МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«___» _____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
СТУПЕНЯ «МАГІСТР»

Тема: Інструменти відслідковування задач та управління проєктами

Виконавець: Олександр БОНДАР

Керівник: Анастасія ВАВІЛЕНКОВА

Нормоконтролер: Євгеній ТУПОТА

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо професійна програма «Системне програмування»

Форма навчання денна

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«_____»_____2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Бондара Олександра Володимировича

1.Тема роботи: «Інструменти відслідковування задач та управління проєктами»

затверджена наказом ректора від «28» серпня 2023 року № 1494 /ст.

2. Термін виконання роботи: з 02.10.2023 до 31.12.2023

3. Вихідні дані до проєкту (роботи): характеристика технології відслідковування задач та управління проєктами, документація *Jira, Trello, C#, MySQL.*

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) основні теоретичні засади управління проєктами;
2) вибір інструментальних засобів розробки;
3) програмна реалізація системи управління проєктами.

5. Перелік обов'язкового графічного матеріалу:

1) організаційна структура програми відсліковування задач;
2) діаграма прецедентів для функцій застосунку;
3) робота програми відслідковування задач та управління проектами;
4) інтерфейс роботи програми відслідковування задач.

6. Календарний план-графік

№ п/п	Етапи виконання кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Вивчення методології управління проєктами	02.10.2023 – 05.10.2023	
2	Визначення програмних продуктів для управління проєктами	11.10.2023 – 13.10.2023	
3	Визначення функціональних можливостей середовища *Jira*	20.10.2023 – 26.10.2023	
4	Вивчення алгоритму роботи з інструментом управління проєктами *Jira*	27.10.2023 – 02.11.2023	
5	Проведення інтегрування методу кластеризації	03.11.2023 – 10.11.2023	
6	Проєктування та розробка структури розробленого сервісу для відслідковування задач та управління проєктом	18.11.2023 – 26.10.2023	
7	Оформлення пояснювальної записки	18.12.2023 –19.12.2023	
8	Підготовка графічного матеріалу	20.12.2023	

7. Дата видачі завдання «02» жовтня 2023 р.

Керівник кваліфікаційної роботи_____Анастасія ВАВІЛЕНКОВА
(підпис)

Завдання прийняв до виконання_____Олександр БОНДАР
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Інструменти відслідковування задач та управління проєктами»: 81 сторінка, 22 рисунки, 7 таблиць, 44 використаних джерела.

УПРАВЛІННЯ ПРОЄКТАМИ, ПРОГРАМНІ ПРОДУКТИ, ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ПРОЄКТІВ.

Об’єктом дослідження даного кваліфікаційної роботи є процес відслідковування задач.

Предметом дослідження кваліфікаційної роботи є технології та інструменти управління проєктами.

Метою дослідження є аналіз технологій відслідковування задач та управління проєктами для виявлення найкращих практик та інструментів управління проєктами. Дослідження спрямоване на встановлення факторів, які впливають на ефективність управління проєктами, і виявлення можливостей для покращення цього процесу.

Було розглянуто ключові характеристики та властивості існуючих рішень, таких як *Jira* та *Microsoft Project,* щоб визначити найбільш важливі аспекти для реалізації власного інструменту. Однією з ключових вимог було забезпечення зручності користування та інтуїтивності графічного інтерфейсу, адаптованого для технічно орієнтованої аудиторії.

Результатом виконання кваліфікаційної роботи є розроблений додаток, управління проєктами та оптимізації поточних завдань, з можливостями його вдосконалення включаючи розширення функціональності для покращення взаємодії користувачів з інструментом, оптимізацію для роботи з великим обсягом даних, та інтеграцію з іншими інструментами розробки для підвищення гнучкості та універсальності.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
РОЗДІЛ 1 ОСНОВНІ ТЕОРЕТИЧНІ ЗАСАДИ УПРАВЛІННЯ ПРОЄКТАМИ 11
1.1. Методології управління проєктами 11
1.2. Програмні продукти для управління проєктами. 29
1.3. Висновки до розділу 32
РОЗДІЛ 2 ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ 34
2.1 Функціональні можливості середовища “*Jira*” 34
2.2 Алгоритм роботи з інструментом управління проєктами “*Jira*” 37
2.3. Інтегрування методу кластеризації в відслідковування задач та управління проєктами 48
2.4. Висновки до розділу 51
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ 53
3.1. Структура розробленого сервісу для відслідковування задач та управління проєктом 53
3.2. Кількісна оцінка параметрів роботи сервісу для відслідковування задач та управління проєктом 60
3.3. Висновки до розділу 75
ВИСНОВКИ 77
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

SQL – *Structured Query Language* (структурована мова запитів)

URL – *Universal Resource Locator* (адреса сторінки в Інтернеті)

UI – *User Interface* (інтерфейс користувача)

UID – *User Interface Design* (дизайн інтерфейсу користувача)

UX – *User Experience* (взаємодія користувача з системою)

АБД – адміністратор баз даних

АІС – автоматизована інформаційна система

ВСТУП

Актуальність теми обумовлена тим, що сучасний світ характеризується зростаючою складністю та конкурентною обстановкою. Організації та команди повинні бути готові швидко реагувати на зміни, швидко розвиватися та досягати своїх цілей в найкоротший термін. У цьому контексті ефективне управління проєктами стає ключовим чинником успіху. Професіонали у сфері управління проєктами мають вміти працювати з обмеженими ресурсами, ризиками та невизначеністю. Дослідження у галузі управління проєктами дозволить розкрити найкращі практики, які допомагають досягати успіху у сучасному бізнесі та професійній діяльності.

Управління проєктами є невід'ємною складовою сучасного бізнесу та технічних проєктів, і його важливість в сучасному світі надзвичайно важлива. Ця сфера діяльності стала ключовою в умовах зростаючого тиску на організації та команди досягати високих результатів і досягати своїх цілей. Управління проєктами дозволяє раціонально розподіляти ресурси, контролювати витрати, зменшувати ризики та помилки, а також поліпшувати співробітництво та комунікацію в команді та замовниками.

Таким чином, дослідження управління проєктами має велике значення для розвитку сучасних організацій та професіоналів, які працюють у цій галузі. Вивчення принципів, методів та інструментів управління проєктами допомагає покращити результативність та досягати успіху у виконанні проєктів будь-якої складності.

Об'єктом дослідження даного кваліфікаційної роботи є процес відслідковування задач.

Предметом дослідження кваліфікаційної роботи є технології та інструменти управління проєктами.

Метою дослідження є аналіз технологій відслідковування задач та управління проєктами для виявлення найкращих практик та інструментів управління проєктами.

Дослідження спрямоване на встановлення факторів, які впливають на ефективність управління проєктами, і виявлення можливостей для покращення цього процесу.

Методи дослідження: Першочерговою задачею було проведення системного аналізу існуючих рішень у галузі управління проєктами для визначення ключових характеристик та недоліків конкурентів.

Для досягнення поставленої мети використовувалися методи комплексного порівняльного аналізу, що дозволив зіставити функціональні можливості, швидкодію та інші технічні аспекти різноманітних інструментів відслідковування задач. Аналіз отриманих даних визначив конкурентні переваги та слабкі сторони існуючих рішень на ринку.

У подальшому, використовуючи методи структурного та об'єктно-орієнтованого проєктування, розроблено власний інструмент для відслідковування завдань та управління проєктами, враховуючи виявлені під час аналізу недоліки існуючих рішень.

На етапі розробки використовувалися методи сучасного програмування, зокрема об'єктно-орієнтованого підходу та принципів *SOLID*, для забезпечення гнучкості та розширюваності створеного програмного продукту. Детальний аналіз вимог та специфікацій системи дозволив ефективно впроваджувати необхідний функціонал та забезпечити його високу якість.

Для перевірки функціональності та визначення продуктивності використовувалися методи тестування, зокрема модульні, інтеграційні та приймальні тести. Автоматизовані тести допомогли виявляти та виправляти помилки на ранніх етапах розробки.

Наукова новизна квалфікаційної роботи полягає у тому, що робота вирізняється новаторським підходом до управління проектами, включаючи технічні та концептуальні нововведення. Проєкт інтегрує передові технології, такі як *DevOps* та *Docker*, для гнучкого розгортання та управління проєктами. Використання методологій *Agile* і *Scrum* підкреслює гнучкість та адаптивність управління завданнями. Застосування алгоритму кластеризації дозволяє аналізувати та прогнозувати динаміку проєктних процесів. Особливість полягає в високій

поєднаності з різними модулями, забезпечуючи інтегровану та узгоджену роботу системи.

Практичне значення отриманих результатів. Кваліфікаційна робота представляє значущий внесок у сферу розробки програмного забезпечення, спрямований на вдосконалення процесів управління проєктами та використання інструментів для відслідковування задач.

Розроблений інструмент, який базується на мові програмування *C#* та використовує базу даних *SQLite* для зберігання інформації, надає користувачам ефективний та інтуїтивно зрозумілий інтерфейс для управління завданнями та проєктами. Враховуючи технічні потреби та специфіку вимог користувачів, інструмент володіє широким набором функціональних можливостей, що спрощує процеси розробки та виконання завдань.

Практичне значення отриманих результатів проявляється в забезпеченні ефективного та систематизованого підходу до управління завданнями в рамках проєктів. Застосування цього інструменту дозволяє збільшити продуктивність команди розробників, деталізувати та структурувати завдання, а також вчасно реагувати на зміни в проєкті. Практичні результати кваліфікаційної роботи дають змогу використовувати отримані дані при проведенні досліджень з теорії управління проєктами, у навчальному процесі фахівців з системного програмування, а також у сферах, пов’язаних з системами для управління проєктами.

Особистий внесок випускника. Всі результати, представлені у кваліфікаційній роботі, отримані випускником особисто. Особливої уваги заслуговує проєктування структури додатку з дотриманням принципів трирівневої архітектури.

Апробація отриманих результатів відбувалася на Міжнародній науково-технічній конференції “Інтелектуальні технології лінгвістичного аналізу”.

Публікації.

Бондар О.В. Інструменти відслідковування задач та управління проєктами: зб. тез доп. Міжнародної науково-технічної конференції “Інтелектуальні технології лінгвістичного аналізу”, 24–25 жовтня 2023р. – Київ, НАУ, 2023. – С. 63.

Прогнозні припущення про розвиток об’єкту та предмету дослідження. Полягають у можливості застосування розробленого інструменту як системи для управління проєктами у різноманітних сферах життєдіяльності, у тому числі авіаційній. Подальший розвиток даного інструменту відкриває перспективи для оптимізації робочих процесів та забезпечення високого ступеня гнучкості в управлінні завданнями. Здатність інтегрувати різні функціональні модулі, такі як фільтрація, сортування, відстеження часу та інші, відкриває нові можливості для розробників та керівників проєктів у досягненні поставлених цілей.

РОЗДІЛ 1
ОСНОВНІ ТЕОРЕТИЧНІ ЗАСАДИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1. Методології управління проєктами

Управління проєктами є необхідною складовою сучасного бізнесу та організаційної діяльності в різних галузях. Цей підхід до планування, виконання та контролю проєктів має велике значення завдяки своїм можливостям підвищити ефективність та організованість робіт, зменшити ризики та помилки, а також покращити комунікацію як всередині команди, так і з замовниками.

У сучасному світі динаміка розвитку та конкурентність на ринках значно зросли, що призвело до того, що підприємства та організації повинні бути гнучкими та швидкими в адаптації до змін [1]. У таких умовах важливо мати чіткий контроль над ресурсами, які витрачаються на проєкти, і забезпечувати якісне виконання завдань в строк. Управління проєктами допомагає організаціям досягати цих цілей.

Однією з основних переваг управління проєктами є можливість ефективного розподілу ресурсів, включаючи гроші, людські ресурси, обладнання і час. Враховуючи скільки проєктів запускається в сучасних компаніях, керування ресурсами стає вкрай важливою задачею. За допомогою правильної методології та інструментів управління, можна підвищити ефективність використання ресурсів та забезпечити виконання всіх проєктів в рамках бюджету.

Помилки та ризики є невід’ємною частиною будь-якого проєкту. Однак, завдяки управлінню проєктами, ці ризики можна ідентифікувати, аналізувати та зменшувати [2]. За допомогою ретельного планування та контролю можна попередити потенційні проблеми та забезпечити безперешкодне виконання проєкту. Крім того, управління проєктами дозволяє реагувати на зміни в середовищі та швидко вносити корективи у план проєкту.

Окрім цього, важливо враховувати фактор комунікації як ключовий елемент управління проєктами. Звітність перед замовниками, ефективний обмін інформацією в команді, а також взаємодія зі зацікавленими сторонами є
критичними для успіху проєкту [2]. Управління проєктами надає інструменти та методи для покращення комунікації та забезпечення розуміння всіма сторонами проєкту.

Вчені, що досліджували проєктування, грали та грають важливу роль у розвитку цієї сфери. Їх внесок у розробку методологій та підходів до управління проєктами, створення інструментів та розширення нашого розуміння процесів проєктування неоціненно.

Серед видатних вчених, які внесли суттєвий внесок у сферу проєктування, варто відзначити таких дослідників, як Генрі Гантт, Вільям Перрі, Фредерік Брукс, Вінсент Серф, Девід Парнас, Томас Гілб.

Генрі Лоренс Гантт – американський інженер, відомий своєю роботою в галузі управління проєктами. Він створив Ганттіву діаграму, яка стала популярним інструментом для візуалізації та контролю проєктів. Гантт також вніс вагомий внесок у розвиток технік проєктного управління та стандартизацію методів планування проєктів.

Вільям Перрі – інженер та менеджер, відомий своєю роботою в галузі управління проєктами та методологією "Критичний шлях". Він розробив метод визначення критичного шляху у проєктах, що дозволяє ідентифікувати найбільш критичні завдання та етапи. Цей метод став популярним інструментом для аналізу та планування проєктів.

Фредерік Брукс – відомий інженер-програміст та автор книги "Міфі про програмне проєктування". Його робота сфокусована на питаннях якості та продуктивності в розробці програмного забезпечення. Брукс вважається автором принципу "Шаблони проєктування", які стали важливим інструментом у розробці програмного забезпечення.

Вінсент Серф – відомий як один з батьків Інтернету та автор технології передачі даних по мережі. Він також активно досліджував проблеми забезпечення

якості великих проєктів, що мають глобальний вплив. Серф вніс вагомий внесок у розробку стандартів та підходів до мережевого проєктування.

Девід Парнас – відомий своєю роботою в області інженерії програмного забезпечення. Він вважається одним із піонерів модульного проєктування та розробив концепцію "Інформаційного приховання", що допомагає розділити систему на незалежні компоненти та модулі. Ця концепція допомагає підвищити стійкість та зручність розробки програмного забезпечення.

Томас Гілб – відомий своєю роботою в галузі екстремального програмування (*Extreme Programming, XP*). Він розробив принципи та методи цього підходу, який спрямований на покращення якості та продуктивності розробки програмного забезпечення. Гілб є прихильником агільних методологій та активно сприяє їх поширенню.

Усі ці вчені внесли вагомий внесок у сферу проєктування та управління проєктами. Їх дослідження та розробки стали основою для багатьох сучасних методологій та інструментів управління проєктами. Вони допомогли вирішувати проблеми, пов’язані з якістю, продуктивністю та ефективністю проєктів у різних галузях.

Значення вчених у сфері проєктування неможливо переоцінити. Їх робота сприяє розвитку нових підходів та методологій, що дозволяють робити проєкти більш успішними та конкурентоспроможними. У сучасному світі важко уявити управління проєктами без врахування внеску цих видатних вчених.

Сьогодні існує безліч методологій та підходів до управління проєктами, які допомагають організаціям досягати поставлених цілей. Традиційні методології, такі як *Waterfall,* передбачають послідовне виконання етапів проєкту. Цей підхід підходить для проєктів з чіткою визначеною метою та статичними вимогами.

Модель "*Waterfall*", також відома як “класична модель проєктування”, є однією з найстаріших та найпоширеніших методологій управління проєктами [3]. Вона була розроблена у 1956 році і з того часу знайшла своє застосування в безлічі галузей, включаючи інженерію, програмне забезпечення, будівництво та інші. Модель "*Waterfall*" представлена на рисунку 1.1.

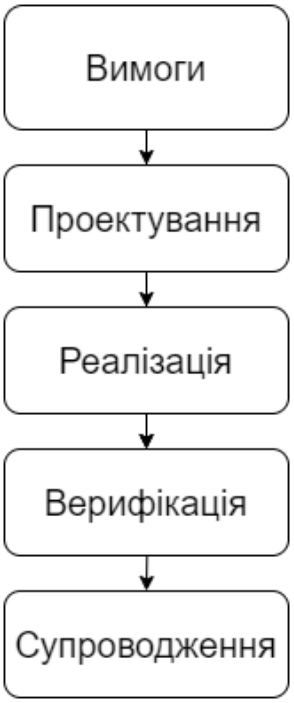

Рис. 1.1. Структура моделі "*Waterfall*"

Основним принципом моделі "*Waterfall*" є послідовне виконання етапів проєкту. Проєкт розбивається на кілька фаз, і кожна фаза розпочинається тільки після завершення попередньої. Основні етапи моделі "*Waterfall*" включають:

Аналіз вимог: На цьому етапі визначаються всі необхідні вимоги до проєкту. Це включає в себе ідентифікацію завдань, обсягу робіт, термінів, бюджету та інших параметрів.

Проєктування: На даному етапі розробляються технічні специфікації та плани виконання проєкту. Це може включати в себе проєктування системи, створення схем, архітектури та інших технічних деталей.

Розробка: Після завершення фази проєктування, розпочинається робота над реалізацією проєкту. Програмісти та розробники створюють код, реалізують функціональність та виконують інші завдання.

Тестування: Після завершення розробки проєкту, він піддається тестуванню. В цьому етапі перевіряється, чи відповідає продукт вимогам, виявляються та виправляються помилки.

Впровадження та підтримка: Останній етап включає в себе впровадження розробленого продукту та надання підтримки користувачам.

Переваги моделі "*Waterfall*":

– простота та легкість в розумінні: Модель "*Waterfall*" є дуже простою та легкою в розумінні. Її можна використовувати в різних галузях без великих зусиль у навчанні персоналу;

– чіткість та структурованість: Кожен етап проєкту має чітко визначені цілі та завдання, що сприяє легкому керуванню та контролю;

– ефективність при статичних вимогах: Модель "*Waterfall*" ідеально підходить для проєктів з фіксованими та стабільними вимогами, де зміни рідко виникають.

Недоліки моделі "*Waterfall*":

– неможливість адаптації до змін: Модель "*Waterfall*" не передбачає можливості адаптуватися до змін у процесі виконання проєкту. Це може призвести до проблем, коли вимоги змінюються;

– довгі та дорогі процеси: Послідовність фаз моделі "*Waterfall*" може призвести до тривалих процесів та збільшення витрат на проєкт;

– низька якість на ранніх етапах: В разі виявлення помилок на пізніших етапах, виправлення може бути вкрай складним та дорогим.

Модель "*Waterfall*" найчастіше використовується в проєктах, де вимоги стабільні та невелика можливість змін. Це може включати в себе розробку програмного забезпечення, будівництво інженерних споруд, виробництво та інші галузі.

Модель "*Waterfall*" є однією з найбільш відомих та використовуваних методологій управління проєктами [3]. Вона відома своєю структурованістю та послідовністю, а також відома своєю непридатністю до змін у вимогах. Хоча вона може бути ефективною для певних видів проєктів, важливо розуміти її обмеження та недоліки. В сучасному світі, де зміни швидко відбуваються, модель "*Waterfall*" може вимагати адаптації або використання інших методологій управління проєктами для досягнення успіху в проєктах.

