МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту (декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (прізвище та ініціали)	(підпис) (прізвище та ініціали)
«02» ____червня_____ 2025 р.	«02» ____червня_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код і назва спеціальності)
освітньо-професійної програми Комп'ютерні науки
на тему: «Розроблення вебдодатку моніторингу виконання завдань компанії з розробки програмного забезпечення "Rain"»

Виконав: здобувач 4 курсу, групи КН-4-3 .
Харченко Даніл Євгенійович _____
(прізвище, ім'я та по батькові повністю) (підпис)

Керівник Ліманська Наталія Володимирівна _____
(прізвище, ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ — 2025 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 комп'ютерні науки
(код і назва)
Освітньо-професійна програма Комп'ютерні науки
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки _____ Сергій ГРИБКОВ

«28» квітня 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Харченко Даніл Євгенійович
(прізвище, ім'я, по батькові)

1. Тема роботи «Розроблення вебдодатку моніторингу виконання завдань компанії з розробки програмного забезпечення "Rain"»
керівник роботи Ліманська Наталія Володимирівна, старший викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «28» квітня 2025 р. № 254-кс
2. Строк подання здобувачем роботи: 30.05.2025 р.
3. Вихідні дані до роботи: Загальна інформація про компанію, дані про структуру підприємства
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1. Загальна характеристика компанії
2. Структура організації
3. Опис поточних бізнес-процесів
4. Виявлені проблеми
5. Огляд існуючих рішень
6. Задачі на автоматизацію
7. Економічний ефект від впровадження нової системи
8. Технічне завдання
9. Опис функцій системи
10. Проектування бази даних
11. Реалізація описаних функцій

5. Перелік графічного матеріалу:
Моделі бази даних

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	ст. викладач Лімaнська Н.В.	28.04.25	12.05.2025
2	ст. викладач Лімaнська Н.В.	28.04.25	19.05.2025
3	ст. викладач Лімaнська Н.В.	28.04.25	26.05.2025

7. Дата видачі завдання: 28 квітня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз структури підприємства	29.04.2025	
2	Постановка задачі	30.04.2025	
3	Створення бізнес-моделі процесів	01.05.2025	
4	Огляд рішень-аналогів	02.05.2025	
5	Створення технічного завдання	03.05.2025	
6	Проектування БД	04.05.2025	
7	Створення інформаційної системи	06.05.2025	
8	Створення інструкції користувача	11.05.2025	
9	Оформлення пояснювальної записки	11.05.2025	
10	Створення презентації	16.05.2025	

Здобувач _____Харченко Даніл_____

Керівник роботи _____Наталія Лімaнська _____

Анотація

Метою кваліфікаційної роботи є закріплення знань і практичних навичок розробки інформаційних систем, зокрема інтерфейсу користувача, відповідно до сучасних принципів побудови програмного забезпечення.

У межах проєкту реалізовано інформаційну систему, яка дає змогу працювати з даними про проекти, співробітників (включаючи їх статус, деталі контракту та обладнання), команди розробників та клієнтів. Розробка включала моделювання бази даних, її реалізацію у середовищі системи управління базами даних, а також створення клієнтського застосунку з графічним інтерфейсом та відповідної серверної частини.

Для реалізації бази даних було використано реляційну СУБД PostgreSQL. Для створення інтерфейсу користувача та серверної логіки — мову програмування Java з використанням фреймворків JavaFX (для графічного інтерфейсу) та Spring Boot (для бізнес-логіки та роботи з даними за допомогою Spring Data JPA).

Дипломний проект обсягом у 72 сторінок, містить 74 сторінок пояснювальної записки, 2 сторінки графічних матеріалів, 31 літературних джерел.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, КЕРУВАННЯ ПРОЕКТАМИ, КЕРУВАННЯ ПЕРСОНАЛОМ, БАЗА ДАНИХ, ІНТЕРФЕЙС КОРИСТУВАЧА, JAVA, JAVAFX, SPRING BOOT, POSTGRESQL, СУБД, JPA

SUMMARY

The objective of the course project is to consolidate knowledge and practical skills in developing information systems, particularly user interfaces, in accordance with modern software development principles.

Within the project, an information system has been implemented that allows working with data on projects, employees (including their status, contract details, and equipment), development teams, and clients. The development included database modeling, its implementation within a database management system environment, and the creation of a client application with a graphical user interface and a corresponding server-side component.

The **PostgreSQL** relational DBMS was used for the database implementation. The **Java** programming language was used to create the user interface and server-side logic, utilizing the **JavaFX** framework (for the graphical interface) and the **Spring Boot** framework (for business logic and data handling via Spring Data JPA).

The course project is 72 pages long, contains 74 pages of explanatory notes, 3 pages of graphic materials, and 31 literary sources.

Keywords: INFORMATION SYSTEM, PROJECT MANAGEMENT, PERSONNEL MANAGEMENT, DATABASE, USER INTERFACE, JAVA, JAVAFX, SPRING BOOT, POSTGRESQL, DBMS, JPA.

ЗМІСТ

ВСТУП

У сучасному світі індустрія інформаційних технологій (IT) та розробки програмного забезпечення займає провідне місце серед галузей, що стрімко розвиваються та визначають економічний і технологічний прогрес. Постійне зростання складності програмних продуктів, скорочення термінів виходу на ринок, швидка зміна технологій, необхідність ефективного управління розподіленими командами та зростаючі очікування клієнтів ставлять перед IT-компаніями та відділами розробки нові виклики. Для ефективного функціонування та збереження конкурентоспроможності підприємствам у цій сфері необхідно не лише володіти передовими технологіями, а й забезпечити прозоре, кероване та надійне виконання проектів.

Важливу роль у досягненні цих цілей відіграють сучасні інформаційні системи управління проектами та ресурсами. Вони дозволяють автоматизувати ключові бізнес-процеси життєвого циклу розробки, оптимізувати розподіл завдань та керування ресурсами (як людськими, так і технічними), підвищити якість моніторингу виконання проектів, мінімізувати людський фактор у рутинних операціях та забезпечити швидкий доступ до актуальної інформації про статус проектів, завантаженість команд та співробітників. Завдяки впровадженню таких програмних рішень компанії можуть ефективно зберігати та аналізувати дані про клієнтів, проекти, вимоги, завдання, команди розробників, навички співробітників, використане обладнання тощо.

Інформаційні технології в управлінні розробкою також сприяють інтеграції з іншими інструментами (системами контролю версій, CI/CD, баг-трекерами), впровадженню методологій гнучкої розробки, формуванню аналітичної звітності (наприклад, про хід виконання проекту, витрачений час, бюджет) та вдосконаленню взаємодії з клієнтами. Використання баз даних, спеціалізованих інтерфейсів користувача та програмного забезпечення дозволяє не лише

спростити повсякденні операції з управління проектами та персоналом, а й забезпечити надійність збереження інформації, її безпечну обробку та простежуваність змін.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

В наш час, коли умови динамічного розвитку індустрії інформаційних технологій, є дійсно важливою та невід'ємною часткою успішного проєкту, ключовим фактором є правильне управління проєктами, людськими ресурсами та взаємодією з клієнтом. І це становить основну частину успіху, і метою розробки є розробити таке програмне рішення, яке буде призначене для оптимізації та контролю операційної діяльності IT-компанії або окремого відділу розробки.

Розроблювана система буде сфокусована на керуванні життєвим циклом програми або задач, які будуть виконуватися для розробки певного програмного забезпечення.

Ключовим завданням розробки системи є надання потужного інструменту для того, щоб підвищити ефективність роботи співробітників. Це досягається шляхом впровадження так званої баг-трекінгової системи зі зрозумілим простим графічним інтерфейсом. Система даватиме можливість для виконання всіх основних операцій, наприклад, як CRUD (створення, читання, оновлення, видалення) над ключовими об'єктами. Під об'єктами системи розуміються завдання.

1.1. Загальна характеристика ТБК-ЕНЕРГІЯ

Предметом діяльності ТБК "Енергія"

- проектування та монтаж систем освітлення;
- будівництво та електрифікація житлових будинків;
- надання послуг з обслуговування електромереж;
- консультаційні послуги у сфері енергоефективності;
- торгівельна діяльність, в тому числі продаж електрообладнання;
- розробка та впровадження автоматизованих систем управління;

- проектування та встановлення "розумних" систем освітлення;
- надання послуг з енергоаудиту.

Таблиця 1.1. Загальна інформація про підприємство

Скорочена назва	ТОВ "ТБК-ЕНЕРГІЯ"
ЄДРПОУ	33501576
Зареєстрований:	04.05.2005
Діялність	43.22 Електромонтажні роботи 43.22 Монтаж водопровідних мереж, систем опалення та кондиціонування 43.29 Інші будівельно-монтажні роботи 46.13 Діяльність посередників у торгівлі деревиною, будівельними матеріалами та санітарно-технічними виробами 93.19 Інша діяльність у сфері спорту 41.20 Будівництво житлових і нежитлових будівель
Керівник	ХАРЧЕНКО ІГОР ДМИТРОВИЧ, Директор

Основними напрямками діяльності компанії є: монтаж освітлення, будівництво житлових об'єктів, сервісне обслуговування, впровадження автоматизованих систем. Компанія обслуговує понад 100 об'єктів. Всі роботи виконуються відповідно до державних стандартів та проходять перевірку за різними технічними показниками власною службою контролю якості.

1.2 Організаційна структура підприемства “Rain”

Головний директор компанії має в управлінні декілька ключових відділів компанії (Рис. 1.1), таких як служба монтування світла, служба управління обслуговування клієнтів, служба бухгалтерії та служба ІТ відділу на

підприємстві. На підприємстві є наявна структура, за якою кожен з відділів підпорядковується безпосередньо одному керівнику

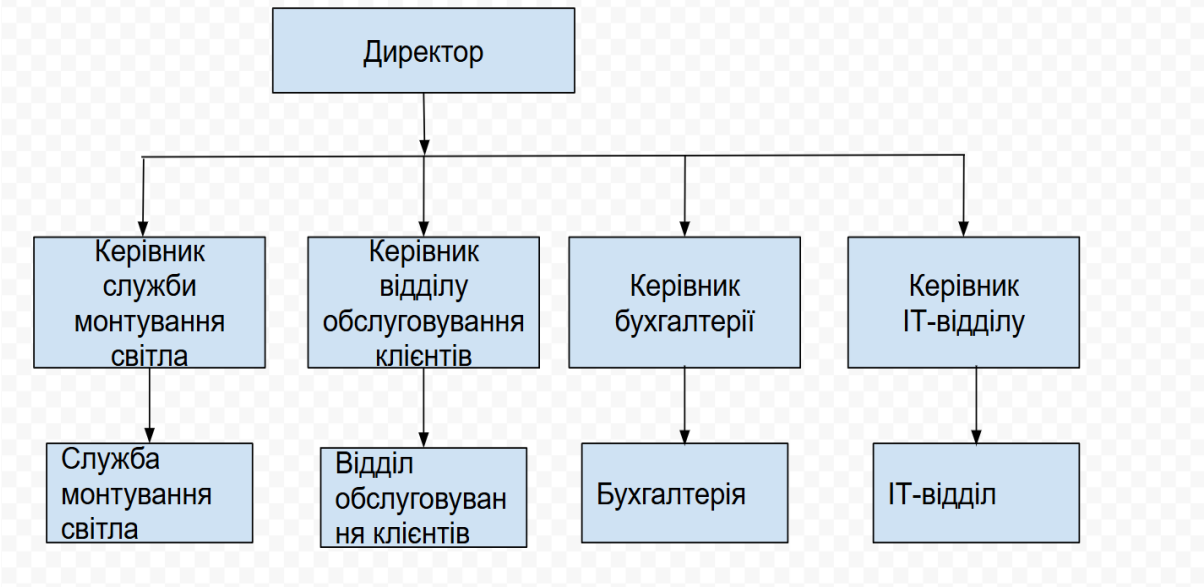

Рисунок 1.1— Організаційна структура підприємства

Наразі розділи взаємодіють за допомогою сторонніх ресурсів, таких як месенджери або телефонні розмови, але в подальшому буде закладено як фундамент те, що всі комунікації та робочі моменти відбуватимуться через розроблену систему. На ранніх етапах розробки вона буде сфокусована на IT відділі, але в подальшому її можна буде легко масштабувати, оскільки вибрано дуже гнучкі технології.

Основними користувачами системи буде персонал, який залучений до IT відділу, такий як менеджери проєктів, керівники команд та, звісно, співробітники розробки. Менеджери проєктів використовуватимуть систему для відстеження етапів виконання, термінів завдань та залучення людських ресурсів. Керівники команд — для формування команд, розподілу ресурсів та призначення співробітників на ті чи інші завдання. Співробітники використовуватимуть систему для відстеження наявних задач, що їм треба зробити, що вони вже зробили і що вони роблять наразі.

1.2.2 Структура It-відділу

Головні функції іт відділу:

- Розробка та впровадження автоматизації бухгалтерських задач.
- Налаштування та обслуговування вже розроблених проектів.
- Розробка модулів як для існуючих систем, так і для нових.
- Контроль за роботою сервісного обладнання, Забезпечення безпеки даних за допомогою резервного копіювання та шифрування.
- Контролювання несанкціонованого доступу до системи.
- Моніторинг продуктивності системи.
- Технічна підтримка для інших відділів, яким може бути складно розібратися в нових системах.
- Постійне покращення через систему через впровадження нових технологічних рішень.

В it-відділі працюють на таких посадах: Системний адміністратор, Розробники програмного забезпечення, Спеціаліст кібербезпеки та DevOps спеціаліст.

Отже, IT відділ невід'ємний від підприємства, тому що він бере участь в роботі підприємства та взаємодіє з багатьма відділами.

Таблиця 1.2. Інформація про прослуги які надає it-відділ

Відділ	**Що надає ІТ відділ**	**Що отримує ІТ відділ**
Бухгалтерія	Він надає систему, яка є автоматизованим рішенням для ведення обліку та бухгалтерії, та технічну підтримку для цієї системи. А також надає інструкцію та допомогу, для того, щоб працівники бухгалтерії могли ефективно працювати з системою.	З бухгалтерії він отримує запити на якісь нові модулі системи та вимоги до формування бухгалтерської звітності, інформацію про можливі помилки в розробленій системі та дані для зберігання в цій системі.
Обслуговування клієнтів	Відділ надає автоматизовану систему, яка може обробляти заявки, та інструменти, які допомагають працівникам відділу аналізувати клієнтські дані.	Від працівників отримує інформацію про потреби для оновлення функціоналу, проаналізовану інформацію, про нові об'єкти системи, пропозиції щодо покращення якогось інтерфейсу системи та інформацію про виявлені помилки.
Служба монтування світла	ІТ відділ надає: Систему слідкування статусу об'єктів, Дані про хід виконання роботи на цих об'єктах, Автоматизоване рішення звітності про виконані об'єкти.	А отримує від служби монтування світла інформацію про статус об'єктів, дані про хід роботи, технічні характеристики та фото- та відеоматеріали для більш детального формування звітності про об'єкти.

1.3 Аналіз нинішнього стану комп’ютеризації

Як не парадоксально, але у цьому відділі немає єдиної централізованої системи баг-трекінгу, що є невід'ємною часткою для правильного функціонування та оцінювання продуктивності, і для забезпечення максимально ефективної роботи над задачами та розробкою нових рішень.

Отже, можна зробити висновок, що наявність такого стану речей є абсолютно неприйнятним для будь-якої їх компанії, а тим паче для цього відділу, який залучений в багатьох процесах компанії. І це призводить до того, що ефективності роботи відділу немає, чітко поставлених задач немає, чітко сформульованих завдань для розробників, і наразі це потребує дійсно покращення, яке й запропонує розроблена система.

1.4 Функціональне моделювання та аналіз існуючих бізнес-процесів.

Аналіз існуючих бізнес-процесів був виконаний під час проходження практики на підприємстві

1.4.1. Функціональна модель відділу розробки

Отже, як ми можемо бачити на рисунку 1.2, детально описані всі бізнес-процеси, які наявні у цьому відділі.

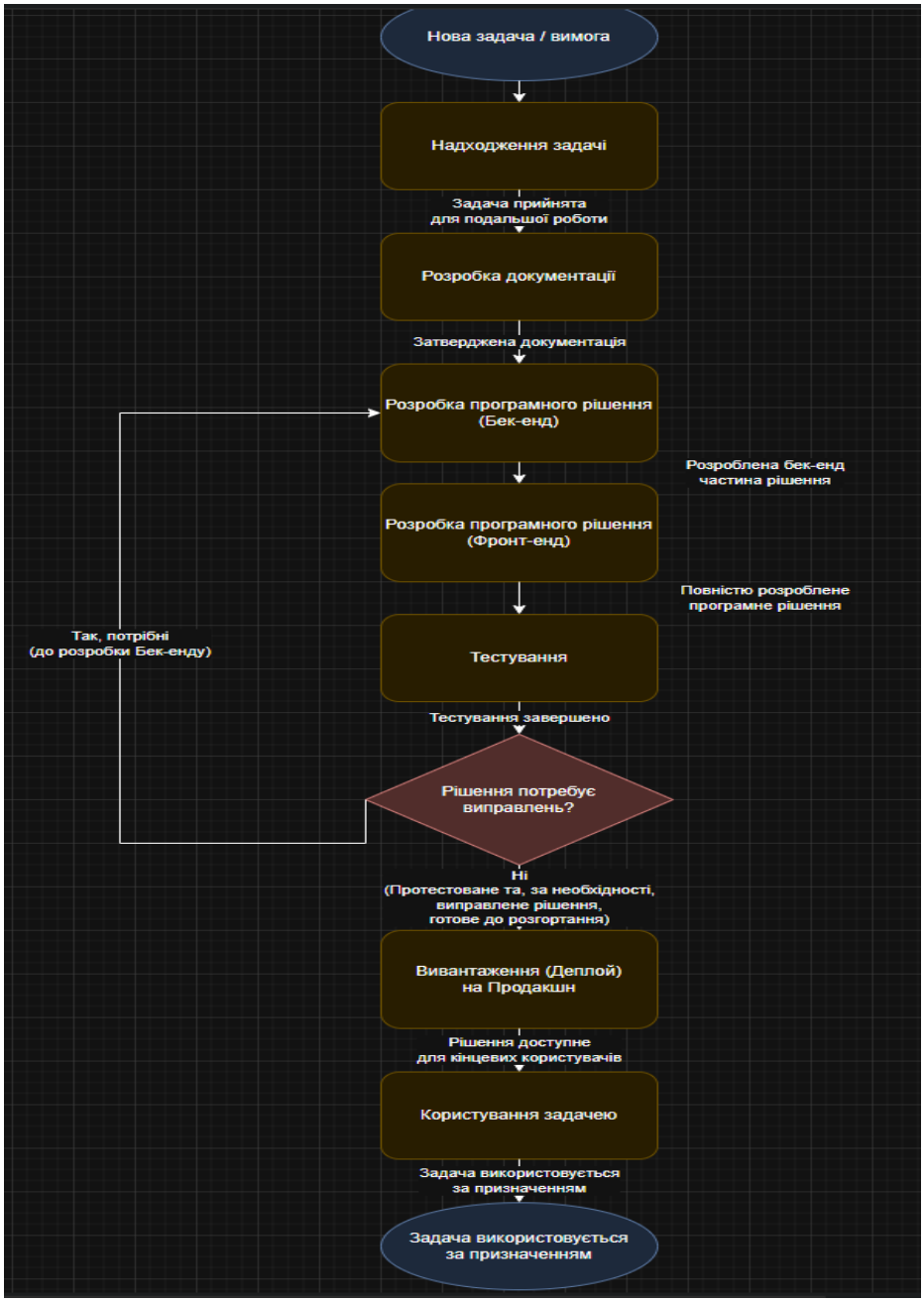

Рисунок 1.2 — Моделювання бізнес-процесів.

