МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра оптоелектроніки та інформаційних технологій

Допустити до захисту

Завідувач кафедри

_____ проф. Кушнір О. С.

«___» _____2023 р.

Кваліфікаційна робота

Бакалавр

(освітній ступінь)

ЗАСТОСУНОК ДЛЯ КОНТРОЛЮ РОБОЧОГО ЧАСУ ТА ВИКОНАННЯ ПОСТАВЛЕНИХ ЗАВДАНЬ

Виконав:

студент IV курсу групи ФЕП – 42с

спеціальності 121 Інженерія програмного забезпечення

_____ ПЛОТНІКОВ А. О.

(підпис) (прізвище та ініціали)

Науковий керівник:

_____ доц. ГРАБОВСЬКИЙ В. А.

(підпис) (прізвище та ініціали)

«____» _____2023 р.

Рецензент:

_____ доц. БОЙКО Я. В.

(підпис) (прізвище та ініціали)

Львів 2023

Анотація

Дипломна робота містить 62 сторінок, 15 таблиць, 27 рисунків, список використаних джерел із 29 найменувань та 3 додатки.

Метою дипломної роботи є розробка застосунку, який забезпечує моніторинг, комунікацію, актуалізацію даних при виконання IT проектів абсолютно безкоштовним з повним функціоналом та open source.

Об’єктом дослідження є процеси розробки програмного забезпечення з використанням таск-трекінгових систем.

Предметом дослідження є сучасні методи, засоби та технологій для розробки таск-трекінгових систем.

Методи розробки базуються на технології JAVA з використанням бази даних PostgreSQL [https://www.postgresql.org/] [https://uk.wikipedia.org/wiki/SQL.]

Одержані результати полягають в розробці системи, яка надає можливість для розробника бачити повний список завдань, які потрібно виконати, а для менеджера - побачити на якій стадії знаходиться розробка проекту для інформування замовника.

Annotation

Thesis contains 62 pages, 25 tables, 27 figures, list of sources with 29 titles and 3 applications.

The aim of the thesis is the development of an application that provides monitoring, communication, updating of data while executing IT projects completely free with full functionality and open source.

Object of research are software development processes using task tracking systems.

The subject of research are modern methods, tools and technologies for developing traction tracking systems.

Development methods are based on JAVA technology using the PostgreSQL database.

The resulting is to develop a system that gives developers the opportunity to see a complete list of tasks that need to be done, and for the manager to see at what stage is the development of the project to inform the customer.

ЗМІСТ

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ, АЛГОРИТМІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7

1.1 Дослідження предметної області 7

1.2 Опис об'єкта дослідження 15

1.3. Аналіз існуючих програмних рішень та постановка задач проекту 16

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ 22

2.1 Розроблення архітектури програмної системи 22

2.2 Аналіз існуючих алгоритмів вирішення поставленої задачі 28

2.3 Проектування структури бази даних 30

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 35

3.1 Програмна реалізація системи 35

3.2. Програмні модулі системи 40

3.2.1. Компоненти ПЗ 40

3.2.2. Встановлення ПЗ 41

3.2.3 Базові функції ПЗ. 43

3.3 Тестування та верифікація розробленого програмного забезпечення 53

3.3.1 Функціональне тестування 53

3.3.2 GUI тестування 54

3.3.3. Тестування безпеки 56

ВИСНОВКИ 57

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58

Додаток А 61

Вихідний код класів сутностей 61

x ..... **Error! Bookmark not defined.**

ВСТУП

Сьогодні IT-компанії створюють великі комерційні проекти, що залучає величезну кількість людських ресурсів. Сам процес розробки, звичайно, розподіляється між членами команди. Щоб керівник проекту міг ефективно керувати проектом, він повинен знати, на якому етапі знаходиться процес розробки того чи іншого прикладного модуля. Перед початком розробки проекту його менеджери планують роботу таким чином, щоб усі, хто бере участь у процесі розробки, співпрацювали. Проте все ще трапляються випадки, коли деякі члени команди працюють віддалено. У цьому випадку завдання супервайзерів полягає в тому, щоб організувати робочий процес таким чином, щоб усі співробітники могли легко спілкуватися один з одним у разі необхідності. Так, ви можете використовувати альтернативні джерела спілкування, такі як телефон, соціальні мережі або Skype. Але є набагато простіший і практичніший спосіб - використовувати таск - трекінгову систему, яка надзвичайно полегшує цей процес. У цьому випадку керівник може легко контролювати результати роботи конкретного співробітника або всього колективу. По-перше, метою цих систем є контроль якості виконання продукції, оскільки більшість помилок можливо виявити на етапі розробки, тоді як критичні помилки неможливо виявити на етапі підтримки. І наступна мета - забезпечити спілкування всіх членів команди для полегшеної співрпаці між собою. Як правило, такі системи здатні надавати такі функції за додаткову оплату і коштують само по собі недешево. Малий бізнес зазвичай не може дозволити собі такі інструменти. Крім того, у процесі розробки продукту необхідно використовувати й інші, можливо, дорожчі програми. Практично всі IT-компанії використовують такі програмні продукти. Актуальним завданням є розробка програми Lazy_Track, яка забезпечує моніторинг, зв’язок, оновлення даних під час реалізації IT-проектів, абсолютно безкоштовно з повним функціоналом та відкритим кодом. Щоб досягнути даної мети необхідно виконати наступні завдання:

- дослідити існуючі аналоги;
- спроектувати для розроблюваного продукту функціональні та нефункціональні вимоги;
- розробити модель бази даних.

Об’єктом дослідження є розробка та реалізація програмного продукту з застосуванням застосунку для контролю часу та виконання поставлених задач. Предметом дослідження є метод, який використовує засіб та технології для того, щоб розробити даний застосунок. Для розробки системи скористаюсь технологіями необхідними для розробки: SQL - це стандартна мова запитів, що використовується для управління та маніпулювання реляційними базами даних. За допомогою SQL я можу:

- вивчити існуючі аналоги;
- планувати функціональні та нефункціональні вимоги до розробленого продукту;
- розробляти модель бази даних.

Тема роботи - розробка та впровадження програмного продукту з використанням систем відстеження завдань та створення чату для комунакації між працівниками над проектом.

Об’єктом дослідження є метод, який використовує інструменти та технології для розробки системи відстеження завдань.

Я використовую технології для розвитку системи бази даних, які, у свою чергу, є пов’язаними даними, що зберігаються в таблиці. JAVA [https://uk.wikipedia.org/wiki/JavaScript] - це об’єктно – орієнтовна мова програмування, яка використовується для створення програм, що виконують різні важливі та другорядні функції. Слід зазначити, що JAVA - це не лише мова сама по собі, а й платформа, на якій можна створювати та використовувати програми, написані на мові JAVA. Важливо є те, що за допомогою такого продукту розробник бачить весь перелік завдань, які необхідно реалізувати, а менеджер стежить за стадією розробки продукту, щоб звітувати перед замовником, та в разі чого керівник може вносити якісь певні корективи при виконанні робити над

проектом.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ, АЛГОРИТМІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Дослідження предметної області

IT-компанії розробляють великі проекти і потребують залучення висококваліфікованих спеціалістів. Завдання розподіляються між командою. Тому керівник повинен весь час контролювати виконання проекту. Робота планується так, щоб працювала вся команда. Але іноді співробітники працюють віддалено. Крім того, виникає низка комунікаційних проблем, коли зовнішні працівники переглядають виконані завдання. Варіантів багато: соц. Мережа, мобільний зв'язок, Skype. Але найкращим чином це так зване використання системи відстеження задач (англ. task tracking system). Такі системи дозволяють розподіляти завдання всім співробітникам, дистанційно контролювати їх виконання та негайно вирішувати проблеми. Для формалізації та опису бізнес-процесів використовується мова моделювання IDEF0. Мова моделювання IDEF0 (Integration Definition for Function Modeling) є одним з нотаційних засобів, що використовується для моделювання бізнес-процесів, функціональних структур та інших системних аспектів.

Основні елементи, які використовуються у мові IDEF0, включають:

- Функції (Function): Представлені у вигляді блоків, що виконують операції або процеси. Функції можуть мати вхідні та вихідні потоки даних.
- Вхідні/вихідні потоки даних (Input/Output Data): Показують потоки інформації або даних, які входять у функцію або виходять з неї.
- Управління (Control): Відображає механізми керування функціями, які визначають порядок виконання операцій.
- Механізми (Mechanism): Показують ресурси, що використовуються для виконання функцій, такі як обладнання, програмне забезпечення, люди і т.д.
- Об'єкти (Object): Використовуються для позначення об'єктів або сутностей, які взаємодіють з функціями.

Методологія цієї системи являє собою серію робіт і заходів. Було побудовано ієрархічно - об'єктну модель, яка спрощує розуміння предметної області. IDEF0 показує логічні зв'язки між роботами, а не послідовність виконання в часі.

Найважливі процеси застосунку:

1. написання UserStory;
2. комплектування команди;
3. робота на час виконання;
4. завершення проекту;

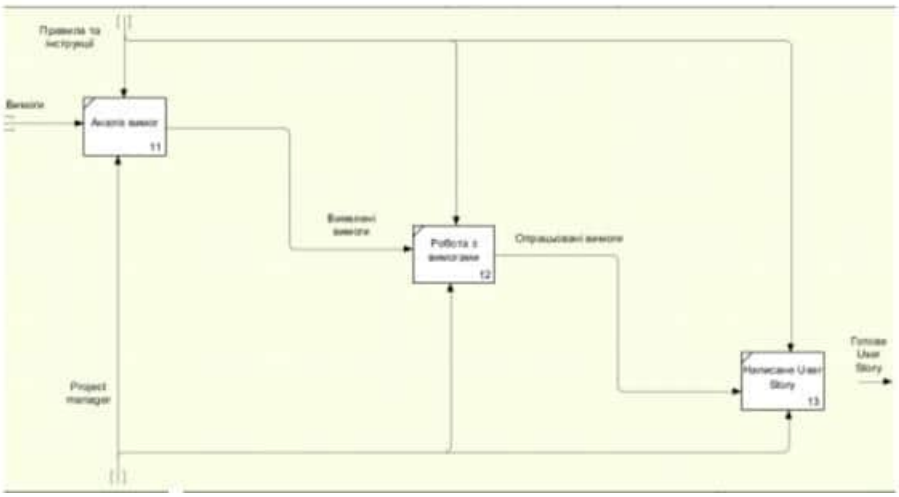

Рис. 1.1 – Діаграма бізнес-процесів розроблюваного програмного продукту.

Давайте детальніше розглянемо кожен з перерахованих вище процесів. На малюнку 1.2 показано процес написання UserStory. Щоб написати історію користувача, менеджери повинні проаналізувати всі умови. Всі вимоги необхідно проаналізувати з замовником, тільки після цього можна приступати до написання. Характеристики процесу написання історії користувача наведено в таблиці 1.1.

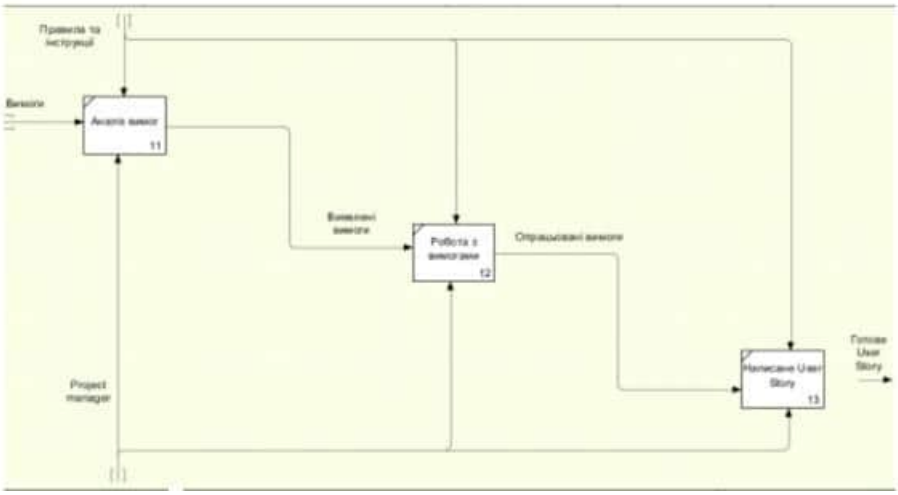

Рис. – 1.2 Діаграма процесу написання User Story

Таблиця 1.1 – Характеристика процесу “Написання User Story”

Назва характеристики	Значення характеристики
Ім'я процесу	Написання User Story
Основні учасники	Замовник, PM
Вхідна подія	Виявлення вимог
Вхідні документи	Список виявлених вимог
Вихідна подія	Готова User Story
Вихідні документи	Список User Stories
Клієнти процесу	Користувач

Наступний процес – формування команди для роботи над написанням проекту. Процес створення групи описано на рисунку 1.3

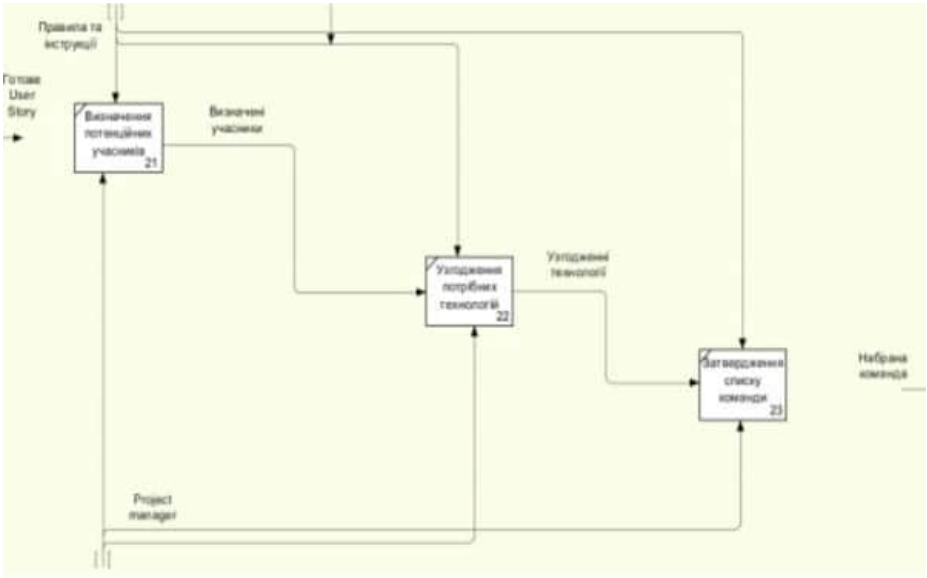

Рис. 1.3 – Діаграма процесу створення команди

Перед формуванням команди необідно написати UserStory користувачів, що необхідно для визначення необхідної кількості співробітників та їхніх знань для залучення до розробки проекту. Після визначення групи необхідно обговорити технології, необхідні для реалізації проекту. Тільки після цього остаточно затвердити команду для розробки, адже та чи інша особа може бути не повністю кваліфікована в виконанні певних задач, тому може зустрітися така ситуація, що працівника потрібно буде замінити на іншого. Характеристика процесу набору групи представлена в таблиці 1.2. Після затвердження команди можна приступати до написання проекту. Для цього створюється Sprint, так зване діалогове вікно де описуються завдання та стани їх виконанні при створенні проекту. Процес роботи в Sprint описаний на рисунку 1.4. Перед початком створення нового Sprint необхідно розробити завдання, які потрібно реалізувати. Наступний крок – виконання поставлених завдань. Після виконання завдань їх необхідно перевірити на наявність помилок і при їх виявленні обов’язково виправити.