Однак, в умовах зростаючої нестабільності та змін, *Agile* методології стають дедалі популярнішими. Найпоширеніші з них - *Scrum* і *Kanban*. *Scrum* передбачає роботу над проєктом у коротких ітераціях (спринтах), під час яких розробляється

певна функціональність. Кожен спринт завершується презентацією готового результату, що дозволяє швидко змінювати пріоритети та враховувати зміни.

Scrum є однією з найпопулярніших методологій розробки програмного забезпечення та управління проєктами (рис. 1.2). Вона була розроблена як підхід до виконання проєктів, які вимагають високої гнучкості, швидкості та здатності до адаптації до змін. *Scrum* базується на принципах *Agile,* які включають у себе ітераційний підхід до розробки та спрямованість на задоволення потреб клієнта. Ця методологія виникла в 1980-х роках, але набула популярності в останні десятиріччя.

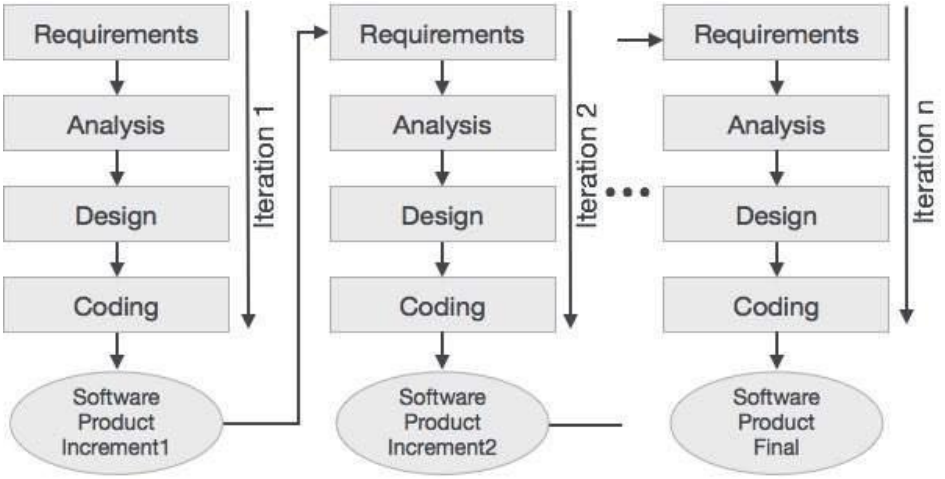

Рис. 1.2 Методологія *Scrum*

Однією з ключових ідей *Scrum* є розподіл проєкту на короткі ітерації, які називаються спринтами [4]. Кожен спринт має фіксований термін, який зазвичай триває від двох до чотирьох тижнів. Під час кожного спринту розробники працюють над визначеною функціональністю та завершують її в кінці спринту. Це дозволяє швидко створювати робочий продукт та забезпечувати регулярні демонстрації замовнику.

Іншою важливою складовою *Scrum* є ролі та відповідальності учасників команди. Тут існують такі основні ролі: *Scrum Master, Product Owner ma Development Team. Scrum Master* відповідає за забезпечення правильної реалізації методології *Scrum* в команді. *Product Owner* визначає вимоги до продукту та пріоритети роботи над ним. *Development Team* виконує роботу над функціональністю та відповідає за якість роботи.

Один із основних артефактів *Scrum* – це *Product Backlog*. Це список всіх функцій та задач, які потрібно виконати для створення продукту. *Product Backlog* постійно оновлюється та пріоритизується *Product Owner*. На початку кожного спринту вибираються певні елементи з *Product Backlog* і додаються в *Sprint Backlog*. Це стає основою для роботи команди протягом спринту.

Ще однією важливою частиною *Scrum* є регулярні зустрічі та облік часу. Денні *Scrum* зустрічі (*Daily Standup*) дозволяють членам команди взаємодіяти, обговорювати свій прогрес та визначати завдання на кожен день. Спринт-планування (*Sprint Planning*) проводиться на початку кожного спринту і включає вибір завдань для наступного спринту. *Sprint Review* проводиться в кінці спринту для демонстрації результатів та отримання відгуків від замовника. *Sprint Retrospective* дозволяє команді аналізувати свою роботу та виявляти можливі покращення.

Scrum дозволяє створювати програмне забезпечення, яке відповідає потребам замовника та змінюється разом із змінами в оточуючому середовищі. Використання цієї методології допомагає зменшити ризики, підвищити продуктивність та покращити співпрацю в команді. *Scrum* використовується в багатьох галузях, а не тільки в розробці програмного забезпечення, і стає дедалі популярнішим підходом до управління проєктами.

Kanban, з іншого боку, базується на візуалізації процесу розробки та обмеженні роботи в процесі, щоб збільшити продуктивність та видаляти завдання з черги на обробку (рис. 1.3).

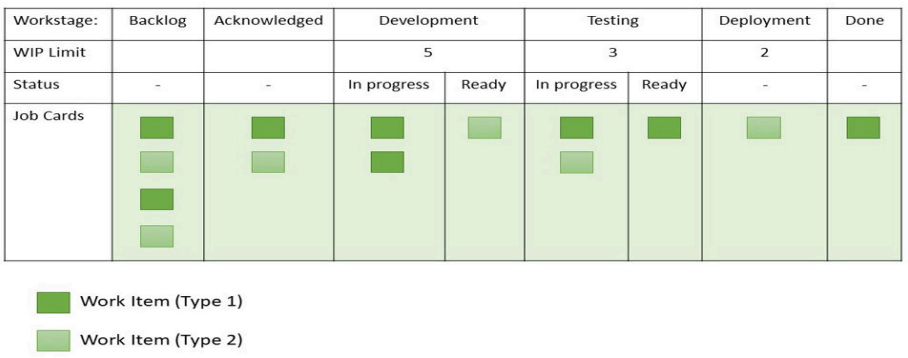

Рис. 1.3. Модель *Kanban*

Канбан – це система управління, що має коріння в японському методі оптимізації робочих процесів [4]. Слово "*Kanban*" перекладається як "дошка" або "таблиця". Вона розроблена в рамках Тойота Продакшн Систем (*Toyota Production System*) для підвищення продуктивності і зниження запасів в автомобільному виробництві, але згодом була успішно впроваджена в інших сферах.

Основний принцип *Kanban* – візуалізація робочого процесу. Для цього використовується дошка з канбанами – візуальними картками, які представляють завдання. Кожен канбан містить інформацію про завдання, таку як опис, пріоритет, термін виконання і інші параметри. Вони розміщені на дошці в певному порядку, що відображає поточний стан робочого процесу.

Однією з ключових ідей *Kanban* є обмеження робочого процесу – кількість завдань, які можуть перебувати на кожному етапі. Це допомагає уникнути перевантаження і збільшити продуктивність.

Система *Kanban* включає такі принципи:

– візуалізація робочого процесу;

– обмеження робочого процесу;

– керування потоком роботи;

– покращення системи;

– підтримка визначеного ритму роботи.

Переваги *Kanban* включають покращення видимості робочого процесу, мінімізацію витрат, легку адаптацію, зниження ризику, підвищення продуктивності і простоту впровадження.

Система *Kanban* застосовується в різних галузях, включаючи виробництво, розробку програмного забезпечення, маркетинг та обслуговування. Вона використовується великими корпораціями та малими компаніями та знайшла застосування в різних проєктах.

В табл. 1.1 продемонстровано порівняння різних методов реалізації та управління проєктами.

Таблиця 1.1

Порівняння проєктів

Метод	Переваги	Недоліки
Традиційний метод	Простий у використанні, не вимагає спеціального навчання	Негнучкий, не дозволяє ефективно керувати великими проєктами
Agile-методології	Гнучкі, дозволяють адаптуватися до змін, ефективні для великих проєктів	Можуть бути складними у використанні, вимагають спеціального навчання
Waterfall-методологія	Ефективний для невеликих проєктів, простий у використанні	Негнучкий, не дозволяє адаптуватися до змін
Scrum	Гнучкий, ефективний для великих проєктів, дозволяє залучати користувачів до розробки	Може бути складним у використанні, вимагає спеціального навчання
Kanban	Гнучкий, ефективний для великих проєктів, простий у використанні	Не дозволяє ефективно керувати складними проєктами

Крім цих основних напрямків, існує безліч інших методологій, таких як *Lean, PRINCE2, і Xtreme Programming (XP),* які можуть бути використані в залежності від специфіки проєкту та завдань. *Lean,* який також відомий як "Лейн-виробництво" або "Лейн-менеджмент", є стратегією оптимізації виробництва та управління процесами [5]. Цей підхід має глибокі коріння в автомобільній промисловості та був спочатку розроблений японськими компаніями, зокрема *Toyota,* в середині 20-го століття. *Lean* став важливим компонентом управління бізнесом у багатьох галузях та організаціях. Методологія *Lean* представлена на рис. 1.4.

Розглянемо *Lean* як стратегію оптимізації виробництва та управління, розглянемо його ключові принципи та методи, і детально розглянемо, як *Lean* допомагає підвищити ефективність, знизити витрати та поліпшити якість виробництва.

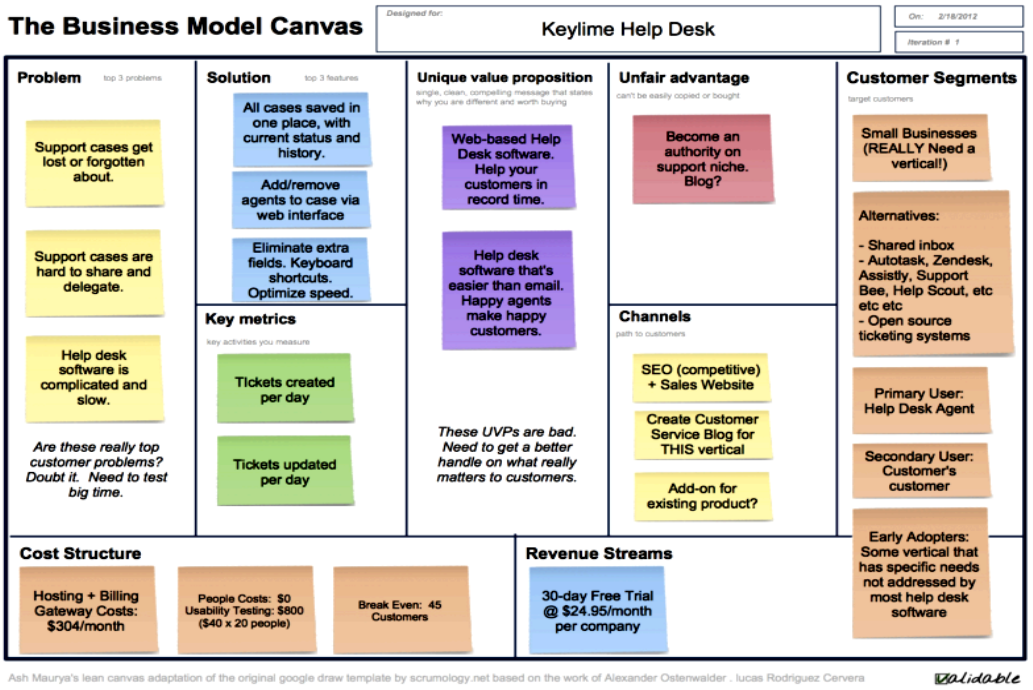

Рис. 1.4. Модель *Lean* [5]

Однією з основних ідей *Lean* є концепція мінімізації витрат та максимізації значення для клієнта. У той час як багато традиційних систем виробництва спрямовані на максимізацію обсягів виробництва, *Lean* ставить перед собою завдання досягнення максимальної продуктивності при мінімальних витратах.

Одним з ключових принципів *Lean* є ідентифікація та видалення всіх видів втрат. Це включає в себе втрати часу, матеріалів, ресурсів, і навіть втрати, які пов’язані з низькою якістю або дефектами. *Lean* надає особливий акцент на оптимізацію робочих процесів, а також на видаленні будь-яких дій або етапів, які не додають цінності продукту.

Ще одним важливим аспектом *Lean* є створення систеи управління, спрямованої на саморегулювання співробітників. Ця система дозволяє працівникам брати активну участь у виробництві та управлінні процесами. *Lean* надає працівникам право зупинити виробництво, якщо вони виявляють проблеми або відхилення від стандартів якості. Це стимулює відповідальність та власництво за результатами.

Крім того, *Lean* акцентує на впровадження системи "*Just-in-Time*", передбачає, ресурси подаються на виробництво в той момент, коли потрібні. Це дозволяє зменшити запаси та витрати на зберігання, а також сприяє зменшенню часу проходження продукції від початку до кінця процесу.

Один із найважливіших інструментів *Lean* – це "5*S*." Це п'ять основних принципів для організації робочого місця: сортування, систематизація, прибирання, стандартизація та самодисципліна. Ці принципи допомагають створити чисте та організоване робоче середовище, де легко виявляти будь-які аномалії та втрати.

Дослідники та практики активно застосовують *Lean* у різних сферах, включаючи виробництво, послуги, логістику та інші. У виробництві *Lean* дозволяє покращити продуктивність, знизити витрати на виробництво та покращити якість продукції. У послуговому секторі *Lean* допомагає оптимізувати робочі процеси та поліпшити обслуговування клієнтів.

Lean також активно використовується у логістиці для оптимізації поставок та управління запасами. Великі компанії, такі як *Toyota,* здійснюють постійну оптимізацію своїх логістичних ланцюгів за допомогою *Lean*.

Однією з особливостей *Lean* є його постійна орієнтація на пошук можливостей для вдосконалення. *Lean* надає підприємствам інструменти для постійного моніторингу та аналізу їхніх процесів, а також для виявлення можливостей для оптимізації та вдосконалення.

Слід зазначити, що *Lean* є потужним і ефективним підходом до управління бізнесом та оптимізації виробництва. Він дозволяє підприємствам підвищити ефективність, знизити витрати та поліпшити якість продукції. Його принципи та методи можуть бути застосовані у різних сферах та галузях, і вони продовжують розвиватися та адаптуватися до сучасних викликів та можливостей. Усе це робить *Lean* надзвичайно актуальним та важливим інструментом для бізнесу сьогодні.

PRINCE2, що розшифровується як "*PRojects IN Controlled Environments*", представляє собою широко використовувану методологію управління проєктами, яка була розроблена в Великобританії і вперше опублікована у 1996 році (рис. 1.5).

PRINCE2 став однією з найбільш популярних систем управління проєктами та знайшов застосування в різних галузях та організаціях по всьому світу.

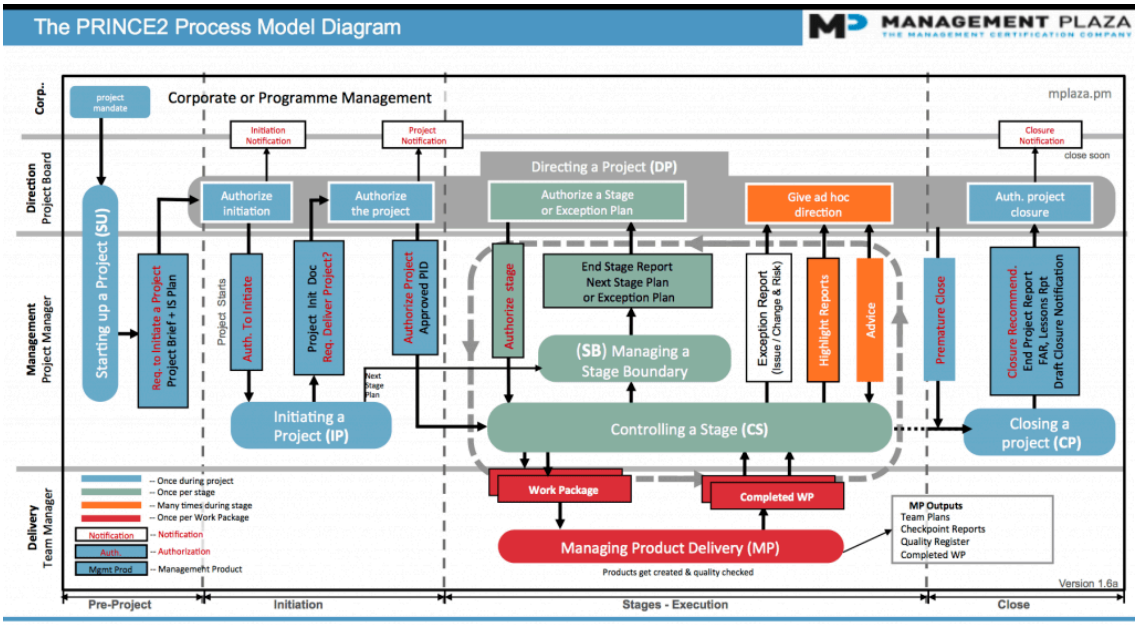

Рис. 1.5. Модель *PRINCE2*

PRINCE2 відомий своєю структурованістю та підходом до управління проєктом, який базується на наборі чітких принципів та процедур. Ця методологія розглядає проєкт як тимчасову організацію, яка створюється для досягнення конкретної мети, і надає інструменти та процедури для керування кожним етапом проєкту, починаючи від ініціалізації і закінчуючи закриттям проєкту.

Однією з ключових особливостей *PRINCE2* є його адаптивність. Ця методологія може бути успішно застосована для різних типів проєктів, від малих і короткострокових до великих та складних. *PRINCE2* може бути застосований в будь-якій галузі та організації, що робить його надзвичайно універсальним і привабливим для багатьох фахівців з управління проєктами.

PRINCE2 складається з набору засад, процесів, ролей та шаблонів, які спрямовані на покращення керування проєктом та забезпечення досягнення поставлених цілей [5]. Однією з ключових засад *PRINCE2* є "бізнес-мотивація", яка визначає, що будь-який проєкт повинен мати чітко визначену бізнес-мету та відповідати стратегічним цілям організації.

Процеси *PRINCE2* передбачають системний аналіз ризиків, постійний моніторинг проєкту та звітування перед вищими органами. Ця методологія надає інструменти для забезпечення якості продукції та досягнення високих стандартів.

У *PRINCE2* роль проєктного менеджера відіграється "керівником проєкту". Він або вона відповідає за планування, виконання та контроль проєкту, а також за комунікацію з замовником та іншими учасниками проєкту. Проєктний менеджер виконує роль лідера команди та забезпечує співпрацю всіх учасників.

PRINCE2 також встановлює чіткі процедури для прийняття рішень, що дозволяє уникнути затягування та ускладнення управління. Процес прийняття рішень включає в себе визначення альтернатив, оцінку ризиків та вибір оптимального варіанту.

Крім того, *PRINCE2* надає інструменти для ефективного управління комунікацією. Звітність перед замовниками, учасниками команди та іншими стейкхолдерами ретельно регламентується та забезпечується за допомогою визначених процедур та шаблонів.

Важливою частиною *PRINCE2* є засоби контролю, які допомагають виявляти зміни та адаптувати плани до нових умов. *PRINCE2* передбачає регулярну оцінку проєкту та аналіз результатів для прийняття необхідних коректив.

Ця методологія також ставить особливий акцент на роботу з ресурсами та їх ефективну розподілу. *PRINCE2* надає інструменти для оцінки необхідних ресурсів та забезпечення їх правильного використання.

PRINCE2 є відкритою методологією, що означає, що вона може бути адаптована до потреб конкретного проєкту або організації. Існує безліч навчальних матеріалів та сертифікаційних програм, що дозволяють фахівцям навчитися та використовувати *PRINCE*2 для успішного управління проєктами.

Завдяки своїй універсальності, структурованості та увазі до деталей, *PRINCE2* залишається однією з провідних методологій управління проєктами, яка допомагає організаціям досягати своїх цілей та реалізовувати проєкти з високою якістю та ефективністю.

