Більше 3000 додатків дозволяють розширювати функціональність Jira та адаптувати її під специфічні потреби проєктів. Розрахунок швидкості та обсягу роботи команди за спринт допомагає краще планувати роботу та оцінювати продуктивність команди. Вимірювання та візуалізація прогресу роботи дозволяють отримувати візуальні звіти про стан проєкту, що спрощує контроль та управління.

Недоліки Jira включають складність в освоєнні, що може бути викликом, особливо для "не-технарів", і потребує значного часу на навчання. Часті зміни інтерфейсів можуть створювати незручності для користувачів, оскільки доводиться постійно адаптуватися до нових змін. Складність налаштування власних робочих процесів може бути викликом для кожного окремого проєкту, особливо якщо потрібно гнучко налаштовувати робочі процеси. Мало можливостей для автоматизації може обмежувати ефективність використання

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 14   |

системи у великих проєктах. Повільне завантаження сторінок також може бути проблемою для користувачів із розширеними правами та великою кількістю надбудов, що впливає на швидкість роботи системи.

Отже, хоча Jira надає потужний інструментарій для управління проєктами, її використання може бути складним та потребувати значних ресурсів на навчання та адаптацію під конкретні потреби команд.

На основі аналізу аналогів програмних продуктів було складено порівняльну таблицю (табл. 1.1):

Таблиця 1.1

Порівняння існуючих аналогів

|                                 | <b>Airtable</b> | <b>Asana</b> | <b>Jira</b> | <b>TeamWorker</b> |
|---------------------------------|-----------------|--------------|-------------|-------------------|
| Інтуїтивно зрозумілий інтерфейс | +               | +            | -           | +                 |
| Швидкість та продуктивність     | +               | +            | -           | +                 |
| Базові функції                  | +               | +            | +           | +                 |
| Ієрархія співробітників         | -               | +            | +           | +                 |
| Доступність (ціна)              | -               | +            | -           | +                 |
| Складність в освоєнні           | +               | +-           | -           | -                 |
| Додатковий функціонал           | -               | +-           | +           | -                 |

TeamWorker об'єднує багато позитивних аспектів програм Airtable, Asana і Jira, забезпечуючи широкий функціонал для управління проєктами і командами. Він є доступною альтернативою з точки зору ціни і функціоналу.

### 1.3 Вибір архітектури Вебдодатку системи управління проєктами

Вибір архітектури вебдодатку системи управління проєктами побудований на необхідності розробки системи, що поєднує високу ефективність і безпеку. В основі архітектури лежать Java для серверної розробки, Angular для створення динамічного інтерфейсу користувача, а також ключові компоненти Spring (Spring Boot, Spring Security, Spring MVC) для забезпечення роботи з бізнес-логікою, безпеки та масштабованості додатку [5]. Використання HTML і CSS дозволяє створити зручний інтерфейс для взаємодії з користувачем. Вибір оптимальної архітектури враховує потреби проєкту у масштабованості, надійності та зручності використання, щоб забезпечити успішну імплементацію системи управління проєктами.

Java — це об'єктно-орієнтована мова програмування, яка широко застосовується для створення серверних компонентів додатків [6]. Вона славиться своєю стабільністю, надійністю та високою продуктивністю. Java підтримує багатопоточність, що дозволяє ефективно обробляти значну кількість одночасних запитів.

Angular — це фронтенд-фреймворк, створений компанією Google, призначений для створення динамічних вебдодатків [7]. Він дозволяє розробникам створювати складні та інтерактивні інтерфейси користувача за допомогою компонентного підходу. Angular використовує двостороннє зв'язування даних, що робить синхронізацію між моделлю даних і її представленням значно простішою.

Spring Boot — це фреймворк, який спрощує процес створення, налаштування і розгортання готових до використання додатків на базі Spring [8]. Він забезпечує широкий набір інструментів для автоматичної конфігурації додатків і має вбудований веб-сервер, що дозволяє розробникам фокусуватися на логіці додатку, замість того щоб витрачати час на налаштування інфраструктури.

Spring Security є потужним інструментом для забезпечення безпеки вебдодатків [9]. Він забезпечує аутентифікацію та авторизацію користувачів,

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 16   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

управління сесіями і захист від різних видів атак, включаючи CSRF. Цей фреймворк дозволяє гнучко налаштовувати безпеку додатку відповідно до конкретних потреб проєкту.

Spring MVC (Model-View-Controller) — це фреймворк для розробки вебдодатків, який використовує принцип розділення логіки програми (модель), управління даними (контролер) і представлення інформації (вид) [10]. Ця архітектура дозволяє полегшити модульність та підтримуваність коду проєкту. Spring MVC є частиною інтегрованої екосистеми Spring і забезпечує ефективну обробку HTTP-запитів і відповідей.

HTML (HyperText Markup Language) — це стандартна мова розмітки для створення веб-сторінок, що визначає структуру контенту [11]. CSS (Cascading Style Sheets) — це мова стилізації, яка використовується для визначення вигляду веб-сторінок [12]. Використання HTML і CSS дозволяє створювати зручні та естетичні інтерфейси, які працюють на різних пристроях і браузерах.

Взаємодія клієнта з сервером буде здійснюватись шляхом виконання HTTP запитів клієнтом до RESTful API серверу [13,14].

REST, або REpresentational State Transfer, – це архітектурний стиль для забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємодію систем між собою. REST-сумісні системи, які часто називають системами RESTful, характеризуються тим, що вони не мають стану та відокремлюють проблеми клієнта і сервера.

Сервер буде напряду взаємодіяти з базою даних, опрацьовувати дані та повертати їх клієнту у форматі JSON.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними, що базується на JavaScript [15,16]. Легко читається людиною та машиною. Часто використовується в REST API (частіше, ніж XML) [17].

Використання сучасних технологій, таких як Java, Angular, Spring Boot, Spring Security, Spring MVC, HTML і CSS, дозволить створити високопродуктивний, безпечний і зручний для користувачів вебдодаток для управління проєктами.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 17   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Java забезпечить надійність і ефективність серверної частини. Angular надасть потужний інструмент для створення інтерактивного інтерфейсу. Spring Boot спростить налаштування і розгортання додатку. Spring Security захистить додаток від загроз. Spring MVC забезпечить чітку архітектуру додатку. HTML і CSS допоможуть створити сучасний і зручний інтерфейс користувача.

Цей набір технологій забезпечує комплексний підхід до розробки, що гарантує високу якість кінцевого продукту, здатного задовольнити вимоги сучасних користувачів щодо продуктивності, безпеки та зручності використання.

#### **1.4 Обґрунтування вибору інструментальних засобів**

Розробка вебдодатків у сучасному програмуванні вимагає обдуманого вибору технологій. Angular, в якості потужного клієнтського фреймворку, визначається не лише його здатністю ефективно створювати складні користувацькі інтерфейси, але й можливістю управління станом додатків і зручним взаємодією з серверними API.

Angular є потужним фреймворком для створення сучасних вебдодатків. Він відомий своєю компонентною архітектурою, яка сприяє чіткому розділенню функціональності на невеликі, самодостатні компоненти. Це значно спрощує розробку, тестування та підтримку коду, особливо великих проєктів, де структура має критичне значення.

Крім цього, Angular відрізняється продуктивністю розробки завдяки своєму інструментарію, зокрема Angular CLI [18]. Цей інструмент автоматизує рутинні завдання, такі як створення компонентів, модулів і роутінгу. Це дозволяє розробникам зосередитися на бізнес-логіці додатків, мінімізуючи час, витрачений на налаштування середовища розробки.

Angular також має потужну систему шаблонів і директив, які спрощують створення динамічних інтерфейсів користувача. Вони надають зручний синтаксис для роботи з подіями, зв'язком між компонентами і управлінням станом додатку.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 18   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Додатковою важливою перевагою Angular є підтримка від Google, що забезпечує стабільність і постійний розвиток фреймворку. Це також означає доступ до багатьох корисних ресурсів, таких як документація, навчальні матеріали і сторонні бібліотеки.

Java є однією з найпопулярніших мов програмування в світі, що має екосистему інструментів і бібліотек.

Java відома своєю надійністю та стабільністю. Вона широко використовується в критичних для бізнесу додатках і має доведену здатність до масштабування. Це робить її відмінним вибором для проєктів, де важлива висока доступність і оптимальна продуктивність.

Java пропонує потужну систему управління пам'яттю, що робить її особливо ефективною для довгоживучих застосунків з великим обсягом даних. Керування пам'яттю в Java автоматизоване завдяки сборщику сміття, що спрощує розробку і зменшує кількість помилок, пов'язаних з утечками пам'яті [19].

Java є мультиплатформеною мовою, що дозволяє запускати код на різних операційних системах без значних модифікацій [20]. Це особливо корисно у великих підприємствах або у випадку, коли необхідно забезпечити сумісність з різними середовищами.

І нарешті, Java має багато інструментів для розробників, таких як популярні фреймворки Spring і Hibernate, що значно спрощують розробку та підтримку бекенд-систем [21]. Ці інструменти роблять Java потужним інструментом для будь-якого типу веб-розробки, від простих веб-сайтів до складних мікросервісних архітектур.

Отже, обираючи Java для бекенд-розробки, ви отримуєте надійність, масштабованість, ефективне управління пам'яттю, мультиплатформеність і широкий вибір інструментів, що роблять її чудовим вибором для бізнесових застосунків і веб-систем усіх рівнів складності.

Щодо середовища розробки, IntelliJ IDEA вважається однією з найкращих IDE для розробки на Java [22]. Вона надає широкий набір функцій, таких як

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 19   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

автодоповнення коду, візуальне налагодження, підтримку фреймворків і інструментів для аналізу коду. IntelliJ IDEA сприяє зручному розвитку програмного забезпечення будь-якої складності.

Розробка серверної частини системи відбудеться за допомогою фреймворку Spring, який є одним з найбільш популярних інструментів для створення вебдодатків на Java. Spring представляє собою не конкретний фреймворк, а скоріше загальну назву для ряду компонентів, які можуть виконувати різноманітні функції. Він має модульну структуру, що дозволяє включати лише необхідні для проєкту модулі і не використовувати ті, які не потрібні.

Щодо WebStorm, це інтегроване середовище розробки для JavaScript і супутніх технологій від JetBrains [23]. Як і інші IDE від цієї компанії, WebStorm автоматизує рутинні завдання і дозволяє ефективно справлятися з складними викликами, що робить процес розробки більш захоплюючим. Вона надає автодоповнення, аналіз коду на льоту, навігацію по проєкту, можливості для рефакторингу, налагодження коду і інтеграцію з системами керування версіями.

Використання бази даних (БД) в проєкті є критично важливим з кількох причин, які сприяють ефективному і безпечному управлінню даними.

По-перше, БД забезпечує постійний доступ до даних, що є особливо важливим у сучасних додатках, де доступність інформації є ключовою. Система управління базами даних (СУБД) дозволяє ефективно зберігати, оновлювати та витягувати дані, забезпечуючи їх надійність і консистентність [24].

По-друге, використання БД сприяє забезпеченню безпеки даних. Сучасні СУБД надають механізми автентифікації, авторизації та захисту даних, що дозволяє уникнути несанкціонованого доступу і зберігати конфіденційність інформації.

По-третє, використання БД спрощує масштабування додатків. Велика кількість даних легко обробляється і швидко доступна для великої кількості користувачів, завдяки оптимізації запитів та індексації даних, що забезпечує ефективну роботу додатків під час зростання їх обсягів і навантажень.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 20   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

По-четверте, використання БД сприяє забезпеченню послуг вищого рівня, таких як аналітика та звітність. СУБД дозволяють проводити складні аналітичні операції, генерувати звіти та статистику, що допомагає в прийнятті стратегічних рішень на основі даних.

MySQL Workbench є потужним інструментом для роботи з базами даних, який дозволяє виконувати різноманітні завдання, починаючи від проєктування та моделювання структури даних до створення та адміністрування [25]. Цей інструмент особливо корисний для:

Резервного копіювання та відновлення баз даних: MySQL Workbench дозволяє створювати регулярні резервні копії баз даних, що забезпечує збереження даних у випадку втрати чи пошкодження.

Перенесення баз даних: Інструмент спрощує процес перенесення баз даних з одного сервера на інший, що є корисним при оновленні обладнання або зміні інфраструктури.

Підключення до віддалених баз даних: MySQL Workbench підтримує налаштування для роботи з віддаленими базами даних через мережу, що дозволяє віддалено управляти та моніторити їх.

Перегляд статистики та моніторингу баз даних: Інструмент надає корисну інформацію про різні показники, такі як використання пам'яті, кількість активних з'єднань та швидкість операцій, що сприяє вчасному виявленню проблем і оптимізації роботи системи.

Використання MySQL Workbench дозволяє ефективно розробляти та управляти базами даних, забезпечуючи високу надійність та ефективність обробки великих обсягів даних.

## **Висновки до розділу 1**

У даному розділі було розглянуто важливість вибору мови програмування Java для розробки бекенду. Java відзначається значною популярністю, надійністю та стабільністю, що робить її привабливим вибором для великих проєктів і систем з високим навантаженням. Вона підтримується широкою спільнотою розробників і має велику кількість доступних інструментів і

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 21   |

бібліотек, що спрощують розробку і підтримку коду. Крім того, Java забезпечує ефективне керування пам'яттю, мультиплатформеність і зручність у використанні.

Додатково, Java є ідеальним вибором для проєктів, які потребують високої масштабованості і розширюваності. Її екосистема включає потужні фреймворки, такі як Spring, які дозволяють швидко розгортати мікросервісні архітектури та інтегрувати з різноманітними технологіями. Це робить Java ідеальним вибором для сучасних додатків, що вимагають гнучкості і можливості швидко адаптуватися до змін у вимогах бізнесу і технологій.

