**Державний торговельно-економічний університет**
**Кафедра інженерії програмного забезпечення та кібербезпеки**

# ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

**на тему:**

## «Програмний додаток контролю виконання завдань команди аутсорс-компанії»

Студента 4 курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» освітньої програми «Інженерія програмного забезпечення» _____ підпис студента — Руденка Вадима Юрійовича

Науковий керівник асистент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Хорольська Карина Вікторівна

Гарант освітньої програми кандидат технічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Рзаєва Світлана Леонідівна

КИЇВ – 2023

**Державний торговельно-економічний університет**

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

**Затверджую**
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

# Завдання
## на випускний кваліфікаційний проєкт студентові

Руденку Вадиму Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Програмний додаток контролю виконань завдання команди аутсорс-компанії»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

*Мета проєкту* розробка ефективного програмного додатку для контролю виконання завдань аутсорс-компанії.

*Об'єкт дослідження* процес управління виконанням завдань аутсорс-компанії .

*Предмет дослідження*,розробка програмного додатку контролю виконання завдання аутсорс-компанії.

4. Консультанти проєкту із зазначенням розділів, які консультують:

| Розділ | Консультант (прізвище, ініціали) | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ДОДАТКУ ДЛЯ КОНТРОЛЮ ВИКОНАННЯ ЗАВДАНЬ

1.1. Огляд існуючих рішень та програм в сфері управління виконання завдань

1.2. Опис технічних та функціональних характеристик додатків та програм управління виконання завдань аутсорс-компанії

1.3. Аналіз переваг і недоліків наявних рішень та програм

1.4. Висновок до розділу 1

РОЗДІЛ 2. ПРОБЛЕМАТИКА ТА ПОСТАНОВКА ЗАВДАННЯ

2.1. Визначення проблем, які вирішує розроблюваний програмний додаток для управління завданними аутсорс-компанії

2.2. Опис способу вирішення проблеми та його обґрунтування

2.3. Формулювання технічних вимог до розроблюваної програми

2.4. Висновок до розділу 2

РОЗДІЛ 3. ОПИС РОЗРОБКИ ТЕХНІЧНОГО РІШЕННЯ ПРОБЛЕМАТИКИ

3.1. Концептуальна модель бази даних

3.2. Макет програми

3.3. Опис архітектури програмного забезпечення

3.4. Опис розробки та використання алгоритмів та методів для програмного додатку управління завданними аутсорс-компанії

3.5. Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

| № пор. | Назва етапів випускного кваліфікаційного проєкту | Строк виконання етапів проєкту | |
|---|---|---|---|
| | | за планом | фактично |
| 1 | 2 | 3 | 4 |
| 1. | *Вибір теми випускного кваліфікаційного проєкту* | *21.09.2022* | *21.09.2022* |
| 2. | *Розробка та затвердження завдання на проєкт* | *14.11.2022* | *14.11.2022* |
| 3. | *Вступ та перелік літературних джерел* | *23.12.2022* | *23.12.2022* |
| 4. | *Розділ 1. Аналіз предметної області застосування додатку для контролю виконання завдань* | *27.01.2023* | *27.01.2023* |
| 5. | *Розділ 2. Проблематика та постановка завдання* | *03.03.2023* | *03.03.2023* |
| 6. | *Розділ 3. Опис розробки технічного рішення проблематики* | *14.04.2023* | *14.04.2023* |
| 7. | *Висновки* | *28.04.2023* | *28.04.2023* |
| 8. | *Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)* | *17.05.2023* | *17.05.2023* |
| 9. | *Підготовка автореферату та презентації доповіді* | *26.05.2023* | *26.05.2023* |
| 10. | *Попередній захист випускного кваліфікаційного проєкту* | *29.05.2023 – 02.06.2023* | *29.05.2023* |
| 11. | *Зовнішнє рецензування випускного кваліфікаційного проєкту* | *05.06.2023* | *05.06.2023* |
| 12. | *Здача прошитого випускного кваліфікаційного проєкту на кафедру* | *05.06.2023* | *05.06.2023* |
| 13. | *Публічний захист випускного кваліфікаційного проєкту* | *19.06.2023* | *19.06.2023* |