Таблиця 1.2 – Характеристика процесу “Набір команди”

Назва характеристики	Значення характеристики
Ім'я процесу	Набір команди
Основні учасники	PM, потенційні учасники команди
Вхідна подія	Виявлення User Stories
Вхідні документи	Список User Stories
Вихідна подія	Затвердження списку учасників команди
Вихідні документи	Список учасників команди
Клієнти процесу	Користувач

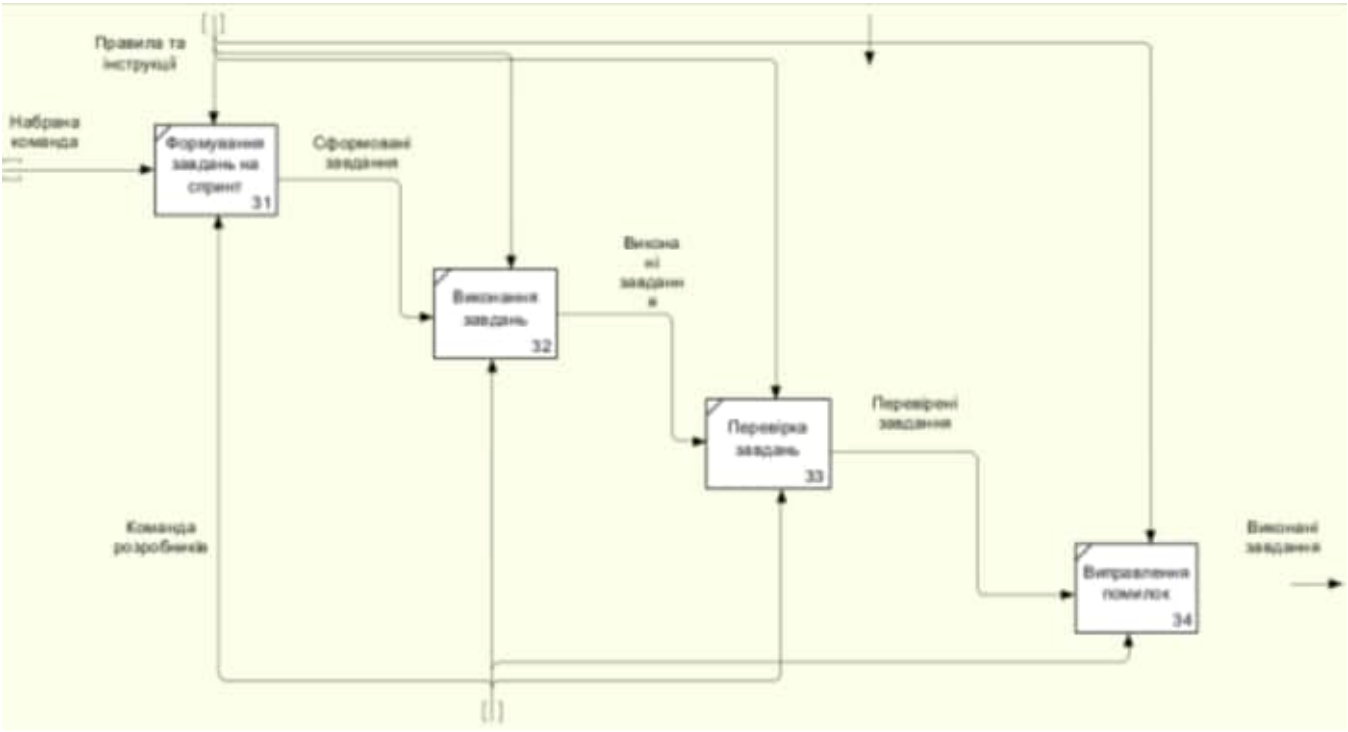

Рис. 1.4 – Діаграма процесу роботи зі спринтом

Характеристику процесу роботи зі спринтом наведено в таблиці 1.3.

Таблиця 1.3 – Характеристика процесу “Робота зі спринтом”

Назва характеристики	Значення характеристики
Ім'я процесу	Робота зі спритом
Основні учасники	Учасники команди
Вхідна подія	Затвердження списку учасників команди
Вхідні документи	Список учасників команди
Вихідна подія	Виконані завдання
Вихідні документи	Список виконаних завдань
Клієнти процесу	Користувач

Коли всі завдання виконані та ви переконані, що помилки знайдено та виправлено проект можна завершувати. На рисунку 1.5 показана схема процесу закриття проекту.

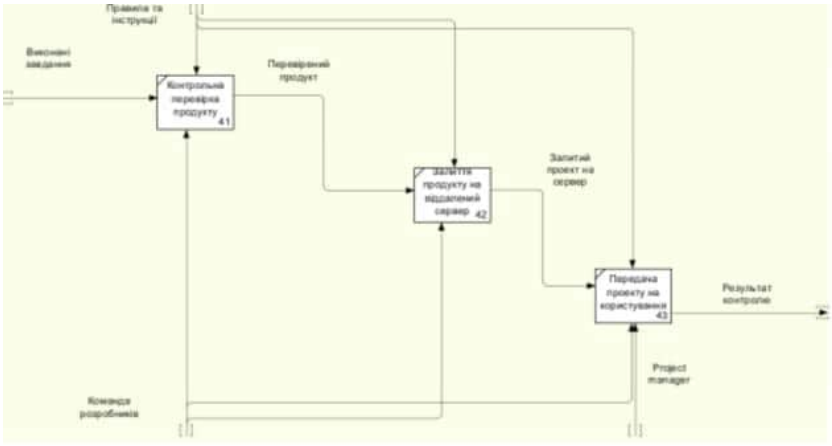

Рисунок 1.5 – Діаграма процесу закриття проекту

Характеристики на момент завершення проекту наведені в таблиці 1.4

Таблиця 1.4 – Характеристика бізнес-процесу “Закриття проекту”

Назва характеристики	Значення характеристики
Ім'я процесу	Закриття проекту
Основні учасники	Учасники команди, PM
Вхідна подія	Виконані завдання
Вхідні документи	Список виконаних завдань
Вихідна подія	Результат контролю перевірки

1.2 Опис об'єкта дослідження

Сьогодні IT-галузь розвивається дуже швидко. Практично кожен день відкриваються нові компанії в цьому напрямку. До прикладу візьмемо Taste Technologies – це досить нова компанія, яка займається розробкою різного програмного забезпечення. Компанія була заснована в 2015 році в Україні. Персонал досить молодий, але досвідчений і компетентний. Компанія фокусується на впровадженні особливих рішень, що відрізняються унікальністю та відповідають вимогам клієнтів. Співробітники мають досвід розробки сучасних веб-додатків. Складність і розмір проектів не заважає компанії пропонувати своїм клієнтам нові технології, сучасні розробки та відмінну дизайнерську допомогу.

Рис. 1.6 – Структура проектної організації

Сьогодні компанія реалізує кілька проектів електронного маркетингу. Через великий обсяг проекту керівник повинен контролювати виконання всіх покладених на нього завдань. Для таких цілей компанія використовує систему відстеження завдань «YouTrack». Завдяки такому продукту можна легко підтримувати поточний статус проекту. Необхідно лише стежити за сторінкою, яка показує обсяг виконаної роботи. Компанія Taste Technologies

1.3. Аналіз існуючих програмних рішень та постановка задач проекту

Сьогодні існують різні програми для відстеження завдань. Вони були створені відповідно до конкретних потреб користувача. Перед початком розробки такої програми необхідно оцінити та проаналізувати вже існуючі інструменти, щоб зрозуміти специфікації таких систем і врахувати все це при розробці нової технологічної системи для завдань моніторингу. Для порівняння вибрано дві поширені системи, такі як YouTrack і Bugzilla [https://www.bugzilla.org/.]. Одна з цих систем платна, а інша безкоштовна. YouTrack [https://www.jetbrains.com/youtrack/.] - це комерційний продукт для управління проектами відстеження дефектів, розроблений JetBrains. JetBrains є компанією, що спеціалізується на розробці інтегрованих середовищ розробки (IDE) та інших інструментів для програмістів і розробників програмного забезпечення. Вони створюють популярні продукти, які допомагають розробникам писати код більш ефективно і продуктивно. Одним з найвідоміших продуктів JetBrains є IntelliJ IDEA, інтегроване середовище розробки для Java. YouTrack підтримує пошук, автозавершення, маніпулювання наборами завдань, встановлення набору атрибутів завдання, створення користувацьких робочих процесів і активне використання клавіатури в інтерфейсі користувача (що важливо для багатьох програмістів).

Після реєстрації користувач переходить на нову сторінку, де можна створити новий проект. Наступним кроком є вікно, яке містить дошку та беклог, вони описують усі завдання, які необхідно виконати для проекту. Це зображено на рисунку 1.6

Рис. 1.6 – Вікно процесу реєстрації

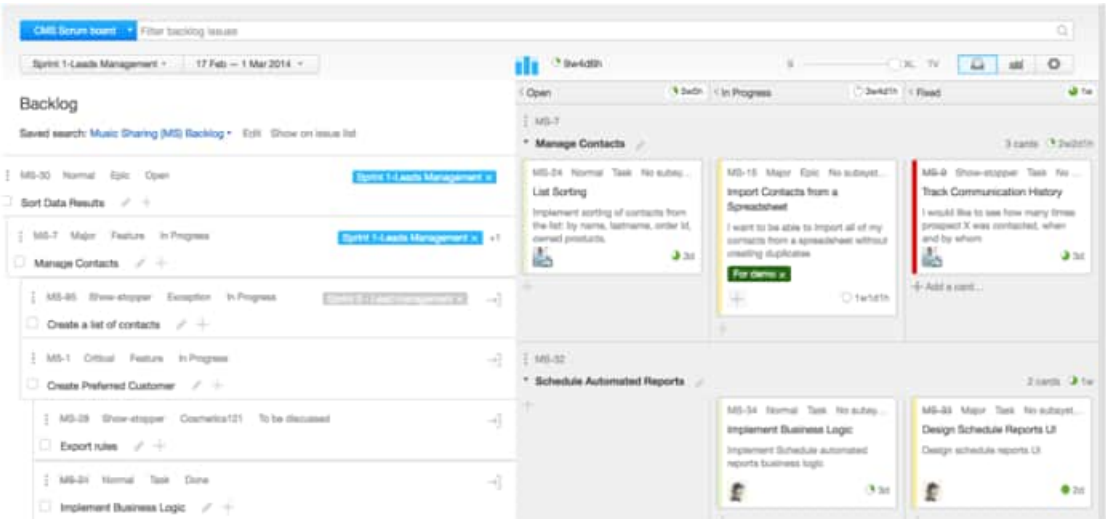

Рис. 1.7 – Дошка з беклогами

Bugzilla [https://www.bugzilla.org/features/.] - це система управління помилками та відстеження проблем, що виникають під час розробки програмного забезпечення. Вона була розроблена і поширюється як вільне програмне забезпечення, що дозволяє розробникам ефективно відстежувати, керувати та вирішувати проблеми, пов'язані з програмними дефектами, помилками та іншими завданнями розробки.

Bugzilla дозволяє створювати, відстежувати та організовувати списки завдань та проблем, що потребують уваги розробників. Вона дозволяє розробникам фіксувати, описувати та класифікувати помилки, а також встановлювати пріоритети, відповідальних осіб та інші атрибути для кожного завдання. Також можна використовувати коментарі та відстежувати статуси для кожної проблеми. Bugzilla дозволяє команді розробників ефективно спілкуватися між собою, спільно працювати над вирішенням проблем і забезпечує централізовану базу знань, яка допомагає уникати повторних помилок та використовувати вже наявний досвід. Ця система стала дуже популярною у галузі розробки програмного забезпечення, і вона використовується великим числом проектів та організацій для відстеження та управління помилками в їхніх програмах.

Рис. 1.8 – Головна сторінка Bugzilla

Після етапу реєстрації користувач повинен створити проект. Щоб завершити цей процес, ви повинні вводити помилки. На рисунку 1.9 показана форма для створення звітів про помилки.

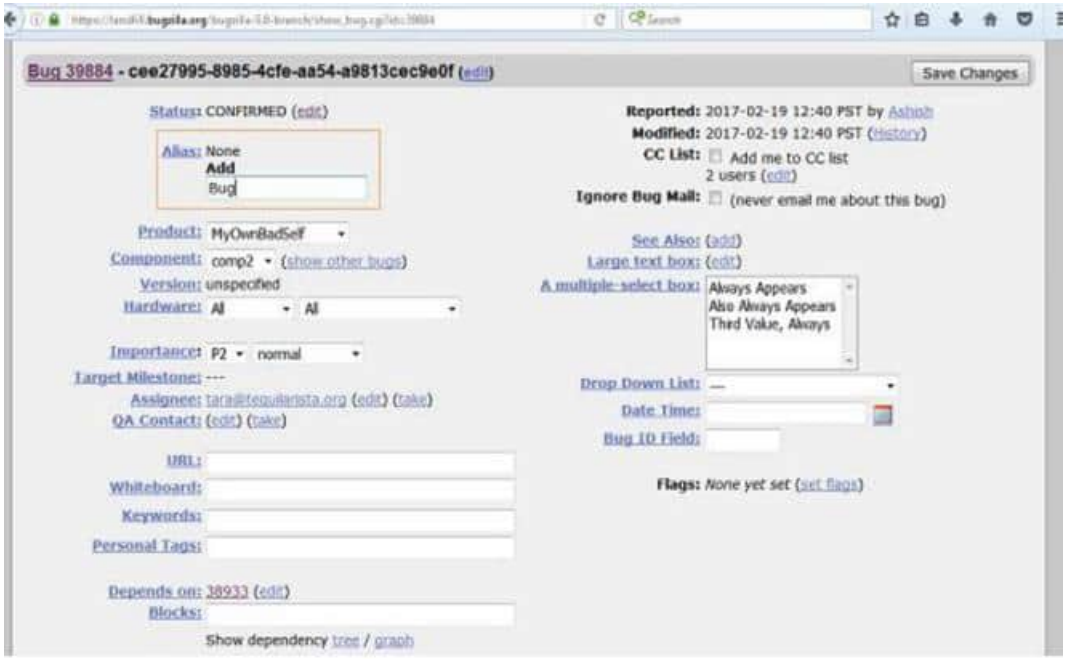

Рис. 1.9 Форма для створення звітів про помилки.