Також, вибір Java для бекенд-розробки сприяє забезпеченню високої продуктивності команди розробників. Стандартизація мови та велика кількість ресурсів і підтримки дозволяють зменшити час розробки і підтримки проєктів, знижуючи загальні витрати на ведення ділової діяльності і збільшуючи швидкість випуску нових функцій і можливостей.

Отже, вибір Java для бекенд-розробки є обґрунтованим і може значно сприяти успішному впровадженню і подальшому розвитку інформаційного проєкту.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 22   |

## РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБДОДАТКУ ДЛЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

### 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

При проектуванні будь-якої інформаційної системи ключовим етапом є визначення варіантів використання та створення об'єктно-орієнтованої структури, яка відповідає потребам і вимогам користувачів і бізнесу. Варіанти використання визначають функціональність системи з точки зору її взаємодії з користувачем і іншими системами, в той час як об'єктно-орієнтована структура забезпечує гнучкість і модульність в розробці програмного забезпечення. У даному контексті важливо ретельно розглянути і проаналізувати різноманітні сценарії використання та вибрати оптимальні рішення для побудови ефективної та ергономічної системи.

#### Функціональні вимоги

Зважаючи на потреби та функціональні вимоги інформаційної системи, її структура та можливості визначаються кількома ключовими аспектами.

Реєстрація та авторизація користувачів: Ця функціональна вимога передбачає можливість користувачам створювати облікові записи в системі та авторизуватися для доступу до функціоналу. Реєстрація забезпечує ідентифікацію користувачів і зберігає їхні особисті дані, щоб забезпечити безпеку і відповідність вимогам доступу.

Створення та керування проєктами: Ця функція дозволяє адміністраторам та менеджерам створювати нові проєкти в системі з необхідною інформацією, такою як назва, опис, строки виконання та відповідальні особи. Вони також мають право на редагування та видалення проєктів відповідно до поточних потреб.

Генерація звітів: Функція забезпечує можливість автоматизованої генерації різноманітних звітів, які можуть включати інформацію про стан

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 23   |

проектів, виконані завдання, ефективність роботи співробітників тощо. Це допомагає в управлінні і прийнятті стратегічних рішень.

Управління персоналом: Функціональність передбачає можливість адміністраторів додавати, редагувати та видаляти дані про співробітників, включаючи їхні ролі та права доступу. Також важливо відстежувати зайнятість працівників та їхню участь у проєктах для оптимального розподілу ресурсів.

Мобільний доступ: Щоб забезпечити зручність і доступність для користувачів, система повинна включати мобільний додаток або адаптивний веб-інтерфейс. Це дозволяє користувачам працювати з проєктами та завданнями з будь-яких мобільних пристроїв, забезпечуючи повний функціонал навіть під час відсутності комп'ютера.

Ці функціональні вимоги забезпечують комплексний підхід до управління проєктами і персоналом, забезпечуючи ефективне використання ресурсів і покращення процесів в організації.

### **Нефункціональні вимоги**

Сприйняття: Система повинна надавати відповідь на звичайні запити протягом не більше 3 секунд, а на складні — не більше 15 секунд. Інтерфейс користувача має бути легким у використанні та інтуїтивно зрозумілим, щоб користувачі могли без зусиль здійснювати необхідні дії.

Надійність: Час, який витрачається на обслуговування платформи, не повинен перевищувати 10% від загального робочого часу. Середній час безвідмовної роботи системи повинен становити не менше 60 робочих днів. Максимальна частка помилок та дефектів в роботі системи не повинна перевищувати одну помилку на 1000 запитів користувачів.

Продуктивність: Система має бути швидкою і масштабованою, здатною обробляти великий потік користувачів і запитів без помітних затримок.

Можливість експлуатації: Система повинна мати можливість масштабуватися для відповіді на зростаючий обсяг користувачів, не знижуючи швидкість роботи. Оновлення версій повинні відбуватися в часи мінімального

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 24   |

навантаження на платформу, з мінімізацією перерв у роботі сайту під час завантаження оновлень.

Часові характеристики реакції системи: Система має надавати інстантний відгук на запити користувачів, забезпечуючи швидкий доступ до інформації та послуг. Мінімізація часу завантаження сторінок є критичною для забезпечення миттєвого доступу користувачів до основних функціональних можливостей системи. Також важливо, щоб система миттєво відображала зміни, внесені адміністратором, такі як додавання нових завдань чи оновлення вже виконаних. Негайна реакція на помилки та проблеми є необхідною для забезпечення ефективного вирішення проблем та надання необхідної підтримки користувачам.

#### Часові характеристики реакції системи

1. Інстантний відгук на запити користувачів. Система має надавати швидкий доступ до інформації та послуг.
2. Мінімізація часу завантаження сторінок. Система має швидко завантажувати основні сторінки магазину для миттєвого доступу користувачів до інформації.
3. Негайне оновлення задач. Система має миттєво відображати зміни, внесені адміністратором, такі як додавання нових задач чи оновлення вже виконаних.
4. Мінімізація часу реагування на помилки. Система повинна негайно реагувати на будь-які помилки чи проблеми під час взаємодії з користувачем, щоб швидко вирішувати проблеми та надавати необхідну допомогу.

Система розроблена з урахуванням різноманітних сценаріїв використання, щоб задовольнити потреби різних категорій користувачів. Гнучка архітектура системи дозволяє ефективно виконувати завдання навіть у вимогливих умовах і забезпечує зручний і надійний інтерфейс для користувачів усіх категорій.

Діаграма варіантів використання представляє візуальну взаємодію між системою та користувачами (рис. 2.1) .

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 25   |

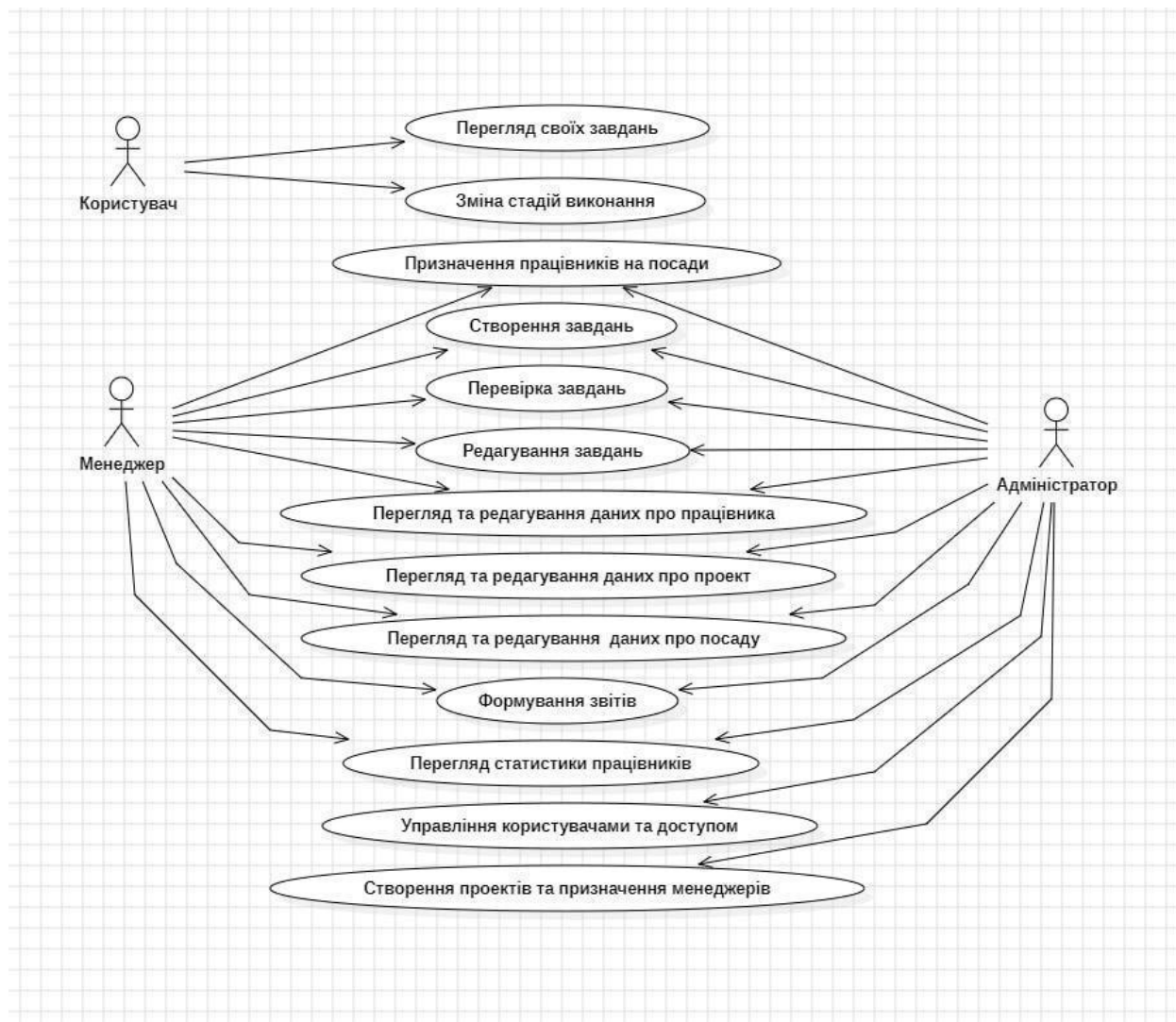


Рисунок 2.1 – Діаграма варіантів використання

### Основні актори:

1. Користувач – виконує завдання, призначені йому в системі. Може переглядати свої завдання та змінювати стадії виконання завдань.
2. Менеджер – призначає працівників на посади, створює, перевіряє та редагує завдання, переглядає та редагує дані про працівників, проекти та посади, формує звіти та переглядає статистику працівників.
3. Адміністратор – має повний доступ до управління користувачами та їхніми доступами, створює проекти та призначає менеджерів, переглядає та редагує дані про працівників, проекти та посади, формує звіти та переглядає статистику працівників.

### Короткий опис прецедентів:

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 26   |

1. Перегляд своїх завдань. Користувач переглядає список завдань, які йому призначені.
2. Зміна стадій виконання. Користувач може змінювати стадії виконання своїх завдань (наприклад, з «Виконується» на «Завершено»).
3. Призначення працівників на посади. Менеджер призначає працівників на певні посади в проєктах.
4. Створення завдань. Менеджер створює нові завдання для працівників.
5. Перевірка завдань. Менеджер перевіряє виконання завдань, що були призначені працівникам.
6. Редагування завдань. Менеджер редагує завдання, змінюючи їхні параметри або призначення.
7. Перегляд та редагування даних про працівника. Менеджер і Адміністратор можуть переглядати та редагувати інформацію про працівників.
8. Перегляд та редагування даних про проєкт. Менеджер і Адміністратор можуть переглядати та редагувати інформацію про проєкти.
9. Перегляд та редагування даних про посаду. Менеджер і Адміністратор можуть переглядати та редагувати інформацію про посади.
10. Формування звітів. Менеджер і Адміністратор формують звіти для аналізу ефективності працівників та проєктів.
11. Перегляд статистики працівників. Менеджер і Адміністратор переглядають статистику працівників для оцінки їхньої продуктивності.
12. Управління користувачами та доступом. Адміністратор керує користувачами, призначає їм права доступу та видаляє їх з системи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 27   |

13. Створення проєктів та призначення менеджерів. Адміністратор створює нові проєкти та призначає відповідальних менеджерів.

Діаграма варіантів використання надає всебічне і наочне уявлення про функціональні можливості системи. Вона є важливим інструментом для аналізу системи, збору вимог та проєктування, оскільки дозволяє визначити та задокументувати варіанти використання.

**Об'єктно-орієнтована структура.** Забезпечує можливість створення модульного дизайну системи, де компоненти представляються як об'єкти з властивостями та методами. Така структура дозволяє ефективно інкапсулювати функціональність всередині об'єктів, що сприяє повторному використанню, модульності та простоті розширення системи. Узагальнена діаграма об'єктно-орієнтованої структури вебдодатку для системи управління проєктами (рис. 2.2) ілюструє взаємозв'язки та взаємодію між різними об'єктами. Кожен об'єкт у цій діаграмі представляє окремий компонент або модуль програмного забезпечення зі своїми властивостями та методами. Діаграма класів є візуальним відображенням архітектури системи, що полегшує розуміння її загальної структури та взаємодії між компонентами. Крім того, вона допомагає визначити можливі місця для розширення та оптимізації системи.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 28   |