Перш за все, це все починається з надходження задачі. Це може бути задача як для розробки нової системи або нового модулю до системи, так і для покращення цієї ж системи, виправлення багів і таке інше. Далі починається етап розробки та документації, що є дійсно дуже важливим, оскільки без документації буде складно зробити так, щоб хтось знав щось про наш продукт. А після написання документації починається розробка програмного рішення на стороні

бек-енду. Також відділ фронтенду починає розробляти систему. Вони роблять систему зі своєї сторони. Далі, це все переходить у тестування. Воно тестується і визначається, треба доробка чи не треба. Якщо доробки не треба, то ми переходимо до фінального етапу. Це впровадження у продакшн, де ми його розгортаємо і завершуємо наш цикл розробки або покращення якоїсь задачі.

1.4.2 Виявлені проблеми.

Після детального дослідження цього відділу, був зроблений такий висновок, що хоч і всі виглядають доволі оптимізованими, але є дещо суттєве, що дійсно заважає максимально ефективно виконувати завдання, які поставлені, — це відсутність будь-якої баг-трекінгової системи. Було відмічено на підприємстві, що це є серйозною проблемою, оскільки у розробників часто виникає потреба аналізувати завдання та уточнювати деталі. Але без централізованої системи знайти тих, хто поставив задачу, довго. Зі сторони менеджерів складно відстежувати, на якому етапі завдання, оскільки треба постійно питати у розробників "update" по завданню, складно оцінити ефективність розробника, оскільки не можна сказати, який об'єм роботи він виконує.

1.4.3 Пропозиції щодо усунення наявних проблем

Можемо бачити на рисунку 1.3 наведену схему ситуації, коли розробнику незрозуміла задача.

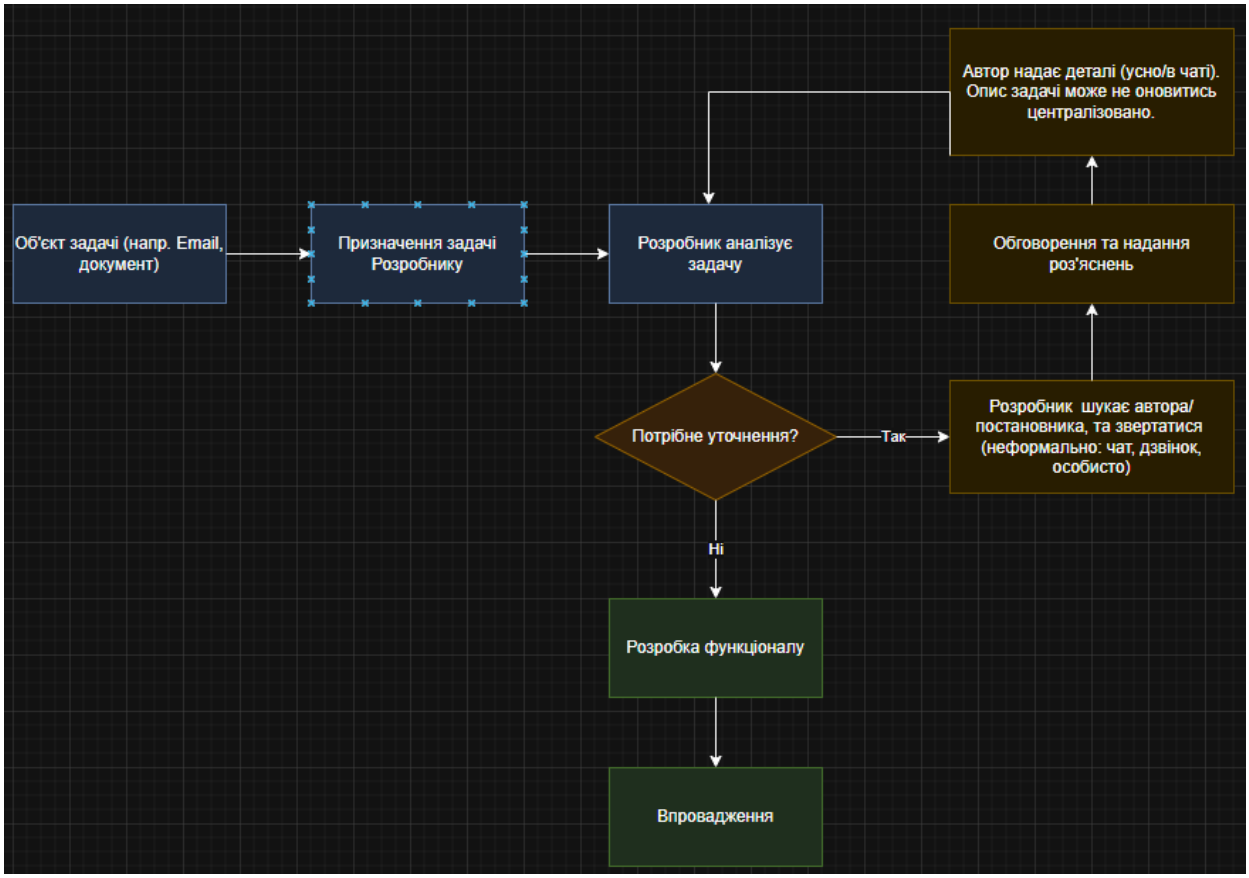

Рисунок 1.3 – Модель процессу AS IS

1. Все починається з того, що розробник отримує завдання через "email" або через "Google Документ", призначається до цього завдання.
2. Далі розробник аналізує завдання. І якщо у розробника потрібні уточнення, що дуже часто виникає, то розробник шукає автора, потім шукає його контактні дані для звернення.
3. Потім автор усно з розробником, вирішують проблеми по завданню.

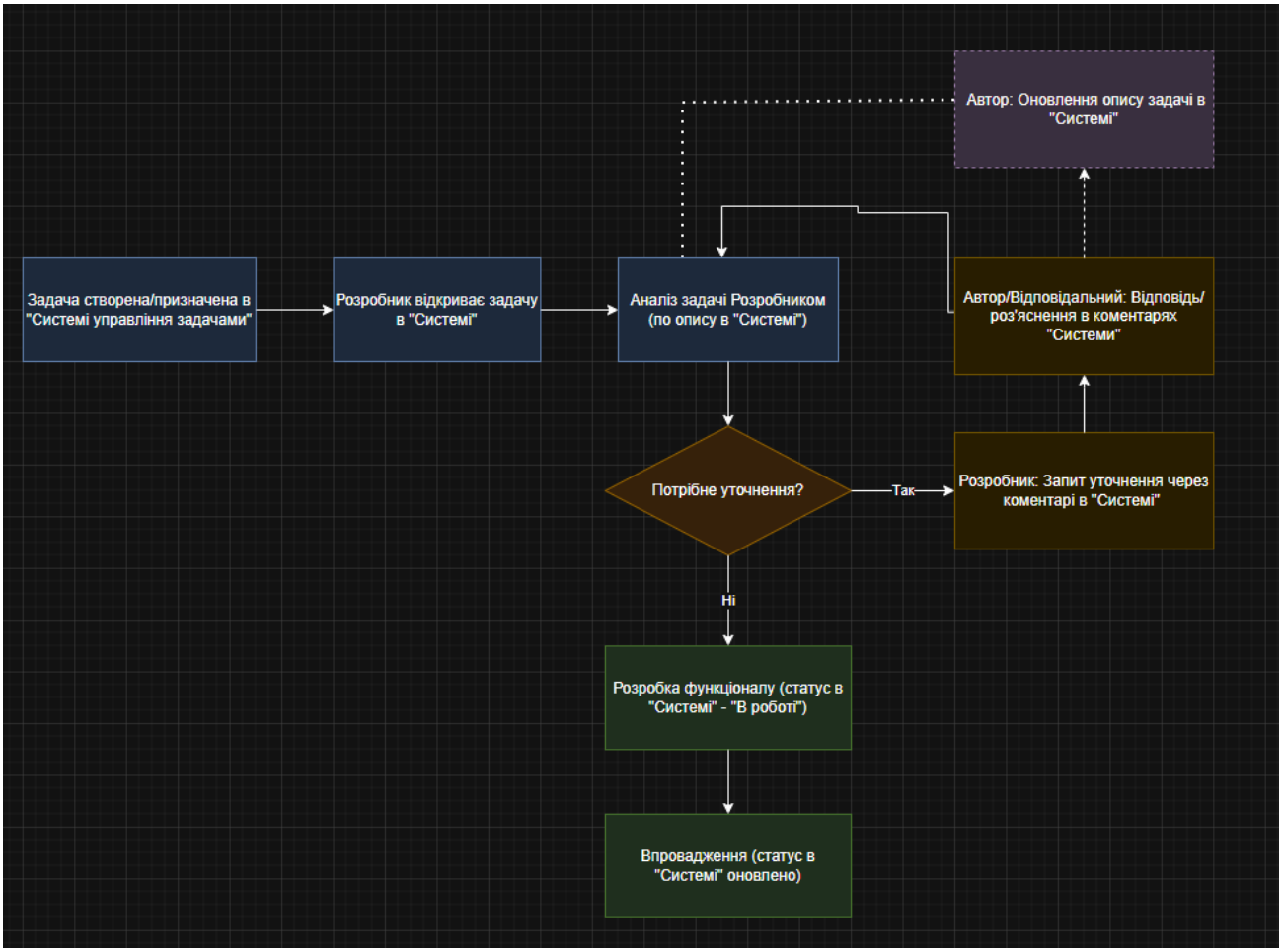

Рисунок 1.4 – Модель TO BE

Можемо бачити на рисунку 1.4 наведену схему ситуації, коли розробнику незрозуміла задача, але з використанням централізованої системи.

Використання "Системи управління задачами" значно структурує та покращує процес уточнення:

Задача в "Системі": Задача створюється та призначається розробнику в "Системі управління задачами". Вона містить опис, критерії приймання, пріоритет та іншу релевантну інформацію. Аналіз задачі в "Системі": Розробник відкриває задачу в "Системі" та аналізує її на основі наданої інформації.

Виявлення незрозумілих моментів:

Запит на уточнення через "Систему": Розробник не йде шукати когось особисто, а ставить свої питання безпосередньо в коментарях до задачі в

"Системі". Він може поачити автора задачі, менеджера або іншу відповідальну особу.

Сповіщення та відповідь в "Системі": Відповідальна особа отримує автоматичне сповіщення з "Системи" про новий коментар/питання. Вона надає відповідь також у коментарях до задачі.

Централізована історія: Все листування, всі питання та відповіді зберігаються в історії задачі. Це доступно всім, хто має доступ до задачі, і може бути використано в майбутньому.

Такожварто виділити такі плюси після введення системи:

1. Економія часу та підвищення продуктивності. Перш за все, можна назвати те, що впровадження баг-трекінгової системи дозволить скоротити час на комунікацію та уточнення деталей деяких завдань з 20-40 хвилин, які наразі було нараховано на проекті, до кількох хвилин. І можна сказати, що розробники можуть миттєво бачити, хто створив завдання, і звісно, до кого звертатися за інформацією про завдання: історія обговорень цього, яке завдання, і які там є фото- та відеоматеріали. За умови, що команда складається з 5 розробників, кожен з яких щодня витрачає в середньому 50 хвилин на уточнення завдань, місячна економія робочого часу становитиме: 5 розробників × 50 хвилин × 22 робочих дні = 5500 хвилин = 91,7 години Це еквівалентно звільненню майже 11 робочих днів одного розробника щомісяця. Прискорення розробки цільового продукту введення подібної системи позитивно вплине на підвищення ефективності роботи команду та безпосередньо буде мати значення на швидкість розробки нашого продукту.
2. Один із найважливіших пунктів, які хотілось би виділити, — це зниження вартості розробки, оскільки підвищення ефективності використання нашого робочого часу воно прямо постійно впливає на те, що розробник програмного забезпечення буде використовувати свій час більш

раціонально. Отже, ми зможемо платити йому тільки за його працю і мінімізувати таким чином його непродуктивні дії. А це насамперед може вплинути на скорочення термінів розробки, що теж може зменшити загальні витрати на проєкт.

3. Зменшення часу на комунікацію.
4. Запобігання повторному виконанню тих самих завдань.
5. Зниження кількості помилок через неправильне розуміння вимог.
6. Краща координація між членами команди.

1.5 Огляд існуючих рішень для розв'язання виявлених проблем

Основна пропозиція — це введення баг-трекінгової системи. І після дослідження ринку можна виділити таких мастодонтів ринку: Jira, Trello та Asana. Тож розглянемо їх дещо детальніше:

Jira

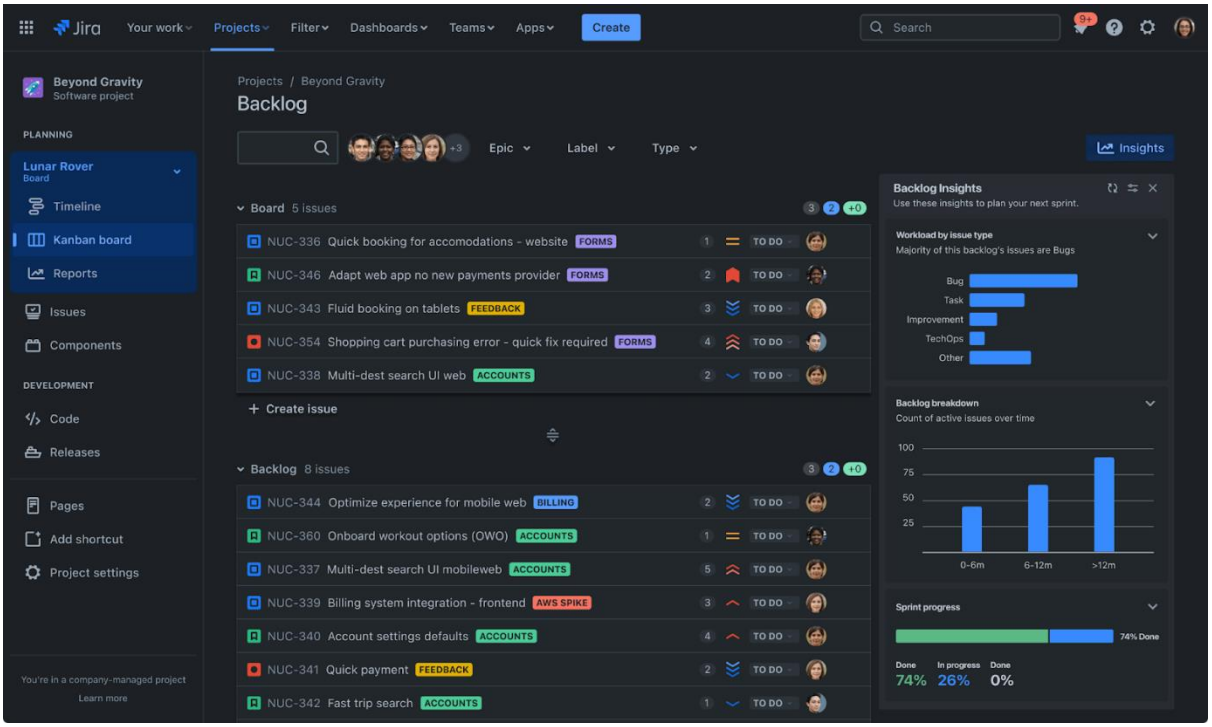

Рисунок 1.5 — Приклад інтерфейсу системи Jira

Функціонал:

1. Підтримка всіх методологій розробки (Scrum, Kanban).
2. Повний цикл від постановки задач до їх вирішення, та баг-трекінг.

3. Інтеграція з інструментами розробки (Git, Bitbucket, Jenkins).
4. Гнучке налаштування робочих процесів та доступів.
5. Розширена аналітика та звіти.

Переваги:

- Потужна система управління проектами;
- Гнучкі інтеграції з інструментами розробки;
- Масштабованість для різних команд;
- Автоматизація робочих процесів;
- Розширені можливості звітності;

Недоліки:

- Висока складність налаштування;
- Висока вартість;
- Повільний інтерфейс при значному навантаженні.

Trello

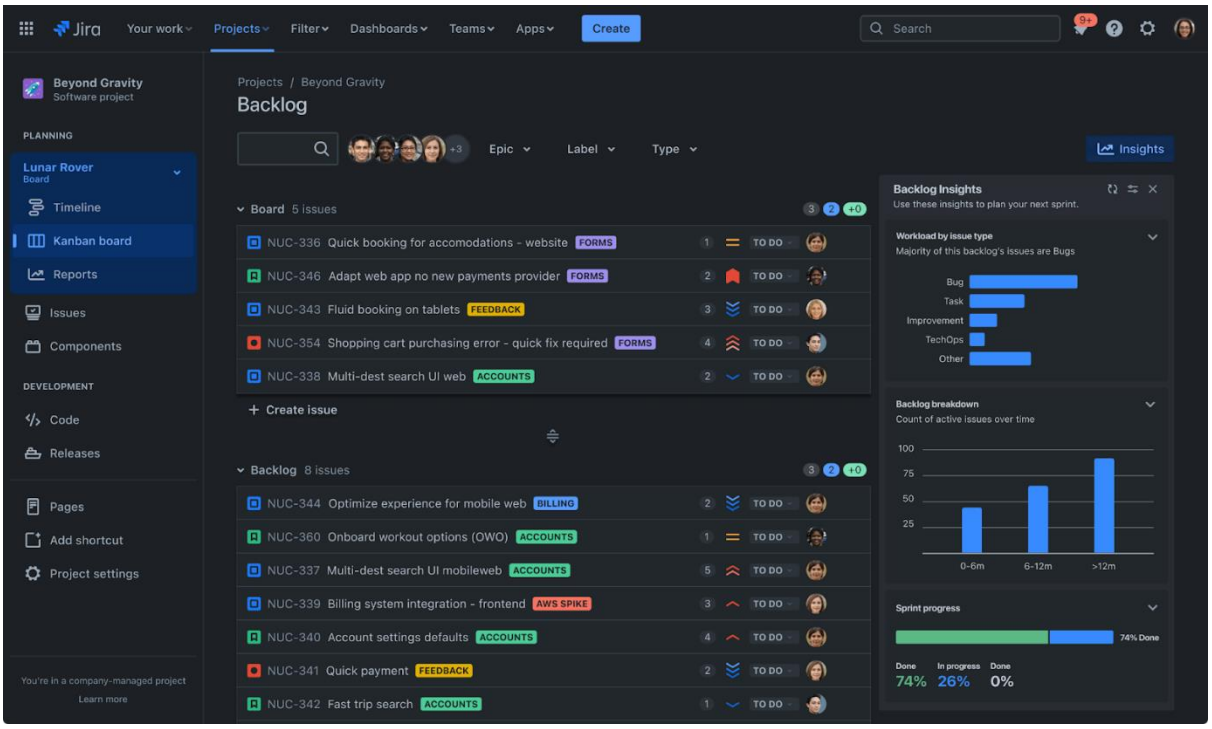

Рисунок 1.6 — Приклад інтерфейсу системи Trello

Функціонал:

1. Простий інтерфейс
2. Робота в реальному часі.
3. Інтеграція з сервісами Slack, Google Drive, GitHub.