Таблиця 1.5 – Порівняльна характеристика найбільш поширених систем

Фірма-розробник	JetBrains	Mozilla
1	2	3
Назва програмного продукту	YouTrack	Bugzilla
Остання версія	2018.1	5.1.2
Функціональність	- створення завдань; - оформлення багів; - можливість створювати Scrum та Kanban дошки для роботи; - індикатор прогресу; - пошук завдань; - експорт всіх завдань;	-розширені можливості пошуку - списки помилок у кількох форматах - заплановані звіти - звіти та діаграми - відстеження часу
Інтеграція з сторонніми сервісами	GitHub, IntelliJ IDEA, TestLink, TestRail,	TestLink, інтеграція з системою електронної

	Subversion, CVS	пошти.
Інтерфейс користувача	Дружній та зрозумілий	Застарілий інтерфейс з низьким рівнем юзабіліті.
Допомога користувачу	Для допомоги користувачу створено багато відео-уроків, де показано як створити	Є багато документації по роботі з системою

Продовження таблиці 1.5

1	2	3
Фірма-розробник	JetBrains	Mozilla
Ціна за користування	від $200 за рік	безкоштовна
Переваги	- швидкодія; - зрозумілий інтерфейс; - інтеграція з	- безкоштовна - простота - автоматизація роботи з

	сторонніми продуктами;	документацією - тісна інтеграція з системою електронної пошти.
Недоліки	- ціна; - відсутня можливість спілкування з командою за допомогою чату;	- обмежений функціонал; - відсутня можливість спілкування з командою за допомогою чату;

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Розроблення архітектури програмної системи

При проектуванні системи використовувалася трирівнева архітектура.

Трирівнева архітектура (також відома як трьохрівнева архітектура) є популярним підходом до проектування та розробки програмного забезпечення. Вона розділяє програму на три основні рівні або компоненти, кожен з яких відповідає за певні функції та завдання. Ця архітектура забезпечує розподіл функціональності та зменшує залежності між компонентами, що полегшує розробку, супровід та масштабування системи. Основні рівні трьохрівневої архітектури зазвичай включають:

- Представлення (Presentation Layer): Цей рівень відповідає за відображення інтерфейсу користувача та взаємодію з ним. Він включає компоненти, такі як веб-сторінки, графічні інтерфейси користувача, служби API тощо. Його головна мета - забезпечити користувачам доступ до функціональності системи та обробку їхніх взаємодій.
- Бізнес-логіка (Business Logic Layer): Цей рівень містить логіку та правила, які визначають основні функції та обробку даних системи. Він включає компоненти, такі як сервіси, контролери або модулі, які відповідають за обробку запитів, виконання бізнес-правил, доступ до бази даних та інші функціональні операції.
- Дані (Data Layer): Цей рівень відповідає за збереження та доступ до даних системи. Він включає базу даних або інші засоби збереження даних, а також компоненти для забезпечення доступу, зчитування та запису даних. Його головна мета - забезпечити надійне збереження та обробку дани

. Структурна схема програмної системи наведена на рисунку 2.1.

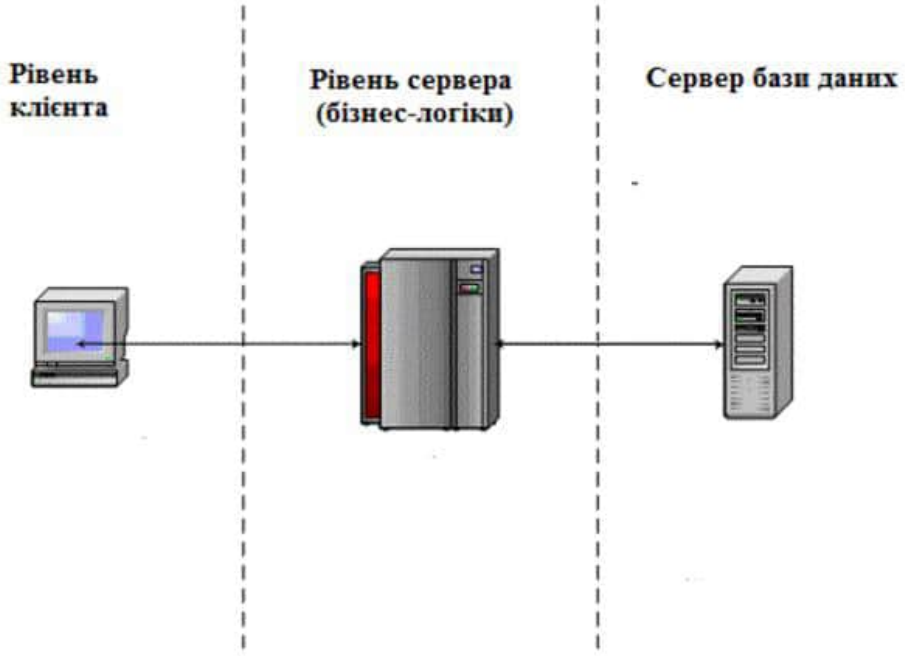

Рис. 2.1 – Структурна схема програмної системи

Рівень клієнта – створюється і розраховується під користувача. Перший рівень не повинен мати прямого зв’язку з базою даних, він не повинен завантажувати основну бізнес-логіку та не повинен зберігати стан програми.

Цей рівень повинен містити найлегшу логіку: блок авторизації, який перевіряє введені значення відповідно до вимог і обов'язків конкретної організації.

Другий рівень — серверний. Цей рівень показує основну частину бізнес-логіки. Він має програмний інтерфейс, який підключає частини користувача до логіки бази даних. Гарантію безпеки даних забезпечує сервер бази даних, він входить до третього рівня. Він являє собою базу даних процедур, тригерів і умов для реляційної моделі. Для представлення статичної сторони архітектури представлена діаграма компонентів, яка показана на малюнку 2.2

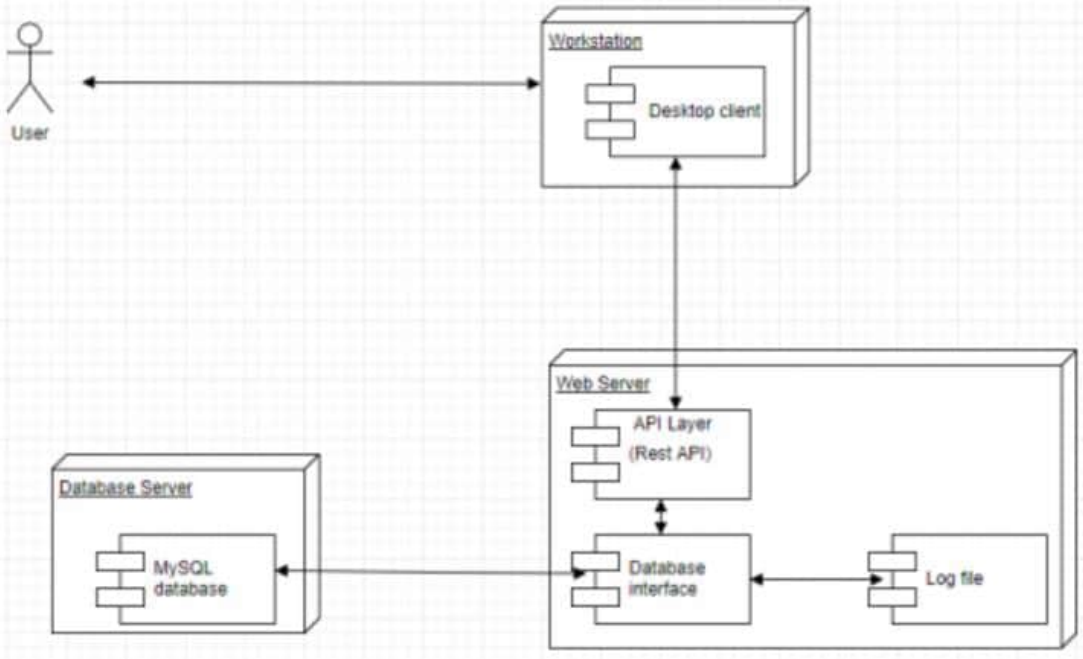

Рис. 2.2 – Діаграма компонентів

За введення даних клієнта відповідає компонент «АРМ». Компонент АРМ (Application Performance Monitoring) - це інструмент або програмний продукт, який використовується для моніторингу та управління продуктивністю додатків. АРМ допомагає розробникам та операторам систем відстежувати та аналізувати продуктивність додатків в реальному часі з метою виявлення проблем, оптимізації роботи та підвищення якості виконання.

Компоненти АРМ зазвичай надають наступні можливості:

- Моніторинг продуктивності: АРМ збирає дані про час відповіді, завантаженість ресурсів, обсяги трафіку та інші метрики, що відображають продуктивність додатків.

- Трасування та аналіз: APM відстежує шляхи виконання запитів та транзакцій у системі, що дозволяє виявляти проблемні місця та здійснювати аналіз продуктивності з точки зору коду, бази даних, мережі та інших компонентів.
- Алерти та сповіщення: APM може надсилати повідомлення або сповіщення про виявлені проблеми або несправності, допомагаючи операторам систем швидко реагувати та вирішувати проблеми продуктивності.
- Профілювання: APM дозволяє аналізувати роботу додатків на рівні коду, ідентифікувати гарячі точки, витратні операції та інші елементи, що можуть впливати на продуктивність.
- Аналітика та звітність: APM надає засоби для аналізу та візуалізації даних продуктивності, які допомагають зрозуміти та вдосконалити працездатність системи.

Компонент APM може бути інтегрований у різноманітні середовища розробки та експлуатації програмного забезпечення для покращення управління продуктивністю та якості додатків.

Компонент «Веб-сервер» відповідає за отримання вхідних даних від компонента Workstation за допомогою API, обробку їх за допомогою бізнес-логіки та підготовку до взаємодії з базою даних. Компонент «Сервер баз даних» обробляє запити, підготовлені на попередньому кроці.

Компонент архітектури представлений за допомогою діаграм стану.

Діаграма стану для створення проекту показана на рисунку 2.3.