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Хорольська К.В.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Руденко В.Ю.
(прізвище, ініціали, підпис)

**11. Відгук керівника випускного кваліфікаційного проєкту**

Науковий керівник випускного кваліфікаційного проєкту

_____

*(підпис, дата)*

Відмітка про попередній захист _____

*(ПІБ, підпис, дата)*

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____ Руденка В.Ю.

*(прізвище, ініціали)*

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Рзаєва С.Л.

*(прізвище, ініціали, підпис)*

Завідувач кафедри _____ Криворучко О. В.

*(підпис, прізвище, ініціали)*

«_____»_____20 ___ р.

## АНОТАЦІЯ

Відповідно до мети дослідження, робота присвячена розробці програмного забезпечення для ефективного управління завданними аутсорс-компанії.

В результаті порівняльного аналізу аналогічних було визначено ключові функціональні та технічні вимоги до системи.

Розробка програмного забезпечення виконана у середовищі MS SQL Server за допомогою мови програмування C#, що забезпечило надійність та гнучкість системи.

Готовий програмний комплекс успішно протестовано та демонструє високу продуктивність, ефективність використання та зручність інтерфейсу.



**Ключові слова:** управління завданними, програмний додаток, MS SQL Server, C#, аутсорс-компанія, аналіз вимог, трьохрівнева архітектура, система управління завданними.



## ABSTRACT

According to the purpose of the research, the work is devoted to the development of software for the effective management of the tasks of the outsourced company.

As a result of the comparative analysis of similar ones, the key functional and technical requirements for the system were determined.

Software development was carried out in the MS SQL Server environment using the C# programming language, which ensured system reliability and flexibility.

The ready-made software complex has been successfully tested and demonstrates high performance, efficiency of use and user-friendliness of the interface.

**Keywords**: task management, software application, MS SQL Server, C#, business entity, requirements analysis, three-level architecture, task management system.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API - Application Programming Interface - Інтерфейс програмування додатків

DB - Database - База даних

GUI - Graphical User Interface - Графічний користувацький інтерфейс

ML.NET - Бібліотека машинного навчання для .NET

MS SQL Server - Система управління реляційними базами даних - СУБД, розроблена корпорацією Microsoft

MVC - Model-View-Controller - Модель-представлення-контролер, патерн проектування для реалізації користувацького інтерфейсу

CRUD - Create, Read, Update, Delete - Операції, що є основою більшості систем з веб-доступом до баз даних

UI - User Interface - Інтерфейс користувача

UX - User Experience - Досвід користувача

UML - Unified Modeling Language - Уніфікована мова моделювання

**ЗМІСТ**



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *ДТЕУ 121 06-20.БР* | | |
| *Зм.* | *Аркуш* | *№ докум.* | *Підпис* | *Дата* | | | |
| Зав. каф. | | Криворучко О.В. | | 23.12.22 | Програмний додаток контролю виконання завдань команди аутсорс-компанії | *Стадія* | *Арку* | *Аркушів* |
| Керівник | | Хорольська К.В. | | 23.12.22 | | *З* | *3* | *61* |
| Гарант | | Рзаєва С.Л. | | 23.12.22 | | *Факультет інформаційних технологій 4 курс, 6 група* | | |
| Розробив | | Руденко В.Ю. | | 23.12.22 | *Зміст* | | | |

# ВСТУП

*Актуальність.* В сучасному технологічному контексті, де стрімкість розвитку інформаційних технологій є невід'ємним аспектом, ефективне управління процесами перетворюється на одну з ключових детермінант успіху будь-якої аутсорс-компанії. Раціоналізація робочого процесу, суворий контроль та систематичне моніторинг виконання завдань визначають продуктивність організації, стимулюючи її конкурентні переваги на ринку. Відсутність або неефективність такого контролю можуть призвести до деградації продуктивності, простоїв у виконанні завдань і до потенційної втрати бізнесу.