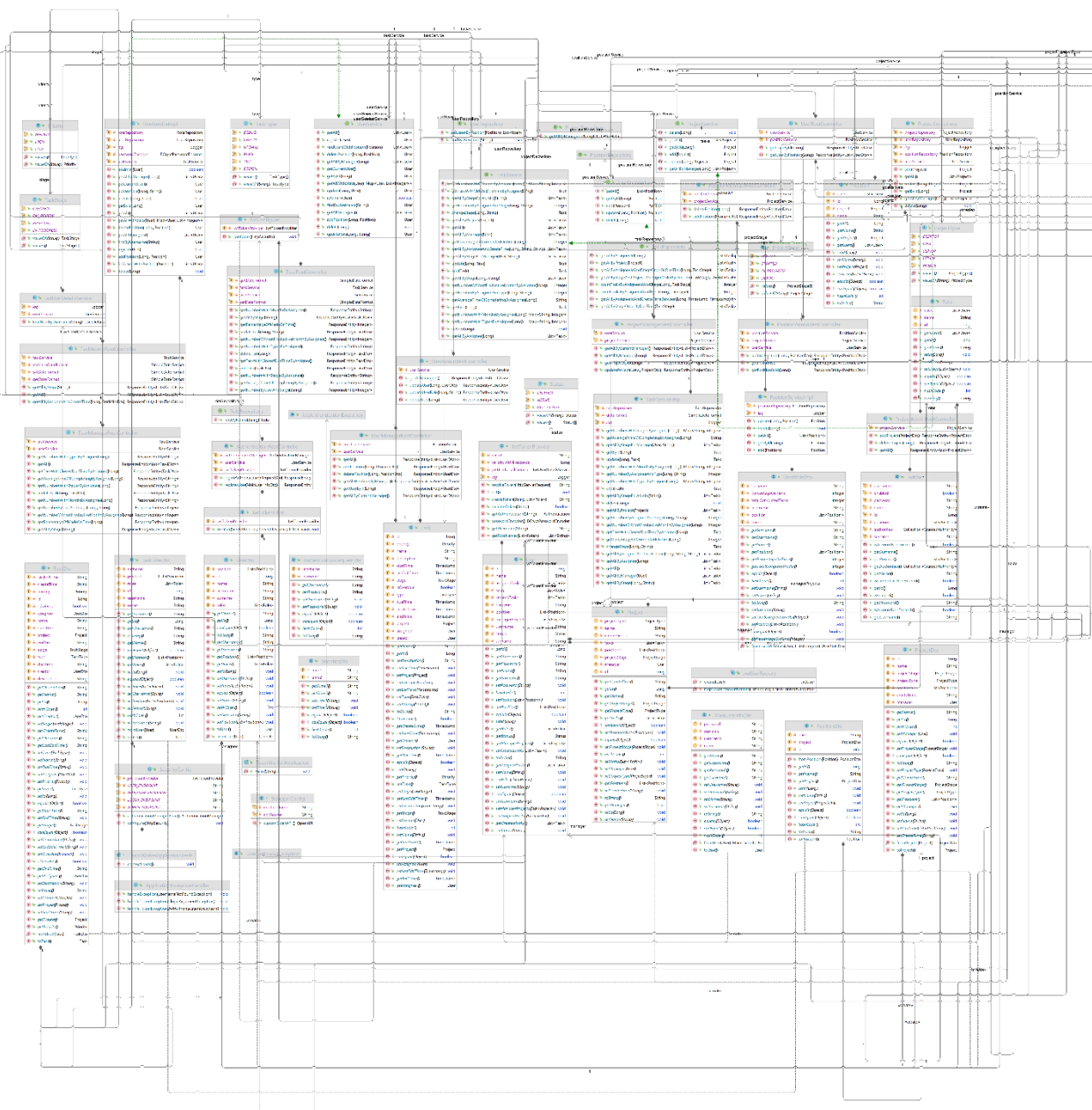


Рисунок 2.2 – Діаграма класів

User – представляє користувача системи з атрибутами, такими як ім'я користувача, пароль, електронна пошта та ролі.

Project – представляє проєкт з деталями, такими як назва проєкту, опис, дата початку, дата завершення та статус.

Task – представляє завдання в межах проєкту, включаючи атрибути, такі як назва завдання, опис, призначений користувач, дата початку, дата завершення та статус.

Team – представляє команду, яка бере участь у проєкті, включаючи назву команди, учасників та пов'язані проєкти.

Role – визначає ролі, які можуть бути призначені користувачам, з атрибутами, такими як назва ролі та дозволи.

Permission – представляє конкретні дозволи, які можуть бути надані ролям, включаючи назву дозволу та опис.

ActivityLog – записує дії, виконані в системі, включаючи тип дії, користувача, мітку часу та опис.

Priority – представляє рівень пріоритету завдань, з атрибутами, такими як назва пріоритету та ранг.

Status – визначає можливі статуси для завдань і проєктів, включаючи назву статусу та опис.

TimeLog – записує витрачений на завдання час, включаючи користувача, завдання, тривалість та дату.

TaskTag – зв'язок між завданням і тегом, включає ідентифікатори завдання та тегу.

Log – журнал дій користувача, включає атрибути, такі як тип дії, деталі дії, дата створення.

## 2.2 Розробка бази даних системи

Управління файлами та інформацією у вебдодатку – це важливий аспект, який вимагає відповідних інструментів для забезпечення ефективності та надійності. Один із таких інструментів – MySQL, система керування базами даних, яка базується на реляційній моделі [26]. Ця модель дозволяє структурувати дані у вигляді таблиць, де кожен рядок представляє окремий екземпляр сутності. Поля цих таблиць можуть містити індекси, що вказують на зв'язки між різними сутностями. Такий підхід дозволяє ефективно зберігати, організовувати та отримувати доступ до даних, що є важливим для роботи вебдодатку.

ER-модель (модель сутності-зв'язок) – це ще один важливий аспект при розробці баз даних [27]. Вона допомагає визначити сутності, їх атрибути та зв'язки між ними. Зазвичай ER-модель реалізується у вигляді баз даних, де

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 30   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

відображаються зв'язки між таблицями. Це дозволяє структурувати дані так, щоб вони були легко зрозумілі та оптимально використовувались в додатку.

Важливо враховувати, що обрана система керування базами даних має забезпечувати надійність, швидкість та зручний доступ до інформації (рис. 2.3).

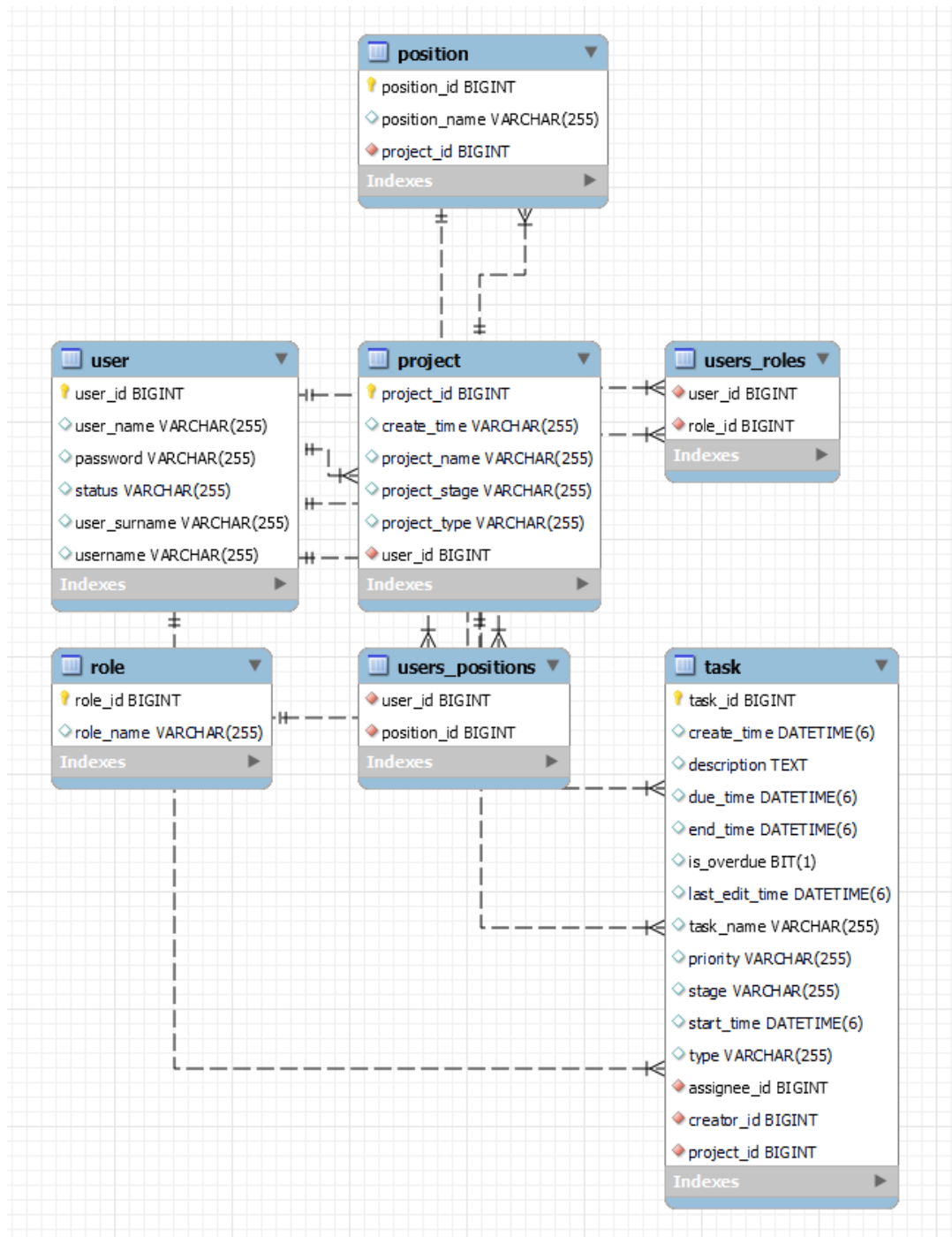


Рисунок 2.3 – ER-модель

Проаналізуємо структуру бази даних для вказаної предметної області.

Основні сутності цієї структури включають:

- 1) користувачі – включає дані про користувачів програми;

- 2) ролі – включає дані про ролі користувачів для забезпечення гнучкого доступу;
- 3) посади – включає дані про посади в проєктах компанії;
- 4) проєкти – включає дані про проєкти компанії;
- 5) завдання – включає детальні дані про завдання для працівників.

Структура полів таблиці User наведено у таблиці 2.1.

Таблиця 2.1 Структура полів таблиці User

| Назва поля | Інформаційний зміст  |
|------------|----------------------|
| user_id    | id користувача       |
| login      | логін користувача    |
| password   | пароль користувача   |
| name       | ім'я користувача     |
| surname    | прізвище користувача |
| status     | статус користувача   |

Ролі – таблиця, що зберігає дані про ролі користувача.

Структура полів таблиці Roles наведено у таблиці 2.2.

Таблиця 2.2 Структура полів таблиці Roles

| Назва поля | Інформаційний зміст |
|------------|---------------------|
| Role_id    | Код ролі            |
| Role_name  | Назва ролі          |

Посади – таблиця, що зберігає дані про посади користувачів.

Структура полів таблиці Position наведено у таблиці 2.3.

Таблиця 2.3 Структура полів таблиці Position

| Назва поля    | Інформаційний зміст |
|---------------|---------------------|
| Position_id   | Код посади          |
| Position_name | Назва посади        |
| Project_id    | Код проєкту         |

Проєкти – таблиця, що зберігає дані про проєкти.

Структура полів таблиці Project наведено у таблиці 2.4.

Таблиця 2.4 Структура полів таблиці Project

| Назва поля    | Інформаційний зміст |
|---------------|---------------------|
| Project_id    | id проєкту          |
| Project_name  | назва проєкту       |
| Create_time   | Дата створення      |
| Project_stage | стадія проєкту      |
| Project_type  | тип проєкту         |

Завдання – таблиця, що зберігає дані про всі завдання компанії.

Структура полів таблиці Task наведено у таблиці 2.5.

Таблиця 2.5 Структура полів таблиці Task

| Назва поля     | Інформаційний зміст                 |
|----------------|-------------------------------------|
| Task_id        | Код завдання                        |
| Create_time    | Час створення                       |
| Description    | Опис                                |
| Due_time       | Дедлайн завдання                    |
| End_time       | Дата закінчення виконання           |
| Is_overdue     | Дедлайн минув                       |
| Last_edit_time | Дата останнього змінення            |
| Task_name      | Назва завдання                      |
| Priority       | Пріоритет виконання                 |
| Stage          | Стадія виконання                    |
| Start_time     | Дата початку виконання завдання     |
| type           | Тип завдання                        |
| Assignee_id    | Код виконавця завдання              |
| Creator_id     | Код працівника що створив завадання |
| Project_id     | Код проєкту                         |

## 2.3 Проєктування та реалізація алгоритмів роботи системи

Система управління проєктами є досить простою для розуміння та має нескладний алгоритм роботи. На рисунку 2.4 представлено схему роботи з платформою.

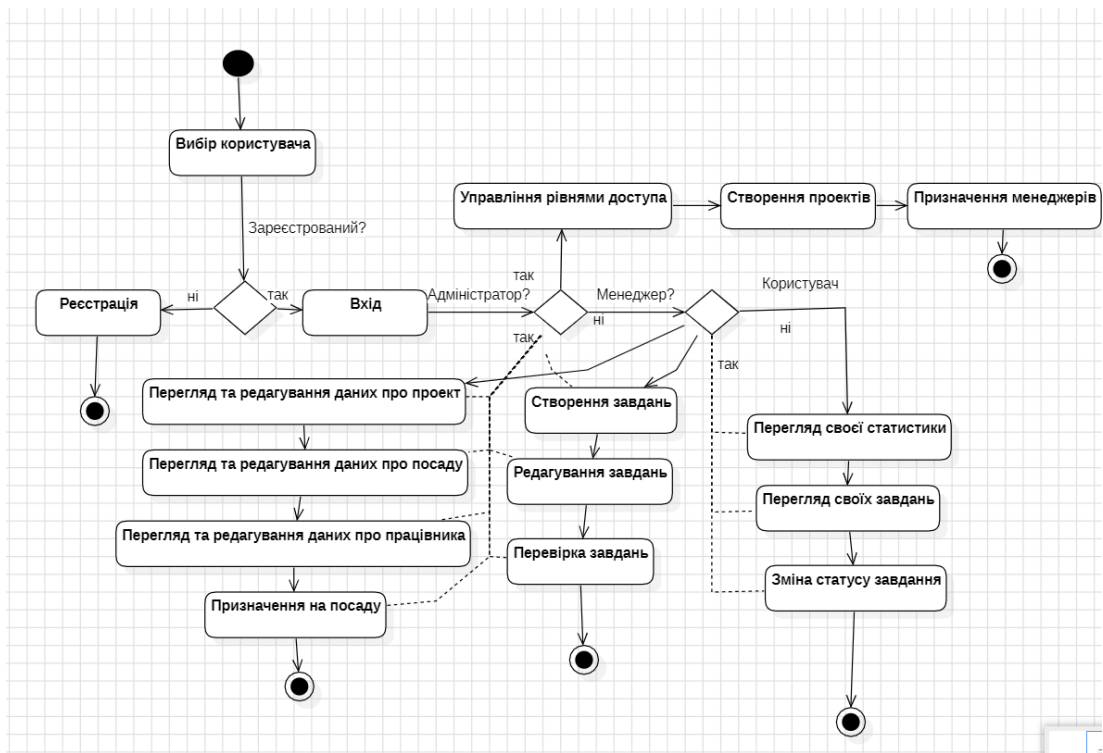


Рисунок 2.4 – Діаграма активності [28]

#### Можливості адміністратору:

- створення задач;
- модифікація задач;
- статистика виконання задач;
- перегляд задач;
- перегляд інформації про працівників та проекти;
- редагування інформації про працівників та проекти;
- управління юзерами та їх доступами;
- перегляд та редагування всієї інформації системи;
- завантаження звітів по всім юзерам і проектам.