Extreme Programming, скорочено *XP,* представляє собою важливий підхід до розробки програмного забезпечення, який виник в 1990-х роках і досі користується значною популярністю серед розробників і фахівців у галузі IT (рис. 1.6). Ця методологія розробки ПЗ є особливою в своєму роді і надає значний акцент на ефективну комунікацію, спільну роботу команди, тестування та гнучкість в процесі розробки.

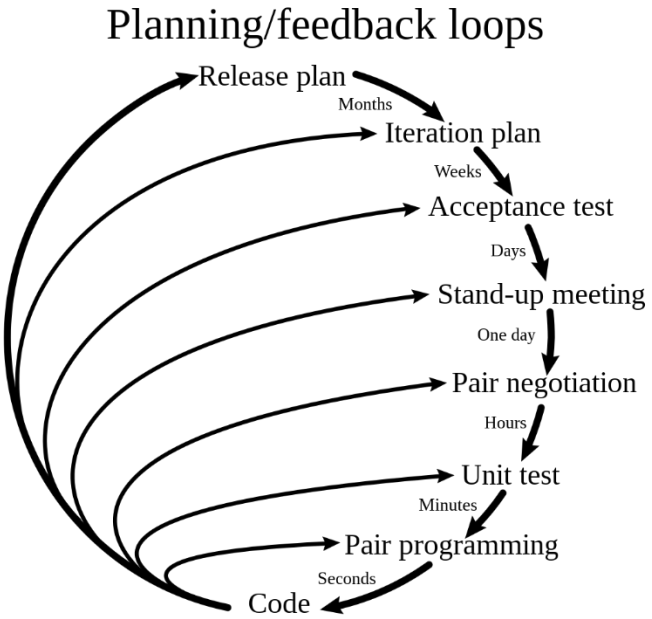

Рис. 1.6. Модель *Extreme Programming*

Однією з ключових особливостей *Extreme Programming* є унікальний підхід до власності коду. За цією методологією весь код вважається спільною власністю всієї команди розробників. Кожен член команди має право вносити зміни в будь-яку частину коду, що сприяє співпраці і взаєморозумінню між розробниками.

Другим ключовим аспектом *XP* є гнучкість у відношенні до змін вимог до програмного забезпечення. Методологія розробки визнає, що вимоги можуть змінюватися в процесі роботи над проєктом, і це повинно бути ретельно враховано. Замість того, щоб опиратися змінам, *XP* активно підтримує процес адаптації до нових обставин і вносить необхідні корективи в код.

Тестування є ще однією ключовою складовою *Extreme Programming*. За цією методологією кожен компонент програми повинен бути покритий тестами, що

дозволяє виявити та виправити помилки на ранніх етапах розробки. Тестування допомагає забезпечити високу якість коду і функціональність програмного продукту.

Ще однією важливою практикою *XP* є парне програмування. Це означає, що два розробники працюють разом над однією задачею, що сприяє якості коду і взаєморозумінню в команді. Парне програмування дозволяє виявляти помилки та проблеми швидше і вирішувати їх ефективніше.

Однією з основних ідей *XP* є також ідея спільної робочої обстановки. У команди, що практикує *XP,* розробники працюють в одному великому приміщенні, що сприяє легшій комунікації, обміну ідеями та вирішенню проблем. Гнучкість графіку є ще однією важливою частиною *XP*. Розробники працюють у коротких ітераціях, зазвичай тривалістю від одного до трьох тижнів. Це дозволяє замовнику бачити результати роботи команди швидко і вносити необхідні зміни в вимоги на кожному етапі розробки.

Принцип простоти є ще однією характеристикою *XP*. Усі рішення та проєкти повинні бути якнайпростішими, і це сприяє зрозумінню та підтримці коду. Методологія акцентує на тому, що простий код зазвичай є найефективнішим і найменш вразливим до помилок.

Іншою важливою рисою *XP* є прагнення до вдосконалення процесу розробки через систематичні зміни та постійний навчальний процес. Кожен етап роботи ретельно аналізується, і проводяться відповідні корективи. Методологія розробки визнає важливість рефакторингу, тобто процесу поліпшення якості коду та архітектури через систематичні зміни.

Особливу увагу приділяється роботі з замовником. Відкрита співпраця та зворотній зв’язок з замовником допомагають забезпечити, що розроблюваний продукт відповідає його потребам і очікуванням.

Extreme Programming активно застосовується в різних галузях, де важливо розробляти програмне забезпечення в інноваційний, ефективний і якісний спосіб. Ця методологія використовується у розробці програмного забезпечення для веб-програмування, мобільних додатків, інформаційних систем, вбудованих систем та багатьох інших областях. Вона дозволяє створювати програмне забезпечення, яке

відповідає високим стандартам якості і може бути швидко адаптоване до змін в ринкових умовах.

Багато компаній та розробників програмного забезпечення активно застосовують *Extreme Programming* у своїй роботі. Вони переконані, що цей підхід допомагає підвищити ефективність, знизити ризики і забезпечити високу якість програмних продуктів. Також слід зазначити, що *XP* постійно розвивається, і нові методи та ідеї додаються до цієї методології з метою її вдосконалення та пристосування до сучасних вимог та технологій.

Extreme Programming – це інноваційний підхід до розробки програмного забезпечення, який акцентує на комунікації, гнучкості та високій якості коду. Він допомагає розробникам створювати якісне програмне забезпечення, яке може бути легко адаптоване до змін вимог і ринкових умов. Ця методологія застосовується з успіхом в різних галузях і продовжує розвиватися для вдосконалення процесу розробки ПЗ.

З іншого боку, інструменти управління проєктами є необхідною складовою успішного виконання проєктів [6]. Програмні продукти, такі як *Microsoft Project*, допомагають планувати, відстежувати та аналізувати проєкти. За допомогою цих інструментів можна створювати графіки, визначати залежності між завданнями, розподіляти ресурси, а також слідкувати за витратами та виконанням термінів.

Microsoft Project – це комплексне програмне забезпечення, розроблене корпорацією *Microsoft,* яке використовується для управління проєктами та завданнями. Ця програма надає інструменти для планування, виконання, моніторингу та контролю проєктів у різних галузях та галузях діяльності. *Microsoft Project* є однією з провідних програм для управління проєктами та широко використовується в багатьох організаціях.

Основні можливості *Microsoft Project* включають у себе створення графіків, завдань, визначення залежностей між ними, розподіл ресурсів, визначення бюджету та моніторинг витрат, а також генерацію звітів та аналітику. Програма підтримує можливість працювати з різними видами проєктів, від малих завдань до великих, довгострокових проєктів.

Однією з ключових можливостей *Microsoft Project* є можливість створення робочих графіків, які відображають послідовність виконання завдань, їх тривалість та залежності. Графіки можуть бути розгорнуті на різні терміни, включаючи дні, тижні, місяці та роки. Це дозволяє проєктним менеджерам та командам точно планувати виконання проєктів та визначати критичні шляхи, які впливають на строк виконання проєкту.

Microsoft Project також надає інструменти для створення завдань та підзавдань, визначення структури проєкту та ресурсів, які будуть призначені для їх виконання. Кожне завдання може мати визначену тривалість, призначених ресурсів та залежності від інших завдань. Це дозволяє створювати докладні плани виконання проєкту та визначати, хто та коли повинен виконувати певні завдання.

Для великих проєктів, де задіяні різні ресурси та підрозділи організації, *Microsoft Project* дозволяє визначити ресурси та їх доступність. Це важливо для ефективного розподілу робіт та визначення, які ресурси будуть зайняті на повну зайнятість, а які - на часткову.

Програма підтримує можливість створення бюджету для проєкту та моніторингу витрат. Користувачі можуть додавати витрати на ресурси, матеріали та інші витрати та відслідковувати, чи не перевищено бюджет проєкту.

Microsoft Project також дозволяє генерувати різноманітні звіти та аналітику. Це допомагає командам та керівникам проєктів зрозуміти стан проєкту, виявити потенційні проблеми та приймати обґрунтовані рішення.

Однією з важливих переваг *Microsoft Project* є інтеграція з іншими програмами *Microsoft,* такими як *Microsoft Excel* та *Microsoft Outlook*. Це дозволяє легко обмінюватися даними та інформацією між різними програмами та платформами.

Програмне забезпечення *Microsoft Project* є платним продуктом, і доступні різні версії та плани ціноутворення для користувачів різних потреб та бюджетів. Версії включають *Microsoft Project Standard, Microsoft Project Professional ma Microsoft Project Server* для підприємств.

У сучасному світі *Microsoft Project* використовується в багатьох галузях, включаючи будівництво, інженерію, IT, маркетинг, охорону здоров’я, освіту та

багато інших. Ця програма допомагає організаціям краще управляти своїми проєктами, знижувати ризики та витрати, покращувати комунікацію та досягати бажаних результатів.

Загалом, *Microsoft Project* є потужним інструментом для управління проєктами, який надає інструменти для планування, виконання та контролю проєктів у різних галузях та галузях діяльності (рис. 1.7). Ця програма сприяє покращенню продуктивності, зменшенню ризиків та забезпечує ефективний контроль над проєктами, допомагаючи організаціям досягати успіху в сучасному бізнесі.

Додаток "Облік книг у бібліотеці"	**36 дней?**	**Вс 14.02.21**	**Пн 05.04.21**			**200 560,00 ₴**
Прийняття рішення про початок проектування	1 день	Вс 14.02.21	Вс 14.02.21			0,00 ₴
Постановка	**9 дней**	**Пн 15.02.21**	**Чт 25.02.21**		**Федорів Михайло**	**30 400,00 ₴**
Опис форм	2 дней	Пн 15.02.21	Вт 16.02.21		Федорів Михайло	3 200,00 ₴
Опис запитів	3 дней	Ср 17.02.21	Пт 19.02.21		Федорів Михайло	4 800,00 ₴
Опис звітів	3 дней	Сб 20.02.21	Вт 23.02.21		Федорів Михайло	4 800,00 ₴
Утвердження ТЗ	2 дней	Ср 24.02.21	Чт 25.02.21		Федорів Михайло	3 200,00 ₴
Кодування	**15 дней**	**Пн 01.03.21**	**Сб 20.03.21**			**72 160,00 ₴**
Форми	**5 дней**	**Пн 01.03.21**	**Сб 06.03.21**		**Михайлюк Петро**	**30 800,00 ₴**
Функціонування елементів форм	4 дней	Пн 01.03.21	Чт 04.03.21		Михайлюк Петро	11 200,00 ₴
Дизайн форм	2 дней	Пт 05.03.21	Сб 06.03.21		Михайлюк Петро	5 600,00 ₴
Запити	**5 дней**	**Пн 08.03.21**	**Сб 13.03.21**		**Качанюк Максим**	**13 200,00 ₴**
Запити зроблені бібліотекарем	4 дней	Пн 08.03.21	Чт 11.03.21		Качанюк Максим	4 800,00 ₴
Запити зроблені відвідувачем	2 дней	Пт 12.03.21	Сб 13.03.21		Качанюк Максим	2 400,00 ₴
Звіти	**5 дней**	**Пн 15.03.21**	**Сб 20.03.21**		**Дмитрук Олег**	**28 160,00 ₴**
Звіти для бібліотекаря	3 дней	Пн 15.03.21	Ср 17.03.21		Дмитрук Олег	7 680,00 ₴
Звіти для відвідувача	3 дней	Чт 18.03.21	Сб 20.03.21		Дмитрук Олег	7 680,00 ₴
Тестування	**10 дней**	**Пн 22.03.21**	**Вс 04.04.21**		**Стефак Андрій**	**98 000,00 ₴**

Рис. 1.7. План проєкту створений в *Microsoft Project*

Крім того, управління проєктами має свою власну класифікацію інструментів, які включають у себе:

– інструменти для планування і контролю проєктів;

– інструменти для визначення обсягу та вимог проєкту;

– інструменти для управління командою та співпраці;

– інструменти для аналізу ризиків та управління ними;

– інструменти для моніторингу та звітування про стан проєкту.

Управління проєктами стає необхідністю для різних галузей, включаючи *IT,* будівництво, маркетинг, охорону здоров'я та багато інших. Всі ці галузі вимагають відповідних методологій та інструментів для успішного виконання проєктів.

Управління проєктами не тільки допомагає досягати поставлених цілей, але і сприяє зростанню продуктивності та конкурентоспроможності організацій на ринку.

Усі ці фактори роблять управління проєктами надзвичайно актуальним та важливим для сучасного бізнесу та організацій. Ретельне планування, ефективний контроль та здатність до адаптації роблять управління проєктами ключовим елементом успіху у сучасному світі.

1.2. Програмні продукти для управління проєктами.

Програмні продукти для управління проєктами є невід'ємною частиною сучасного бізнесу та технічних проєктів. Вони грають важливу роль у полегшенні процесів планування, виконання та моніторингу проєктів у різних сферах діяльності. Програмні засоби для управління проєктами дозволяють організаціям та командам ефективно спрямовувати свої зусилля на досягнення поставлених цілей, контролювати та аналізувати роботу над проєктами, а також підвищувати продуктивність та знижувати ризики.

Програмні продукти для управління проєктами можуть бути величезною допомогою в будь-якій галузі, де потрібно планувати та виконувати проєкти. Ці інструменти дозволяють збалансувати ресурси, визначити завдання, створити графіки та діаграми, слідкувати за прогресом та бюджетами, а також забезпечують ефективну комунікацію в команді та замовниками [8].

Управління проєктами полягає в плануванні, виконанні та контролі набору дій, спрямованих на досягнення конкретної мети в рамках визначених обмежень, таких як обсяг, бюджет та час [7]. Для досягнення успіху в управлінні проєктами потрібно мати системний підхід, використовувати ефективні методології та відповідні програмні інструменти.

Програмні продукти для управління проєктами можна розділити на кілька категорій в залежності від їх функціональності та призначення. Одні засоби призначені для створення планів та розкладів, інші - для моніторингу та аналізу

прогресу, а є також інструменти, які поєднують усі необхідні функції управління проєктами.

Однією з популярних категорій програмних продуктів для управління проєктами є програми для створення графіків та діаграм. Вони дозволяють користувачам створювати та відстежувати графіки завдань, визначати послідовність виконання, розраховувати критичний шлях та залежності між завданнями. Ці інструменти забезпечують візуалізацію проєктів та полегшують планування робіт.

Іншою категорією програмних засобів для управління проєктами є інструменти для моніторингу та контролю. Вони дозволяють слідкувати за прогресом виконання завдань, визначати завдання, які потребують уваги, та вчасно реагувати на зміни у проєкті. Ці інструменти надають змогу керівникам та командам ефективно виправляти проблеми та дотримуватися графіку.

Окремою категорією програмних продуктів для управління проєктами є інструменти для спільної роботи та комунікації в команді. Вони дозволяють співробітникам спільно працювати над проєктами, обмінюватися інформацією, документами та коментарями. Ці інструменти покращують комунікацію та сприяють ефективному співробітництву.

Також існують програмні продукти для фінансового управління проєктами, які дозволяють слідкувати за бюджетом та витратами, розраховувати планові та фактичні витрати, а також генерувати фінансові звіти та аналізувати ефективність витрат.

Класифікація програмних продуктів для управління проєктами також може включати інструменти для ризик-менеджменту, які допомагають ідентифікувати та оцінювати ризики, а також планувати заходи для їх зменшення [8]. Такі інструменти роблять проєкти більш надійними та стійкими до непередбачуваних ситуацій.

Важливо також відзначити, що програмні продукти для управління проєктами можуть бути локальними, які встановлюються на комп’ютери користувачів, або хмарними, які працюють в онлайн-режимі та не вимагають встановлення. Вибір конкретного інструмента залежить від потреб користувачів, обсягу та складності проєктів.

Деякі з популярних програмних продуктів для управління проєктами включають такі найменування, як "*Microsoft Project", "Asana", "Trello", "Jira", "Basecamp", "Smartsheet", "Wrike", "Monday.com", "TeamGantt", "Notion", "Clarizen", "Workfront", "ClickUp", "Redmine", "Podio", "LiquidPlanner", "Zoho Projects", "Airtable", "Celoxis", "ProWorkflow", "Planview", "Sciforma", "WorkOtter", "Workamajig",* та багато інших [8]. Кожен з цих продуктів має свої особливості та переваги, що відповідають різним потребам користувачів.

Microsoft Project – це програма для управління проєктами, розроблена *Microsoft*. Вона надає інструменти для планування, відстеження та управління проєктами будь-якої складності. Графічне представлення графіків, можливість встановлення залежностей між задачами, ресурсне управління.

Asana – це платформа для спільної роботи та управління завданнями. Забезпечує командам зручний інструмент для планування та спілкування.

Trello – це інструмент для керування завданнями за принципом дошки з картками. Дозволяє візуально відстежувати стан завдань. З особливостей: Дошки, списки, картки, календар, легка інтерфейс.

Jira – це продукт компанії *Atlassian* для управління розробкою програмного забезпечення та проєктами. Має систему відстеження помилок, завдань, керування проєктами, звітність.

Basecamp – це платформа для організації комунікації та управління проєктами. Простий та ефективний інтерфейс. Має можливість створювати завдання, календарі, документи, чати.

Також слід зазначити, що програми для управління проєктами можуть бути спеціалізованими для певних галузей, наприклад, будівництва, інформаційних технологій, маркетингу чи дизайну. Вони мають функції та інструменти, спеціально розроблені для вирішення конкретних завдань та проблем у відповідних сферах.

У сучасному світі управління проєктами виконує важливу функцію в бізнесі та технічних проєктах. Це допомагає організаціям та командам досягати своїх цілей, зменшувати ризики та помилки, покращувати комунікацію та співробітництво. Програмні продукти для управління проєктами роблять цей процес більш

організованим та ефективним, допомагаючи керівникам та командам ефективно робити свою роботу і досягати успіху у своїх проєктах.

1.3. Висновки до розділу

У висновку дослідження в галузі методологій управління проєктами, можна відзначити, що ці методи та підходи відіграють критичну роль у сучасному бізнесі та технічних проєктах. Їх застосування дозволяє організаціям та командам досягати високої продуктивності, досягати своїх цілей та впевнено конкурувати на ринку.

Методології управління проєктами відіграють ключову роль у забезпеченні системного та методичного підходу до керування проєктами. Вони надають структурованість та організованість у роботі над проєктами, що дозволяє уникнути хаосу та невизначеності. Кожна методологія має свої унікальні особливості та принципи, але в цілому вони спрямовані на досягнення високої якості та вчасного завершення проєктів.

Також слід відзначити, що методології управління проєктами є еволюційними та постійно вдосконалюються. Сучасні вимоги до управління проєктами вимагають адаптації та розробки нових методологій, які враховують зміни в бізнес-середовищі та технологічний прогрес.

Щодо програмних продуктів для управління проєктами, вони грають важливу роль у полегшенні рутинних завдань, автоматизації процесів та забезпеченні доступу до важливої інформації. Ці програми допомагають командам та менеджерам ефективно планувати, виконувати моніторинг та контроль, спілкуватися та співпрацювати над проєктами.

Програмні продукти для управління проєктами класифікуються за функціональністю, типами проєктів та масштабами. Їх можна використовувати як для невеликих командних завдань, так і для великих корпоративних проєктів. Серед найпопулярніших програм можна виділити *Microsoft Project, Trello, Asana, Jira,* і багато інших. Кожен з цих продуктів має свої переваги та обмеження, і вибір залежить від потреб користувачів.

Актуальність методологій та програмних продуктів управління проєктами полягає в їх здатності допомогти організаціям досягти успіху в умовах зростаючого конкурентного тиску та швидких змін. Управління проєктами стає ключовим чинником успіху у сучасному бізнесі та технічних проєктах, і правильний вибір методології та програмного продукту може забезпечити високу результативність та ефективність.

Загалом, методології управління проєктами та програмні продукти для управління проєктами грають важливу роль у сучасному світі. Вони допомагають організаціям та командам досягати своїх цілей, ефективно розподіляти ресурси та керувати ризиками. Ці інструменти та методи дозволяють досягати високої продуктивності та конкурентоспроможності в сучасному бізнесі та технічних проєктах.

