Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу та інформаційних технологій

ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ

Пояснювальна записка до магістерської кваліфікаційної роботи

Виконав: студент 2 курсу, групи 2ІСТ-19м
спеціальності 126 – «Інформаційні системи та технології»
Лишак О.М.

Керівник: д.т.н., проф. Мокін О. Б. _____

Рецензент: к.т.н., доц. Бойко О. Р. _____

Вінниця ВНТУ – 2020 року

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу та інформаційних технологій

Освітньо-кваліфікаційний рівень магістр
Спеціальність 126 - Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ

_____ д.т.н., проф. В. Б. Мокін

"__" _____ 2020 р.

ЗАВДАННЯ
на магістерську кваліфікаційну роботу студенту
Лишаку Олександру Михайловичу

1. Тема роботи: «Інформаційна система управління проєктами»,
керівник роботи: Мокін О. Б., д.т.н., проф.,
затверджені наказом закладу вищої освіти від "__"_____2020 року №__
2. Строк подання студентом роботи_____
3. Вихідні дані до роботи:
   - результати експертного оцінювання існуючих систем управління проєктами;
   - специфікації agile підходів управління проєктами.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   - обґрунтування доцільності створення інформаційної системи управління проєктами;
   - проєктування інформаційної системи управління проєктами;
   - розробка інформаційної системи управління проєктами.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
   - структурна схема серверної частини інформаційної системи;
   - структурна схема клієнтської частини інформаційної системи;
   - UML діаграма прецедентів робітника;
   - діаграма діяльності процесу додавання та зміни завдань робітником;
   - UML діаграма прецедентів керівника;
   - діаграма діяльності процесу редагування дошки керівником;
   - UML діаграма прецедентів власника;
   - діаграма діяльності процесу редагування дошки власником;
   - UML діаграма класів бази даних інформаційної системи управління проєктами;
   - вкладки сторінки статистики.

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Руда Л.П., к.е.н., доц. каф. ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МКР	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області		
2	Проєктування інформаційної системи		
3	Розробка інформаційної системи		
4	Економічна частина		
5	Розробка інструкції користувача		
6	Оформлення матеріалів до захисту МКР		

Студент _____ Лишак О. М.

Керівник роботи _____ Мокін О. Б.

Рецензент _____ Бойко О. Р.

РЕФЕРАТ

Магістерська кваліфікаційна робота: 115 стор., 6 табл., 54 рис., 30 джерел.

Об’єкт досліджень – є процес управління проєктами.

Мета роботи – підвищення ефективності роботи команди над проєктом за допомогою інформаційної системи управління проєктами, яка відрізняється від існуючих спрощеним інтерфейсом та розширеним модулем аналітики.

Проведено аналіз існуючих систем для управління проєктами. Здійснено проєктування модулів для реалізації інформаційної системи. Визначено оптимальні технології для реалізації.

Прогнозні припущення про розвиток об’єкта дослідження – розробка інформаційної системи, яка буде мати простий та зрозумілий інтерфейс, а також буде надавати можливість перегляду в інформацію про процеси які виконуються в проєкті в зручному вигляді.

Галузь застосування – веб-сервіси, що спеціалізуються на організації робочого процесу.

ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ ПРОЄКТАМИ, ВЕБ-СЕРВІС, МОДУЛЬ АНАЛІТИКИ, РОБОЧИЙ ПРОЦЕС.

ABSTRACT

Master's qualification work: 115 pages, 6 tables, 54 figures, 30 sources.

The object of research is the project management process.

The purpose of the work is to increase the efficiency of the team's work on the project with the help of the project management information system, which differs from the existing ones by a simplified interface and an advanced analytics module.

The analysis of existing systems for project management is carried out. Modules for information system implementation have been designed. The optimal technologies for implementation are determined.

Predictive assumptions about the development of the object of study - the development of an information system that will have a simple and clear interface, as well as will provide an opportunity to view information about the processes performed in the project in a convenient way.

Field of application - web services that specialize in the organization of the workflow.

INFORMATION SYSTEM, PROJECT MANAGEMENT, WEB SERVICE, ANALYTICS MODULE, WORKING PROCESS.

ЗМІСТ

ВСТУП

У сучасному світі часто виникає питання поліпшення робочого процесу для покращення ефективності розподілу робочого часу. Інформаційна система управління проєктами, це інструмент який допомагає спростити процес роботи. Використовуючи такі сервіси користувачі будуть заощаджувати свій час, які вони могли потратити на непотрібні наради та обговорення. Менеджеру не потрібно особисто повідомляти працівників про нові задачі, про помилки в реалізації того чи іншого завдання і так далі.

Також деякі платформи надають функціонал аналізу процесів проєкту, що також заощаджує час керівника та спрощує створення звітів. Крім цього аналіз проєкту може допомогти своєчасно побачити проблему в організації процесу, що зможе заощадити виділені ресурси.

Аналіз існуючих систем показав, що кожна з них має як і свої переваги, так і недоліки. Основними недоліками існуючих системи є:

– досить складний та заплутаний інтерфейс;

– обмежений набір функцій які доступні для безкоштовного використання;

– погана візуалізація або відсутність аналізу проєкту.

Отже, розробка інформаційної системи управління проєктами, яка буде мати простий та зрозумілий інтерфейс, а також буде надавати можливість перегляду в інформацію про процеси які виконуються в проєкті в зручному вигляді, є актуальною.

Об’єктом дослідження є процес управління проєктами.

Предметом дослідження є інформаційна система управління проєктами.

Метою дослідження є підвищення ефективності роботи команди над проєктом за допомогою інформаційної системи управління проєктами, яка відрізняється від існуючих спрощеним інтерфейсом та розширеним модулем аналітики.

Наукова новизна одержаних результатів. Основні результати, які були отримані в процесі вирішення поставлених завдань та становлять наукову новизну дослідження, полягають у розробці інформаційної системи управління проєктами, яка відрізняється від існуючих спрощеним інтерфейсом та розширеним модулем аналітики, що дозволяє підвищити ефективність роботи команди над проєктом.

Практичне значення одержаних результатів полягає у більш ефективному використанні часу та кращій координації роботи команди за рахунок впровадження таких особливостей інформаційної системою управління проєктом:

– реалізовано більш простий та зручний інтерфейс;

– реалізовано модуль розширеної аналітики;

– побудовано UML-діаграму класів, яка дозволяє швидко відслідковувати взаємодію елементів проєкту.

Достовірність теоретичних положень магістерської кваліфікаційної роботи обґрунтовується детальним ознайомленням з предметною областю, у тому числі зі схожими рішеннями, для коректного розуміння значення та розробки технології, аналізом та поясненням доцільності застосування тих чи інших методів та інструментів реалізації.

Апробація результатів роботи. Результати роботи були апробовані на XV на міжнародній конференції "Контроль і управління в складних системах" (КУСС-2020).

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано: 1 тези на XV міжнародній конференції "Контроль і управління в складних системах" (КУСС-2020) [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Аналіз предметної області

У даній дипломній роботі за мету поставлено задачу створення інформаційної системи для управління проєктами. Управління проєктами - це процес керування роботою команди для досягнення цілей та задоволення критеріїв успіху у визначений час. Основною проблемою управління проєктами є виконання усіх цілей проєкту в рамках заданих обмежень. Ця інформація зазвичай описується в проєктній документації, створеній на початку процесу розробки. Найбільш частими обмеженнями є обсяг, час, якість та бюджет. Другорядною задачею є оптимізація розподілу необхідних ресурсів та їх застосування для досягнення заздалегідь визначених цілей.

Мета управління проєктами - виготовлення проєкту, який відповідає цілям. Після чіткого встановлення цілей вони повинні впливати на всі рішення, прийняті іншими людьми, що приймають участь у проєкті - наприклад, керівниками проєктів, дизайнерами, підрядниками та субпідрядниками. Невизначені або занадто чітко прописані цілі управління проєктами шкодять прийняттю рішень [2, 3].

Проєкт - це тимчасова робота, спрямована на виготовлення унікального продукту, послуги чи результату з визначеним початком і кінцем (як правило, обмежена часом, а часто обмежена фінансуванням чи персоналом), що здійснюється для досягнення особливих завдань та цілей, як правило, для здійснення корисних змін або додану вартість [4]. На практиці управління проєктами вимагає розвитку чітких технічних умінь та стратегій керування.

Загальноприйняті методи управління проєктами можна застосувати до будь-якого проєкту. Вони часто пристосовуються до конкретного типу проєкту на основі його типу, галузі, розміру і так далі [5]. Наприклад, для будівельної галузі було розроблено спеціалізовану форму управління проєктами, яку називають

«управлінням будівельними проєктами», і в якій керівники проєктів можуть пройти навчання та сертифікацію. Індустрія інформаційних технологій також отримала свою форму управління, яку називають «управління IT-проєктами», і яка зосереджена на наданні технічних послуг, які потрібні для виконання різних етапів життєвого циклу розробки, таких як планування, проєктування, розробка, тестування і впровадження. Існує державне управління проєктами, яке охоплює всі державні роботи, які повинні виконуватися державними установами або залучатися до підрядників [6].

Спільним для усіх видів управління проєктами є те, що вони сконцентровані на трьох основних цілях: час, обсяг та бюджет. Успішні проєкти повинні бути виконані вчасно, в межах бюджету та згідно з узгодженим попередньо обсягом завдань. Ці критерії часто називають «золотим трикутником» проєкту [7]. Схематичне відображення трьох основних цілей проєкту зображено на рисунку 1.1.

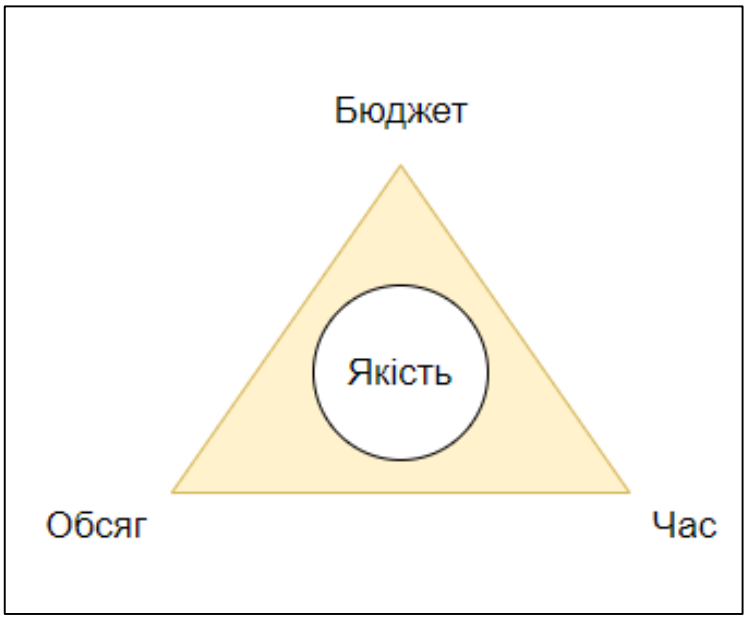


Рисунок 1.1 – «Золотий трикутник» проєкту

Для кожного типу управління проєктами керівники розробляють і застосовують повторювані шаблони, характерні для тієї чи іншої галузі. Це дозволяє побудувати чіткі плани, з конкретним наміром підвищити якість, знизити витрати та скоротити час на досягнення результатів проєкту [8].

Зазвичай керування проєктами містить в собі такі елементи:

– система контролю;

– чотири групи процесів.

Застосовуються ті самі основні процеси управління, не дивлячись на термінології та методології.

Основні етапи реалізації проєктів:

– ініціювання;

– розробка чи проєктування;

– виконання або використання;

– контроль та моніторинг;

– закінчення.

Порядок етапів виконання проєктів зображено на рисунку 1.2.

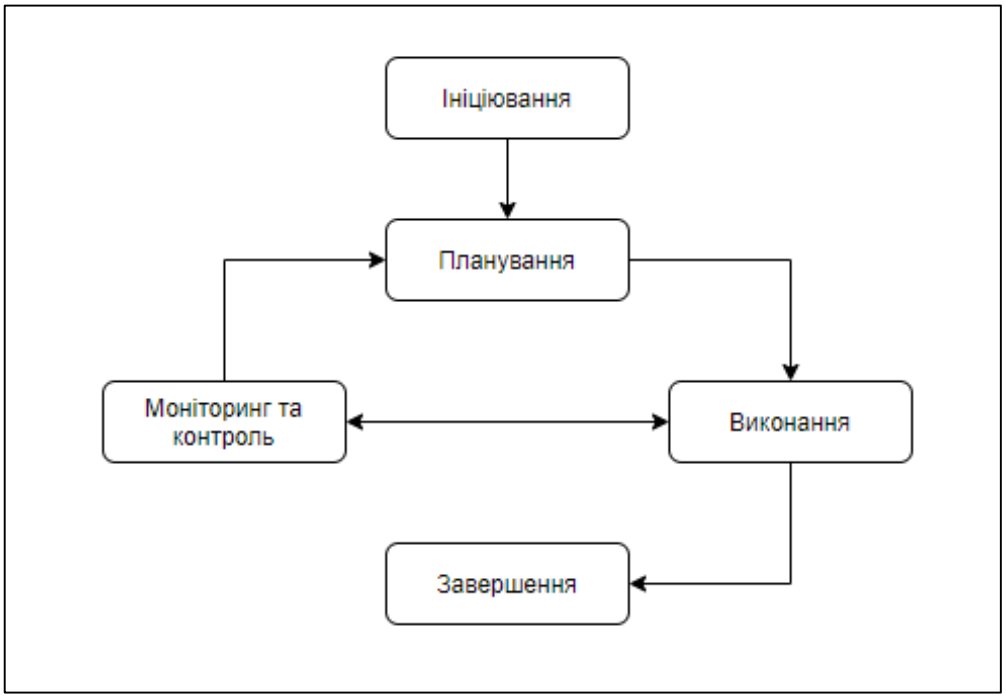


Рисунок 1.2 – Порядок етапів виконання проєктів

У дослідницьких проєктах вказані етапи деколи доповнюються етапом прийняття рішень, на якому вирішують чи слід продовжувати чи завершувати створення проєкту [9, 10].

1.2 Огляд існуючих програмних продуктів

Зараз існує велика кількість інструментів які допомагають ефективно організувати робочий процес. Розглянемо три найпопулярніші рішення:

- asana;
- jira;
- trello;
- redmine.

Asana - це інструмент для управління завданнями, призначений для допомоги командам в організації, відстеженні та керуванні своєю роботою. Платформа надає можливість створювати проєкти, а потім завдання, які потрібно виконати в рамках проєкту, також є можливість додати членів своєї команди до конкретних завдань або проєктів [11].

Asana більше орієнтована для роботи з малими проєктами. Платформа підтримує 6 мов, але української серед них немає.

Структура системи складається з організацій, відділів, проєктів і завдань. Кожна наступна категорія створюється в складі попередньої. Також доступні теги для об'єднання завдань і секції для їх поділу. У Asana передбачений пошук, який дозволяє виявити підзадачі серед великого обсягу інформації, і фільтр, за допомогою якого користувачі можуть зберегти необхідні варіанти пошуку в обраному.

Головне вікно проєкту на даній платформі зображено на рисунку 1.3.

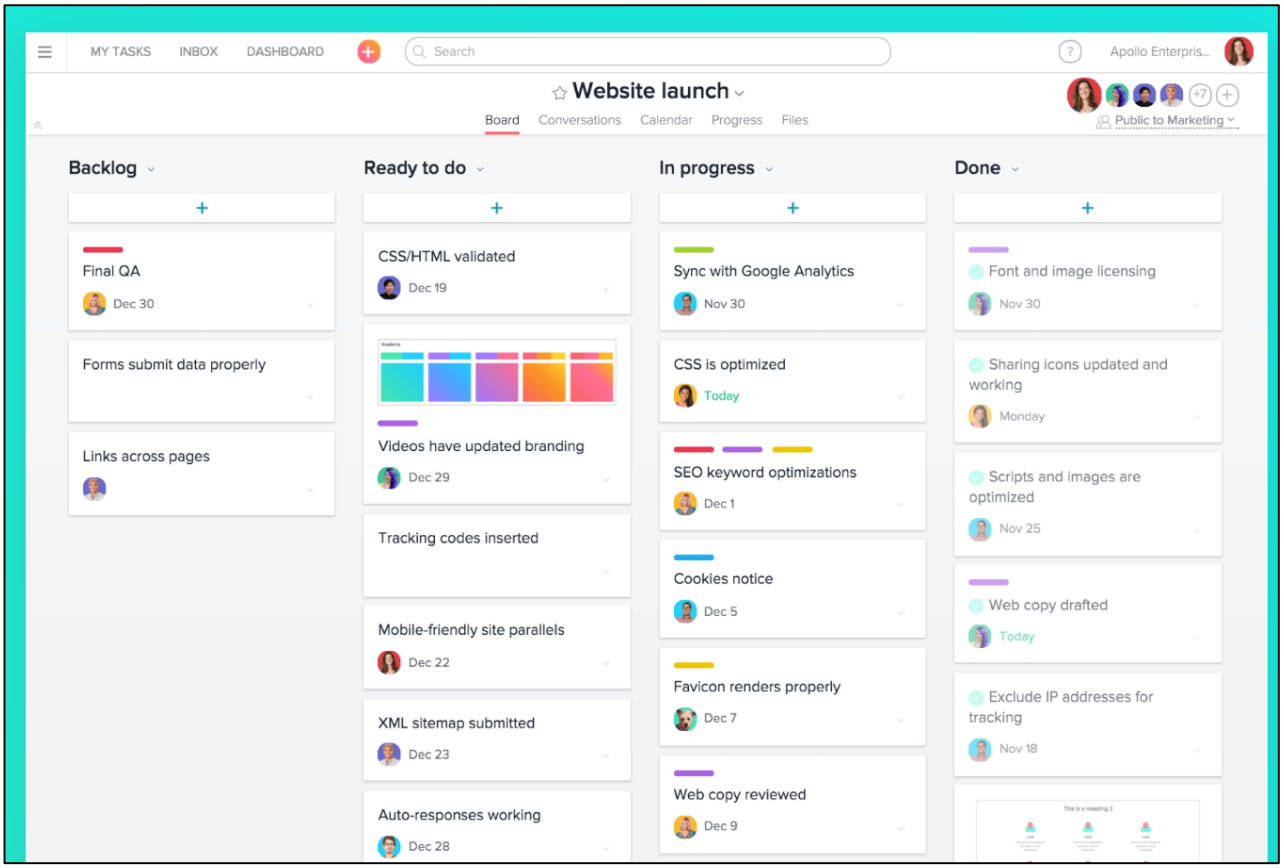


Рисунок 1.3 – Головне вікно проєкту на платформі Asana

Asana має три рівні ціноутворення. Мінімальна ціна для одного користувача – 10 доларів, максимальна – 24 долари.

Asana має наступні переваги та недоліки.

Переваги сервісу Asana:

– повторювані завдання;

– відстеження термінів.

Недоліки сервісу Asana:

– неможливість призначати завдання декільком користувачам відразу;

– відсутня українська локалізація.

Jira - це інструмент відстеження проблем та помилок для управління проєктами. Існує веб-версія інструменту, а також додатки для мобільних на операційних системах iOS та Android [12].

Інструмент містить в собі різні методології: дошки канбан, scrum та змішані, це допомагає охопити їй велику групу користувачів. Також існує система

оцінювання: користувачі можуть розробити нову або використовувати існуючу. Є розподіл проєктів і завдань за пріоритетністю.

Головне вікно проєкту на даній платформі зображено на рисунку 1.4.

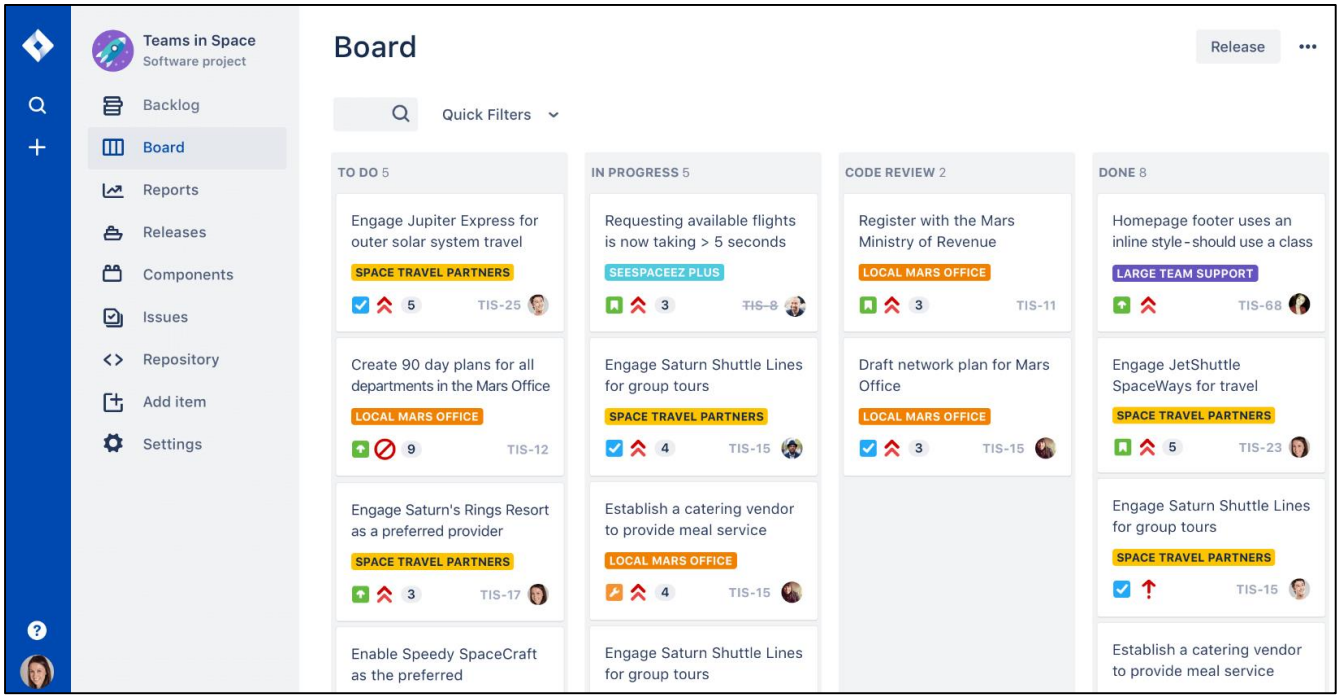


Рисунок 1.4 – Головне вікно проєкту на платформі Jira

Jira має два стандартні рівні ціноутворення. Мінімальна ціна для одного користувача – 1 долар, максимальна – 7 доларів.

Jira має наступні переваги та недоліки.

Переваги сервісу Jira:

– змішані методологій і scrum;

– система оцінювання.

Недоліки сервісу Jira:

– висока ціна;

– складність налаштування;

– велика кількість функціоналу доступна тільки при використанні сторонніх плагінів;

– відсутня українська локалізація.

Trello – направлений на організацію проєктів у дошки. Основною перевагою є відсутність двозначності, кожна функція інтуїтивно зрозуміла [13]. Головне вікно проєкту на даній платформі зображено на рисунку 1.5.

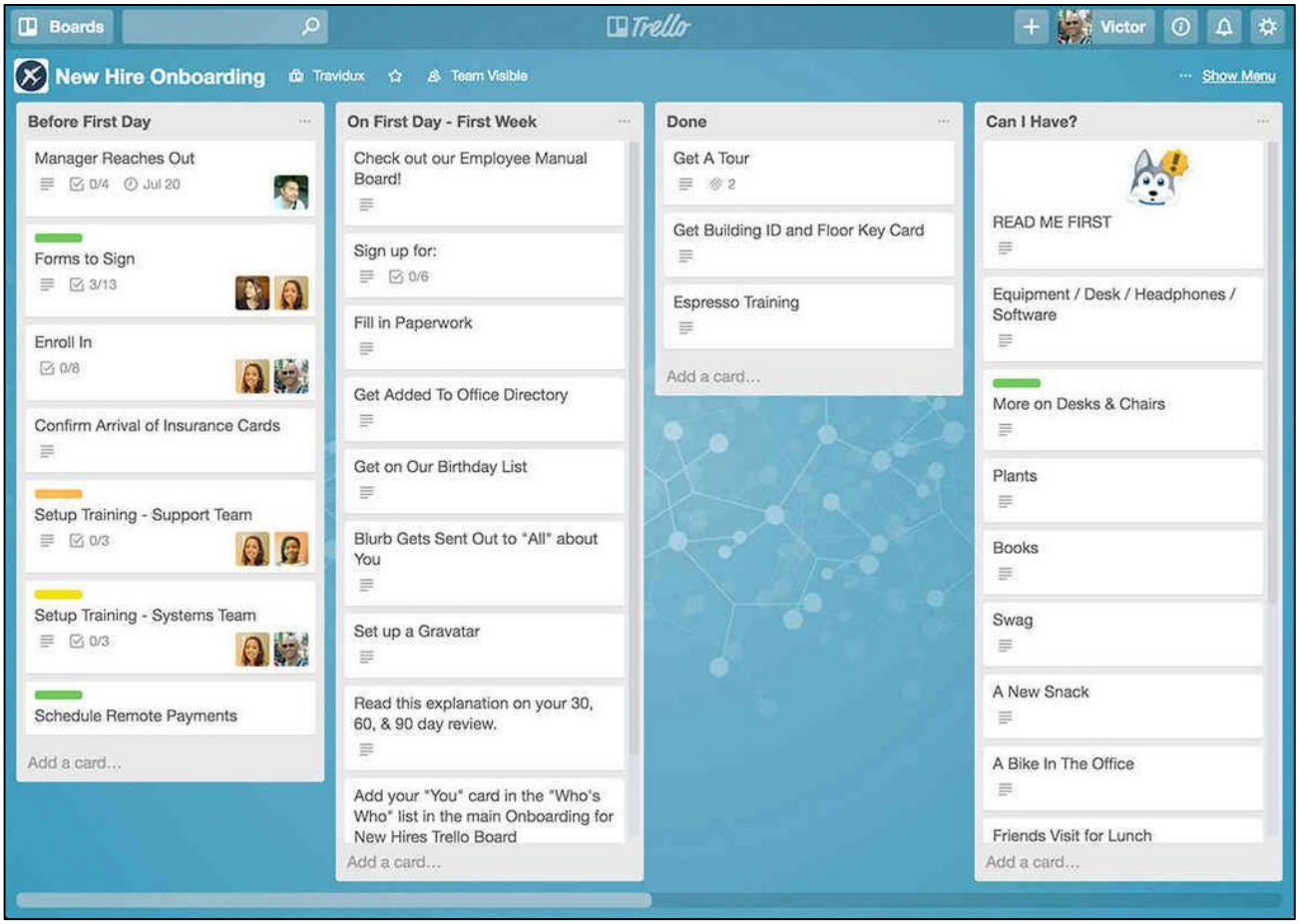


Рисунок 1.5 – Головне вікно проєкту на платформі Trello.

Trello має три стандартні рівні ціноутворення. Мінімальний рівень є безкоштовним, максимальний – 13 доларів для одного користувача.

Trello має наступні переваги та недоліки.

Переваги сервісу Trello:

– наочність роботи та управління;

– створення власних приватних дощок.

Недоліки сервісу Trello:

– відсутня можливість відстеження часу;

– відсутня будь-яка статистика проєктів;

– немає можливості додавати опис проєктів.

Redmine – сервіс який надає можливість не лише керувати проєктами та завданнями, але й відслідковувати помилки. Головною особливістю системи є підтримка декількох проєктів одночасно [14].

У користувача є можливість додавати підпроєкти. Завдання, звіти і помилки можуть переміщатися між підпроєктами. Кожному завданню можна призначати унікальний список користувачів і надавати доступ до функціоналу окремо для кожного. Користувачам призначається роль, яку можна змінювати відповідно до завдань користувача на конкретному проєкті.

Платформа містить такі інструменти, як діаграми Ганта та календар. Доступні наступні можливості: форум, особисті завдання, повідомлення, керування файлами, підрахунок часу, теги, репозиторій.

Головне вікно проєкту на даній платформі зображено на рисунку 1.6.

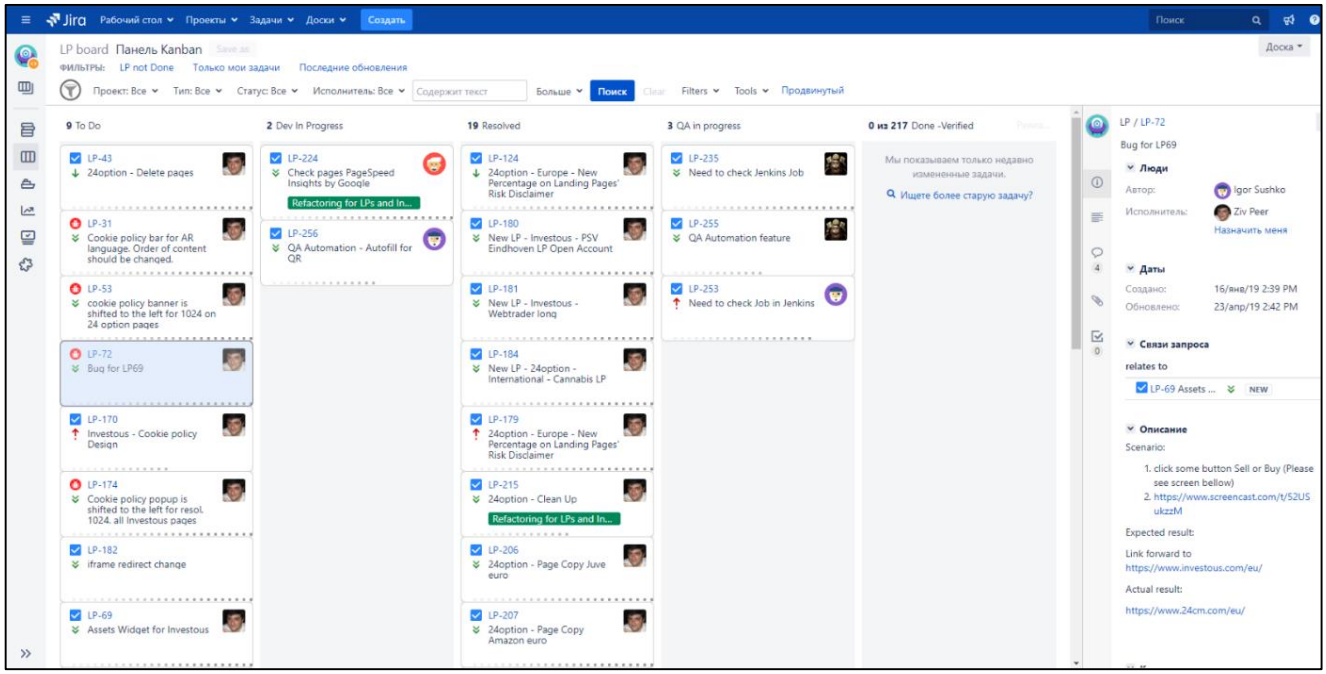


Рисунок 1.6 – Головне вікно проєкту на платформі Redmine

Redmine є повністю безкоштовним.

Redmine має наступні переваги та недоліки.

Переваги сервісу Redmine:

– сервіс є безкоштовним;
– створення форумів і вікі-сторінок проєктів.

Недоліки сервісу Redmine:

– застарілий та не зручний інтерфейс;
– відсутня будь-яка статистика проєктів;
– нестабільність роботи.

1.3 Висновки

В розділі розглянуто і проаналізовано проблему управління проєктами. Було розглянуто методи, цілі, типи, етапи управління проєктами та їх послідовність.

Проведено огляд існуючих інформаційних систем для управління проєктами. Як результат порівняння було вирішено розробити інформаційну систему, яка б містила основний функціонал даних систем, а також б мала такі переваги над оглянутими системами: українська локалізація, можливість відслідковувати час виконання завдань, проста в налаштуванні та використані, статистика проєктів.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

2.1 Моделювання структури програмного забезпечення

Розглянемо загальну архітектуру розроблюваного додатку. Оскільки інформаційна система повинна бути доступна в мережі інтернет, було вибрано клієнт-серверну архітектуру, оскільки вона є найоптимальнішою на даний момент, тому система буде складається з двох частин: серверна частина та клієнтська частина. Загальна структура зображена на рисунку 2.1.