Освітлюючи вищезгадане, актуальність розробки програмного додатку для контролю виконання завдань в аутсорс-компанії виступає незаперечною. Попит на такі системи поширюється в усіх сферах діяльності: від IT-компаній до послуг, промисловості та освіти. Проективований додаток спрямований на оптимізацію робочого процесу, покращення комунікації в команді та підвищення продуктивності праці.

Сучасний спектр програмних продуктів, представлених для управління проектами та завданнями, включає широкий вибір інструментів, зокрема Trello, Asana, Jira та інші. Ці інструменти мають розширений набір функцій, проте можуть виявитися складними для користувачів без технічної освіти або не відповідати специфічним вимогам окремих аутсорс-компаній.

Інтегральність, простота використання, адаптація до вимог конкретного бізнесу та можливість інтеграції з іншими системами стануть ключовими відмінностями представленого програмного додатка. Основні технічні характеристики продукту включають гнучкість налаштувань, інтуїтивно

зрозумілий інтерфейс, здатність ефективно відстежувати процеси та стан завдань, а також систему сповіщень для своєчасного виконання завдань. Програмний додаток буде розроблено з використанням сучасних технологій, що забезпечать його надійність, швидкість роботи та безпеку даних.

*Об'єкт дослідження*: процес управління виконанням завдань у аутсорс-компанії.

*Предмет дослідження*: розробка програмного додатку контролю виконання завдання аутсорс-компанії.

*Мета проекту*: розробка ефективного програмного додатку для контролю виконання завдань аутсорс-компанії, який би враховував специфіку та потреби її працівників.

*Завдання проекту*:

– проаналізувати існуючі програмні рішення;

– визначити основні вимоги до програмного додатка;

– розробити технічні характеристики та функціональні можливості додатка;

– розробити програмний додаток і провести його тестування.

Практична значущість проекту полягає в розробці програмного додатку, який допоможе покращити процес управління завданнями, збільшити продуктивність та ефективність роботи аутсорс-компанії.

Рівень вивченості теми є досить високим, оскільки існує багато наукових робіт, що досліджують проблеми управління завданнями та різні аспекти використання програмних додатків для цього. Однак, незважаючи на велику кількість існуючих рішень, не всі вони враховують специфіку та вимоги аутсорс-компаній. Це включає в себе вимоги до гнучкості, швидкості реагування на зміни, можливості адаптації до специфічних умов виконання завдань.

Враховуючи це, існує потреба в подальшому науковому дослідженні та практичній розробці програмних додатків для контролю виконання завдань аутсорс-компанії, які би враховували ці специфічні вимоги та тенденції.

Наукова новизна дослідження полягає в розробці інтуїтивно зрозумілого та привабливого інтерфейсу користувача, що спрощує процес управління завданнями та їх контролю.

Практичне значення дослідження включає:

– значне підвищення продуктивності роботи аутсорс-компанії завдяки автоматизації управління завданнями;

– ефективніше використання ресурсів завдяки автоматизованому розподілу завдань і плануванню роботи;

– покращення внутрішньої комунікації за допомогою інтегрованих інструментів комунікації, що полегшують обмін інформацією і координацію завдань;

– зменшення ризику помилок та недорозумінь, що часто виникають при ручному управлінні завданнями.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАСТОСУВАННЯ ДОДАТКУ ДЛЯ КОНТРОЛЮ ВИКОНАННЯ ЗАВДАНЬ

## 1.1. Огляд існуючих рішень та програм в сфері управління виконання завданнями

У сучасному бізнес-оточенні ефективне управління проектами та завданними є одним із ключових факторів успіху. Аутсорс-компанії, які працюють в умовах високої конкуренції та динамічного ринку, стикаються з необхідністю оптимізувати процеси управління завданними, що включає планування, координацію, контроль та аналіз виконання завдань. Проте, багато організацій стикаються з рядом проблем, які обмежують їхню ефективність, таких як відсутність централізованого управління завданними, труднощі з координацією між відділами та співробітниками, недостатня видимість процесу виконання завданьта ін.