Переваги:

- Простота використання;
- Візуалізація робочого процесу;
- Безкоштовна базова версія;
- Зручність для мобільних пристроїв.

Недоліки:

- Обмежені можливості для масштабування;
- Базова аналітика;
- Обмежена підтримка Agile методологій.

Asana

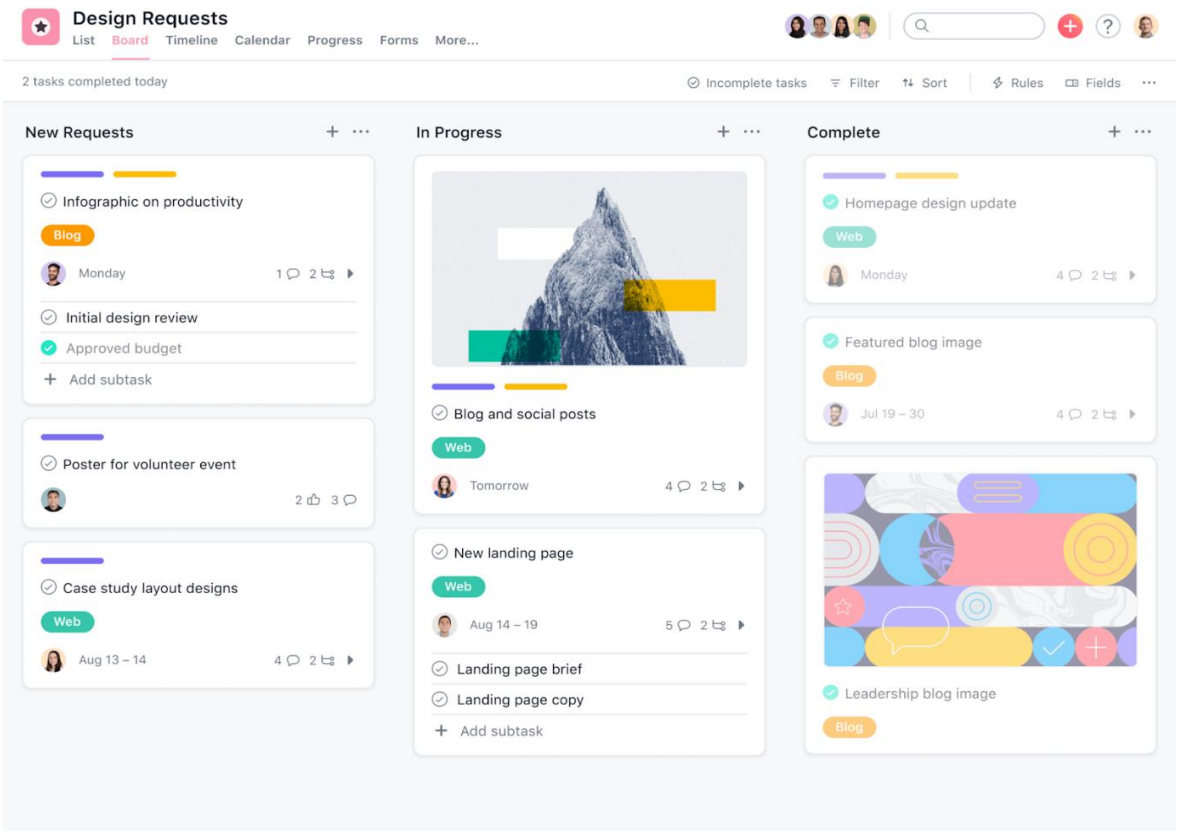

Рисунок 1.6 — Приклад інтерфейсу системи Asana

Функціонал:

1. Управління завданнями та проєктами.
2. Інтеграція з Slack, Google Drive
3. Аналітика та звітність.

Переваги:

- Інтуїтивний інтерфейс;
- Гнучке налаштування робочих процесів;
- Автоматизація рутинних завдань;
- Інтеграції з іншими інструментами;

Недоліки:

- Обмежена підтримка Agile методологій;
- Відсутність вбудованого трекінгу часу;
- Менша кастомізація, ніж у Jira.

Таблиця 1.3. Порівння існуючих систем

Параметри	**Jira**	**Trello**	**Asana**
Управління проектами (Scrum, Kanban)	+	-	+
Інтеграція з розробницькими інструментами	+	-	+
Трекінг прогресу та звіти	+	+	+
Управління командою	+	-	+
Контроль часу	+	-	-
Гнучкість налаштувань	+	-	+
Моніторинг завдань та дефектів	+	-	+

Після аналізу існуючих інструментів, які є на ринку, які було визначено як найпопулярніші, можна зробити висновок, що Jira є одним із найкращих варіантів для нашого відділу, але це тільки на перший погляд, тому що перш за все значний мінус Jira — це ціна, вона дійсно кусається.

Також Jira — це дуже велика система, і для будь-якої компанії, будь-якого відділу, який хоче працювати з цією системою, її налаштування та інтеграція в робочі процеси є великим стресом та коштує дуже дорого.

У цьому контексті вважаю доцільним розробку власної баг-трекінгової системи, яка буде адаптована під нагальні потреби компанії, тому що розроблення система дозволить:

- уникнути зайвих витрат на ліцензії;
- адаптувати всі процеси до специфіки роботи;
- інтегрувати тільки необхідний функціонал;
- закласти фундамент для розширення до великої багатофункціональної системи.

1.6. Розрахунок техніко-економічного обґрунтування впровадження створюваного програмного забезпечення

Зниження часу для уточнення всіх завдань та підвищення ефективності розробників може спричинити економію 2% від прибутку.

Ступінь новизни розроблюваних задач — "В" — використання типових проектних рішень за умови їх змін, розробка проектів, що мають аналогічні рішення. Група складності алгоритму — 2

Таблиця 1.4.Визначення виду інформації

Тип інформації	**Позначення**	**Кількість**
Кількість змінних даних	ЗД	m = 8
Кількість нормативно-довідкових даних	НДД	n = 2
Кількість баз даних	БД	P = 1
Обробка даних у режимі реального часу	РЧ	Так
Підтримка телекомунікаційної обробки для віддаленого управління	ТКО	Ні

Таблиця 1.5. Визначення витрат часу

Вид системи	Стадія розробки системи	
	Ескізний проект	Технічне завдання
	В	В
Управління задачами	53	42

Визначимо витрати часу на стадіях «технічний проект», «робочий проект» і «впровадження». Вхідними даними для визначення є:

- кількість форм вхідної інформації 8;
- кількість форм вихідної інформації 2;
- базове значення витрат часу для стадії «Технічний проект» ТБ3=90;
- базове значення витрат часу для стадії «Робочий проект» ТБ4=153;
- базове значення витрат часу для стадії «Впровадження» ТБ5=62.

Базове значення витрат часу ТБ коригується за допомогою поправочних коефіцієнтів для всіх стадій розробки автоматизованої системи.

Визначення витрат часу для стадії «Технічний проект»

Таблиця 1.6. Коефіцієнти k_1, k_2, k_3 для стадії «Технічний проект»

Вид використаної інформації	Ступінь новизни
	В
k1 (ЗІ)	1.0
k2 (НДІ)	0.72
k3 (БД)	2.08

Таблиця 1.7. Коефіцієнт ступеню новизни проекту (k_O)

Стадія розробки системи	Вид обробки	Ступінь новизни
		В
Технічний проект	РЧ	1,26
Робочий проект	РЧ	1,32
Впровадження	РЧ	1,21

$k_п$ – коефіцієнт трудомісткості робіт на стадії “технічний проект”

$T_з$ – Витрата часу на стадії «технічний проект»

$$k_п = \frac{(1 * 8 + 0.72 * 2 + 2.08 * 1)}{(8 + 2 + 1)} = \frac{11.52}{11} \approx 1{,}047$$

Визначення витрат часу для стадії «Технічного проекту»:

$$T_з = 90 * 1.047 * 1.26 = 118$$

Таблиця 1.8. Коефіцієнти k_1, k_2, k_3 для стадії «Робочий проект»

Вид використаної інформації	Ступінь новизни
	В
k1 (ЗІ)	1.2
k2 (НДІ)	0.62
k3 (БД)	0.54

$k_п$ – коефіцієнт трудомісткості робіт на стадії “ Робочий проект”

T_4 – Витрата часу на стадії «Робочий проект»

$$k_п = \frac{(1.2 * 8 + 0.65 * 2 + 0.54 * 1)}{(8 + 2 + 1)} = \frac{11.44}{11} \approx 1{,}04$$

$$T_4 = 153 * 1.04 * 1.32 * 1.08 = 226$$

Визначення витрат часу для стадії «Впровадження»

T_4 – Витрата часу на стадії «Впровадження»

$$T_5 = 62 * 1.04 * 1.21 * 1.08 = 84$$

Отже, загальні витрати людської праці складають:

$$T_{\Sigma} = 53 + 42 + 118 + 226 + 84 = 523$$

Визначимо чисельність виконавців Ч

Якщо для виконання проекту припустимо кількість робочих годин складає 500 із 8-годинним робочим днем, тому на розробку проекту виділено Ф, днів

$$Ч = \frac{500}{8} = 62 \text{ дні}$$

Для проекту Ф = 62 дні. Тоді визначаймо кількість місяців із розрахунку 20 робочих днів. Кількість місяців на розробку М $m = \frac{62}{20} = 3{,}1$ місяці

Отже, для виконання такого проекту потрібно така чисельність виконавців Ч, яка обраховується за формулою $Ч = \frac{523}{62} = 8$ днів

Приймемо розмір заробітної плати програміста - 25000 грн, тоді загальна сума заробітних плат програмістів складає:

$$V1 = 8 * 3 * 25000 = 600000$$

1.6.1 Витрати, пов'язані з розробкою програми на ПК

Дійсний річний фонд часу ПК у годинах дорівнює числу робочих годин у році для оператора, за винятком часу на технічне обслуговування і ремонт ПК (в середньому 5год/міс + 6 роб.днів/рік)

$$T_{пк} = 2000 - (6 * 8 + 5 * 12) = 1892год.$$

Оскільки під час виконання проекту витрачается в середньому витрачає 500 год. машинного часу, то величина фонду часу ПК дорівнює

$$T_{пк} = 1892 * \left(\frac{500}{2000}\right) = 473год.$$

Поточні витрати на експлуатацію V

Балансована вартість ПК, де Цр - ринкова вартість ПК, орієнтовно складає 30000 грн, кун – коефіцієнт, що враховує витрати на установку ПК . кун=0,12

$$Ц_{пк} = 30000 * (1 + 0.12) = 33600грн$$

Амортизаційні відрахування використання ПК, Зам, обчислюються за формулою

$$З_{ам} = \frac{33600}{5} = 6750грн$$

Витрати на електроенергію, споживану ПК, обчислюються

$$З_{ел} = 0.5 * 473 * 4.32 * 0.9 = 919 \text{ грн}$$

Витрати на поточний ремонт і технічне обслуговування ПК (визначаються як 6% від балансової вартості ПК,

$$З_{р} = 33600 * 0.06 = 2016 \text{ грн}$$

Непрямі витрати, пов'язані з експлуатацією ПК, визначаються як 5% від балансової вартості ПК .

$$З_{мат} = 33600 * 0.05 = 1680 \text{ грн}$$

Поточні витрати на експлуатацію V

Заробітна плата обслуговуючого персоналу складає в середньому - 10000 Тож, поточні витрати на експлуатацію

$$V1 = 10000 + 6750 + 919 + 2016 + 1680 = 21365 \text{ грн.}$$

А, загальні витрати на розробку програмного забезпечення комп'ютерної системи складуть

$$V1 = 600000 + 21365 = 621365$$

1.6.2 Розрахунок витрат на придбання і установку

ПК V2 = Цпк = 30000 грн

Витрати на підготовку приміщення V3 = 0, так як приміщення є в наявності. Витрати на навчання персоналу V4. В середньому навчання персоналу триватиме 1 місяць, тому можна вважати, що V4 = 4500 грн;

$$V_{\Sigma} = 621365 + 30000 \ + 0 + 4500 = 655865$$

Оскільки норма амортизаційних втрат для комп'ютерних систем НА = 5, то для обрахування річного економічного ефекту слід брати до розгляду величину

$$V_p = \frac{655865}{5} = 131173$$

За рахунок збільшення виконувати більше прибуток складатиме 100 000 грн на рік

$$К_{еф} = \frac{100000}{131173} = 0{,}76$$

$$Т_{ок} = \frac{1}{0{,}37} = 1{,}31 \text{ років}$$

1.7. Обґрунтування доцільності проєктування й розроблення (ProTrack)

Як ми можемо бачити у вищезазначених пунктах (пп. 1.3, 1.4.2, 1.5.), ми отримали значення: у 1.31 року окупності, що є дуже гарним показником. Але це не єдине. Головне, що буквально з першого дня розробники, менеджери дійсно відчують полегшення у роботі над своїми завданнями та роботі в цілому.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

Розроблення веб-додатку моніторингу виконання завдань компанії з розробки програмного забезпечення "Rain"

2.1. Загальні відомості

- **Назва проекту:** Розробка інформаційної системи управління проектами та завданнями «**ProjectRain**»
- **Замовник:** Компанія «Rain».
- **Розробник:** Компанія «Danil and DeveolpersCry».
- **Відповідальні особи:**
  - Замовник: Тосиро Мифунє, директор компанії «Rain».
  - Розробник: Харченко Даніл
- Контакти: Toshiro. Mifune@toho.jp, DanilKharchenko@gmail.com
- Дата початку проекту: 2024-10-10.
- Термін виконання: 4 місяці (до 2025-02-10).

2.2. Призначення і цілі створення системи.

Система (ProjectRain) перш за все призначена для автоматизації процесів управління проєктами. Система автоматизує створення та відстеження наших завдань на проєктах, які внесені в цю систему.

2.2.1. Призначення системи "ProjectRain".

Також вона має функціонал призначення виконавців, контроль термінів, генерацію звітів та формування. Також система містить детальні дані про проєкти, завдання, залучених користувачів, які працюють над завданнями або на якихось проєктах, що надає, так би мовити, комплекс додаткових функцій, які можуть бути дуже зручними для аналізу та в подальшому побудови ефективної роботи всього (IT-відділу)

2.2.2. Цілі створення системи "ProjectRain".

Основною метою створення системи "ProjectRain". є забезпечення повний вичерпної інформації щодо проєктів щодо завдань та виконавців цих завдань.[7]

2.3. Характеристика об’єкта автоматизації.

Об'єкта автоматизації в контексті нашої системи, це може бути будь-яка команда розробників, яка є в компанії, де потрібно мати автоматизовану систему баг-трекінгу для забезпечення максимальної ефективності роботи цієї команди.

2.4. Вимоги до системи

Система (ProjectRrain) розроблено як десктопний додаток з складною багатошаровою архітектурою

2.4.1. Архітектура:

Вона використовує (фреймворк Spring Boot) для реалізації серверної логіки, включаючи бізнес-логіку та безпеку, що забезпечення для забезпечення несанкціонованого доступу до даних. Графічний інтерфейс користувача побудований на базі технології (JavaFX)[8, 9]. А база даних використання єдиної бази даних для зберігання всієї інформації про проєкти, завдання, користувачів, ролі, вкладення та інші дані. Для використовування нашої системи вся база даних покладена на (PostgreSQL)[25, 26, 30]. Взаємодія з цією базою відбувається за допомогою бібліотеки (Spring Data JPA) з (Hibernate)[14, 15, 19] у якості провайдера, що забезпечення об'єкта реляційне відображення та управління транзакції. Також є розподілення на ролі в моїй системі. Перш за це роль адміністратора. Вона володіє повним правами системи, відповідає за конфігурацію нашої системи, може створювати нових користувачів, редагувати користувачів, активувати, деактивувати користувачів та може призначати ролі іншим користувачам. Далі, користувач — це стандартна роль системи, яка виступає в ролі менеджера. Вона може створювати проєкти та завдання, взяти виконавців або одного виконавця та працювати з завданнями, розмовляти їх статуси і фіксувати витрачений час.

2.4.2. Діагностування функціонування:

Діагностика стану системи та виявлення відхилень від нормального функціонування. Забезпечується через надійний механізм логування. Використовується бібліотека (SLF4J) з реалізацією (Logback) для запису деяких інформаційних повідомлень що можуть виникати під час роботи системи для подальших їх аналізу.

Реалізований рівень деталізації логів, які можна налаштувати через (application.properties файл). Це може бути корисно, якщо ми хочемо відстежити хід виконання нашої програми або однієї операції та проаналізувати можливі причини збоїв.

Для користувача передбачено виведення інформаційних повідомлень про помилки в результатах операцій через стандартні вікна в (JavaFX Alert), що забезпечує нагальний зворотний зв'язок та допомагає користувачу зрозуміти, що він зробив не так.

Спеціалізований клас (GlobalExceptionHandler), який є основною частиною, яка реалізує централізовану обробку виключень на рівні (JavaFX), що забезпечує загальну стабільність роботи серверної частини та може запобігти неочікуваному завершення роботи програми.

Також взаємозв'язок між логічним компонентами системи, такими як контролер (JavaFX), сервіси (Spring), репозиторії (Spring Data JPA), вони побудовані на принципах інверсії управління та впровадження залежності, що забезпечується фреймворком (Spring), що додає більшої надійності в нашій програмі.

2.4.3. Розвиток і модернізація системи:

Найважливіше для подальшого розвитку — це структура програмного коду, оскільки з цим кодом треба працювати. Структура кода чітко відповідає принципам об'єктно-орієнтованого програмування та загальноприйнятим

шаблоном проєктування, наприклад (MVC) для (JavaFX) частини, що полегшує розуміння,

Під час проєктування системи було закладено прочний фундамент для подальшої розвиток та модернізації, оскільки архітектура системи, заснована на (Spring Boot), вже по замовченню означає чітке розподільність між шарами, такими як: бізнес-логіка, сервіси, доступ до даних, репозиторії, що забезпечує найлегшим чином модифікування існуючих класів та впровадження нових без стресу для всієї системи.