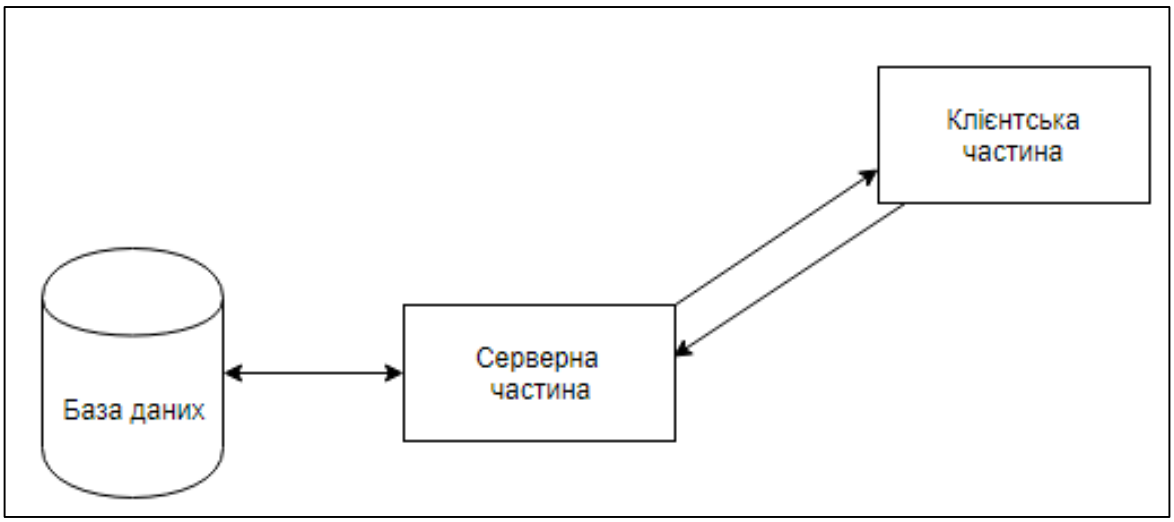


Рисунок 2.1 – Загальна структурна схема інформаційної системи

Розглянемо кожну частину детальніше. Спочатку розглянемо серверну частину системи. Серверна частина являє собою Node.js (Express) сервер, який реалізує архітектуру REST. REST - це стиль архітектури програмного забезпечення для розподілених систем. У загальному випадку REST - це дуже простий інтерфейс для управління інформацією без використання додаткових внутрішніх шарів [15-17]. Кожна інформаційна одиниця однозначно ідентифікується глобальним ідентифікатором, таким як URL-адреса. Кожна URL-адреса, у свою чергу, має строго визначений формат.

Опишемо архітектуру серверного додатку. Додаток складається з файлів маршрутизації запитів та файлів контролерів. Сукупність саме цих файлів визначає які запити прийматиме серверний додаток та яким чином вони будуть оброблені. Необхідно буде по одному файлу кожного типу для кожної сутності, яка використовується в програмному модулі. Іншою важливою частиною є сервіси. Сервіси – це файли, які містять безпосередньо всю логіку, яку реалізовуватиме додаток: робота з базою, проведення обрахунків тощо. Кожна сутність так само потребує по одному сервісу [18]. Крім того, необхідно буде додати файли – утиліти. Утиліти це файли, які містять функції для виконання типових задач. Структура схема серверної частини інформаційної системи зображена на рисунку 2.2.

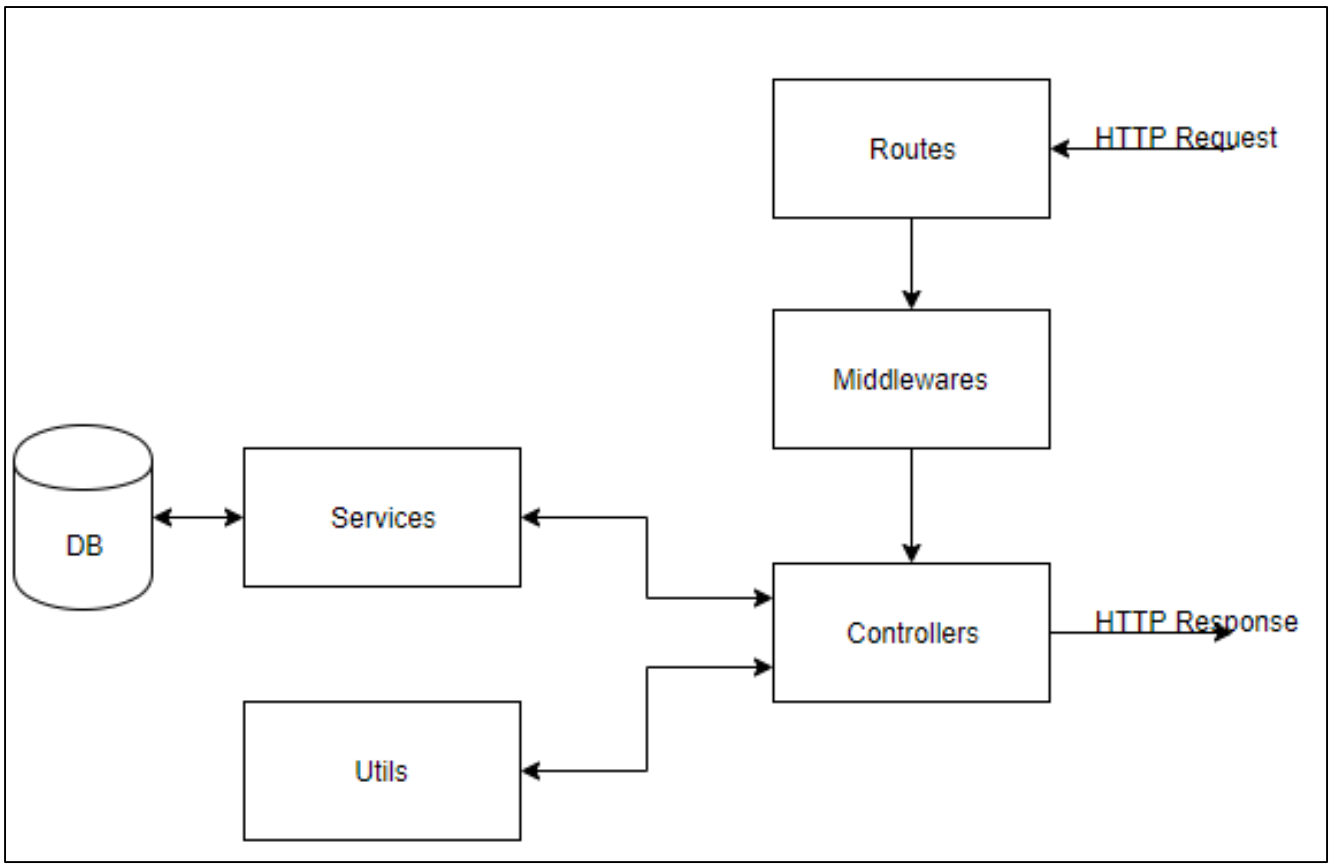


Рисунок 2.2 – Структурна схема серверної частини інформаційної системи

Тепер розглянемо клієнтську частину. Вона буде створюватись за допомогою JavaScript фреймворку – Angular. Головними будівельними блоками будь-якого Angular проєкту є компоненти. Кожен компонент це функція або клас, які

відповідають за певну частину інтерфейсу [19]. Компонуючи менші блоки у групи створюються ще більші компоненти, який можуть представляти цілі сторінки додатку. Компоненти дозволяють розділити інтерфейс користувача на незалежні частини, використовувати їх повторно і сприймати їх як такі, що функціонують окремо один від одного [20]. Також додаток містить додаткові файли – сервіси, які містять у собі логіку для отримання даних від серверу, їх обробки та набір різних додаткових функцій, які використовуються в декількох місцях проєкту [21, 22].

Структура схема клієнтської частини інформаційної системи зображена на рисунку 2.3.

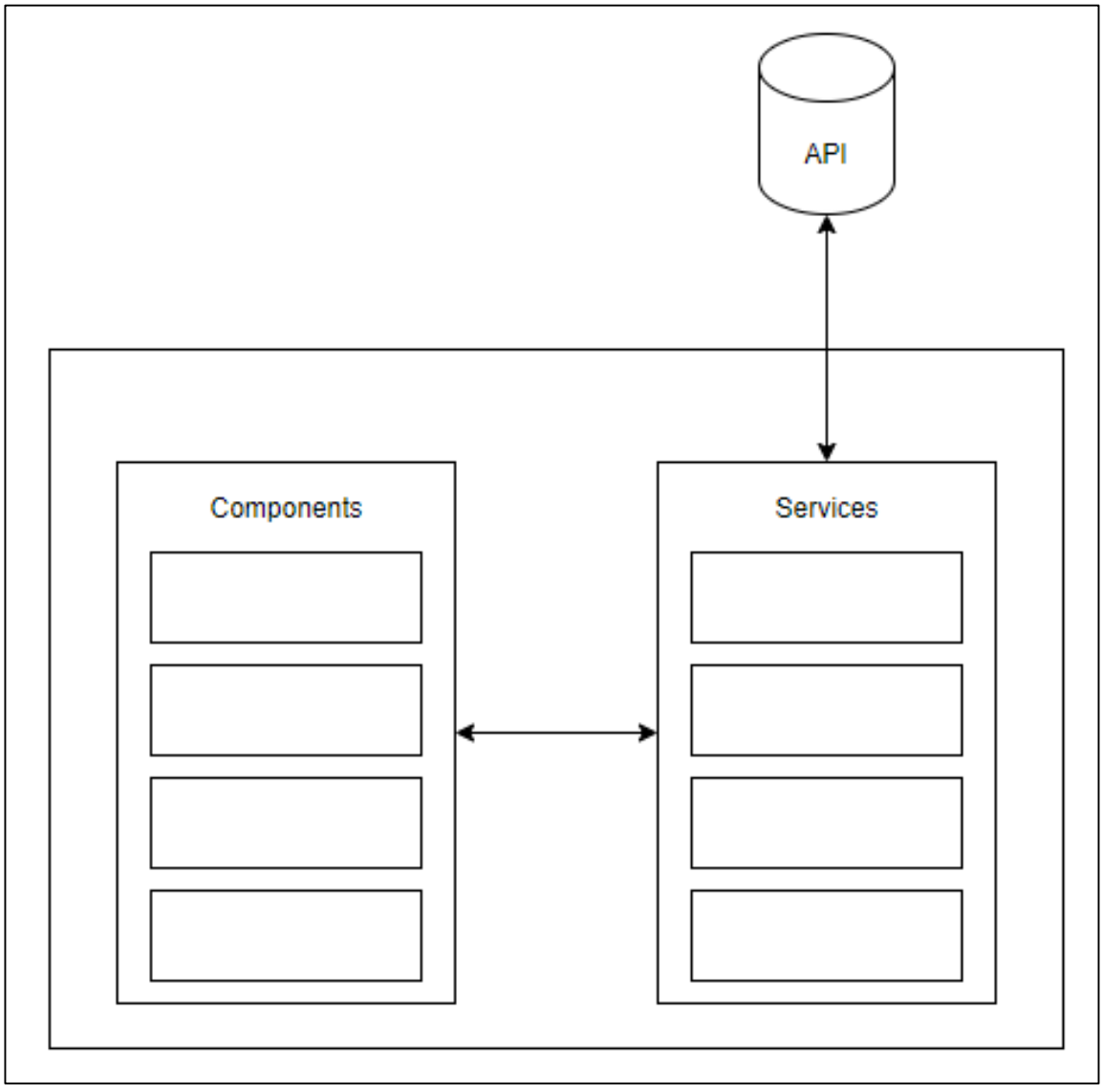


Рисунок 2.3 – Структурна схема клієнтської частини інформаційної системи

2.2 Опис типів користувачів

Для того щоб функціонал, який повинна містити інформаційна система керування проєктами, міг бути реалізований в повній мірі, в системі потрібно визначити три типи користувачів:

- робітник;
- керівник;
- власник.

Тип користувача може бути різним на різних дошках. Також тип може змінюватись в процесі використання системи.

Робітник має найменший набір інструментів для керування дошкою. Кожен наступний тип користувача включає в себе функціонал попереднього та розширюється додатковим. Тип користувача «власник» має найбільший набір інструментів для керування дошкою.

2.3 Формування процесів та методів роботи робітника

Інструменти для відображення, створення, зміни та видалення завдань повинні бути доступні користувачам які мають тип «робітник». Робітник повинен отримати змогу створити чи змінити наступні параметри завдань:

- заголовок;
- опис;
- оцінка часу виконання;
- фактичний час виконання;
- пріоритетність;
- колонка;
- виконавці.

Ці ж параметри будуть доступні при створені нового завдання, обов’язковим параметром повинен бути заголовок завдання. При видаленні завдань користувач

повинен вибрати тип видалення. Тип видалення буде використовуватись в побудові статистики проєкту. Буде доступно 3 типи видалення:

– виконане (done);
– не актуальне (deleted);
– провалене (failed);

Також робітнику мають бути доступні інструменти для відображення дошки.

– Набір функцій, що повинна містити в собі інформаційна система управління проєктами для ролі робітника:

– додавання завдань;
– відображення завдань;
– зміна завдань;
– видалення завдань;
– відображення дошки.

Увесь набір функцій, які повинна містити інформаційна система управління проєктами для ролі робітника, відображений в use-case діаграмі. UML діаграма прецедентів робітника на рисунку 2.4.

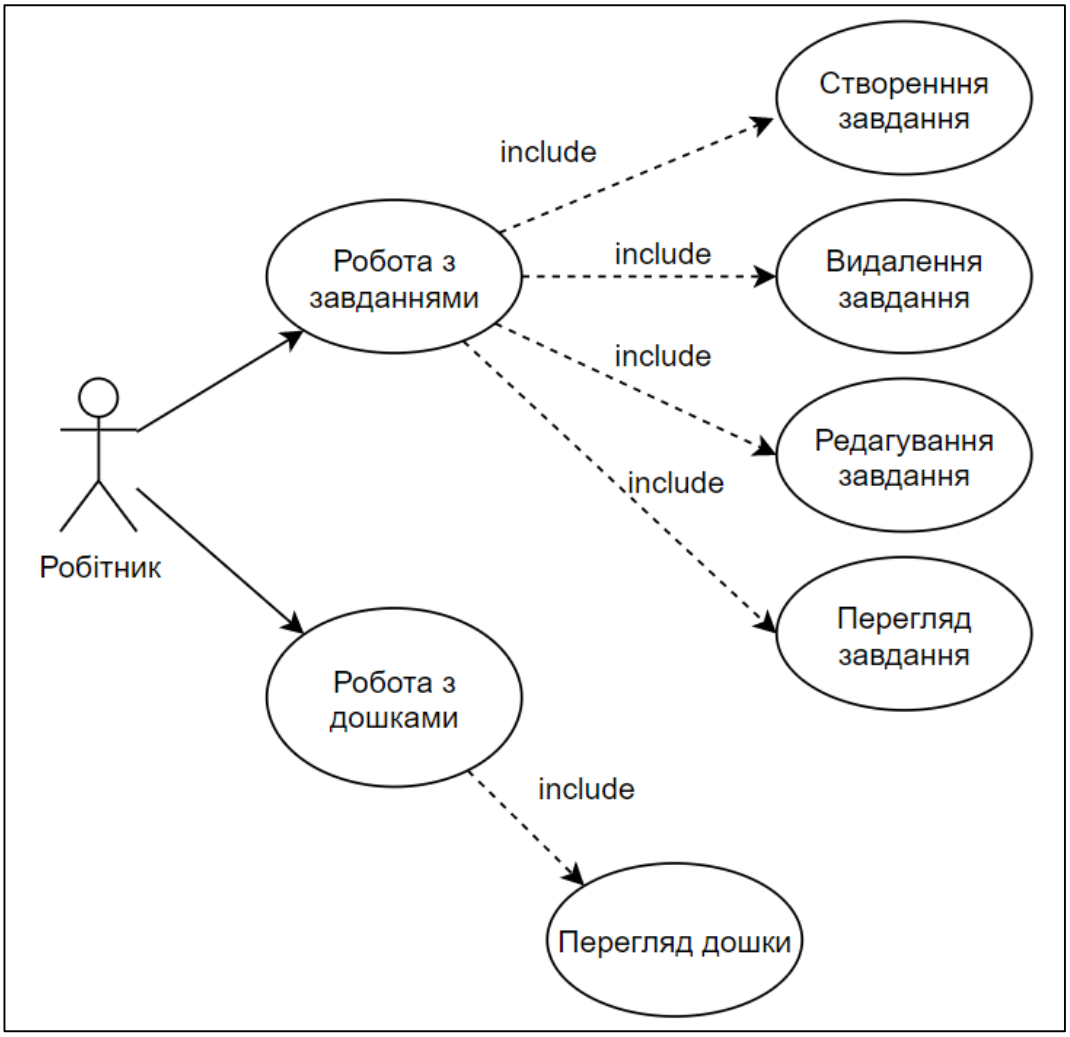


Рисунок 2.4 – UML діаграма прецедентів робітника

Щоб забезпечити просте та зручне додавання та зміну завдань слід зробити цей процес у вигляді редагування потрібних параметрів.

У формі з інформацією про завдання слід також здійснювати перевірку на коректність введеної інформації та виводити відповідні сповіщення у разі помилок.

Інформацію про усі завдання тієї, чи іншої дошки зручно буде показувати користувачам у вигляді колонок з переліком завдань в них. Слід зазначити, що робітники зможе редагувати завдання з моменту створення до моменту видалення. Для цього йому потрібно буде відкрити відповідну дошку на якій знаходиться потрібне завдання і натиснути на це завдання, після чого повинне з’явитися вікно перегляду завдань. В вікні перегляду завдання будуть доступні кнопки для відкриття вікон редагування та видалення завдань. При потребі в вікні редагування завдань робітник зможе змінити усі необхідні поля.

Опишемо усі процеси які можуть виконуватись робітником за допомогою UML діаграми діяльності. Діаграма діяльності процесу додавання та зміни завдань робітником відображена на рисунку 2.5.

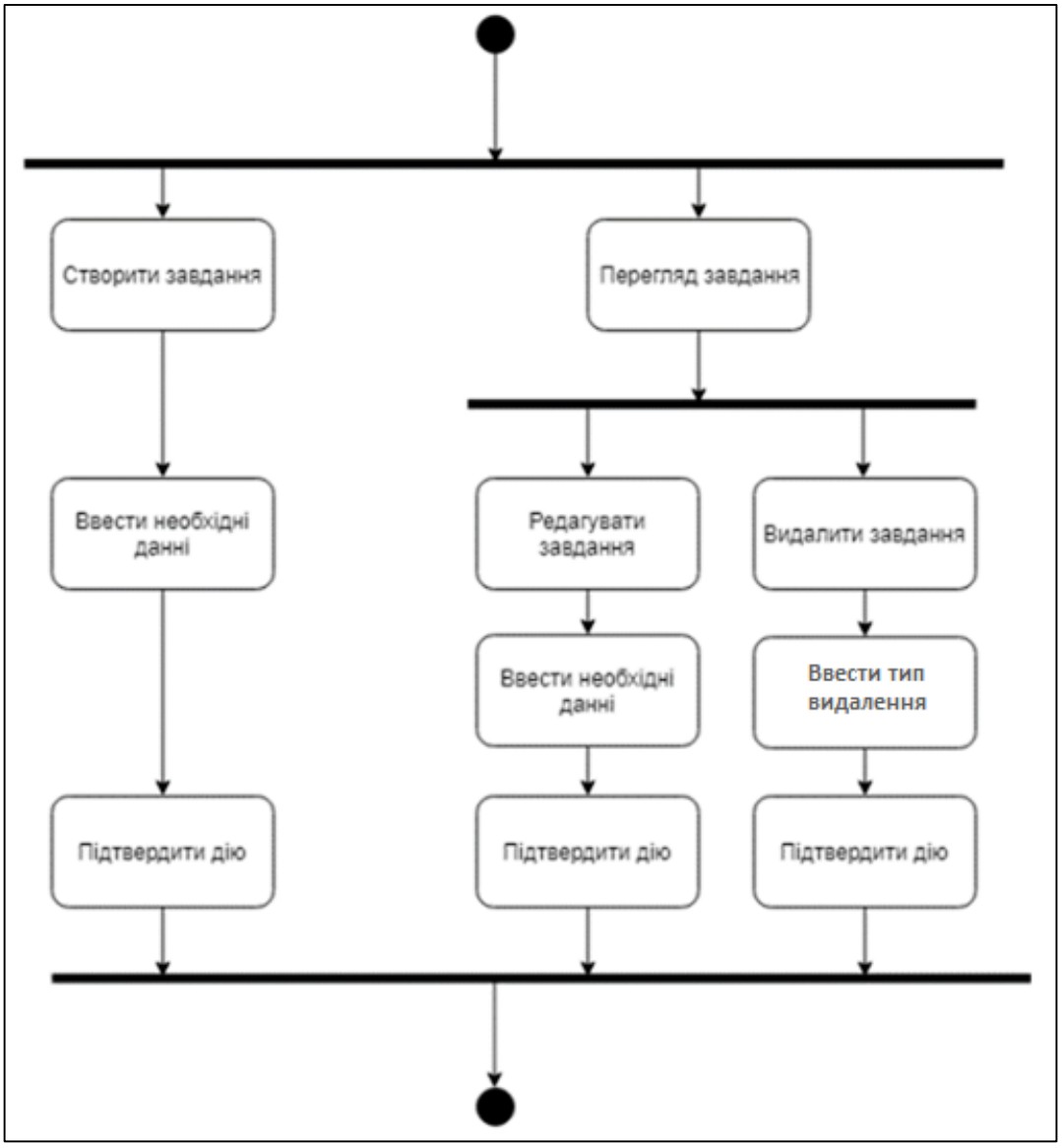


Рисунок 2.5 – Діаграма діяльності процесу додавання та зміни завдань робітником

2.4 Формування процесів та методів роботи керівника

Інструменти для редагування дощок, а також для зміни, додавання та видалення завдань повинні бути доступні керівникам дошки. Користувач типу «керівник» повинен мати можливість додати користувачів до потрібної дошки, чи

видалити їх у разі необхідності, додавати нові колонки дошки, або змінювати такі параметри дошки як:

- назва;
- колір;
- позиція.

Набір функцій, що повинна містити в собі інформаційна система управління проєктами для ролі керівника:

- додавання колонок;
- зміна колонок;
- видалення колонок;
- додавання користувачів;
- видалення користувачів;
- зміна параметрів дошки;
- відображення дошки;
- додавання завдань;
- відображення завдань;
- відображення статистики дошки;
- зміна завдань;
- видалення завдань.

Увесь набір функцій, що повинна містити інформаційна система управління проєктами для ролі керівника, відображений в use-case діаграмі. UML діаграма прецедентів керівника на рисунку 2.6.

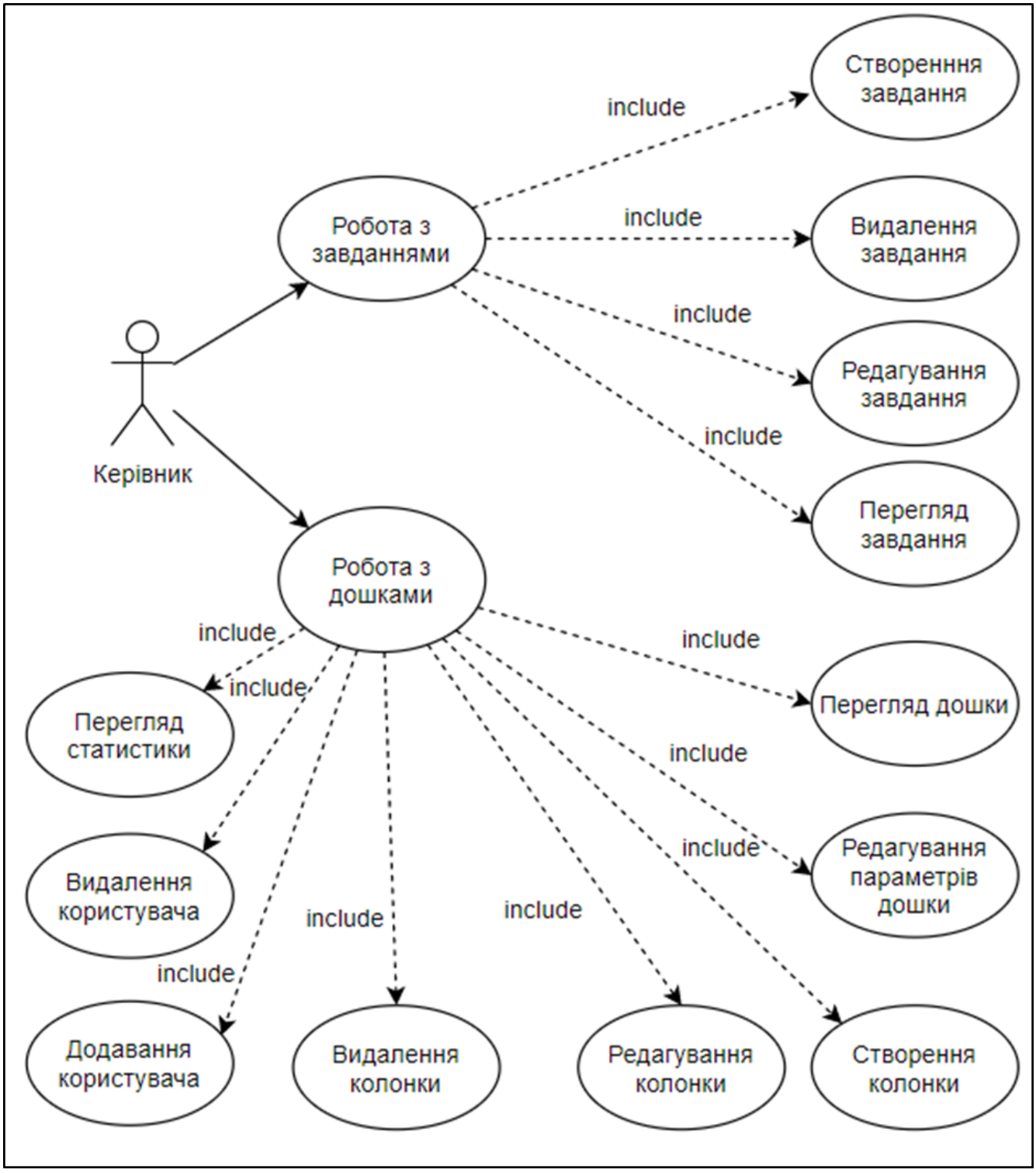


Рисунок 2.6 – UML діаграма прецедентів керівника

Щоб було легко та зручно створювати та модифікувати дошки, цей процес потрібно виконати у вигляді редагування потрібних полів.

У формі, яка містить інформацію про дошку, потрібно перевірити правильність введеної інформації та відобразити відповідні попередження про помилки у разі їх виникнення.

Інформація про дошку буде відображатися в окремому вікні. Керівники зможуть редагувати існуючі дошки. Для цього авторизованим користувачам доведеться відкрити дошку та натиснути кнопку «редагувати», після чого з’явиться форма для редагування дошки. За потреби користувач може змінити будь-які доступні поля.

Опишемо усі процеси які можуть виконуватись керівником за допомогою UML діаграми діяльності. Діаграма діяльності процесу редагування дошки керівником відображена на рисунку 2.7.

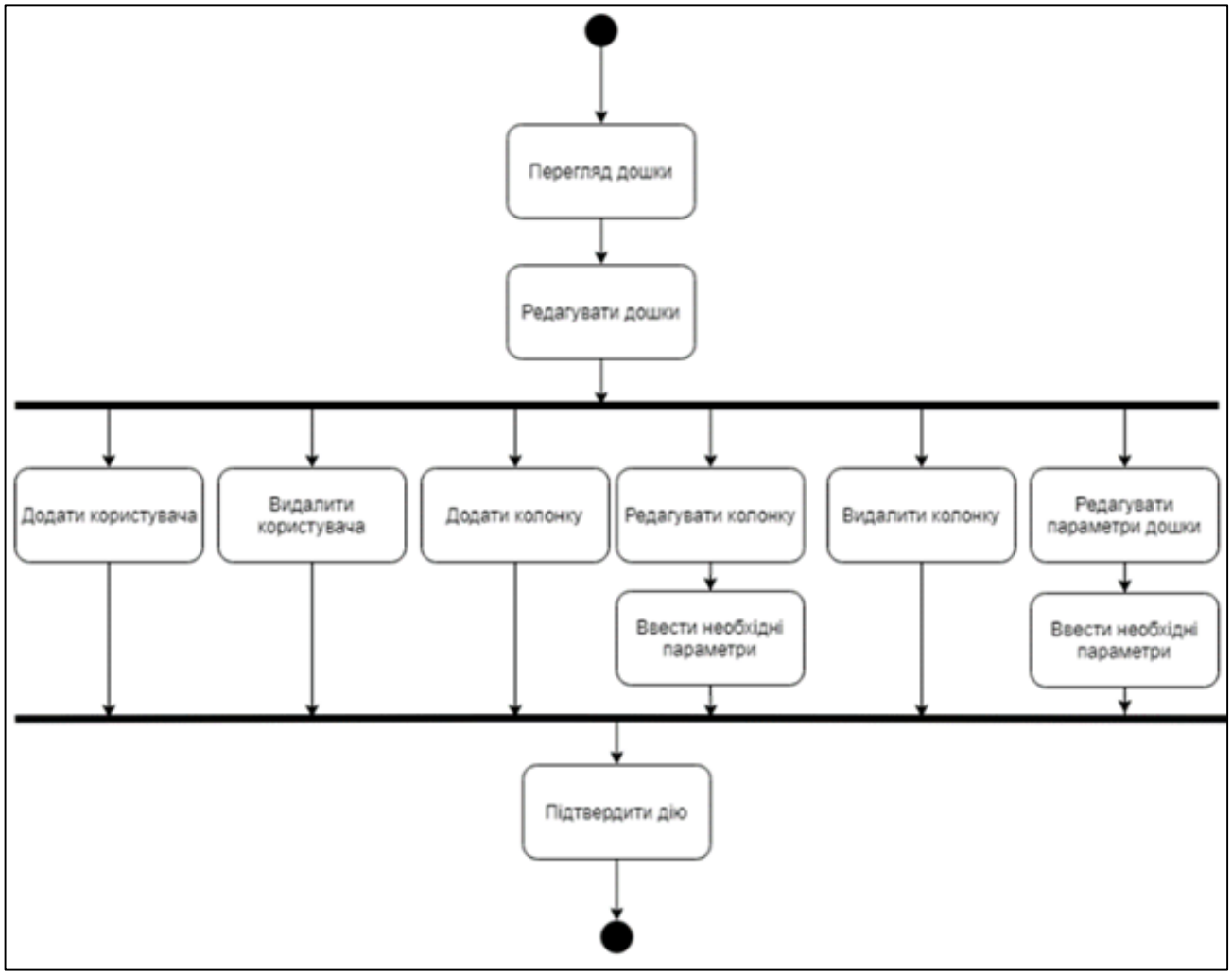


Рисунок 2.7 – Діаграма діяльності процесу редагування дошки керівником

2.5 Формування процесів та методів роботи власника

Користувачам з типом «власник» повинен мати доступ до усіх даних та процесів дошки.

Набір функцій, що повинна містити в собі інформаційна система управління проєктами для ролі власника:

- додавання колонок;
- зміна колонок;
- видалення колонок;
- додавання користувачів;
- зміна типу користувачів;
- видалення користувачів;
- зміна параметрів дошки;
- відображення дошки;
- відображення статистики дошки;
- видалення дошки;
- додавання завдань;
- відображення завдань;
- зміна завдань;
- видалення завдань.

Увесь набір функцій, що повинна містити інформаційна система управління проєктами для ролі власника, відображений в use-case діаграмі. UML діаграма прецедентів власника на рисунку 2.8.