Проблема управління завдань в організаціях була та залишається предметом дослідження багатьох вчених та практиків. На сьогодні існує значна кількість наукових робіт, статей, методик та інструментів, спрямованих на вирішення цієї проблеми. Проте, через особливості кожної аутсорс-компанії, не існує універсального рішення, яке б могло бути ефективно застосоване в будь-якій організації.

В процесі підготовки до розробки програмного додатку для контролю виконання завдань було проведено вивчення різноманітних джерел, що включає наукову літературу, нормативні документи та інші матеріали.

Для розуміння концепцій та методології управління проектами та завданними було вивчено численні наукові роботи. Книга [1] автора Гарольда Керзнера дає всебічний огляд управління проектами, включаючи планування, організацію, реалізацію та контроль проектів. Додатково, було вивчено роботу [2] Девіда Аллена, яка пропонує ефективний метод управління часом та завданними.

У контексті управління проектами та завданними, було вивчено ряд нормативних документів, зокрема гайдлайни [3]. Цей документ є стандартом у галузі управління проектами та містить набір рекомендацій та кращих практик.

Вивчення інших матеріалів, таких як блоги, вебінари, подкасти та онлайн-курси з управління проектами та завданними, дало можливість отримати актуальні знання про останні тенденції у сфері управління проектами, включаючи впровадження Agile-методологій, ефективне комунікування в команді та технології для автоматизації процесів, та переглянути різні підходи до цієї теми.

Проведене дослідження дало можливість отримати глибоке розуміння проблеми недостатньої ефективності і організованості процесу управління завданнями в аутсорс-компаніях, яку має вирішити програмний додаток, і підготувало ґрунт для формулювання вимог до додатку та його подальшої розробки.

В рамках цієї роботи розробляється програмний додаток для управління завданними аутсорс-компанії. Додаток має на меті вирішити низку проблем, пов'язаних з процесом управління завданнями, включаючи відсутність централізованого управління завданнями, труднощі з координацією між відділами та співробітниками, недостатню видимість процесу виконання завдань тощо.

Додаток має забезпечувати наступні функції:

– створення, редагування та видалення завдань;

– призначення виконавців завдань та встановлення термінів виконання;

– відслідковування статусу завдань (наприклад, «до виконання», «у процесі», «завершено»);

– організацію завдань за допомогою проектів та дошок;

– централізовану комунікацію щодо завдань в рамках додатку;

– аналіз ефективності виконання завдань та створення звітів.

Для аналізу рішень проблеми управління завданнями аутсорс-компанії, потрібно розглянути наступні основні підходи:

– використання спеціалізованих програмних продуктів. Це охоплює системи управління проектами, такі як Microsoft Project, Jira, Trello, Asana, та інші. Ці системи надають розширений набір інструментів для планування, контролю та відстеження завдань, але вони можуть бути надмірно складними для деяких організацій або можуть не враховувати специфіку певного бізнесу.

– розробка внутрішніх систем управління завданнями. Деякі організації вирішують проблему, створюючи власні внутрішні системи для управління завданнями. Це дозволяє повністю адаптувати систему під специфіку бізнесу, але це також вимагає значних витрат часу та ресурсів на розробку та підтримку системи;

– використання загальних інструментів організації роботи. Включає використання електронних таблиць, електронної пошти або інших загальних інструментів для управління завданнями. Цей підхід може бути простим і недорогим, але він часто не надає необхідного рівня контролю та видимості, які потрібні для ефективного управління завданнями.

Кожен з цих підходів має свої переваги та недоліки.