2.4.4. Функціонування системи:

Система забезпечує інтуїтивно зрозумілу роботу над всіми основними операціями завдяки інтуїтивно зрозуміло графічному інтерфейсу користувача. Персонал, який використовує автоматизовану систему «ProjectRain», повинен дотримуватися наступних вимог:

- Перш за все — це пройти ознайомчий курс з можливостями системи відповідно до своєї ролі;
- Дотримуватись технічних інструкцій та рекомендацій при роботі з системою;
- Мати базові навички з роботою в баг-трекінгових системах.

2.4.5. Користувачами системи «ProjectRain» можуть виступати:

Вхід у систему реалізовано через діалогове вікно “авторизація”, яке аутентифікує користувачів за допомогою унікального логіна та пароля.

- Адміністратори системи відповідають за загальне налаштування системи управління всіми користувачами та створення нових користувачів;
- Розробники, тестувальники — безпосередньо виконавці завдань у проєкті;
- Менеджери проєктів, які можуть відслідковувати ефективність виконуваних задач та скільки було втрачено часу на виконання задачі.

Реалізована система є гнучкою щодо типів проєктів та завдань, якими можна управляти. Також, наприклад, адміністратор може керувати списком користувачів та їх ролями. Управління проєктами та завданнями, перш за все, використовується через додавання нових типів статусів завдань, пріоритетів чи якихось інших атрибутів шляхом модифікації деяких файлів, наприклад, (Enum: TaskStatus, TaskPriority, TaskVisibility).

2.4.6. Обсяг та обмеження проекту

- Система повинна підтримувати до 500 одночасних користувачів.
- Підтримка мовного інтерфейсу: українська, англійська Обмеження за часом відгуку: завантаження основних сторінок та операції з завданнями — не більше 2 секунд при типовому навантаженні.
- Система повинна забезпечувати надійне зберігання даних та можливість резервного копіювання.
- Обмеження: На початковому етапі не передбачається глибока кастомізація workflow завдань користувачами (використовуються стандартні або налаштовані адміністратором). Детальні інтеграції з зовнішніми системами розглядаються як окремі завдання.

2.4.7 Ключові об'єкти системи

Об'єкт "Завдання" (Task)

Цей об'єкт є основним елементом системи і міститиме такі поля:

- Унікальний ідентифікатор - автоматично генероване число для однозначної ідентифікації завдання;
- Назва - коротке формулювання суті завдання
- Опис (description) - детальний опис завдання з можливістю форматування тексту та додавання файлів;
- Термін виконання (deadline) - дата та час, до якого завдання має бути виконане;

- Пріоритет - ступінь важливості завдання (низький, середній, високий, критичний);
- Статус - поточний стан завдання (створено, в роботі, на перевірці, виконано, відкладено);
- Виконавець - співробітник, відповідальний за виконання завдання
- Автор - співробітник, який створив завдання;
- Коментарі - можливість обговорення завдання між учасниками проєкту.

Об'єкт "Користувач" (User)

Представлятиме співробітника компанії з такими характеристиками:

- Ідентифікатор користувача;
- Ім'я та прізвище;
- Посада;
- Контактна інформація (електронна пошта, телефон);
- Роль в системі - визначає рівень доступу та можливості користувача;
- Список призначених завдань;
- Список створених завдань;
- Статистика активності;
- Об'єкт "Роль" (Role), Визначатиме рівень доступу користувача до функцій системи:
- Адміністратор - повний доступ до всіх функцій системи, включаючи управління користувачами;
- Менеджер проєкту - можливість створювати проєкти, призначати завдання, переглядати звіти.

Об'єкт "Проєкт" (Project)

Об'єднуватиме завдання, пов'язані з конкретним будівельним об'єктом:

- Назва проєкту;
- Опис проєкту;
- Дата початку та завершення;

- Команда проєкту - список користувачів, залучених до проєкту;
- Список завдань проєкту;
- Статус проєкту;
- Звіти та метрики - аналітична інформація про хід виконання проєкту;
- Об'єкт "Звіт" (Report).

Забезпечуватиме аналітичні дані для керівництва:

- Тип звіту (загальний прогрес, продуктивність команди, статистика за часом);
- Параметри фільтрації;
- Можливість експорту у різні формати (PDF, Excel).

РОЗДІЛ 3. ПРОЄКТУВАННЯ, СТВОРЕННЯ ТА АПРОБАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Основна мова програмування, яка була вибрана на проєкті, — це (Java). Цей вибір дуже легко обґрунтувати, оскільки її на 99% об'єктно-орієнтована природа за замовчуванням є дуже потужним інструментом в руках розробника [3, 4, 5, 6].

3.1. Опис та обґрунтування вибору програмно-технічних засобів розроблення програмного продукту

З використанням всіх технік, (SOLID) [1, 2, 23, 22, 11] та об'єктно-орієнтоване програмування, код, написаний на (Java), є дуже легкомасштабованим та підтримуваним. Також у (Java) є дуже-дуже багато бібліотек, фреймворків та вже готових реалізованих рішень та інструментів завдяки високому рівню підтримки з боку спільноти розробників.

Використання актуальної версії, такої як (Java 21), дає простір використовувати найсучасніші можливості мови, наприклад, покращення роботи з колекціями, віртуальні потоки для підвищення продуктивності роботи з великою кількістю операцій в один момент часу.

Ще один плюс (Java) — це її кросплатформність, що означає, вона дуже гарно працює зі всіма операційними системами, на які вона буде використовуватись: (Windows), (Mac OS), (Linux) — однаково гарно.

Другим за важливістю прийнятим програмно-технічним рішенням — це використання певного фреймворку, а саме (Spring Boot). Я обрав як основний фреймворк для побудови серверної логіки та загальної архітектури мого додатку. Який дуже спрощує розробку (Spring)-додатків завдяки автоконфігурації та вбудованому управлінню залежностями. Це стає можливим завдяки потужному інструменту, який дозволяє створювати слабозв'язані, модульні компоненти, що дуже полегшує розробку, тестування [10, 24, 28, 29] та підтримку нашого коду. У

проєкті (Project Rain) дуже активно використовується (DI) для впровадження сервісів, репозиторіїв та інших компонентів, які взаємодіють між собою. Управління транзакціями, надає дуже потужні декларативні засоби через анотації (@Transactional), що є дуже важливими для забезпечення цілісності даних, та атомарностю при роботі з базою даних.

3.1.2. Технології для роботи з даними.

Основна та єдина база даних, яка використовується на продукті, — це реляційна база (PostgreSQL SQL)[25, 26]. Вона є потужною, надійною об'єктно-реляційною базою даних з відкритим вхідним кодом, що означає, що вона є цілком безкоштовною. А також вона підтримує широкий спектр (SQL)-стандартів, розширені типи даних та важливі (ACID) транзакції.

Але база даних — це про зберігання даних, ще треба ці дані якось парсити в (Java)-об'єкти. І з цим допомагає мені (Spring Data JPA) [16, 17, 18, 20, 21] з використанням (Hibernate) у якості провайдера.

Чому саме це рішення? Оскільки це спрощує розробку шару доступу даних, надаючи деяку абстракцію над (JPA), це означає, що ми можемо, створивши деякі об'єкти в (Java), поставивити анотації, такі як (Entity), і ми вже маємо змогу використовувати ці сутності як об'єкти в (Java) та мати над ними базові (CRUD)-операції: читання, редагування, оновлення та видалення.

3.1.3. Технології для розробки графічного інтерфейсу користувача (GUI).

Для створення сучасного інтерфейсу користувача я використовував (JavaFX). Чому саме це рішення? Оскільки це є сучасна платформа для розробки клієнтських додатків на (Java). Система надає вичерпний функціонал кастомізації за допомогою (CSS)[25], , підтримку (FXML) та потужні можливості для роботи з медіа.

3.1.4. Забезпечення безпеки.

Оскільки система, що розробляється, є дуже тендітною в парадигмі безпеки, треба використати дуже серйозний інструмент для забезпечення цієї безпеки, і для цього ідеально підходить (Spring Security).

Це є дуже потужним та гнучким фреймворком для забезпечення безпеки на (Java). Він надає дуже багато комплексних рішень, але було використано:

- Перш за все — це перевірка наших облікових даних користувача: логіна та пароля через (Authentication Manager). Всі паролі зберігаються в базі в хешованому вигляді завдяки використанню (BCryptPasswordEncoder), що є стандартом індустрії для безпечного зберігання паролів;
- Також (Spring) надає захисту поширених атак. Він надає вбудований механізм засоби, наприклад, від (CSRF)-атак за замовчуванням.

3.1.5. Збірка проекту та управління залежностями.

В якості системи оптимізації збірки та управління залежностями проєкту використовується (Apache Maven). (Maven)[12] є стандартом де-факто у (Java)-розробці. Він дозволяє декларативно описувати нашу структуру проєкту, через файл (pom.xml), плагіни, процеси збірки, що забезпечує легке управління версіями бібліотек.

3.1.6. Допоміжні бібліотеки.

Lombok: є дуже корисним інструментом, оскільки анотації, такі як: (@Getter),(@Setter),(@RequiredArgsConstructor), (@Slf4j), (@NoArgsConstructor)[13], автоматично генерують відповідні методи (гетери, сетери, конструктори, логери), що робить код більш читабельним та менш громіздким. (SLF4J) (Simple Logging Facade for Java) є дуже корисним інструментом в системі виховання, оскільки це забезпечує гнучке та потужне логування, що є дійсно важливим, коли потрібно відслідкувати проблему, щоб правильно її виправити.

3.2. Проєктування та створення бази даних.

Опис структури БД «Система Керування Проектами "ProjectRain"» [31].

Проєктування баз даних для системи. Здійснювалося з опром на зберігання конкретної інформації про проєкти, завдання, користувачів, їх ролі, команди розробників, та файлові вкладення. Нижче наведено опис з основних таблиць, який використовується в системі.

3.2.1. Таблиця users

Одна з головних сутностей — це таблиця User (Користувачі). Вона призначена для зберігання інформації про облікові записи користувачів системи:

- user_id (BIGINT, Primary Key, Auto Increment): Унікальний ідентифікатор користувача;
- username (VARCHAR(100), Not Null, Unique): Унікальне ім'я користувача (логін);
- password (VARCHAR(255), Not Null): Хешований пароль користувача.
- enabled (BOOLEAN, Not Null, Default true): Статус активності облікового запису (активний/неактивний)

3.2.2. Таблиця roles

Одна з допоміжних таблиць — “Roles” (Ролі). Призначено для того, щоб зберігати ролі в системі для подальшого їх використання.

- role_id (INTEGER, Primary Key, Auto Increment): Унікальний ідентифікатор ролі;
- name (VARCHAR(50), Not Null, Unique): Унікальна назва ролі (наприклад, "ROLE_ADMIN", "ROLE_USER").

3.2.3. Таблиця user_roles

Оскільки в сутностях "user" та "roles", ми використовуємо зв'язок багато до багатьох, нам потрібна допоміжна таблиця, якою і виступає таблиця "user_roles" (Зв'язок Користувачі-Ролі).

- user_id (BIGINT, Foreign Key -> users.user_id): Ідентифікатор користувача;
- role_id (INTEGER, Foreign Key -> roles.role_id): Ідентифікатор ролі

3.2.4. Таблиця Project

Таблиця "Project" (Проекти) призначена для зберігання даних про проєкти.

- project_id (BIGINT, Primary Key, Auto Increment): Унікальний ідентифікатор проекту;
- name (VARCHAR(255), Nullable): Назва проекту;
- description (TEXT, Nullable): Детальний опис проекту;
- status (VARCHAR(50), Not Null): Поточний статус проекту;
- start_date (DATE, Not Null): Дата початку проекту;
- end_date (DATE, Nullable): Планова або фактична дата завершення проекту;
- command_id (BIGINT, Foreign Key -> Command_Developer.command_id, Nullable): Ідентифікатор команди розробників, призначеної на проект

3.2.5. Таблиця tasks

Таблиця "Task" (завдання) призначена для зберігання даних про завдання.

- task_id (BIGINT, Primary Key, Auto Increment): Унікальний ідентифікатор завдання;
- title (VARCHAR(255), Not Null): Назва завдання;
- description (TEXT, Nullable): Детальний опис завдання;
- status (VARCHAR(20), Not Null, Default 'TO_DO'): Статус виконання завдання;

- priority (VARCHAR(20), Not Null, Default 'MEDIUM'): Пріоритет завдання;
- estimated_time (DECIMAL(5,1), Nullable): Оціночний час на виконання завдання;
- logged_time (DECIMAL(5,1), Nullable): Фактично витрачений час на завдання;
- creation_date (TIMESTAMP, Not Null, Updatable=false): Дата та час створення завдання;
- due_date (DATE, Nullable): Термін виконання завдання;
- project_id (BIGINT, Foreign Key -> Project.project_id, Not Null): Ідентифікатор проекту, до якого належить завдання;
- assignee_user_id (BIGINT, Foreign Key -> users.user_id, Nullable): Ідентифікатор користувача-виконавця завдання;
- reporter_user_id (BIGINT, Foreign Key -> users.user_id, Not Null): Ідентифікатор користувача-автора завдання;
- visibility (VARCHAR(50), Not Null, Default 'ALL_USERS'): Рівень видимості завдання;

3.2.6. Таблиця attachments

Таблиця “attachments” (Вкладення) призначена для зберігання інформації про фотографії, які можуть бути вкалдені до нашого завдання.

- attachment_id (BIGINT, Primary Key, Auto Increment): Унікальний ідентифікатор вкладення;
- file_name (VARCHAR(255), Not Null): Оригінальна назва файлу;
- file_path (VARCHAR(255), Not Null): Унікальний шлях до збереженого файлу на сервері;
- file_type (VARCHAR(100), Nullable): MIME-тип файлу;
- upload_date (TIMESTAMP, Not Null, Updatable=false): Дата та час завантаження файлу;

- task_id (BIGINT, Foreign Key -> tasks.task_id, Not Null): Ідентифікатор завдання, до якого прикріплено файл.

3.2.7. Таблиця Command_Developer

Таблиця " Command_Developer " (команди розробників) призначена для зберігання інформації про команди.

- command_id (BIGINT, Primary Key, Auto Increment): Унікальний ідентифікатор команди;
- comand_type (VARCHAR(100), Not Null): Тип або назва команди;

3.3 Реалізація функцій системи.

Всі функції систем розроблення за допомогою Java FX

3.3.1. Аутентифікація та авторизація користувачів

Функціонал допомагає забезпечити безпечний вхід користувача у систему та авторизацію.

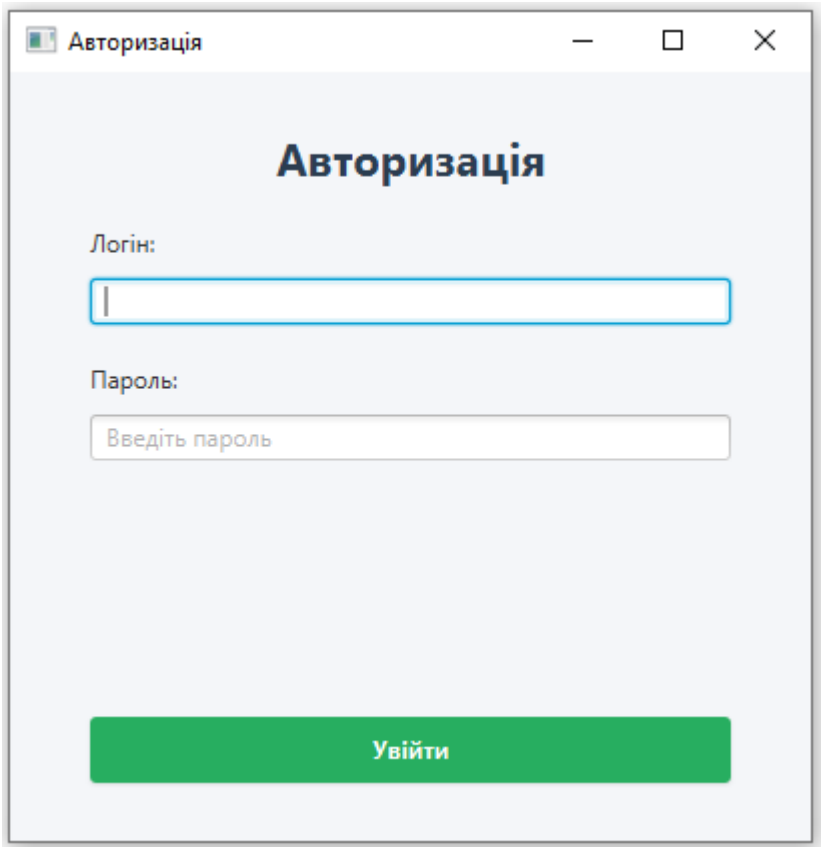

Рисунок 3.1— Форма авторизації

Вікно входу в систему реалізовано у файлі loginPage.fxml та управляється контролером LoginController. Форма вмістить собі два поля: одного для введення імені користувача. А інше для пароля та кнопку входу. Також є область для відображення помилок(Рис. 3.1).

LoginController.handleLogin(): При натисканні кнопки "Увійти", дані з полів зчитуються. Створюється UsernamePasswordAuthenticationToken.

Для перевірки облікових даних використовується AuthenticationManager з Spring Security, який впроваджено в LoginController. AuthenticationManager відправляє перевірку на кдасс UserDetailsService, реалізованому в UserService.java. Метод loadUserByUsername() завантажує користувача з бази даних та його ролі. (Паролі порівнюються з використанням BCryptPasswordEncoder, визначеного в SecurityConfig.java.)

У разі успішної аутентифікації, об'єкт Authentication зберігається в SecurityContextHolder.getContext().setAuthentication(authentication);. Цей метод робить так, що інформація про користувача зберігається по всій системі. А конкретно це потрібно для його ролей доступу. Після успішного входу закривається вікно логіну.

SQL-запити:

Пошук користувача за іменем: SELECT username FROM users WHERE username = ? Завантаження ролей користувача: SELECT ... FROM user_roles ur JOIN roles r ON ur.role_id = r.role_id WHERE ur.user_id = ?

Рядок підключення до БД: Визначається в application.properties через змінні середовища .env, ${DB_URL}, ${DB_USER}, ${DB_PASSWORD}.

3.3.2. Управління користувачами (функціонал Адміністратора)

Як для ролі адміністратора, в неї є можливості створювати та редагувати, переглядати та видаляти облікові записи звичайних користувачів. А також призначати їм ролі.