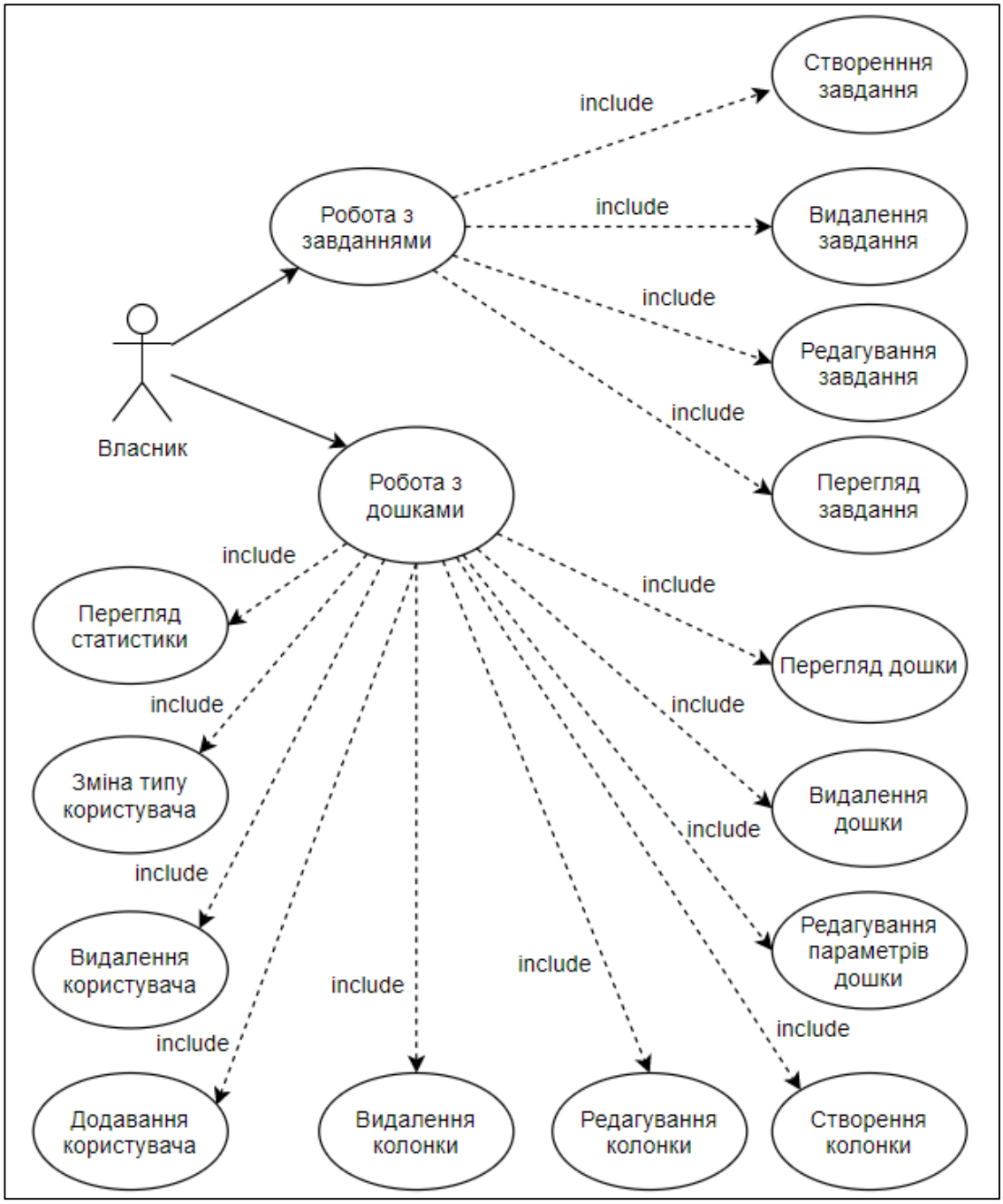


Рисунок 2.8 – UML діаграма прецедентів власника

Щоб було легко та зручно створювати та модифікувати дошки, цей процес потрібно виконати у вигляді редагування потрібних полів.

У формі, яка містить інформацію про дошку, потрібно перевірити правильність введеної інформації та відобразити відповідні попередження про помилки у разі їх виникнення.

Інформація про дошку буде відображатися в окремому вікні. Власники зможуть редагувати існуючі дошки. Для цього авторизованим користувачам

доведеться відкрити дошку та натиснути кнопку «редагувати», після чого з’явиться форма для редагування дошки. За потреби користувач може змінити будь-які доступні поля.

Опишемо усі доступні власнику процеси за допомогою UML діаграми діяльності. Діаграма діяльності процесу редагування дошки власником відображена на рисунку 2.9.

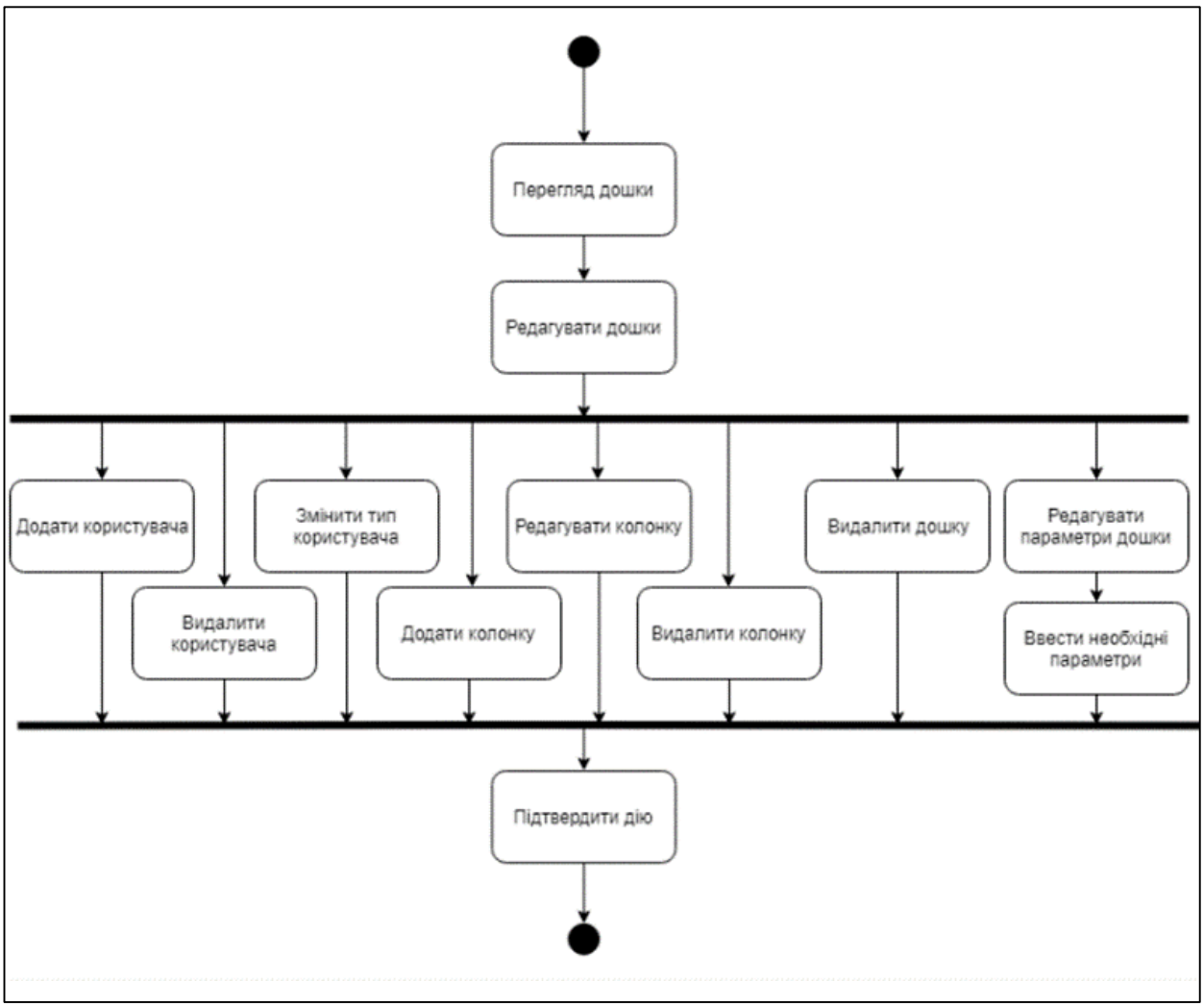


Рисунок 2.9 – Діаграма діяльності процесу редагування дошки власником

2.6 Моделювання структури даних

Для того, щоб увесь функціонал, що був сформований вище міг бути реалізований в інформаційній системі управління проєктами необхідно

змоделювати відповідну структуру бази даних. База повинна містити інформацію про користувачів, дошки, колонки та завдання.

Розглянемо детально інформацію, яка буде використовуватися для представлення основних сутностей бази даних. Для представлення користувача буде використовуватися наступна інформація: ім’я, прізвище, електронна пошта, пароль та зображення. Представлення дошки описується назвою, описом та фоном, колонка – назвою, кольором та позицією на дошці. Представлення завдання описується назвою, описом, пріоритетністю, статусом виконання, оцінкою часу виконання та фактичним часом виконання.

Наступним кроком є введення атрибутів для опису вищезазначених сутностей:

- користувач (ім’я користувача, прізвище користувача, електронна пошта користувача, пароль користувача, зображення користувача);
- дошка (назва дошки, опис дошки, фон дошки);
- колонка (назва колонки, колір колонки, позиція колонки);
- завдання (назва завдання, опис завдання, пріоритетність завдання, статус виконання завдання, оцінка часу виконання завдання, фактичний час виконання завдання);

Отже, універсальне відношення буде мати такий вигляд:

R (ім’я користувача, прізвище користувача, електронна пошта користувача, пароль користувача, зображення користувача, назва дошки, опис дошки, фон дошки, назва колонки, колір колонки, позиція колонки, назва завдання, опис завдання, пріоритетність завдання, статус виконання завдання, оцінка часу виконання завдання, фактичний час виконання завдання).

Ступінь універсального відношення – 17. Універсальне відношення вміщує в собі всі атрибути, що будуть використовуватись в базі даних.

На основі цієї інформації створимо ER-модель бази даних. ER-модель бази даних інформаційної системи управління проєктами зображено на рисунку 2.10.

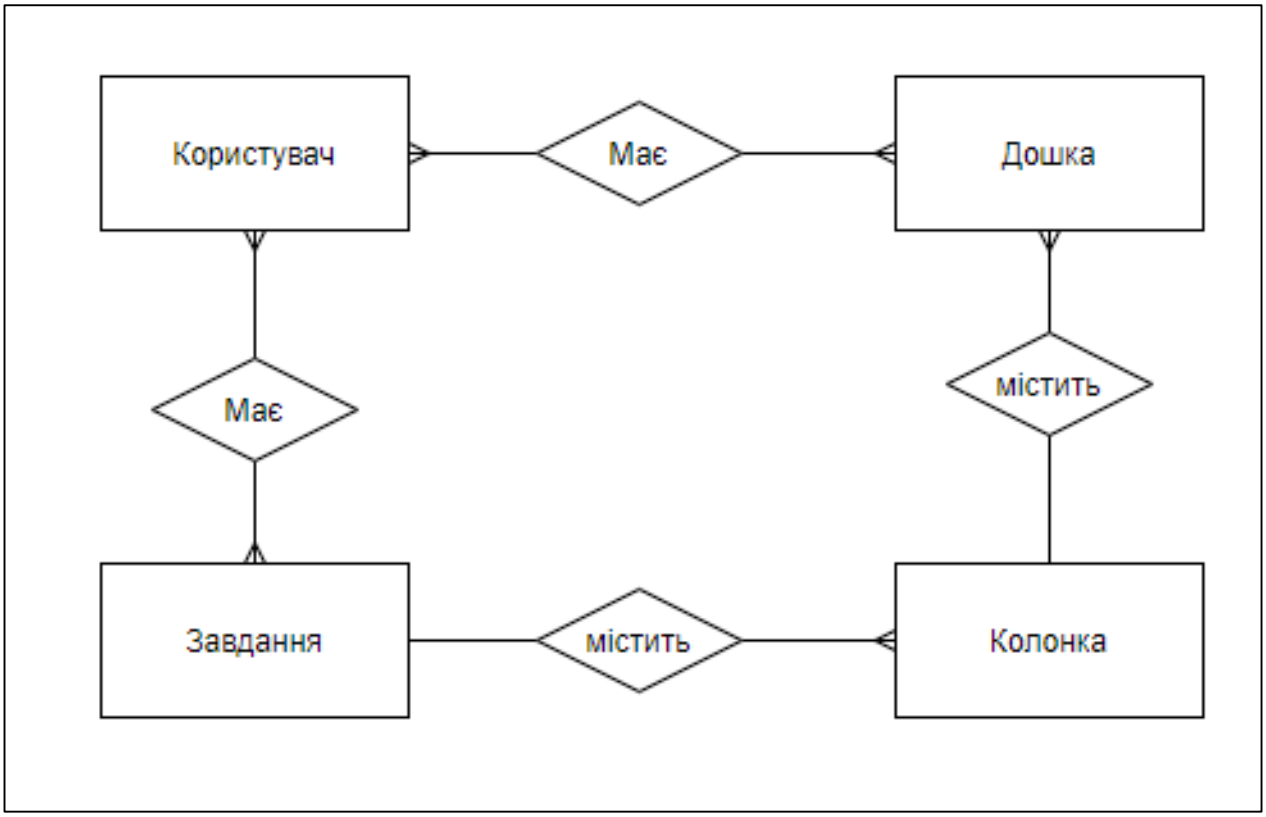


Рисунок 2.10 – ER-модель бази даних інформаційної системи управління проєктами

Характеристики зв’язків бази даних управління проєктами представлено у таблиці 2.1.

Таблиця 2.1 – Характеристики зв’язків бази даних управління проєктами

Ім’я сутності 1	Ім’я сутності 2	Тип зв’язку	Ім’я зв’язку	Клас належності
Користувач	Дошка	Б:Б	Має	Обов’язковий
Дошка	Колонка	Б:1	Містить	Обов’язковий
Колонка	Завдання	Б:1	Містить	Обов’язковий
Завдання	Користувач	Б:Б	Має	Можливий

2.7 Висновки

Отже, в процесі проєктування інформаційної системи управління проєктами було визначено загальну архітектуру системи, та архітектури клієнтської та серверної частин.

Були визначені типи користувачів системи:

- власник;
- керівник;
- робітник.

Були сформовані моделі їх поведінки та описаний функціонал, який буде доступний кожному з типів користувачів та методи доступу до цього функціоналу. Було визначено сутності бази даних:

- користувач;
- дошка;
- колонка;
- завдання.

Після опису сутностей, було сформовано універсальне відношення R, та побудована ER-модель бази даних.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

3.1 Реалізація бази даних

Для створення бази даних на основі отриманої в попередньому розділі інформації найефективніше буде використовувати реляційну модель бази даних, оскільки вона надає змогу уникнути надлишковості, та має широкий вибір систем керування баз даних.

Для системи управління проєктами буде використовуватись СУБД MySQL. MySQL - це безкоштовне програмне забезпечення з відкритим кодом. MySQL працює на багатьох системних платформах, включаючи Linux , macOS , Microsoft Windows , Symbian та більше 20 інших [23, 24].

На основі проведеного моделювання створимо UML діаграму класів бази даних. UML діаграма класів бази даних інформаційної системи управління проєктами зображена на рисунку 3.1.

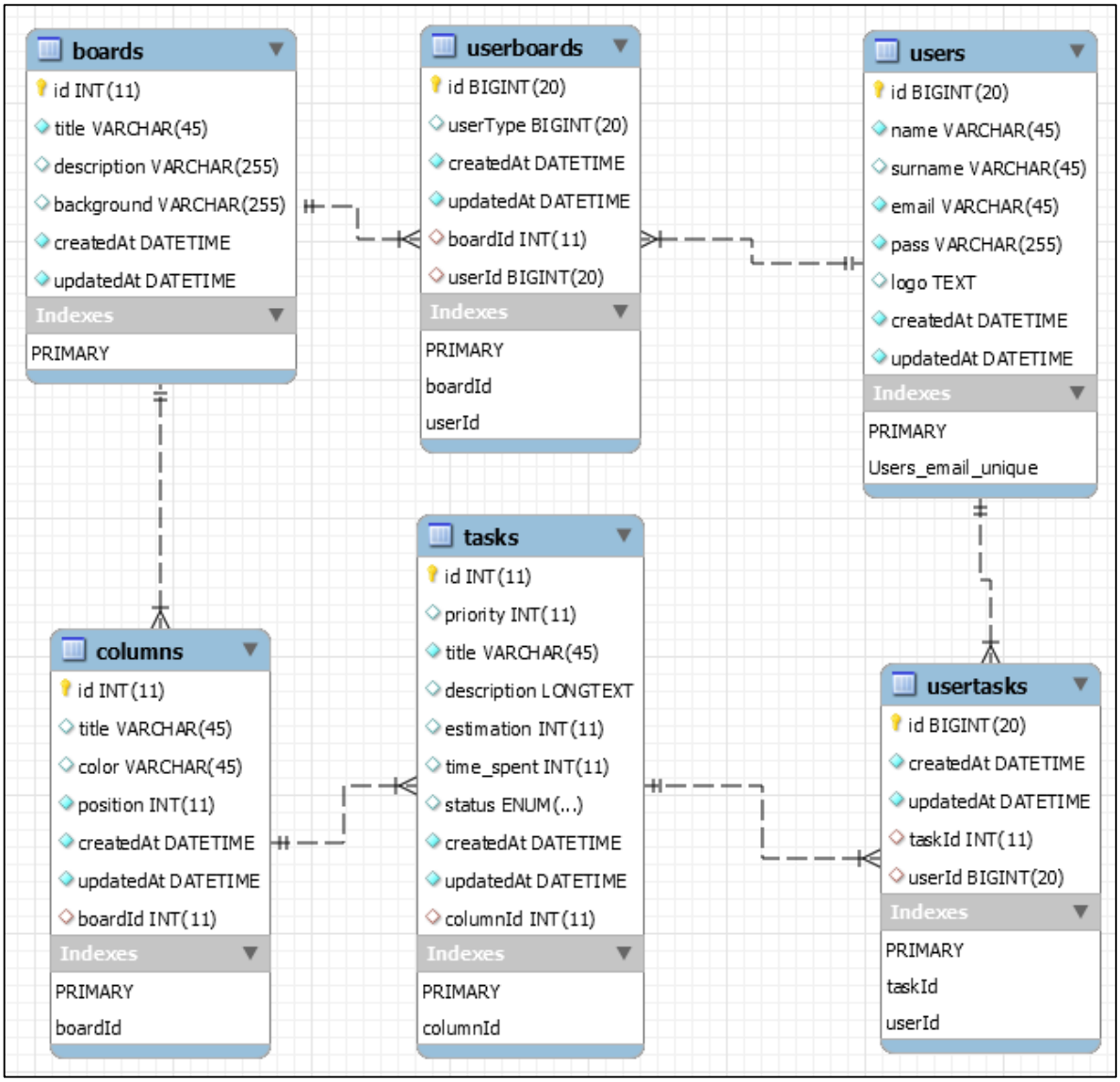


Рисунок 3.1 – UML діаграма класів бази даних інформаційної системи управління проєктами

3.2 Реалізація серверної частини

Для початку потрібно створити кореневий файл серверної частини «index.js», куди в подальшому будуть підключатися усі потрібні модулі. Реалізація кореневого файлу зображена на рисунку 3.2.

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const env = process.env.NODE_ENV || 'dev';
const config = require('./config/' + env + '.js');
const port = process.env.PORT || config.port;

const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

app.use(cors());


app.set('port', port);

app.listen(app.get('port'), () => console.log(`Server started at ${app.get('port')}`));
```

Рисунок 3.2 – Реалізація кореневого файлу

Для коректного запуску серверної частини в різних середовищах, потрібно створити файли конфігурації для кожного середовища. Було створено файл конфігурації для локальної розробки. В файлі будуть зберігатися інформація про базу даних та порт. Файл конфігурації сервера для локального середовища зображено на рисунку 3.3.

```
1   const port = 3000;
2
3   const db = {
4       dialect: 'mysql',
5       host: 'localhost',
6       database: 'workmanager',
7       username: 'root',
8       password: 'root',
9   }
10
11  module.exports = {
12      db,
13      port,
14  }
```

Рисунок 3.3 – Файл конфігурації сервера для локального середовища

Наступним кроком буде підключення бази даних до серверу. Для цього буде використано ORM Sequelize.js. Використання Sequelize.js допомагає пришвидшити розробку програмного забезпечення та уникати SQL ін'єкцій [25]. ORM перетворює реляційне представлення бази даних в більш зручне для використання в Node.js – об'єктне. Sequelize.js вимагає чіткого опису усіх таблиць бази в об'єктному вигляді. Тому потрібно створити моделі кожної таблиці та за допомогою методів Sequelize.js підключити їх до серверу [26, 27]. На рисунку 3.4 зображено модуль підключення бази даних.

```
const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');

const env = process.env.NODE_ENV || 'dev';
const config = require('../../config/' + env + '.js').db;

const basename = path.basename(__filename);
const db = {};

let sequelize;

sequelize = new Sequelize(config.database, config.username, config.password, config);

fs
    .readdirSync(__dirname + '/models')
    .filter((file) => (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-3) === '.js'))
    .forEach((file) => {
        const model = sequelize.import(path.join(__dirname + '/models', file));
        db[model.name] = model;
    });

Object.keys(db).forEach((modelName) => {
    if (db[modelName].associate) {
        db[modelName].associate(db);
    }
});

db.sequelize = sequelize;
db.Sequelize = Sequelize;

module.exports = db;
```

Рисунок 3.4 – Модуль підключення бази даних

Далі необхідно налаштувати роботу авторизації. За основу буде взято бібліотеки passport.js та bcrypt.js. За допомогою та bcrypt.js створимо методи для хешування створення та перевірки токенів користувача. Для посилення безпеки даних користувачів буде використовуватись 2 види токенів:

– токен доступу;

– токен оновлення.

Схема використання токенів наступна. Користувач авторизується в системі, передаючи логін і пароль на сервер. Сервер повертає два токена і час їх життя. Клієнтська частина зберігає токени і використовує токен доступу для подальших запитів. Коли час життя токена доступу добігає кінця , додаток використовує токен оновлення, щоб оновити обидва токена і продовжити використовувати новий токен доступу [28]. Для створення та перевірки токенів використовуються «pem» файли, в яких зберігається ключі для токенів [29]. Реалізація методів для створення та перевірки токенів зображена на рисунку 3.5.

```
const cert = fs.readFileSync('private.pem');
const certRefresh = fs.readFileSync('private-refresh.pem');

module.exports.verify = function verify(token) {
    return new Promise((resolve, reject) => {
        jwt.verify(token, cert, (err, decoded) => {
            if (err) {
                reject(err);
            } else {
                resolve(decoded);
            }
        });
    });
};

module.exports.verifyRefresh = function verify(token) {
    return new Promise((resolve, reject) => {
        jwt.verify(token, certRefresh, (err, decoded) => {
            if (err) {
                reject(err);
            } else {
                resolve(decoded);
            }
        });
    });
};

module.exports.sign = function sign(object, options) {
    return jwt.sign(object, cert, options);
};
module.exports.signRefresh = function sign(object, options) {
    return jwt.sign(object, certRefresh, options);
};
```

Рисунок 3.5 – Реалізація методів для створення та перевірки токенів

За допомогою passport.js створимо метод для авторизації користувачів. Метод шукає користувача в базі за електронною поштою і порівнює пароль який прийшов з запитом реєстрації з паролем, який зберігається в базі даних. Оскільки паролі в базі даних будуть зберігатись у вигляді хешу, то для порівняння буде використовуватись метод «compareSync» бібліотеки bcrypt.js. Реалізацію методу для авторизації користувачів зображено на рисунку 3.6.

```
const bcrypt = require('bcryptjs');
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const db = require('../database');

passport.use(new LocalStrategy(
    {
        usernameField: 'email',
        passwordField: 'pass'
    }, (async (username, password, done) => {
        try {
            const userPassword = password.trim();

            const user = await db.User.findOne({
                where: { email: username },
            });

            if (!user) {
                throw new Error('You entered an invalid email.');
            }
            const { id, pass: passwordFromDb } = user;

            if (!bcrypt.compareSync(userPassword, passwordFromDb)) {
                throw new Error('You entered an invalid password.');
            }

            return done(null, id);
        } catch (error) {
            return done(error, null);
        }
    })
));
```

Рисунок 3.6 – Реалізація методу для авторизації користувачів

Наступним кроком буде створення сервісів, які будуть взаємодіяти з базою даних. Для кожної моделі, не враховуючи допоміжні моделі, які використовуються для створення зв’язку типу «багато до багатьох», буде реалізовано свій сервіс, а також сервіс для авторизації та реєстрації.

Сервіс для авторизації та реєстрації містить в собі 4 методи.

Метод register використовується для додавання нового користувача до бази даних. Метод повертає об’єкт, який містить інформацію про нового користувача. Реалізація методу register зображена на рисунку 3.7.

```
async function register(email, name, surname, pass) {
    const passwordHash = bcrypt.hashSync(pass.trim(), 10);

    const newUser = {
        email,
        name,
        surname,
        pass: passwordHash,
    };

    return db.User.create(newUser);
}
```

Рисунок 3.7 – Реалізація методу register

Метод verifyToken перевіряє токен і якщо він дійсний, та повертає користувача з бази даних, або значення «false», якщо користувача не існує. Реалізація методу register зображена на рисунку 3.8.

```
async function verifyToken(token) {
    const decoded = await verify(token);
    const { id: userId } = decoded;

    const user = await db.User.findOne({
        where: { id: userId },
        attributes: ['id'],
    });
    return user.id ? user.id : false;
}
```

Рисунок 3.8 – Реалізація методу verifyToken

Метод generateTokens створює та повертає токени доступу та оновлення, якщо користувач з ідентифікатором, який приймає метод, існує. Токен доступу

отримує час життя 5 хвилин, а токен оновлення 10 днів. Реалізація методу generateTokens зображена на рисунку 3.9.

```
async function generateTokens(userId) {

    const user = await db.User.findOne({
        where: { id: userId },
        attributes: ['id'],
    });

    if (user) {
        const accessToken = await sign({ id: userId }, { expiresIn: '5m' });
        const refreshToken = await signRefresh({ id: userId }, { expiresIn: '10d' });

        return {
            token: accessToken,
            refresh: refreshToken,
        };
    }

    return false;
}
```

Рисунок 3.9 – Реалізація методу generateTokens

Метод refreshAccessToken перевіряє, чи токен оновлення дійсний та повертає метод generateTokens. У випадку якщо токен не дійсний метод повертає помилку з відповідним текстом. Реалізація методу refreshAccessToken зображена на рисунку 3.10.

```
async function refreshAccessToken(refreshToken) {
    const decoded = await verifyRefresh(refreshToken);
    const { id: userId } = decoded;

    if (userId) {
        return generateTokens(userId);
    } else {
        throw new Error('Refresh token invalid');
    }
}
```

Рисунок 3.10 – Реалізація методу refreshAccessToken

Сервіс для управління дошками містить в собі 11 методів:

– createBoard;
– addUserToBoard;
– getBoards;
– getBoard;
– getUserType;
– deleteBoard;
– editBoard;
– deleteUserFromBoard;
– getBoardUsers;
– editUserType;
– getAllBoardData.

Дані методи реалізовують весь необхідний для керування дошками функціонал: створення, видалення, редагування дощок, додавання, видалення користувачів, зміна типу користувачів, отримання спуску дощок користувача та отримання усієї інформації про дошку.

Сервіс для управління колонками містить в собі 5 методів:

– createColumn;
– getColumns;
– changeColumnPosition;
– editColumn;
– deleteColumn.

Дані методи реалізовують весь необхідний для керування колонками функціонал: створення, видалення, редагування колонок, отримання списку колонок для дошки та зміна позиції колонки.

Сервіс для управління завданнями містить в собі 7 методів:

– createTask;
– changeColumn;
– deleteTask;

– getTask;
– editTask;
– editTaskUsers;
– changeTaskStatus.

Дані методи реалізовують весь необхідний для керування завданнями функціонал: створення, видалення, редагування завдань, зміна колонки завдання, зміна статусу завдання, редагування користувачів завдань та отримання інформації про завдання.

Останній сервіс допомагає керувати користувачами. Він містить 4 методи:

– getUser;
– findUsers;
– changePassword;
– editUser.

Дані методи реалізовують весь необхідний для керування користувачами функціонал: створення, редагування, пошук користувачів та зміна паролю користувача. Пошук користувача відбувається за електронною адресою.

Наступним кроком буде налаштування маршрутизації. Створимо головний файл маршрутизації, в який будуть імпортуватися усі файли маршрутів. Головний файл маршрутизації буде підключений в кореневий файл серверної частини «index.js». Маршрути додатку можна умовно розділити, на два типи: публічні і приватні. Публічні маршрути доступні без використання токену доступу, а приватні виконуються тільки після перевірки токену. Реалізацію головного файлу маршрутизації зображено на рисунку 3.11.

```
const router = require('express-promise-router')();

const authRoutes = require('./auth.routes');       LyshakS, 5 months ago
const boardRoutes = require('./board.routes');
const columnRoutes = require('./column.routes');
const taskRoutes = require('./task.routes');
const userRoutes = require('./user.routes');

const checkToken = require('../controllers/auth.controller').checkToken;

router.use('/auth', authRoutes);
router.use('/api/board', checkToken, boardRoutes);
router.use('/api/column', checkToken, columnRoutes);
router.use('/api/task', checkToken, taskRoutes);
router.use('/api/user', checkToken, userRoutes);

module.exports = router;
```

Рисунок 3.11 – Реалізація головного файлу маршрутизації

Перевірка токену здійснюється методом checkToken, який реалізований в контролері авторизації. Він отримує токен з заголовків HTTP за ключем «x-access-token». При хибному токені метод повертає помилку 401. Якщо ж токен дійсний то в тіло запиту буде доданий параметр «userId» який вказує на користувача, який робив цей запит. Реалізацію методу checkToken зображено на рисунку 3.12.

```
async function checkToken(req, res, next) {
    const { headers } = req;
    const token = headers['x-access-token'];

    const user = await authService.verifyToken(token);
    if (!user) {
        return next(new HttpError(401, 'Unauthorized'));
    }

    req.body.userId = user;
    next()
}
```

Рисунок 3.12 – Реалізація методу checkToken

Головний файл маршрутизації містить маршрути для керування дошками, колонками, завданнями, користувачами та маршрути авторизації. Ці описані в окремих файлах. Для кожного маршруту існує свій метод в контролерах, який взаємодіє з сервісами та перевіряє наявність потрібних даних. Для прикладу опишемо метод контролера зміни статусу завдання. Спочатку метод отримує ідентифікатори завдання та користувача, а також новий статус завдання. Якщо ці дані існують викликається метод changeTaskStatus, який реалізований в сервісі для управління завданнями. Якщо сервіс відпрацював коректно метод контролера повертає результат роботи сервісу, в іншому випадку він повертає помилку. Реалізацію методу контролера зміни статусу завдання зображено на рисунку 3.13.

```
async function checkToken(req, res, next) {
    const { headers } = req;
    const token = headers['x-access-token'];

    const user = await authService.verifyToken(token);
    if (!user) {
        return next(new HttpError(401, 'Unauthorized'));
    }

    req.body.userId = user;
    next()
}
```

Рисунок 3.13 – Реалізація методу контролера зміни статусу завдання

Останнім етапом реалізації серверної частини інформаційної системи управління проєктами буде створення методу, який буде відловлювати та обробляти усі помилки в додатку. Метод буде підключений в кореневий файл серверної частини «index.js». Реалізацію методу відловлювання помилок зображено на рисунку 3.14.