#### Можливості користувача:

- перегляд задач;
- статистика виконання задач;
- зміна статусу задач.

#### Можливості менеджера:

- створення задач;

- модифікація задач;
- зміна статусу задач;
- статистика виконання задач;
- перегляд задач;
- перегляд власних задач;
- редагування задач;
- перегляд інформації про працівників та проєкти;
- редагування інформації про працівників та проєкти;
- завантаження звітів.

Обрана стратегія в розробці вебдодатку системи керування проєктами полягає у впровадженні лише функціоналу ролі користувача. Це рішення спрямоване на створення інтуїтивно зрозумілого та простого інтерфейсу для користувачів. Такий підхід дозволяє користувачам швидко оволодіти додатком і почати ефективно працювати з проєктами, уникнувши зайвих ускладнень та перешкод.

## 2.4 Реалізація вебдодатку

Клас `UserAdminRestController` є контролером для адміністрування користувачів, що дозволяє взаємодію з користувачами через REST API. Він обробляє запити на отримання всіх менеджерів, оновлення користувачів та їх ролей, а також видалення користувачів. Запити обробляються за допомогою сервісу `UserService`, і повертають відповідні HTTP статуси на основі успішності або невдачі операцій.

Кожен метод у цьому контролері відповідає певному типу операції: отримання списку користувачів, оновлення інформації про користувача чи його роль, а також видалення користувача з системи. Ці операції обробляються за допомогою відповідних запитів HTTP GET, PUT і DELETE, які інтерпретуються сервісом та відповідно відображають результат клієнту.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 35   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

Отже, UserAdminRestController існує для забезпечення адміністративного доступу до користувачів через API, з використанням стандартних методів HTTP для реалізації різноманітних операцій з користувачами у системі.

```
public class UserAdminRestController {

    private final UserService userService;

    @Autowired
    public UserAdminRestController(UserService userService) {
        this.userService = userService;
    }

    @PutMapping(value = "/update/{id}")
    @Operation(summary = "Оновити користувача")
    public ResponseEntity<UserDto> updateUser(
        @PathVariable(value = "id") Long id,
        @RequestBody UserDto userDto) {

        User user = userService.update(id, userDto.toUser());

        if(user == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        UserDto result = UserDto.fromUser(user);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PutMapping(value = "/update/role/{id}")
    @Operation(summary = "Змінити роль користувача")
    public ResponseEntity<UserDto> updateUserRole(
        @PathVariable(value = "id") Long id,
        @RequestBody String role) {

        User user = userService.updateRole(id, role);

        if(user == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        UserDto result = UserDto.fromUser(user);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }
}
```

Цей клас TaskManagerRestController є частиною серверного додатка, який надає REST API для керування завданнями менеджерів. Він обробляє різноманітні HTTP запити, що стосуються операцій з завданнями, такі як отримання списків завдань за різними критеріями (наприклад, всіх завдань або

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 36   |

за їхнім станом), оновлення, отримання статистики виконання, а також отримання найближчого за терміном завдання для певного користувача. Кожен метод відповідає конкретній операції з обробкою даних, забезпечуючи відповідність HTTP статусів у відповідь на результати операцій.

```
public class TaskManagerRestController {

    private final TaskService taskService;
    private final UserService userService;

    @Autowired
    public TaskManagerRestController(TaskService taskService, UserService
userService) {
        this.taskService = taskService;
        this.userService = userService;
    }

    @GetMapping(value = "get/all")
    @Operation(summary = "Отримати всі завдання доступних проєктів (manager)")
    public ResponseEntity<List<TaskDto>> getAll() throws ParseException {

        List<Task> tasks =
taskService.getAllByManager(userService.getCurrentUser());

        if (tasks == null) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }

        List<TaskDto> result = new ArrayList<>();

        for (Task task : tasks) {
            result.add(TaskDto.fromTask(task));
        }

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PutMapping(value = "/update/{id}")
    @Operation(summary = "Оновити завдання")
    public ResponseEntity<TaskDto> updateTask(
        @PathVariable(value = "id") Long id,
        @RequestBody TaskDto taskDto) throws ParseException {
        Task task = taskService.update(id, taskDto.toTask());

        if (task == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        TaskDto result = TaskDto.fromTask(task);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }
}
```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 37   |

```

    @GetMapping(value = "/get/all/{stage}")
    @Operation(summary = "Отримати всі завдання за стадією для авторизованого менеджера")
    public ResponseEntity<List<TaskDto>> getAllByStage(@PathVariable(value = "stage") String stageName) throws ParseException {
        List<Task> tasks =
        taskService.getAllByStageForManager(userService.getCurrentUser(), stageName);
        List<TaskDto> result = new ArrayList<>();

        for (Task task : tasks) {
            result.add(TaskDto.fromTask(task));
        }

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

```

Цей клас ProjectManagerRestController є частиною серверного додатка, який надає REST API для керування проєктами менеджерів. Він містить методи для отримання списків проєктів, отримання конкретного проєкту за його ідентифікатором, оновлення інформації про проєкт. Кожен метод обробляє відповідний HTTP запит і виконує необхідні операції через сервіси ProjectService та UserService, які взаємодіють з базою даних і забезпечують відповідність даних у форматі DTO (Data Transfer Object), що використовується для передачі даних через мережу [29]. Клас також використовує анотації Swagger для документування API, @CrossOrigin для дозволу запитів з веб-інтерфейсу за іншою адресою та @RestController для вказання, що клас є контролером REST API [30].

```

public class ProjectManagerRestController {

    private final ProjectService projectService;
    private final UserService userService;

    @Autowired
    ProjectManagerRestController(ProjectService projectService, UserService
userService) {
        this.projectService = projectService;
        this.userService = userService;
    }

    @GetMapping(value = "/get/all/{id}")
    @Operation(summary = "Отримати всі проєкти за менеджером")
    public ResponseEntity<List<ProjectDto>> getAllByManager(@PathVariable(value = "id") Long id) {

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 38   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

        List<Project> projects = projectService.getAllByManager(id);
        List<ProjectDto> result =
projects.stream().map(ProjectDto::fromProject).collect(Collectors.toList());
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @GetMapping(value = "/get/all")
    @Operation(summary = "Отримати всі проєкти авторизованого менеджера")
    public ResponseEntity<List<ProjectDto>> getAllByCurrentManager() {
        List<Project> projects =
projectService.getAllByManager(userService.getCurrentUser().getId());
        List<ProjectDto> result =
projects.stream().map(ProjectDto::fromProject).collect(Collectors.toList());
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

```

Цей клас PositionManagerRestController виконує функції керування посадами через REST API. Він надає методи для додавання нових посад (addPosition) і видалення існуючих посад (deletePosition). Клас використовує сервіси PositionService і ProjectService для взаємодії з базою даних та обробки бізнес-логіки.

```

@RestController
@CrossOrigin(origins = "http://localhost:4200/")
@RequestMapping(value = "/api/v1/manager/position")
@Tag(name = "/api/v1/manager/position", description = "Контролер для керування посадами (manager)")
public class PositionManagerRestController {

    private final PositionService positionService;
    private final ProjectService projectService;

    @Autowired
    public PositionManagerRestController(PositionService positionService,
ProjectService projectService) {
        this.positionService = positionService;
        this.projectService = projectService;
    }

    @PostMapping(value = "add")
    @Operation(summary = "Додати посаду")
    public ResponseEntity<PositionDto> addPosition(@RequestBody PositionDto
positionDto) throws ParseException {
        Project project =
projectService.getById(positionDto.getProject().getId());
        if(project == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }
    }

```

```

        positionService.add(positionDto.toPosition());

        return new ResponseEntity<>(positionDto, HttpStatus.CREATED);
    }

    @DeleteMapping(value = "delete/{id}")
    @Operation(summary = "Видалити посаду")
    public ResponseEntity<PositionDto> deletePosition(@PathVariable(name = "id")
    Long id) {
        Position position = positionService.getById(id);
        if(position == null) {
            return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }

        positionService.delete(id);

        return new ResponseEntity<>(HttpStatus.OK);
    }
}

```

## Висновки до розділу 2

Розробка вебдодатку системи управління проєктами включає декілька ключових етапів. Починаючи з аналізу різних сценаріїв використання, розробники створюють структуру платформи для моделювання проєктів та взаємодії їх елементів. Далі розробляється база даних для зберігання інформації, що включає дані користувачів та завдань на сервері додатку. Описується алгоритм роботи з додатком, а також вивчаються послідовні кроки його використання. Також надається огляд функціоналу, який охоплює формування звітів, перегляд статистики, надання різних ролей користувачам та створення нових проєктів.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 40   |

## **РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ**

### **3.1 Порядок встановлення та налаштування параметрів системи**

Для розгортання вашого проекту необхідно спочатку налаштувати сервер для бекенду. Це важливий етап, оскільки він включає конфігурування середовища, запуск необхідних служб і забезпечення стабільного зв'язку з базою даних. Середовище конфігурування повинне включати налаштування операційної системи, встановлення необхідних бібліотек і модулів, а також створення користувачів і визначення їхніх прав доступу.

Першим кроком є встановлення операційної системи на сервер, яка може бути Linux, Windows або інша за вашим вибором. Після цього слід встановити всі необхідні компоненти і пакети, такі як веб-сервер та інші залежності, які потрібні для роботи вашого бекенду. Також необхідне встановлення бази даних MySQL та налаштування параметрів доступу до неї.

Наступним етапом є запуск необхідних служб, які забезпечать роботу бекенду. Це може включати запуск веб-сервера, налаштування параметрів файлу конфігурації, забезпечення з'єднання з базою даних та налаштування правил брандмауера для забезпечення безпеки. Важливо переконатися, що всі служби працюють коректно і злагоджено.

Після налаштування середовища і запуску служб слід забезпечити стабільний зв'язок з базою даних. Це включає налаштування параметрів підключення, перевірку можливості виконання запитів до бази даних та налаштування механізмів резервного копіювання даних для запобігання їх втраті. Важливо забезпечити швидку і надійну роботу системи, щоб вона могла обробляти запити від мобільного застосунку та фронтенду.

Після успішного налаштування і запуску сервера можна переходити до запуску фронтенду. Фронтенд повинен бути налаштований таким чином, щоб коректно взаємодіяти з бекендом і забезпечувати зручний інтерфейс для

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 41   |

користувачів. Це включає перевірку всіх функцій і елементів інтерфейсу, забезпечення адаптивності для різних пристроїв і браузерів, а також проведення тестування на наявність можливих помилок і проблем.

Завершальним етапом є звернення до сторінки серверу у браузері. Це дозволить перевірити роботу системи, переконатися в коректності взаємодії між фронтендом і бекендом, а також почати використовувати проект у повному обсязі. При цьому важливо забезпечити належний рівень безпеки, захистити дані користувачів і запобігти можливим загрозам.

Отже, розгортання проекту включає декілька важливих етапів: налаштування сервера для бекенду, запуск необхідних служб, забезпечення стабільного зв'язку з базою даних, налаштування і запуск фронтенду, а також тестування і перевірку системи. Це забезпечить швидку, надійну і безпечну роботу вашого проекту.

### **3.2 Структура інтерфейсу вебдодатку системи управління проектами**

Модульна структура програми має форму дерева, де вузли представляють програмні модулі, а спрямовані дуги вказують на статичну підлеглість модулів. Якщо в тексті одного модуля є посилання на інший модуль, їх на структурній схемі з'єднує дуга, яка виходить з першого і входить у другий модуль. Інакше кажучи, кожен модуль може звертатися до своїх підлеглих модулів. Модульна структура програмної системи повинна включати не лише структурну схему, а й сукупність специфікацій модулів, що складають цю систему.

Головний модуль забезпечує виконання функції верхнього рівня; він керує виконанням нижчестоящих функцій, які реалізують підлегли модулі. При визначенні набору модулів, що реалізують функції конкретного алгоритму, потрібно враховувати наступне:

1. Модуль викликається на виконання вищестоящим по ієрархії модулем і, завершивши роботу, повертає йому управління.
2. Прийняття основних рішень в алгоритмі відноситься на максимально високий по ієрархії рівень.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 42   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

3. Якщо в різних частинах алгоритму використовується одна й та сама функція, вона оформлюється в окремий модуль, який буде викликатися за необхідністю.

Склад, призначення і характер використання програмних модулів значною мірою залежать від інструментальних засобів.

Модульна структура системи зображена на рисунку 3.1.