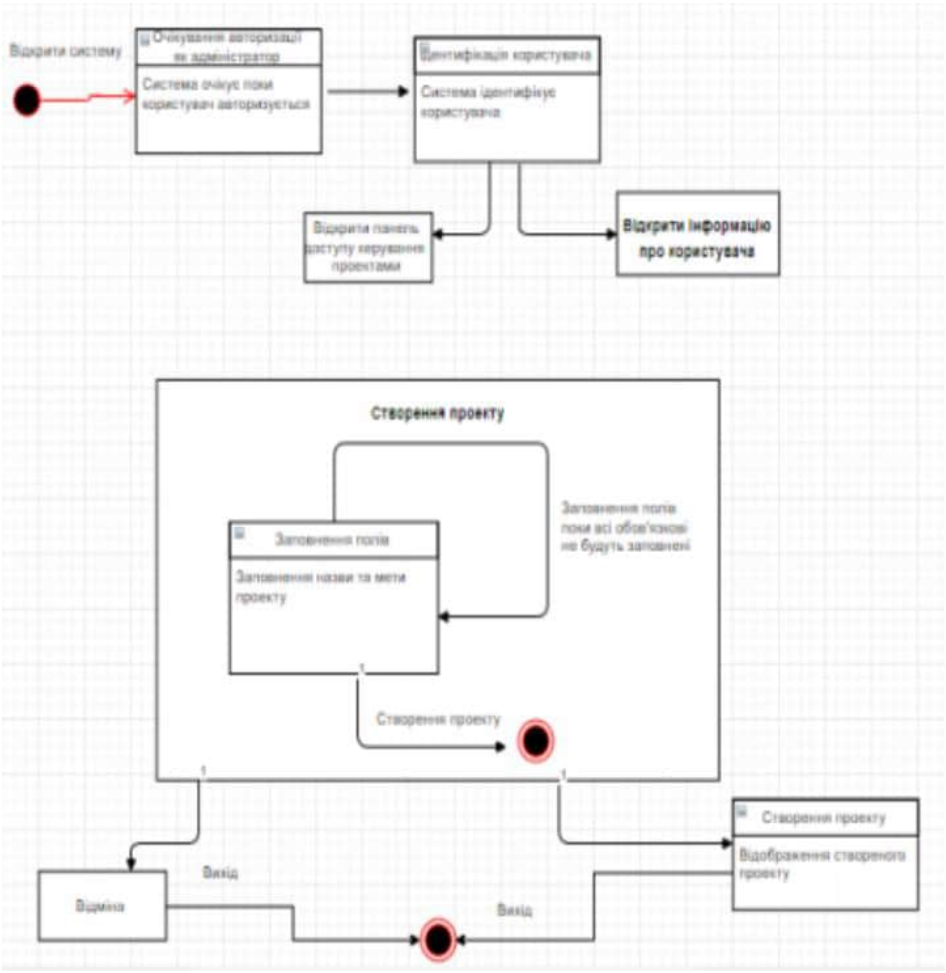

Рисунок 2.3 – Діаграма станів для створення проекту

На рисунку 2.4 зображено діаграму станів для створення Sprint

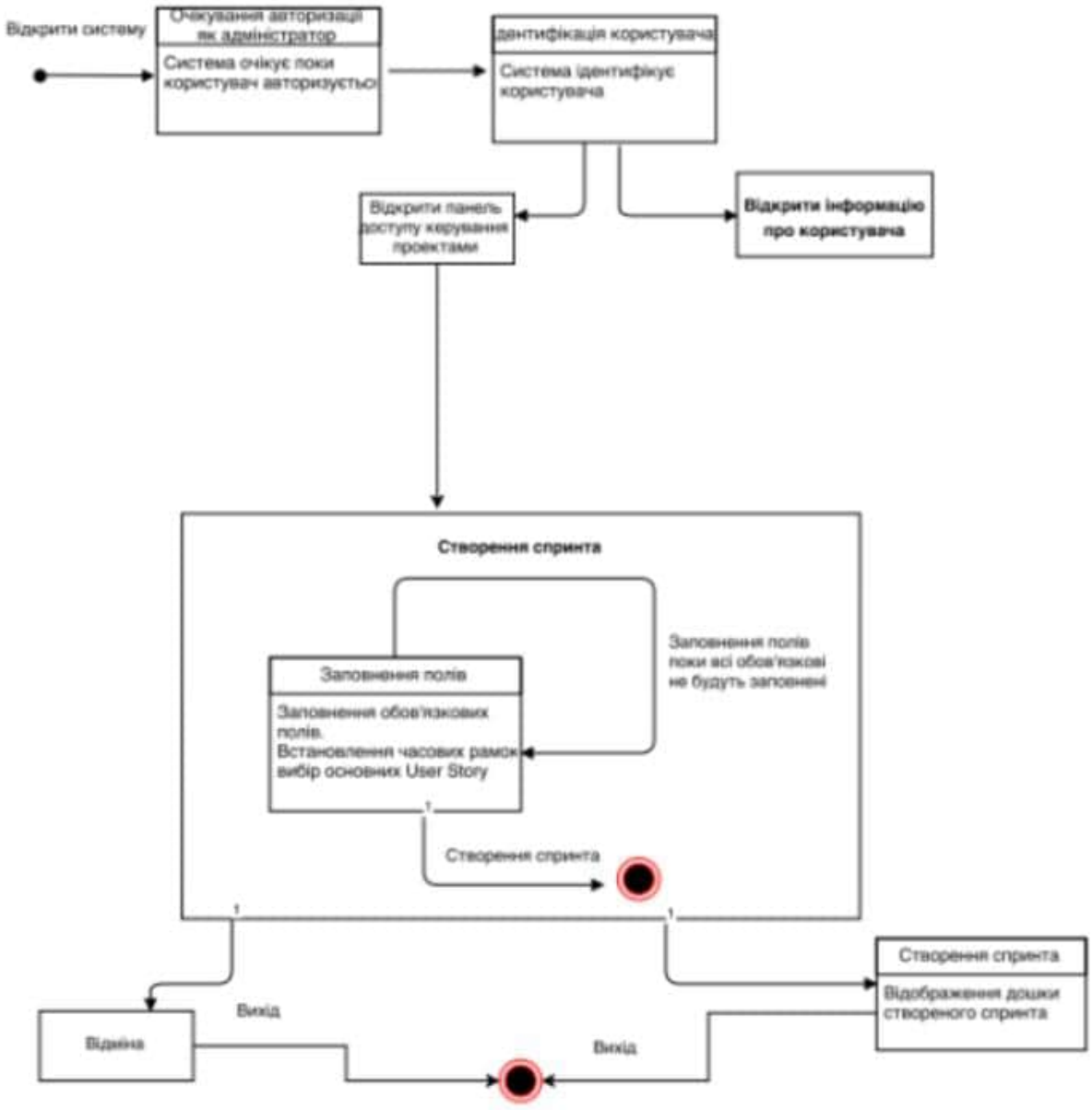

Рис. 2.4 – Діаграма станів для створення спринта

На риснку 2.5 зображена діаграма станів для створення карт. Створити логічне представлення допомагає діаграма класів, на її основі створюється вихідний код описаних класів. Взаємозв’язки між класами та інтерфейсом можна побачити по значкам діаграм.

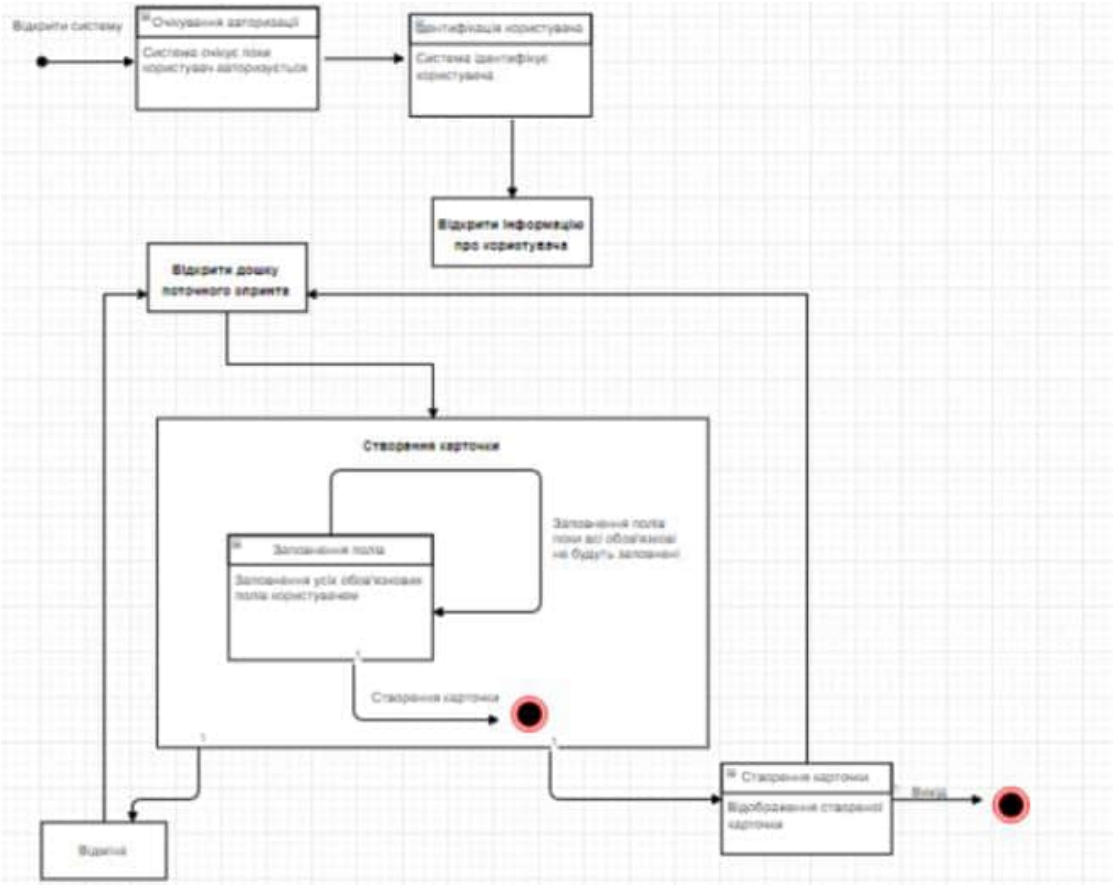

Рис. 2.5 – Діаграма станів для створення картки

2.2 Аналіз існуючих алгоритмів вирішення поставленої задачі

Шаблони проектування програмного забезпечення є ефективними методами вирішення проблем проектування програмного забезпечення. Шаблон не є готовим прикладом, який можна безпосередньо перевести в код програмування. Об’єктно - орієнтований шаблон часто є шаблоном вирішення проблем і відображає зв’язок між класами та об’єктами, не вказуючи, як цей зв’язок буде в кінцевому підсумку реалізований. Патерн проектування в розробці програмного забезпечення - це повторювана архітектурна структура, яка представляє рішення проблеми проектування в звичайному контексті. Об’єктно-орієнтовані шаблони показують зв’язки та взаємодію між класами або об’єктами, не вказуючи, які кінцеві класи або об’єкти програми будуть використані. «Низькорівневі» шаблони, що враховують специфіку конкретної мови програмування, називаються ідіомами. Це хороші дизайнерські рішення, які є специфічними для певної мови чи програмної платформи і тому не є універсальними.

На найвищому рівні є шаблони архітектури, вони охоплюють архітектуру всієї програмної системи. Алгоритми за своєю природою також є шаблонами, але не чернетками, а обчисленнями, оскільки розв’язують комп’ютерні завдання. Існує ще одна група шаблонів проектування, які називаються GRASP (General Responsibility Assignment Software Patterns) - це набір шаблонів проектування, які допомагають визначити та призначити відповідальності об'єктів в програмному проекті. Ці шаблони допомагають створювати гнучкі та масштабовані системи, забезпечуючи високу зрозумілість, незалежність компонентів та відновлюваність.

Основні шаблони GRASP включають:

- Expert (Експерт): Об'єкт, який має на найбільшу інформацію та найкращі можливості для виконання певного завдання, повинен бути призначений відповідальністю за це завдання. Це допомагає забезпечити високу зрозумілість та ефективність системи.
- Creator (Творець): Об'єкт, який відповідає за створення інших об'єктів, повинен бути призначений для цієї відповідальності. Це допомагає забезпечити належне управління створенням об'єктів та уникнути надмірної залежності між компонентами системи.
- Controller (Контролер): Об'єкт, який відповідає за прийняття та обробку вхідних запитів, повинен бути призначений для цієї відповідальності. Це допомагає забезпечити централізоване управління потоком вхідних даних та зменшити залежності між компонентами.
- Low Coupling (Низька залежність): Об'єкти системи повинні бути слабо залежними один від одного. Це досягається шляхом використання абстракцій, інтерфейсів та інших методів, що дозволяють змінювати компоненти системи незалежно один від одного.
- High Cohesion (Висока сполученість): Об'єкти

Шаблони мають ряд переваг перед повністю самостійними дизайнами. Основною перевагою використання шаблонів є зменшення складності розробки завдяки наявності попередньо створених абстракцій для обробки всього класу.

Використання шаблонів концептуально подібне до використання попередньо упакованих бібліотек коду. Правильно сформульований шаблон дизайну дозволяє знайти виграшне рішення і використовувати його знову і знову. Набір шаблонів допоможе розробнику вибрати найбільш підходящий варіант дизайну.Не завжди заміна коду на шаблон є хорошим рішенням. По-перше, сліпе слідування обраному шаблону може призвести до ускладнень програми. По-друге, розробник може захотіти спробувати шаблон у цьому випадку без особливої причини.Багато шаблонів проектування в об'єктно-орієнтованому проектуванні можна розглядати як ідіоматичне відтворення функціональних елементів мови. Легко зрозуміти, що просте визначення шаблону як «стандартного рішення, але не прямого посилання на бібліотеку» по суті означає відмову від повторного використання на користь дублювання.

2.3 Проектування структури бази даних

Останнім кроком перед впровадженням бази даних програмним способом є проектування структури бази даних. На цьому етапі ми визначаємо та описуємо всі поля та зв’язки, які існують у базі даних. Можна використовувати діаграми потоків даних для представлення інформаційних процесів.

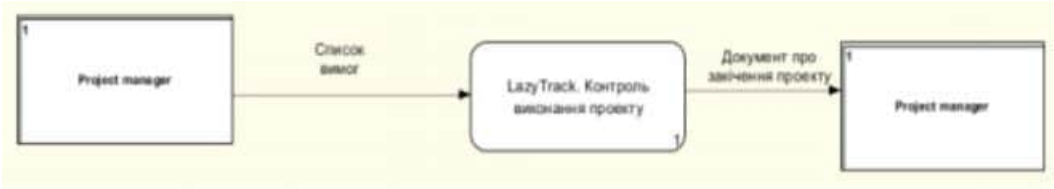

Рис. 2.7 – Діаграма потоків даних

Діаграма декомпозиції зображена на рисунку 2.8

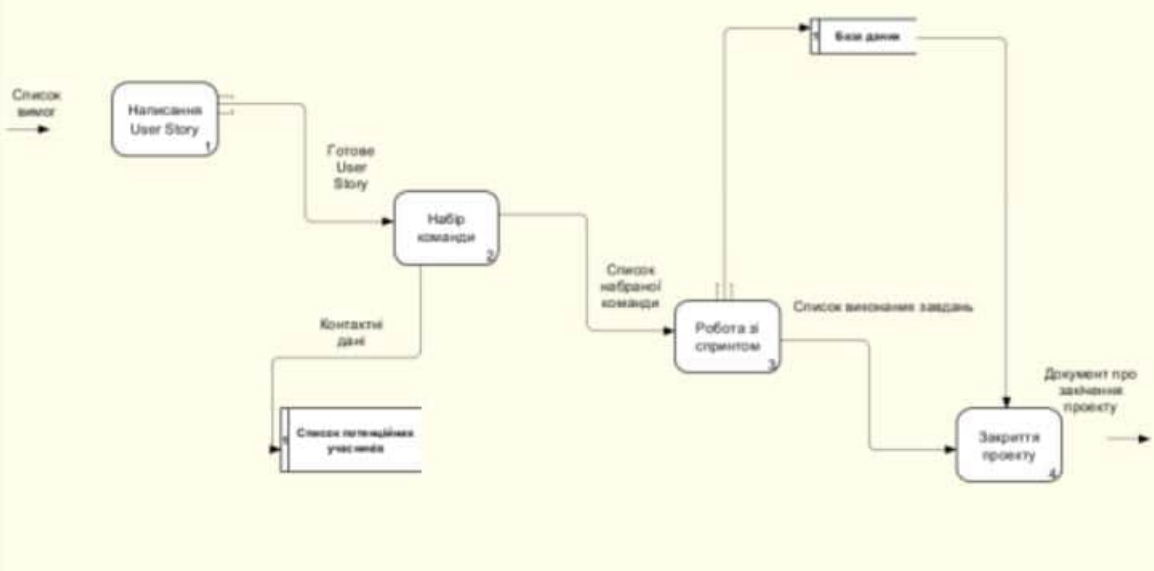

Рисунок 2.8 – Діаграма декомпозиції

На заключному етапі проектування бази даних ми створюємо ER – діаграму.

ER-діаграма (Entity-Relationship diagram) - це графічне представлення структури даних і зв'язків між ними в базі даних. Вона використовується для моделювання сутностей (entities), атрибутів (attributes) та зв'язків (relationships) між ними.