```
module.exports = (err, req, res, next) => {
    console.log(err);
    if (err.name === 'TokenExpiredError' || err.name === 'JsonWebTokenError') {
        res.status(401);
    } else {
        res.status(500);
    }
    res.send({ error: err.message });
};
```

Рисунок 3.14 – Реалізація методу відловлювання помилок

3.3 Реалізація клієнтської частини

Першим кроком реалізації клієнтської частини інформаційної системи управління проєктами буде реалізація компонентів які будуть використовуватись на більшості сторінок системи. Такими компонентами є підвал та шапка сайту. Підвал містить в собі випадаюче меню з списком усіх доступних локалізацій в системі. Користувач може змінити мову системи вибравши з списку потрібну та натиснувши на неї. На рисунку 3.15 зображено підвал з відкритим випадаючим меню вибору локалізації.

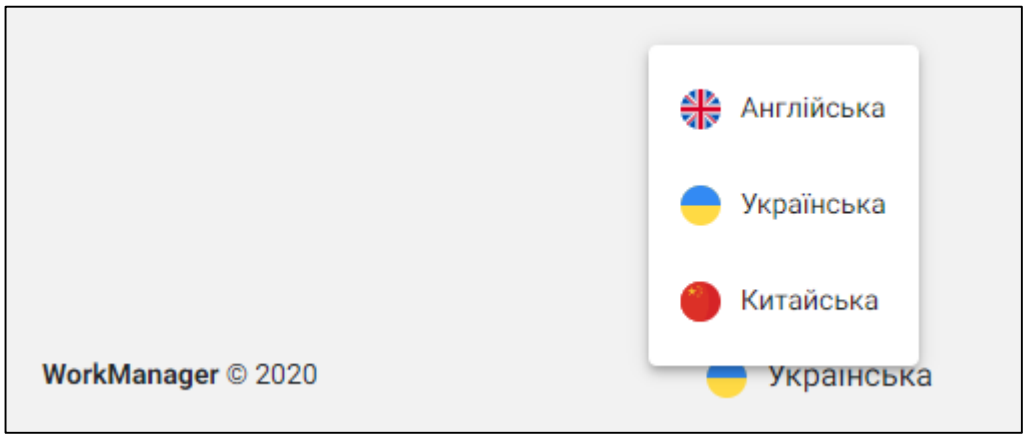


Рисунок 3.15 – Підвал з відкритим випадаючим меню вибору локалізації

Шапка буде відображуватись тільки авторизованим користувачам. Він містить в собі логотип, посилання на дошки та фото користувача. При натисненні на фото користувача з’являється випадаюче меню з посиланнями на сторінки

системи, які дозволяють переглянути профіль користувача, редагувати його, змінити пароль та вийти з системи. Шапка з випадаючим меню зображено на рисунку 3.16.

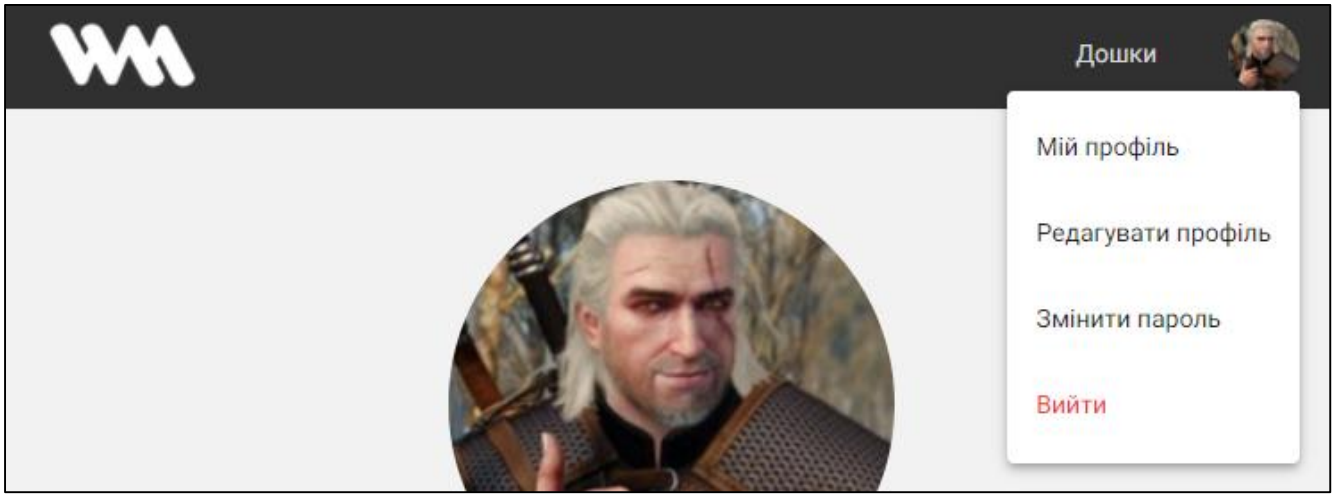


Рисунок 3.16 – Шапка з випадаючим меню

Далі потрібно реалізувати модуль реєстрації. Для реєстрації користувачу потрібно ввести електронну пошту, ім’я та пароль. Усі ці поля є обов’язковими, якщо користувач не введе якісь дані, або введе їх не вірно система сповістить його про це. Процес реєстрації не може бути завершеним поки усі необхідні дані не будуть введені. Вікно реєстрації з прикладом повідомлень системи про хибний ввід даних зображено на рисунку 3.17.

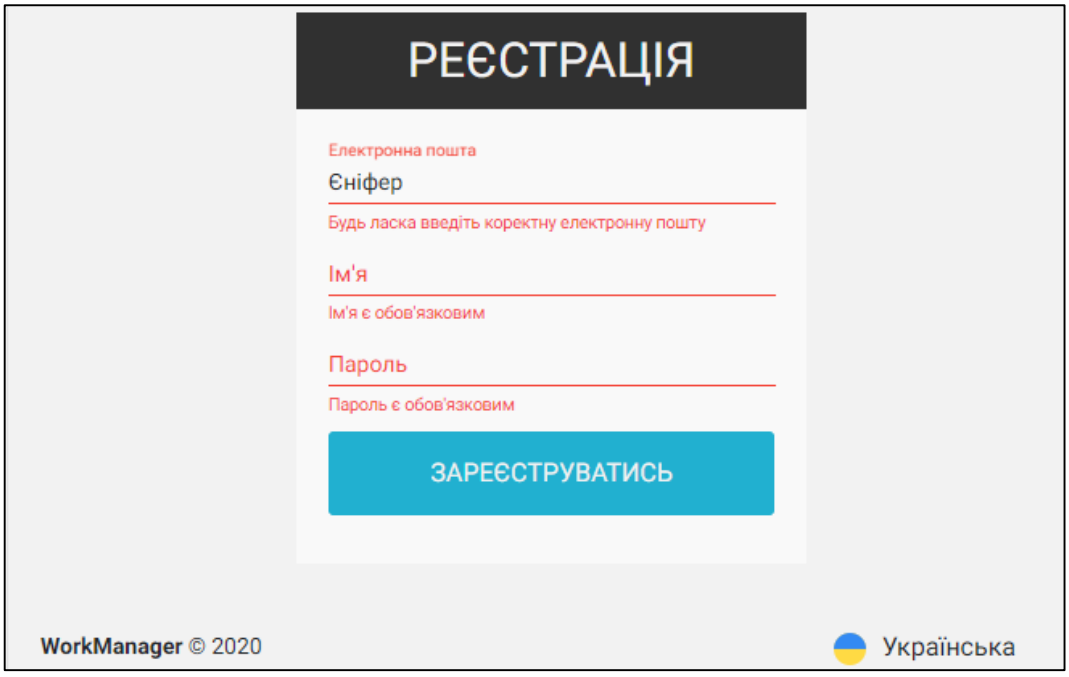


Рисунок 3.17 – Вікно реєстрації з прикладом повідомлень системи про хибний ввід даних

Наступним буде реалізований модуль авторизації. Для авторизації користувачу необхідно ввести електронну пошту та пароль. Ці поля є обов’язковими, якщо користувач не введе якісь дані, або введе їх не вірно система сповістить його про це. Вікно авторизації зображено на рисунку 3.18.

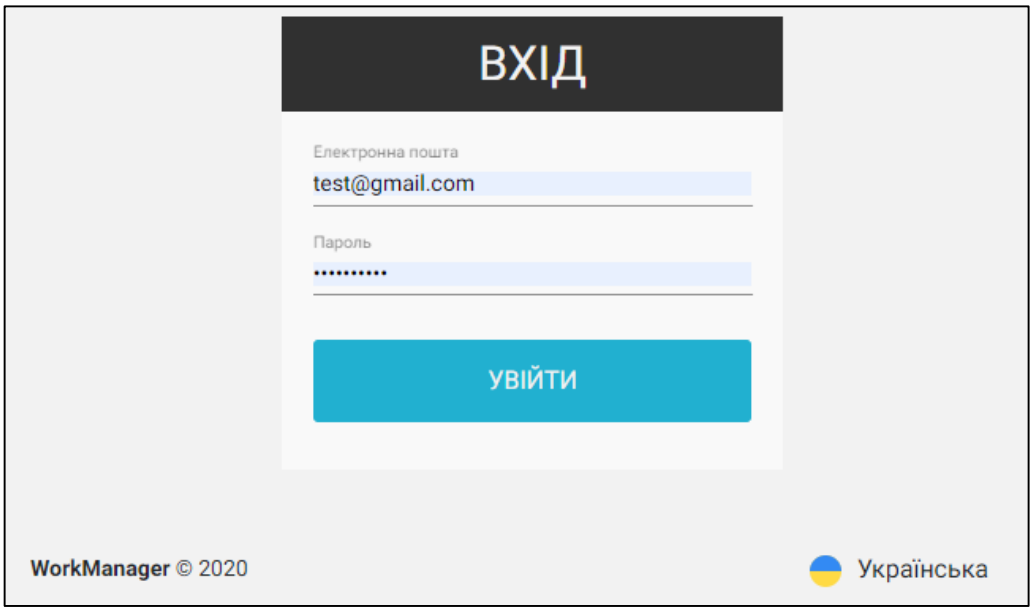


Рисунок 3.18 – Вікно авторизації

Після авторизації користувач потрапляє на сторінку дощок, яка в лівій частині містить меню дощок, а в правій їх перелік. В меню дощок користувач може вибрати фільтр відображення дощок за типом користувача. Тобто, якщо користувач вибере фільтр «менеджерські дошки», то в правій частині сторінки будуть відображенні дошки в яких користувач має тип «керівник». Дошки на сторінці відображаються у вигляді карток, які містять в собі назву дошки та її опис. Сторінка дощок зображена на рисунку 3.19.

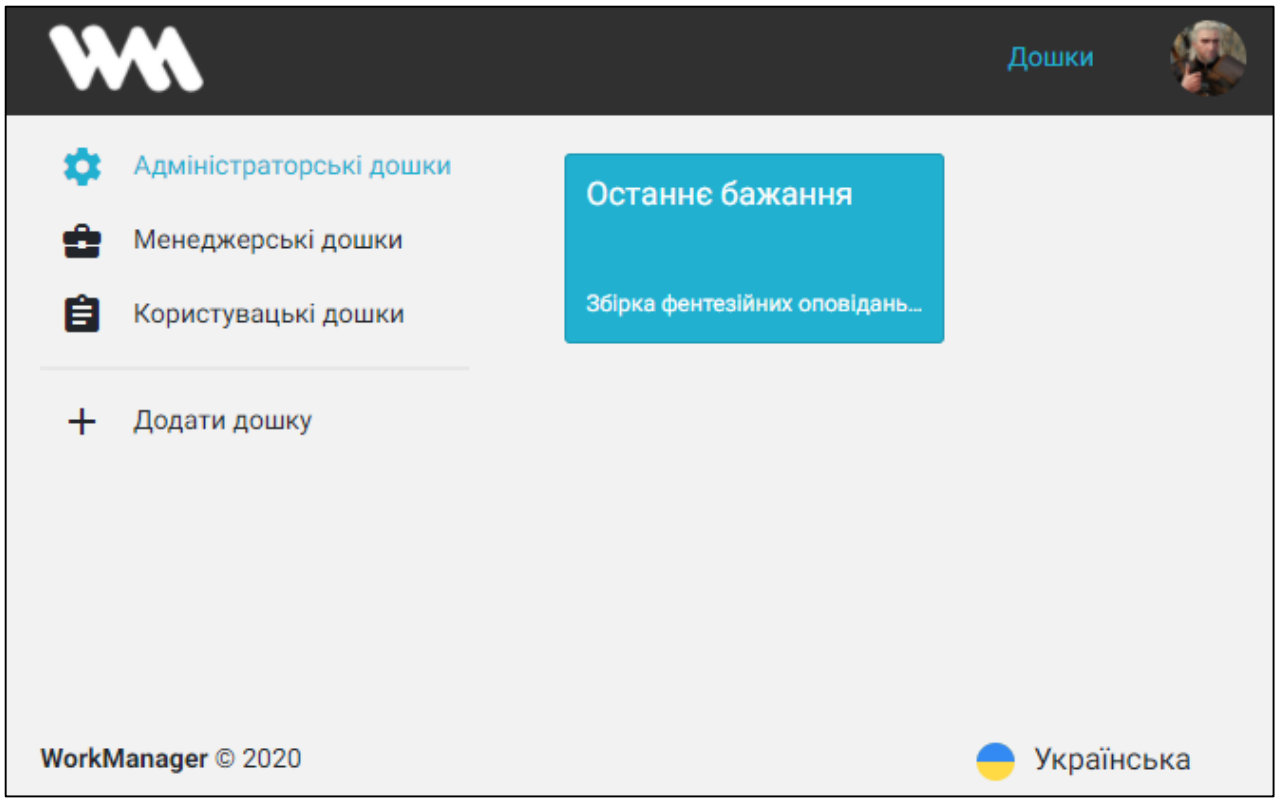


Рисунок 3.19 – Сторінка дощок

Також в меню дощок є пункт «додати дошку», при натисненні користувач потрапить на сторінку створення дошки. Для створення дошки користувачу потрібно вести назву та опис дошки, після чого натиснути на кнопку «створити». Якщо всі дані були ведено вірно, користувач буде перенаправлений на сторінку, щойно створеної дошки. На цій дошці даний користувач буде мати тип «власник». Сторінку створення дошки зображено на рисунку 3.20.

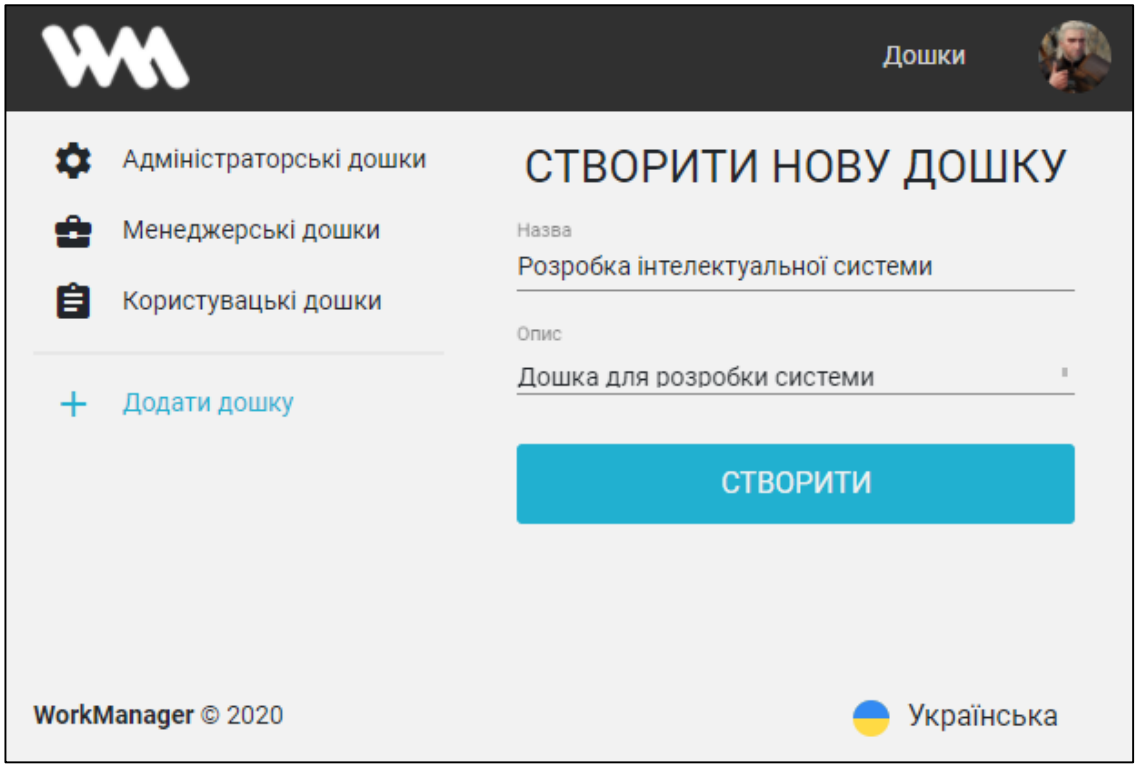


Рисунок 3.20 – Сторінка створення дошки

На новій дошці відображаються назва дошки її опис, які були вказані при створення. Також на сторінці дошки присутні посилання для перегляду статистики, та налаштування дошки. Дані посилання не доступні користувачам з типу робітник. Сторінку нової дошки зображено на рисунку 3.21.

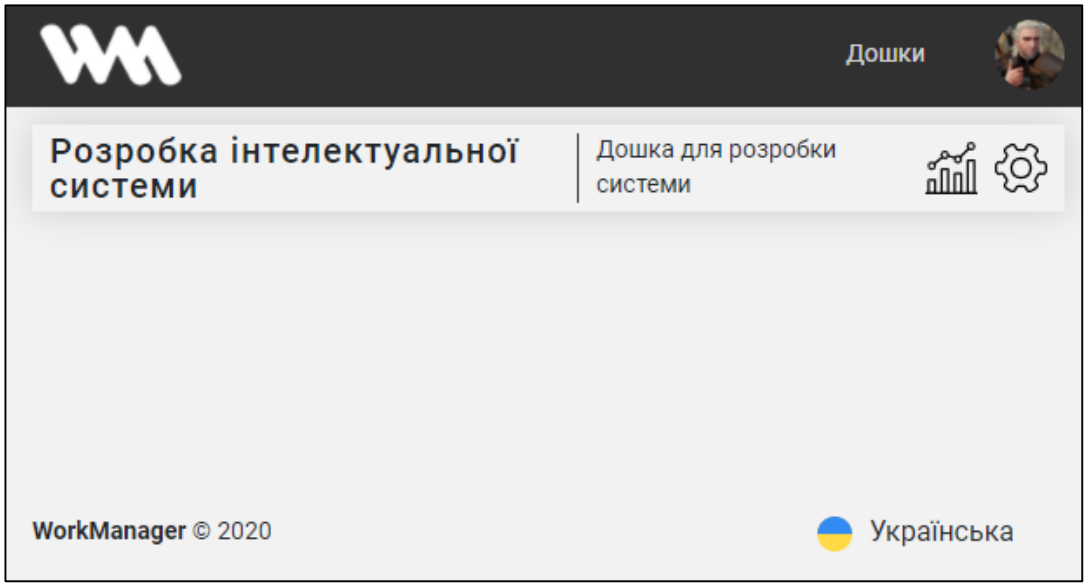


Рисунок 3.21 – Сторінка нової дошки

Натиснувши на кнопку «налаштування» користувач потрапляй на сторінку на сторінку налаштувань дошки, яка має три вкладки: «головна», «користувачі», «колонки». На вкладці «головна» користувач може змінити назву, опис та фон дошки, а також видалити дошку. Видалення дошки доступне тільки користувачам типу «власник». Вкладка «головна» сторінки налаштування зображена на рисунку 3.22.

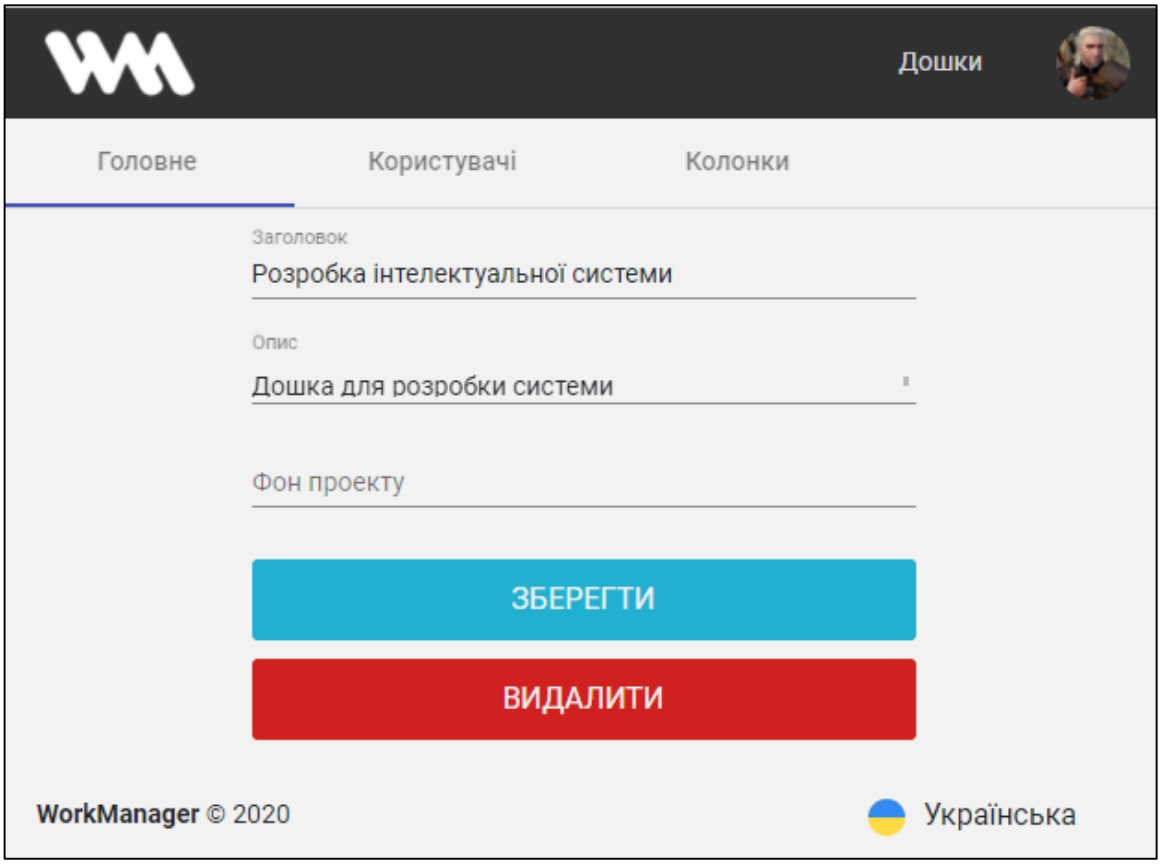


Рисунок 3.22 – Вкладка «головна» сторінки налаштувань

На вкладці «користувачі» відображається список усіх користувачів дошки. В списку вказується ім’я, електронна пошта та тип користувачів. Також присутні кнопки для видалення, додавання користувачів та зміни їх типу. Змінювати тип користувачів може тільки власник дошки. Користувачі типу «керівник» можуть видаляти тільки користувачів типу «робітник». На рисунку 3.23 зображено вкладку «користувачі» сторінки налаштувань дошки.

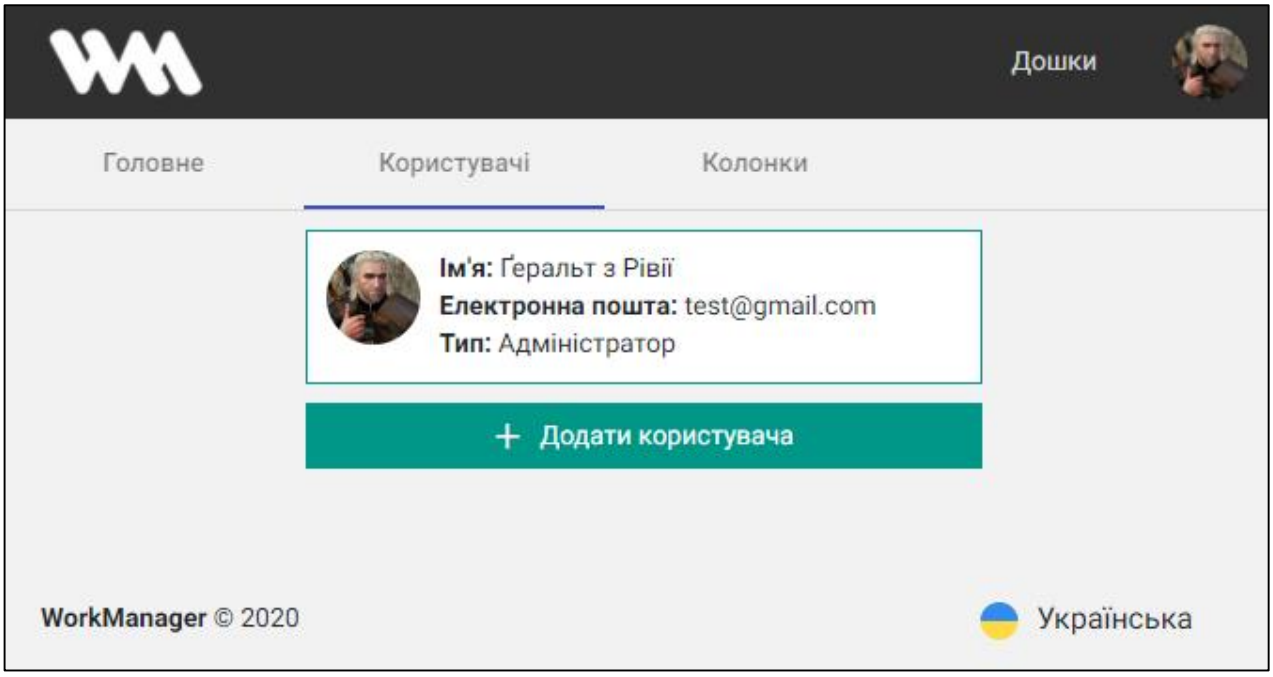


Рисунок 3.23 – Вкладка «користувачі» сторінки налаштувань

Щоб додати нового користувача до дошки потрібно натиснути на кнопку «додати користувача», після чого з’явиться вспливаюче вікно з полями для введення електронної пошти та типу користувача. Після введення необхідних даних потрібно натиснути на кнопку «зберегти», вспливаюче вікно закриється і буде оновлено список користувачів. Вспливаюче вікно додавання користувача зображено на рисунку 3.24.

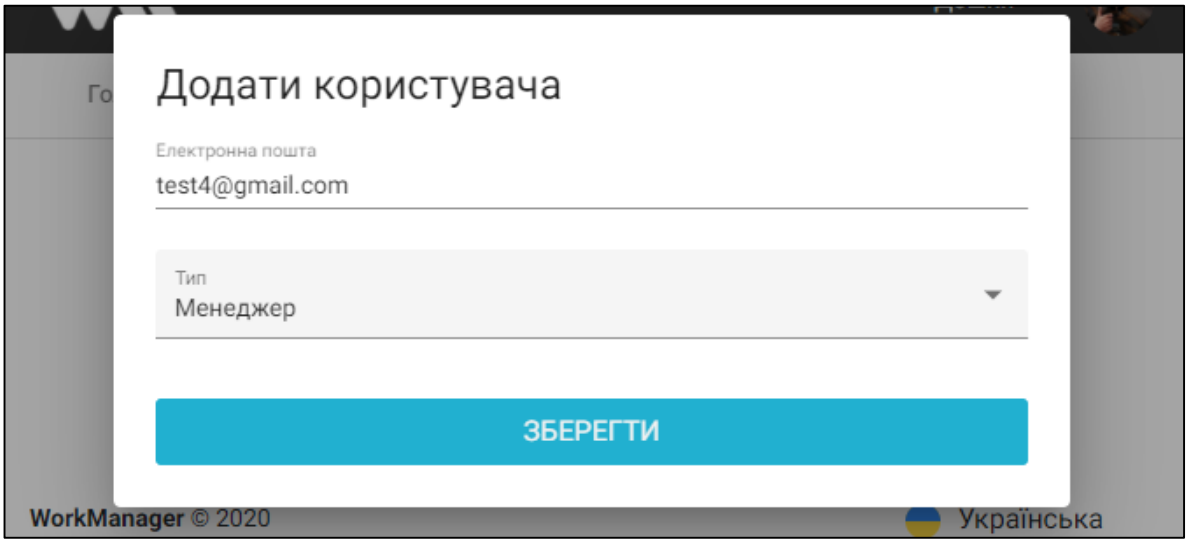


Рисунок 3.24 – Вспливаюче вікно додавання користувача

Для видалення користувача потрібно натиснути на кнопку «видалити», яка розміщена в картці користувача. Для зміни типу користувача потрібно натиснути на кнопку «редагувати», яка також розміщена в картці користувача. Після на тиснення з’явиться вспливаюче вікно зміни типу користувача з полями для електронної пошти та типу користувача. Поле електронної пошти не доступне для зміни в даному вікні. Вспливаюче вікно редагування користувача зображено на рисунку 3.25.

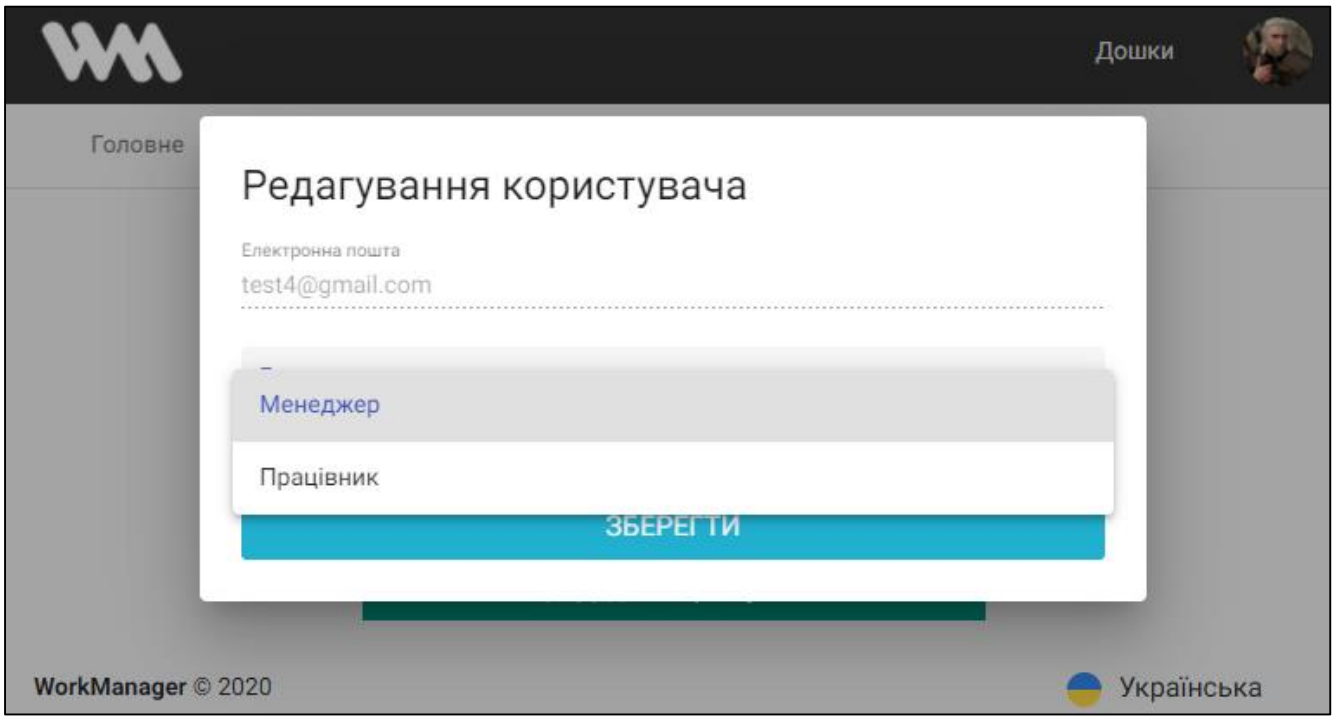


Рисунок 3.25 – Вспливаюче вікно редагування користувача

На вкладці «колонки» відображується список колонок дошки та їх порядок відображення. Також на вкладці колонки знаходиться кнопка додавання нових колонок. Вкладка «колонки» сторінки налаштувань зображена на рисунку 3.26.

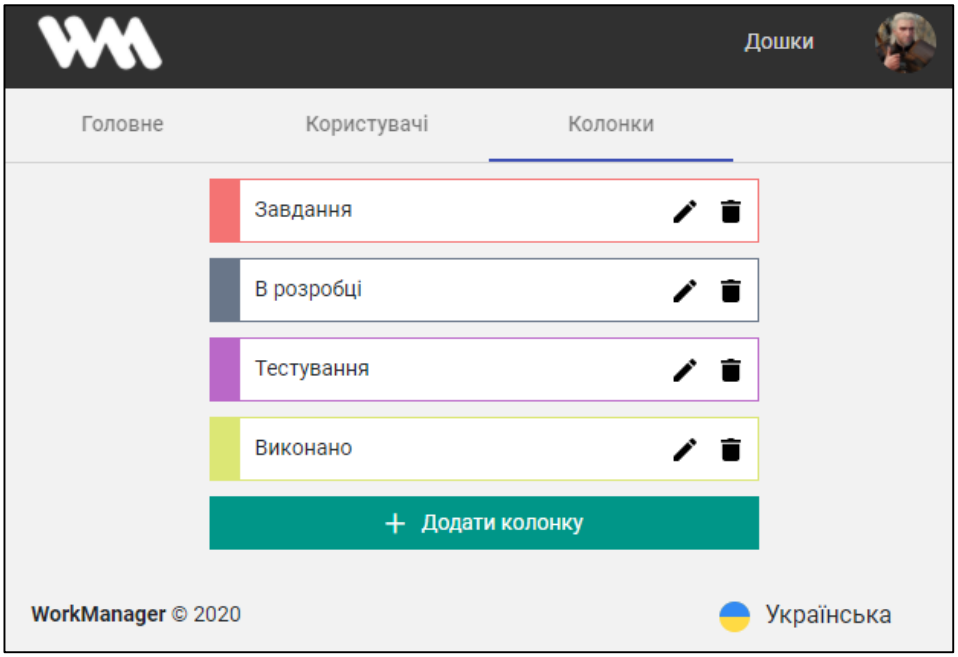


Рисунок 3.26 – Вкладка «колонки» сторінки налаштувань

Для зміни порядку колонок на дошці потрібно перетягнути картку колонки на потрібне місце. Процес перетягування картки колонки зображено на рисунку 3.27.

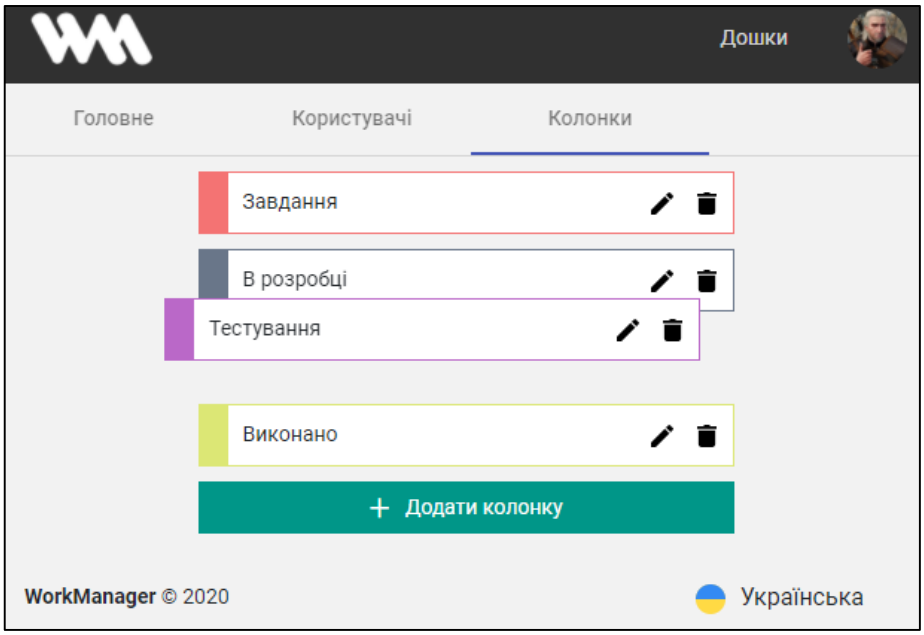


Рисунок 3.27 – Процес перетягування картки колонки

Для видалення колонки потрібно натиснути на кнопку «видалити», яка розміщена в картці колонки. Для редагування колонки потрібно натиснути на

кнопку «редагувати», яка також розміщена в картці колонки. Після на тиснення з’явиться вспливаюче вікно редагування колонки з полями для назви та кольору колонки. Для збереження даних після зміни значень в полях потрібно натиснути на кнопку «зберегти». Після натиснення на кнопку «зберегти» вспливаюче вікно закриється, а список колонок буде оновлено. Вспливаюче вікно редагування колонки зображено на рисунку 3.28.

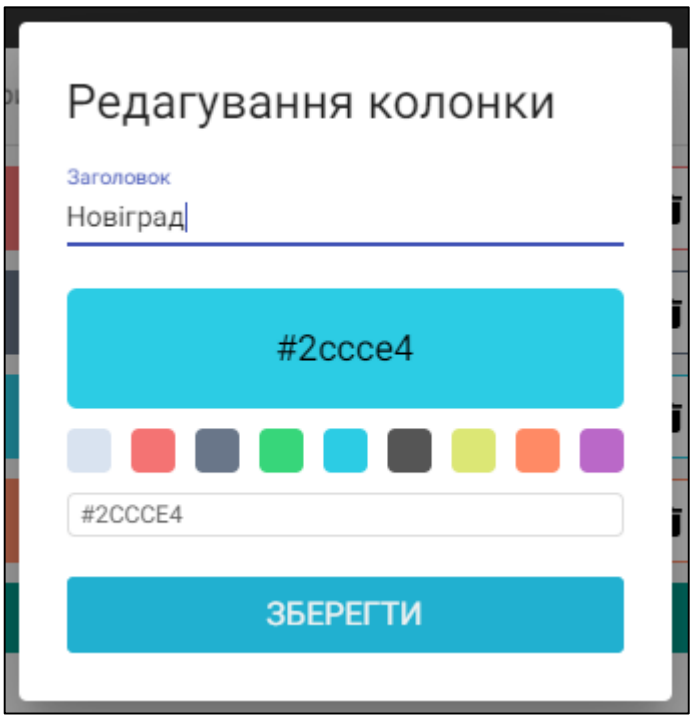


Рисунок 3.28 – Вспливаюче вікно редагування колонки

Для створення нової колонки на вкладці «колонки» потрібно натиснути на кнопку «додати колонку». Після натиснення на кнопку «додати колонку» з’явиться вспливаюче вікно додавання колонки, яке містить форму з полями для введення назви та кольору колонки. Для збереження даних після введення значень в полях потрібно натиснути на кнопку «зберегти». Після натиснення на кнопку «зберегти» вспливаюче вікно закриється, а список колонок буде оновлено. Вспливаюче вікно додавання колонки зображено на рисунку 3.29.

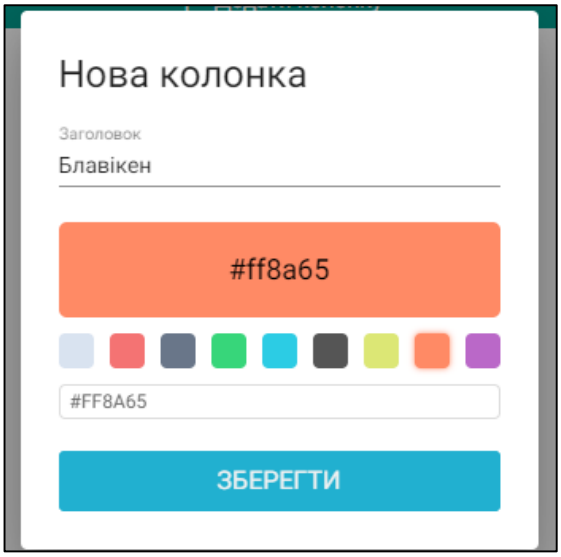


Рисунок 3.29 – Вспливаюче вікно додавання колонки

Після того як колонки були створенні і відсортовані при повернені на сторінку дошки користувач отримає можливість додавати нові завдання. Для цього потрібно натиснути на кнопку «додати завдання» в потрібній колонці на сторінці дошки. Сторінку дошки з колонками зображено на рисунку 3.30.

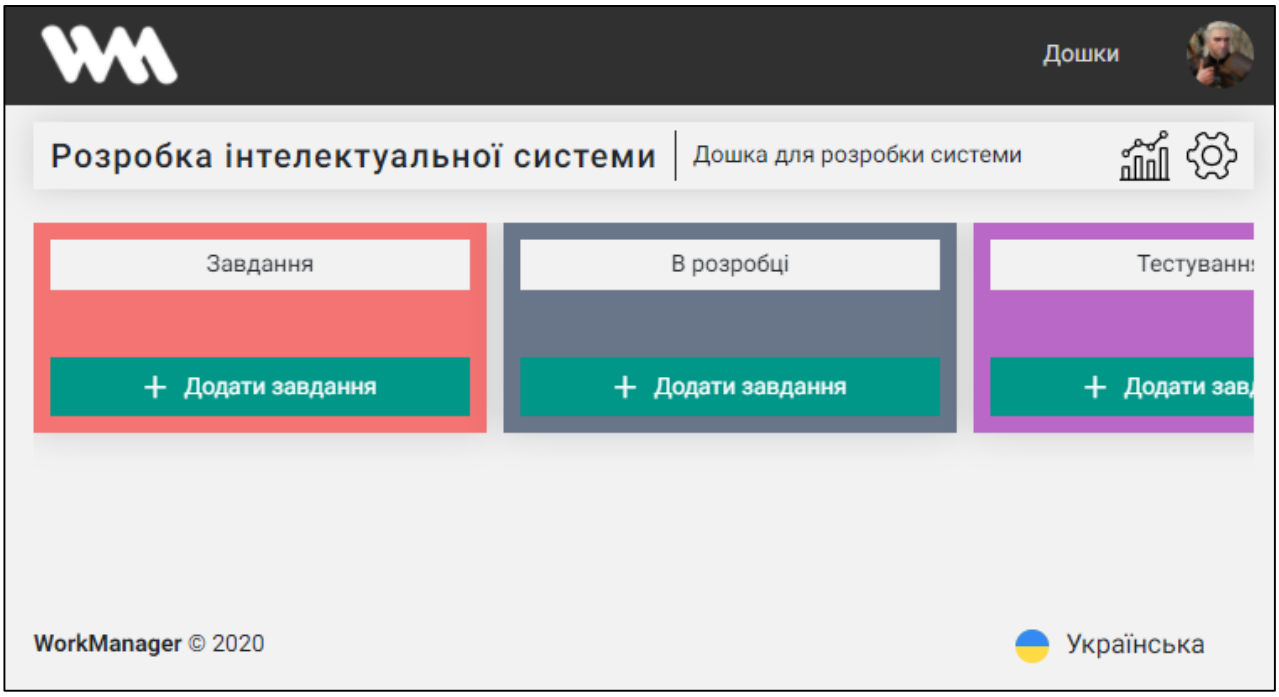


Рисунок 3.30 – Сторінка дошки з колонками

Після натиснення на кнопку «додати завдання» відкриється сторінка додавання завдань, яка містить форму для введення усіх необхідних даних для завдання. Опис завдання вводиться в полі, яке підтримує мову розмітки Markdown. Markdown підтримує введення елементів як:

- заголовок;
- параграф;
- нумерований список;
- маркерований список;
- напівжирний текст;
- текст з стилем курсив;
- блок коду;
- цитата;
- посилання;
- зображення;
- горизонтальна лінія.

Щоб додати користувача до завдання потрібно натиснути на кнопку «додати користувача», після чого з’явиться вспливаюче вікно з полем для введення електронної пошти користувача. Після введення необхідних даних потрібно натиснути на кнопку «зберегти», вспливаюче вікно закриється і буде оновлено список користувачів. Для видалення користувача потрібно натиснути на кнопку «видалити» в картці користувача. Для збереження завдання потрібно натиснути на кнопку «створити». На рисунку 3.31 зображено сторінку створення завдання.

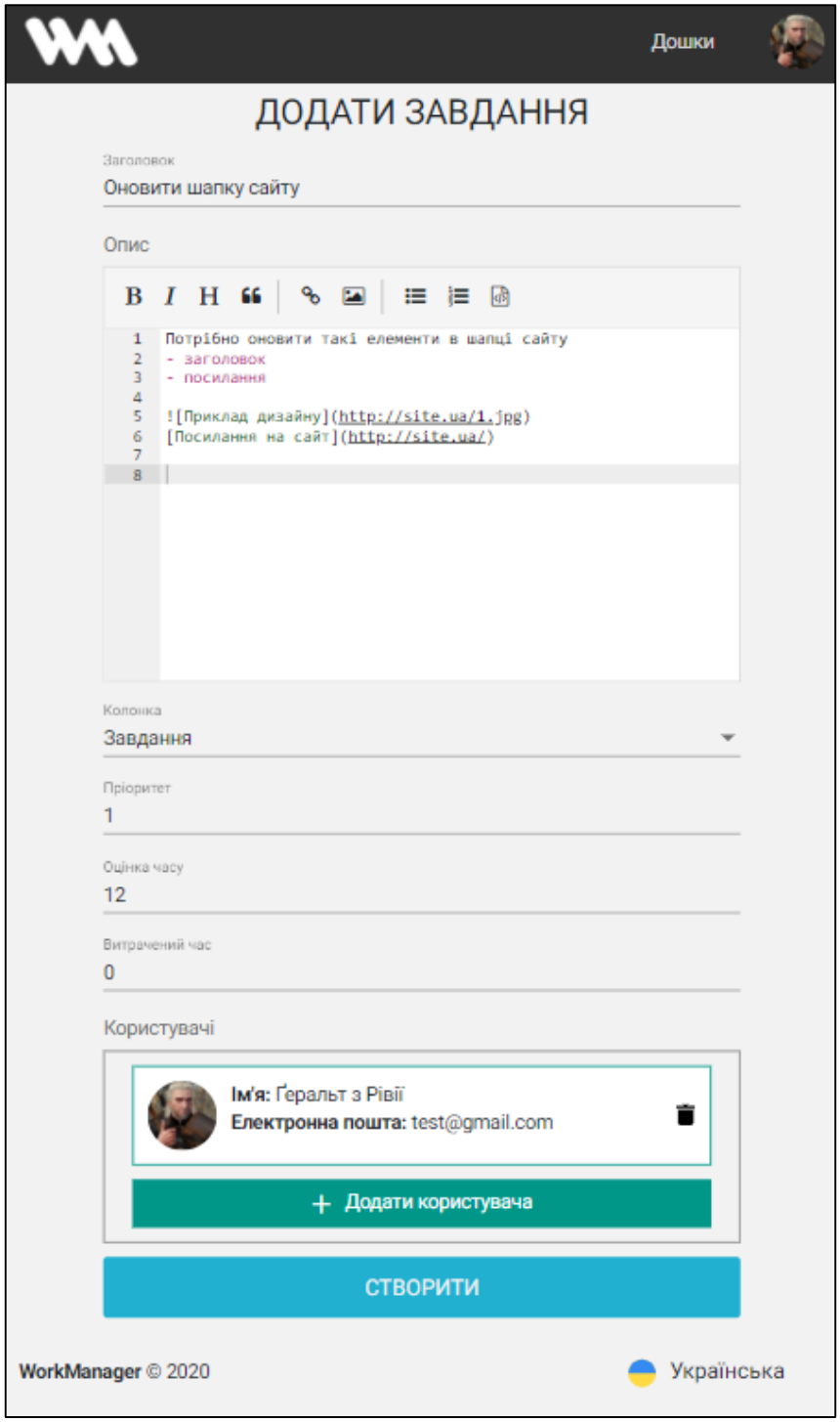


Рисунок 3.31 – Сторінка створення завдання

Після натиснення на кнопку «створити» користувач буде перенаправлений на сторінку дошки, де буде відображатися нове завдання у відповідній колонці. Для зміни колонки для завдання його необхідно перетягнути в потрібну колонку. Процес перетягування завдання в іншу колонку зображено на рисунку 3.32.

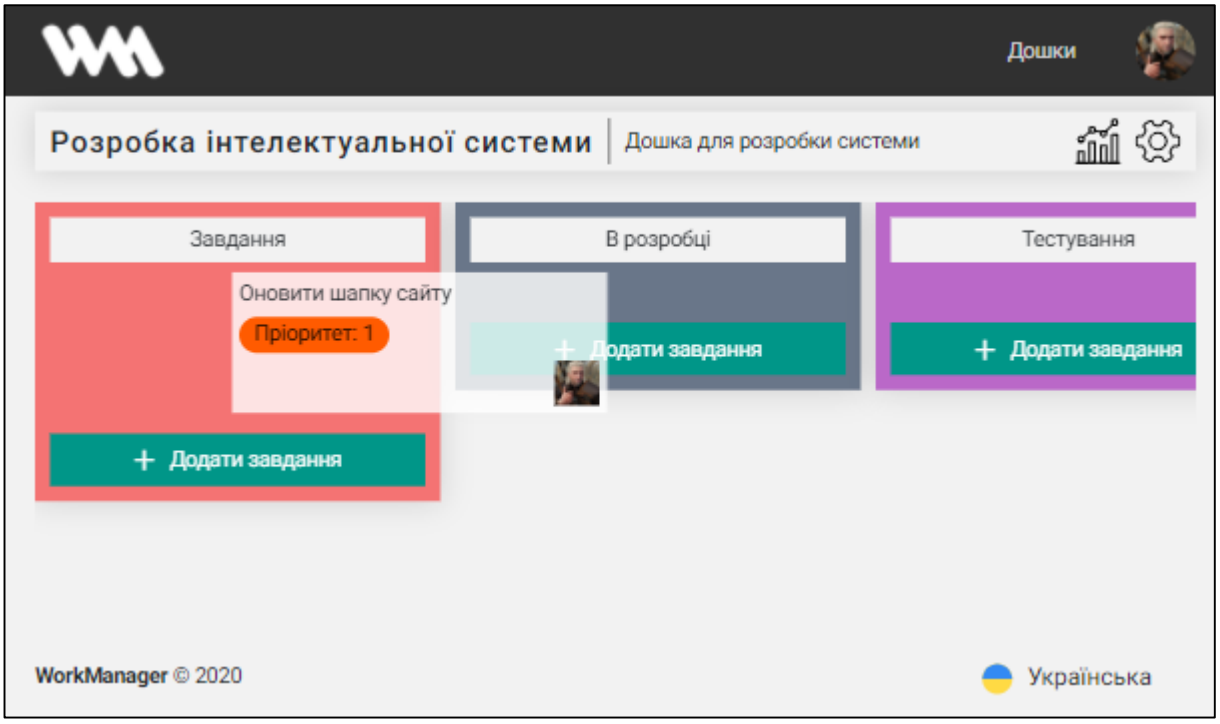


Рисунок 3.32 – Процес перетягування завдання в іншу колонку

Для перегляду деталей завдання потрібно натиснути на його картку, після чого буде відкрито вспливаюче вікно з деталями завдання так кнопками видалення та редагування завдання.

Для видалення завдання користувачу потрібно натиснути на кнопку «видалити», після чого відкриється вспливаюче вікно з полем для введення типу видалення (видалено, провалено, виконано), для підтвердження видалення користувач повинен натиснути кнопку «видалити» в вспливаючому вікні. Вспливаюче вікно видалення завдання зображено на рисунку 3.33.

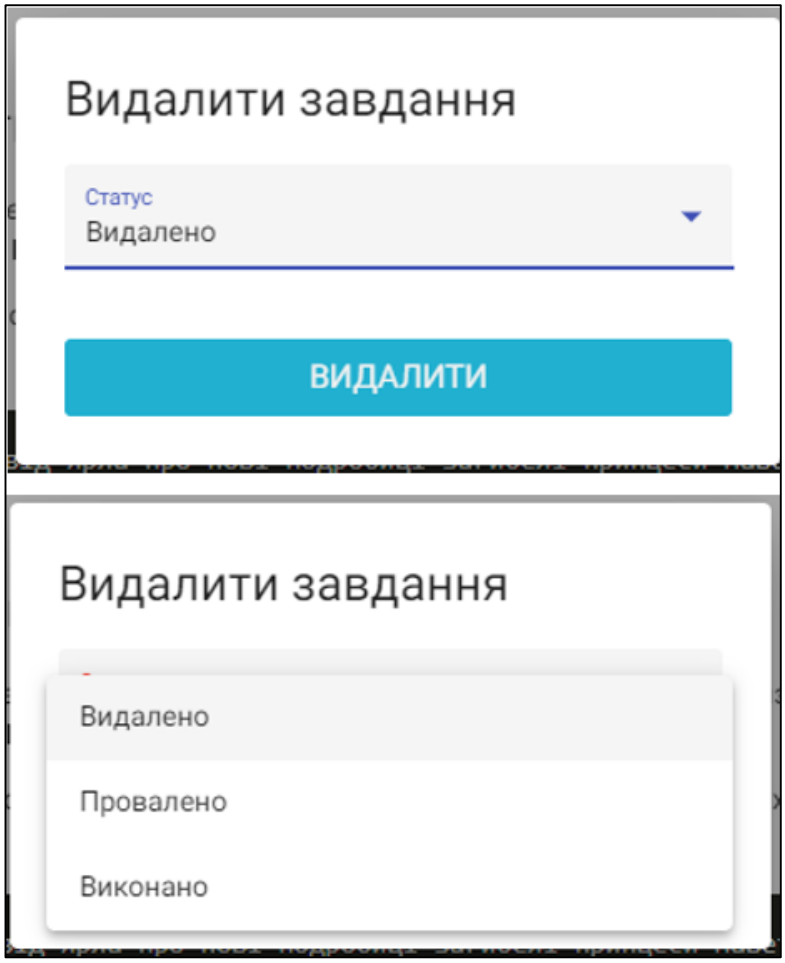


Рисунок 3.33 – Вспливаюче вікно видалення завдання

Для редагування завдання користувач повинен натиснути на кнопку «редагувати», після чого відкриється сторінка редагування завдання, де користувач може змінити усі необхідні дані в завдані. Сторінка редагування завдання є повністю ідентичною до сторінки створення завдання, за винятком кнопки, на сторінці редагування завдання кнопка має назву «редагувати».

Для перегляду статистики дошки на її сторінці потрібно натиснути на кнопку «статистика». Після натиснення користувача буде перенаправлено на сторінку статистики. Сторінка статистики містить в собі 4 вкладки:

– графік статусів;
– розподіл статусів;
– графік створення;
– оцінка/витрачено.

Кожна вкладка, крім вкладки «графік створення», містить фільтри по даті та користувачу. Вкладка «графік створення» містить тільки фільтр по даті.

Графік статусів відображує зміну статусів завершених завдань групуючи їх по місяцям. При наведені на відповідний пункт в легенді відповідний графік буде виділятись. Також при наведені на лінію яка розділяє графік по місяцях буде виведено кількість завдань по статусах в відповідний місяць. Графік статусів зображено на рисунку 3.34.

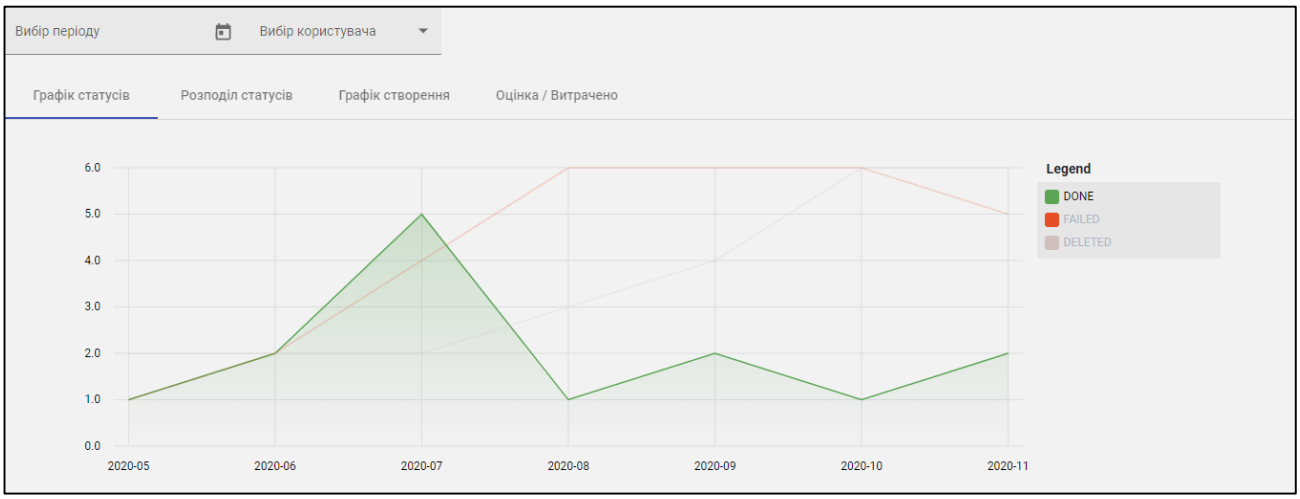


Рисунок 3.34 – Графік статусів

На вкладці «розподіл статусів» відображається співвідношення статусів. При наведені на елемент легенди відповідний сектор буде виділятися. Також при наведені на сегмент буде відображатись кількість завдань відповідного статусу. На рисунку 3.35 зображено вкладку сторінки статистики «розподіл статусів».

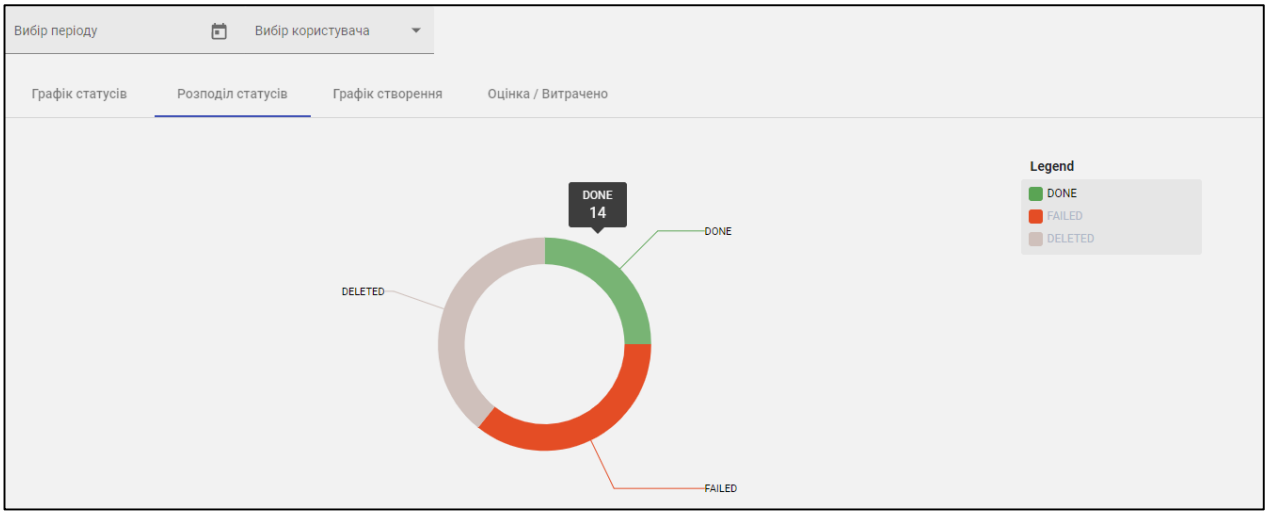


Рисунок 3.35 – Вкладка сторінки статистики «розподіл статусів»

На вкладці «графік створення» відображається діаграма на якій зображено кількість створених завдань за відповідні місяці. При наведені на елемент легенди відповідний стовбець буде виділятися. Також при наведені на стовбець буде відображатись кількість створених завдань відповідного того, чи іншого місяця. На рисунку 3.36 зображено вкладку сторінки статистики «графік створення».

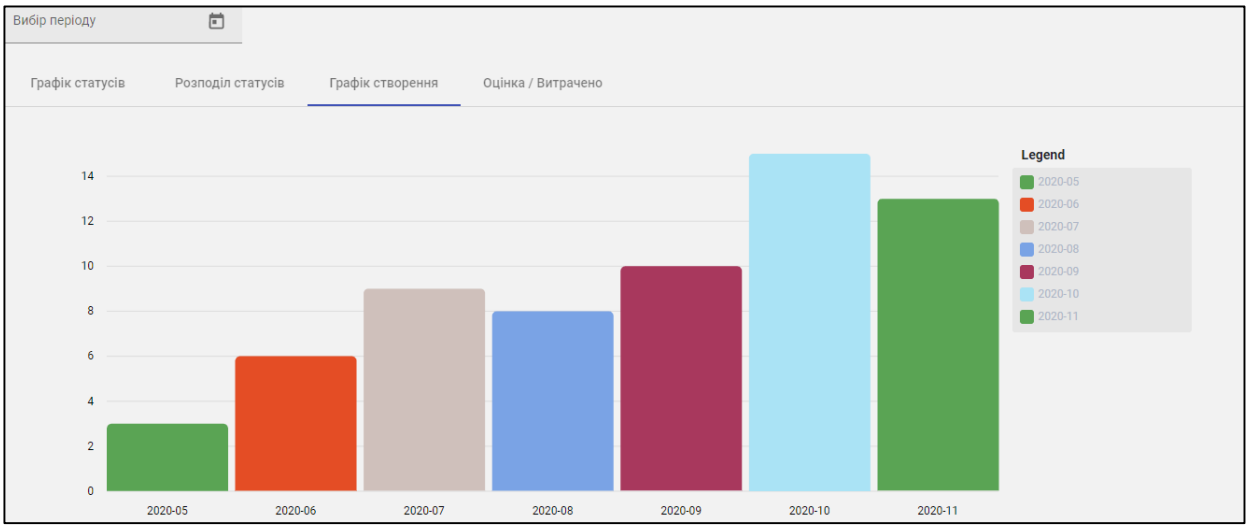


Рисунок 3.36 – Вкладка сторінки статистики «графік створення»

На вкладці «оцінка/витрачено» відображується співвідношення часу в який оцінювали користувачі виконання завдань, до часу, який був фактично потрачений на їх виконання. При наведені на відповідний пункт в легенді відповідний графік