– спеціалізовані програмні продукти надають потужні інструменти для управління проектами та завданнями, але вони можуть бути складними у використанні, а також вимагають значних витрат на налаштування та обслуговування. Крім того, вони можуть не враховувати специфіку конкретного бізнесу або галузі;

– внутрішні системи дозволяють максимально адаптуватися під потреби організації, але їх розробка та підтримка вимагає значних витрат часу та ресурсів. Це також може бути ризиковано, оскільки в разі проблем з системою вся відповідальність лягає на організацію;

– загальні інструменти можуть бути простими у використанні та недорогими, але вони не забезпечують необхідного рівня контролю та видимості для ефективного управління завданнями. Крім того, вони можуть бути неефективними при великих об'ємах завдань або при роботі в команді.

Тому, розробка власного програмного додатку, який би враховував специфіку аутсорс-компанії, має великі переваги. Він може бути простим у використанні, адаптованим під специфіку бізнесу та надавати необхідний рівень контролю та видимості завдань.

## 1.2. Опис технічних та функціональних характеристик додатків та програм управління виконання завдань аутсорс-компанії

Сьогодні існує велика кількість програмних продуктів, які спрямовані на полегшення управління завданнями. Ці системи включають такі відомі продукти, як Asana, Trello, Jira та інші. Кожен з цих продуктів має свої унікальні особливості, але всі вони дозволяють користувачам створювати, присвоювати та відслідковувати завдання.

Розглянемо детальніше технічні та функціональні характеристики деяких з цих систем.

*Asana*

Asana є веб та мобільним застосунком для керування проектами та спільної роботи в команді (див. рис. 1.1). Застосунок пропонує широкий спектр функцій та можливостей для організації завдань, спільної комунікації та відстеження прогресу.

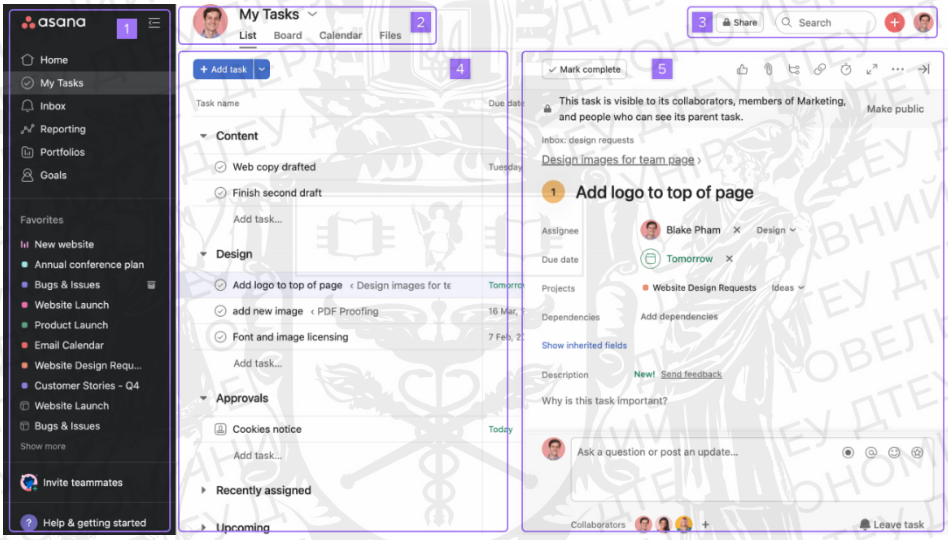


Рис. 1.1. Інтерфейс користувача додатку «Asana»

*Джерело: знімок з екрану [9]*

Основне призначення Asana полягає в полегшенні керування проектами і забезпеченні ефективної спільної роботи в команді [4]. Застосунок дозволяє створювати проекти, завдання, підзадачі та призначати їх учасникам команди. Кожному завданню можуть бути присвоєні терміни виконання, пріоритети та теги для категоризації.