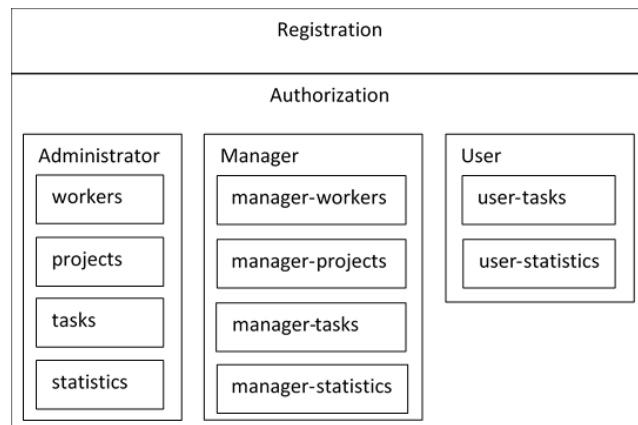


Рисунок 3.1 – Модульна структура системи

У вебдодатках взаємодія між введенням і виведенням даних на серверній частині здійснюється через вебсторінки. Використання вебсторінок для цієї мети є більш зручним і допомагає уникнути помилок при роботі з базою даних. Вебсторінка дозволяє об'єднати всю необхідну інформацію в одному місці, позбавляючи користувача від необхідності шукати її у різних джерелах.

Сторінка /login містить стартову форму проєкту, через яку користувач може увійти в програму, ввівши свій логін і пароль. Якщо користувач не зареєстрований, він може зареєструватися за допомогою форми на сторінці /register.

Сторінка /login зображена на рисунку 3.2.

**Авторизація**

Введіть логін \*

admin@test.com

Введіть пароль \*

.....

**Увійти**

[Зареєструватися](#)

Рисунок 3.2 – Вигляд сторінки /login

Сторінка /register зображена на рисунку 3.3.

**Реєстрація нового користувача**

[Вже зареєстровані? Увійти](#)

Ім'я \*

Прізвище \*

Логін \*

Пароль \*

**Реєстрація**

Рисунок 3.3 – Вигляд сторінки /register

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          | 44   |

Сторінка "admin/users" надає можливість переглядати та редагувати детальну інформацію про всіх працівників компанії. Вона забезпечує зручний інтерфейс для управління персоналом, де ви можете змінювати особисті дані працівників, їхні контактні дані, посади та інші важливі атрибути.

Крім того, на цій сторінці є функціонал для додавання працівника до проєкту. Це дозволяє швидко та легко призначати працівників на різні завдання та проєкти, забезпечуючи ефективну координацію робочих процесів.

Також, сторінка "admin/users" надає можливість формувати звіти. Ви можете створювати різноманітні звіти для аналізу роботи працівників, їхньої залученості до проєктів, продуктивності та інших аспектів діяльності.

Сторінка admin/users зображена на рисунку 3.4.

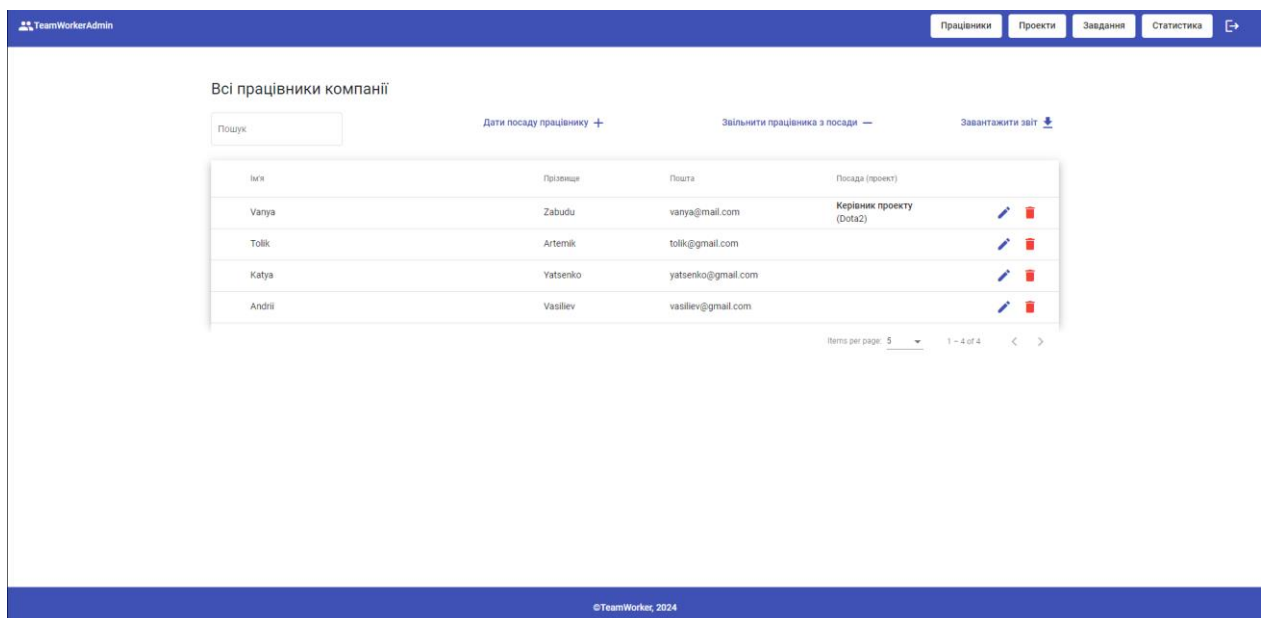


Рисунок 3.4 – Вигляд сторінки admin/users.

Сторінка admin/projects дозволяє переглядати, додавати та редагувати проєкти компанії. Вона забезпечує зручний інтерфейс для управління всіма аспектами проєктів. Також на цій формі можна додати або видалити посаду на конкретний проєкт, що дозволяє гнучко управляти командним складом. Крім того, доступна можливість перегляду детальних звітів, які містять всю необхідну інформацію про стан проєкту, його прогрес та ключові показники ефективності.

Сторінка admin/projects зображена на рисунку 3.5.

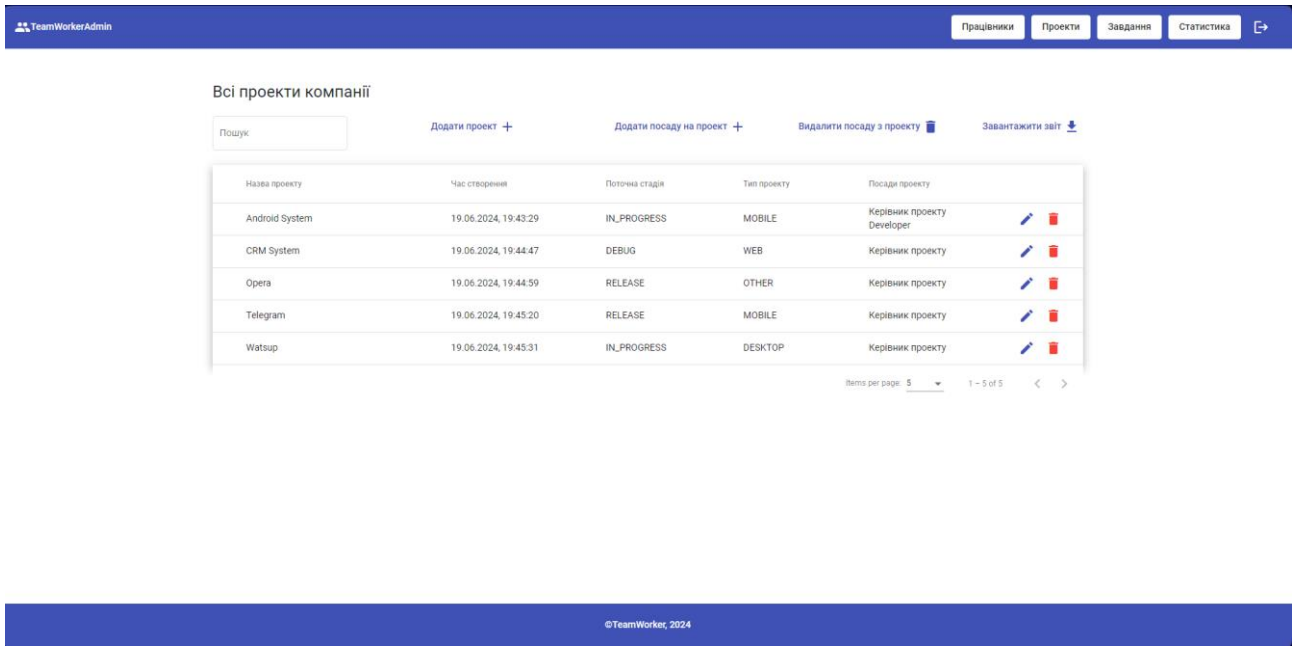


Рисунок 3.5 – Вигляд сторінки admin/users

Форма admin/tasks дозволяє переглядати всі завдання, створювати звіти, створювати нові завдання, редагувати існуючі, а також приймати виконані завдання працівників. Ця форма забезпечує зручний інтерфейс для повного управління завданнями, надаючи можливість швидко і ефективно контролювати робочий процес. Крім того, завдяки функції створення звітів, можна аналізувати продуктивність команди та відстежувати прогрес виконання завдань, що сприяє підвищенню ефективності роботи та досягненню поставлених цілей.

Екранна форма admin/tasks зображена на рисунку 3.6.

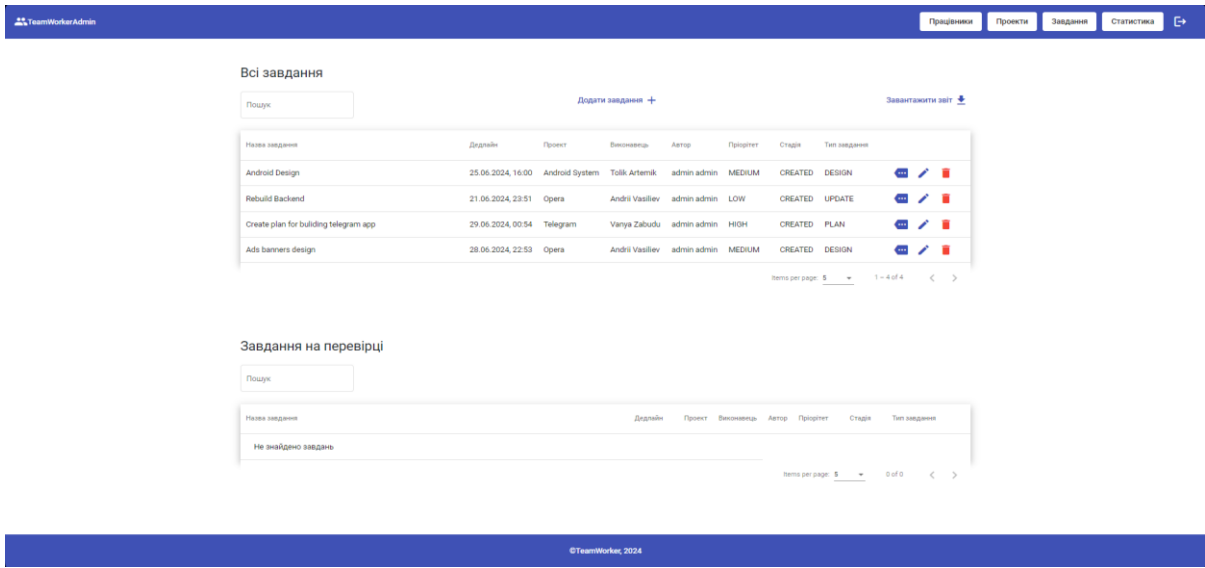


Рисунок 3.6 – Вигляд сторінки admin/tasks.

Сторінка admin/statistics дозволяє переглядати детальну інформацію про завдання, які виконує конкретний працівник. На цій сторінці можна побачити, які завдання були призначені працівнику, їх статус, терміни виконання та рівень завершеності. Крім того, доступні аналітичні дані, які допомагають оцінити продуктивність працівника, виявити можливі проблеми та визначити області для покращення. Завдяки цій функціональності, керівники можуть ефективніше моніторити роботу своїх співробітників та приймати обґрунтовані рішення для оптимізації робочого процесу.

Сторінка admin/statistics зображена на рисунку 3.7 та 3.8.

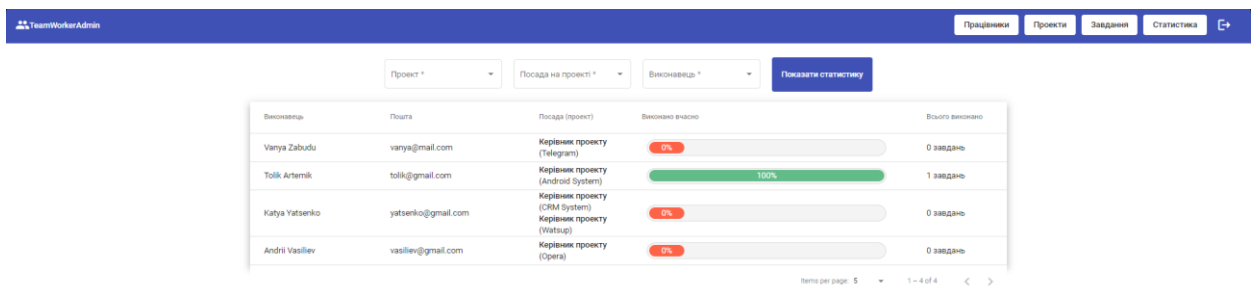


Рисунок 3.7 – Вигляд сторінки admin/statistics.