буде виділятись. Також при наведені на лінію яка розділяє графік по місяцях буде виведено витрачений та оцінений час в відповідний місяць. На рисунку 3.37 зображено вкладку сторінки статистики «графік створення».

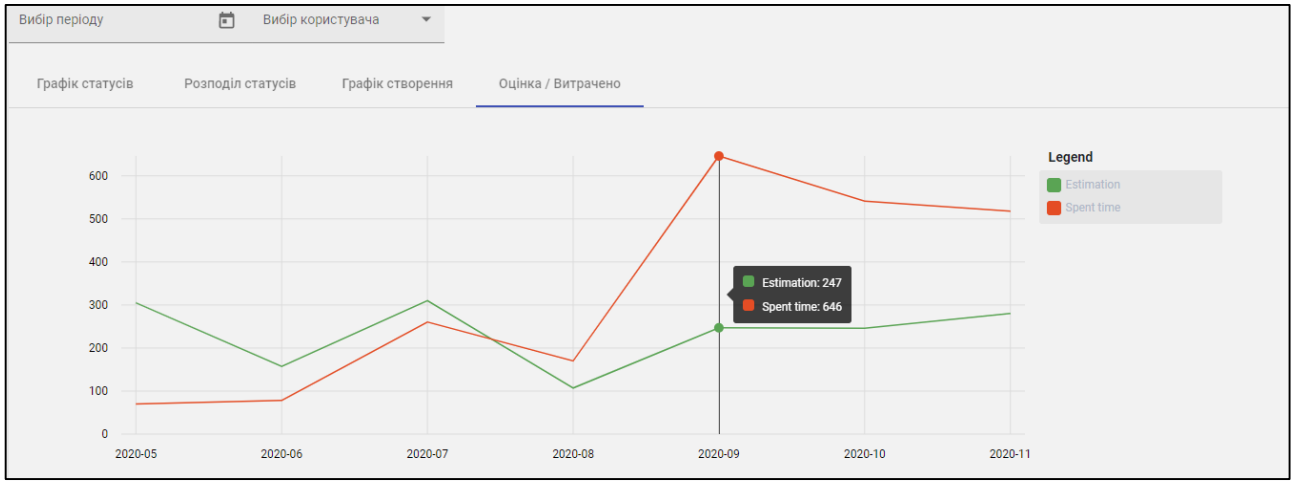


Рисунок 3.37 – Вкладка сторінки статистики «оцінка/витрачено»

3.4 Висновки

В даному розділі створено базу даних інформаційної системи управління проєктами, яка забезпечує надійне зберігання усієї інформації системи.

Розроблено серверну частину інформації системи, в який реалізовано усі необхідні запити до бази даних, та налаштовано маршрути для взаємодії з клієнтською частиною. Також на серверній частині було реалізовано модуль реєстрації та авторизації.

Реалізовано клієнтську частину інформації системи. Клієнтська частина має обмеження для користувачів по їх типам.

Також налаштовано взаємодію усіх частин інформаційної системи управління проєктами.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу

Для того, щоб визначити комерційний потенціал дослідження потрібно залучити двох, або трьох незалежних експертів.

Для оцінки комерційного потенціалу дослідження будуть використовуватись дванадцять критеріїв, які наведені в таблиці 4.1.

Таблиця 4.1 – Критерії оцінювання комерційного потенціалу дослідження та їх можлива бальна оцінка

Критерії оцінювання та бали					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали					
Критерій	0	1	2	3	4
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Малий ринок з позитивною динамікою	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 4.1

Критерії оцінювання та бали					
Критерій	0	1	2	3	4
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу дослідження заносимо в таблицю 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу дослідження

Критерії	Експерти		
	№ 1	№ 2	№ 3
	Виставлені бали		
1	4	3	3
2	3	3	4
3	2	3	2
4	3	3	3
5	1	2	1
6	4	4	4
7	3	3	3
8	4	4	4
9	2	2	2
10	2	2	3
11	4	3	4
12	4	4	3
Сума балів	$СБ_1 = 38$	$СБ_2 = 36$	$СБ_3 = 36$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = 36{,}67 \approx 37$		

Середня сума балів 37 оцінки потенціалу дослідження експертами вказує на те, що рівень комерційного потенціалу дослідження вище середнього.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Кошторис витрат на дослідження може складатися з таких етапів:

1-й етап: кошторис витрат на дослідження передбачає розрахунок таких основних витрат:

1) Основна заробітна плата розробників розраховується за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \quad (4.1)$$

де М – місячний посадовий оклад конкретного розробника у 2020 році не менше розміру мінімальної заробітної плати 5000 грн;

$Т_р$ – число робочих днів в місяці, $Т_р = 20$;

t – число робочих днів роботи розробника.

Підставимо дані до формули (4.1) та отримаємо:

$$З_о = \frac{5500}{20} \cdot 15 = 4125 \text{ (грн)},$$

$$З_о = \frac{8000}{20} \cdot 20 = 8000 \text{ (грн)},$$

$$\sum З_о = 4125 + 8000 = 12125 \text{ (грн)}.$$

Зроблені розрахунки заносимо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату розробників

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Розробник	5500	367	15	4125
Інженер-програміст	8000	200	20	8000
Всього				$\sum З_о = 12125$

3) Додаткова заробітна плата ($З_д$) всіх розробників, які приймали участь в дослідженні.

Розрахуємо додаткову заробітну плату як 10...12% від суми основної заробітної плати всіх розробників та робітників. Приймаємо додаткову заробітну плату 10% від основної:

$$З_д = 0{,}10 \cdot (З_о + З_р) \text{ [грн]}, \quad (4.2)$$

$$З_д = \frac{(12125 + 279{,}4) \cdot 10}{100\%} = 1240{,}44 \text{ (грн)}.$$

4) Нарахування на заробітну плату $Н_{зп}$ розробників та робітників становить 22 % (у 2020 році) від суми основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування у 2020 році, %. $\beta = 22\%$.

$$Н_{зп} = (12125 + 279{,}4 + 1240{,}44) \cdot \frac{22}{100} = 3001{,}87 \text{ (грн)}.$$

5) Амортизація обладнання, комп'ютерів та приміщень А, які використовувались для дослідження. У спрощеному вигляді амортизаційні відрахування розраховуються за формулою (4.4):

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ [грн]}, \quad (4.4)$$

де Ц – загальна вартість обладнання, або приміщень, грн;

$Н_а$ – річна норма амортизаційних відрахувань. $Н_а = 20\%$;

Т – термін використання обладнання, або приміщень, місяці.

Всі проведені розрахунки заносимо в таблицю 4.4.

Таблиця 4.4 – Розрахунок амортизаційних відрахувань амортизаційних відрахувань

Найменування комплектуючих	Балансова вартість, грн.	Норма амортизації, %	Термін використання міс.	Величина амортизаційних відрахувань, грн.
Комп’ютер	17300	20	2	574
Програмне забезпечення	6750	20	2	223,29
Приміщення	110 000	5	3	1371
Всього				$А = 2168,29$

7) Витрати на послуги, що були використані під час виконання наукової розробки, заносимо у таблицю 4.5

Таблиця 4.5 – Розрахунок витрат на послуги

Найменування послуг	Термін використання, місяців	Ціна за місяць, грн	Вартість послуг
Послуги інтернет	1	120	120
Послуги розміщення програмного продукту	2	380	760
Всього			8980

8) Витрати на силову електроенергію $В_е$ розраховуємо за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (4.5)$$

де В – вартість 1 кВт-год. електроенергії, в 2020 році $В = 2{,}5$ грн./кВт;

П – установлена потужність обладнання, кВт. Споживана електрична потужність комп'ютера та інших пристроїв при проведенні досліджень $П = 0{,}2$ кВт;

Ф – фактична кількість годин роботи обладнання, годин. Фактична кількість годин роботи комп'ютера та інших пристроїв при проведенні досліджень $Ф = 20 \cdot 8 = 160$ годин;

$К_п$ – коефіцієнт використання потужності, $К_п < 1$. Обираємо $К_п = 0{,}9$.

Отже, витрати на силову електроенергію становлять:

$$В_е = 2{,}5 \cdot 0{,}2 \cdot 160 \cdot 0{,}9 = 72 \text{ (грн)}.$$

9) Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які виконували дослідження, тобто:

$$В_{ін} = (З_о + З_р) \cdot (100\% \ldots 300\%) \text{ [грн]}, \quad (4.6)$$

$$В_{ін} = \frac{(12125 + 279{,}4) \cdot 200\%}{100\%} = 24808{,}8 \text{ (грн)}.$$

10) Сума всіх попередніх статей витрат дає загальні витрати на виконання даної частини дослідження – В.

$$В = З_о + З_р + З_д + Н_{зп} + А + М + К + В_е + В_{ін} \text{ [грн]}, \quad (4.7)$$

$$В = 12125 + 1240{,}44 + 3001{,}87 + 2168{,}29 +$$

$$+166{,}75 + 28106 + 880 + 72 + 24808{,}8 = 73048{,}55 \text{ (грн)}.$$

2-й етап: розрахунок загальних витрат на дослідження. Загальна вартість дослідження $В_{заг}$ розраховується за формулою:

$$В_{заг} = \frac{В}{\alpha} \text{ [грн]}, \quad (4.8)$$

де α – частка витрат, які здійснює виконавець дослідження, у відносних одиницях. Приймемо $\alpha = 0{,}7$.

$$В_{заг} = \frac{73048{,}55}{0{,}9} = 81165{,}05 \text{ (грн)}.$$

3-й етап: прогнозування загальних витрат на виконання та впровадження наукової розробки. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів дослідження здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \text{ [грн]}, \quad (4.9)$$

де β – коефіцієнт, який характеризує етап виконання дослідження. Приймаємо $\beta = 0{,}9$.

$$ЗВ = \frac{81165{,}55}{0{,}9} = 90183{,}38 \text{(грн)}.$$

4.3 Прогнозування комерційних ефектів від реалізації результатів дослідження

Спрогнозуємо яку вигоду можна отримати в майбутньому, якщо впроваджувати результати даної наукової роботи.

Спочатку потрібно:

а) вказати, скільки часу триватиме виконання даної наукової роботи та впровадження її результатів.

Приймаємо, що виконання наукової роботи та впровадження її результатів займе 1 рік.

б) зазначити, коли очікуються основні позитивні результати від впровадження дослідження.

Приймаємо, що основні позитивні результати від впровадження дослідження очікуються протягом 3-х років.

в) назвати ці позитивні результати та кількісно їх оцінити по роках.

В ринкових умовах позитивним результатом є збільшення чистого прибутку підприємства

Існує два основних сценарії прогнозування зростання чистого прибутку компанії. Використаємо другий сценарій, коли неможливо безпосередньо оцінити зростання чистого прибутку компанії від впровадження результатів наукового дослідження. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від впровадження дослідження, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \text{ [грн]}, \qquad (4.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_0$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість. У 2018 році ставка податку на додану вартість встановить на рівні 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = 0{,}2 \dots 0{,}3$. Приймемо $\rho = 0{,}25$;

ϑ – ставка податку на прибуток. У 2020 році $\vartheta = 18\%$.

Припустимо що в результаті впровадження, на території України, результатів наукового дослідження покращується якість розробки, що дозволяє підвищити ціну їх реалізації на 600 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 5000 шт., протягом другого року – ще на 3000 шт., протягом третього року – ще на 1000 шт.

Орієнтовно: реалізація продукції до впровадження результатів розробки складає 12000 шт., а її ціна – 2800 грн.

Тоді збільшення чистого прибутку підприємства $\Delta П_i$ протягом першого року складе:

$$\Delta П_1 = [600 \cdot 12000 + (2800 + 600) \cdot 5000] \cdot 0{,}8333 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = \\ = 4134001 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукового дослідження) складе:

$$\Delta П_2 = [600 \cdot 12000 + (2800 + 600) \cdot (5000 + 3000)] \cdot 0{,}8333 \cdot 0{,}25 \cdot \\ \cdot \left(1 - \frac{18}{100}\right) = 5876431 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом третього року (відносно базового року, тобто року до впровадження результатів наукового дослідження) складе:

$$\Delta П_3 = [600 \cdot 12000 + (2800 + 600) \cdot (5000 + 3000 + 1000)] \cdot$$
$$\cdot 0{,}8333 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 6457241 \text{ (грн)}.$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для розрахунку ефективності вкладених інвестицій потрібно виконати наступні дії.

Спершу потрібно розраховати вартість інвестицій PV, що вкладають в дослідження. Прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів дослідження можна вважати такою вартістю. Вона була розрахована раніше, тобто можемо вважати, що $ЗВ = PV = 90183{,}38 (грн)$.

Далі розраховати очікуване збільшення прибутку $\Delta П_i$, для кожного із років, починаючи з першого. Таке збільшення прибутку також було розраховане раніше. За даною формулою: $\Delta П_1 = 4134001$ (грн), $\Delta П_2 = 5876431$ (грн), $\Delta П_3 = 6457241$ (грн).

Для спрощення подальших розрахунків потрібно побудувати вісь часу, на яку наносять всі платежі, що мають місце під час виконання дослідження та впровадження його результатів.

Схематичне відображення руху інвестицій та додаткових прибутків зображено на рисунку 4.1.

Рисунок 4.1 – Схематичне відображення руху інвестицій та додаткових прибутків

Наступним кроком буде розрахунок абсолютної ефективності вкладених інвестицій $E_{абс}$ за формулою:

$$E_{абс} = (ПП - PV)\ [грн], \tag{4.11}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукового дослідження, грн.;

PV – теперішня вартість інвестицій $PV = ЗВ$, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}\ [грн], \tag{4.12}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженого дослідження, грн.;

Т – період часу, протягом якого виявляються результати впровадження дослідження, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівня 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки «0».

$$ПП = \frac{4134001}{(1+0{,}1)^2} + \frac{5876431}{(1+0{,}1)^3} + \frac{6457241}{(1+0{,}1)^4} = 12241961 \text{ (грн)}.$$

Тепер обрахуємо абсолютну ефективність вкладених інвестицій:

$$Е_{абс} = (12241961 - 90183{,}38) = 12151777{,}6 \text{ (грн)}.$$

Оскільки $Е_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток.

Далі потрібно розрахувати відносну (щорічну) ефективність вкладених в наукове дослідження інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \qquad (4.13)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій $PV = ЗВ$, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[4]{1 + \frac{12151777{,}6}{90183{,}38}} - 1 = 2{,}41 \text{ або } 241\%.$$

Отже, відносна (щорічна) ефективність вкладених інвестицій становить 319,9%.

Далі ефективність вкладених інвестицій потрібно порівняти з мінімальною ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку

інвестиції вкладатися не будуть. У загальному вигляді мінімальна ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; $d = (0{,}14 \dots 0{,}2)$;

f – показник, що характеризує ризикованість вкладень, зазвичай величина $f = (0{,}05 \dots 0{,}1)$.

$$\tau_{мін} = 0{,}2 + 0{,}1 = 0{,}3 \text{ або } 30\%.$$

Як видно з розрахунків $E_в = 319{,}9\% > \tau_{мін} = 30\%$, тому потенційний інвестор буде зацікавлений у фінансуванні даного наукового дослідження, а не у вкладенні коштів на депозит у комерційному банку.

Наступним кроком буде розрахунок терміну окупності вкладених у реалізацію наукового проєкту інвестицій $T_{ок}$ за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [років]}, \quad (4.15)$$

$$T_{ок} = \frac{1}{2{,}41} = 0{,}41 \text{ (року)}$$

Оскільки термін окупності вкладених у реалізацію наукового проєкту інвестицій знаходиться в допустимих межах $T_{ок} = 3{,}6 \text{ місяця} < 3 \dots 5 -$ ти років, то фінансування даної наукової розробки є доцільним.

4.5 Висновки

В даному розділі зроблено попередні розрахунки для економічної частини дослідження «Інформаційна система управління проєктами». Три незалежні експерти оцінили комерційний потенціал проекту розвитку та визначили, що рівень комерційного потенціалу проекту розвитку був вище середнього. У процесі розрахунку проводилась оцінка витрат, вартість заробітної плати підрядника, додаткова зарплата всіх підрядників, вартість праці, вартість амортизації, вартість матеріалів та комплектуючих та вартість електроенергії та інші види витрат. Пораховано що загальні витрати на виконання даної частини дослідження становлять $73048{,}55$ гривень. Також пораховано, що загальна вартість становить $81165{,}05$ гривень. Здійснено прогнозування загальних витрат на виконання та впровадження результатів та встановлено, що $ЗВ = 90183{,}38$ гривень.

Проведено прогнозування збільшення чистого прибутку підприємства від впровадження результатів наукового дослідження у кожному з трьох років відносно базового.

Проаналізовано ефективність інвестицій та термін їх окупності. Розрахунок абсолютної ефективності вкладених інвестицій показав, що результат від проведення наукових досліджень та їх впровадження принесе прибуток. Відносна (річна) ефективність інвестицій в дане наукове дослідження свідчить про те, що потенційні інвестори будуть зацікавлені у фінансуванні даного дослідження, оскільки термін окупності вкладених у реалізацію наукового проєкту інвестицій дорівнює $Т_{ок} = 0{,}41$ року.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було розроблено механізми для роботи власника, керівника та робітника в інформаційній системі управління проєктами.

Було розроблено та реалізовано:

– базу даних для оптимального представлення усієї інформації предметної області;

– прикладний програмний інтерфейс, що має можливість швидко читати та записувати інформацію до бази даних, а також легко обмінюватися цими даними з клієнтською частиною;

– клієнтську частину, що має зручний та зрозумілий інтерфейс, який може забезпечувати в режимі реального часу та швидку взаємодію між користувачами, системою та її базою даних.

Розроблена система містить в собі основний функціонал аналогів, який вдосконалено спрощеним інтерфейсом та розширеним модулем аналітики. Також в системі присутні такі особливості як: українська локалізація, можливість відслідковувати час виконання завдань.

На всіх етапах систему було проаналізовано з метою зменшення можливості виникнення помилок та незапланованих сценаріїв роботи.

У ході виконання економічної частини кваліфікаційної роботи на основі розрахунків було доведено, що продукт є економічно доцільним, оскільки витрати на розробку вказаної системи становлять 90183,38 грн. Розрахунок відносної ефективності вкладених в наукове дослідження інвестицій показав, що потенційний інвестор буде зацікавлений у фінансуванні даного наукового дослідження, а термін окупності інвестицій становить 0,41 року.

За результатами магістерської кваліфікаційної роботи опубліковано тези доповідей на науковій конференції. Отже, поставлені задачі магістерської кваліфікаційної роботи були виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лишак О.М. Інформаційна управління проєктами / О. М. Лишак, О. Б. Мокін, // Матеріали XV міжнародної конференції "Контроль і управління в складних системах (КУСС-2020)", Вінниця.: 8-10 жовтня 2020. – Режим доступу: http://ir.lib.vntu.edu.ua//handle/123456789/30594

2. Бурков В. Н. Введение в теорию управления организационными системами. Учебник / В. Н. Бурков, Н. А. Коргин, Д. А. Новиков - Москва: ЛИБРОКОМ, 2009. – 264 с.

3. Guidelines for Managing Projects from the UK Department for Business, Enterprise and Regulatory Reform (BERR) [Електронний ресурс]. - Режим доступу до ресурсу: http://www.berr.gov.uk/files/file40647.pdf

4. Батенко Л. П. Управління проектами: навчальний посібник / Батенко Л. П. – Київ: КНЕУ, 2003. – 231 с.

5. Open Source Project Management manual [Електронний ресурс]. - Режим доступу до ресурсу: https://www.projectmanagement-training.net/articles-tools/book/

6. Боголюбський О. В. Методи управління: сутність, різновиди, взаємозв'язок функцій / О. В. Боголюбський, М. В. Федоренко, М. О. Кошолап – Житомир: ЖІ МАУП, 1996. – 374 с.

7. Мазур В. О. Оптимізація робочого часу. Особистий та командний тайм-менеджмент / В. О. Мазур – Харків: ІНЖЕК, 2005. – 195 с.

8. Управление проєктами (логистика, транспорт и др.) [Електронний ресурс]. – Режим доступу до ресурсу: http://project.logistics-gr.com/

9. Max Wideman's «Open Source» Comparative Glossary of Project Management Terms [Електронний ресурс]. – Режим доступу до ресурсу: http://www.maxwideman.com/

10. Mesly Olivier. Project feasibility – Tools for uncovering points of vulnerability/ Mesly Olivier - New York: Taylor and Francis, CRC Press. 2017. - 546 с.

11. Manage your team’s work, projects, & tasks online | Asana [Електронний ресурс]. – Режим доступу до ресурсу: https://asana.com/home

12. Jira | Программное обеспечение для отслеживания задач и проектов [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.atlassian.com/software/jira

13. Trello [Електронний ресурс]. – Режим доступу до ресурсу: https://trello.com/uk

14. Overview - Redmine [Електронний ресурс]. – Режим доступу до ресурсу: https://www.redmine.org

15. Встановлення та налаштування Node.js [Електронний ресурс]. – Режим доступу до ресурсу: https://internetdevels.ua/blog/node-js-installation-setting

16. Node.js: JavaScript ты ли это? [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/post/182628

17. Express/Node introduction [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.mozilla.org/enUS/docs/Learn/Serverside/Express_Nodejs

18. Express.js [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Express.js

19. Angular [Електронний ресурс]. – Режим доступу до ресурсу: https://angular.io/

20. Shyam S. / Angular: Up and Running: Enhanced Productivity with Structured Web Apps / S. Shyam, G. Brad - Amazon Digital Services LLC : 2016 – 322 с.

21. Angular Material Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angular_material/

22. Керівництво по TypeScript [Електронний ресурс]. – Режим доступу до ресурсу: https://metanit.com/web/typescript

23. MySQL [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MySQL

24. Мартін Груббер. Розуміння SQL. / Мартін Груббер. – Санкт-Петербург: ПИТЕР, 1993. – 289с.

25. Карпова В. П. Організація баз даних: підручник / В. П. Карпова - Київ, 2001 - 260 с.

26. Object–relational mapping [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping

27. ORM-система Sequelize для Node.js [Електронний ресурс]. – Режим доступу до ресурсу: https://hackerx.ru/orm-sequelize-node-js/.https://www.zina.design/uk/dictionary/bootstrap/http://webstudio2u.net/ua/programming/120-jquery.html

28. Навіщо потрібен Refresh Token, якщо є Access Token? [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/company/Voximplant/blog/323160/

29. Privacy-Enhanced Mail [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Privacy-Enhanced_Mail

30. Методичні вказівки до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 –«Інформаційні системи та технології» денної форми навчання / Уклад. В. Б. Мокін, С. О. Жуков, А. Р. Ящолт, О. М. Козачко, Л. М. Скорина. – Електронне видання. – Вінниця : ВНТУ, 2018. – 48 с.

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. В.Б. Мокін
(підпис)
"__" _____ 2020

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ»

08–53.МКР.002.02.00.ТЗ

Керівник магістерської кваліфікаційної роботи
д.т.н., проф.
_____ О. Б. Мокін
(підпис)
"__" _____ 2020 р.

Розробив студент гр. 2ІСТ-19м
_____ О. М. Лишак
(підпис)
"__" _____ 2020 р.

Вінниця 2020

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № _ по ВНТУ від «_» _____ 2020 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «_» _____ 2020 р.

2. Джерела розробки:

- Express/Node introduction [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.mozilla.org/enUS/docs/Learn/Serverside/Express_Nodejs

- Angular [Електронний ресурс]. – Режим доступу до ресурсу: https://angular.io

3. Мета і призначення роботи

Розробка інформаційної системи управління проєктами, яка відрізняється від існуючих спрощеним інтерфейсом та розширеним модулем аналітики.

4. Вихідні дані для проведення робіт:

- результати експертного оцінювання існуючих систем управління проєктами;
- специфікації agile підходів управління проєктами.

5. Методи дослідження:

- порівняльний аналіз даних;
- моделювання системи.

6. Етапи роботи і терміни їх виконання

a) Аналіз предметної області ..... __. – __
b) Проєктування інформаційної системи..... __ – __
c) Розробка інформаційної системи ..... __ – __
d) Розробка інструкції користувача..... __ – __

7. Очікувані результати та порядок реалізації

Отримання програмного забезпечення інформаційної системи управління проєктами.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист……__.__.2020 р.
Початок розробки «___» _____ 2020 р.
Граничні терміни виконання МКР «___» _____2020 р.

Розробив студент групи 2ІСТ-19м _____ Лишак О.М.

Додаток Б

Інструкція користувача

Для початку роботи з програмою користувачу потрібно авторизуватися в системі. Якщо користувач не має облікового запису, йому потрібно пройти процес реєстрації. Форми авторизації, та реєстрації зображені на рисунку Б.1.

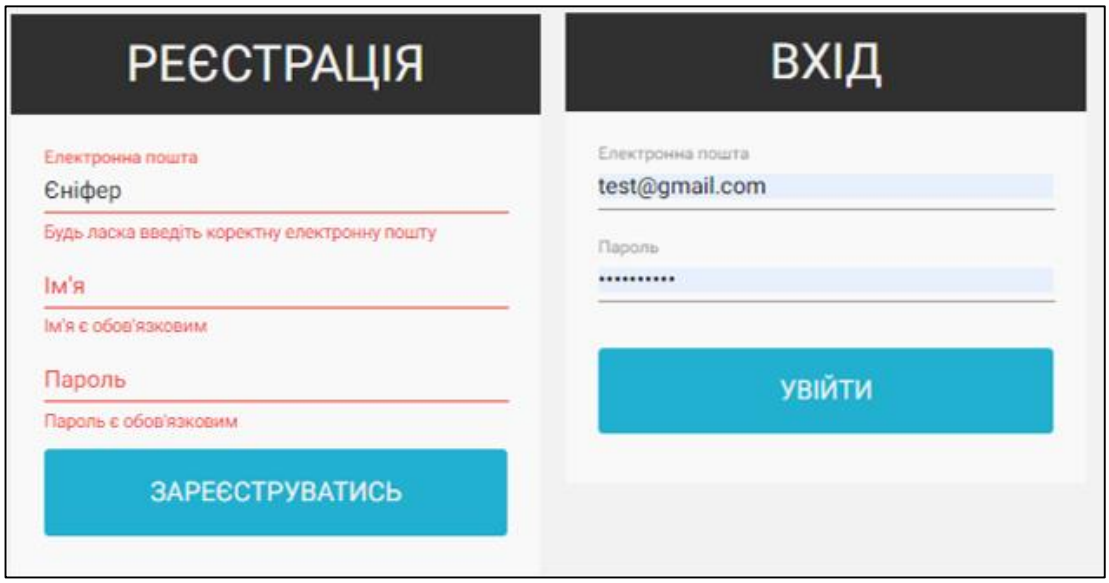


Рисунок Б.1 – Форми авторизації та реєстрації

Після авторизації програма відображає головну сторінку, де користувач, може переглянути список доступних йому дошок, та перейти на сторінку створення нової дошки. Головна сторінка програми зображено на рисунку Б.2.

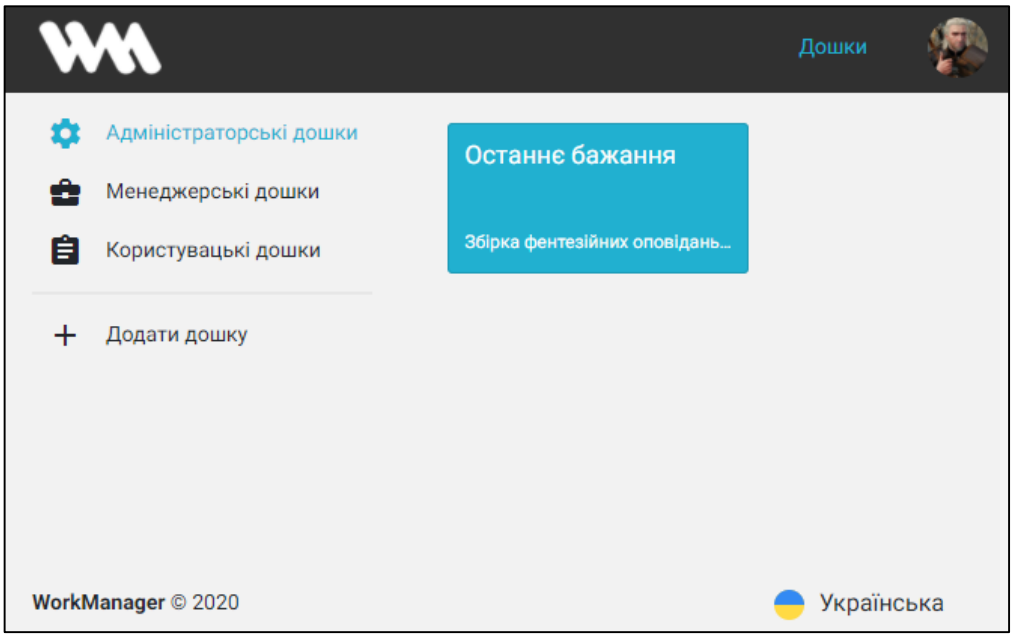


Рисунок Б.2 – Головна сторінка програми

Для створення нової дошки потрібно натиснути кнопку «додати дошку», програма відкриє сторінку з формою створення дошки. Для збереження дошки, після заповнення необхідних полів потрібно натиснути кнопку «створити». Програма створить нову дошку, та автоматично відкриє її користувачу. Форма створення дошки зображена на рисунку Б.3.

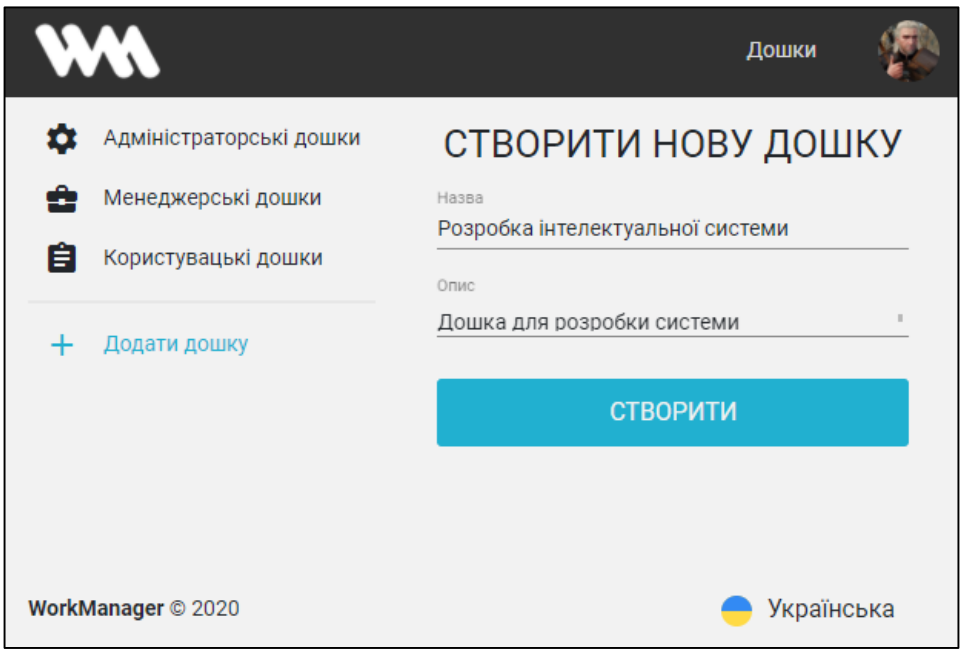


Рисунок Б.3 – Форма створення дошки

На сторінці дошки користувач має можливість створити нове завдання, переглянути, редагувати та видалити існуюче. Якщо користувач є власником, або керівником, то у нього доступна функція редагування дошки. Сторінка дошки зображена на рисунку Б.4.

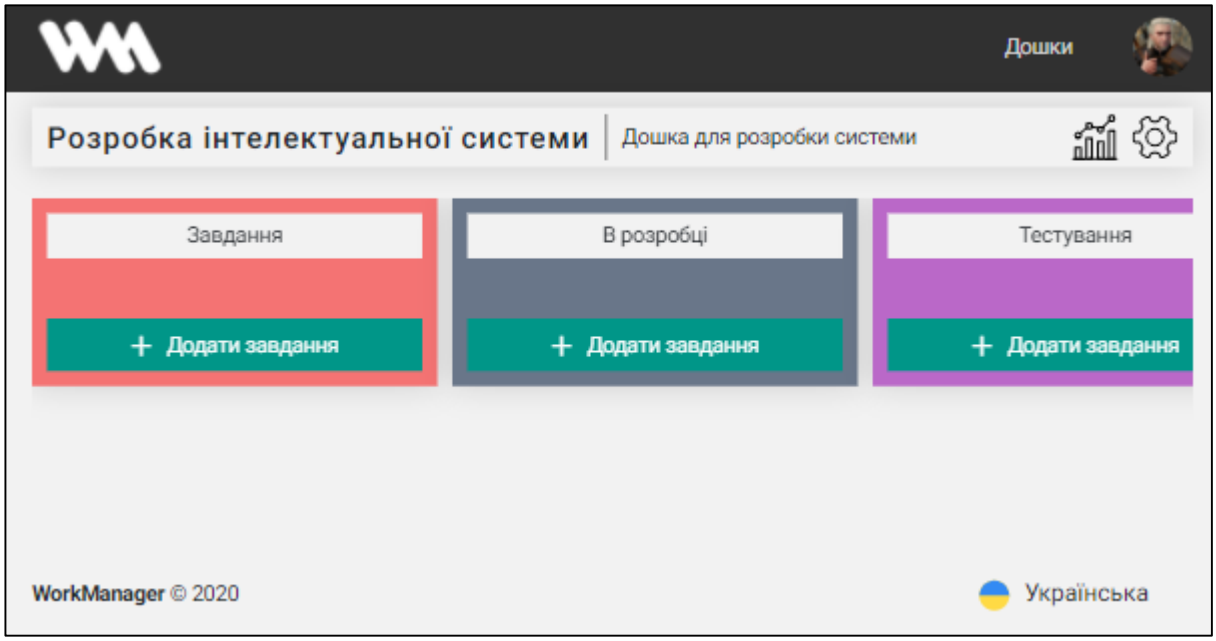


Рисунок Б.4 – Сторінка дошки

На сторінці редагування дошки користувач має можливість змінити назву та опис дошки, додати, видалити користувачів дошки, або редагувати їх права доступу, редагувати, видаляти та створювати нові колонки. Сторінки редагування дошки зображені на рисунках Б.5-Б.7 відповідно.

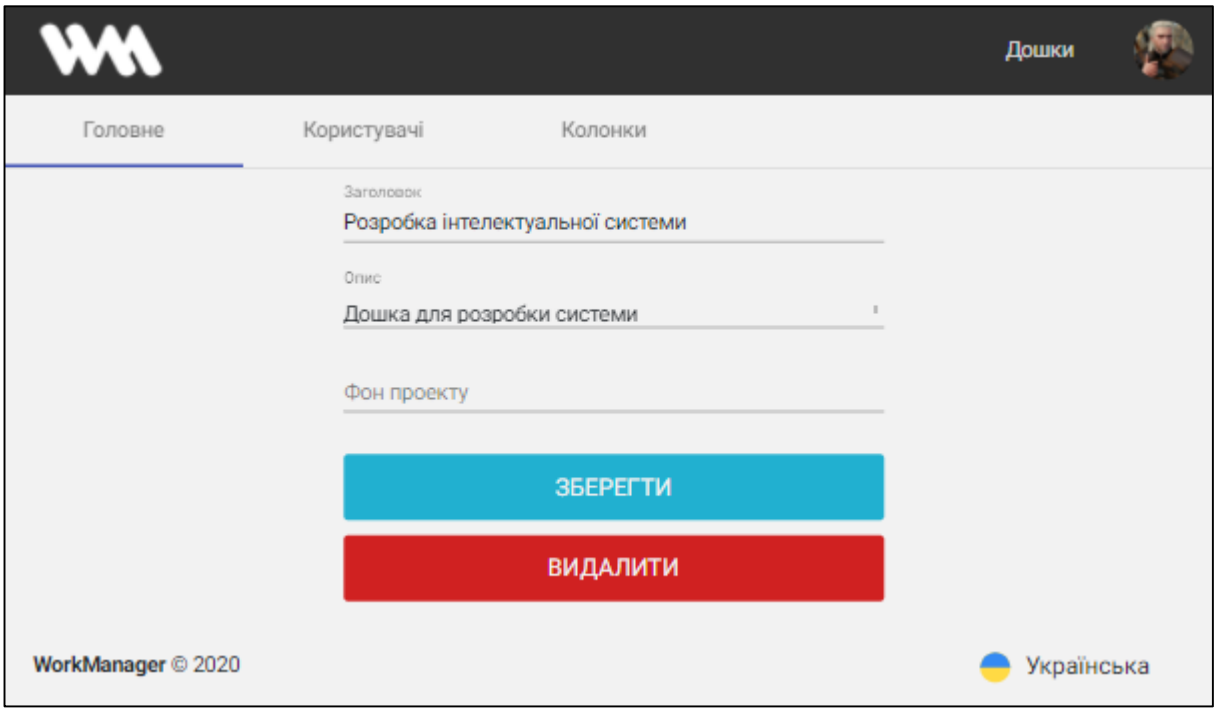


Рисунок Б.5 – Сторінка редагування головної інформації дошки

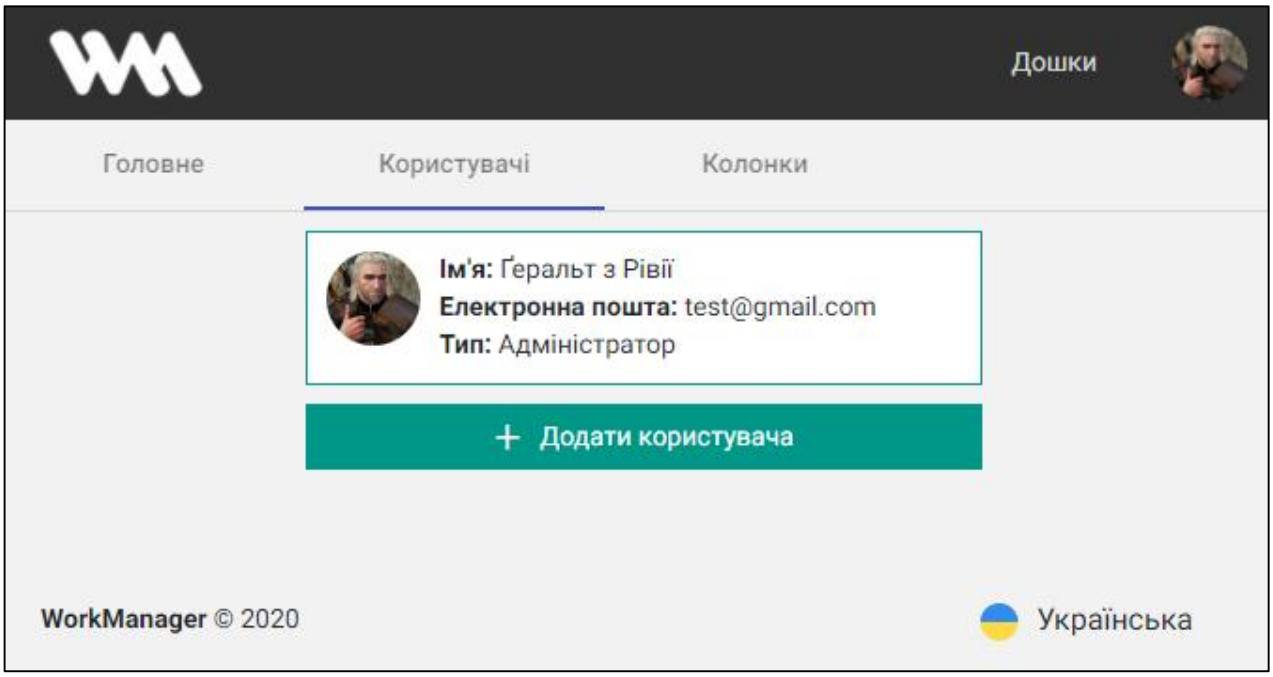


Рисунок Б.6 – Сторінка редагування користувачів дошки

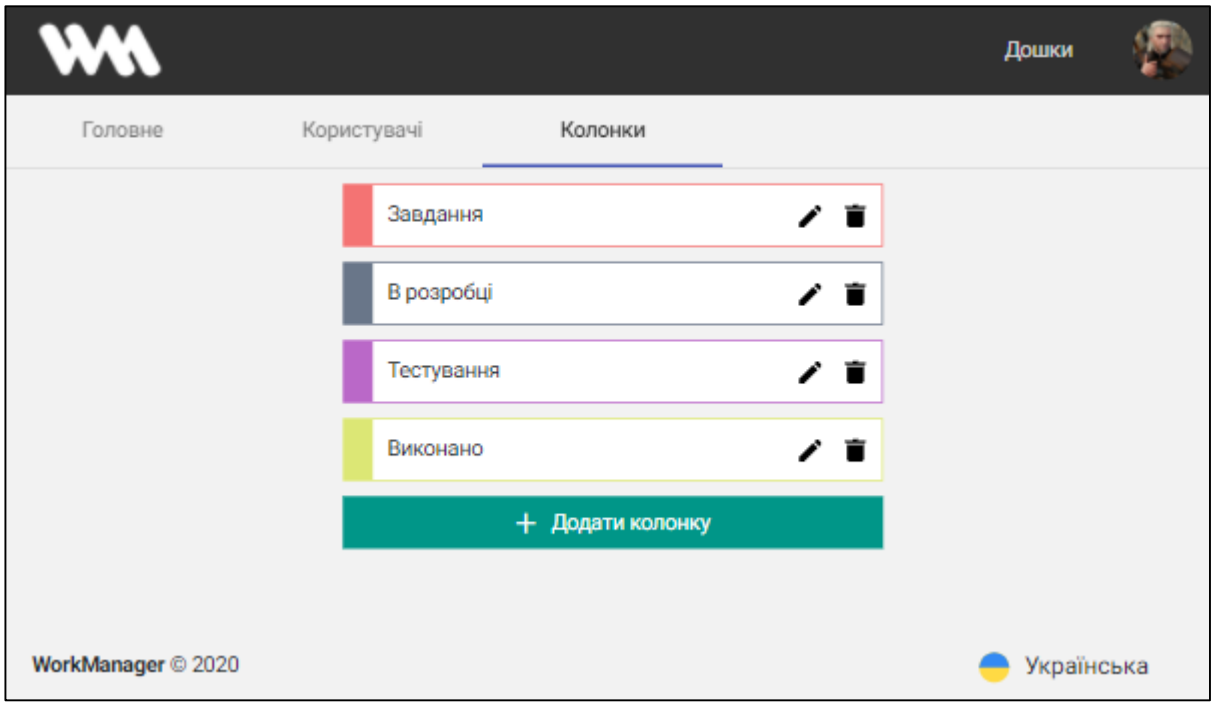


Рисунок Б.7 – Сторінка редагування колонок дошки

Щоб створити нове завдання на сторінці дошки потрібно натиснути на кнопку «додати завдання», програма відкриє сторінку створення завдання. Для збереження завдання, після заповнення необхідних полів потрібно натиснути кнопку «створити». Сторінку створення завдання зображено на рисунку Б.8.

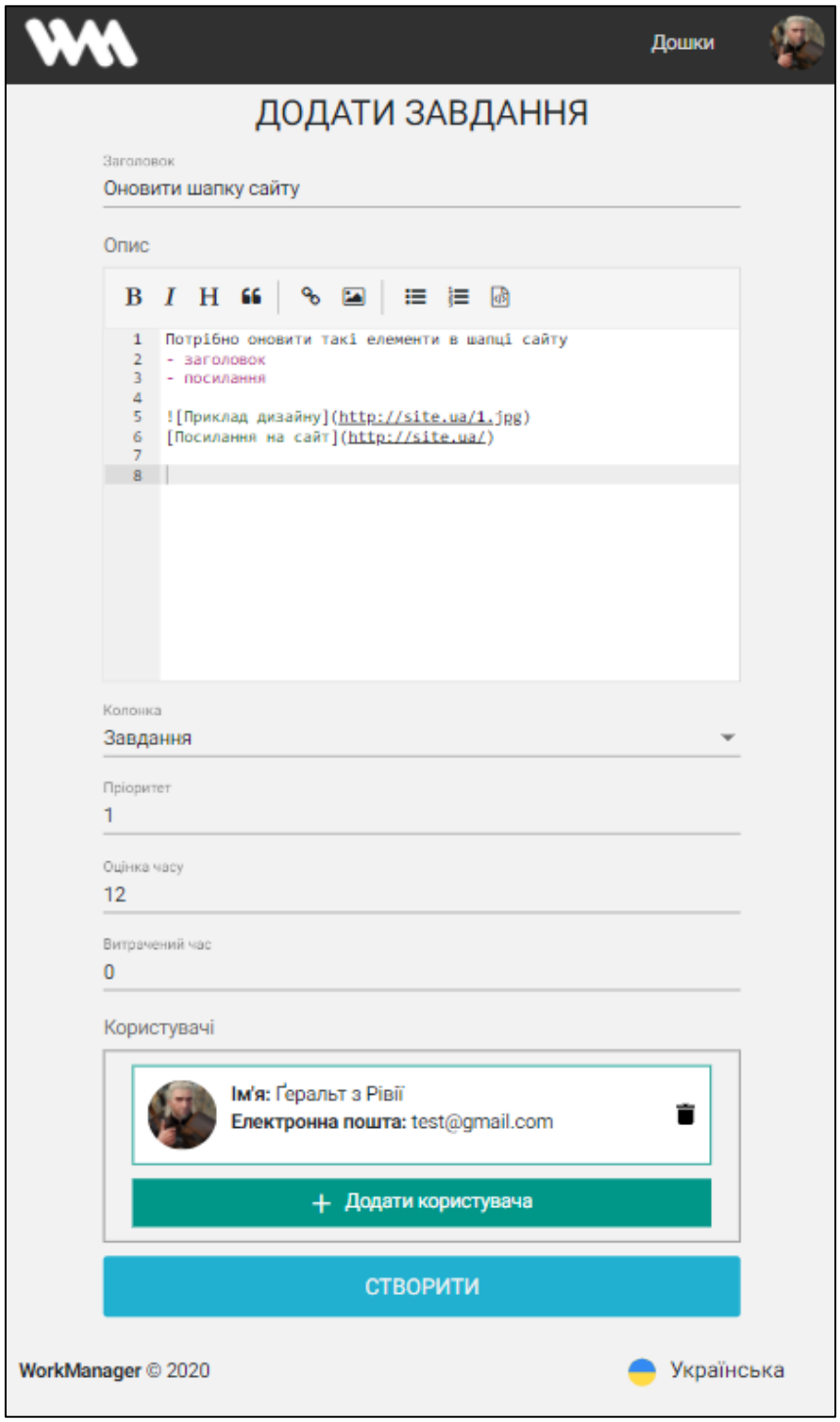


Рисунок Б.8 – Сторінка створення завдання

Додаток В

Лістинг програми

1. Реалізація сервісу AuthenticationService