Архітектура Asana заснована на хмарному розрахунку, що дозволяє користувачам отримувати доступ до своїх проектів та даних з будь-якого

пристрою з Інтернет-підключенням. Інтерфейс Asana інтуїтивно зрозумілий та легкий у використанні, що дозволяє командам швидко оволодіти системою та розпочати роботу.

Основні функції Asana включають [5]:

– завдання та проекти. Користувачі можуть створювати завдання, групувати їх у проекти та організовувати структуру робочих потоків. Це дозволяє відстежувати роботу, встановлювати пріоритети та надсилати сповіщення про зміни;

– календар та терміни. Надає календарний перегляд, де можна відображати завдання та терміни виконання. Це допомагає командам керувати графіками та планувати робочі процеси;

– спільна комунікація. Дозволяє командам обговорювати завдання, обмінюватися коментарями та прикріплювати файли. Користувачі можуть вести дискусії в рамках кожного завдання або проекту, спрощуючи комунікацію та зберігання всіх відповідних даних в одному місці;

– відстеження прогресу. Дозволяє відстежувати прогрес виконання завдань, встановлювати статуси та надсилати сповіщення про зміни. Користувачі можуть легко відслідковувати, які завдання вже виконані, які знаходяться в процесі та які ще не розпочаті. Це допомагає забезпечити прозорість та контроль над проектом;

– дошки та перегляди. Надає гнучкість в організації завдань за допомогою дошок та різних переглядів. Користувачі можуть використовувати дошки Kanban для переміщення завдань між колонками статусів або використовувати інші перегляди, такі як списки або календарі, для відображення завдань по-різному;

– звіти та аналітика. Дозволяє генерувати звіти та аналітичні дані про прогрес проектів. Це дає можливість оцінювати продуктивність, витрати

часу, завдання, що чекають виконання, та інші показники для покращення ефективності робочих процесів.

Загалом, Asana є потужним інструментом для організації, спільної роботи та керування проектами. Він допомагає командам ефективно співпрацювати, відстежувати прогрес та досягати поставлених цілей.

*Trello*

Trello – система управління проектами, яка використовує методологію Kanban. Вона дозволяє користувачам створювати дошки (boards), списки та картки для організації та відслідковування завдань (див. рис. 1.2).