.

РОЗДІЛ 2
ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

2.1. Функціональні можливості середовища “*Jira*”

Середовище "*Jira*" є популярною системою управління проєктами та відстеження задач, розробленою компанією *Atlassian*. Ця платформа забезпечує широкий спектр функціональних можливостей для керування проєктами, спрощуючи процеси розробки програмного забезпечення та управління задачами в будь-якій галузі. Серед ключових можливостей "*Jira*" варто виділити створення, відстеження та оновлення задач, присвоєння їм пріоритетів та відповідальних осіб, а також встановлення термінів виконання. Кожна задача може бути описана докладною інформацією, включаючи текстові описи, вкладені файли, посилання на інші задачі та користувачів. Багатофункціональний інтерфейс дозволяє користувачам створювати, редагувати та видаляти задачі, а також взаємодіяти з ними шляхом коментування та прикріплення файлів.

Структура середовища продемонстрована на рис. 2.1.

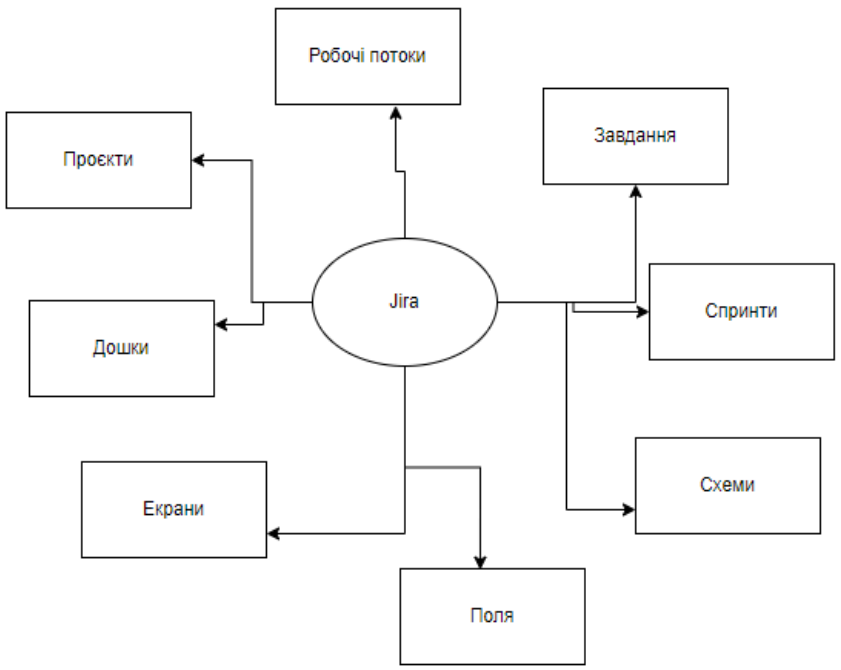

Рис. 2.1. Середовище *Jira*

Структура *Jira* може змінюватися залежно від конкретних потреб проєкту та вибору команди, проте загальна структура включає в себе:

1.Проєкти:

– проєкт у *Jira* - це контейнер для завдань та користувацьких даних.

– кожен проєкт має унікальні параметри, пов’язані із типом проєкту та вимогами команди.

2.Типи завдань:

– завдання в *Jira* можуть бути різних типів, таких як задачі (*Task*), помилки (*Bug*), нові можливості (*Feature*) та інші.

– кожен тип завдання може мати свої унікальні поля та робочі процеси.

3. Схеми:

– схеми включають налаштування для різних аспектів *Jira,* таких як схеми роботи, схеми полів та схеми екранів.

– схеми дозволяють настроїти, які елементи доступні для різних типів завдань.

4. Екрани:

– екрани визначають, які поля відображаються при створенні, редагуванні або перегляді завдання;

– вони пов’язані з операціями, такими як "Створити" чи "Змінити".

5. Поля:

– *Jira* надає різноманітні поля для завдань, такі як опис, пріоритет, виконавець та інші;

– також можна створювати користувацькі поля залежно від вимог проєкту.

6. Робочі потоки:

– робочий потік визначає життєвий цикл завдання від його створення до завершення;

– включає в себе стани та переходи між ними, відображаючи етапи обробки завдання.

7. Дошки:

– дошки в *Jira* можуть бути *Scrum*-дошкою, *Kanban*-дошкою чи дошкою за замовчуванням.

– вони забезпечують візуальне відстеження завдань та працюють в режимі реального часу.

8. Спринти:

– у контексті *Scrum*-проєктів спринт представляє собою фіксований часовий інтервал, протягом якого команда виконує певний обсяг роботи.

Ці елементи можуть бути налаштовані та адаптовані в залежності від конкретних потреб проєкту та команди. Структура *Jira* може бути більш або менш складною в залежності від вимог проєкту та вибору команди.

Серед важливих можливостей є можливість створення і керування робочими потоками задач, що дає змогу визначити послідовність операцій, необхідних для виконання конкретної задачі, та автоматизувати їх обробку. "*Jira*" підтримує створення користувацьких полів, що дозволяє користувачам налаштовувати інформацію, яку вони вводять при створенні задач. Це розширює можливості адаптації системи до конкретних вимог проєкту та галузі.

Управління проєктами в "*Jira*" включає створення та призначення задач різним командам та користувачам, а також відслідковування часу, витраченого на виконання кожної задачі. За допомогою графіків та звітів можливо аналізувати продуктивність команди та визначити темпи виконання проєкту.

"*Jira*" підтримує інтеграцію з іншими інструментами розробки програмного забезпечення, такими як системи контролю версій і засоби автоматизації тестування. Це полегшує спільну роботу розробників та тестувальників, дозволяючи їм спільно відстежувати прогрес інтеграції та виявлення помилок.

Серед можливостей "*Jira*" варто відзначити можливість визначення та відстежування епіків, які об’єднують набір задач за спільною метою або функціоналом. Це допомагає керувати великими проєктами та встановлювати пріоритети між різними частинами розробки. Додаток "*Jira Agile*" дозволяє використовувати методики керування проєктами *Scrum* та *Kanban* для організації роботи команд.

Платформа "*Jira*" також підтримує розширення через додатки, які розробляються спільнотою та сторонніми розробниками. Це дозволяє розширити

функціональність системи за допомогою різноманітних додатків, включаючи засоби для автоматизації процесів та інтеграції з іншими системами.

Особливу увагу варто приділити засобам безпеки та доступності в "*Jira*". Система надає можливість налаштування прав доступу до задач, обмеження прав редагування та видалення, а також ведення журналу подій для відстежування дій користувачів. Таким чином, забезпечується конфіденційність та цілісність даних проєкту.

"*Jira*" дозволяє інтегрувати систему зі засобами автоматизації, які спрощують рутинні операції та дозволяють автоматично створювати задачі, оновлювати статуси та виконувати інші дії відповідно до заданих правил і умов. Це підвищує продуктивність та забезпечує точність процесів управління проєктами.

У великих організаціях "*Jira*" надає можливість розділення проєктів на велику кількість команд та проєктних просторів. Кожен проєктний простір має власний набір налаштувань, прав доступу та користувацьких полів, що дозволяє дотримуватися специфічних потреб кожного проєкту.

У підсумку, середовище "*Jira*" володіє розширеними функціональними можливостями для управління проєктами та задачами, що дозволяє організаціям розробляти програмне забезпечення та виконувати інші проєкти з високою продуктивністю, контролем і безпекою. Інтеграція з іншими інструментами розробки, розширення через додатки та можливості налаштування робочих потоків роблять "*Jira*" потужним інструментом для управління проєктами у будь-якій галузі.

2.2. Алгоритм роботи з інструментом управління проєктами “*Jira*”

Jira представляє собою систему управління проєктами, яка забезпечує можливість вирішення майже всіх завдань у сфері управління проєктами за допомогою єдиного інструменту: від планування до моніторингу процесів та результатів. Вона є комплексом IT-рішень від компанії *Atlassian* і об’єднується в сім’ю продуктів *Jira: Jira WM* для взаємодії з бізнес-процесами, *Jira SM* для

створення сервіс-десків, *Jira Software* для проєктів з розробки програмного забезпечення [42]. Продукти *Atlasion* представлені на рис. 2.2.

Рис. 2.2. Програмні продукти *Atlasion*

Переваги *Jira*:

– інтеграція та об'єднання: *Jira* дозволяє легко інтегрувати різноманітні інструменти і об'єднує їх в одній системі, спрощуючи взаємодію між командами та проєктами.

– гнучкість: система *Jira* дуже гнучка, що дозволяє налаштовувати та адаптувати її під різні потреби та види проєктів.

– велика спільнота та підтримка: оскільки *Jira* є популярною системою управління проєктами, існує велика спільнота користувачів та широкі ресурси для підтримки.

– моніторинг та звітність: *Jira* надає потужні інструменти для моніторингу процесів та створення детальних звітів, що сприяє більш ефективному управлінню проєктами.

– широкий функціонал: залежно від конкретної потреби, *Jira* пропонує різні модулі та додатки, такі як *Jira Software, Jira Service Management, Jira Work Management,* що дозволяє використовувати систему для різноманітних цілей.

Недоліки *Jira*:

– складність навчання: використання всіх функцій *Jira* може вимагати певного часу для навчання, і новачкам може знадобитися певний час, щоб зрозуміти всі можливості системи;

– вартість: *Jira* є комерційним продуктом, і вартість може бути високою, особливо для невеликих команд або стартапів з обмеженим бюджетом.

– можливість перевантаження: завелика кількість функцій та налаштувань може призвести до перевантаження користувача, який може втратити орієнтацію в системі.

– не завжди орієнтована на не-*IT* проєкти: в деяких випадках, інструмент може бути занадто націленим на *IT*-проєкти, що робить його менш зручним для деяких не-технічних команд.

– залежність від Інтернет-з'єднання: робота з *Jira* часто потребує стабільного Інтернет-з'єднання, що може бути недоліком у випадку обмеженого доступу до мережі.

Розробляючи проєктні рішення для управління задачами та контролю процесів, проєктним менеджерам сьогодні необхідно враховувати високий рівень складності та динамічності індустрії розробки програмного забезпечення [42]. В контексті цього важливим елементом стає вибір оптимальних інструментів для відслідковування задач та управління проєктами. Одним із ключових засобів, які варто вивчати програмним менеджерам, є *Jira*.

Jira, як інтегрована система від компанії *Atlassian,* вирізняється високою гнучкістю та широким функціоналом, спрямованим на різноманітні потреби управління проєктами. Вивчення цього інструменту надає *PM*-ам можливість ефективно оптимізувати процеси взаємодії великих розробницьких команд, забезпечуючи найвищий рівень координації та співпраці.

При використанні *Jira PM* отримує зручний інтерфейс для розподілу завдань, визначення пріоритетів та встановлення термінів виконання. Система автоматизовано відстежує прогрес кожного завдання, надаючи *PM*-у широкі можливості для аналізу та моніторингу.

Однією з ключових переваг *Jira* є інтеграція з іншими інструментами розробки програмного забезпечення. Це дозволяє плавно впроваджувати *Jira* в існуючі розробницькі процеси та забезпечує єдиноцентровий погляд на весь життєвий цикл проєкту.

Необхідно враховувати також велику спільноту користувачів *Jira,* що сприяє обміну досвідом та надає різноманітні ресурси для вирішення технічних питань. При вивченні *Jira, PM* здобуває можливість максимально ефективно використовувати цей інструмент для досягнення стратегічних цілей проєктів та оптимізації управлінських рішень.

Однією з ключових функцій *Jira* є можливість створення та відстеження задач. Система надає механізм для детального опису задач, включаючи технічні специфікації, терміни виконання, та інші параметри [42]. Крім того, присутня можливість призначення відповідальних осіб та визначення пріоритетів для кожної задачі.

Jira також володіє розширеними можливостями для організації робочих процесів. Модуль *Workflows* дозволяє визначити послідовність етапів виконання задачі, включаючи етапи перевірки, тестування та затвердження. Це дозволяє точно відслідковувати прогрес та визначати етапи, на яких можливі затримки.

Крім того, важливо відзначити можливості моніторингу та аналітики, які пропонує *Jira*. Система дозволяє створювати звіти з різноманітних параметрів, таких як час виконання завдань, продуктивність команди та інші аспекти проєктного управління.

Важливим елементом функціоналу *Jira* є можливість інтеграції з іншими інструментами розробки та управління версіями. Це забезпечує єдиний доступ до інформації, сприяє уніфікації робочих процесів та покращує зручність взаємодії між різними аспектами проєктної діяльності.

У контексті використання *Jira,* можна виділити три основних типи проєктів (рис. 2.3), кожен із яких визначається своєю структурою та методологією управління завданнями.

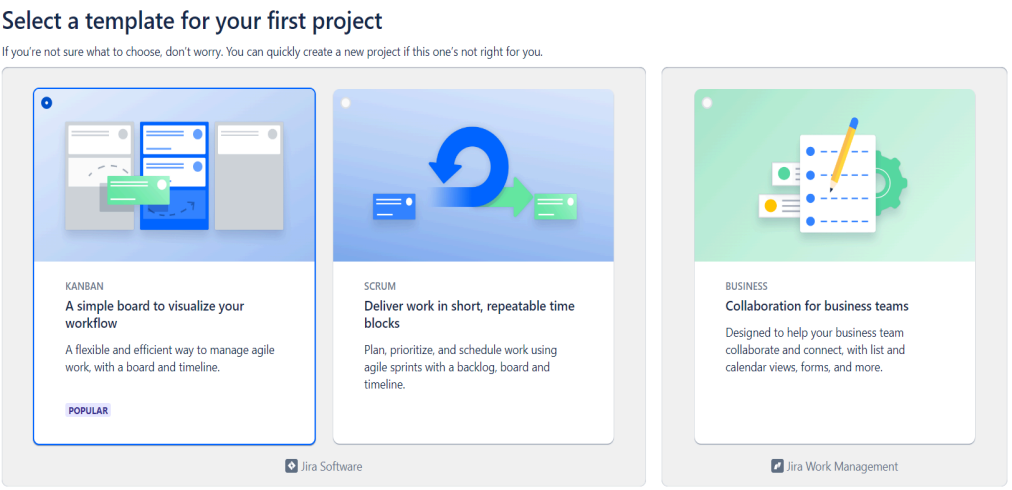

Рис. 2.3. Шаблони проєкту *JIRA*

Перший тип – *Scrum,* представляє собою гнучку методологію розробки, спрямовану на ітераційну постачання функціоналу. В *Scrum* проєкті використовуються спринти, що є фіксованими часовими періодами, під час яких команда працює над конкретними задачами.

Другий тип – *Kanban,* базується на концепції візуального управління потоком робіт. Тут завдання представлені на дошці, і команда переводить їх від статусу "В очікуванні" до "Завершено". Кожна задача може рухатися через ці стани в залежності від поточного навантаження та пріоритетів.

Третій тип – *Bug Tracking,* призначений для відстеження та управління помилками під час розробки програмного забезпечення. Такі проєкти дозволяють точно визначити, де виникають проблеми та визначити їх пріоритети для подальшого виправлення.

Важливим аспектом є можливість комбінування цих типів проєктів в межах одного інструменту, що забезпечує гнучкість та адаптабельність до різноманітних вимог та специфікацій розробки. Такий підхід дозволяє забезпечити ефективне управління завданнями, що включаються в різні фази життєвого циклу розробки програмного забезпечення.

В табл. 2.1 продемонстровано різницю в проєктах *Jira* для компанії та для команди.

Таблиця 2.1

Порівняння проєктів

Параметр	Керований командою	Керований компанією
Структура завдань	Робочі завдання створюються та розподіляються самою командою. Є велика автономія в розробці.	Завдання визначаються компанією та призначаються розробникам, які спеціалізуються на виправленні конкретних помилок.
Гнучкість та адаптивність	Гнучкість у визначенні та зміні завдань під час розробки. Можливість адаптації до змін у вимогах.	Завдання часто фіксуються та призначаються із значним заздалегідь. Менше гнучкості в реакції на невідомі фактори.
Співпраця та комунікація	Комунікація через спринти, зустрічі та регулярні огляди. Звернення до *Scrum Master або Product Owner.*	Завдання та відстеження помилок можуть бути ретельно відслідковані та наглядатися. Більше орієнтації на офіційну документацію.
Звітність та аналітика	Велика кількість звітів, таких як *Burndown* чи *Velocity Charts,* для моніторингу та оцінки продуктивності.	Орієнтованість на статистику та аналіз дефектів для вирішення проблем. Звіти можуть бути спрямовані на аналіз ефективності процесу виправлення помилок.

У контексті *Jira,* права доступу в проєктах можуть бути налаштовані на три основних рівні:

1. *Private* (приватний): у режимі *Private* лише адміністратор та учасники проєкту мають повний доступ до всіх завдань. Це означає, що інші користувачі компанії *Jira* не мають можливості переглядати, створювати чи редагувати завдання в рамках цього проєкту. Такий підхід забезпечує максимальний рівень конфіденційності та обмежує доступ лише до обраної групи учасників;

2. *Limited* (обмежений): у випадку обмежених прав доступу, будь-який користувач компанії *Jira* може переглядати завдання всередині проєкту. Однак лише учасники проєкту мають право створювати та редагувати завдання. Цей режим забезпечує більший ступінь відкритості, дозволяючи більшій кількості користувачів переглядати вміст, але при цьому зберігає контроль над редагуванням завдань в руках учасників проєкту;

3. *Open* (відкритий): в режимі *Open* надається практично повний та відкритий доступ до всіх завдань проєкту. Будь-який користувач компанії *Jira* може не тільки переглядати, але і створювати та редагувати завдання. Цей режим найбільш відкритий та використовується у випадках, коли важлива максимальна доступність для широкого кола користувачів.

В системі *Jira* існують 3 основних типи користувачів:

– *Administrator* – адміністратор у системі *Jira* володіє повним контролем та доступом до всіх аспектів проєкта. Йому надається влада управління всіма завданнями, ресурсами та конфігурацією проєкту. Адміністратор має можливість визначати та налаштовувати ролі учасників, контролювати доступ до конфіденційної інформації, а також встановлювати права користувачів у рамках проєкту. Завдяки повному доступу, адміністратор може ефективно керувати всіма аспектами роботи команди та забезпечувати оптимальне функціонування проєкту;

– *Member* – учасник з роллю "*Member*" у *Jira* має обмежений, але важливий функціонал. Ця роль дає можливість створювати та редагувати завдання в межах проєкту. Крім того, учасник може взаємодіяти з іншими учасниками команди, обмінюватися інформацією та співпрацювати над вирішенням завдань. Це сприяє активній участі у розробці та підтримці проєкту;

– *Viewer* – учасник, який має роль "*Viewer*", обмежений у можливостях внесення змін в систему. Він може лише переглядати завдання та коментувати їх, не маючи можливості вносити зміни або створювати нові. Ця роль підходить для тих, хто не бере активної участі у виконанні завдань, але має необхідність слідкувати за процесами та отримувати інформацію про хід проєкту.

Jira використовує різні типи завдань для визначення різних видів робіт або проблем, які можуть виникнути в ході проєкту. Серед найбільш поширених типів можна виділити "Завдання" для конкретних задач (рис. 2.4), "Підзадача" для деталізації певного завдання, "Історія" для відстеження змін та "Помилка" для виправлення виявлених дефектів.