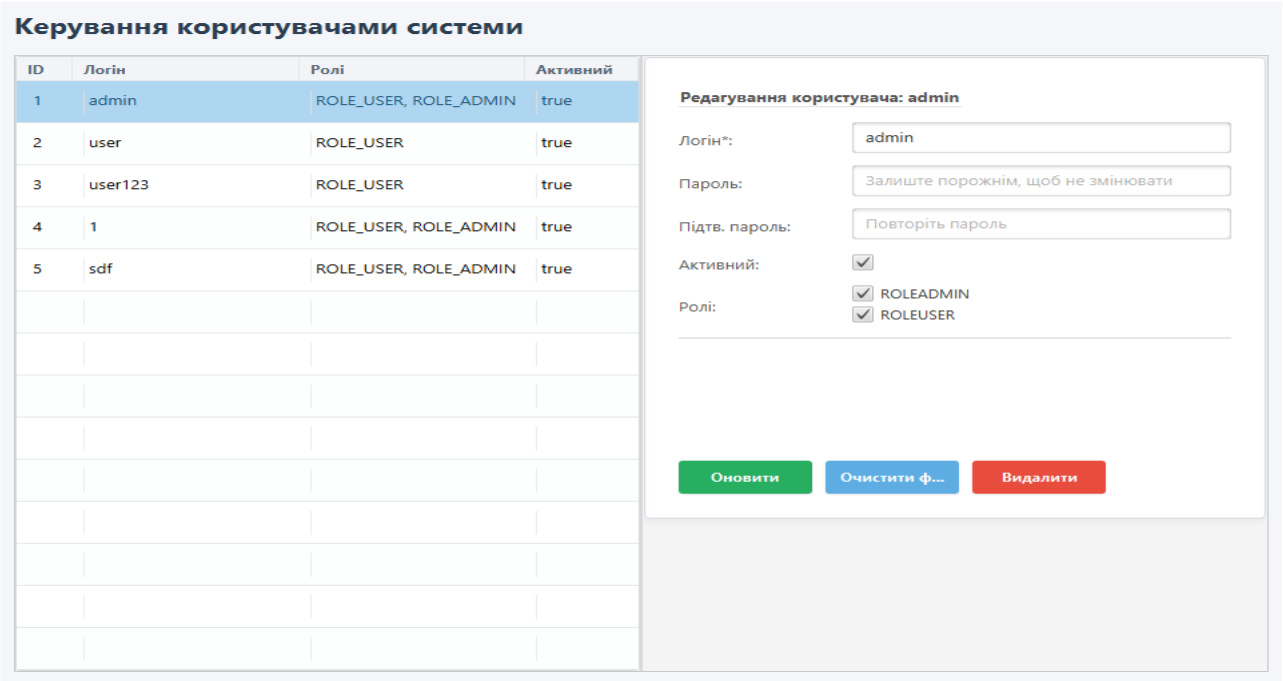

Рисунок 3.2— Форма управління користувачами

Реалізовано у файлі userManagementPage.fxml та управляється контролером UserManagementControllerFx.java. Інтерфейс має таблицю (userTableView) для відображення користувачів (ID, ім'я, ролі, статус активності) та форму для введення/редагування даних користувача(Рис. 3.2)

UserManagementControllerFx.initialize(): Завантажує список доступних ролей (userService.getAllRoles()) та список всіх користувачів що відображается в таблиці (userService.getAllUsers()). Якщо selectedUser == null, створюється новий користувач. Викликається userService.createUser(username, password, selectedRoleNames). Пароль хешується в сервісі. Якщо selectedUser != null, оновлюється існуючий користувач. Викликається userService.updateUser(userId, newUsername, selectedRoleNames). Якщо введено новий пароль, викликається userService.changePassword(). UserManagementControllerFx.handleDeleteUser(): Після підтвердження видаляє вибраного користувача через userService.deleteUser(userId).

SQL-запити:

Отримання всіх користувачів: SELECT * FROM users, отримання всіх ролей: SELECT * FROM roles, збереження/оновлення користувача: INSERT INTO users ... / UPDATE users SET ... WHERE user_id = ?.

3.3.3. Управління проектами

Один з основних функціоналів програми — це управління проєктами. Тут ми можемо створювати проєкти(Рис. 3.3).

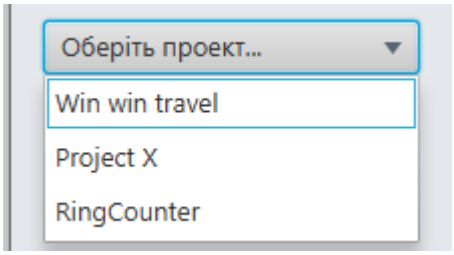

Рисунок 3.3— Вибір проекта

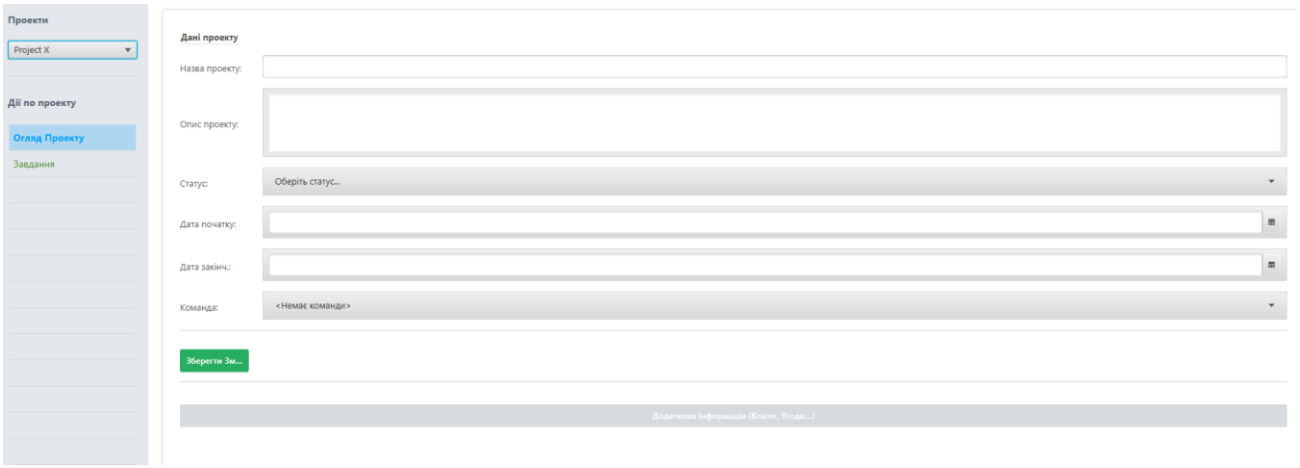

Рисунок 3.4— Редагування проекта

В головному вікні присутній ComboBox<Project> projectSelectorComboBox для вибору активного проекту. Також у файлі projectPage.fxml файл управляється контролером ProjectController Та містить поля для назви проекту (projectNameField), опису (projectDescriptionArea), вибору статусу (statusComboBox), дати початку/закінчення (startDatePicker, endDatePicker),

вибору команди (commandComboBox) та кнопку збереження (saveButton)(рисунок 3.4).

ProjectRainController.loadAvailableProjects(): Завантажує список всіх проектів через projectService.getAllProjects() та заповнює projectSelectorComboBox. ProjectRainController.projectSelected(): При виборі проекту в ComboBox, цей проект стає поточним (currentProject). Завантажуються дії, доступні для проекту (Огляд, Завдання) в projectActionListView. За замовчуванням може завантажуватися "Огляд Проекту".

ProjectRainController.loadView(): Завантажує FXML для огляду проекту (projectPage.fxml) в центральну область mainContentArea.

ProjectController.setProjectData(): Метод в ProjectController для отримання об'єкта Project та заповнення полів форми його даними.

ProjectController.handleSaveButton(): Збирає дані з форми та викликає projectService.updateProject(...) для збереження змін існуючого проекту. (Створення нового проекту, ймовірно, відбувається через інший механізм або діалогове вікно, яке не деталізовано в наданих файлах ProjectController, але метод projectService.createProject існує).

Додаткова інформація про проект (ProjectInformationController.java) завантажується окремо і пов'язана з основним проектом через projectId.

Використані елементи GUI: ComboBox, ListView, StackPane, TextField, TextArea, DatePicker, Button.

SQL-запити:

Отримання всіх проектів: SELECT * FROM Project (через projectRepository.findAll()).

Оновлення проекту: UPDATE Project SET ... WHERE project_id = ?.

Отримання команд розробників: SELECT * FROM Command_Developer (через commandDeveloperRepository.findAll()).

3.3.4. Управління Завданнями

У користувачів та адміністраторів є можливість створювати, редагувати завдання, які є ключовим об'єктом системи.

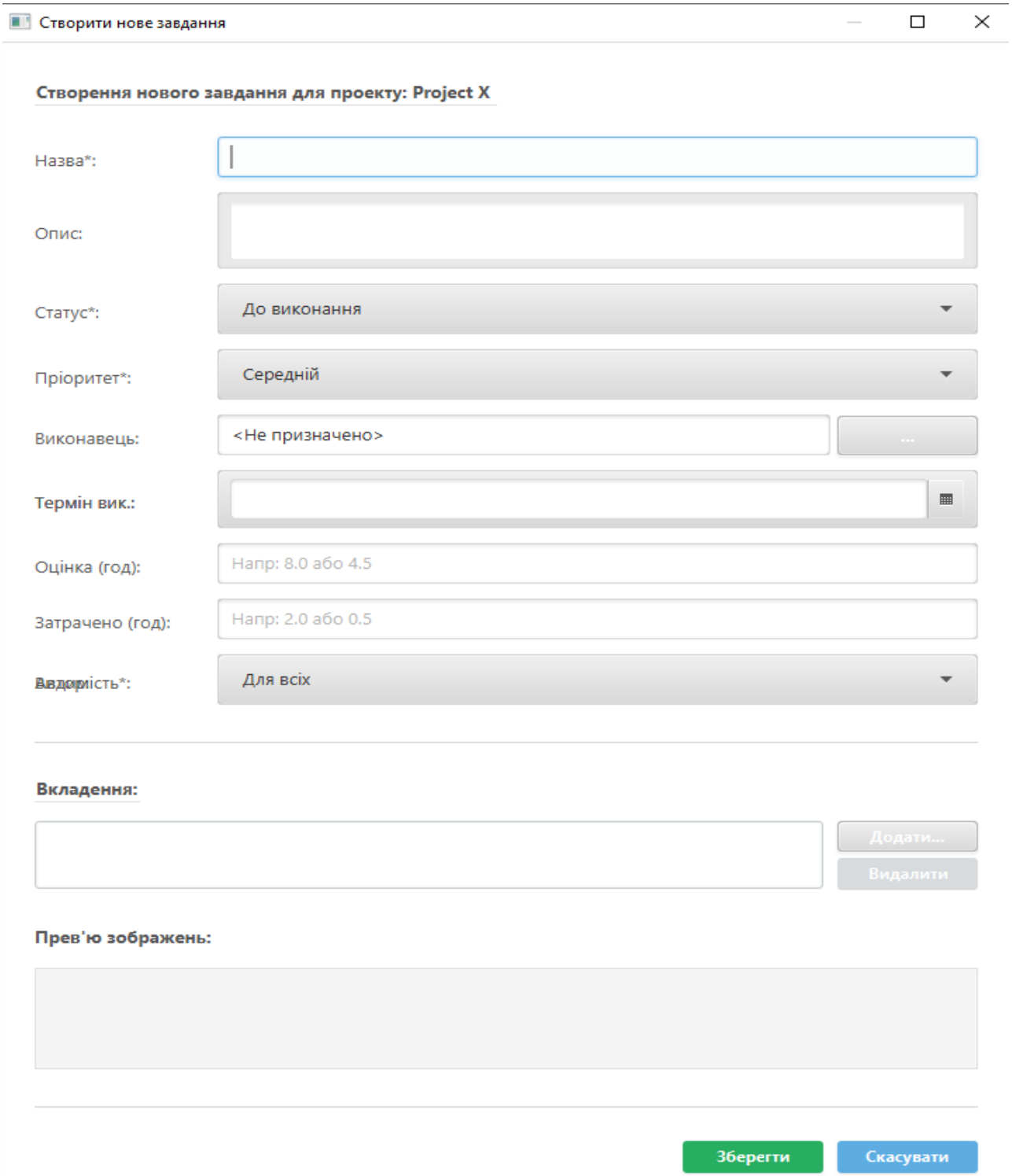

Рисунок 3.5— Інтерфейс для створення завдання

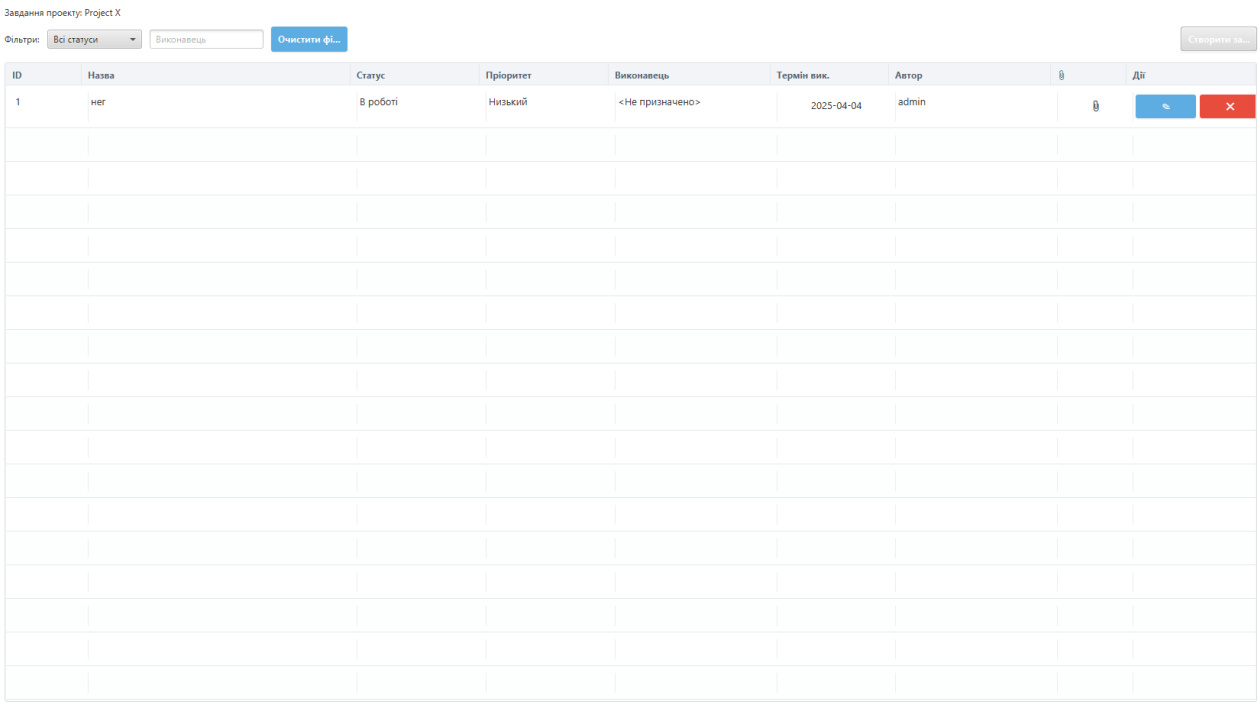

Рисунок 3.6— Інтерфейс беклогу

Реалізовано у файлі taskPage.fxml та управляється контролером TaskControllerFx. Інтфейс містить таблицю (taskTableView) для відображення завдань поточного проекту з колонками: ID, Назва, Статус, Пріоритет, Виконавець, Термін виконання, Автор, Вкладення, Дії.(Рис. 3.6.)

Надає елементи для фільтрації завдань: ComboBox для статусу (filterTaskStatusComboBox) та текстове поле для фільтрації по виконавцю (filterAssigneeField) (Рис. 3.5).

Кнопка "Створити завдання" (createTaskButton) для відкриття діалогу створення нового завдання. Також реалізовано у файлі taskDialog.fxml та управляється контролером TaskDialogController. Також форма містить поля для всіх атрибутів завдання: Назва (titleField), Опис (descriptionArea), Статус (statusComboBox), Пріоритет (priorityComboBox), Виконавець (assigneeDisplayField з кнопкою вибору selectAssigneeButton), Термін виконання (dueDateDatePicker), Оцінка часу (estimatedTimeField), Затрачений час

(loggedTimeField), Автор (reporterLabel – відображається), Видимість (visibilityComboBox).

TaskControllerFx.loadTasks(): Завантажує завдання для вибраного проекту (currentProject) за допомогою taskService.findTasksByProjectId(). Результат відображається в taskTableView. Враховує права доступу (адмін бачить усі, інші – з видимістю ALL_USERS).

TaskControllerFx.handleCreateTask():Створює нове завдання. Отримує користувача (getCurrentSystemUser()) для встановлення як автора та викликає openTaskDialog() для відображення вікна.

TaskControllerFx.handleEditTask(): Завантажує повні дані завдання через taskService.findTaskById() та передає їх у openTaskDialog().

TaskDialogController.setDialogData(): Заповнює поля форми даними завдання (при редагуванні) або встановлює значення за замовчуванням та автора (при створенні нового).

TaskDialogController.handleSaveTask(): Обробляє дані з форми, проводить валідацію (isInputValid()).

Для нового завдання: передає об'єкт Task та ID автора (currentSystemUser.getUserId()) в taskService.createTask(Task task, Long reporterId).

Для існуючого завдання: викликає taskService.updateTask(taskId, taskDetails).

TaskDialogController.handleSelectAssignee(): Відкриває діалог вибору виконавця TaskDialogController.handleAddAttachment(), handleRemoveAttachment(): Керують додаванням та видаленням файлів-вкладень через AttachmentService.java.

Всі операції з базою даних для завдань та вкладень інкапсульовані в TaskService.java та AttachmentService.java.

Фонові операції виконуються за допомогою кастомного методу executeTaskSimplified (в TaskDialogController) або executeTask (в TaskControllerFx), щоб не блокувати UI.

SQL-запити:

Створення завдання: INSERT INTO tasks (title, description, status, priority, reporter_user_id, project_id, ...) VALUES (?, ?, ?, ?, ?, ?, ...)

Оновлення завдання: UPDATE tasks SET title = ?, ... WHERE task_id = ?

Отримання завдань по проекту: SELECT ... FROM tasks t LEFT JOIN FETCH t.assignee LEFT JOIN FETCH t.reporter ... WHERE t.project_id = ?

3.4. Інструкція користувача

При вході в програму вас зустрічає вікно авторизації.

3.4.5 Авторизація

Потрібно ввести свій логін та пароль, який вам повинен надати адміністратор вашої системи. Якщо його немає, ви можете запросити в нього. Він вам залюбки його надасть(Рис. 3.7).