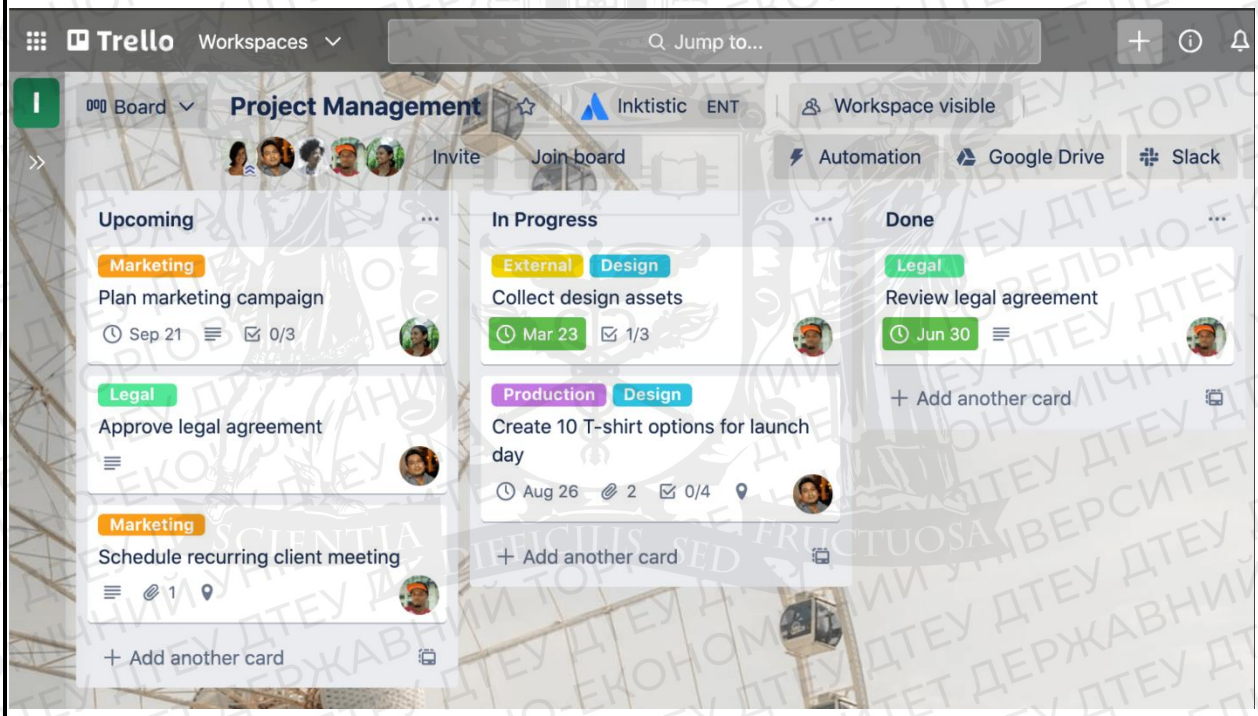


Рис. 1.2. Інтерфейс користувача додатку «Trello»

*Джерело: Знімок з екрану* [10]

Основне призначення Trello полягає в створенні віртуальних дошок, на яких можна створювати картки для представлення окремих завдань [6]. Картки

потім розміщуються на дошці залежно від їх статусу або категорії, що допомагає візуалізувати робочі процеси та прогрес проекту.

Архітектура Trello базується на хмарному розрахунку, що дозволяє користувачам отримувати доступ до своїх дошок та карток з будь-якого пристрою. Інтерфейс Trello є простим та інтуїтивно зрозумілим, з фокусом на простоті використання та візуальній організації даних.

Основні функції Trello включають [7]:

– дошки та картки. Користувачі можуть створювати необмежену кількість дошок та додавати картки на них. Картки можна надавати назву, опис, мітки та додавати прикріплені файли. Картки можуть бути переміщені між списками, відображаючи їх статус або етап виконання;

– списки та статуси. Картки розташовуються в списках, які можуть представляти стадії виконання завдань, наприклад, «В очікуванні», «В процесі», «Завершено». Користувачі можуть легко переміщати картки між списками, відображаючи прогрес роботи;

– коментарі та обговорення. Користувачі можуть обговорювати картки, залишати коментарі та обмінюватися відгуками в команді. Це дозволяє легко спілкуватися щодо конкретних завдань, обговорювати деталі та робити замітки;

– учасники та призначення. Користувачі можуть додавати учасників до карток, призначати відповідальних за їх виконання та встановлювати терміни. Це допомагає забезпечити відповідальність та розподіл завдань в команді;

– оповіщення та нагадування. Надає можливість налаштовувати оповіщення та нагадування щодо термінів, коментарів або змін, що стосуються карток;

– інтеграції. Підтримує широкий спектр інтеграцій з іншими інструментами та сервісами. Це дозволяє підключати зовнішні додатки, такі як календарі, зберігання файлів, спілкування та інші, для полегшення робочих процесів та покращення продуктивності.

Загалом, Trello є потужним інструментом для організації проектів та спільної роботи в команді. Він надає просту, але ефективну систему дошок та карток для відстеження завдань, спілкування та управління робочими процесами.

*Jira*

Jira - це популярний інструмент для керування проектами та управління завданнями, розроблений спеціально для розробників програмного забезпечення (див. рис. 1.3). Він використовується для організації робочих процесів, відстеження прогресу завдань і спільної роботи в команді.