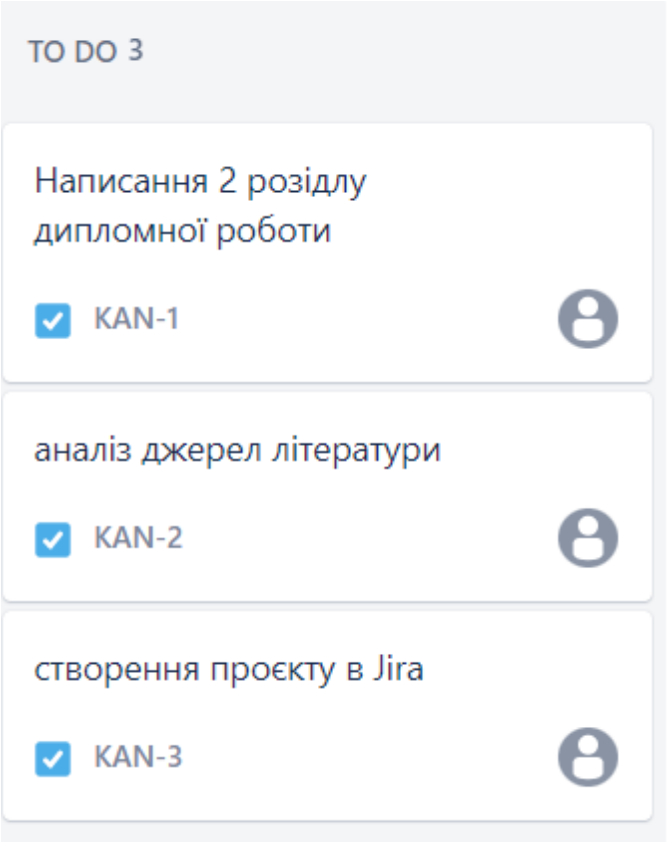

Рис. 2.4. Перелік завдань *Jira*

Щоб дослідити ієрархію в *Jira*, слід звернутися до поняття "Епік", що представляє собою великий об’єкт, що може бути розкладений на більш деталізовані "Підзадачі" або "Історії". Також, для організації завдань на більш високому рівні, використовується "Компонент", який дозволяє групувати схожі за функціональністю частини проєкту.

Епік є важливим компонентом у системі управління проєктами *Jira*. Це великий блок роботи, що об’єднує ряд пов’язаних завдань або історій, зазвичай спрямованих на досягнення великого, стратегічного метапроєкту. Епік виступає в ролі високорівневого контейнера, який дозволяє групувати та керувати великими обсягами роботи, розбиваючи його на менші та керовані блоки.

Ключовою особливістю епіка є його здатність розглядатися як стратегічний об’єкт, який визначає широкі цілі та напрямки. В процесі виконання епіку можуть бути призначені конкретні завдання, підзавдання або історії, які вже визначаються як частини виконавчого плану для досягнення загальної мети епіка.

Епік допомагає розробникам та управлінцям отримати стратегічний огляд на робочий процес, підвищуючи ефективність управління великими та складними проєктами. Використання епіків в *Jira* сприяє визначенню пріоритетів, забезпечує структурований підхід до великих завдань та сприяє гнучкому керуванню стратегією розвитку проєкту.

У *Jira* поля представляють собою ключові атрибути для опису і визначення властивостей завдань. Поля можуть бути стандартними або визначатися користувачем залежно від конкретних потреб проєкту. Наприклад, "Заголовок" і "Опис" визначають основні характеристики завдань, тоді як "Пріоритет", "Статус" та "Відповідальний" надають додаткову інформацію для керування процесами та взаємодії з командою.

Враховуючи ці аспекти, *PM* або розробник може ефективно використовувати *Jira* для визначення, відслідковування та керування завданнями в рамках проєкту, надаючи структурований та систематизований підхід до управління завданнями та їх реалізацією.

Workflow у Jira є ключовим елементом, визначаючим послідовність станів та дій, через які проходять завдання від створення до завершення. Це концептуальна схема, яка відображає весь життєвий цикл задачі в процесі її обробки. *Workflow* дозволяє визначити, як завдання переходять від одного стану до іншого та які дії виконуються на кожному етапі.

Ключові компоненти *Workflow*

1. Стани (*Status*): Кожен *Workflow* складається з різних станів, таких як "Відкрито", "В роботі", "На перевірці", "Завершено" тощо. Кожен стан представляє конкретний етап життєвого циклу завдання.

2. Транзакції (*Transition*): Транзакції визначають переходи між різними станами. Наприклад, завдання може перейти зі стану "Відкрито" до "В роботі" після

того, як воно було призначено відповідальному.

3. Дії (*Action*):Дії вказують, які конкретні дії або зміни відбуваються при переході між станами. Наприклад, при переході в стан "На перевірці" може бути автоматично відправлене повідомлення про необхідність рецензії.

4. Умови (*Condition*): Умови визначають, за яких обставин та умов переходить завдання від одного стану до іншого. Наприклад, можливо встановити умову, що завдання може перейти в стан "Завершено" лише після виконання всіх попередніх завдань.

5. Параметри *Workflow*: Це налаштування, які визначають особливості роботи *Workflow,* такі як автоматичні тригери, виконання скриптів або налаштування прав доступу.

Визначення та налагодження *Workflow* дозволяє адаптувати систему *Jira* до конкретних потреб проєкту. Це сприяє створенню структурованого та ефективного процесу управління завданнями, підвищує прозорість робочого процесу та допомагає уникнути збоїв в комунікації між членами команди. Працюючи з *Workflow* в *Jira,* користувач може точно контролювати і керувати кожним етапом життєвого циклу завдання.

У *Jira,* часова шкала проєкту представлена кількома ключовими елементами, які допомагають відстежувати та керувати часовими аспектами роботи над завданнями та проєктами (рис. 2.5).

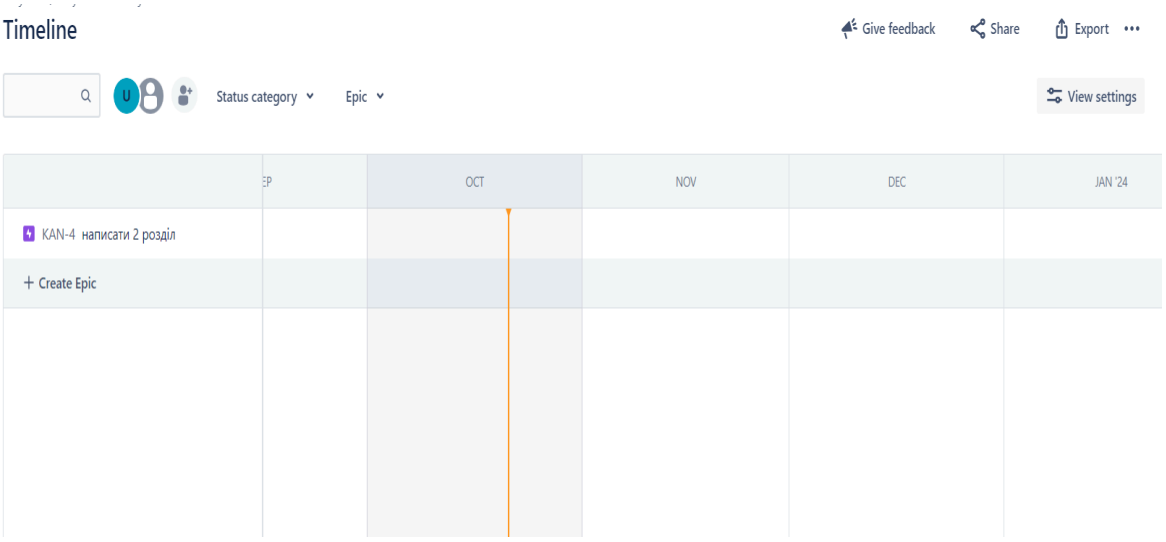

Рис. 2.5. Часова шкала проєкту

Основні елементи часової шкали в *Jira* включають:

1.Терміни (*Due Dates*):

Опис: терміни представляють собою конкретні дати, до яких повинні бути завершені завдання або епіки.

Використання: додаючи терміни до завдань, учасники проєкту можуть чітко визначити та відстежувати крайні терміни для виконання робіт.

2. Спринти (*Sprints*):

Опис: спринти в *Jira* визначають обмежений період часу, протягом якого команда працює над конкретним набором завдань.

Використання: спринти дозволяють впроваджувати методологію *Scrum,* де робота зосереджена на конкретних завданнях протягом фіксованого терміну.

Графіки Ганта (*Gantt Charts*):

Опис: графіки Ганта в *Jira* надають візуальне відображення завдань та їх термінів у вигляді графічного плану.

Використання: графіки Ганта допомагають керівникам проєкту бачити весь обсяг робіт та залежності між завданнями.

Таймлайн проєкту (*Project Timeline*):

Опис: Таймлайн проєкту відображає ключові події та мілістоуни проєкту на певному відрізку часу.

Використання: Це інструмент для відстеження та візуалізації головних етапів розвитку проєкту протягом усього терміну його виконання.

Часові Звіти (*Time Tracking Reports*):

Опис: Можливість відстежування часу, витраченого на виконання кожного завдання чи епіки.

Використання: Забезпечує аналіз витрат часу та оцінки продуктивності команди, а також може бути використаний для планування майбутніх проєктів.

Використання цих елементів в *Jira* дозволяє командам та керівникам проєктів ефективно відслідковувати, аналізувати та керувати часовими ресурсами під час виконання робіт над проєктом.

Jira є потужним інструментом для проджект менеджерів, який забезпечує ефективне управління завданнями та ресурсами, поліпшуючи організацію робочих процесів та сприяючи швидкому прийняттю високоякісних управлінських рішень. Використання різноманітних функцій, таких як ієрархія завдань та часове відстеження, робить *Jira* необхідним інструментом для успішного керування проєктами.

2.3. Інтегрування методу Кластеризації в відслідковування задач та управління проєктами

В ході реалізації алгоритму кластеризації для інструменту управління задачами, виникла потреба в ретельному виборі та оптимізації методів для досягнення високої точності та швидкодії. Виходячи із специфічних вимог проєкту, була обрана техніка ієрархічної кластеризації з використанням методу агломеративного з'єднання.

Кластеризація в програмуванні – це методологія або процес групування схожих об'єктів або даних в кластери або категорії. Метою кластеризації є знаходження прихованих структур у наборах даних та визначення взаємозв'язків між елементами.

Цей підхід застосовується в різних галузях програмування і має різні варіації в залежності від конкретних завдань. Наприклад:

Архітектура ПЗ: В програмній інженерії кластеризація може використовуватися для групування схожих компонентів або модулів у великих програмах.

Кластеризація дозволяє виявляти структуру та порядок в даних, що є важливим для аналізу та прийняття рішень в різних областях програмування.

Ієрархічна кластеризація реалізована з використанням методу агломеративного з'єднання, представляє собою алгоритм, що починає з кожного елемента роздільної множини, розглядає його як окремий кластер, а потім поступово об'єднує найбільш подібні кластери, наближаючись до одного загального кластера.

Процес агломеративного з’єднання визначається відстанями між кластерами та критеріями подібності. На кожному кроці обираються два найбільш подібні кластери, які об’єднуються в новий кластер. Цей процес триває до того моменту, поки всі елементи не утворять один великий кластер.

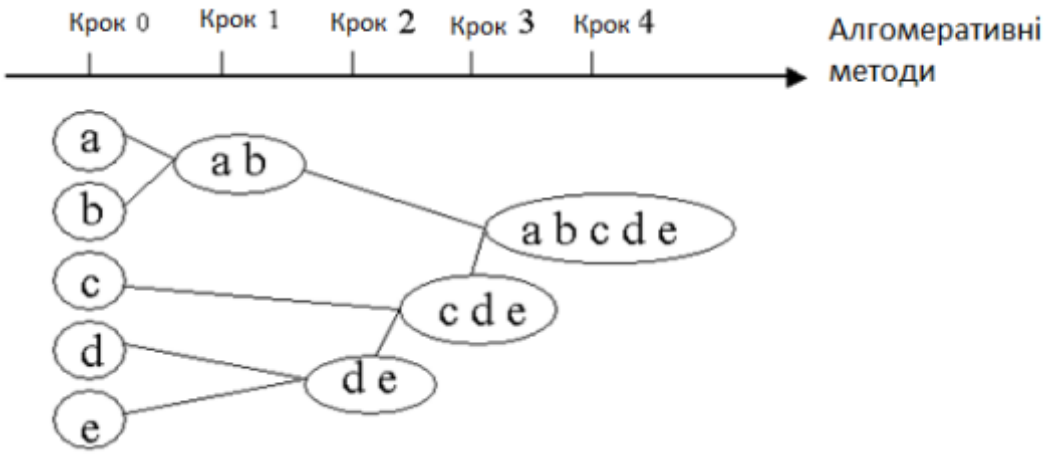

Рис. 2.6. Візуальне представлення кластеризації

Використання ієрархічної кластеризації дозволяє створити структуровану ієрархію задач, що відображає їхні взаємозв’язки та подібності. При цьому, кожен крок агломеративного з’єднання визначається відстанями між кластерами, які можуть бути обчислені різними методами, такими як евклідова відстань, косинусна схожість чи інші відстані, залежно від вимог завдання.

Реалізація методу агломеративного з’єднання включає в себе оптимізації для забезпечення швидкої та ефективної обробки великої кількості завдань. Використання підходів, таких як кластеризація на основі підсумовування, дозволяє враховувати внутрішню структуру кластерів та підвищити точність алгоритму.

Такий підхід до ієрархічної кластеризації, з використанням методу агломеративного з’єднання, забезпечує баланс між рівнем складності алгоритму та його ефективністю у вирішенні завдань управління задачами.

Визначення ключових критеріїв, за якими будуть групуватися задачі, таких як терміни виконання, пріоритет, типи завдань чи інші важливі характеристики.

Для реалізації алгоритму кластеризації в інструментах відстеження задач, спочатку йде отримання данних з цих інструментів. Це включає в себе інформацію

про кожну задачу та її характеристики, такі як назва, опис, дедлайн та відомості про те, на кого призначена задача.

Після отримання даних обирається алгоритм кластеризації, у даному випадку - ієрархічна кластеризація. При цьому враховуються характеристики кожної задачі для визначення схожості між ними та створення ієрархічної структури.

Нормалізація даних – це процес стандартизації значень характеристик, щоб забезпечити однаковий масштаб для різних фізичних величин. У контексті алгоритму кластеризації це може виявитися корисним у випадках, коли характеристики мають різний діапазон значень або величини.

Наприклад, якщо одна характеристика вимірюється в одиницях часу, а інша - в кількості ресурсів, їхні значення можуть бути дуже відмінними за порядком величин. Нормалізація дозволяє зменшити вплив цих різниць в шкалі та забезпечити адекватну обробку даних алгоритмами, чутливими до величини значень.

Отримавши всі потрібні характеристики, можна застосувати алгоритм кластеризації, який ефективно групує задачі відповідно до їхніх спільних рис, полегшуючи управління та взаємодію в команді.

На етапі розробки алгоритму було враховано потребу в ефективному обробленні великої кількості завдань та їх взаємозв’язків. Використання ієрархічної кластеризації забезпечує структурованість отриманої інформації та сприяє визначенню груп подібних задач.

Алгоритм реалізовано з урахуванням особливостей завдань управління задачами, використовуючи техніку векторизації для оптимізації обчислювальних процесів. Це дозволяє забезпечити ефективність аналізу та обробки даних навіть при значних обсягах введених інформаційних потоків.

Для досягнення оптимального розділення завдань на кластери використовується метрика подібності, яка базується на комплексному аналізі критеріїв, таких як терміни виконання, пріоритетність та взаємозалежність. Впровадження такого підходу дозволяє динамічно адаптувати структуру кластерів до змінних умов та вимог користувача.

Необхідно відзначити, що впровадження алгоритму кластеризації в інструмент управління задачами дозволяє значно поліпшити організацію та моніторинг завдань, спрощуючи процес прийняття рішень та планування. Використання інтелектуального підходу до кластеризації завдань є ключовим елементом для підвищення продуктивності та оптимізації управління проєктами у сфері завдань та проєктів.

2.4. Висновки до розділу

Jira відзначається не лише своїм функціоналом, але й можливістю легко адаптуватися до потреб різних типів проєктів. Завдяки гнучкості конфігурації, користувачі можуть налаштовувати різні види завдань, додавати власні поля та створювати унікальні робочі потоки, що робить *Jira* ідеальним інструментом для команд різної спрямованості та величини.

Крім того, *Jira* надає широкі можливості для співпраці та комунікації в межах команди. Функції коментування, прикріплення файлів та спільний доступ до інформації спрощують обмін ідеями та забезпечують прозорість в робочих процесах.

Ще однією значущою перевагою *Jira* є можливість інтеграції з іншими популярними інструментами розробки, такими як *Bitbucket* чи *Confluence*. Це створює єдиний екосистему для управління проєктом, де можливості кожного інструменту доповнюють один одного, забезпечуючи гладке та зручне ведення робочих завдань та проєктів.

Великою перевагою є також наявність широкого спектру додатків та розширень, які можна інтегрувати в *Jira*. Це дозволяє користувачам налаштовувати інструмент під свої унікальні потреби та розширювати його функціоналість, щоб відповідати специфічним вимогам їхніх проєктів.

Крім того, використання *Jira* сприяє створенню прозорого середовища для відстеження продуктивності та вимірювання результатів. Система збору та аналізу даних дозволяє здійснювати періодичний огляд прогресу проєкту, ідентифікувати можливі ризики та шляхи їх усунення. Завдяки цьому команди можуть оперативно

реагувати на зміни, удосконалювати стратегії та підтримувати високий рівень якості та ефективності впродовж усього життєвого циклу проєкту. Таким чином, *Jira* не лише надає засоби для управління завданнями, але і стає інструментом для аналізу та оптимізації робочих процесів на основі зібраних даних.

Такий підхід до ієрархічної кластеризації, з використанням методу агломеративного з'єднання, забезпечує баланс між рівнем складності алгоритму та його ефективністю у вирішенні завдань управління задачами.

В ході реалізації алгоритму кластеризації для інструменту управління задачами, виникла потреба в ретельному виборі та оптимізації методів для досягнення високої точності та швидкодії. Виходячи із специфічних вимог проєкту, була обрана техніка ієрархічної кластеризації з використанням методу агломеративного з'єднання.

Кластеризація дозволяє виявляти структуру та порядок в даних, що є важливим для аналізу та прийняття рішень в інструменті відслідковування та управління проєктів.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

3.1. Структура розробленого сервісу для відслідковування задач та управління проєктом

Структура розробленого сервісу для відслідковування задач та управління проєктом визначається високою рівновагою між ефективністю та гнучкістю використання. Програмний продукт побудований на основі архітектурного підходу, який враховує потреби розробників, керівників проєктів та кінцевих користувачів. Задля досягнення цієї рівноваги, структура сервісу визначається кількома ключовими складовими.

Перш за все, система має розширену базу даних, яка оптимально використовує можливості реляційних СУБД. Ця база даних включає таблиці для зберігання інформації про завдання, проєкти, користувачів та їхні ролі. Використання *SQLite* забезпечує надійність та швидкість операцій з базою даних.

Крім того, ключовою складовою структури є модуль управління користувацькими ролями та правами доступу. Цей модуль забезпечує гнучкість у визначенні рівнів доступу для кожного користувача в залежності від його ролі в проєкті. Наприклад, керівник проєкту може мати розширені права для редагування та видалення завдань, тоді як звичайний учасник має обмежений доступ.

Архітектурно, сервіс побудований на основі модульного підходу, що сприяє простоті розширення та модифікації функціоналу. Кожен модуль відповідає за конкретний аспект системи, такий як керування завданнями, відображення графіків чи сповіщення користувачів.

Окрім того, система має інтерфейс взаємодії для кінцевих користувачів, який враховує сучасні стандарти дизайну та взаємодії. Графічний інтерфейс дозволяє зручно взаємодіяти з функціоналом системи та надає інтуїтивно зрозумілий доступ до всіх необхідних опцій.