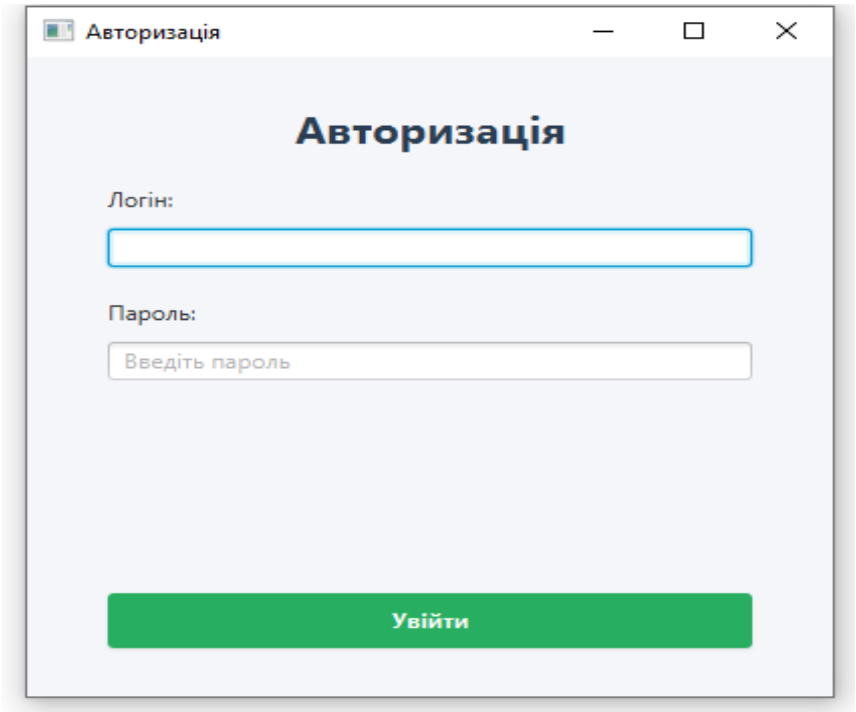

Рисунок 3.7— Вікно аворизації

3.4.6 Вибір проекта

Після успішної авторизації (як авторизуватися, наведено в пункті 3.4.5 авторизація.) вас зустрічає головне вікно програми, де ви можете обрати свій проєкт. Який вас більше всього цікавить. Це ви можете зробити на бічній панелі у випадаючому списку “Оберіть проєкт”(рисунок 3.8).

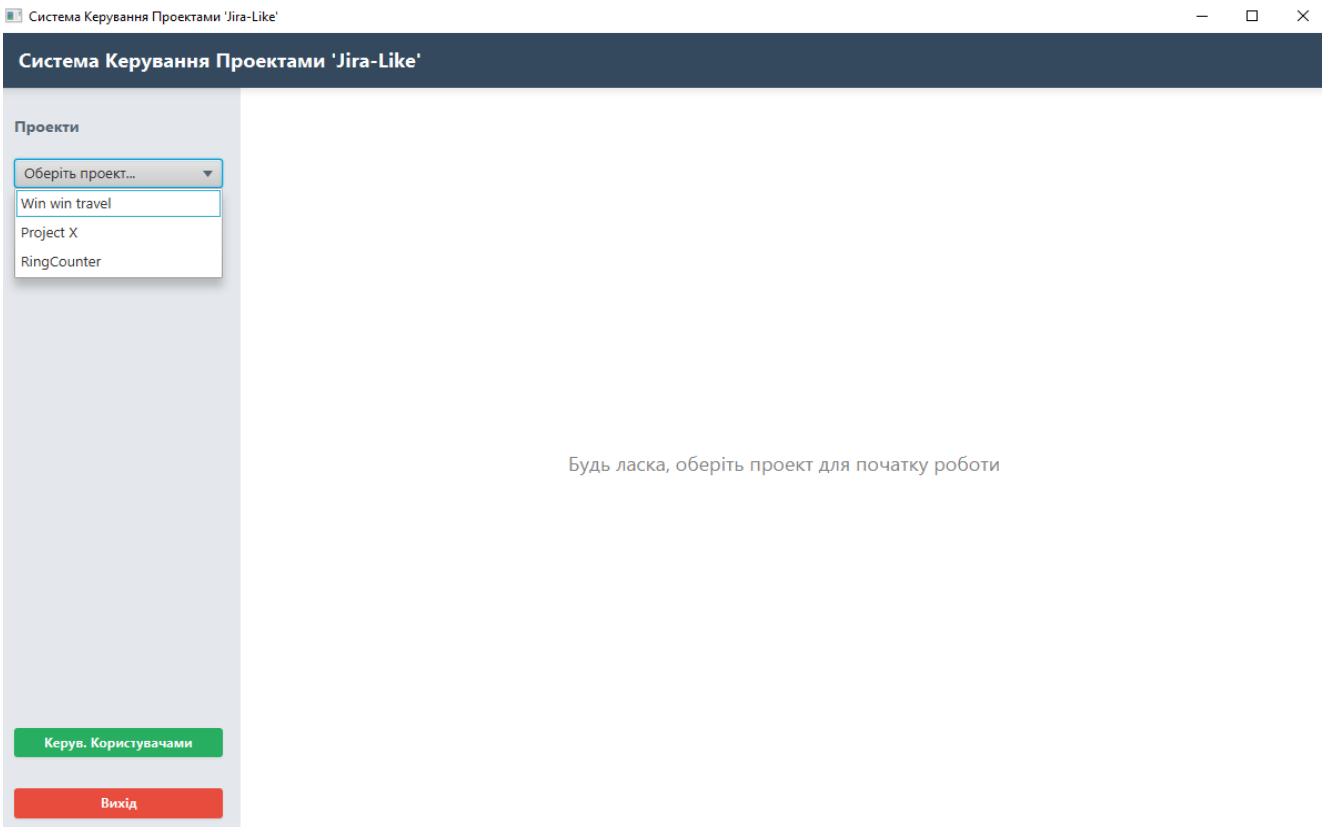

Рисунок 3.8— Вибір проєкту

3.4.7 Керування проектом

Після успішної авторизації та вибору проекта (див. Пункт 3.4.5, 3.4.6), вас зустрічає вікно редагування даних про проєкт. Тут ви можете змінити дані про проект: його назву, його опис, його статус, його дати початку та дату закінчення. Також ви можете зазначити команду, яка буде працювати над цим проєктом. Якщо ви внесли всі зміни, які вам потрібні. Треба тиснути на кнопку "Зберегти зміни", і всі ці зміни автоматично збережуться в базі даних.

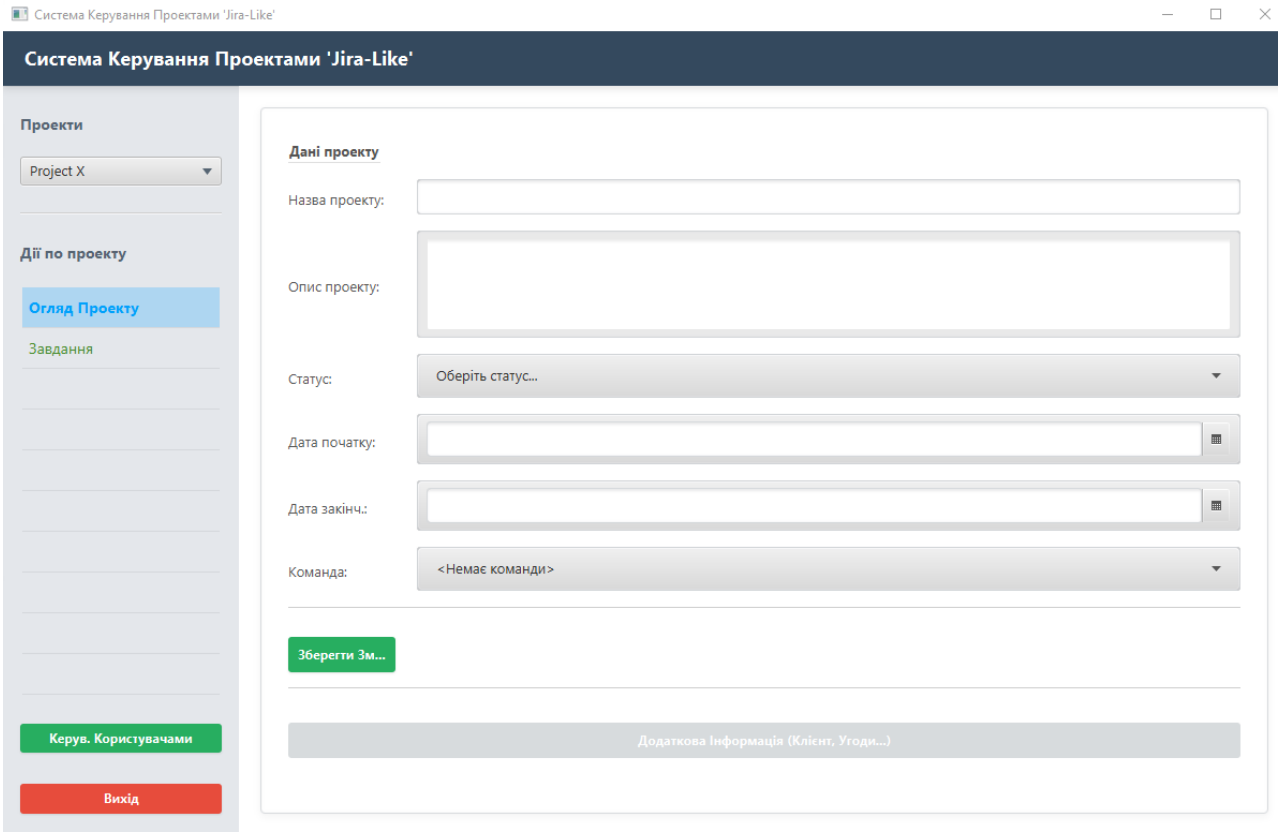

Рисунок 3.9— Керування проектом

3.4.7 Створення проекту

Після успішної авторизації та вибору проєкта (див. Пункт 3.4.5, 3.4.6), якщо ви маєте права адміністратора та авторизовані під цими правами, ви можете створити новий проєкт. На вкладці "Дані проєкту", треба ввести його назву, опис, статус, дату початку, дату закінчення та команду. Натиснути кнопку "Зберегти зміни". Але варто зауважити, що для цього вас у бічному вікні "Проєкти" у випадаючому списку, не повинен міститися ніякий проєкт, оскільки ви будете змінювати вже створений проєкт, а не створювати новий (Рис. 3.9).

3.4.8 Робота з беклогом задач

Після успішної авторизації та вибору проєкта (див. Пункт 3.4.5, 3.4.6), відображається беклог, де ви можете побачити всі завдання, які є на проєкті. Але якщо ви "user", ви не бачите всіх завдань. Якщо ви "admin", ви бачите всі завдання. Вгорі ви можете побачити фільтрування завдань по проєкту. Є два фільтри. 1 — це фільтр за статусом, де ви можете відкрити випадаючий список

та вибрати статус, який вас більше всього цікавить. Та 2 — виконавець, де ви можете ввести ім'я виконавця. І побачити у головному вікні всі його завдрання. Також ви можете натиснути на кнопку "Очистити фільтр". І буде очищено фільтр (Рис. 3.10).

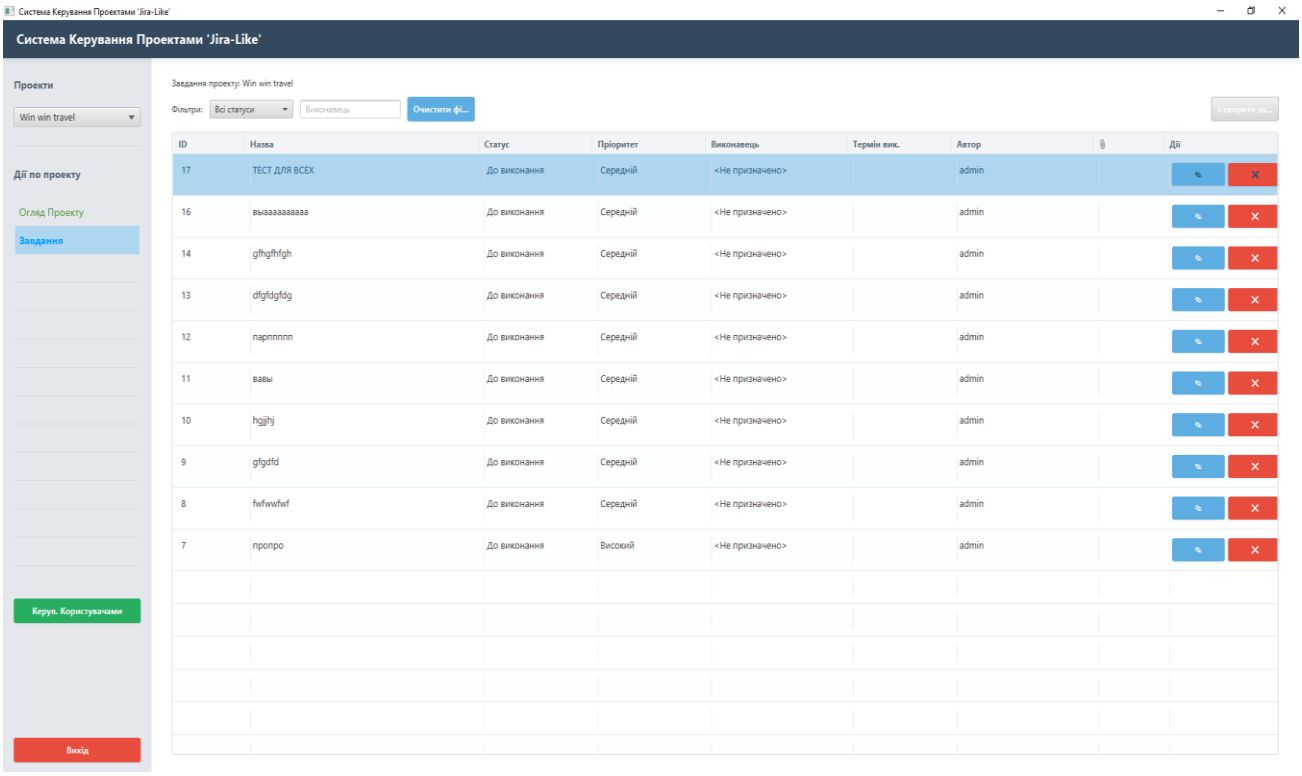

Рисунок 3.10— Приклад “Беклогу “

3.4.9 Редагування/Видалення задач

Після успішної авторизації та вибору проєкта (див. Пункт 3.4.5, 3.4.6) та відкриття беклогу (див Пункт 3.4.8) У звичайного "user" є можливість редагувати тільки свої завдання, та видаляти тільки свої завдання. У "Admin", є можливість редагувати та видаляти будь-які завдання, які наявні у беклозі. Для редагування завдання вам потрібно в колонці "Дії" обрати кнопку "Редагувати". Після натискання на кнопку вам відкриється вікно редагування завдання, де ви можете його відредагувати. А саме поля: назва, опис, статус, пріоритет. Зазначити іншого виконавця, зробити інший термін виконання. Або змінити оцінки часу на більше або менше. Для видалення завдання потрібно обрати кнопку "Видалити". Підтвердити видалення. І, зазначу, буде видалено з бек-логу (Рис. 3.11, 3.12).

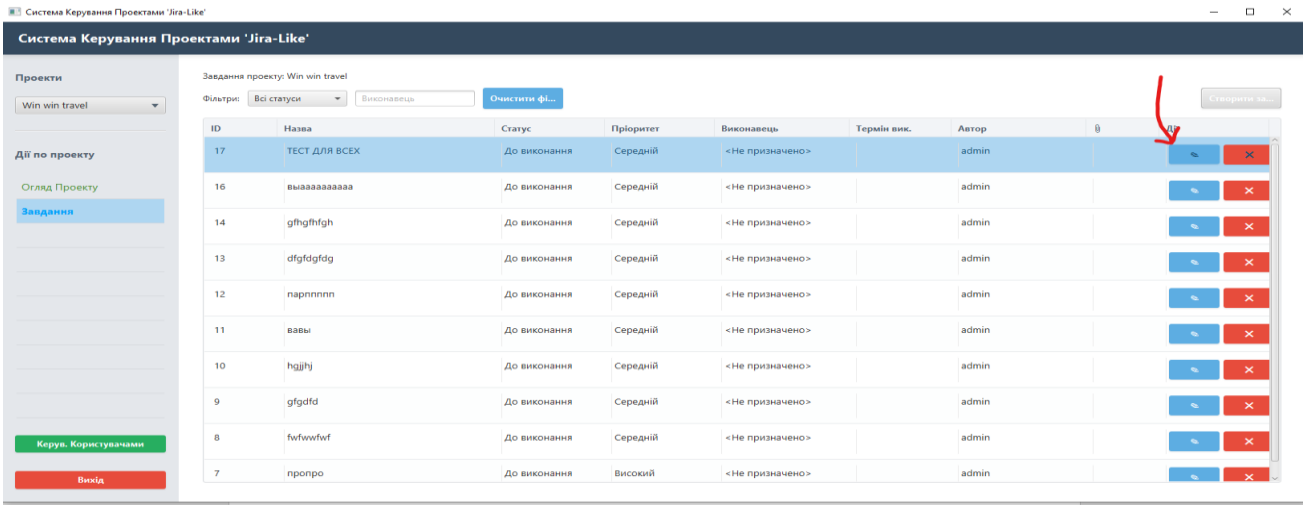

Рисунок 3.11— Кнопка редагування завдання

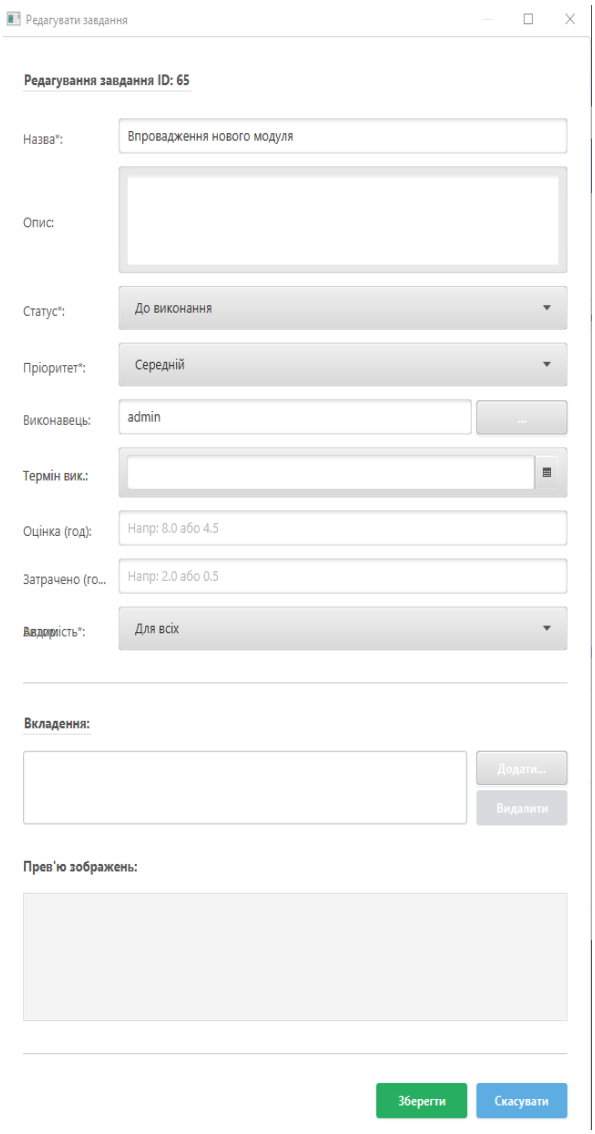

Рисунок 3.12— Редагування завдання

3.4.10 Створення нової задачи

Як у "user", так і у "admin" є можливість створити нове завдання у обраному проєкті (див. Пункт 3.4.5, 3.4.6 відповідно) та відкриття беклогу (див Пункт 3.4.8). Для створення нового завдання потрібно натиснути на кнопку "Створити завдання", Ви можете створити нове завдання. З такими полями: назва, опис, статус, пріоритет. Ви можете назаначити виконавця, (дивіться пункт 3.4.11), термін виконання. Оцінку скільки часу затрачено, можна заповнити потім після створення завдання. Також ви можете обрати видимість: для всіх або тільки для адміністраторів, далі, ви можете додати будь-які вкладення, наприклад, фото, де ви можете у формі прев'ю подивитися, що конкретно ви вклали в завдання, щоб не було помилок (Рис. 3.13).