Основні компоненти ER-діаграми включають:

- Сутності (Entities): Сутність - це об'єкт або поняття, про яке зберігається інформація в базі даних. В ER-діаграмі сутності представляються у вигляді прямокутників з назвами, наприклад, "Користувач", "Продукт" або "Замовлення".
- Атрибути (Attributes): Атрибути - це властивості або характеристики сутностей, які описують їх властивості. В ER-діаграмі атрибути зображаються в середині сутностей, наприклад, "Ім'я", "Вік" або "Ціна".
- Зв'язки (Relationships): Зв'язки вказують на взаємозв'язок між сутностями. Вони показують, як сутності пов'язані одна з одною в базі даних. Наприклад, зв'язок "Має" між сутностями "Користувач" і "Замовлення" вказує на те, що користувач може мати декілька замовлень.
- Ключі (Keys): Ключі використовуються для ідентифікації унікальних записів

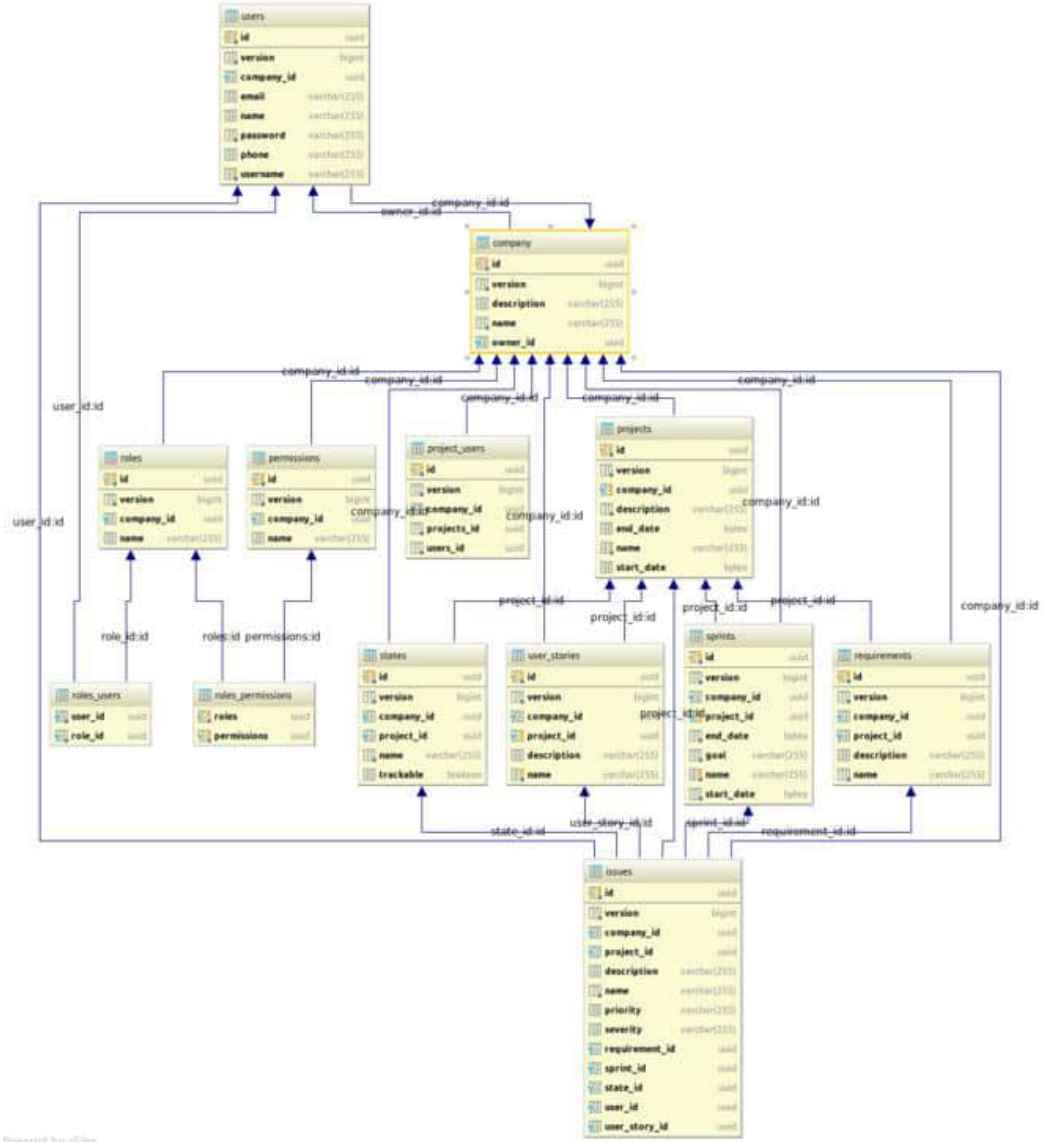

Рис. 2.9 – ER-діаграма

Система, що розробляється, повинна виконувати такі функції:

1. Створювати, редагувати та видаляти нові компанії;
2. Створювати, редагувати та видаляти проекти;
3. Створювати, редагувати та видаляти Sprint;
4. Створювати, редагувати та видаляти картки;
5. Запрошувати, редагувати та видаляти нових користувачів.

Функція «Створення, редагування та видалення нової компанії». Призначення системи – моніторинг виконання проекту. Цей функціонал служить для створення компанії, яка згодом буде власником проекту.

Функція «Створення, редагування та видалення проекту» відповідає за додавання проекту для розробки. Надалі буде контроль за виконанням.

Функція «Створення, редагування та видалення Spring» дозволяє користувачеві додавати період часу, протягом якого виконуватимуться певні завдання.

Функція створення, редагування та видалення карток дозволяє користувачеві додавати картки – завдання, які потрібно виконати. Вкладка також може бути типу «Помилка» або «Історія користувача».

У наведеній нижче таблицій показано відношення між таблицями у базі даних.

Таблиця 2.1 _ Відношення між таблицями у базі даних

Назва таблиці	Тип зв’язку	Відношення
company	one to one	user
user	many to many	user_roles
user	many to one	company
permissions	many to one	company
roles	many to many	permissions
roles	many to one	company
projects	many to one	company

projects	many to many	users

Таблиця 2.2 – Компоненти таблиць та їхні властивості

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Користувач	Ім’я	varchar	20	User name
	Логін	varchar	20	username
	Пароль	varchar	30	password
	Електронна адреса	varchar	50	e-mail
	Номер	varchar	20	phone
Компанія	Назва	varchar	50	Company name
	Опис	varchar	50	description
Проект	Назва	varchar	200	Project name
	Опис	varchar	200	description
	Дата початку	timestamp		start_date
	Дата кінця	timestamp		end_date
Спринт	Назва	varchar	200	sprint name

	Мета	varchar	200	goal
	Дата початку	timestamp		start_date
	Дата кінця	timestamp		end_date
User Story				user_stories
	Назва	varchar	200	name
	Опис	varchar	200	description
Картка	Ім’я	varchar	200	Issues name
	Опис	varchar	200	description
	Пріоритет	varchar	200	priority
	Серйозність	varchar	200	severity

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Програмна реалізація системи

Метою дипломної роботи є розробка застосунку котрий забезпечує моніторинг, комунікацію, актуалізацію даних при виконання IT проектів абсолютно безкоштовним з повним функціоналом та open source.

Об’єктом порівння є програмне забезпечення з використанням таск-трекінових систем, в моєму випадце це Your Track. YouTrack – комерційна система відстеження помилок, програмне забезпечення для управління проектами розроблене компанією JetBrains. Ця система має хорошу швидкодію, зрозумілий інтерфейс та допускає інтеграці з сторонніми продуктами.

З основних недоліків – це ціна (від 200 доларів в залежності від версії), відсутня можливість спілкування з командою за допомогою чату, для забезпечення налагодженого робочого процесу потрібно все ж користуватись іншими засобами.

Для реалізації завдання мною було обрано мову програмування JAVA та фреймворки безпеки Spring. Основними факторами, які вплинули на вибір мови програмування на користь Java стали кросплатформеність, простота, безпека, об’єктна орієнтованість, надійність, багатопоточність та висока продуктивність Java - це об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (згодом придбаною Oracle Corporation). Вона була випущена у 1995 році і стала однією з найпопулярніших мов програмування в індустрії програмного забезпечення. Деякі ключові особливості мови програмування Java:

- Об'єктно-орієнтований підхід: Java побудована на базі об'єктно-орієнтованої парадигми програмування, що дозволяє створювати модульні, масштабовані та повторно використовувані компоненти.
- Платформонезалежність: Java використовує віртуальну машину Java (JVM), яка перетворює інструкції Java у виконуваний код для конкретної операційної системи. Це дозволяє запускати Java-програми на різних платформах без необхідності перекомпіляції.
- Безпека: Java має вбудовану систему безпеки, яка дозволяє управляти доступом до ресурсів системи, обмежувати можливості програм та запобігати вразливостям.
- Велика стандартна бібліотека: Java поставляється з великою стандартною бібліотекою, яка містить готові класи та інструменти для роботи з різними завданнями, включаючи мережеве програмування, роботу з базами даних, роботу з файлами, графічний інтерфейс та інше.
- Многопотоковість: Java підтримує многопотоковість, що дозволяє створювати програми, які можуть виконувати декілька завдань паралельно. Це дозволяє покращити продуктивність і відгук системи.
- Простота використання: Java була спроектована з урахуванням простоти використання та зрозумілості для розробників. Вона має чітку синтаксичну

- структуру та використовує зрозумілі поняття та концепції.

Java застосовується в різних областях, включаючи розробку веб-додатків, мобільних додатків, вбудованих систем, наукових досліджень та багато іншого.

Мова програмування Java була розроблена в 1991 році Джеймсом Гослінгом, Патріком Нотоном, Крісом Вартом, Едом Френком і Майком Шеріданом у Sun Microsystems, Inc. Перша робоча версія проіснувала 18 місяців. Перша назва мови була Oak, але в 1995 році її перейменували на Java. До першої реалізації мови Oak наприкінці 1992 року та публічної презентації розробки Java на початку 1995 року над розробкою працювало багато інших експертів. Зокрема, Білл Джой, Артур ван Хофф і Тім Ліндгольм зробили значний внесок у розробку прототипу ядра Java. Для бек - енд частини виробу використовується фреймворк SPING. Spring - це відкрите програмне забезпечення для розробки та управління додатками на платформі Java. Він надає різні функціональні компоненти і фреймворки, які спрощують розробку веб-додатків та забезпечують інверсію керування (IoC) і аспектно-орієнтоване програмування (AOP). Головною перевагою SPRING є можливість розробляти програму як набір слабко пов’язаних компонентів. Чим менше компоненти програми знають один про одного тим легше розробляти нові функції програми та підтримувати наявні. Класичним прикладом є управління транзакціями. За допомогою SPRING ви можете керувати транзакціями повністю незалежно від основної логіки взаємодії з базою даних. Зміна цієї логіки не порушує можливості транзакцій, так само як зміна логіки керування транзакціями не порушує логіку програми. SPRING заохочує модульність. Компоненти можна додавати і видаляти практично незалежно. В принципі, додаток можна спроектувати таким чином, щоб воно навіть не здогадувалося, що ним керує SPRING. SPRING також значно спрощує модульне тестування: дуже легко ввести фальшиві залежності в компонент, призначений для роботи в контейнері IoC, і перевірити роботу лише цього компонента. Ну, і як приємне доповнення, SPRING значно спрощує ініціалізацію та конфігурацію компонентів програми, дозволяючи гнучко конфігурувати програму без істотних змін у коді Java. Щоб використання програми було безпечним, в роботі використовувався Spring Security.

Spring Security — це платформа Java/JavaEE, яка надає механізми для створення систем автентифікації та авторизації та інших функцій безпеки для додатків, створених за допомогою SRPING Framework. Проект був започаткований Беном Алексом наприкінці 2003 року під назвою «Acegi Security», а перша версія з'явилася в 2004 році. Пізніше проект став офіційним Spring Children's Project. Вперше він був публічно представлений у квітні 2008 року під новою назвою Spring Security 2.0.0.

Розроблене програмне забезпечення дозволяє виконувати наступні функції:

- реєструвати нового користувача;
- реєструвати новий проект;
  задавати кінцеву ціль на новий проект;
- переглядати завдання які поставленні до виконання та вже виконані розробниками;
- вести онлайн спілкування між розробниками

Для зберігання та керування даними використовував систему управління базами даних – PostgreSQL.

PostgreSQL є потужною та розширюваною системою управління базами даних (СУБД), яка використовуєся для зберігання та керування великими обсягами даних. Вона є однією з найпопулярніших відкритих СУБД і має широке застосування в різних сферах, включаючи веб-додатки, аналітику, громадські бази даних та багато іншого.Основні особливості PostgreSQL:

- Розширюваність: PostgreSQL дозволяє розширювати функціональність шляхом створення нових типів даних, операторів, функцій та мов програмування. Це дозволяє розробникам визначати власні доменні моделі та реалізовувати специфічні для своїх потреб функціональність.
- Підтримка стандартів: PostgreSQL підтримує багато стандартів SQL і дотримується ACID (Atomicity, Consistency, Isolation, Durability) принципів для забезпечення надійності транзакцій та цілісності даних.

- Розширені можливості: PostgreSQL надає розширені можливості, такі як географічні об'єкти та геодані, повнотекстовий пошук, розширені можливості реплікації та кластеризації, аналітичні функції та багато іншого.
- Безпека: PostgreSQL має вбудовану систему аутентифікації та авторизації, що дозволяє контролювати доступ до бази даних і забезпечувати безпеку даних. Вона підтримує різні рівні доступу, ролі користувачів та SSL-шифрування для захисту важливої інформації.
- Сумісність: PostgreSQL може взаємодіяти з різними програмними мовами та фреймворками, включаючи Java, Python, Ruby, PHP та інші, завдяки наявності різних драйверів та інтерфейсів.

Загалом, PostgreSQL є потужним і надійним вибором для розробки та управління базами даних, забезпечуючи багатий набір функцій та гнучкість для вирішення різних вимог проектів.

3.2. Програмні модулі системи

3.2.1. Компоненти ПЗ

Цей програмний продукт розроблено на мові програмування Java, а базою даних, яка використовується в розробці, є PostgreSQL. Щоб програма працювала, потрібно встановити ORACLE JRE версії 8 або новішої.

Oracle JRE (Java Runtime Environment) є програмним середовищем, необхідним для виконання Java-додатків. Воно включає в себе виконавчий двигун Java Virtual Machine (JVM), необхідний для інтерпретації та виконання Java-коду.