Необхідно відзначити, що вся структура спроєктована з урахуванням вимог щодо масштабованості, що дозволяє системі ефективно працювати як для невеликих проєктів, так і для великих підприємств з великою кількістю задач та учасників.

В рамках розробки структури сервісу для відслідковування задач та управління проєктами, необхідно також враховувати аспекти забезпечення безпеки та масштабованості. Безпека є важливою складовою будь-якого інструменту, який обробляє конфіденційну інформацію про проєкти та завдання. Забезпечення правильної автентифікації, авторизації та захисту даних має велике значення для забезпечення конфіденційності та цілісності інформації.

Структура також повинна враховувати можливості інтеграції з іншими інструментами розробки та управління проєктами. Наприклад, можливість імпорту та експорту даних у різноманітних форматах, взаємодія з системами версійного контролю, а також інтеграція з іншими популярними інструментами може значно покращити функціональність та гнучкість сервісу.

З огляду на масштабованість, важливою є можливість ефективно працювати з великою кількістю проєктів та завдань, управляти багатьма користувачами та забезпечити стабільну роботу системи під час інтенсивного використання. Система повинна бути здатна швидко обробляти запити, ефективно використовувати ресурси та легко масштабуватися для росту обсягів даних та користувачів.

Важливим елементом структури є також можливість регулярних оновлень та підтримки. Забезпечення актуальності та стабільності сервісу вимагає систематичного вдосконалення та виправлення помилок, що дозволяє підтримувати високу якість роботи та відповідати змінюючимся вимогам користувачів.

Таким чином, структура розробленого сервісу для відслідковування задач та управління проєктом повинна бути гнучкою, забезпечуючи високий рівень функціональності та ефективності, одночасно враховуючи питання безпеки, інтеграції та масштабованості.

Структура розробленої програми представлена на рис. 3.1.

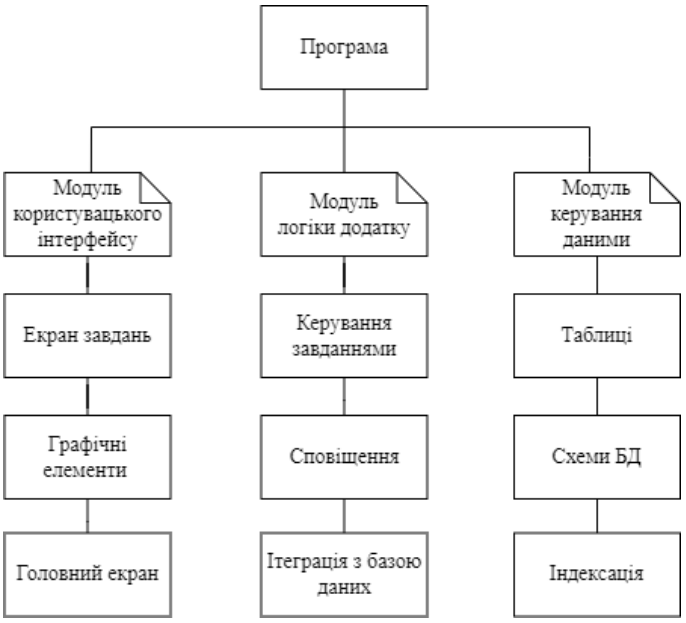

Рис. 3.1. Структура програми

Модуль користувацького інтерфейсу – у цьому модулі розташований графічний інтерфейс для взаємодії з користувачами. Він містить різні екрани та панелі для введення даних, відображення завдань та проєктів, а також опції управління ролями та налаштуваннями.

Модуль логіки додатку – у цьому модулі відбувається обробка логіки бізнес-процесів. Сюди входять різні сервіси та компоненти, такі як:

– керування завданнями;

– управління проєктами;

– база даних;

Модуль даних – у цьому модулі розташована база даних, яка забезпечує надійне зберігання та управління даними. Використовуючи, наприклад, *SQLite*, база даних містить таблиці для кожної основної сутності, такі як *`Tasks`*, *`Projects`*, *`Users`*, та інші.

Така трьохрівнева структура дозволяє ефективно розділити логіку додатку та забезпечити легку розширюваність та модульність системи. Кожен рівень має свою конкретну відповідальність, що сприяє чіткості та підтримці коду.

Модуль користувацького інтерфейсу включає в себе ряд компонентів та елементів, призначених для комфортної та ефективної взаємодії користувачів із системою. Основні елементи цього модулю включають:

Головний екран – огляд завдань та проєктів: на цьому екрані користувач може переглядати загальний огляд усіх завдань та проєктів. Можливість сортування, фільтрації та пошуку дозволяє швидко знаходити потрібну інформацію.

Екран завдань – створення та редагування завдань: форма для введення нових завдань або редагування існуючих. Включає поля для опису, дедлайну та інших важливих параметрів.

Відображення деталей завдань: користувач може переглядати деталі кожного завдання, включаючи коментарі, відзначення статусу та інші додаткові дані.

Графічні елементи – повідомлення та сповіщення: система повідомлень для сповіщення користувачів про важливі події, такі як нові завдання чи зміни в проєктах.

Цей модуль забезпечує інтуїтивний та зручний інтерфейс для користувачів, спрощуючи їх взаємодію з системою та забезпечуючи доступ до всіх необхідних функцій.

Модуль логіки додатку включає ряд компонентів та сервісів, які забезпечують виконання бізнес-логіки та обробку основних функціональних елементів системи.

Створення завдань: дозволяє користувачам створювати нові завдання, визначаючи їхні атрибути, такі як назва, опис, термін виконання тощо.

Редагування та видалення Завдань: забезпечує можливість змінювати параметри завдань або видаляти їх при необхідності.

Сповіщення про терміни виконання завдань: надсилає користувачам сповіщення та нагадування про наближення терміну виконання завдань.

Взаємодія з *SQLite*: забезпечує обробку запитів до бази даних, включаючи читання, запис та оновлення інформації. Цей модуль логіки додатку покликаний об’єднати різні компоненти, щоб забезпечити сумісність та взаємодію між ними. Додатково він може містити обробники подій, перевірки даних та інші елементи, які гарантують правильну роботу системи та дотримання бізнес-правил.

Модуль даних в сервісі для відслідковування задач та управління проєктом відіграє критичну роль у забезпеченні ефективного та надійного зберігання інформації. В даному контексті, модуль Даних базується на використанні реляційної бази даних *SQLite,* яка забезпечує структуроване зберігання даних.

Діаграма прецедентів наведена на рис. 3.2.

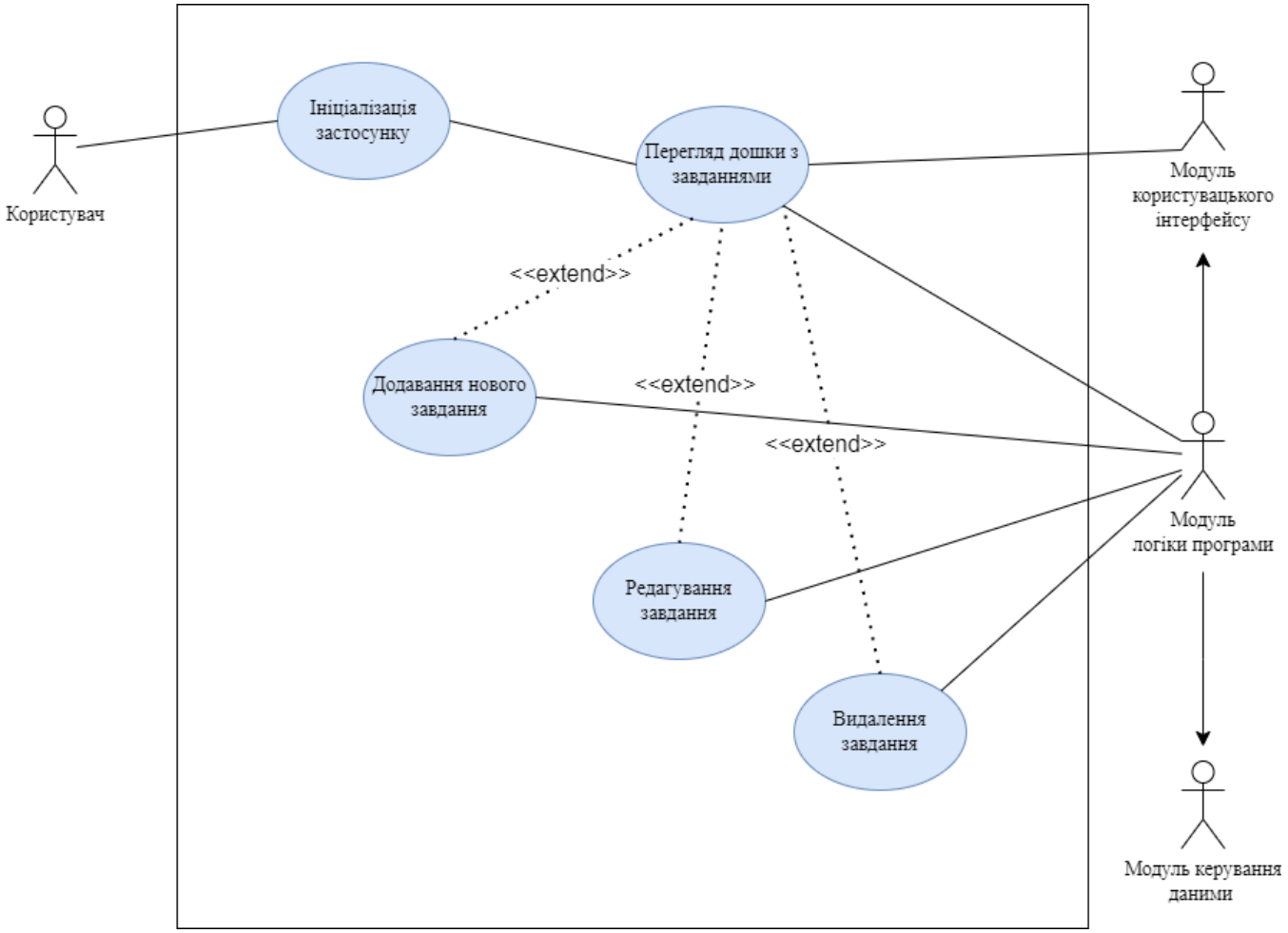

Рис. 3.2. Діаграма прецедентів

Описання таблиць БД:

– Завдання (`*Tasks*`): таблиця, що містить інформацію про кожне завдання, таку як назва, опис, статус, термін виконання та інші атрибути.

– Проєкти (`*Projects*`): таблиця для зберігання даних проєктів, включаючи назву проєкту, опис, терміни, та інші атрибути.

– Користувачі (`*Users*`): таблиця, що містить інформацію про користувачів, таку як ім’я, електронна пошта, роль в системі, тощо.

Зв’язки між Таблицями: використовуються зовнішні ключі для встановлення зв’язків між таблицями. Наприклад, кожне завдання пов’язане з конкретним проєктом, що відображається через зовнішній ключ.

Індексація таблиць: використання індексів для прискорення пошуку та вибірки даних з бази даних.

Оптимізація запитів: впровадження оптимізаційних стратегій для підвищення швидкодії роботи додатку, наприклад, використання кешування.

Модуль даних узагальнює всю інформацію, необхідну для взаємодії з базою даних, та забезпечує необхідну інфраструктуру для зберігання та опрацювання даних у сервісі.

У рамках розробки програмного продукту Інструменти відслідковування задач та управління проєктами потрібно визначити алгоритм роботи програми з метою визначення основних кроків та процесів, що відбуваються під час її використання.

Алгоритм роботи програми починається із завантаження та ініціалізації основних компонентів. Після цього відбувається підключення до бази даних *SQLite,* де зберігаються дані про завдання та проєкти. Важливою частиною цього етапу є перевірка наявності необхідних таблиць у базі даних та їх структури.

Далі виконується ініціалізація графічного інтерфейсу, де користувач може взаємодіяти із програмою. Це включає введення та відображення завдань, редагування їх параметрів та видалення.

Ключовим етапом алгоритму є обробка дій користувача. Коли користувач додає нове завдання чи редагує існуюче, дані вносяться в базу даних, забезпечуючи стійку та ефективну збереження інформації.

Алгоритм роботи програми при створенні нового завдання представлено на рис. 3.3. За подібною схемою працюють практично всі операції з завданнями, за кожне з яких відповідає окрема процедура.

Важливою функціональністю програми є відображення завдань на графічному інтерфейсі. Для цього використовуються відповідні компоненти, які взаємодіють із даними в базі даних та надають користувачеві зручний перегляд активних завдань та їх статусу.

Не менш важливим етапом є можливість видалення завдань, що вимагає відповідної обробки та оновлення бази даних. Після видалення програма автоматично оновлює відображення на графічному інтерфейсі.

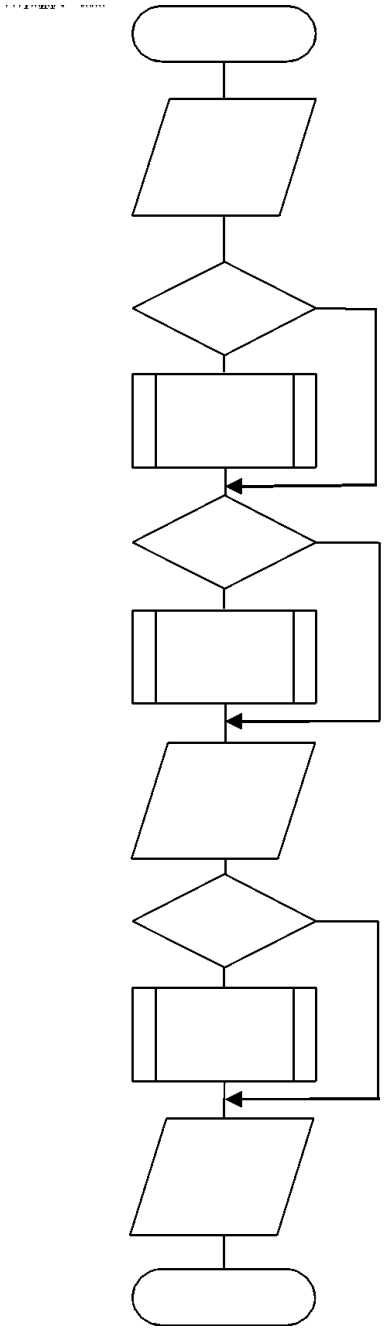

Рис. 3.3. Робота програми відслідковування задач та управління проектами

Передбачена можливість вибору користувачем різних параметрів для сортування та фільтрації завдань, що забезпечує зручну навігацію та швидкий доступ до необхідної інформації.

Крім того, програма реалізує можливість вибору користувачів зі спадного списку, які пов’язані із конкретними завданнями. Це враховує аспекти розподілу відповідальностей у команді та сприяє більш ефективному управлінню проєктами.

Завершальним етапом алгоритму є вихід з програми або збереження внесених змін. Під час виходу програма автоматично виконує необхідні операції для збереження даних та закриває з’єднання з базою даних.

3.2. Кількісна оцінка параметрів роботи сервісу для відслідковування задач та управління проєктом

Для розробки програмного забезпечення для управління проєктами було використано середовище програмування *Visual Studio,* мову програмування *C#* та базу даних для зберігання завдань проєкту.

Microsoft Visual Studio – це інтегроване середовище розробки, розроблене корпорацією *Microsoft,* яке відзначається високою популярністю серед програмістів та розробників по всьому світу. Ця платформа надає широкі можливості для розробки різних типів програмного забезпечення, починаючи від десктопних додатків і завершуючи веб-сервісами та мобільними додатками.

Однією з ключових переваг *Visual Studio* є підтримка різних мов програмування. Розробники можуть працювати з такими мовами, як *C++, C#, Visual Basic, Python,* і багатьма іншими. Це робить *Visual Studio* універсальним інструментом для різних проєктів та завдань.

У складі *Visual Studio* існує потужний редактор коду, який наділений функціями автодоповнення, підсвічування синтаксису, і підтримки відлагодження. Цей редактор спрощує написання коду і полегшує пошук та виправлення помилок.

Однією з ключових можливостей *Visual Studio* є інтегрований дебагер. Він дозволяє програмістам аналізувати та виправляти помилки у програмному коді під час його виконання, що є надзвичайно важливим для створення стабільних та надійних додатків.

Visual Studio також підтримує розробку графічних інтерфейсів для десктопних додатків за допомогою *Windows Forms* та *WPF,* що дозволяє створювати додатки зі сучасним та привабливим дизайном.

За допомогою платформи *Xamarin,* яка інтегрована в *Visual Studio,* розробники можуть створювати мобільні додатки для платформ *Android ma iOS*. Це робить *Visual Studio* цінним інструментом для розробки додатків для різних мобільних пристроїв.

Серед інших можливостей *Visual Studio* варто відзначити підтримку розробки веб-додатків з використанням технологій, таких як *ASP.NET, HTML, CSS, ma JavaScript*. Це дозволяє розробникам створювати сучасні та інтерактивні веб-сайти та додатки.

Окрім власних можливостей, *Visual Studio* підтримує різноманітні плагіни та розширення, які дозволяють розширити функціональність *IDE* і налаштувати її під конкретні потреби розробника.

У підсумку, *Microsoft Visual Studio* є потужним та універсальним інструментом для розробки програмного забезпечення різних видів і гарантує продуктивну роботу розробників завдяки широкому спектру функцій та інструментів, які надаються в цій інтегрованій середовищі розробки.

Мова програмування *C#* є однією з об'єктно-орієнтованих мов програмування, розробленою корпорацією *Microsoft* і вперше представленою в 2000 році. *C#* входить до родини мов програмування *C,* і його синтаксис має багато спільних рис з *C* і *C++*. Однак, він також має свої унікальні особливості та розширення.

Спроєктований як мова для розробки програмного забезпечення для платформи *Microsoft .NET, C#* відзначається сильною типізацією, що дозволяє створювати безпечні та надійні додатки. Вона підтримує два основних типи даних: значущі типи та посилальні типи.

Система типів *C#* дозволяє розробникам визначати власні класи та інтерфейси, використовуючи спадкування і реалізацію інтерфейсів, що сприяє створенню багатошарових та структурованих додатків.

Мова *C#* підтримує обробку винятків для обробки помилок під час виконання програми. Це дозволяє розробникам створювати надійні програми, які можуть відповідати на непередбачені ситуації.

Однією з ключових особливостей *C#* є автоматичне управління пам'яттю. Мова використовує систему збору сміття, яка дозволяє розробникам уникнути ручного вивільнення пам'яті та управління неявними посиланнями.

C# також підтримує делегати і події, що дозволяє створювати подійно-орієнтовані програми та реалізовувати шаблони проєктування, такі як спостерігач.

Важливою частиною *C#* є доступ до бібліотек класів *.NET Framework,* які містять багато готових рішень та функцій для розробників, що спрощує створення додатків.

Мова *C#* має вбудовану підтримку паралельного програмування, що дозволяє розробникам створювати ефективні додатки, які використовують багатоядерні процесори.

Загалом, *C#* є мовою програмування з багатою функціональністю та широким спектром можливостей, яка підходить для різних типів додатків, від десктопних програм до веб-сервісів та мобільних додатків. Вона надає розробникам інструменти для створення високоякісного програмного забезпечення, що відповідає сучасним стандартам розробки.

База даних (БД) - це організована колекція інформації, яка зберігається, керується та доступна для операцій обробки даних. Бази даних відіграють важливу роль у сучасних інформаційних системах та є основою для зберігання та обробки великих обсягів даних.

Бази даних поділяються на кілька основних видів: реляційні, нереляційні і об'єктно-орієнтовані бази даних..

Реляційні бази даних використовують модель реляцій, в якій дані представлені у вигляді таблиць (реляцій). Кожна таблиця складається з рядків і стовпців, де кожен стовпець має свою назву та тип даних. Відомим прикладом реляційної СУБД є *Microsoft SQL Server, Oracle, MySQL та PostgreSQL*.