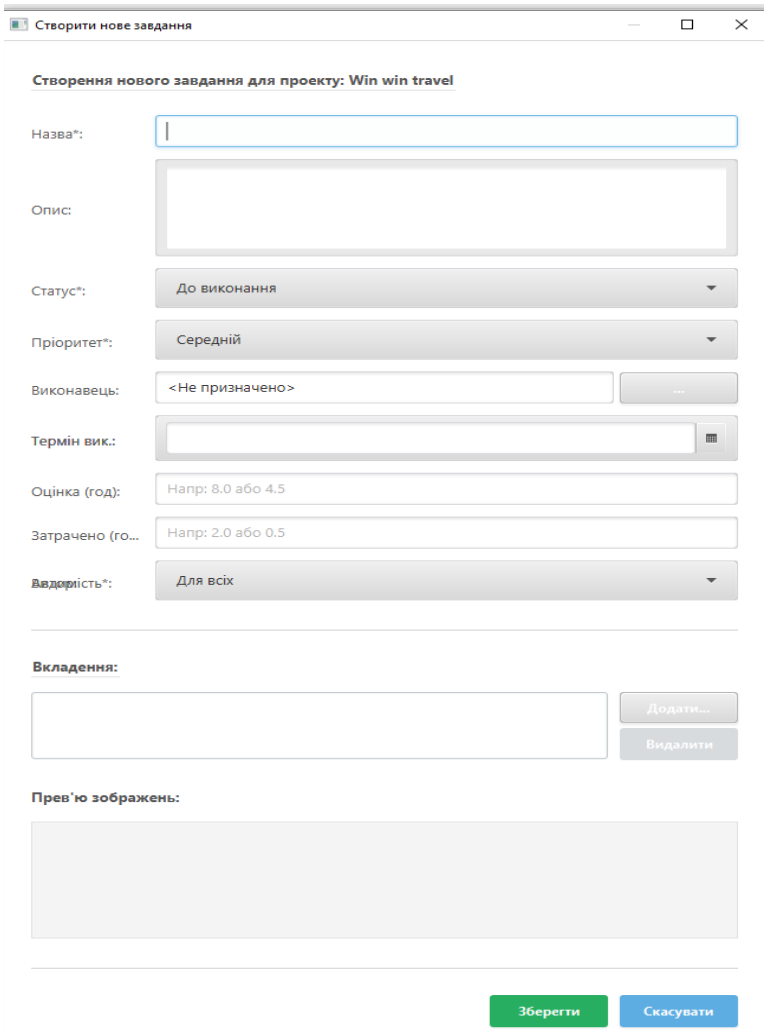

Рисунок 3.13— Форма створення нового завдання

3.4.11 Призначення виконавця

При створенні або редагуванні завдання, ви можете призначити виконавця в цьому полі

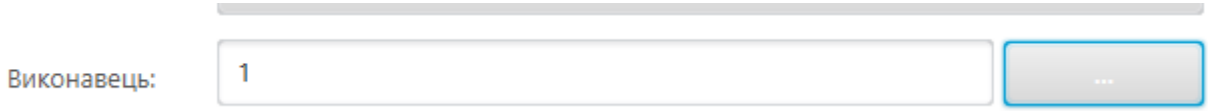

Рисунок 3.14— Форма призначення викноавця

Натиснувши на кнопку "Призначити виконавця", для більшого розуміння ви можете прочитати пункти (3.4.10, 3.4.9**,** 3.4.5, 3.4.6). Після кліка у вас відкриється вікно призначення виконавця (Рис. 3.15). У цьому вікні можна скористатися фільтром пошуку виконавців. Ви можете ввести ім'я або частку імені і обрати потрібного вам виконавця після його обрання та кліку на нього. Вам треба натиснути кнопку "Призначити", і виконавець буде автоматично призначений на це завдання (Рис. 3.14).

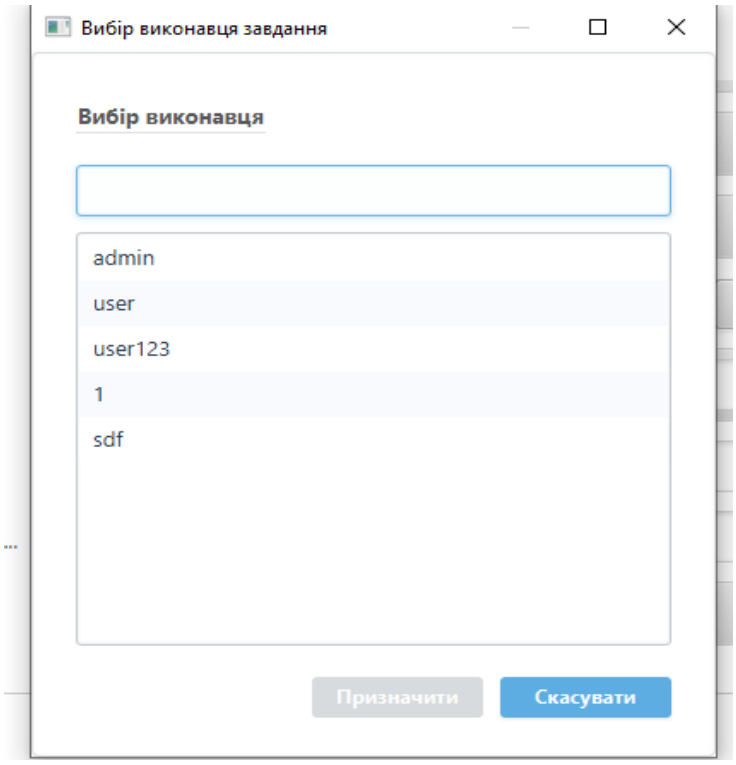

Рисунок 3.15— Вікно призаначення виконавця

3.4.12 Керування користувачами

Цей функціонал доступний лише "admin", коли ви авторизуєтесь (дивіться пункт 3.4.5), під Роль "Admin". У вас на головному вікні буде кнопка "Керування користувачами" (Рис. 3.16), після кліку на кнопку відкриется вікно

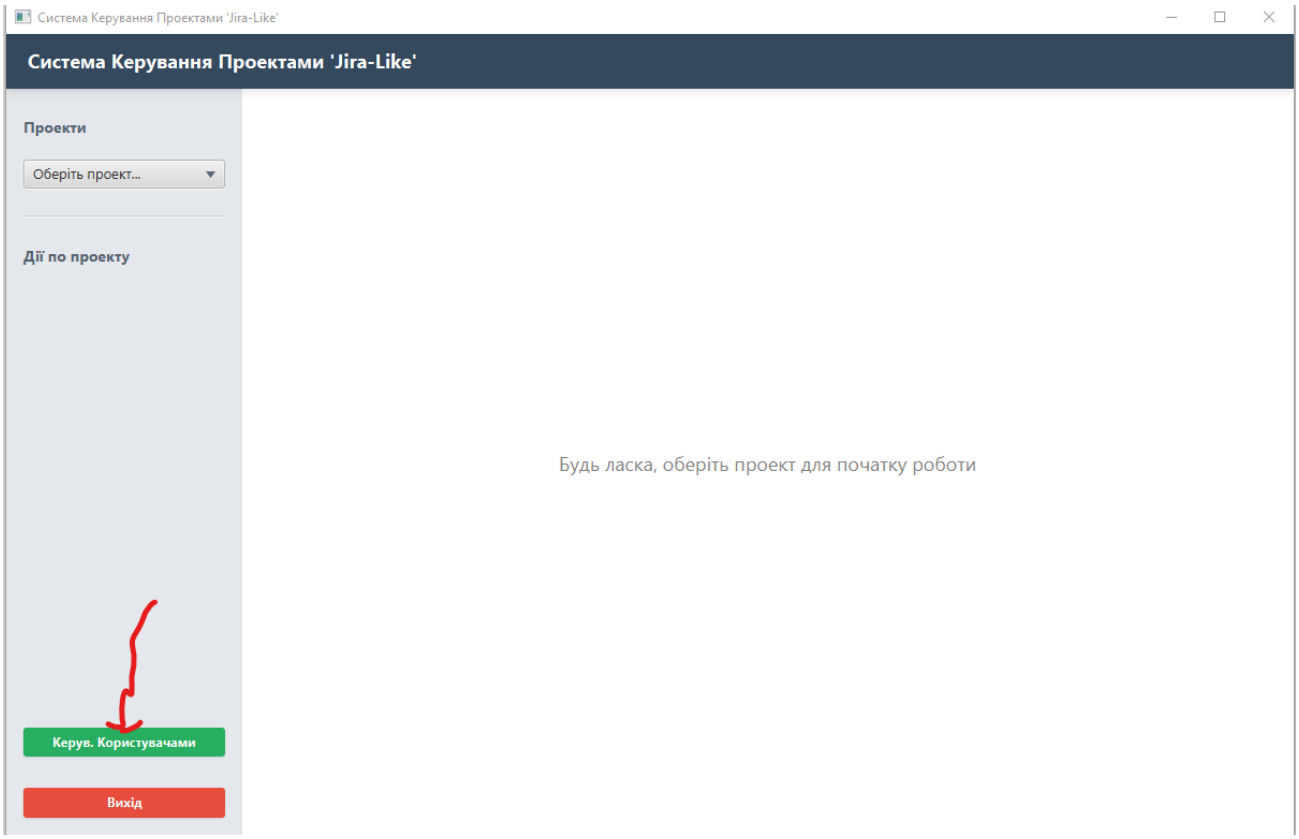

Рисунок 3.16— Кнопка для керування користувачами

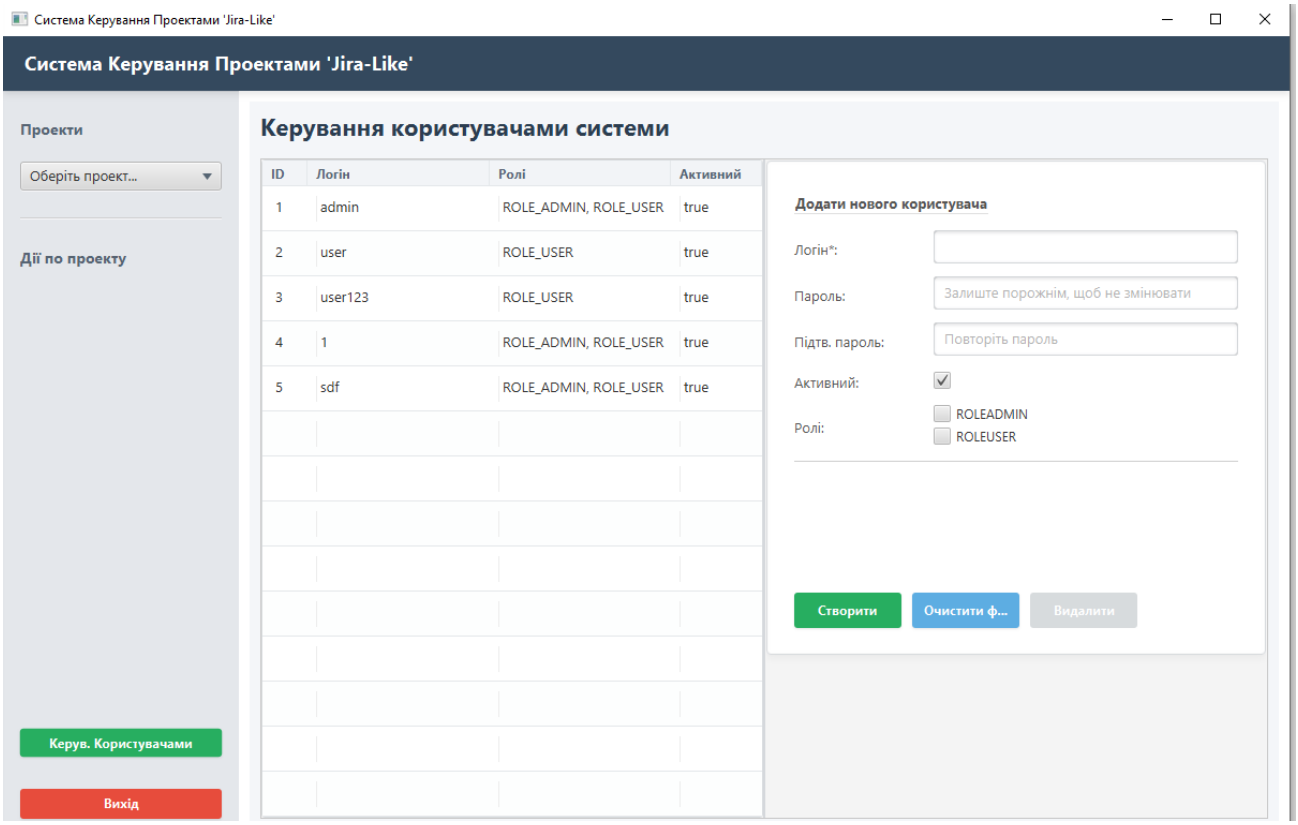

Рисунок 3.17— Вікно призначення виконавця

У цьому вікні ви можете побачити всіх користувачів, які є в системі: їх роль, логін і пароль. Та статус: активний чи деактивний. Справа в частині вікна. Є поля, які ми можемо заповнювати для створення нового користувача. Вам не потрібно обирати ніякого старого, а просто заповнити нові поля. Даними логін пароль підтвердження пароля, щоб вони збігалися. Його Активність(так/ні). Та його роль: "admin" чи "user". Можна вибрати дві ролі одночас. Якщо ви хочете редагувати якогось користувача, ви можете зліва з таблиці вибрати користувача та у формі редагування редагувати всі його дані, поставити його активним чи неактивним та забрати або підвищити роль (Рис. 3.17).

3.5 Тестування програмного продукту.

Мета тестування програмного засобу:

- Перш за все є перевірка відповідності реалізованого функціоналу, який був описаний у вимогах та його реальному імплементуванню;

- Виявявлення та документування більшості дефектів, які наявні в програмному забезпеченні;
- Оцінка надійнійності системи;
- Перевірка зручність інтерфейсу користувача;
- Перевірка механізма взаємодії з базами даних ;
- Перевірка механізма безпеки ідентифікація, авторизація.

3.5.1. Загальні положення Тест-Плану

Об'єкти тестування:

- Модуль аутентифікації та авторизації;
- Модуль управління користувачами (CRUD операції);
- Модуль управління проектами (CRUD операції);
- Модуль управління завданнями (CRUD операції);
- Інтерфейс користувача (UI) та взаємодія з ним (UX);
- База даних та взаємодія з нею через сервісний шар та репозиторії;
- Програмний продукт в цілому «Система Керування Проектами "ProjectRain"».

Рівні тестування:

- Модульне тестування: тестування окремих методів та деяких класів;
- Інтеграційне тестування: тестування взаємодії класів між собою;
- Системне тестування: тестування всього продукту на відповідність функціональним та нефункціональним вимогам.

Види тестування:

- Функціональне тестування: перевірка коректності роботи основних функцій системи;
- Тестування графічного інтерфейсу: перевірка коректності відображення (UI) та зручності використання;
- Тестування безпеки: перевірка механізмів аутентифікації та хешування;

- Тестування надійності і стабільності: тестування при тривалій роботі та обробці якихось помилок.

Критерії початку тестування:

- Завершення розробки відповідного модуля;
- Наявність тестових сценаріїв, даних;
- Наявність середовища для тестування.

Критерії завершення тестування:

- Успішне проходження всіх запланованих тестів;
- Відсутність критичних та блокуючих помилок.

3.5.2. Модульне тестування (Unit Tests)

Мета: Тестування окремих методів для виявлення слабких місць.

Об'єкти: Методи класів UserService.java, TaskService.java, ProjectService.java, AttachmentService.java

Інструменти: JUnit 5, Mockito.

Тест для методу UserService.createUser():

Назва тесту: Створення нового користувача з валідними даними;

Вхідні дані: username = "testuser", rawPassword = "password123", roleNames = {"ROLE_USER"}.

- Дії:
- Замокати (mock) UserRepository та RoleRepository, викликатиuserService.createUser("testuser","password123", Set.of("ROLE_USER")).
- Очікуваний результат:
- Метод повертає об'єкт User;
- user.getUsername() дорівнює "testuser";
- passwordEncoder.matches("password123"; user.getPassword()) повертає true;

- user.isEnabled() дорівнює true;
- user.getRoles() містить одну роль з ім'ям "ROLE_USER";
- Метод userRepository.save() був викликаний один раз.

Тест для TaskService.createTask(Task task, Long reporterId):

Назва тесту: Створення нового завдання з валідними даними.

Вхідні дані: Об'єкт Task з заповненими полями (title, project), reporterId = 1L.

- Дії:
- Замокати TaskRepository, ProjectRepository, UserRepository;
- Налаштувати when(projectRepository.findById(anyLong())).thenReturn(Optional.of(new Project()));
- Налаштувати when(userRepository.findById(1L)).thenReturn(Optional.of(new User("reporterUser", "pass") {{ setUserId(1L); }}));.
- Налаштувати when(taskRepository.save(any(Task.class))).thenAnswer(invocation -> invocation.getArgument(0));.
- Створити об'єкт Task taskToCreate, встановити йому проект;
- Викликати taskService.createTask(taskToCreate, 1L).
- Очікуваний результат:
  - Метод повертає об'єкт Task;
  - returnedTask.getReporter() не null і returnedTask.getReporter().getUserId() дорівнює 1L;
  - returnedTask.getProject() не null;
  - taskRepository.save() був викликаний.

3.5.3. Інтеграційне тестування (Integration Tests)

Мета: Перевірка взаємодії між класами в системі

Об'єкти: Ланцюжки викликів: Controller -> Service -> Repository -> Database.

Інструменти: Spring Boot Test (@SpringBootTest), Testcontainers (для запуску реального PostgreSQL в Docker).

Приклад тесту для UserService (аутентифікація та створення користувача):

Назва тесту: Повна перевірка створення користувача та його аутентифікації.

- Дії:
- Запустити тест в контексті Spring Boot;
- Використовуючи UserService, створити нового користувача (userService.createUser("integrationUser", "testPass", Set.of("ROLE_USER")));
- Спробувати аутентифікувати цього користувача через AuthenticationManager з тими ж обліковими даними;
- Перевірити отриманий об'єкт Authentication.
- Очікуваний результат:
- Користувач успішно створюється та зберігається в базі даних;
- Аутентифікація проходить успішно;
- Об'єкт Authentication містить коректне ім'я користувача та роль ROLE_USER.
- Додаткові перевірки: Спроба створити дублікат користувача (очікується помилка).