Основні особливості Oracle JRE:

- Виконання Java-додатків: Oracle JRE дозволяє запускати і виконувати Java-додатки, написані з використанням мови програмування Java. Це включає в себе веб-додатки, десктопні програми, мобільні додатки та інше.
- Java Virtual Machine (JVM): Oracle JRE містить виконавчий двигун JVM, який інтерпретує байт-код Java та виконує його на різних платформах. JVM забезпечує платформонезалежність Java, дозволяючи виконувати Java-додатки на будь-якій підтримуваній платформі.
- Бібліотеки та інструменти: Oracle JRE включає набір стандартних бібліотек Java, які містять готові класи та методи для різних завдань програмування. Воно також постачається з інструментами, такими як Java Development Kit (JDK), для розробки та налагодження Java-додатків.
- Безпека: Oracle JRE надає заходи безпеки, такі як управління правами доступу до ресурсів, перевірка підпису цифрових сертифікатів, контроль доступу до системних ресурсів та інше. Це допомагає забезпечити безпеку виконання Java-додатків та запобігти можливим загрозам безпеки.

Oracle JRE є однією з реалізацій Java Runtime Environment і підтримується компанією Oracle. Варто зазначити, що з моменту мого навчання відомо про перехід Oracle до нової ліцензійної політики, тому рекомендується ознайомитися з

офіційними джерелами Oracle для отримання актуальної інформації щодо ліцензування та підтримки Oracle JRE.

3.2.2. Встановлення П

Отже, для роботи системи необхідно встановити ORACLE JRE версії 8 і вище. Оскільки програма розроблена на мові програмування Java і скомпільована в байт-коді, її можна запускати в будь-якій операційній системі. Я описую встановлення ORACLE JRE на операційних системах, таких як Windows, Linux і MacOC.

Щоб інсталювати JRE у цій операційній системі Windows, вам потрібно:

1.Завантажити ORACLE JRE версії 8 і вище.

2. Відкрити завантажений файл.

3. Натиснути кнопку «Встановити», щоб прийняти умови ліцензійної угоди та продовжити встановлення.

4. Oracle працює з компаніями, які пропонують різні продукти. Під час інсталяції Java вам може бути запропоновано інсталювати такі програми. Переконайтеся, що потрібні програми вибрано, і натисніть «Далі», щоб продовжити встановлення. Опісля застосунок встановиться, діалогове вікно потрібно буде закрити.

Щоб інсталювати ORACLE JRE у цій операційній системі Linux, вам потрібно:

1. Відкрити термінал.
2. Ввести команду «sudo apt-get install default-jre». Після цього встановлюється JRE.
3. З’явиться кілька діалогових вікон із запитом на підтвердження останніх кроків інсталяції. У фінальному діалоговому вікні натисніть кнопку Закрити. Процес встановлення завершено.

Щоб інсталювати JRE у цій операційній системі MacOC, вам потрібно:

1. Завантажити ORACLE JRE версії 8 і вище.

2. Відкрити завантажений файл.

3. Двічі клацнути на піктограму застосунку, щоб запустити майстер встановлення.

4. Майстер встановлення відобразить екран привітання JRE. Натисніть Next (Далі)..

5. Після завершення встановлення з’явиться екран підтвердження.

Натисніть Закрити, щоб завершити встановлення.

Крім того, необхідно встановити систему управління базою даних PostgreSQL.

Щоб встановити цю базу даних на Windows та MacOC, вам потрібно:

1.Завантажити PostgreSQL.

2. Відкрити завантажений файл.

3. Натиснути кнопку «Встановити», щоб прийняти умови ліцензійної угоди та продовжити встановлення. Опісля застосунок встановиться, діалогове вікно потрібно буде закрити.

Щоб встановити цю базу даних на Linux, вам потрібно:

1. Відкрити термінал

2. Виконати команду sudo apt-get install postgresql postgresql-contrib

3. Підключитися до бази даних.

3.2.3 Базові функції ПЗ.

Основним завданням було – розробити застосунок для контролю робочого часу та виконання поставлених задач. Такі застосунки відомі як «Таск – трекінгова система». Таск-трекінгова система (також відома як система керування завданнями або система керування проектами) - це програмне забезпечення, яке використовується для організації, відстеження та керування завданнями та проектами в рамках організації. Таск-трекінгова система дозволяє створювати та призначати завдання, встановлювати їх пріоритети, встановлювати терміни виконання та відстежувати статус виконання кожного завдання. Вона також забезпечує можливість спілкування та співпраці між учасниками проекту, надаючи засоби коментування, відстеження змін та обмін файлами. Деякі основні функції таск-трекінгових систем включають:

- Створення та призначення завдань: Користувачі можуть створювати нові завдання, описувати їх, встановлювати пріоритети та призначати відповідальних осіб.
- Відстеження статусу: Система дозволяє відстежувати поточний статус виконання кожного завдання, відображаючи, наприклад, чи завдання є в роботі, завершене або відкладене.
- Керування термінами: Таск-трекінгова система дозволяє встановлювати терміни виконання для кожного завдання, що допомагає контролювати часові рамки проекту.
- Коментування та спілкування: Користувачі можуть коментувати завдання, обговорювати деталі, задавати питання та обмінюватися інформацією.
- Звітність: Таск-трекінгові системи надають можливість генерувати звіти та діаграми, які демонструють прогрес проекту, виділені ресурси та іншу статистику.

Таск-трекінгові системи допомагають покращити організацію роботи, забезпечуючи централізоване місце для керування та відстеження завдань, спрощуючи спілкування та співпрацю в команді, а також допомагаючи планувати

та контролювати проекти. Вони широко використовуються в різних галузях, таких як розробка програмного забезпечення, маркетинг, проектний менеджмент та інші.

Тому для адміністратора розроблені наступні основні функції:

- авторизація;
- ресєтрація нової компанії;
- реєстрація нового проекту;
- опрацювання проекту (його ціль та опис);
- видалення проекту;
- створення спринту (часовий відрізок певною тривалістю, протягом якого створюється «готовий», тобто придатний до використання й релізу інкремент продукту);
- редагування спринту;
- скасування спринту;
- моніторинг часу;
- використання чату;
- запрошення нового користувача до проекту;

Також розроблені функції для користувача:

- реєстрація;
- авторизація;
- створення спринту (адміністратор – менеджер проекту може задавати завдання та спринт «голові» проекту, а сам «голова», тобто якийсь з користувачів, може задавати завдання та спринти між іншими розробниками);
- редагування спринту;
- скасування спринту;
- відстеження відліку часу;
- використання чату;

Вікно реєстрації користувача та компанії показано на рисунку 3.2

Рис. 3.2 – Вікно реєстрації користувача та компанії

«Register new company owner». В полі «login» вводимо свій логін для подальшого логування в застосунок. В полі «password» вводимо свій пароль. В полі «name» вводи своє справжнє ім’я та прізвище для подальшої легшої комункації між користувачами програми. В поелі «email» вводимо свою електронну пошту. В полі «phone» вводимо свій номер телефону. «Register new company». В полі «name» вводимо назву компанії. В полі «description» вводимо опис компанії, до прикладу чим вона займається та якого роду виконує проекти.

Програмний код даного вікна зображений на рисунку 3.3

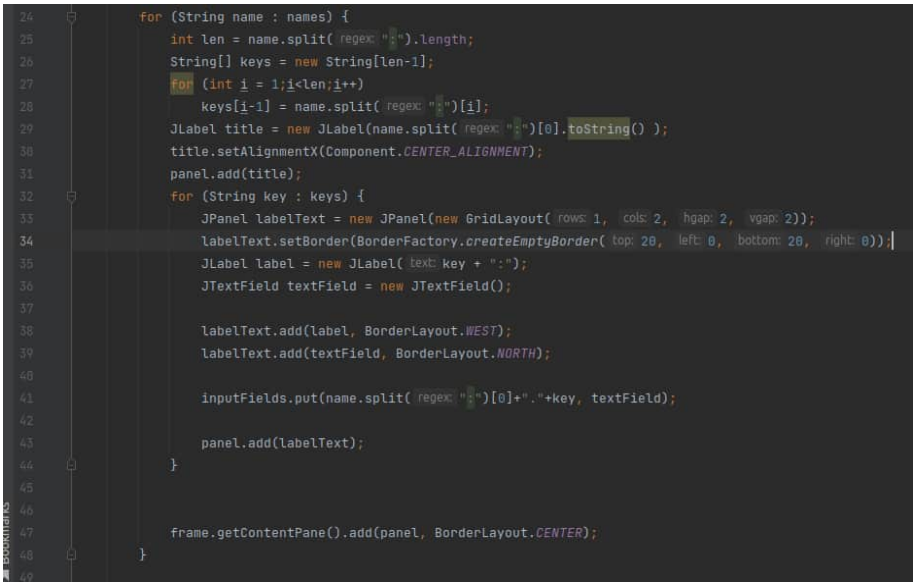

Рис. 3.3 – програмний код вікна реєстрації

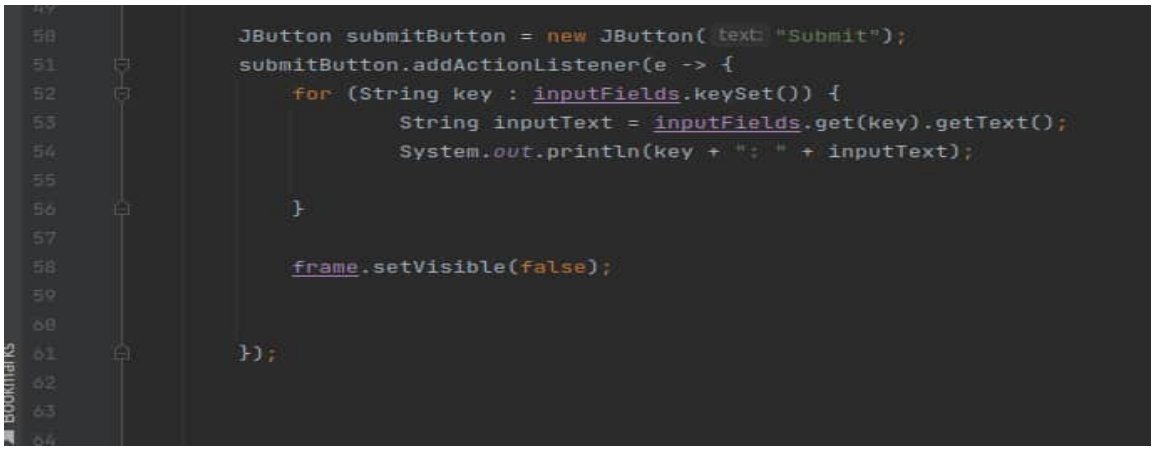

Рис 3.4 – продовження програмного коду вікна реєстрації

Дані стрічки запоневні через цикл for. Java, цикл for є одним з найпоширеніших способів ітерації (повторення) певного блоку коду певну кількість разів. Вона дозволяє виконувати певні дії або обробку для кожного елементу у заданому діапазоні або колекції. Синтаксис циклу for в Java виглядає так:

- ініціалізація: Це початкова інструкція, виконується один раз перед початком циклу. Вона може включати ініціалізацію змінних, встановлення початкових значень ітераторів і т.д.
- умова: Це умова, яка перевіряється перед кожною ітерацією циклу. Якщо умова є істинною, виконується тіло циклу. Якщо умова є хибною, виконання циклу завершується.

- ітерація: Це інструкція, яка виконується після кожної ітерації циклу. Вона зазвичай використовується для зміни значень змінних, що контролюють ітерацію.
- тіло циклу: Це блок коду, який виконується при кожній ітерації циклу. Він містить дії, які ви бажаєте виконати для кожного елементу у діапазоні або колекції.

Вікно авторизації до програмного застосунку зображена на рисунку 3.5

Рис. 3.5 – Вікно авторизації

В полі «login» - вводимо те значення, котре ввели при реєстрації, з полем «password» - аналогічно. Після чого нажимаємо «Login In». Також можемо натиснути «Sign IN» для нової реєстрації.

Рис. 3.6 – програмний код вікна авторизації

```
23
24      JLabel title = new JLabel( text: "Welcome to Lazy Track");
25      title.setBorder(BorderFactory.createEmptyBorder( top: 0,  left: 0,  bottom: 20,  right: 0));
26      title.setFont(new Font(title.getFont().getName(), Font.PLAIN,  size: 23));
27      title.setAlignmentX(Component.CENTER_ALIGNMENT);
28      panel.add(title);
30      for (String key : keys) {
31
32          JLabel label = new JLabel(key);
33          label.setAlignmentX(Component.CENTER_ALIGNMENT);
34
35          JTextField textField = new JTextField();
36
37          panel.add(label);
38          textField.setBorder(BorderFactory.createEmptyBorder( top: 4,  left: 0,  bottom: 5,  right: 0));
39          textField.setMaximumSize(new Dimension( width: 10000,  height: 50));
40          panel.add(textField);
41
42          inputFields.put(key, textField);
43
44
45      }
```

Вікно створення нового проекту зображено на рисунку 3.7

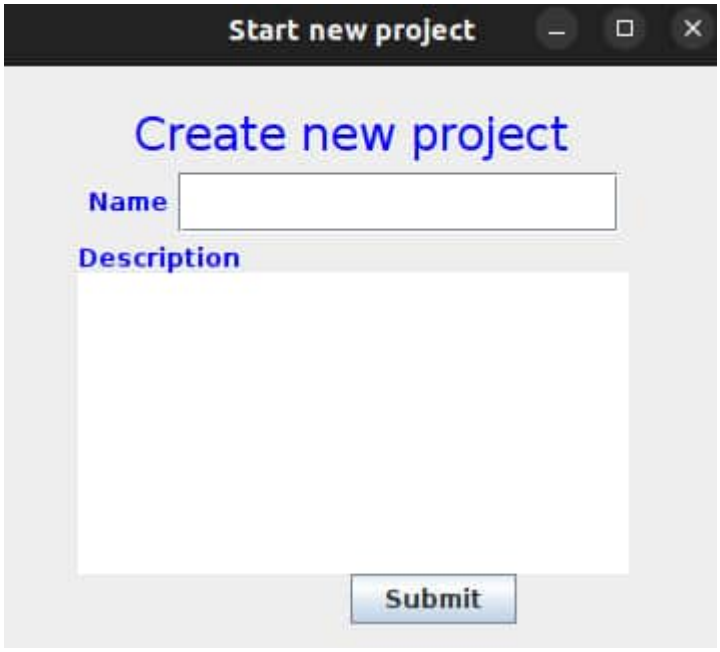

Рис. 3.7 – Вікно створення нового проекту

В полі «Name» - вводимо назву проекту. В полі «desctiption» - вводимо опис проекту.

```
26
27          JPanel labelText = new JPanel();
28          labelText.setBorder(BorderFactory.createEmptyBorder( top: 0,  left: 0,  bottom: 0,  right: 0));
29          JLabel labelN = new JLabel( text: "Name");
30          labelN.setForeground(Color.blue);
31          JTextField textFieldN = new JTextField();
32          textFieldN.setLayout(new BoxLayout(textFieldN, BoxLayout.X_AXIS));
33          textFieldN.setPreferredSize(new Dimension( width: 220,  height: 30));
34          textFieldN.setMaximumSize(new Dimension( width: 220,  height: 30));
35          labelText.add(labelN);
36          labelText.add(textFieldN);
37          panel.add(labelText);
38
```

```
39          JPanel labelTextD = new JPanel(new GridLayout( rows: 1,  cols: 1,  hgap: 0,  vgap: 0));
40          JLabel labelD = new JLabel( text: "Description");
41          labelD.setBorder(BorderFactory.createEmptyBorder( top: 0,  left: 0,  bottom: 0,  right: 0));
42          labelD.setForeground(Color.blue);
43          labelTextD.add(labelD);
44          panel.add(labelTextD);
45
46          JPanel labelTextF = new JPanel(new GridLayout( rows: 1,  cols: 1,  hgap: 0,  vgap: 0));
47          JTextArea textFieldD = new JTextArea( rows: 10,  columns: 10);
48          labelTextF.add(textFieldD);
49          panel.add(labelTextF);
50
```

Рис. 3.8 – Програмний код вікна створення нового проекту

На рисунку 3.9 зображено вікно створення нового спринта.