```
import { Injectable } from '@angular/core';
import { LocalStorageService } from './local-storage.service';

@Injectable()
export class AuthenticationService {

  constructor(
    private localStorageService: LocalStorageService,
  ) { }

  public checkLogin(): boolean {

    const localToken = this.localStorageService.get('token');

    return !!localToken;
  }

  public getToken(): string {
    return this.localStorageService.get('token');
  }

  public deleteToken() {
    this.localStorageService.remove('token');
  }

  public setToken(token: string) {
    this.localStorageService.set('token', token);
  }

  public getRefreshToken(): string {
    return this.localStorageService.get('refresh-token');
  }

  public deleteRefreshToken() {
    this.localStorageService.remove('refresh-token');
  }

  public setRefreshToken(token: string) {
    this.localStorageService.set('refresh-token', token);
  }
}
```

2. Реалізація моделі RequestModel

```
import { IRequestData } from '../interfaces/request.interface';

export class RequestModel {

    private method: string;
    private url: string;
    private params: any;
    private body: any;

    constructor(method: string, url: string) {
        this.method = method;
        this.url = url;
    }


    public setParams(params) {
        this.params = params;
    }

    public setBody(body) {
        this.body = body;
    }

    public getMethod(): string {
        return this.method
    }

    public getUrl(): string {
        return this.url
    }

    public getRequestData(): IRequestData {

        const requestData: IRequestData = {}

        if (this.params) {
            requestData.params = this.params;
        }
        if (this.body) {
            requestData.body = this.body;
        }

        return requestData;
    }

}
```

3. Реалізація класу LoginGuard

```
import { Injectable } from '@angular/core';
import { CanActivate, Router, UrlTree, RouterStateSnapshot, ActivatedRouteSnapshot }
from '@angular/router';
import { AuthenticationService } from '../services/authentication.service';

@Injectable()
export class LoginGuard implements CanActivate {
    constructor(
        private router: Router,
        private authService: AuthenticationService,
    ) { }

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean |
 UrlTree {

        const isLogined = this.authService.checkLogin();
        const isLoginPatch = state.url.split('/')[1] === 'login';
        const isRegisterPatch = state.url.split('/')[1] === 'register';

        if (isLogined && (isLoginPatch || isRegisterPatch)) {
            return this.router.createUrlTree(['']);
        }
        else if (isLogined && !isLoginPatch && !isRegisterPatch) {
            return true;
        }
        else if (!isLogined && !isLoginPatch && !isRegisterPatch) {
            return this.router.createUrlTree(['login']);
        }
        else if (!isLogined && (isLoginPatch || isRegisterPatch)) {
            return true;
        }
        return false;
    }
}
```

4. Реалізація класу TokenInterceptor

```
import { Injectable } from "@angular/core";
import { HttpInterceptor } from "@angular/common/http";
import { BehaviorSubject } from 'rxjs';
import { catchError, filter, switchMap, take } from 'rxjs/operators';
import { ApiService } from '../services/api.service';
import { AuthenticationService } from '../services/authentication.service';
import { Router } from '@angular/router';
```

```
@Injectable()
export class TokenInterceptor implements HttpInterceptor {

    private refreshTokenInProgress = false;
    private refreshTokenSubject: BehaviorSubject<any> = new BehaviorSubject<any>(null
);


    constructor(
        private apiService: ApiService,
        private authenticationService: AuthenticationService,
        private router: Router,
    ) { }

    intercept(request, next) {
        return next.handle(this.addAuthenticationToken(request))
            .pipe(
                catchError(
                    error => {

                        if (error.url.includes('auth/refresh')) {
                            this.refreshTokenInProgress = false;
                            this.authenticationService.deleteToken();
                            this.router.navigateByUrl('/login')
                            console.log('here');
                            return error;

                        } else if (this.refreshTokenInProgress) {
                            return this.refreshTokenSubject
                                .pipe(
                                    filter(result => result !== null),
                                    take(1),
                                    switchMap(() => next.handle(this.addAuthenticatio
nToken(request)))
                                )
                        } else if (error.status === 401) {
                            this.refreshTokenInProgress = true;
                            this.refreshTokenSubject.next(null);

                            return this.apiService.refreshToken()
                                .pipe(
                                    switchMap((token: any) => {
                                        this.refreshTokenInProgress = false;
                                        this.authenticationService.setToken(token.tok
en)
                                        this.authenticationService.setRefreshToken(to
ken.refresh)
                                        this.refreshTokenSubject.next(token);
                                        return next.handle(this.addAuthenticationToke
n(request));
                                    })
```

```
                            )
                        } else {
                            return error
                        }
                    })
            )
    }

    addAuthenticationToken(request) {
        const accessToken = this.authenticationService.getToken();
        if (!accessToken) {
            return request;
        }
        return request.clone({
            setHeaders: {
                'Content-Type': 'application/json',
                'x-access-token': accessToken
            }
        });
    }
}
```

5. Реалізація сервісу ApiService

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { RequestModel } from '../models/request.model';
import { Methods } from '../enums/methods.enum';
import { AuthenticationService } from './authentication.service';
import { tap } from 'rxjs/operators';

@Injectable()
export class ApiService {

    constructor(
        private http: HttpClient,
        private authService: AuthenticationService,
    ) { }

    private call(request: RequestModel): Observable<any> {

        return this.http.request(
            request.getMethod(),
            request.getUrl(),
            request.getRequestData()
        )
    }
```

```
    public refreshToken(): Observable<any> {
        const body = {
            token: this.authService.getRefreshToken()
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/auth/refres
h');
        newRequest.setBody(body);

        this.authService.deleteRefreshToken();

        return this.http.request(
            newRequest.getMethod(),
            newRequest.getUrl(),
            newRequest.getRequestData()
        )

    }

    public login(email: string, pass: string): Observable<any> {

        const body = {
            email,
            pass
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/auth/signin
');

        newRequest.setBody(body);

        return this.call(newRequest).pipe(
            tap((resp) => {
                if (resp.token) {
                    this.authService.setToken(resp.token);
                    this.authService.setRefreshToken(resp.refresh);
                }
            })
        );
    }

    public register(data): Observable<any> {
        const body = {
            ...data
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/auth/signup
');

        newRequest.setBody(body);

        return this.call(newRequest).pipe(
            tap((resp) => {
                if (resp.token) {
```

```
                    this.authService.setToken(resp.token);
                    this.authService.setRefreshToken(resp.refresh);
                }
            })
        );
    }

    public getMe() {

        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/user/me
');

        return this.call(newRequest)
    }

    public getUser(id) {
        const body = { id }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/user/ge
t');
        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public editUser(data) {
        const body = {
            logo: data.logo,
            name: data.name,
            surname: data.surname,
            email: data.email
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/user/ed
it');
        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public findUsers(value) {
        const body = {
            value
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/user/fi
nd');
        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public changePass(oldPass, newPass) {
        const body = { oldPass, newPass }
```

```
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/user/pa
ss');
        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public getBoards(userType): Observable<any> {
        const body = {
            userType
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/g
et');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public getBoard(boardId): Observable<any> {
        const body = {
            boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/g
etone');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public getBoardData(boardId): Observable<any> {
        const body = {
            boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/s
tats');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public editBoard(data, boardId) {
        const body = {
            title: data.title,
            description: data.description,
            bg: data.bg ? data.bg : '',
            boardId
        }
```

```
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/e
dit');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public deleteBoard(boardId) {
        const body = {
            boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/d
elete');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public createBoard(title, description) {
        const body = {
            title,
            description
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/a
dd');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public getUsers(boardId) {
        const body = {
            boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/u
sers');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public addUserToBoard(boardId, newUser, userType) {
        const body = {
            boardId,
            newUser,
            userType
        }
```

```
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/u
sers/add');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public editUserType(boardId, user, userType) {
        const body = {
            boardId, user, userType
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/u
sers/edit');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public deleteUserFromBoard(boardId, user) {
        const body = {
            boardId, user
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/board/u
sers/delete');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public getColumns(boardId): Observable<any> {
        const body = {
            boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/column/
get');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public changeColumnPosition(column, position) {

        const body = { position, columnId: column };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/column/
position');

        newRequest.setBody(body);
```

```
        return this.call(newRequest)

    }

    public createColumn(data, position, boardId) {
        const column = {
            color: data.color,
            title: data.title,
            position, boardId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/column/
add');
        newRequest.setBody(column);

        return this.call(newRequest)
    }

    public editColumn(data, columnId) {
        const column = {
            color: data.color,
            title: data.title,
            columnId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/column/
edit');
        newRequest.setBody(column);

        return this.call(newRequest)
    }

    public deleteColumn(columnId) {
        const column = {
            columnId
        }
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/column/
delete');
        newRequest.setBody(column);

        return this.call(newRequest)
    }

    public createTask(data, users) {
        const { priority, title, description, estimation, timeSpent, columnId } = dat
a;

        const newTask = {
            priority,
            title,
            description,
            estimation,
```

```
            time_spent: timeSpent,
            columnId,
            users,
        };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/task/ad
d');

        newRequest.setBody(newTask);

        return this.call(newRequest)
    }

    public editTask(data, users, taskId) {
        const { priority, title, description, estimation, timeSpent, columnId } = dat
a;

        const task = {
            priority,
            title,
            description,
            estimation,
            time_spent: timeSpent,
            columnId,
            taskId,
            users,
        };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/task/ed
it');

        newRequest.setBody(task);

        return this.call(newRequest)
    }

    public changeTaskStatus(status, taskId) {

        const task = {
            status,
            taskId
        };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/task/ch
ange-status');

        newRequest.setBody(task);

        return this.call(newRequest)
    }

    public getTask(id) {

        const body = {
```

```
            task: id
        };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/task/ge
t');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

    public changeColumn(task, column) {
        const body = { task, column };
        const newRequest: RequestModel = new RequestModel(Methods.POST, '/api/task/ch
ange-column');

        newRequest.setBody(body);

        return this.call(newRequest)
    }

}
```

6. Реалізація сервісу LanguageService

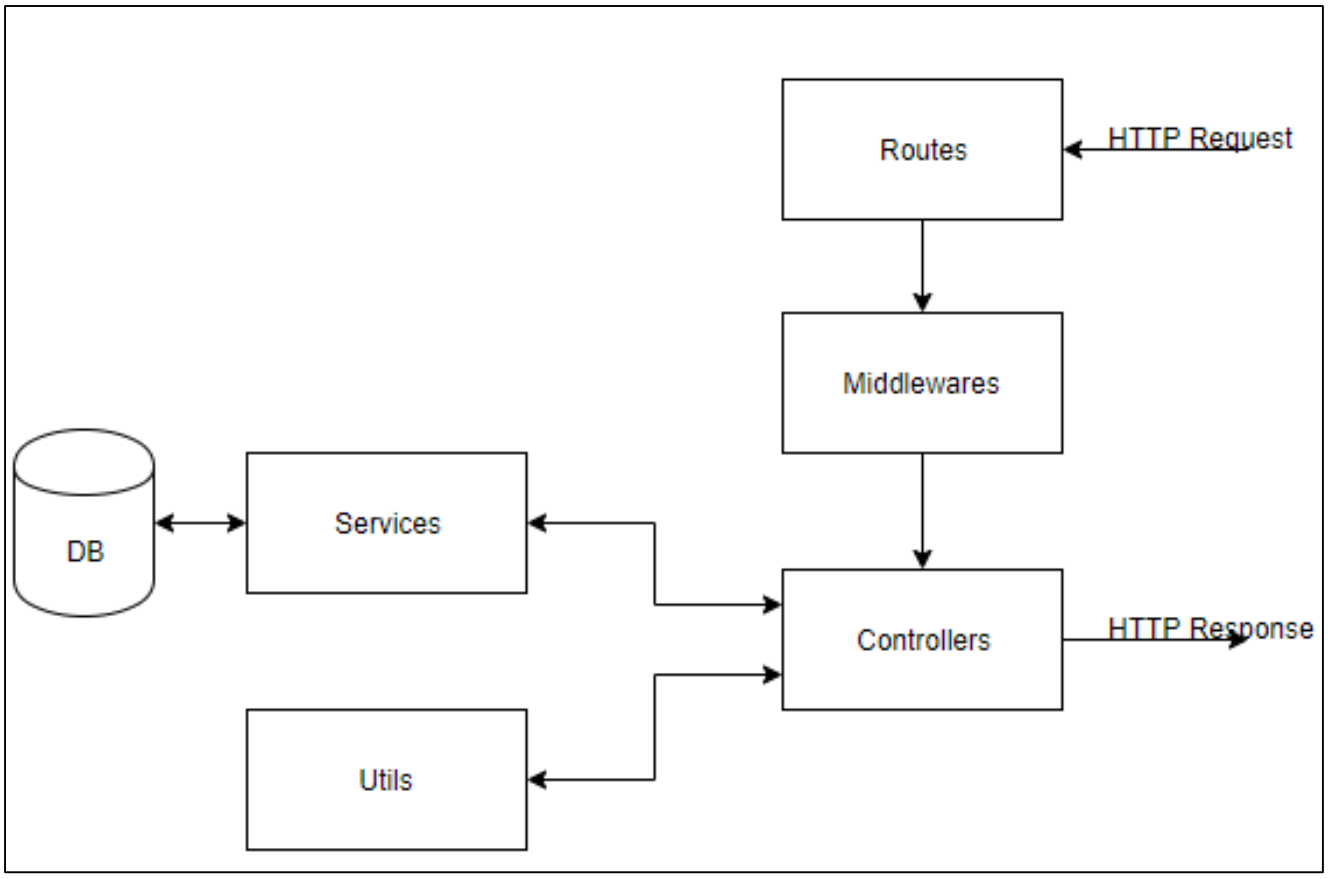
```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
import { TranslateService } from '@ngx-translate/core';
import { ILanguage } from '../interfaces/language.interface';
import { environment } from 'src/environments/environment';

const languagesList = {
  en: {
    value: 'en',
    label: 'main.langs.en',
    image: '/assets/images/langs/en.svg'
  },
  ua: {
    value: 'ua',
    label: 'main.langs.ua',
    image: '/assets/images/langs/ua.svg'
  },
  zh: {
    value: 'zh',
    label: 'main.langs.zh',
    image: '/assets/images/langs/zh.svg'
  },
}

@Injectable()
```

```
export class LanguageService {

  private languageSubject = new BehaviorSubject<ILanguage>(null);
  public language$ = this.languageSubject.asObservable();

  constructor(
    private translateService: TranslateService,
  ) { }

  public setDefaultLanguage() {
    const defaultLang = this.getDefaultLanguage();

    this.setLanguage(defaultLang.value);
  }

  public setLanguage(newLanguage: string): void {
    const supportLanguages: Array<ILanguage> = this.getSupportLangs();
    const language = supportLanguages.find(lang => lang.value === newLanguage);

    if (language) {
      this.translateService.use(language.value);
      this.languageSubject.next(language);
    }
  }

  public getLanguage(): ILanguage {
    return this.languageSubject.getValue();
  }

  public getSupportLangs(): Array<ILanguage> {
    const langs: Array<ILanguage> = environment.supportLangs
      .map(langKey => {
        return languagesList[langKey]
      });

    return langs;
  }

  public getDefaultLanguage(): ILanguage {
    const languages: Array<ILanguage> = this.getSupportLangs();
    const languageFromBrowser = this.translateService.getBrowserLang();
    const defaultLanguage: ILanguage = languages.find(language => language.value ===
environment.defaultLanguage)

    return languages.find(language => language.value === languageFromBrowser) || defa
ultLanguage;
  }
}
```

Додаток Г

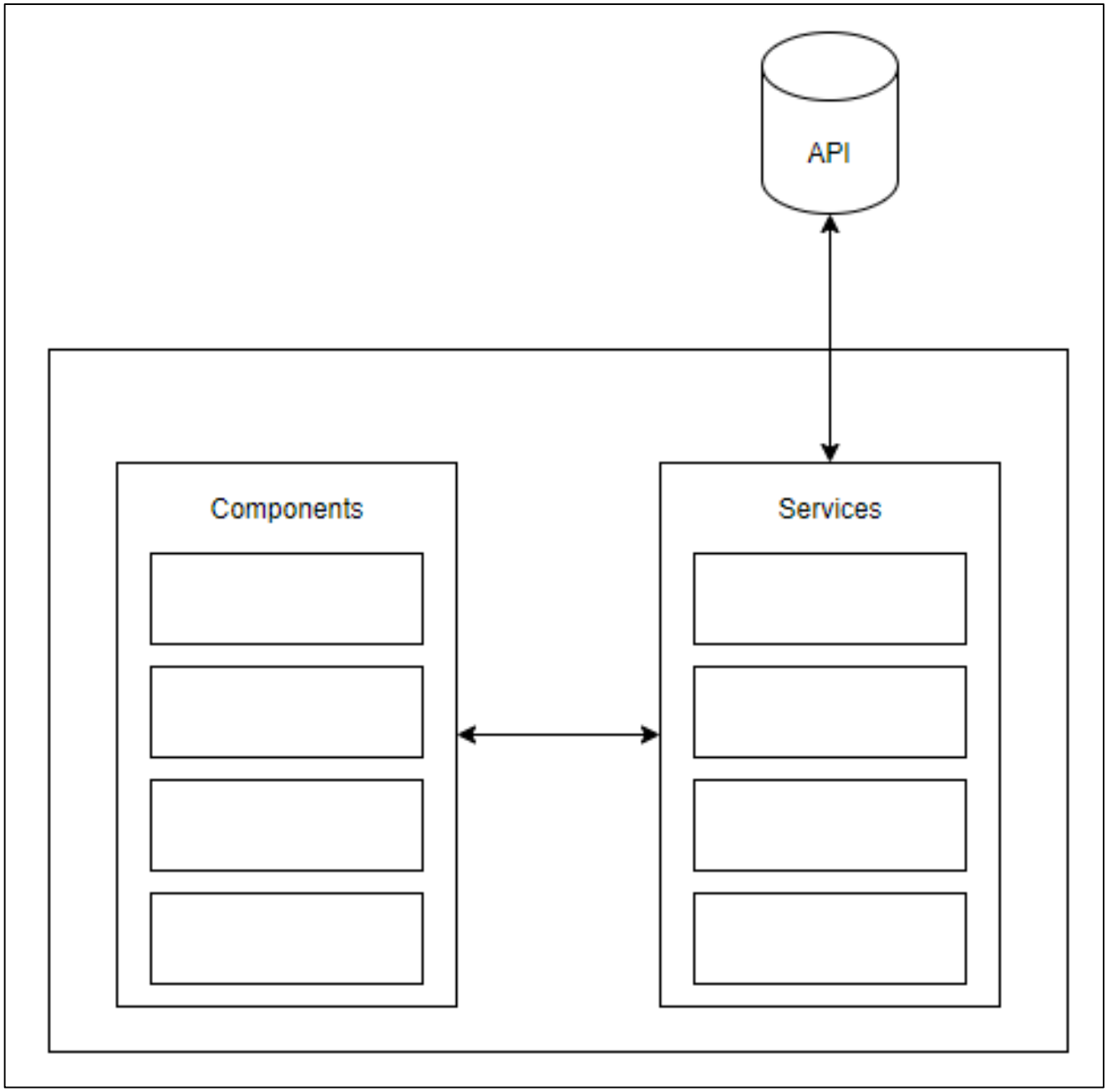
Графічна частина

Структурна схема серверної частини інформаційної системи



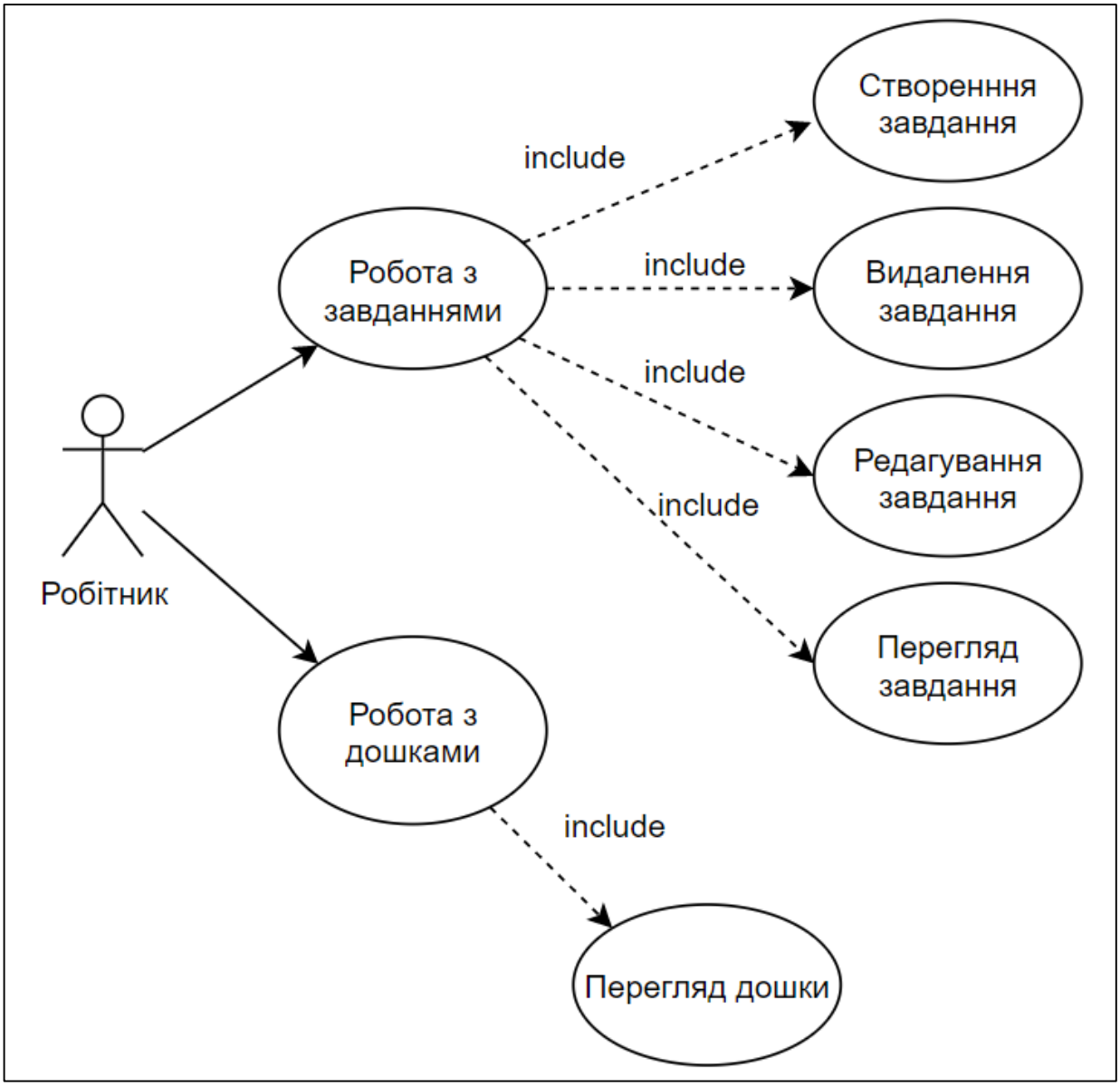
					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (1)	Аркушів (10)	
Перевірив		Мокін О.Б.			Структурна схема серверної частини інформаційної системи	2ІСТ-19м		
Рецензент		Бойко О. Р.						
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