Приклад тесту для TaskService (створення завдання):

Назва тесту: Створення завдання з усіма залежностями.

- Дії:
- Запустити тест в контексті Spring Boot;
- Створити (або отримати існуючого) користувача-автора (User reporter) та проект (Project project) в базі даних;

- Створити новий об'єкт Task, встановити йому project;
- Викликати taskService.createTask(task, reporter.getUserId());
- Завантажити збережене завдання з бази даних за його ID.
- Очікуваний результат:
- Завдання успішно збережено;
- Завантажене завдання містить коректні дані, включаючи reporter_id, що відповідає ID переданого автора, та project_id.

3.5.4. Системне тестування (End-to-End GUI Testing)

Мета: Перевірка роботи всієї системи через графічний інтерфейс користувача, емулюючи дії реального користувача.

Об'єкти: Усі UI сценарії, визначені функціональними вимогами.

Інструменти: TestFX (для автоматизації тестів JavaFX GUI).

Приклад сценарію: "Створення нового завдання користувачем"

Назва тесту: Успішне створення завдання через UI.

- Передумови:
- Додаток запущено;
- Користувач з роллю ROLE_USER (або ROLE_ADMIN) успішно залогінився;
- Існує принаймні один проект, до якого користувач має доступ.
- Кроки:
- Запустити додаток "ProjectRain";
- Ввести логін та пароль користувача (наприклад, "user", "userpass" з DataInitializer.java). Натиснути "Увійти";
- У головному вікні вибрати існуючий проект з projectSelectorComboBox;
- У списку дій по проекту вибрати "Завдання" (для переходу до TaskControllerFx);
- Натиснути кнопку "Створити завдання" (createTaskButton);

- У діалоговому вікні TaskDialogController заповнити обов'язкові поля: "Назва", "Статус", "Пріоритет", "Видимість";
- (Опціонально) Заповнити "Опис", вибрати "Виконавця", "Термін виконання", додати вкладення;
- Натиснути кнопку "Зберегти".
- Очікуваний результат:
- Діалогове вікно створення завдання закривається;
- Нове завдання з'являється у таблиці завдань (taskTableView в TaskControllerFx) з коректно відображеними даними;
- При перевірці в базі даних, для цього завдання коректно встановлені всі поля, включаючи reporter_id (який має відповідати ID залогіненого користувача).
- Додаткові перевірки:
- Спроба зберегти завдання з незаповненими обов'язковими полями (очікується повідомлення про помилку валідації);
- Перевірка коректного відображення автора завдання в UI (reporterLabel в TaskDialogController та відповідна колонка в TaskControllerFx).

Приклад сценарію: "Фільтрація завдань"

Назва тесту: Коректна робота фільтрів списку завдань.

- Передумови: Користувач залогінений, вибрано проект, відображається список завдань з різними статусами та виконавцями.
- Кроки:
- У filterTaskStatusComboBox вибрати конкретний статус;
- Ввести ім'я існуючого виконавця в filterAssigneeField;
- Натиснути кнопку "Очистити фільтри".
- Очікуваний результат:
- При виборі статусу в таблиці залишаються тільки завдання з цим статусом;

- При введенні імені виконавця в таблиці залишаються тільки завдання, призначені на цього виконавця (або тих, чиє ім'я містить введений текст);
- При комбінації фільтрів результат відповідає обом критеріям;
- При натисканні "Очистити фільтри" відображаються всі завдання проекту.

3.5.5. Тестування безпеки

Мета: Перевірка механізмів аутентифікації та авторизації.

Об'єкти: LoginController, SecurityConfig, UserService (в частині UserDetailsService), доступ до функцій UI залежно від ролі.

Приклад тесту: "Вхід з невірними обліковими даними"

- Кроки:
- Запустити додаток;
- У вікні логіну ввести неіснуюче ім'я користувача або невірний пароль;
- Натиснути "Увійти".
- Очікуваний результат: Система не дозволяє вхід, відображається повідомлення про помилку "Неправильний логін або пароль" на errorLabel.

Приклад тесту: "Доступ до функцій адміністратора"

- Кроки:
- Залогінитися як користувач з роллю ROLE_USER (наприклад, "user"/"userpass" з DataInitializer.java);
- Спробувати отримати доступ до функціоналу управління користувачами (наприклад, чи видима кнопка userManagementButton в ProjectRainController);
- Залогінитися як користувач з роллю ROLE_ADMIN (наприклад, "admin"/"adminpass" з DataInitializer.java);
- Перевірити доступ до функціоналу управління користувачами.

- Очікуваний результат:
- Користувач ROLE_USER не бачить кнопку/не має доступу до управління користувачами;
- Користувач ROLE_ADMIN бачить кнопку/має доступ до управління користувачами.

3.5.6. Очікувані результати тестування

Після проведення всіх запланованих тестів очікуються наступні результати:

- Функціональність: Усі заявлені функції системи працюють коректно відповідно до вимог. Введення, обробка, зберігання та відображення даних відбувається без помилок;
- Інтерфейс користувача: Елементи GUI відображаються коректно, взаємодія з ними інтуїтивно зрозуміла та не викликає труднощів у користувача. Система надає адекватний зворотний зв'язок на дії користувача;
- Безпека: Механізми аутентифікації надійно захищають систему від несанкціонованого доступу. Розмежування прав на основі ролей працює коректно;
- Надійність та стабільність: Система стабільно працює при типових сценаріях використання. Обробка помилок введення та непередбачених ситуацій реалізована коректно, без аварійного завершення роботи програми;
- Дефекти: Всі виявлені критичні та блокуючі дефекти виправлені. Кількість некритичних дефектів знаходиться в межах, узгоджених із замовником (якщо такий є);
- Документація: Тест-план виконано, результати тестів задокументовані (наприклад, у вигляді звітів про тестування або записів у системі управління дефектами).

ВИСНОВКИ

Отже, у цій кваліфікаційній роботі було наведено розробку інформаційної системи "Project Rain" для компанії "Rain", щоб допомогти зробити централізовану систему управління завданнями, яка допомагала розробникам швидко відстежувати створені завдання та витрачати мінімальний час на з'ясування деталей та колег, які відповідальні за те чи інше завдання. При розробці системи як фундамент було вибрано неймовірно гнучкі технології, такі як: (Java), (Docker), (PostgreSQL), які в подальшому можуть стати у пригоді для розширення всієї системи та додавання нового функціоналу. Наприклад, розробка окремого модуля для управління персоналом: їх відпустками, їх святковими днями та інформацією про всіх співробітників, які є в системі. Очікуваний ефект від впровадження описано вище системою — це скорочення часу витрат на те, щоб аналізувати ту чи іншу задачу. У розробників з 15 хвилин в день до однієї-двох, що в довгій перспективі значно покращує і ефективність в роботі та якість виконуваних завдань завдяки наявному представленню одного завдання в одній системі. Щоб система вже могла використовуватись правильно, для цього потрібно перенести всі завдання, які були наявні в таблицях "Google Таблиці" до цієї системи, що вже значно покращує використання вже створених завдань. Для цього рекомендується визначити одного співробітника, який перенесе всі завдання до моєї системи. Для співробітників була написана документація до цієї системи, яку можна знайти в пункті 3.4.4 "Інструкція користувача". Розроблена система буде дуже часто використовуватися і вже в найкоротші терміни принесе небувалого росту продуктивності у відділ "IT".

Використані джерела

1. Блох Дж. “Effective Java 3rd ed”, 2018. “ Addison-Wesley” Ч. 6. С. 215.
2. Robert C. “Clean Code” 2015, “Addison-Wesley Professional”. Ч. 6. С. 345.
3. М’якшило Е., Харкянен В. Проектування та розробка програмного забезпечення: лабораторний практикум до виконання лабораторних робіт для здобувачів освітнього ступеня «бакалавр» спеціальності 122 «комп‘ютерні науки», опп «комп’ютерні науки» і «інформаційні системи та штучний інтелект», денної і заочної форм навчання. Нухт, 2022. – 102 с (дата звернення: 15.04.2025)
4. М’якшило О.М. CASE-технології у проектуванні інформаційних систем: навч. посібник для здобувачів вищих навчальних закладів. НУХТ, 2017. – 190 с. (дата звернення: 15.04.2025)
5. М’якшило О.М. Моделювання баз даних засобами CASE – технології ERWin: конспект лекцій/ О.М. М’якшило – Київ.:НУХТ, 2007 – 60 с. (дата звернення: 15.04.2025)
6. О. М. М’якшило, О. В. Харкянен. Проектування інформаційних систем : методичні рекомендації до виконання курсового проекту для студентів освітнього ступеня «Бакалавр» спеціальності 122 «Комп’ютерні науки» денної та заочної форм навчання: НУХТ, 2018. – 24 с.4 (дата звернення: 15.04.2025)
7. Грибков С. В., Лімaнська Н. В., Костіков М. П. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання кваліфікаційної роботи на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки», освітньо-професійної програми «Комп’ютерні науки» денної та заочної форм навчання. Київ Нухт 2025. С 15-43. (дата звернення: 15.04.2025)
8. Jasper P., Nancy H., Joni G., Cindy C. “JavaFX Getting Started with JavaFX Release 8”. 2014. С 10-54. URL:

https://docs.oracle.com/javase/8/javafx/JFXST.pdf (дата звернення: 15.04.2025)

9. Cindy C. “JavaFX Scene Builder Getting Started with JavaFX Scene Builder Release 2.0”. 2014. C 10-49. URL: https://docs.oracle.com/javase/8/scene-builder-2/JSBGS.pdf (дата звернення: 15.04.2025)
10. Oracle documentation. “Catching and Handling Exceptions”. 2021. URL: https://dev.java/learn/exceptions/catching-handling/ (дата звернення: 15.04.2025)
11. Baeldung. “Java Collections Series”. 2024. UPL: https://www.baeldung.com/java-collections (дата звернення: 15.04.2025)
12. Apache Maven. “Introduction to the Build Lifecycle” URL: https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html (дата звернення: 15.04.2025)
13. Baeldung “Introduction to Project Lombok” URL: https://www.baeldung.com/intro-to-project-lombok (дата звернення: 15.04.2025)
14. Gavin K. “A Short Guide to Hibernate 7”. 2025. URL: https://docs.jboss.org/hibernate/orm/7.0/introduction/html_single/Hibernate_Introduction.html (дата звернення: 15.04.2025)
15. Gavin K. “A Guide to Hibernate Query Language”. 2025. URL: https://docs.jboss.org/hibernate/orm/7.0/querylanguage/html_single/Hibernate_Query_Language.html (дата звернення: 15.04.2025)
16. Baeldung “Spring Framework Introduction”. 2023. URL: https://www.baeldung.com/spring-intro (дата звернення: 15.04.2025)
17. Baeldung “Learn Spring Boot Series”. 2024. URL: https://www.baeldung.com/spring-boot (дата звернення: 15.04.2025)

18. Baeldung “REST with Spring Series”. 2023. URL: https://www.baeldung.com/rest-with-spring-series (дата звернення: 15.04.2025)
19. Baeldung “Spring Data Series”. 2023. URL: https://www.baeldung.com/spring-data (дата звернення: 15.04.2025)
20. Baeldung “Security with Spring Series”. 2024. URL: https://www.baeldung.com/security-spring (дата звернення: 15.04.2025)
21. Baeldung “Spring Dependency Injection Series”. 2024. URL: https://www.baeldung.com/spring-dependency-injection (дата звернення: 15.04.2025)
22. Sam M. “A Solid Guide to SOLID Principles”. 2025. URL: https://www.baeldung.com/solid-principles (дата звернення: 15.04.2025)
23. Nageswara R. “Object Oriented Programming through JAVA”. 2025. UPL: https://mrcet.com/downloads/digital_notes/CSE/II%20Year/CS/OBJECT%20ORIENTED%20PROGRAMMING%20THROUGH%20JAVA.pdf (дата звернення: 15.04.2025)
24. Anshul B. “Best Practices for Unit Testing in Java”. 2024 URL: https://www.baeldung.com/java-unit-testing-best-practices (дата звернення: 15.04.2025)
25. PostgreSQL Global Development Group. “Chapter 2. The SQL Language”. 2025. URL: https://www.postgresql.org/docs/current/tutorial-sql-intro.html (дата звернення: 15.04.2025)
26. PostgreSQL Global Development Group. “Chapter 4. SQL Syntax”. 2025. URL: https://www.postgresql.org/docs/17/sql-syntax.html (дата звернення: 15.04.2025)
27. Mozilla Foundation. “CSS: Cascading Style Sheets”. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 15.04.2025)

28. Kalana De S. “The Decorator Design Pattern: Simplifying Dynamic Feature Addition”. 2025. URL: https://medium.com/@kalanamalshan98/the-decorator-design-pattern-simplifying-dynamic-feature-addition-446de438c5b6 (дата звернення: 15.04.2025)
29. Refactoring guru. “Builder’ URL: https://refactoring.guru/design-patterns/builder (дата звернення: 15.04.2025)
30. Pragimtech. “How is data stored in SQL database” URL: https://www.pragimtech.com/blog/sql-optimization/how-is-data-stored-in-sql-database/
31. Erwin. “Documentation” 2021. URL: https://bookshelf.erwin.com/bookshelf/public_html/2021R1/Content/Release%20Notes/Data%20Modeler%20Release%20Notes/6940.html

ДОДАТКИ

Додаток А. Логічна модель БД

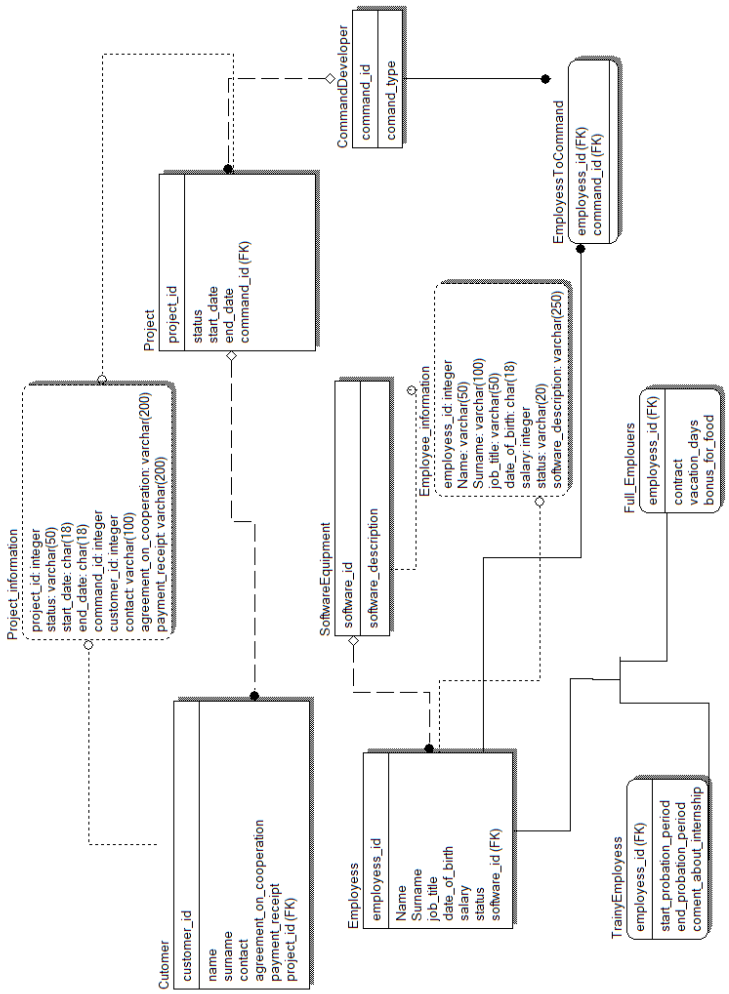

Рисунок А.1— Логічна схема бази даних

Додаток Б. Фізична модель БД

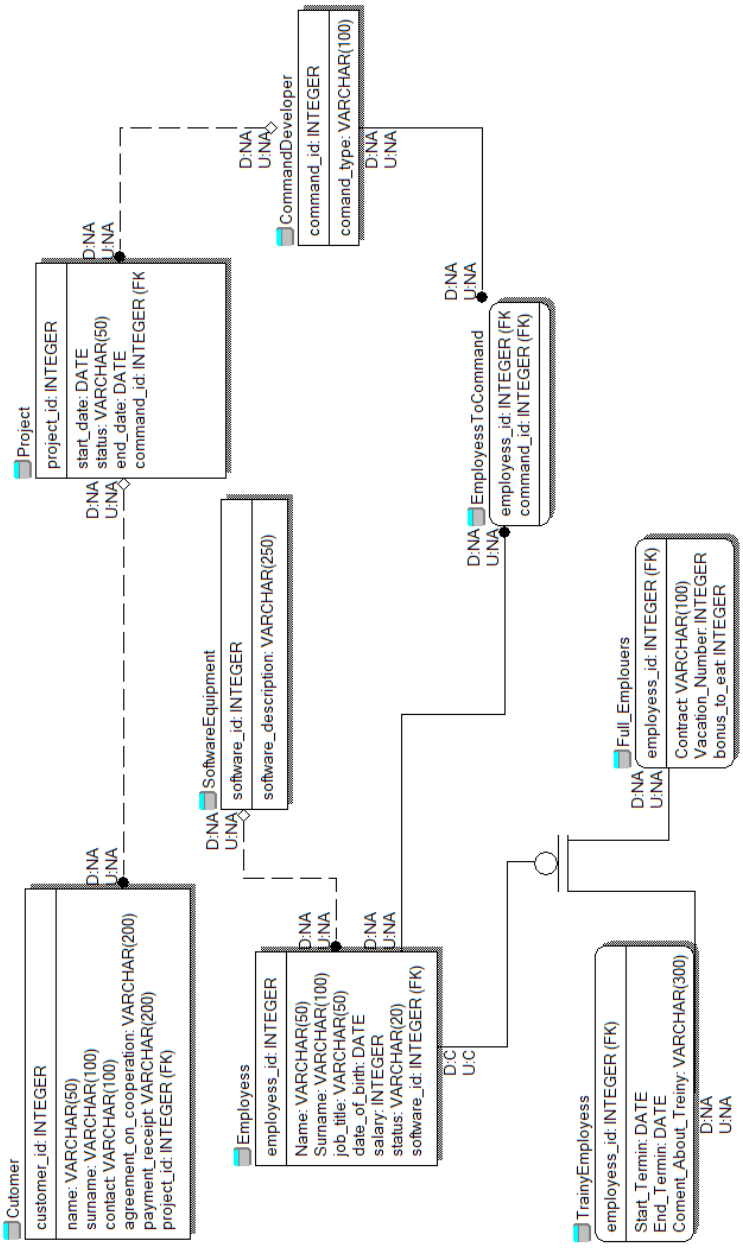

Рисунок Б.1— Фізична схема бази даних