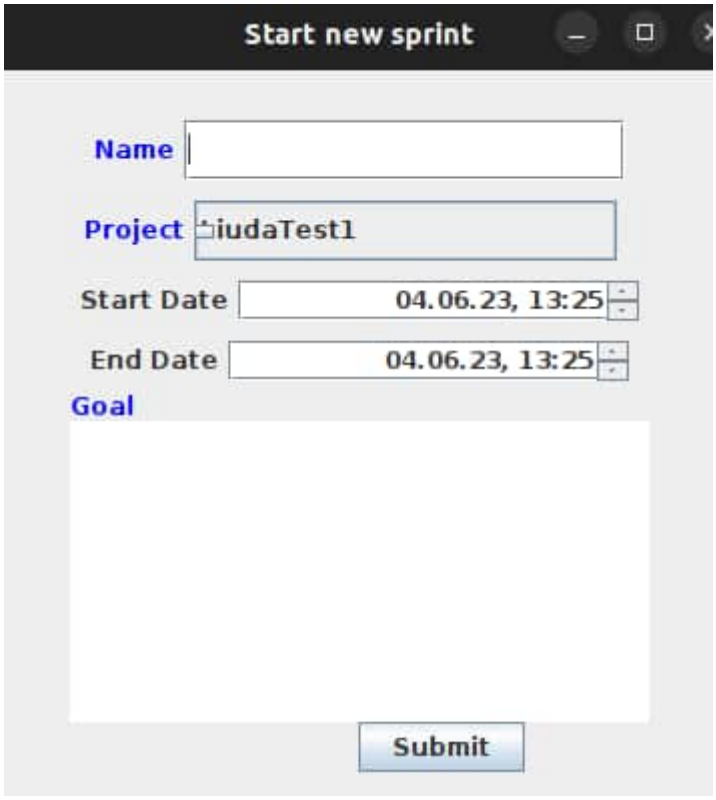

Рис. 3.9 – Вікно створення нового спринта

В полі «name» вводимо назву спринта, можна також назвати як проект. В полі «project» вибираємо зі списку доступних проектів проект, до якого ми створюємо спринт. В полях «start date» та «end date» задаємо дедлайн. В полі «Goal» ставимо завдання до виконнання одному з розробників проекту.

```
45
46      String sb[] = { "LiudaTest1", "LiudaTest2", "LiudaTest3", "LiudaTest4", "LiudaTest5" };
47
48
49      JPanel labelSelect = new JPanel();
50      labelSelect.setBorder(BorderFactory.createEmptyBorder( top: 0     , left: 0,  bottom: 0,  right: 0));
51      JLabel labelS = new JLabel( text: "Project");
52      labelS.setForeground(Color.blue);
53      JComboBox textS = new JComboBox(sb);
54      textS.setLayout(new BoxLayout(textS, BoxLayout.X_AXIS));
55      textS.setPreferredSize(new Dimension( width: 220,  height: 30));
56      textS.setMaximumSize(new Dimension( width: 220,  height: 30));
57      labelSelect.add(labelS);
58      labelSelect.add(textS);
59      panel.add(labelSelect);
60
```

Рис. 4.0 – програмний код вікна створення нового спринта

```
61
62          JPanel lSD = new JPanel();
63          labelSelect.setBorder(BorderFactory.createEmptyBorder( top: 0     , left: 0,  bottom: 0,  right: 0));
64          JLabel labelSD = new JLabel( text: "Start Date");
65          labelS.setForeground(Color.blue);
66          JSpinner panelSD = new JSpinner(dateModel);
67          lSD.add(labelSD);
68          lSD.add(panelSD);
69          panel.add(lSD);
70
71          JPanel lED = new JPanel();
72          labelSelect.setBorder(BorderFactory.createEmptyBorder( top: 0     , left: 0,  bottom: 0,  right: 0));
73          JLabel labelED = new JLabel( text: "End Date");
74          labelS.setForeground(Color.blue);
75          JSpinner panelED = new JSpinner(dateModel);
76          lED.add(labelED);
77          lED.add(panelED);
78          panel.add(lED);
79
80
81
```

Рис 4.1 – продовження програмного коду вікна створення нового проекту

Також був створений для спілкування користувачів між собою.
Чат - це форма електронного спілкування, яка дозволяє користувачам обмінюватися текстовими повідомленнями у режимі реального часу. Чати можуть бути вбудовані у веб-сайти, мобільні додатки, месенджери або спеціалізовані програми. Основні характеристики чату:

- Текстові повідомлення: Користувачі вводять текстові повідомлення, які відображаються для інших учасників чату. Це основний спосіб комунікації у чаті.
- Режим реального часу: Чати надають миттєву доставку повідомлень, що дозволяє користувачам спілкуватися майже без затримок. Це робить чат ефективним для швидкої комунікації.
- Багатокористувацькі розмови: Учасники чату можуть приєднуватися до спільних розмов або створювати приватні чати для обміну повідомленнями з конкретними особами або групами.
- Медіафайли: Деякі чати дозволяють відправляти не лише текстові повідомлення, але й медіафайли, такі як фотографії, відео, аудіо, документи і т.д.
- Емодзі та смайли: Чати можуть підтримувати використання емодзі, смайлів та інших символів для виразності та вираження емоцій.
- Нотифікації: Чати можуть надсилати сповіщення або повідомлення про нові повідомлення, які дозволяють користувачам бути в курсі активності в чаті.

Чати широко використовуються як засіб комунікації в різних сферах, включаючи особисте спілкування, бізнес-комунікацію, технічну підтримку, колективну роботу над проектами, веб-семінари та багато іншого.

На рисунку 4.2 зображений чат.

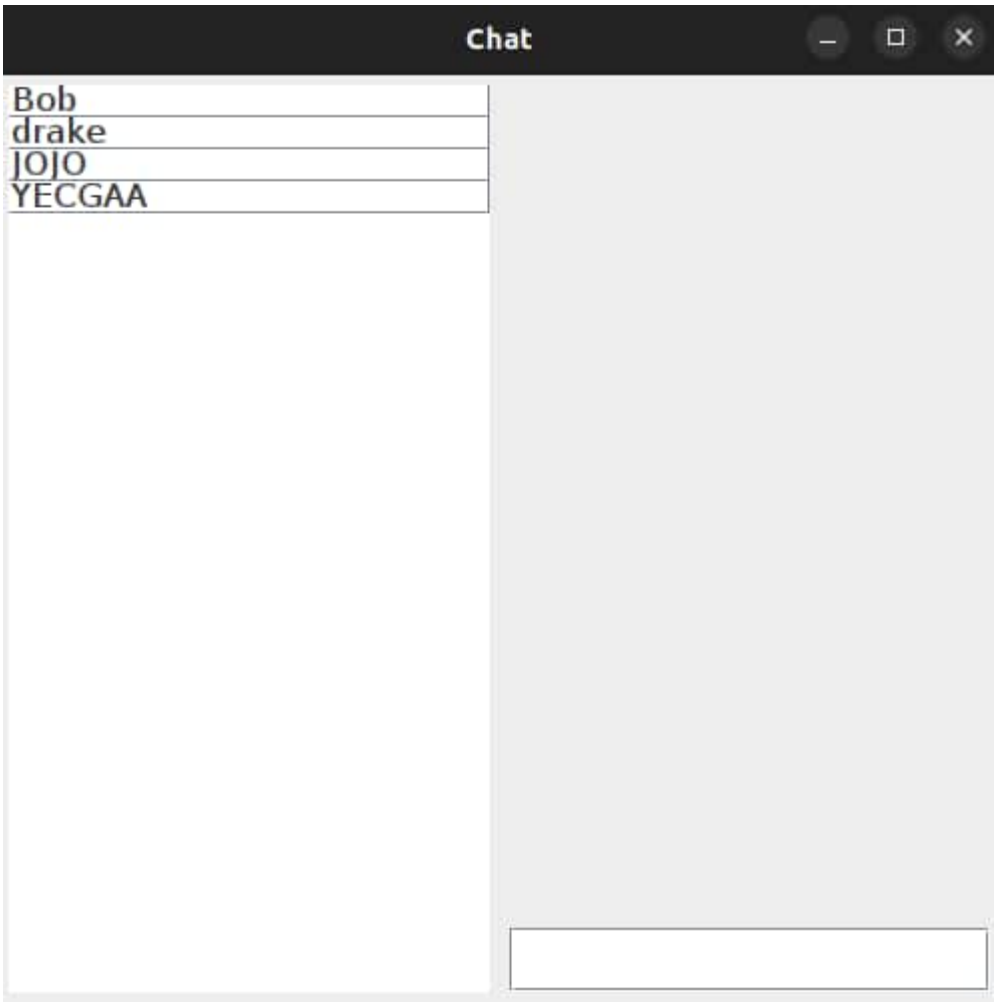

Рис. 4.2 - чат

Зліва зображені користувачі, клацнувший лівою кнопкою миші на яких ми зможемо розпочати спілкування.

Нижче на рисунку 4.3 наведу код , за допомогою якого був створений даний чат.

```
        JPanel root = new JPanel(new GridLayout( rows: 0,  cols: 2));
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(BorderFactory.createEmptyBorder( top: 5,  left: 5,  bottom: 5,  right: 5));
        DefaultTableModel  tableModel=new  DefaultTableModel();
        JTable jTable = new JTable(tableModel);
        jTable.setFont(new Font(jTable.getFont().getName(), Font.PLAIN,  size: 15));
        tableModel.addColumn( columnName: "mes");
        tableModel.addColumn( columnName: "time");
        jTable.getColumn( identifier: "mes").setPreferredWidth(200);


        JTextField labelB = new JTextField("");
        labelB.setFont(new Font(labelB.getFont().getName(), Font.PLAIN,  size: 23));

        labelB.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println(labelB.getText());
                Date nowDate = new Dat
                SimpleDateFormat sdf =
                tableModel.insertRow(t                                    lB.getText().toString(),sdf.format(nowDate)});


                frame.revalidate();
                frame.repaint();
                labelB.setText("");

            }
        });
```

```
JTextField labelB = new JTextField("")
projectcrm
```

```
        DefaultTableModel userList =new  DefaultTableModel();
        JTable usersT = new JTable(userList);
        usersT.setFont(new Font(usersT.getFont().getName(), Font.PLAIN,  size: 17));
        userList.addColumn( columnName: "user");
        String[] users  ={"Bob","drake","JOJO","YEC6AA"};
        for (String user : users)
            userList.insertRow(userList.getRowCount(), new Object[]{user});
        JPanel NN = new JPanel(new BorderLayout());
        NN.add(usersT);

        NN.setBorder(BorderFactory.createEmptyBorder( top: 5,  left: 5,  bottom: 5,  right: 5));
        root.add(NN);
        root.add(panel);
```

Рис. 4.3 – програмний код створення чату спілкування.

На рисунку 4.4 зображене повне діалогове вікно застосунку.

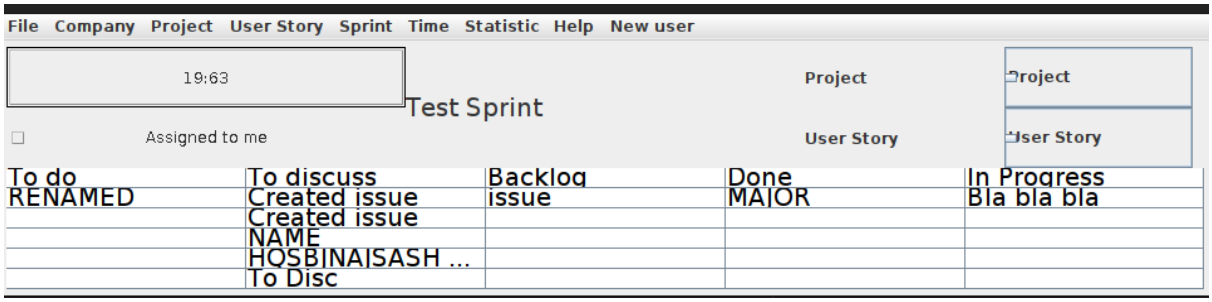

Рис. 4.4 – діалогове вікно застосунку.

3.3 Тестування та верифікація розробленого програмного забезпечення

Були проведені необхідні випробування для перевірки відповідності розробленого продукту вимогам, виявлення та виправлення помилок. На цьому етапі перевірка є дуже важливою, оскільки вона визначає гнучкість системи для конкретного використання. Сьогодні існують різні види тестів, і кожен з них дозволяє перевірити той чи інший аспект продукту, що розробляється.

Були проведені наступні тести та враховані характеристики програмного продукту:

- функціональна проба;

- GUI тести;

- тести безпеки.