Структурна схема клієнтської частини інформаційної системи



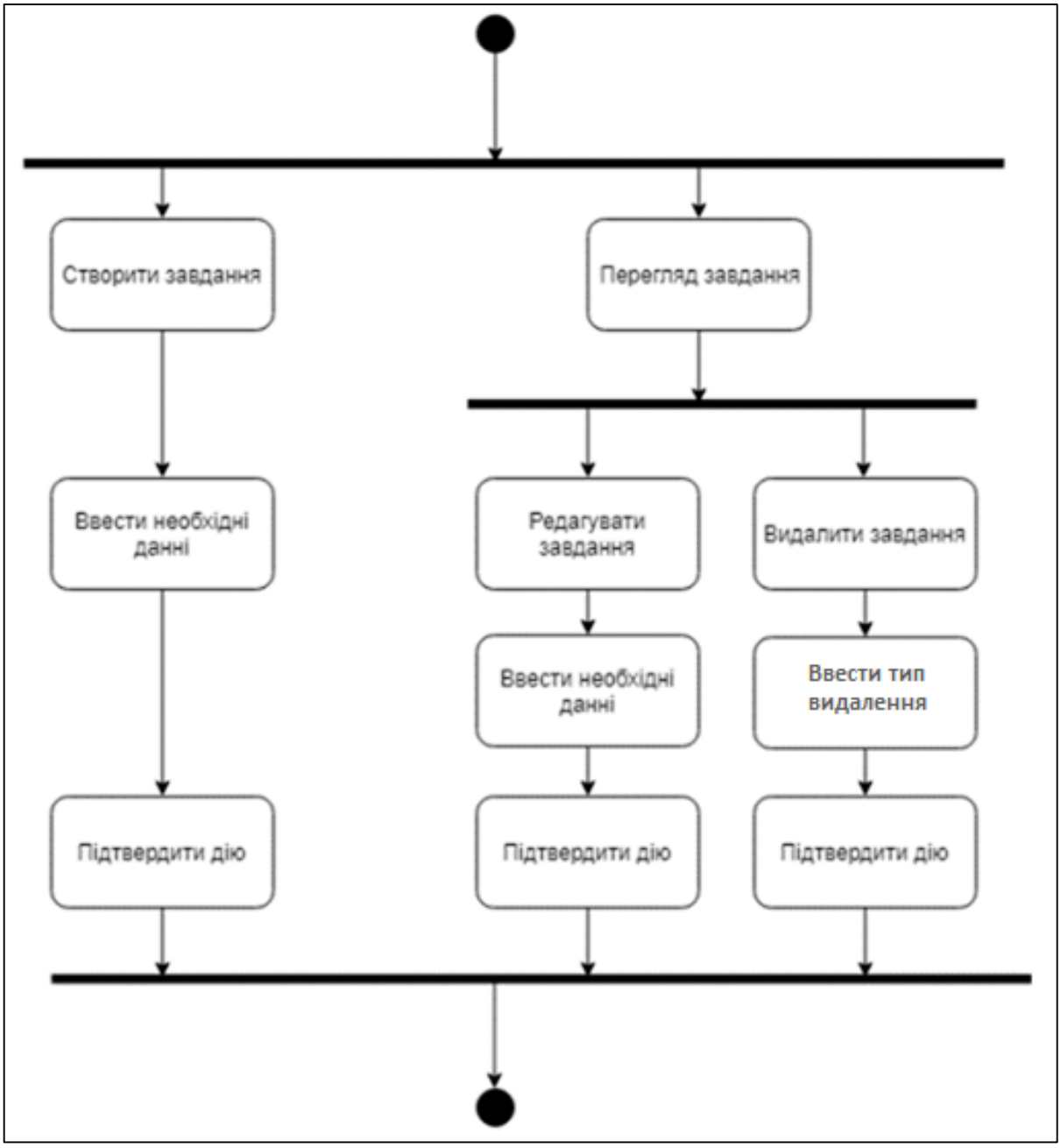
					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (2)	Аркушів (10)	
Перевірив		Мокін О.Б.						
Рецензент		Бойко О. Р.						
					Структурна схема клієнтської частини інформаційної системи	2ІСТ-19м		
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

UML діаграма прецедентів робітника



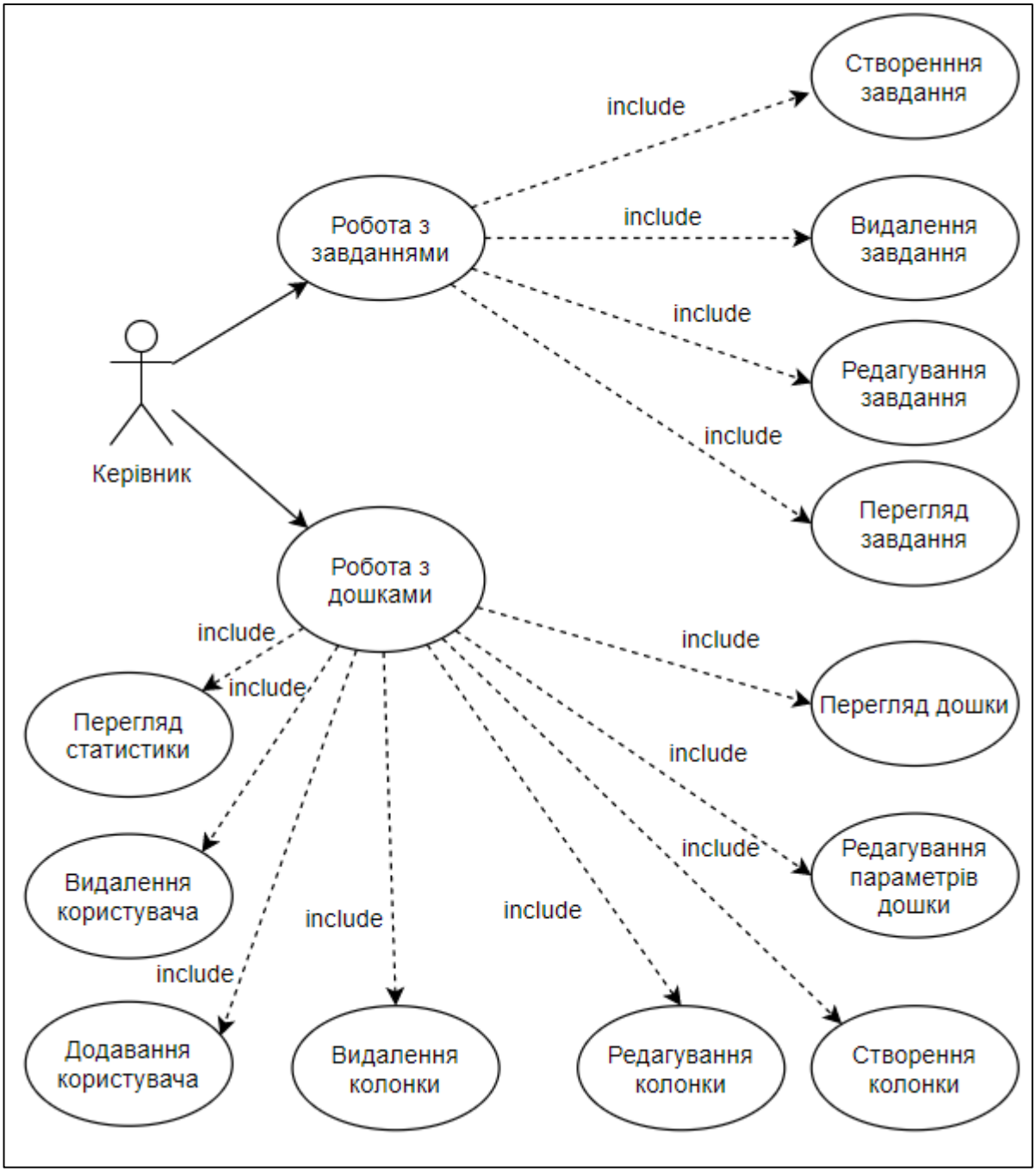
					08–53.МКР.002.02.00.ПЛ	
					Інформаційна система управління проєктами	Літ. / Маса / Масштаб: 1 : 1
Зм.	Арк.	№ докум.	Підпис	Дата		Аркуш (3) / Аркушів (10)
Розробив		Лишак О.М.			UML діаграма прецедентів робітника	2ІСТ-19м
Перевірив		Мокін О.Б.				
Рецензент		Бойко О. Р.				
Н. Контр.		Жуков С.О.				
Зав. каф.		Мокін В.Б.				

Діаграма діяльності процесу додавання та зміни завдань робітником



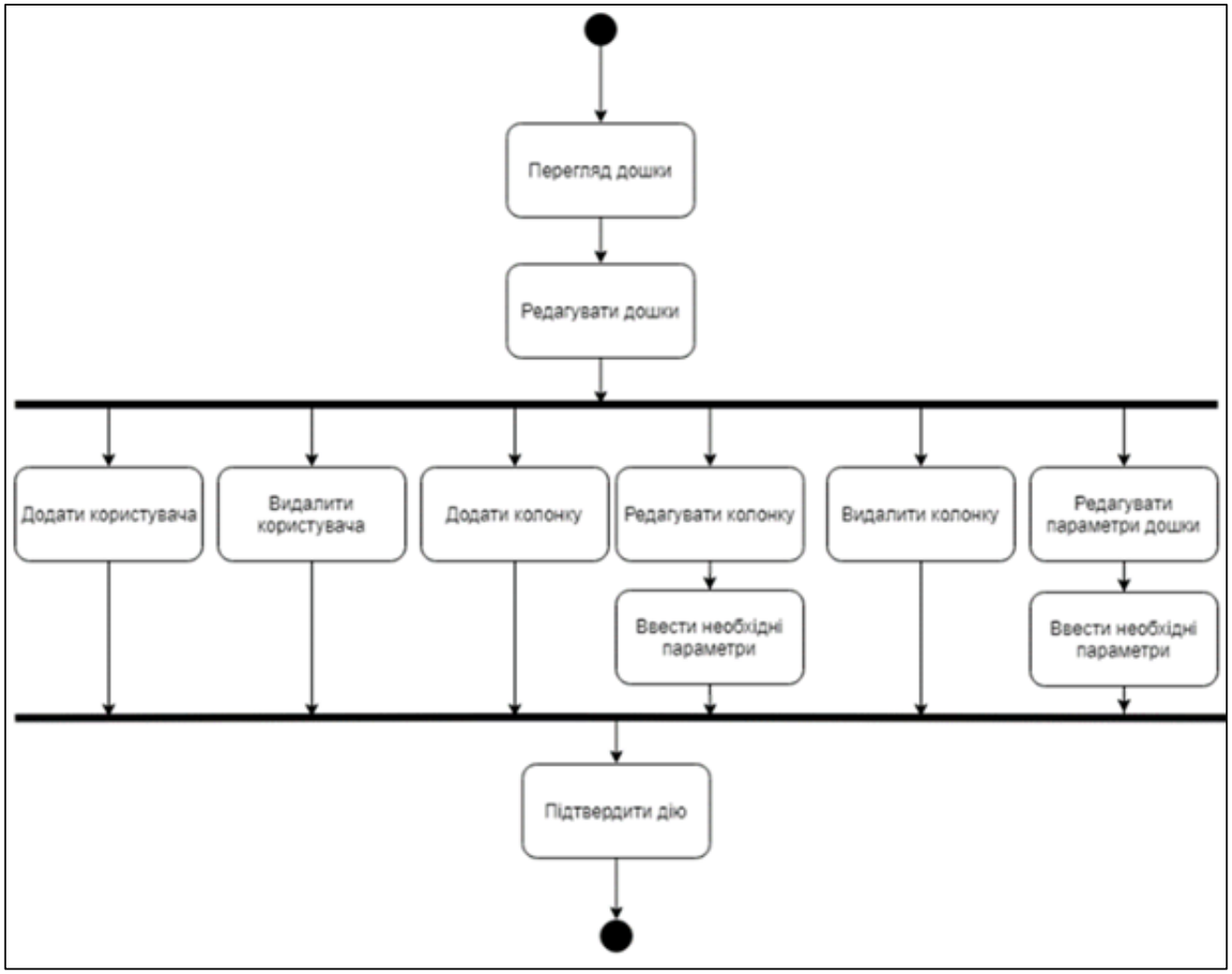
					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (4)	Аркушів (10)	
Перевірив		Мокін О.Б.			Діаграма діяльності процесу додавання та зміни завдань робітником	2ІСТ-19м		
Рецензент		Бойко О. Р.						
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

UML діаграма прецедентів керівника



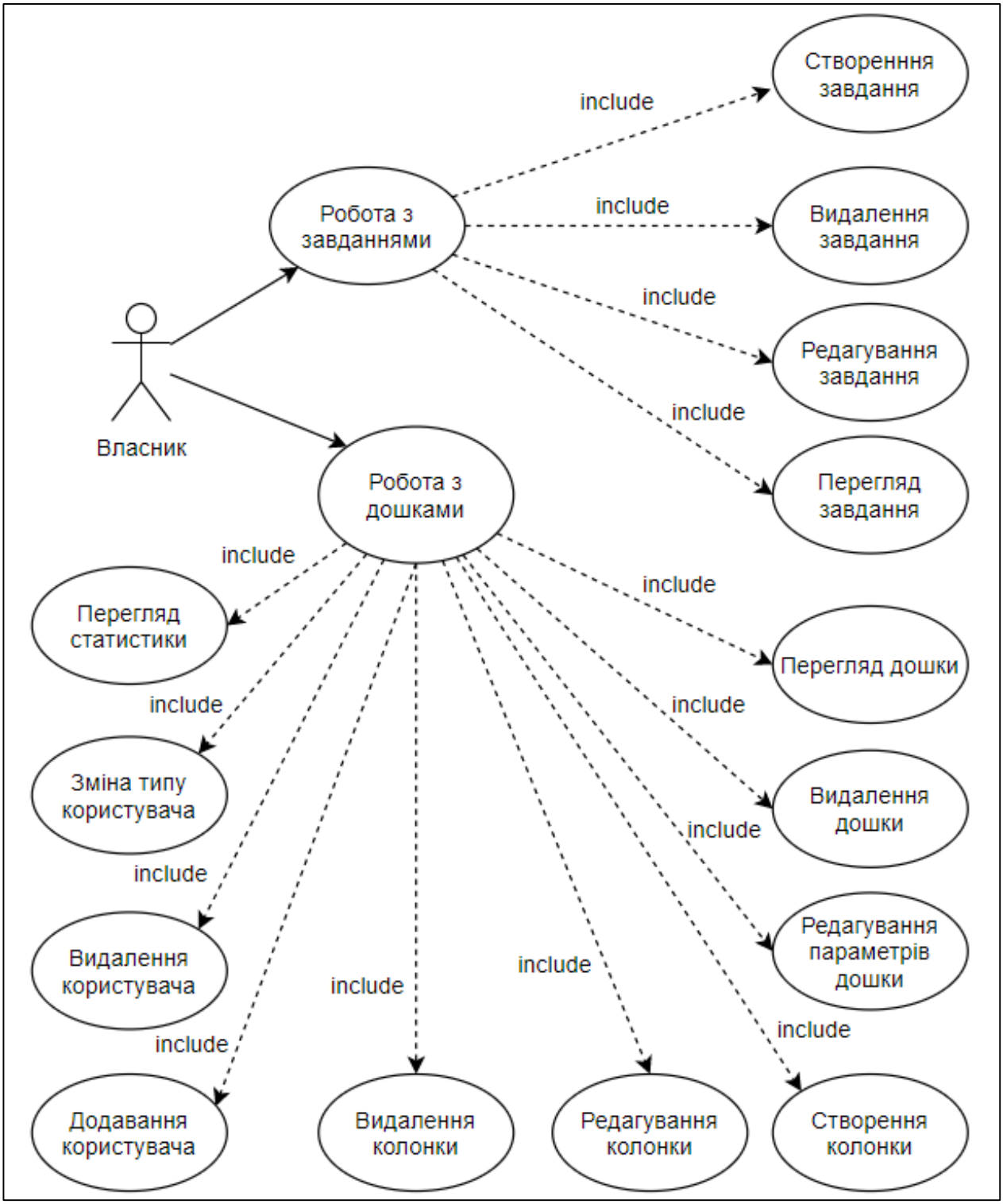
					08–53.МКР.002.02.00.ПЛ	
					Інформаційна система управління проєктами	Літ. / Маса / Масштаб: 1 : 1
Зм.	Арк.	№ докум.	Підпис	Дата		Аркуш (5) / Аркушів (10)
Розробив		Лишак О.М.			UML діаграма прецедентів керівника	2ІСТ-19м
Перевірив		Мокін О.Б.				
Рецензент		Бойко О. Р.				
Н. Контр.		Жуков С.О.				
Зав. каф.		Мокін В.Б.				

Діаграма діяльності процесу редагування дошки керівником



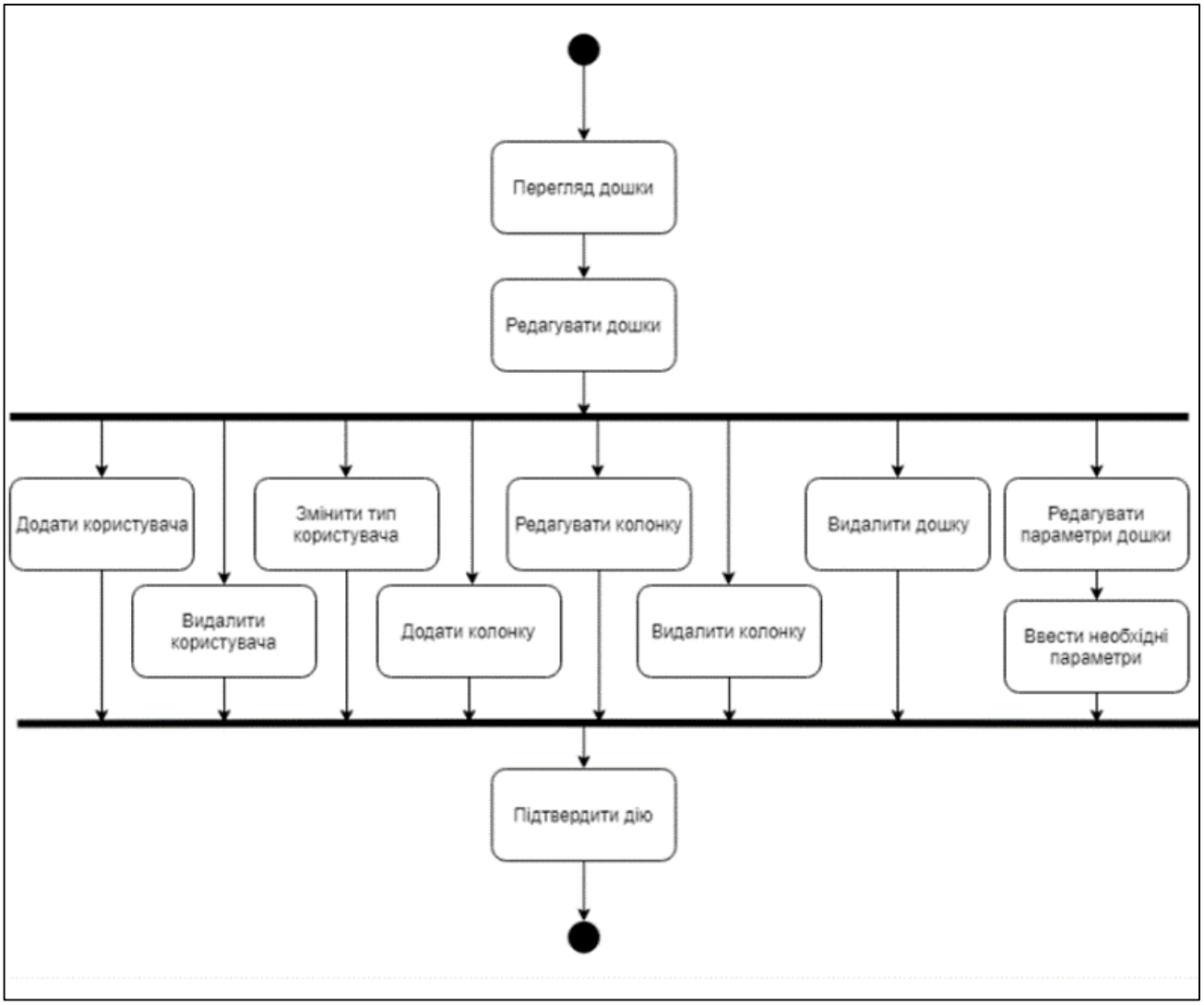
					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (6)	Аркушів (10)	
Перевірив		Мокін О.Б.			Діаграма діяльності процесу редагування дошки керівником	2ІСТ-19м		
Рецензент		Бойко О. Р.						
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

UML діаграма прецедентів власника



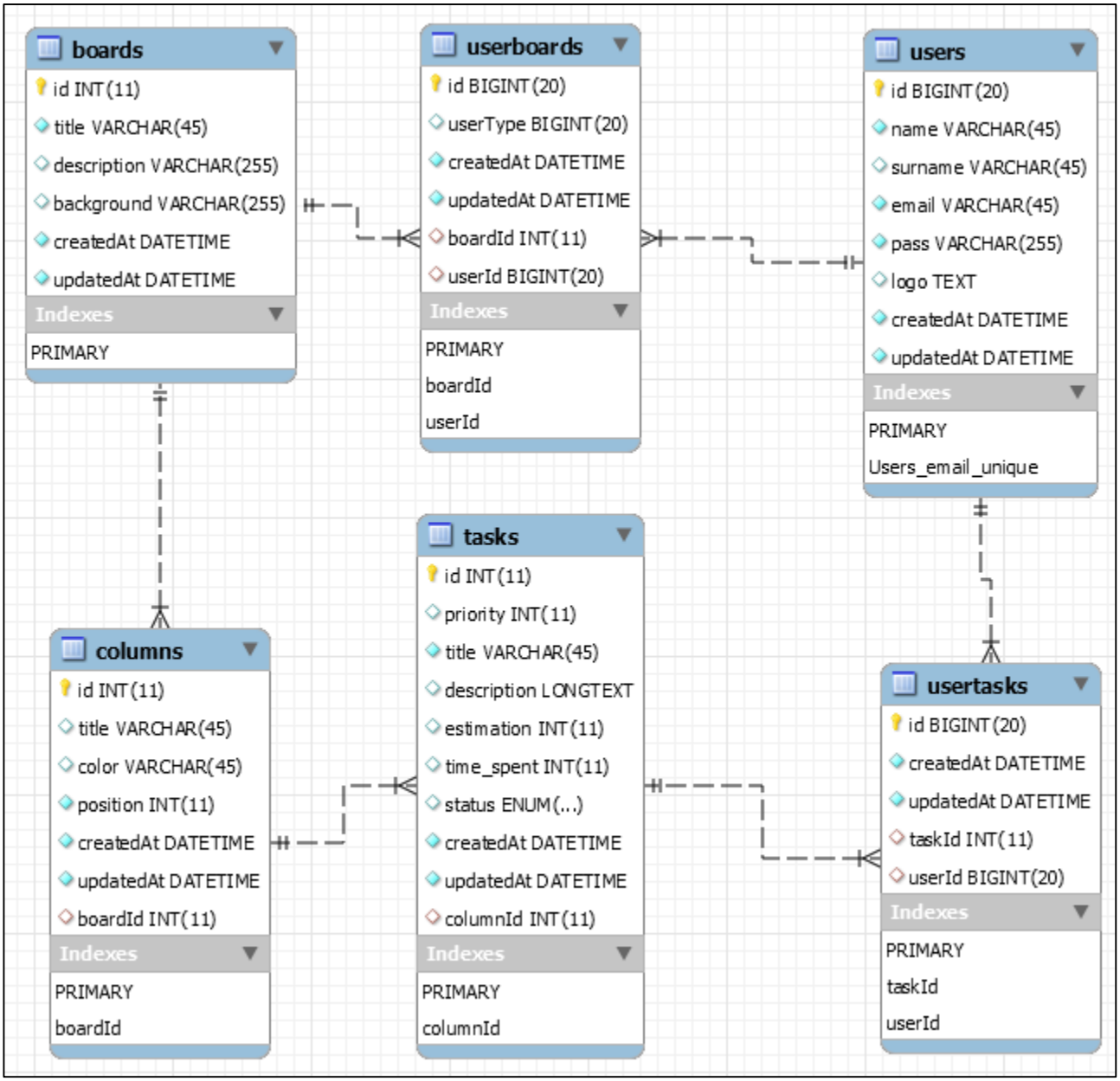
					08–53.МКР.002.02.00.ПЛ	
					Інформаційна система управління проєктами	Літ. / Маса / Масштаб: 1 : 1
Зм.	Арк.	№ докум.	Підпис	Дата		Аркуш (7) / Аркушів (10)
Розробив		Лишак О.М.			UML діаграма прецедентів власника	2ІСТ-19м
Перевірив		Мокін О.Б.				
Рецензент		Бойко О. Р.				
Н. Контр.		Жуков С.О.				
Зав. каф.		Мокін В.Б.				

Діаграма діяльності процесу редагування дошки власником



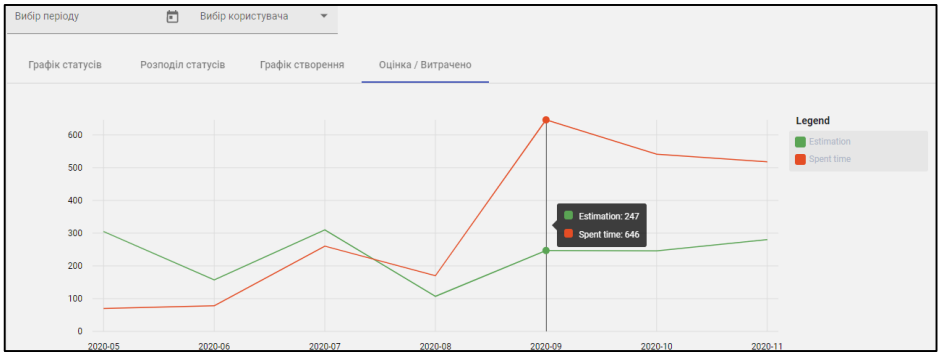
					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (8)	Аркушів (10)	
Перевірив		Мокін О.Б.			Діаграма діяльності процесу редагування дошки власником	2ІСТ-19м		
Рецензент		Бойко О. Р.						
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

UML діаграма класів бази даних інформаційної системи управління проєктами

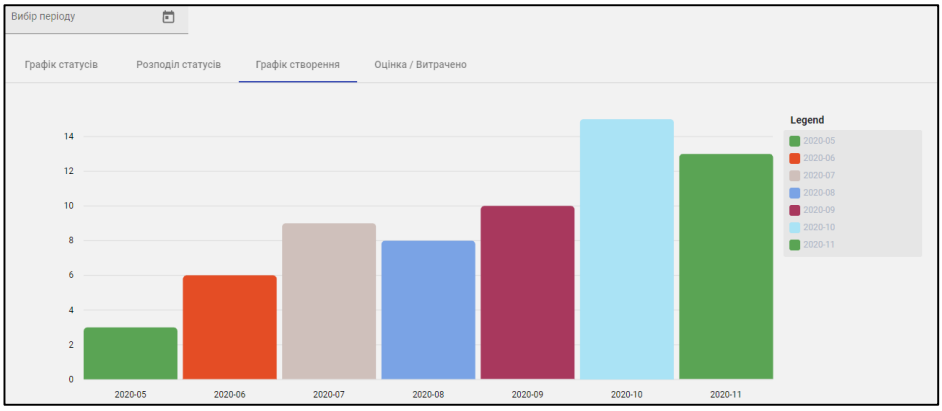


					08–53.МКР.002.02.00.ПЛ		
					Інформаційна система управління проєктами	Літ.	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				1 : 1
Розробив		Лишак О.М.				Аркуш (9)	Аркушів (10)	
Перевірив		Мокін О.Б.						
Рецензент		Бойко О. Р.						
					UML діаграма класів бази даних інформаційної системи управління проєктами	2ІСТ-19м		
Н. Контр.		Жуков С.О.						
Зав. каф.		Мокін В.Б.						

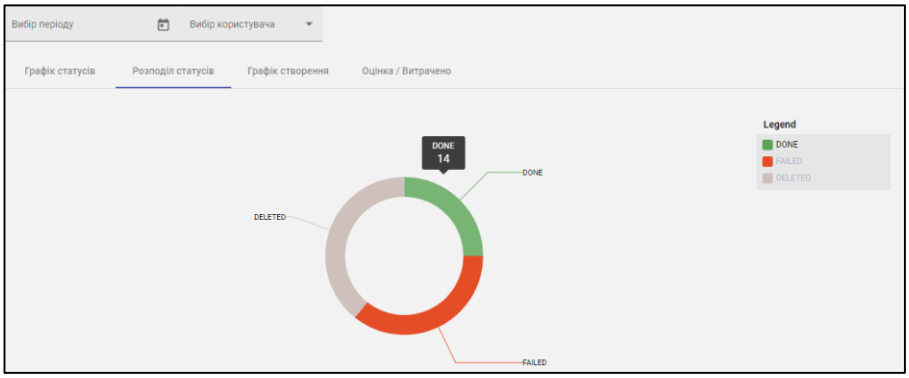
Вкладки сторінки статистики



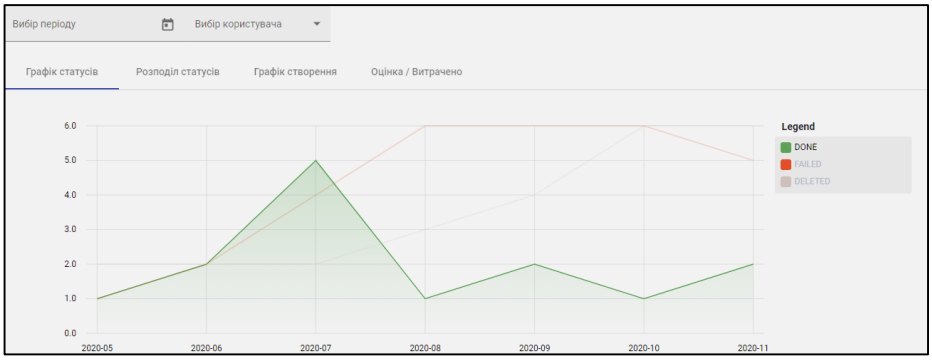
Вкладка «Оцінка / Витрачено»



Вкладка «Графік створення»



Вкладка «Розподіл статусів»



Вкладка «Графік створення»

					08–53.МКР.002.02.00.ПЛ	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Лишак О.М.			Інформаційна система управління проєктами	Літ. / Маса / Масштаб: 1 : 1
Перевірив		Мокін О.Б.				Аркуш (10) / Аркушів (10)
Рецензент		Бойко О. Р.				
					Вкладки сторінки статистики	2ІСТ-19м
Н. Контр.		Жуков С.О.				
Зав. каф.		Мокін В.Б.				