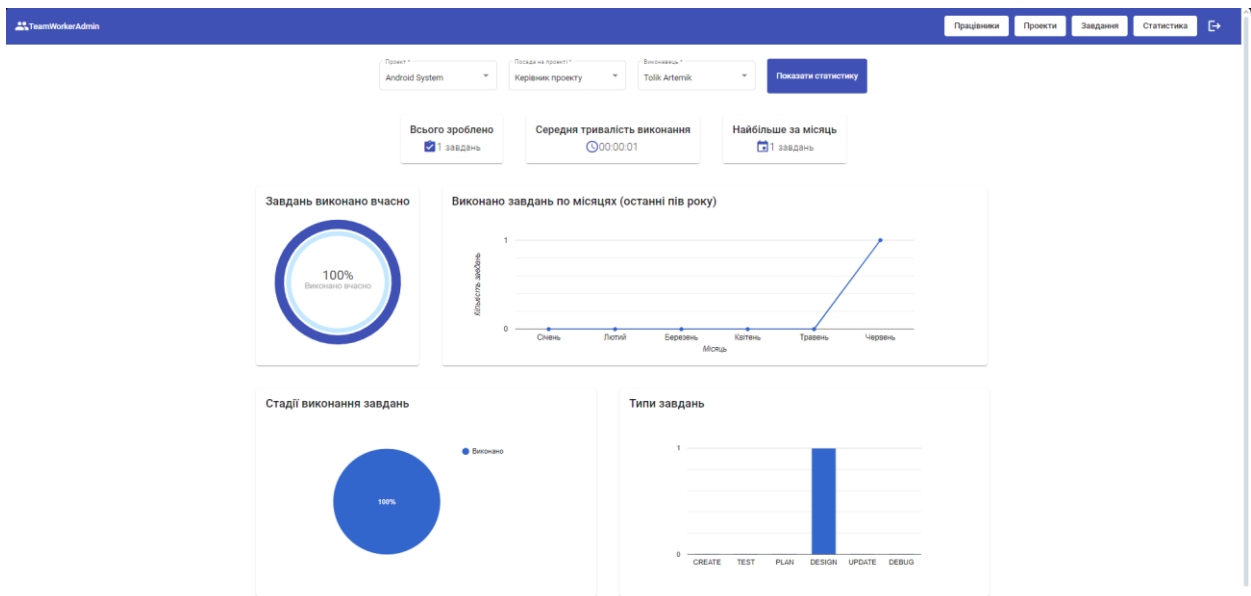


Рисунок 3.8 – Вигляд сторінки admin/statistics.

На сторінці user/tasks працівник може побачити список призначених йому завдань, їх пріоритети, терміни виконання та статус. Крім того, він може редагувати деталі завдання та оновлювати статус виконання. Ця сторінка забезпечує зручний інтерфейс для управління завданнями, допомагаючи працівнику організовувати свою роботу, слідкувати за дедлайнами та ефективно виконувати поставлені перед ним завдання.

Сторінка user/tasks зображена на рисунку 3.9.

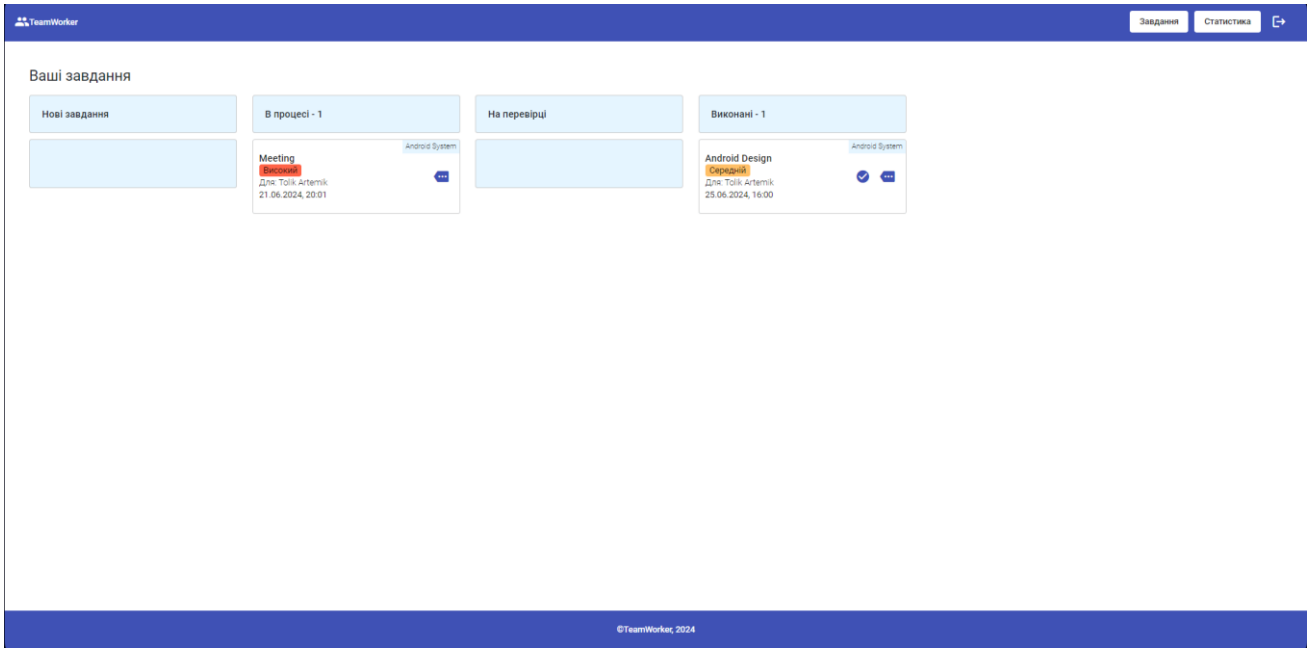


Рисунок 3.9 – Вигляд сторінки user/tasks

Сторінка user/statistics дозволяє працівнику переглядати детальну особисту статистику, що надає важливу інформацію про його продуктивність та ефективність. На цій сторінці працівник може бачити загальну кількість виконаних завдань, поточний прогрес по активних завданнях, а також розподіл часу, витраченого на різні проєкти та завдання. Крім того, доступні графіки та діаграми, що візуалізують продуктивність за певний період, допомагаючи виявити сильні та слабкі сторони в роботі.

Сторінка user/statistics зображена на рисунку 3.10.

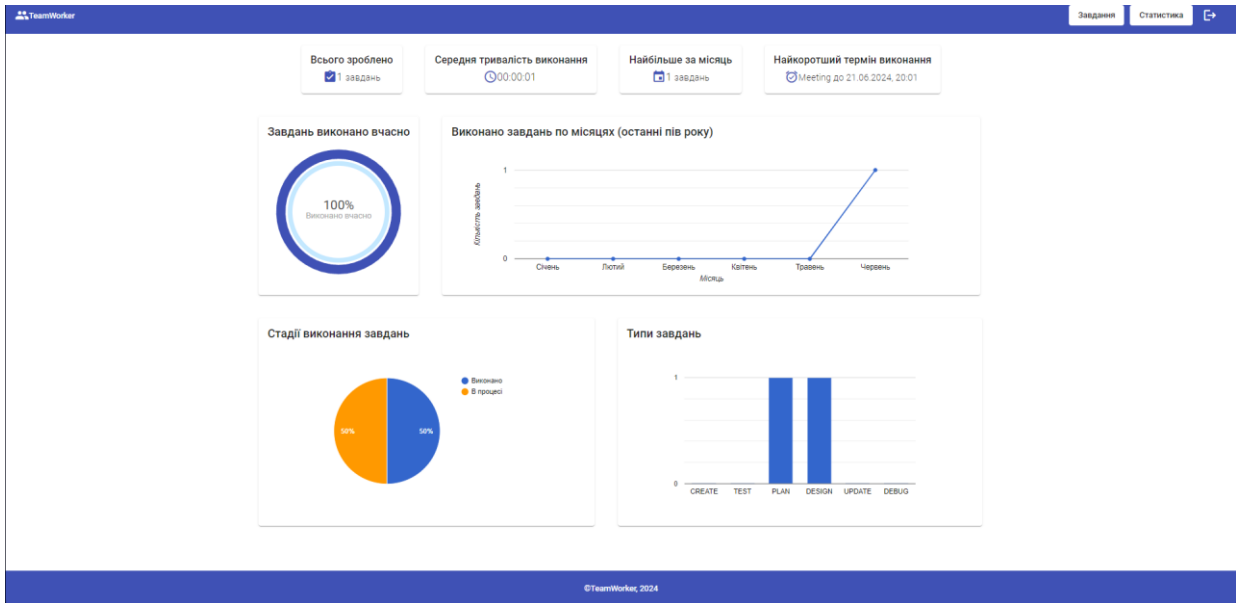


Рисунок 3.10 – Вигляд сторінки user/statistics

Сторінка manager/users дозволяє менеджеру працівників компанії або своїх проєктів та дозволяє наймати або звільняти працівників.

Сторінка manager/workers зображена на рисунку 3.11.

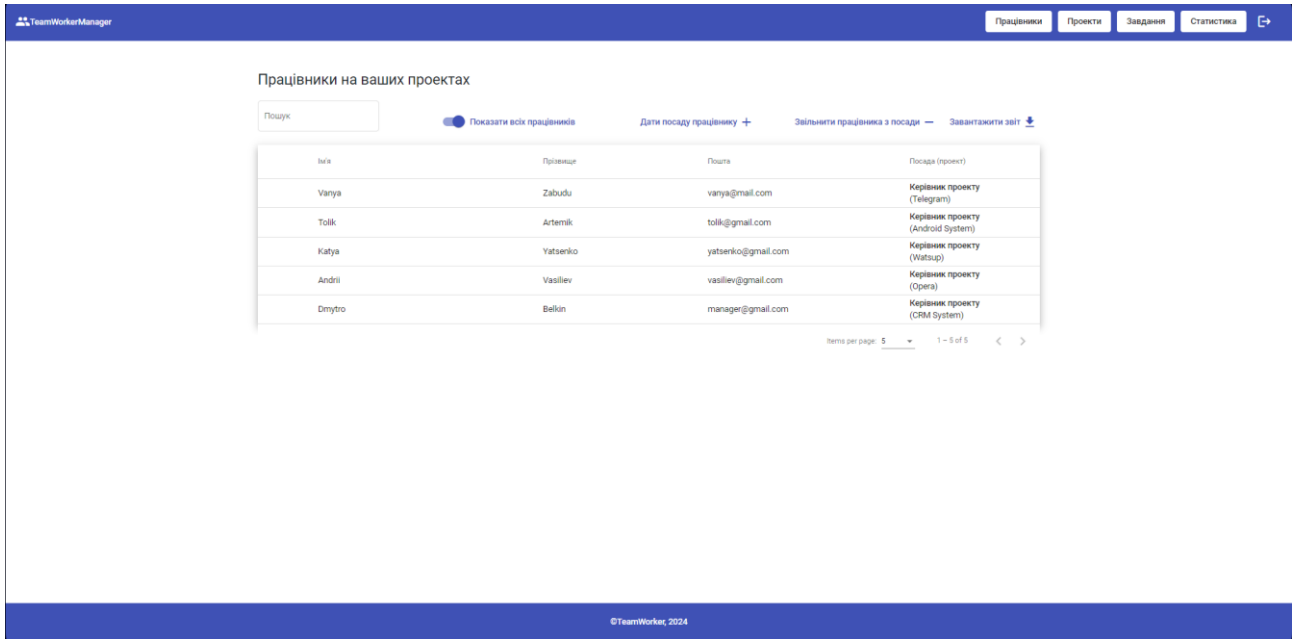


Рисунок 3.11 – Вигляд сторінки manager/workers

Сторінка manager/projects дозволяє менеджеру переглядати свої проєкти, редагувати їх та додавати/видаляти посади.

Сторінка manager/projects зображена на рисунку 3.12.

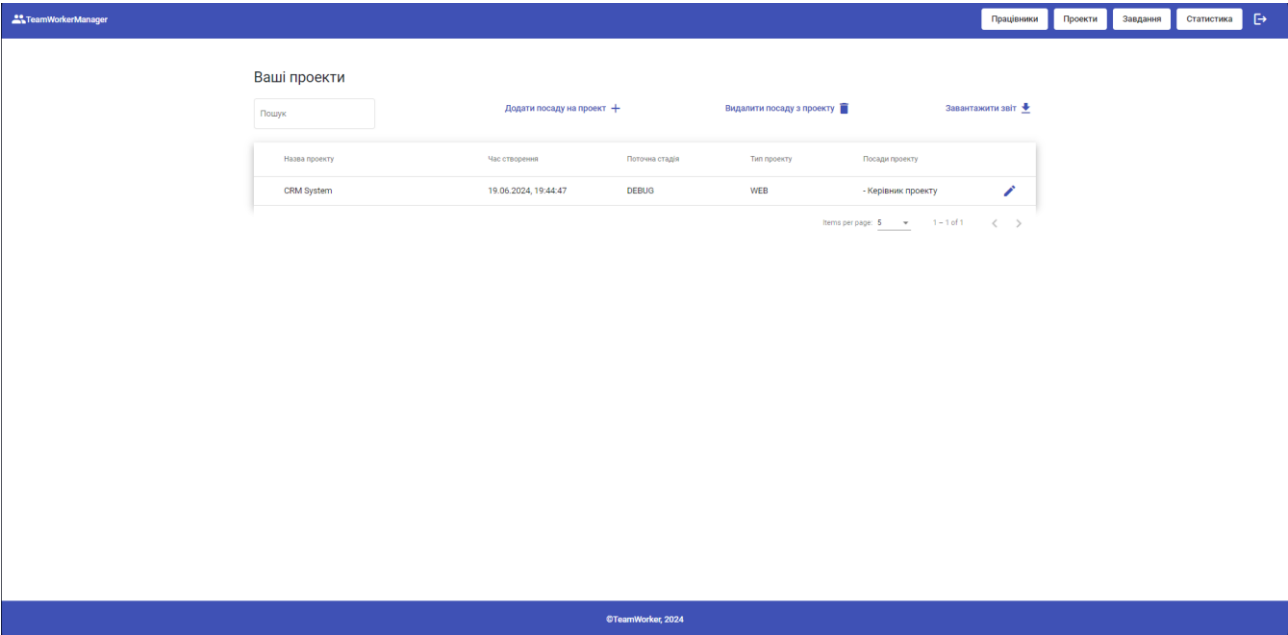


Рисунок 3.12 – Вигляд сторінки manager/projects

Сторінка manager/statistics дозволяє менеджеру переглядати загальну статистику та персональну статистику кожного працівника з своїх проєктів.