Реляційні бази даних (РБД) є фундаментальною концепцією в області управління базами даних і використовуються для зберігання та організації даних в багатьох сферах, від бізнесу та науки до медицини та інформаційних технологій. Теорія реляційних баз даних базується на принципах, вперше визначених Едгаром Коддом у 1970 році, і включає в себе кілька ключових аспектів.

Основні поняття реляційних баз даних включають у себе таблиці (реляції), які представляють собою двовимірні матриці з рядками і стовпцями. Кожен рядок таблиці відображає запис, а кожний стовпець відображає конкретну атрибут або поле. Ключі, такі як первинний та зовнішній, визначають унікальні ідентифікатори та зв'язки між таблицями.

Нормалізація бази даних – це процес, який допомагає усунути аномалії та покращити ефективність бази даних, розбиваючи таблиці на менші, взаємопов'язані структури. Цей процес включає в себе першу, другу, третю та інші нормальні форми.

Мова структурованих запитів (*SQL*) грає важливу роль у роботі з реляційними базами даних. *SQL* містить команди для визначення та модифікації структури бази даних, такі як *CREATE TABLE, ALTER TABLE* і *DROP TABLE,* а також команди для вставки, оновлення та видалення даних, такі як *SELECT, INSERT, UPDATE* та *DELETE*.

Транзакції та контроль цілісності даних визначають основні принципи, що забезпечують цілісність та надійність даних під час операцій з базою. Крім того, оптимізація запитів та використання індексів є важливим етапом у роботі з реляційними базами даних.

Застосування реляційних баз даних у сучасному світі розповсюджене в різних галузях. У бізнесі вони використовуються для відстеження клієнтів, замовлень, фінансів тощо. У науці – для зберігання та обробки великих обсягів наукових даних. В медичній інформатиці РБД відповідають за управління медичними записами, тестами та іншою інформацією. Інтеграція з іншими інструментами розробки та постійна підтримка дозволяють реляційн ні бази даних залишатися актуальними та ефективними в різноманітних сценаріях використання.

Процес оптимізації та управління реляційними базами даних також включає в себе розгляд застосування мови *SQL,* яка є стандартною мовою для взаємодії з РБД. *SQL* дозволяє виконувати різноманітні завдання, починаючи від створення та модифікації структур даних до отримання, фільтрації та агрегації інформації.

У сучасному програмуванні та розробці програмного забезпечення використання реляційних баз даних стає стратегічно важливою складовою. Вони дозволяють забезпечити структуроване зберігання даних, зручний доступ до них та ефективне виконання операцій з обробки інформації. Реляційні бази даних є основою для багатьох застосунків, включаючи веб-розробку, бізнес-застосунки, аналітику даних та інші області.

Розглянемо роль транзакцій та контролю цілісності даних у реляційних базах даних. *ACID*-властивості (атомарність, консистентність, ізольованість та довершеність) гарантують, що операції з базою даних відбуваються надійно та безпечно, забезпечуючи цілісність і стабільність даних.

Застосування реляційних баз даних у сучасному світі розповсюджене в різних галузях. У бізнесі вони використовуються для відстеження клієнтів, замовлень, фінансів тощо. У науці – для зберігання та обробки великих обсягів наукових даних. В медичній інформатиці РБД відповідають за управління медичними записами, тестами та іншою інформацією. Інтеграція з іншими інструментами розробки та постійна підтримка дозволяють реляційній базі даних залишатися актуальною та ефективною в різноманітних сценаріях використання.

У майбутньому можливості розвитку реляційних баз даних включатимуть у себе посилення їхньої ефективності, забезпечення підтримки великого обсягу даних та адаптації до зростаючих вимог сучасних застосунків.

Нереляційні бази даних використовують різні моделі для організації та зберігання даних, такі як ключ-значення, стовпці, документи та графи. Вони часто використовуються для зберігання великих обсягів неструктурованих або напівструктурованих даних. Прикладами *NoSQL*-систем є *MongoDB, Cassandra, Redis та Neo4j*.

Нереляційні бази даних (*NoSQL*) є важливою складовою сучасних інформаційних технологій, які дозволяють зберігати та обробляти дані з новим підходом. У відміну від традиційних реляційних баз даних, *NoSQL* пропонують гнучкі та масштабовані рішення для різноманітних завдань, включаючи великі обсяги даних, розподілені системи та потужні аналітичні завдання.

Основні принципи *NoSQL*:

– гнучкі схеми: *NoSQL* бази даних дозволяють зберігати дані без фіксованої структури, що дозволяє легко вносити зміни до схеми без необхідності перебудови всієї бази даних. Це особливо корисно в ситуаціях, коли структура даних може змінюватися з часом або в разі роботи з неструктурованими даними;

– горизонтальне масштабування: *NoSQL* бази даних легко масштабуються горизонтально, додаючи нові сервери або вузли до системи. Це дозволяє обробляти великі обсяги даних та велику кількість одночасних запитань;

– швидкий доступ до даних: У багатьох *NoSQL* системах використовуються спрощені моделі доступу до даних, такі як ключ-значення, документи чи стовпці, що дозволяє забезпечити швидкий доступ до конкретних елементів даних.

Типи *NoSQL* баз даних;

– ключ-значення (*Key-Value*) бази даних: цей тип бази даних використовує структуру ключ-значення для зберігання даних. Кожен елемент має унікальний ключ, за яким його можна ідентифікувати.

– документ-орієнтовані бази даних: використовують документи (зазвичай у форматі *JSON* або *BSON*) для зберігання та організації даних. Документ може містити різні типи інформації.

– стовпцеві (*Wide-Column*) бази даних: дані в таких базах даних організовані у вигляді стовпців, а не рядків. Це дозволяє ефективно зчитувати та аналізувати великі обсяги даних.

– графові бази даних: використовують структуру графів для представлення та зберігання даних, особливо корисні для моделювання зв’язків між об’єктами.

NoSQL бази даних широко використовуються у великому спектрі сфер, включаючи веб-розробку, аналітику даних, мобільні додатки та інші галузі, де

необхідна гнучкість та швидкодія. Вони стають невід'ємною частиною сучасної архітектури додатків, дозволяючи ефективно вирішувати завдання обробки великих обсягів і розподілених даних.

Об'єктно-орієнтовані бази даних використовують об'єктну модель для зберігання та управління даними. Вони ідеально підходять для додатків, які використовують об'єктно-орієнтований підхід у програмуванні. Продуктом цього типу є *ZODB (Python), db4o (Java), Versant* (*C++*).

Інколи важливо виділити також спеціалізовані бази даних, які призначені для конкретних галузей або завдань. Прикладами є географічні бази даних, біологічні бази даних, медичні бази даних тощо.

Об'єктно-орієнтовані бази даних (*OODB*) є важливим елементом сучасних систем управління базами даних, в яких використовується об'єктно-орієнтований підхід для організації та зберігання інформації. Цей підхід дозволяє моделювати дані у вигляді об'єктів, об'єднуючи дані та методи їх обробки в єдиному змістовному контексті.

У порівнянні з традиційними реляційними базами даних, об'єктно-орієнтовані бази даних надають кілька ключових переваг.

Гнучкість та розширюваність: *OODB* дозволяють легко змінювати структуру даних, додавати нові атрибути та методи без необхідності модифікації всієї бази даних. Це сприяє адаптації до змін у вимогах системи.

Підтримка комплексних структур даних: *OODB* можуть зберігати складні об'єкти, включаючи зв'язки між ними, у вигляді цілих графів об'єктів. Це дозволяє більш точно відображати реальні структури даних.

Наслідування та поліморфізм: *OODB* підтримують концепції наслідування та поліморфізму, що дозволяє створювати ієрархії класів об'єктів та використовувати їх у більш гнучких та ефективних способах.

Об'єктно-орієнтовані бази даних широко використовуються в різних областях.

Програмування та розробка програмного забезпечення: *OODB* дозволяють розробникам працювати з даними у термінах об'єктів, що спрощує взаємодію з базою даних та полегшує процес програмування.

Графічні системи та комп’ютерна графіка: *OODB* використовуються для зберігання графічних об’єктів, їх взаємодії та атрибутів, що дозволяє створювати складні графічні інтерфейси та сцени.

Системи управління базами даних для об’єктно-орієнтованих мов програмування: *OODB* використовуються для забезпечення ефективної підтримки об’єктно-орієнтованих мов програмування, таких як *Java* або *C#*.

Розподілені бази даних розміщені на різних серверах або вузлах та можуть бути розташовані в різних географічних регіонах. Вони використовуються для забезпечення доступу до даних з різних місць та для підтримки великих завдань обробки даних.

Тимчасові бази даних зберігають дані, що існують лише на протязі обмеженого часу. Вони використовуються для тимчасового зберігання даних під час виконання операцій або завдань.

Інтернет-розподілені бази даних спроєктовані для забезпечення доступу до даних через Інтернет. Вони дозволяють розробникам створювати веб-сервіси та додатки, які можуть обмінюватися даними через мережу.

Бази даних використовуються у різних галузях, включаючи бізнес, науку, медицину, фінанси, соціальні мережі та інші. Вони є важливою складовою сучасних інформаційних систем і дозволяють зберігати, оновлювати, отримувати та аналізувати дані для прийняття рішень та виконання різних операцій.

Бази даних використовуються для зберігання інформації про клієнтів, товари, угоди та інші аспекти діяльності компанії. Це допомагає в управлінні взаємодією з клієнтами, прогнозуванні попиту на товари та послуги, аналізі ефективності маркетингових стратегій та прийнятті стратегічних управлінських рішень.

У науці та дослідженнях бази даних використовуються для зберігання великих обсягів даних, що дозволяє вченим здійснювати експерименти, виконувати аналіз результатів досліджень та долучати нові знання до існуючих теорій. Бази даних також використовуються для підтримки спільної роботи вчених над проєктами та обміну даними між науковими групами.

У медицині бази даних грають ключову роль у зберіганні медичної інформації про пацієнтів, діагнози, призначення лікування та інші клінічні дані.

У фінансовому секторі бази даних використовуються для зберігання фінансових транзакцій, клієнтських даних, управління портфелем та ризиками. Це допомагає фінансовим установам забезпечувати точність фінансової звітності, виявляти аномалії та шахраївську діяльність.

Для розробки бази даних системи відслідковування задач було створено таблиці, що представлені нижче (табл. 3.1-3.3).

Таблиця 3.1

Структура таблиці “Задачі”

Назва поля	Тип даних	Призначення
ID	*INT*	Унікальний ідентифікатор задачі
Назва	*VARCHAR* (255)	Назва задачі або опис
Дата створення	*DATETIME*	Дата та час створення задачі
Статус	*VARCHAR* (50)	Поточний статус задачі
Відповідальний	*INT*	Ідентифікатор користувача, відповідального за задачу
Термін виконання	*DATETIME*	Дедлайн для задачі, дата, до якої вона повинна бути завершена
Пріоритет	*VARCHAR* (20)	Важливість задачі
Категорія	*VARCHAR* (50)	Категорія або тег задачі

Таблиця 3.2

Структура таблиці “Користувачі”

Назва поля	**Тип даних**	**Призначення**
ID	*INT*	Унікальний ідентифікатор користувача
Ім’я	*VARCHAR* (50)	Ім’я користувача
Прізвище	*VARCHAR* (50)	Прізвище користувача
Електронна пошта	*VARCHAR* (255)	Адреса електронної пошти користувача

Таблиця 3.3

Структура таблиці “Журнал подій”

Назва поля	**Тип даних**	**Призначення**

ID	*INT*	Унікальний ідентифікатор запису в журналі
Дата і час	*DATETIME*	Дата та час події
Користувач	*INT*	Ідентифікатор користувача, який виконав дію
Дія	*VARCHAR* (100)	Опис дії (наприклад, створення, оновлення, видалення задачі)

Для системи відслідковування задач було використано наступні зв’язки між таблицями:

1. Зв’язок між таблицею "Задачі" і "Користувачі"

Поле "Відповідальний" в таблиці "Задачі" пов’язане з полем "*ID*" в таблиці "Користувачі". Цей зв’язок вказує на те, який користувач відповідає за виконання кожної задачі. Це дозволяє визначити, хто відповідає за кожну конкретну задачу.

2. Зв’язок між таблицею "Задачі" і "Журнал дій":

Поле "*ID*" в таблиці "Задачі" пов’язане з полем "*ID*" в таблиці "Журнал дій". Цей зв’язок дозволяє відстежувати дії, які виконуються з кожною задачею. Наприклад, коли задача створюється, оновлюється або видаляється, відповідний запис додається до таблиці "Журнал дій" з вказівкою на задачу.

Структура бази даних представлена на рис. 3.4.

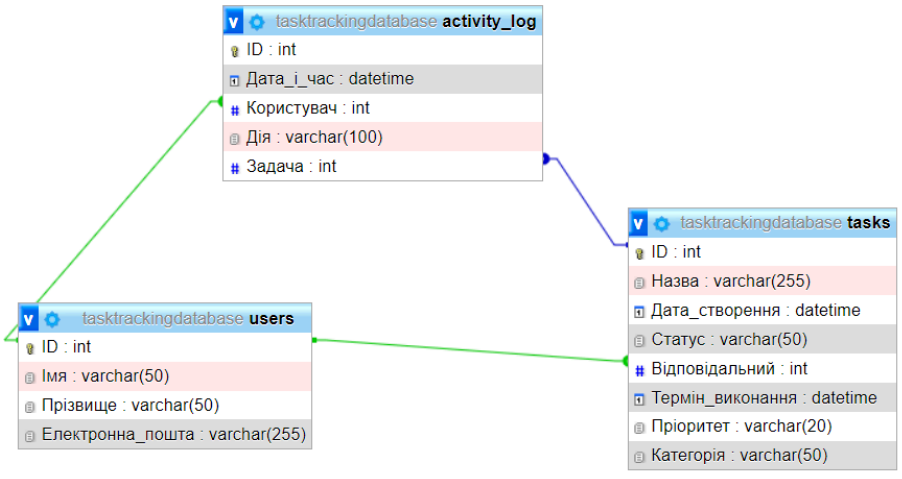

Рис. 3.4. Зв’язки між таблицями бази даних

Такі зв’язки допомагають відстежувати, які користувачі відповідають за які задачі і які дії виконуються з кожною задачею, що робить систему відслідковування задач більш зручною та інформативною.

Структура програми для відслідковування задач включає:

1. Користувацький інтерфейс:

– дашборд: головна сторінка програми, на якій користувач може переглядати свої задачі та загальний статус системи;

– список задач: сторінка, де користувач може переглядати всі свої задачі, фільтрувати їх за різними параметрами і вносити зміни;

– створення задачі: форма для створення нової задачі, в якій користувач вказує назву, опис, термін виконання, відповідального і інші атрибути;

– редагування задачі: форма для редагування існуючої задачі;

– журнал дій: сторінка, на якій користувач може переглядати історію дій, пов’язаних з задачами.

2. Серверна частина:

– база даних: база даних для зберігання інформації про користувачів, задачі та журнал дій;

– *API*: серверний *API* для обробки запитів від клієнтського додатку, включаючи створення, оновлення, видалення і запити про задачі та користувачів.

3. Логіка додатку:

– модель даних: код, що представляє структуру даних, таку як користувачі, задачі та журнал дій;

– контролери: логіка обробки запитів від користуача та взаємодії з базою даних через *API*;

– сервіси: додаткові функції, такі як нагадування про терміни виконання, фільтрація та сортування списку задач;

– авторизація користувача: можливість надавати права доступу до певних функцій і даних в залежності від ролі користувача.

4. Нагадування та сповіщення:

– система нагадувань: можливість надсилання користувачам нагадувань про наближені терміни виконання задач;

– повідомлення по електронній пошті: автоматичне надсилання повідомлень користувачам по електронній пошті.

5. Зв’язок з базою даних: код для взаємодії з базою даних для отримання, оновлення та зберігання даних.

Ця структура програми допомагає організувати розробку системи відслідковування задач, забезпечити її функціональність, безпеку та надійність.

Роботу інструменту відслідковування задач можна детальніше розібрати на прикладі розробленої програми. При запуску програми користувач потрапляє на форму, що представлена на рис. 3.5.

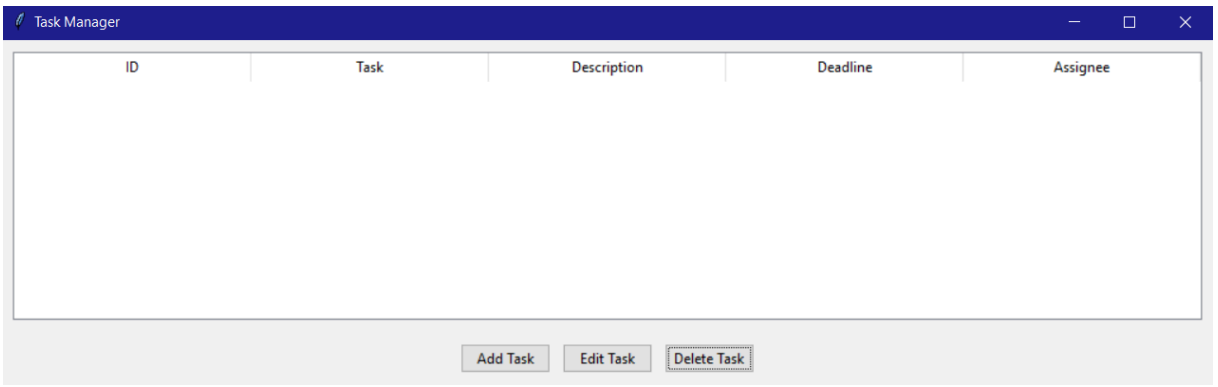

Рис. 3.5. Головне вікно програми

У цьому вікні користувачі мають змогу переглядати список завдань, редагувати їх, та видаляти. До завдання можна прикріпити та присвоїти дедлайн.

Додавання нової задачі представлено на рис. 3.6.

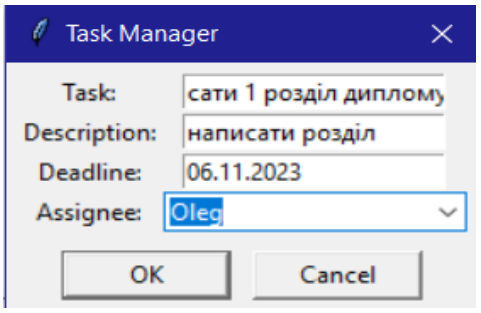

Рис. 3.6. Процес додавання задачі до програми

В результаті виконання дій, що зазначені на рис. 3.7 та натискання кнопки “*Add task*” програма збереже завдання в списку завдань.

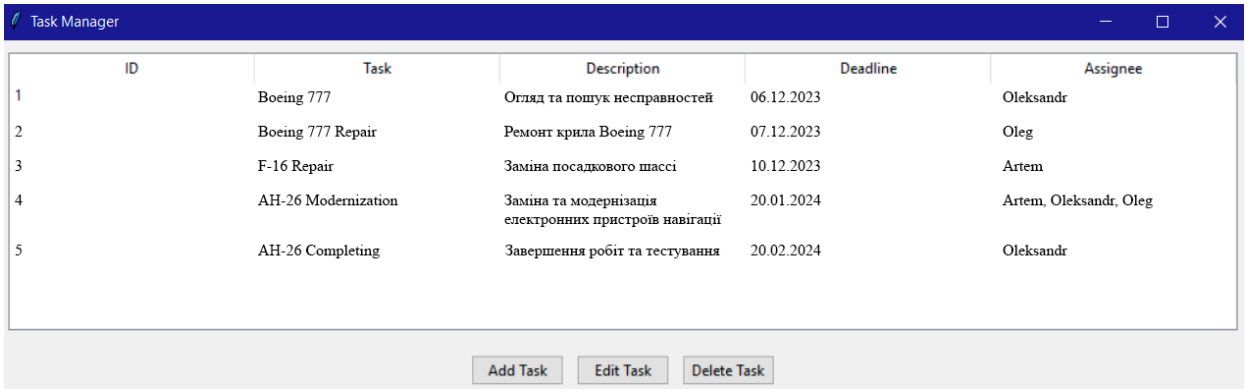

Рис. 3.7. Результат додавання завдання

Для редагування завдання потрібно натиснути кнопку “*Edit task*” та внести необхідні зміни (рис. 3.8).