3.3.1 Функціональне тестування

Були проведені функціональі тести розробленої системи. Це дозволяє перевірити відповідність вимогам, викладеним у специфікації.

Функціональний тест системи - це вид тестування програмного забезпечення, який перевіряє, чи відповідає система вимогам та специфікаціям функціональності. Його ціль полягає в перевірці правильності роботи окремих функцій, модулів або системи в цілому, забезпечуючи коректність їх виконання та взаємодії.

Основні характеристики функціонального тестування системи:

- Перевірка функцій: Функціональний тест перевіряє, чи працюють функції системи відповідно до вимог та очікувань.
- Вхідні дані та вихідні результати: Тестування перевіряє правильність обробки вхідних даних системою і відповідність очікуваним вихідним результатам.
- Сценарії тестування: Функціональне тестування включає розробку сценаріїв, які відображають типові взаємодії користувача з системою та перевіряють, чи працюють ці взаємодії правильно.

- Покриття функціональності: Тестування ставить за мету перевірити всі функції системи та їх можливі комбінації, а також перевіряє взаємодію між функціями.
- Валідація та верифікація: Функціональне тестування допомагає підтвердити, що система відповідає вимогам і специфікаціям, що були визначені на початку розробки.
- Регресійне тестування: Функціональне тестування включає перевірку системи на наявність відновлення функцій після внесення змін або вдосконалення.
- Функціональне тестування є важливим етапом в процесі розробки програмного забезпечення, оскільки воно дозволяє впевнитися в правильності роботи системи та забезпечити високу якість функціональності для користувачів.

Цей тест показує, що всі вимоги дотримані. Текст функціонального тесту можна знайти в Додатку В

3.3.2 GUI тестування

Будь-яка програма повинна чітко спілкуватися з користувачем, від цього безпосередньо залежить її успіх. Несправна система може легко мати низку проблем. GUI тест (англ. Graphical User Interface test) - це вид тестування програмного забезпечення, спрямований на перевірку коректності та працездатності графічного інтерфейсу користувача (GUI). GUI тести перевіряють функціональність, елементи керування, взаємодію з користувачем, зовнішній вигляд та інші аспекти, пов'язані з графічним інтерфейсом програми.

Основні характеристики GUI тестування:

- Елементи керування: GUI тести перевіряють правильність роботи різних елементів керування, таких як кнопки, меню, текстові поля, радіокнопки, флажки тощо. Вони переконуються, що елементи коректно реагують на дії користувача та виконують передбачені функції.
- Взаємодія з користувачем: GUI тести перевіряють, як користувач може взаємодіяти з програмою через графічний інтерфейс. Вони перевіряють

відповідність очікуваного результату діям користувача, таким як введення даних, натискання кнопок, переміщення елементів тощо.

- Зовнішній вигляд: GUI тести перевіряють правильність відображення елементів на екрані. Вони переконуються, що елементи, шрифти, колірні схеми та інші атрибути графічного інтерфейсу відповідають вимогам та виглядають належним чином.
- Функціональність: GUI тести перевіряють правильність реалізації функціональних вимог через графічний інтерфейс. Вони переконуються, що програма відповідає на правильність введення даних, виконує запити користувача, відображає очікувані результати тощо.
- Регресійне тестування: GUI тести також включають перевірку коректності графічного інтерфейсу після внесення змін або оновлення програмного забезпечення. Вони переконуються, що зміни не порушують працездатність GUI та взаємодію з користувачем.

GUI тести допомагають впевнитися, що програмне забезпечення працює належним чином та забезпечує зручну та коректну взаємодію з користувачем через графічний інтерфейс.

Для проведення тестів графічного інтерфейсу було створено контрольний список, згідно з яким проводилися тести. Контрольний список можна знайти в Додатку В.

За результатами перевірки згідно з контрольним листом дефектів не виявлено було.

3.3.3. Тестування безпеки

Тестування безпеки є дуже важливою частиною будь-якого продукту, що розробляється. Оскільки в системі 2 ролі (клієнт і адміністратор), необхідно забезпечити максимальний захист даних кожного учасника. Для забезпечення захисту основного користувача деякі функції, доступні лише адміністратору, заблоковані. Наприклад, запрошення нового користувача в систему. Це може зробити лише користувач із роллю адміністратора. На рисунку 4.5 показано меню, які доступні лише адміністратору.

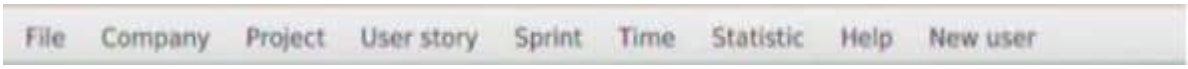

Рисунок 4.5 – Функції, які доступні адміністратору

На рисунку 4.6 зображені меню, які бачить простий користувач.

Рисунок 4.6 – Функції, доступні користувачу

Усі паролі користувачів зберігаються в базні даних у зашифрованому вигляді за допомогою хешу BCrypt. Коли пароль вводиться у форму, він відображається із символом «•», що не дозволяє іншим особам підглядати за паролем. Процес авторизації та автентифікації дуже важливий. Після авторизації всі дані користувача передаються на сервер за допомогою токена. Час життя жетона 2 хвилини. Після закінчення цього часу він буде автоматично відновлений, запобігаючи його використанню різними шкідливими програмами. Це також запобігає втраті даних. Тест безпеки перевіряв можливість доступу до сторінки, доступ до якої має лише адміністратор. Для цього використовувалася програма Postman, перевірка здійснювалася через API. Усі спроби були невдалими, запит було повернуто з кодом HTTP 403, що означає, що доступ було заборонено. Таким чином зловмисник також не може отримати жодної інформації.

ВИСНОВКИ

Результатом роботи є створення нового програмного застосунку – системи контролю робочого часу та виконання поставлених цілей «Lazy_Track». Перед початком роботи було досліджено тему та проаналізовано вже існуючі аналоги даного типу систем, щоб виявити переваги та недоліки таких інструментів та врахувати їх при розробці свого додатку. Також було створено діаграму варіантів використання та дошку сценаріїв. Наступним кроком стало формування функціональних і нефункціональних вимог до майбутнього продукту.

1. Визначено архітектуру програмного продукту. Потім були створені діаграма стану та діаграма класів UML. При роботі з базою даних було створено діаграму ER для PostgreSQL і визначено зв’язки між таблицями.

Останнім кроком на цьому етапі було створення таблиці ідентифікаторів.

2. Додаток створено на мові програмування Java з використанням фреймворку SPRING.

3. Були створенні основні функції застосунку які є доступними для адміністратора та більш скорочені функції для користувача.

4. Проведені функціональні випробування та тести на безпеку, що дуже є важливо.

5. Проведемо порівняння з існуючим аналогом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт системи Bugzilla.Features– Як отримати доступ до ресурсу: https://www.bugzilla.org/features/.

2. Офіційний веб-сайт системи YouTrack – Як отримати доступ до ресурсу: https://www.jetbrains.com/youtrack/.

Джерело: https://uk.wikipedia.org/wiki/%D0%9A%D0%D0%BD%D1%82-%D1%81%D0.

3. Журнал - Як отримати доступ до ресурсу:

https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3.

Офіційний веб-сайт Bugzilla – Як отримати доступ до ресурсу: https://www.bugzilla.org/.

4. Вельма А. М. Вибір системи відстеження помилок залежно від конфігурації програмного забезпечення / А. М. Вельма, Є. Ю. Вельма. Лактіонов // Вісник НТУУ “КПІ”. Містер. : Інформатика, управління та обчислювальна техніка. - 2010 (перегляд).

- Завдання 52. – Ст.137-141

5. Офіційний веб-сайт ScalHive – Як отримати доступ до ресурсу: https://scalhive.com/.

6. Офіційний веб-сайт Spring Securiry Framework – Як отримати доступ до ресурсу: https://projects.spring.io/spring-security/.

7. Що таке JAVA? - https://uk.wikipedia.org/wiki/JavaScript

8. Що таке історії користувачів?– Спосіб доступу

Джерело: https://www.quality-assurance-group.com/user-stories/.

9. Вислобоков В. Що таке PostgreSQL? Для тих хто сумнівається] / Віктор Вислобоков – https://www.postgresql.org/

10. Сидоров К. Про роль PM і управління великими масштабованими проектами / Кирило Сидоров –

https://dou.ua/lenta/articles/scalable-project-management/.

11. Трекер помилок: як вибрати найкращі?

http://qatestlab.com/ru/knowledge-center/QA-Testing-Materials/Bug-Trackers-What-to-Choose/.

12. Як інсталювати Java за допомогою Apt-Get в Ubuntu 16.04 - метод доступу до ресурсу:

https://www.digitalocean.com/community/tutorials/java-apt-get-ubuntu-16-04-ru.

13. Клієнт - Сервер:

https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%E2%80%94_%D1%81%D0% B5%D1%80%D0%B2%D0%B5% 80.

14. Короткий огляд Spring Securit: https://habr.com/post/203318/.

15. Система відстеження помилок:

https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%

16. Система керування базами даних PostgreSQL

http://bourabai.kz/dbt/servers/postgresql.htm

17. SCRUM – ЦЕ ЕФЕКТИВНЕ УПРАВЛІННЯ ПРОЕКТАМИ

https://brainrain.com.ua/%D1%81%D0%BA%D1%80%D0%B0%D0%BC-%D1%8D%D1 %82 %D0%BE.

18. Bugzilla

https://uk.wikipedia.org/wiki/Bugzilla.Вислобоков В. Що таке PostgreSQL? Для тих хто сумнівається! / Виктор Вислобоков

19. Java Runtime Environment

https://uk.wikipedia.org/wiki/Java_Runtime_Environment.

20. Spring Framework

https://uk.wikipedia.org/wiki/Spring_Framework.

21. SQL https://uk.wikipedia.org/wiki/SQL.
22. Task management https://en.wikipedia.org/wiki/Task_management.
23. Tracking system https://en.wikipedia.org/wiki/Tracking_system.
24. YouTrack https://uk.wikipedia.org/wiki/YouTrack.

Додаток А

Вихідний код класів сутностей

Project.java

```
import com.lazytrack.core.entity.CompanyAssociated; import
java.time.LocalDate; import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint; import
lombok.Getter; import lombok.Setter;

@Table(name = "projects",

  uniqueConstraints = @UniqueConstraint(columnNames = {"name", "company_id"}))


public class Project extends CompanyAssociated { @Column(name =
  "name", nullable = false) private String name;


  @Column(name = "description", nullable = false)

  private String description;

  @Column(name = "start_date")

  private LocalDate startDate;

  @Column(name = "end_date")

  private LocalDate endDate;

}
```

Sprint.java

```
import com.fasterxml.jackson.databind.annotation.JsonDeserialize; import
com.fasterxml.jackson.databind.annotation.JsonSerialize; import
com.lazytrack.core.serialization.LocalDateDeserializer; import
com.lazytrack.core.serialization.LocalDateSerializer; import
com.lazytrack.project.entity.ProjectAssociated; import java.time.LocalDate;
```

```
import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.UniqueConstraint; import
lombok.Getter; import lombok.Setter;


@Getter

@Setter

@Entity

@Table(name = "sprints",

  uniqueConstraints = @UniqueConstraint(columnNames = {"name", "project_id"}))

public class Sprint extends ProjectAssociated { @Column(name =
  "name", nullable = false) private String name;

  @Column(name = "goal", nullable = false)

  private String goal;

  @JsonDeserialize(using = LocalDateDeserializer.class)

  @JsonSerialize(using = LocalDateSerializer.class)

  @Column(name = "start_date", nullable = false)

  private LocalDate startDate;

  @JsonDeserialize(using = LocalDateDeserializer.class)

  @JsonSerialize(using = LocalDateSerializer.class)

  @Column(name = "end_date", nullable = false)

  private LocalDate endDate;

}
```

UserStory.java

```
import com.lazytrack.project.entity.ProjectAssociated; import
javax.persistence.Column; import javax.persistence.Entity;


import javax.persistence.Table;
```

```
import javax.persistence.UniqueConstraint;

import javax.validation.constraints.NotNull;

import lombok.Getter;


import lombok.Setter;

@Getter

@Setter

@Entity

@Table(name = "user_stories",

uniqueConstraints = @UniqueConstraint(columnNames = {"name",

"project_id"}))

public class UserStory extends ProjectAssociated {

@NotNull

@Column(name = "name", nullable = false)

private String name;

@Column(name = "description")

private String description;

}
```

Issue.java

```
import com.lazytrack.board.entity.enumeration.Priority; import
com.lazytrack.board.entity.enumeration.Severity; import
com.lazytrack.core.entity.User;

import com.lazytrack.project.entity.ProjectAssociated; import java.util.UUID;

import javax.persistence.Column; import
javax.persistence.Entity; import
javax.persistence.EnumType; import
javax.persistence.Enumerated; import
javax.persistence.ManyToOne;


import javax.validation.constraints.NotNull; @Entity(name =
"issues")
```

```
public class Issue extends ProjectAssociated {

    @NotNull

    @Column(name = "name", nullable = false)

    private String name;

    @Column(name = "description", nullable = false) private String
    description; @Enumerated(EnumType.STRING) private Severity
    severity;

    @Enumerated(EnumType.STRING)

    private Priority priority;

    @ManyToOne

    private Requirement requirement;

    @ManyToOne

    private State state;

    @Column(name = "sprint_id", insertable = false, updatable = false)

    private UUID sprintId;

    @ManyToOne

    private Sprint sprint;

    @ManyToOne

    private UserStory userStory;

    @ManyToOne

    private User user

}
```

Додаток Б

Таблиця Б.1 – Функціональне тестування

Варіанти використання	Тестові випадки	Тестові дані
Реєстрація користувача та компанії в системі	5	10
Авторизація користувача у системі	3	3
Створення проекту	2	4
Редагування проекту	3	4
Видалення проекту	1	-
Створення спринта	3	5
Редагування спринта	3	5
Видалення спринта	1	-

Створення User story	4	7
Редагування User story	5	10
Видалення User story	1	-
Створення карточки	9	30
Редагування карточки	12	45
Користування чатом	4	13
Загалом	60	136

Назва перевірки	Пройшла	Не пройшла
Помилки в функціональності за допомогою інтерфейсу	✓	
Необроблені виключення при взаємодії з інтерфейсом	✓	
Втрата або спотворення даних, що передаються через елементи інтерфейсу	✓	
Помилки в інтерфейсі (невідповідність проектної документації,	✓	

відсутність елементів інтерфейсу)		