Сторінка manager/statistics зображена на рисунках 3.13 і 3.14.

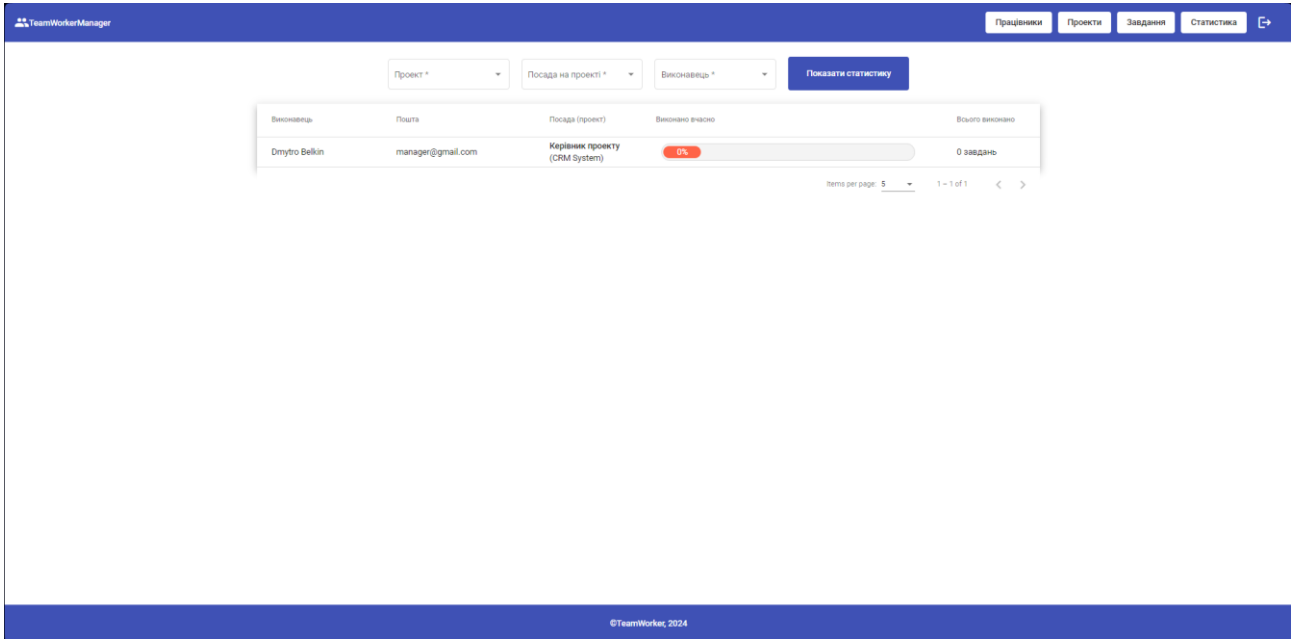


Рисунок 3.13 – Вигляд сторінки manager/statistics

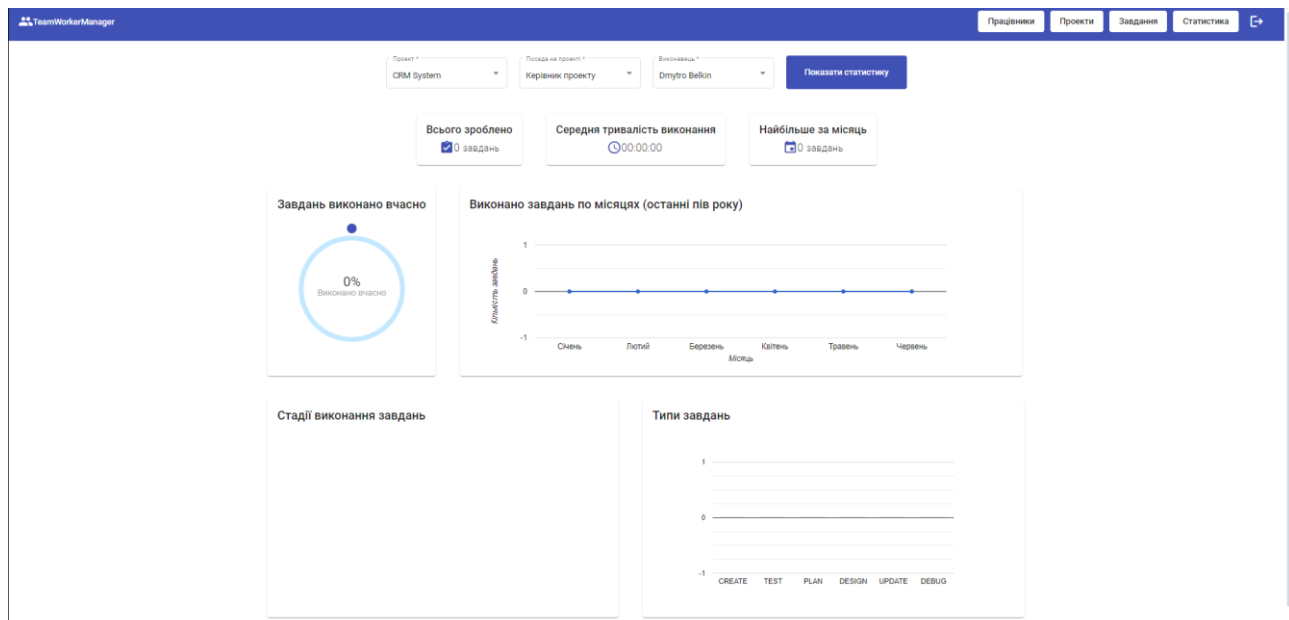


Рисунок 3.14 – Вигляд сторінки manager/statistics

Система також дозволяє формувати звітність по певним параметрам.

Звіт є формою спеціального типу, призначеною для виведення підсумкової інформації на друк або на екран. Звіт може містити елементи управління, результати підсумкових розрахунків та аналізу даних, графічні компоненти. Зокрема вони призначені для створення документів, які надалі можуть бути роздруковані або включені в документ іншого застосування.

У даному дипломному проєкті було реалізовано 4 звіти. Кожен звіт можна сформувати на відповідній до його вмісту сторінки натиснувши кнопку “Завантажити звіт”. Звіти генеруються в зручному для друку .pdf форматі

Звіт по всіх користувачам зображений на рисунку 3.15.

### Звіт по працівникам

| Ім'я   | Прізвище | Пошта              | Посада (проект)                   |
|--------|----------|--------------------|-----------------------------------|
| Vanya  | Zabudu   | vanya@mail.com     | Керівник проекту (Telegram)       |
| Tolik  | Artemik  | tolik@gmail.com    | Керівник проекту (Android System) |
| Katya  | Yatsenko | yatsenko@gmail.com | Керівник проекту (Watsup)         |
| Andrii | Vasiliev | vasiliev@gmail.com | Керівник проекту (Opera)          |
| Dmytro | Belkin   | manager@gmail.com  | Керівник проекту (CRM System)     |

Дата видачі звіту: 20.06.2024, 01:19:08,  
Створив: Dmytro Belkin

Рисунок 3.15 – Вигляд звіту по всіх користувачах

Звіт по проектам зображений на рисунку 3.16.

### Звіт по проектам

| Назва проекту  | Час створення        | Поточна стадія | Тип проекту | Посади проекту                |
|----------------|----------------------|----------------|-------------|-------------------------------|
| Android System | 19.06.2024, 19:43:29 | IN_PROGRESS    | MOBILE      | Керівник проекту<br>Developer |
| CRM System     | 19.06.2024, 19:44:47 | DEBUG          | WEB         | Керівник проекту              |
| Opera          | 19.06.2024, 19:44:59 | RELEASE        | OTHER       | Керівник проекту              |
| Telegram       | 19.06.2024, 19:45:20 | RELEASE        | MOBILE      | Керівник проекту              |
| Watsup         | 19.06.2024, 19:45:31 | IN_PROGRESS    | DESKTOP     | Керівник проекту              |

Дата видачі звіту: 20.06.2024, 01:20:17,

Створив: admin admin

Рисунок 3.16 – Вигляд звіту по проектам

Звіт по завданням зображений на рисунку 3.17.

### Звіт по завданням

| Назва завдання                        | Опис | Дата створення       | Дедлайн           | Дата початку виконання | Дата закінчення      | Проект         | Виконавець      | Автор       | Пріоритет | Стадія      | Тип завдання |
|---------------------------------------|------|----------------------|-------------------|------------------------|----------------------|----------------|-----------------|-------------|-----------|-------------|--------------|
| Android Design                        |      | 19.06.2024, 19:47:44 | 25.06.2024, 16:00 | 19.06.2024, 19:53:12   | 19.06.2024, 19:53:14 | Android System | Tolik Artemik   | admin admin | MEDIUM    | RELEASED    | DESIGN       |
| Rebuild Backend                       |      | 19.06.2024, 19:48:25 | 21.06.2024, 23:51 |                        |                      | Opera          | Andrii Vasiliev | admin admin | LOW       | CREATED     | UPDATE       |
| Create plan for building telegram app |      | 19.06.2024, 19:49:10 | 29.06.2024, 00:54 |                        |                      | Telegram       | Vanya Zabudu    | admin admin | HIGH      | CREATED     | PLAN         |
| Ads banners design                    |      | 19.06.2024, 19:50:48 | 28.06.2024, 22:53 |                        |                      | Opera          | Andrii Vasiliev | admin admin | MEDIUM    | CREATED     | DESIGN       |
| Meeting                               |      | 19.06.2024, 20:01:30 | 21.06.2024, 20:01 | 19.06.2024, 20:01:36   |                      | Android System | Tolik Artemik   | admin admin | HIGH      | IN_PROGRESS | PLAN         |

Дата видачі звіту: 20.06.2024, 01:21:08,

Створив: admin admin

Рисунок 3.17 – Вигляд звіту по завданням

## 3.3 Тестування роботи програмного комплексу

Детальне тестування вебдодатків дозволяє виявити і усунути недоліки на ранніх стадіях розробки, що суттєво зменшує витрати на їх виправлення на пізніших етапах. Це також гарантує стабільну і безпечну роботу вебдодатка на різноманітних пристроях і в різних браузерах, що є критично важливим для забезпечення позитивного користувацького досвіду.

Тестування продуктивності є ключовим аспектом успіху вебдодатка. Воно дозволяє визначити, як додаток справляється з високим навантаженням, оцінити його швидкість відгуку та ефективність використання серверних ресурсів. Важливо, щоб вебдодаток не тільки функціонував бездоганно, але й забезпечував високу продуктивність навіть при значних навантаженнях. Це допомагає

|      |      |          |        |      |                          |  |  |  |  |      |
|------|------|----------|--------|------|--------------------------|--|--|--|--|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ |  |  |  |  | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |  |  |  |  | 52   |

підтримувати лояльність користувачів і знижує ризик відмови від використання додатка через низьку продуктивність.

Підсумовуючи, тестування є необхідним елементом успішного вебдодатка. Воно сприяє створенню надійного, безпечного та зручного продукту, який відповідає найвищим стандартам якості. Інвестування часу та ресурсів у тестування є критичним для зменшення кількості помилок, підвищення задоволеності користувачів та підтримки доброї репутації розробника. Таблиця тестування роботи сайту наведено у таблиці 3.1.

Таблиця 3.1

### Тестування роботи сайту

| Назва тестування              | Кроки тестування                                                                                             | Тестові дані                                 | Очікуваний результат                                                    | Фактичний результат                         | Pass / Fail |
|-------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------|-------------|
| 1.                            | 2.                                                                                                           | 3.                                           | 4.                                                                      | 5.                                          | 6.          |
| Тестування логіну користувача | 1.Відкрити сторінку входу<br>2.Ввести логін<br>3.Ввести пароль<br>4.Натиснути кнопку "Увійти"                | Логін:<br>admin@test.com<br>Пароль:admin     | Користувач успішно авторизується і перенаправляє ся на головну сторінку | Фактичний результат збігається з очікуваним | Pass        |
| Тестування реєстрації         | 1.Відкрити сторінку реєстрації<br>2.Ввести дані нового користувача.<br>3.Натиснути кнопку "Зареєструвати ся" | Логін:<br>andrii@gmail.com<br>Пароль: andrii | Користувач успішно реєструється і перенаправляє ся на сторінку логіну   | Фактичний результат збігається з очікуваним | Pass        |
| Тестування додавання завдання | 1.Відкрити меню<br>2.Заповнити всі пункти<br>3.Натиснути кнопку "Додати"                                     | Немає тестових даних.                        | Створюється нове завдання                                               | Фактичний результат збігається з очікуваним | Pass        |

Продовження таблиці 3.1.

|                                                  |                                                                                                      |                       |                                                                              |                                             |      |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------|-----------------------|------------------------------------------------------------------------------|---------------------------------------------|------|
| Тестування виводу статистики певного користувача | 1.Відкрити сторінку "Статистика"<br>2.Обрати користувача<br>3.Натиснути кнопку "Показати статистику" | Немає тестових даних. | Відкривається нова сторінка в якій зображена повна інформація про працівника | Фактичний результат збігається з очікуваним | Pass |
| Тестування додавання проєкту                     | 1.Відкрити сторінку "Проекти"<br>2.Заповнити всі пункти<br>3.Натиснути кнопку "Додати"               | Немає тестових даних. | Створюється новий проєкт                                                     | Фактичний результат збігається з очікуваним | Pass |
| Формування звітів                                | 1.Відкрити сторінку "Завдання"<br>2. Натиснути кнопку завантажити звіт                               | Немає тестових даних. | Відкривається PDF файл в якому зображена повна інформація про завдання       | Фактичний результат збігається з очікуваним | Pass |
| Тестування зміни статусу завдання                | 1.Відкрити сторінку "Завдання"<br>2. Перенести завдання з пункту «В процесі» на пункт «На перевірці» | Немає тестових даних. | Завдання з'являється в пункті "На перевірці"                                 | Фактичний результат збігається з очікуваним | Pass |

### Висновки до розділу 3

Розробка веб-застосунку для системи управління проєктами пройшла успішно завдяки чіткій структурі процесу та дотриманню найкращих практик програмування. У процесі розробки була врахована низка важливих аспектів, що забезпечили високу якість кінцевого продукту. Особлива увага приділялася створенню інтуїтивного та зручного інтерфейсу для забезпечення зручної навігації та легкого доступу до функцій системи управління проєктами.