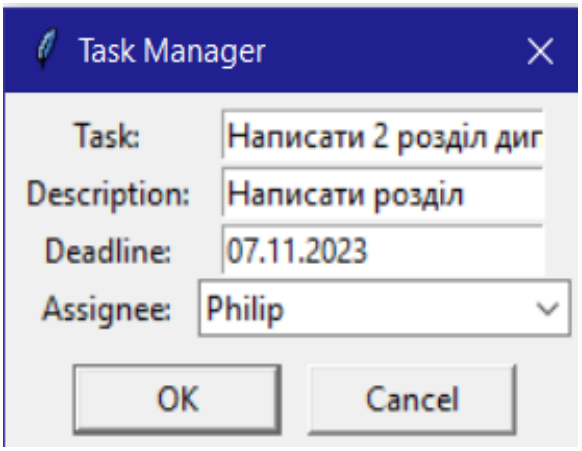

Рис. 3.8. Редагування завдання

Після внесення змін користувач отримає редаговане завдання (рис. 3.9).

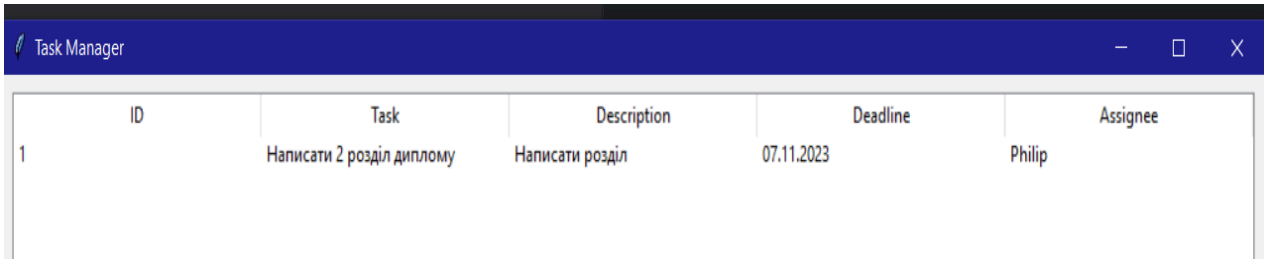

Рис. 3.9. Результат зміни завдання

Для видалення завдання потрібно натиснути кнопку “*Delete task*” після чого воно видалиться з програми (рис. 3.10).

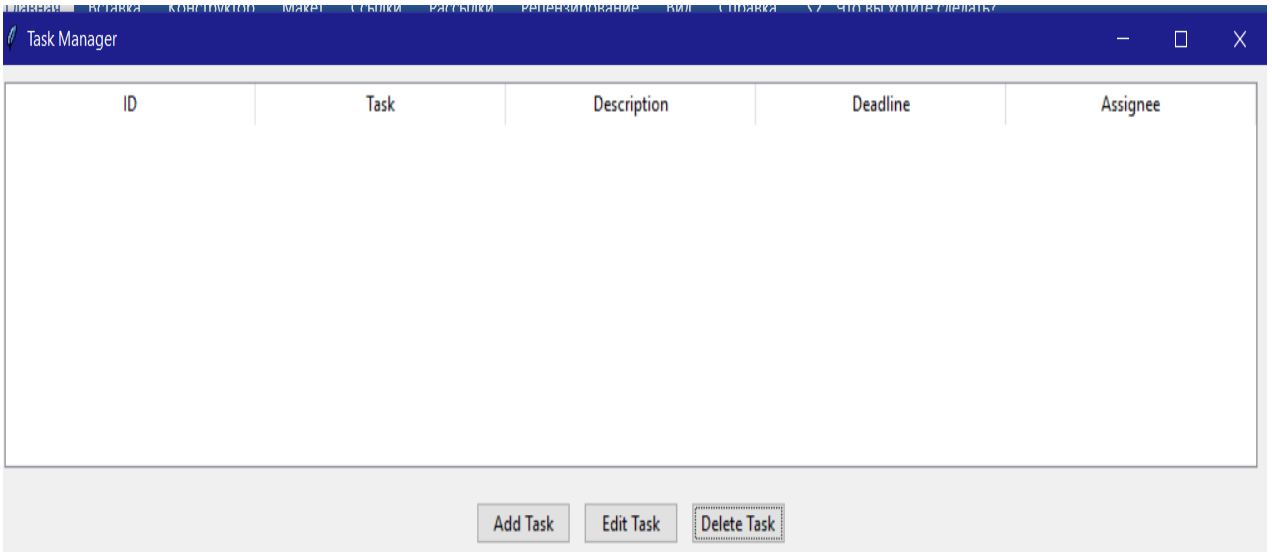

Рис. 3.10. Видалення завдання з програми

Розроблену програму було оцінено за кількома критеріями, результат оцінки представлено у табл. 3.4.

Таблиця 3.4

Оцінка параметрів програмного продукту.

Параметр	Оцінка (від 1 до 5)	Пояснення
Зручність користування	4	Графічний інтерфейс простий та зрозумілий. Інтуїтивно зрозумілі елементи управління.
Функціональність	3	Базові функції працюють добре, але можливості розширення функціональності бажані.
Дизайн інтерфейсу	4	Графічний інтерфейс виглядає професійно та приємно для користувача.
Ефективність	3	Програма працює ефективно для невеликої кількості завдань

Пояснення до таблиці:

– зручність користування: оцінка 4 вказує на те, що інтерфейс є зручним та легким у використанні, проте може бути ще дещо поліпшений для забезпечення ще більшого комфорту користувачам;

– функціональність: оцінка 3 вказує на те, що програма надає базові функції, але є простір для покращень та розширення можливостей, щоб відповідати різним потребам користувачів;

– надійність: оцінка 4 свідчить про стабільність програми та невелику ймовірність виникнення системних помилок чи збоїв;

– дизайн інтерфейсу: оцінка 4 вказує на те, що графічний інтерфейс є естетично приємним та відповідає професійним стандартам;

– ефективність: оцінка 3 вказує на те, що програма працює ефективно для невеликої кількості завдань, але може вимагати оптимізації для обробки великої кількості даних.

Порівняльна харатеристика програми відносно інших систем представлена в табл. 3.5.

Таблиця 3.5

Порівняння систем управління задачами

Параметр	Програма	*Jira*	*Microsoft Project*
Зручність користування	4	3	4
Функціональність	4	4	3
Надійність	4	4	3
Дизайн інтерфейсу	3	4	3

Пояснення до таблиці:

– зручність користування: програма має добре зрозумілий інтерфейс, але сервіси *Jira* та *Microsoft Project* відзначаються високою зручністю користування, зокрема *Jira* визначається високим рівнем інтуїтивності;

– функціональність: програма має базові функції, але сервіси *Jira та Microsoft Project* пропонують більше розширені можливості для управління завданнями, спільною роботою, відстеженням часу тощо;

– надійність: програма і сервіси *Jira ma Microsoft Project* демонструють схожий рівень надійності, при цьому *Microsoft Project* визначається високою стабільністю;

– дизайн інтерфейсу: програма і сервіси *Jira* та *Microsoft Project* мають адекватний та естетичний дизайн інтерфейсу.

– ефективність: сервіси *Jira* та *Microsoft Project* виграють у розділі ефективності, оскільки вони призначені для роботи з великою кількістю завдань та проєктів, а наша програма може потребувати оптимізацій для такого масштабу.

Щоб покращити програму та зрівняти її з великими сервісами, можна розглянути наступні кроки:

– розширення функціональності – додати розширені можливості, такі як фільтрація, сортування, відстеження часу, спільна робота, графіки, розклади та інші аспекти управління завданнями;

– оптимізація ефективності – вдосконалити алгоритми обробки даних та реалізувати оптимізації для роботи з великим обсягом інформації;

– підвищення зручності користування – здійснити аналіз користувацького досвіду та внести зміни в інтерфейс для підвищення зручності користування.

– інтеграція з іншими інструментами – додати можливість інтеграції з іншими популярними інструментами розробки та управління проєктами для забезпечення більшої гнучкості та функціональності.

– підтримка великих проєктів – розглянути можливості для оптимізації роботи з великими проєктами та завданнями для забезпечення ефективної роботи навіть при великій кількості даних.

Реалізація цих покращень дозволить програмі конкурувати з великими інструментами управління проєктами та забезпечить задоволення від використання для різноманітної аудиторії користувачів.

3.3. Висновки до розділу

Розроблено програмний продукт для управління задачами представляє собою ефективний інструмент для простого та зручного керування завданнями. Графічний інтерфейс демонструє непоганий рівень зручності користування, забезпечуючи інтуїтивність та доступність. Програма стабільна та надійна, демонструючи надійність у роботі.

Відмінною особливістю створеного додатка є те, що він за домогою алгоритма кластеризації можливе ефективний розподіл та групування завдань проєкта, без втручання людини. Також підключення в майбутньому найрізноманітших модулів, за допомогою яких можливе більш гнучке та ефективне відслідковування та управління задачами.

Хоча функціональність базова, програма успішно вирішує основні завдання з додавання, редагування та видалення завдань. Проте, для розширення можливостей і конкуренції з великими аналогами, є потреба у додаткових функціях, таких як фільтрація, спільна робота, та відстеження часу.

Дизайн інтерфейсу є естетичним та адекватним, проте можливості покращення інтерфейсу для поліпшення зручності користування ще не вичерпані. Програма також має потенціал для оптимізації роботи з великою кількістю завдань, щоб забезпечити ефективність при обробці великого обсягу даних.

У майбутньому планується вдосконалити програму, додаючи розширені функціональні можливості та оптимізуючи її для використання в різних сценаріях управління проєктами. Все це дозволить програмі стати більш гнучкою та конкурентоспроможною на ринку інструментів для управління завданнями.

ВИСНОВКИ

В рамках вивчення та розробки інструментів відслідковування задач та управління проєктами, було визначено основні аспекти розробки та ефективного використання подібного програмного забезпечення. На основі проведеного аналізу ринку та потреб користувачів, було прийнято рішення розробити власний інструмент для керування завданнями, орієнтований на вимоги та потреби специфічної групи користувачів.

Було розглянено ключові характеристики та властивості існуючих рішень, таких як *Jira ma Microsoft Project,* щоб визначити найбільш важливі аспекти для реалізації власного інструменту. Однією з ключових вимог було забезпечення зручності користування та інтуїтивності графічного інтерфейсу, адаптованого для технічно орієнтованої аудиторії.

Процес розробки програми включав в себе створення бази даних з використанням *SQLite* для ефективного зберігання та управління завданнями. Кожен етап розробки був визначений на основі вимог та функціональності, що вимагала програма. Результатом став не лише функціональний інструмент, але й стабільна та ефективна програма, яка може бути використана для ведення проєктів різного масштабу.

Важливою складовою процесу розробки було порівняння отриманого рішення з існуючими сервісами на ринку. Оцінка за кількома ключовими параметрами, такими як зручність користування, функціональність, надійність, дизайн інтерфейсу та ефективність, дозволила визначити конкурентні переваги та недоліки розробленого програмного продукту.

У подальших напрямках розвитку програми розглядаються можливості розширення функціональності для поліпшення взаємодії користувачів з інструментом, оптимізації для роботи з великим обсягом даних та інтеграції з іншими інструментами розробки, що підвищить його гнучкість та універсальність. В цілому, розробка інструменту для відслідковування задач та управління проєктами є

результатом вдалих зусиль у напрямку вдосконалення робочих процесів та забезпечення ефективного керування проєктами. Цей інструмент становить важливий внесок у сфері управління завданнями, враховуючи специфічні потреби користувачів та високі вимоги до продуктивності.

Окрім того, важливою частиною розробки є постійна підтримка та вдосконалення програмного продукту. Це включає в себе вислуховування фідбеку користувачів, виявлення можливостей для оновлень та впровадження нововведень для вдосконалення функціоналу та досвіду використання. Такий цикл постійного удосконалення дозволяє забезпечити, що програмний продукт відповідає зростаючим потребам користувачів та залишається конкурентоспроможним на ринку.

У майбутньому можливість розширення функціональності для вдосконалення взаємодії з іншими інструментами розробки стане важливим кроком у розвитку інструменту. Інтеграція з іншими популярними платформами та сервісами дозволить розширити можливості користувачів та забезпечити більш гнучке управління завданнями в різноманітних проєктних середовищах.

У результаті кваліфікаційної роботи було розроблено інструмент для відслідковування задач та управління проєктами, що враховує технічні потреби та вимоги користувачів, виступаючи як функціональна та ефективна альтернатива існуючим рішенням на ринку.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

2. ДСТУ 3008–95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – Введ. 1995-23-02. – ІПрІн, УкрІНТЕЇ, Головний відділ стандартизації Технічного центру НАН України, 1995. – *No*58, 88 с.

3. Андрєєва Т. Є. Проєктний менеджмент як засіб досягнення мети підприємства / Т.Є. Андрєєва, Т.Е. Петровська, Т.С. Титар // Вісник економіки транспорту і промисловості. – 2011. – № 34. – С. 364-370.

4. Батенко Л. П. Управління проєктами: навч. посіб. / Л. П. Батенко, О. А. Загородніх, В. В. Ліщинська. – К.: КНЕУ, 2003. – 231 с.

5. Ноздріна Л.В. Управління проєктами: підручник / Ноздріна Л.В., Ящук В.І., Полотай О.І./ За заг.ред.Л.В.Ноздріної. – К.: Центр учбової літератури, 2010. – 432с.

6. Верба В. А. Проєктний аналіз : підручник / В. А. Верба, О. А. Загородніх. — К. : КНЕУ, 2000. – 322 с.

7. Воркут Т. А. Проєктний аналіз : навч. посібник / Т. А. Воркут. – К. : Укр. центр духовної культури, 2000. – 428 с

8. Блага Н. В. Управління проєктами: навч. пос. Львів: Львівський державний університет внутрішніх справ, 2021. 152 с.

9. Ноздріна Л., Ящук В., Полотай О. Управління проєктами. Київ: Центр навчальної літератури, 2020. 432 с.

10. Строкань, О. В., Мірошниченко М.Ю. Управління ІТ-проєктами: лабораторний практикум. Мелітополь: Видавничо-поліграфічний центр “Люкс”, 2020. 135с.

11. Хігні Д. Основи управління проєктами. Харків: Фабула, 2020. 272 с.

12. Качан Г. М. Особливості курсу “Управління ІТ-проєктами” в закладах вищої освіти. Науковий часопис НПУ імені М.П. Драгоманова. Серія 2. Комп’ютерно-орієнтовані системи навчання. 2020. № 22 (29). С. 73–80.

13. *O’Reilly, M. JIRA* 7 *Administration Cookbook*. *Packt Publishing,* 2016. 432 с.

14. Сайт A*tlassian*. *URL*: *https*://*id.atlassian.com*/*login* (дата звернення: 15.10.2023).

15. *Albahari, J., Albahari, B*. *C#* 7.0 *in a Nutshell*: *The Definitive Reference*. *O’Reilly Media,* 2017. 1072 *p*.

16. *Skeet, J. C# in Depth*. *Manning Publications,* 2019. 528 *p*.

17. *Liberty, J., & MacDonald, B*. *Learning C#* 3.0: *Master the fundamentals of C#* 3.0. *O’Reilly Media,* 2009. 800 *p*.

18. *Troelsen, A*. *Pro C#* 2008 *and the .NET* 3.5 *Platform*. *Apress,* 2007. 1360 *p*.

19. *Richter, J. CLR via C#. Microsoft Press,* 2010. 896 *p*.

20. *Albahari, J., Albahari, B*. *C#* 6.0 *in a Nutshell*: *The Definitive Reference*. *O’Reilly Media,* 2015. 1136 *p*.

21. *Sharp, J., & Freeman, A*. *Pro C#* 7: *With .NET and .NET Core*. *Apress,* 2016. 1383 *p*.

22. *Balena, F*. *Programming C#*: *Building .NET Applications with C#*. *O’Reilly Media,* 2006. 864 *p*.

23. *Liberty, J., & MacDonald, B*. *Learning C#* 5.0: *Get up to speed with C#, the language of the .NET Framework*. *O’Reilly Media,* 2013. 862 *p*.

24. *Albahari, J., Albahari, B*. *C#* 5.0 *in a Nutshell*: *The Definitive Reference*. *O’Reilly Media,* 2012. 1060 *p*.

25. *Skeet, J. C# in Depth*. *Manning Publications,* 2008. 584 *p*.

26. *Lippert, E*. *C#* 4.0 *in a Nutshell*: *The Definitive Reference*. *O’Reilly Media,* 2009. 1056 *p*.

27. *Wagner, G*. *More Effective C#*: 50 *Specific Ways to Improve Your C#*. *Addison-Wesley,* 2010. 352 *p*.

28. *Sharp, J., & Nauman, Z. Microsoft Visual C# Step by Step. Microsoft Press,* 2017. 832 *p.*

29. *Liberty, J., & MacDonald, B. Learning C#* 4.0: *Develop the skills to get ahead in a rapidly changing job market. O'Reilly Media,* 2010. 848 *p.*

30. *Watson, R. Visual Studio* 2019 *For Dummies. For Dummies,* 2019. 432 *p.*

31. *Robinson, P., & West, D. C#* 7.0 *All-in-One For Dummies. For Dummies,* 2018. 840 *p.*

32. *Troelsen, A., & Japikse, P. Pro C#* 7: *With .NET and .NET Core. Apress,* 2016. 1552 *p.*

33. *Balena, F. Programming C#* 4.0: *Building Windows, Web, and RIA Applications for the .NET* 4.0 *Framework. O'Reilly Media,* 2010. 864 *p.*

34. *Price, J., & Gill, C. Professional C#* 7 *and .NET Core* 2.0. *Wrox,* 2019. 1440 *p.*

35. *Horgan, P. Professional C#* 4 *and .NET* 4. *Wrox,* 2010. 1536 *p.*

36. *Troelsen, A. Pro C#* 5.0 *and the .NET* 4.5 *Framework. Apress,* 2014. 1560 *p.*

37. *Sharp, J., & Nauman, Z. Microsoft Visual C# Step by Step. Microsoft Press,* 2016. 816 *p.*

38. *MacDonald, M. A. Pro ASP.NET* 3.5 *in C#* 2008. *Apress,* 2007. 1512 *p.*

39. *Albahari, J., Albahari, B. C#* 3.0 *in a Nutshell: A Desktop Quick Reference. O'Reilly Media,* 2007. 1002 *p.*

40. *Freeman, A., & Sweeney, K. C#* 8.0 *and .NET Core* 3.0 *– Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code. Packt Publishing,* 2019. 796 *p.*

41. *Troelsen, A. Pro C#* 2008 *and the .NET* 3.5 *Platform. Apress,* 2008. 1752 *p.*

42. *Nathan, A.* (2005). *C#* 2.0: *Practical Guide for Programmers. Morgan Kaufmann,* 2005. 944 *p.*

43. *Robinson, P., & West, D. C#* 8.0 *All-in-One For Dummies. For Dummies,* 2019. 912 *p.*

44. Що таке *Jira* і як з нею працювати. *URL*: *https*://*iampm.club*/*ua*/*blog*/*shho-take-jira-i-yak-z-neyu-praczyuvati*/ (дата звернення:01.11.2023).