Ключовим було забезпечення сумісності з різними пристроями та браузерами, що досягалося за допомогою адаптивного дизайну та оптимізації для ефективного використання ресурсів. Також були впроваджені механізми безпеки для захисту даних користувачів, що є критично важливим для конфіденційності та надійності системи

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 55   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## ВИСНОВКИ

В результаті виконання бакалаврської кваліфікаційної роботи була створена автоматизована система управління проектами на основі клієнт-серверної архітектури. У роботі проведено аналіз поставленої задачі та досліджено специфіку предметної області. Також було опрацьовано теоретичний матеріал, викладено суть та теоретичні основи досліджуваної проблеми, обрано інструментальні засоби, визначено основні завдання проєкту, а також проведено огляд сучасної літератури, присвяченої розглядуваним питанням.

У ході роботи описано процес проєктування та розробки системи, структуру даних, інтерфейс користувача та основні алгоритми. Також були представлені інструкції з правильного розгортання проєкту.

Створене програмне забезпечення відповідає сучасним вимогам і тенденціям щодо швидкодії, надійності, доступності і повністю реалізує поставлені цілі.

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 56   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зима Д.А., Савіцький Р. С. Важливість веб-систем CRM для ефективної роботи компаній. Подано до друку, 13-17 травня 2024 р. Житомир.
2. Airtable. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.airtable.com/>
3. Asana. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://app.asana.com>
4. Jira. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.atlassian.com/software/jira>
5. Spring Framework. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://docs.spring.io/spring-framework/reference/index.html>
6. Java. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://docs.oracle.com/en/java/>
7. Angular. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://angular.dev/overview>
8. Spring Boot. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://docs.spring.io/spring-boot/index.html>
9. Spring Security. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://docs.spring.io/spring-security/reference/index.html>
10. Spring MVC. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://docs.spring.io/spring-framework/reference/web/webmvc.html>
11. HTML. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://developer.mozilla.org/en-US/docs/Web/HTML>
12. CSS. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://developer.mozilla.org/en-US/docs/Web/CSS>

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 57   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

13. HTTP. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://developer.mozilla.org/en-US/docs/Web/HTTP>
14. RESTful API. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://restfulapi.net>
15. JSON. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.json.org/json-en.html>
16. Java Script. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.json.org/json-en.html>
17. XML. [Електронний ресурс] – Режим доступу до ресурсу:  
[https://developer.mozilla.org/en-US/docs/Web/XML/XML\\_introduction](https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction)
18. Angular CLI. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://angular.dev/tools/cli/>
19. Мультиплатформенність. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://cyberpedia.reasonlabs.com/EN/cross-platform%20compatibility.html>
20. Garbage Collector. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>
21. Hibernate. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://hibernate.org/orm/documentation/6.5/>
22. IntelliJ IDEA. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.jetbrains.com/help/idea/managing-plugins.html>
23. Webstorm. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.jetbrains.com/help/webstorm/managing-plugins.html>
24. Database Management Systems. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://alison.com/learning/courses/236/database-management-systems>

25. MySQL Workbench. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://dev.mysql.com>
26. UML Class Diagram Tutorial. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
27. MySQL. [Електронний ресурс] – Режим доступу до ресурсу:  
[https://docs.oracle.com/cd/E17952\\_01/index.html](https://docs.oracle.com/cd/E17952_01/index.html)
28. Introduction of ER Model. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.geeksforgeeks.org/introduction-of-er-model/>
29. Activity Diagram Tutorial. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://online.visual-paradigm.com/diagrams/tutorials/activity-diagram-tutorial/>
30. Swagger. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://swagger.io/docs/>

# ДОДАТКИ

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 60   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## Реалізація функціоналу адміністратора

## PositionAdminRestController:

```

package com.teamworker.rest.admin;

import com.teamworker.dtos.PositionDto;
import com.teamworker.models.Position;
import com.teamworker.models.Project;
import com.teamworker.services.PositionService;
import com.teamworker.services.ProjectService;
import com.teamworker.services.UserService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.text.ParseException;
import java.util.List;
import java.util.stream.Collectors;

@RestController
@CrossOrigin(origins = "http://localhost:4200/")
@RequestMapping(value = "/api/v1/admin/positions")
@Tag(name = "/api/v1/admin/positions", description = "Контролер для керування посадами")
public class PositionAdminRestController {

    private final PositionService positionService;
    private final ProjectService projectService;
    private final UserService userService;

    @Autowired
    public PositionAdminRestController(PositionService positionService,
    UserService userService, ProjectService projectService) {
        this.positionService = positionService;
        this.userService = userService;
        this.projectService = projectService;
    }

    @GetMapping(value = "get/all")
    @Operation(summary = "Отримати всі посади")
    public ResponseEntity<List<PositionDto>> getPositions() {
        List<Position> positions = positionService.getAll();

        if(positions.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }
        List<PositionDto> result =
positions.stream().map(PositionDto::fromPosition).collect(Collectors.toList()
);
    }

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 61   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @GetMapping(value = "get/{id}")
    @Operation(summary = "Отримати посаду за ідентифікатором")
    public ResponseEntity<PositionDto> getPositionById(@PathVariable(name =
"id") Long id) {
        Position position = positionService.getById(id);
        if(position == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }
        PositionDto result = PositionDto.fromPosition(position);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PutMapping(value = "update/{id}")
    @Operation(summary = "Оновити посаду")
    public ResponseEntity<PositionDto> updatePosition(
        @PathVariable(name = "id") Long id,
        @RequestBody PositionDto positionDto) throws ParseException {
        Position position = positionService.update(id,
positionDto.toPosition());

        if(position == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        positionDto = PositionDto.fromPosition(position);

        return new ResponseEntity<>(positionDto, HttpStatus.OK);
    }
}

```

ДОДАТОК Б

ProjectAdminRestController:

```

package com.teamworker.rest.admin;

import com.teamworker.dtos.PositionDto;
import com.teamworker.dtos.ProjectDto;
import com.teamworker.models.Position;
import com.teamworker.models.Project;
import com.teamworker.models.User;
import com.teamworker.services.ProjectService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.text.ParseException;

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 62   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

import java.util.List;
import java.util.stream.Collectors;

@RestController
@CrossOrigin(origins = "http://localhost:4200/")
@RequestMapping(value = "/api/v1/admin/projects")
@Tag(name = "/api/v1/admin/projects", description = "Контролер для керування проектами (admin)")
public class ProjectAdminRestController {

    private final ProjectService projectService;

    @Autowired
    ProjectAdminRestController(ProjectService projectService) {
        this.projectService = projectService;
    }

    @GetMapping(value = "/get/all")
    @Operation(summary = "Отримати всі проекти")
    public ResponseEntity<List<ProjectDto>> getAll() {
        List<Project> projects = projectService.getAll();
        if(projects == null) {
            return new ResponseEntity<>(HttpStatus.OK);
        }
        List<ProjectDto> result =
projects.stream().map(ProjectDto::fromProject).collect(Collectors.toList());
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PostMapping(value = "/add")
    @Operation(summary = "Додати проект")
    public ResponseEntity<ProjectDto> addProject(@RequestBody ProjectDto
projectDto) {
        Project project = projectService.add(projectDto.toProject());
        if(project == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }
        ProjectDto result = ProjectDto.fromProject(project);

        return new ResponseEntity<>(result, HttpStatus.CREATED);
    }

    @DeleteMapping(value = "/delete/{id}")
    @Operation(summary = "Видалити проект")
    public ResponseEntity<PositionDto> deleteProject(@PathVariable(value =
"id") Long id) {
        if(projectService.getById(id) == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }
        projectService.delete(id);
        return new ResponseEntity<>(HttpStatus.OK);
    }
}

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 63   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

## TaskAdminRestController:

```

package com.teamworker.rest.admin;

import com.teamworker.dtos.StatisticsDto;
import com.teamworker.dtos.TaskDto;
import com.teamworker.models.Project;
import com.teamworker.models.Task;
import com.teamworker.services.TaskService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

@RestController
@CrossOrigin(origins = "http://localhost:4200/")
@RequestMapping(value = "/api/v1/admin/tasks")
@Tag(name = "/api/v1/admin/tasks", description = "Контролер для адміністрування завдань")
public class TaskAdminRestController {

    private final TaskService taskService;
    private final SimpleDateFormat getDateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm");
    private final SimpleDateFormat setDateFormat = new
SimpleDateFormat("dd.MM.yyyy, HH:mm");
    private final SimpleDateFormat statisticsDateFormat = new
SimpleDateFormat("MM/dd/yyyy");

    @Autowired
    public TaskAdminRestController(TaskService taskService) {
        this.taskService = taskService;
    }

    @GetMapping(value = "get/all")
    @Operation(summary = "Отримати всі завдання доступних проєктів")
    public ResponseEntity<List<TaskDto>> getAll() throws ParseException {

        List<Task> tasks = taskService.getAll();

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 64   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

        if (tasks == null) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }

        List<TaskDto> result = new ArrayList<>();

        for (Task task : tasks) {
            result.add(TaskDto.fromTask(task));
        }

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @GetMapping(value = "/get/all/{stage}")
    @Operation(summary = "Отримати всі завдання за стадією для адміністратора")
    public ResponseEntity<List<TaskDto>> getAllByStage(@PathVariable(value = "stage") String stageName) throws ParseException {
        List<Task> tasks = taskService.getAllByStageForAdmin(stageName);
        List<TaskDto> result = new ArrayList<>();

        for (Task task : tasks) {
            result.add(TaskDto.fromTask(task));
        }

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PostMapping(value = "/get/all/{id}")
    @Operation(summary = "Отримати всі завдання працівника за певний період")
    public ResponseEntity<List<TaskDto>> getAllByAssigneeAndCreateTime(@PathVariable(value = "id") Long id,
    @RequestBody StatisticsDto statisticsDto) throws ParseException {
        List<Task> tasks = taskService.getAllByAssigneeAndCreateTime(id,
        statisticsDto.getTime1(), statisticsDto.getTime2());
        List<TaskDto> result = new ArrayList<>();

        for (Task task : tasks) {
            result.add(TaskDto.fromTask(task));
        }

        return new ResponseEntity<>(result, HttpStatus.OK);
    }
}

```

## ДОДАТОК Г

### UserAdminRestController:

```

package com.teamworker.rest.admin;

import com.teamworker.dtos.MainUserInfoDto;
import com.teamworker.dtos.PositionDto;
import com.teamworker.dtos.UserDto;

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 65   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

import com.teamworker.models.User;
import com.teamworker.services.UserService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.text.ParseException;
import java.util.List;
import java.util.stream.Collectors;

@RestController
@CrossOrigin(origins = "http://localhost:4200/")
@RequestMapping(value = "/api/v1/admin/users")
@Tag(name = "/api/v1/admin/users", description = "Контролер адміністрування користувачів")
public class UserAdminRestController {

    private final UserService userService;

    @Autowired
    public UserAdminRestController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping(value = "get/all/managers")
    @Operation(summary = "Отримати всіх менеджерів")
    public ResponseEntity<List<UserDto>> getAllManagers() {
        List<User> users = userService.getAllManagers();

        if (users == null) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }
        List<UserDto> result =
users.stream().map(UserDto::fromUser).collect(Collectors.toList());

        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @PutMapping(value = "/update/{id}")
    @Operation(summary = "Оновити користувача")
    public ResponseEntity<UserDto> updateUser(
        @PathVariable(value = "id") Long id,
        @RequestBody UserDto userDto) {

        User user = userService.update(id, userDto.toUser());

        if (user == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        UserDto result = UserDto.fromUser(user);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 66   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |

```

    }

    @PutMapping(value = "/update/role/{id}")
    @Operation(summary = "Змінити роль користувача")
    public ResponseEntity<UserDto> updateUserRole(
        @PathVariable(value = "id") Long id,
        @RequestBody String role) {

        User user = userService.updateRole(id, role);

        if(user == null) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        UserDto result = UserDto.fromUser(user);
        return new ResponseEntity<>(result, HttpStatus.OK);
    }

    @DeleteMapping(value = "/delete/{id}")
    @Operation(summary = "Видалити користувача")
    public ResponseEntity<UserDto> deleteUser(@PathVariable(value = "id")
Long id) {

        if(userService.getById(id) == null ||
!userService.getById(id).getAssignedTasks().isEmpty()
|| !userService.getById(id).getCreatedTasks().isEmpty()) {
            return new ResponseEntity<>(HttpStatus.UNPROCESSABLE_ENTITY);
        }

        userService.delete(id);
        return new ResponseEntity<>(HttpStatus.OK);
    }
}

```

|      |      |          |        |      |                          |      |
|------|------|----------|--------|------|--------------------------|------|
|      |      |          |        |      | ІПЗ.КР.Б – 121 – 24 – ПЗ | Арк. |
|      |      |          |        |      |                          | 67   |
| Змн. | Арк. | № докум. | Підпис | Дата |                          |      |