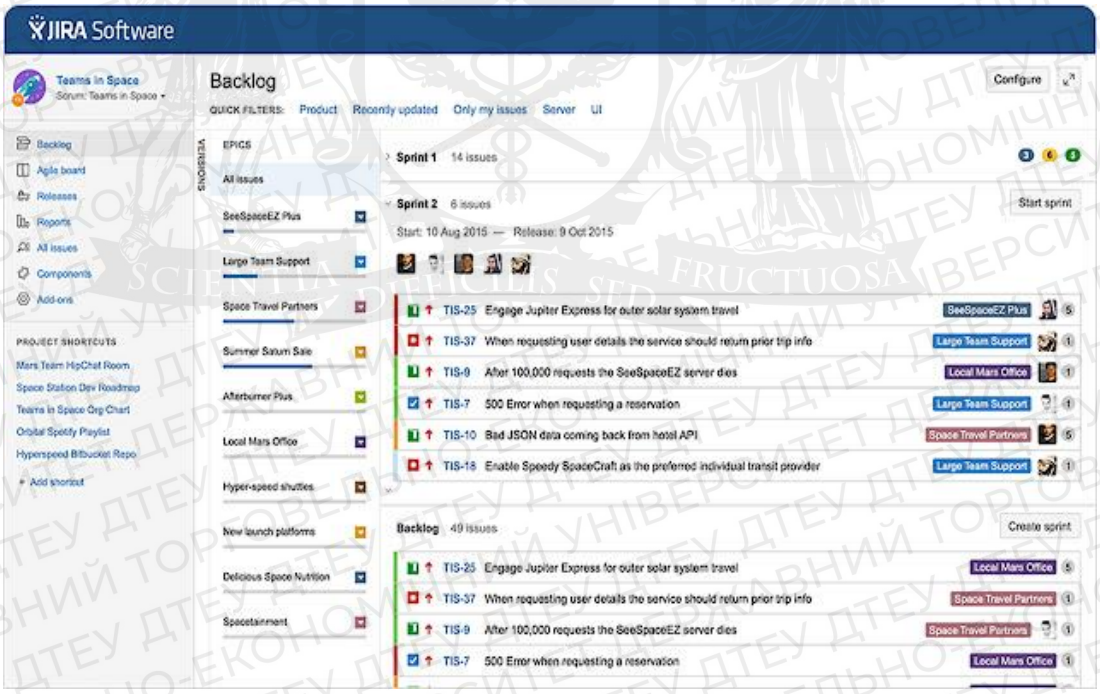


Рис. 1.3. Інтерфейс користувача додатку «Jira»

*Джерело: Знімок з екрану* [11]

Основне призначення Jira полягає в спрощенні керування проектами, відстеженні помилок та управлінні завданнями розробки програмного забезпечення [8]. Це допомагає розробникам та командам ефективно працювати над проектами, збільшувати продуктивність та забезпечувати якість продукту.

Архітектура Jira заснована на серверній і хмарній моделі. Вона включає серверну частину, яка зберігає дані та забезпечує функціонал, та клієнтські додатки або веб-інтерфейс для доступу користувачів до системи.

Основні функції Jira включають:

– створення завдань. Користувачі можуть створювати завдання, описувати їх, встановлювати пріоритети та призначати відповідальних. Завдання можуть відображати нові функції, помилки або інші завдання, що потребують виконання;

– спринти та дошки. Дозволяє організовувати робочі процеси за допомогою спринтів та дошок. Спринти визначають обмежений часовий проміжок для виконання завдань, а дошки відображають стан завдань та їх прогрес;

– відстеження прогресу. Надає засоби для відстеження прогресу завдань та проектів. Користувачі можуть оновлювати статус завдань, додавати коментарі, вказувати залежності та встановлювати терміни виконання;

– управління помилками. Дозволяє відстежувати та керувати помилками в проекті. Користувачі можуть створювати та відстежувати помилки, присвоювати їм пріоритети, вказувати статуси виправлення та виконувати інші дії, пов'язані з усуненням помилок;

– управління версіями. Дозволяє керувати різними версіями продукту або програмного забезпечення. Користувачі можуть визначати та

відстежувати релізи, планувати функціональність для кожної версії та встановлювати терміни випуску;

– звіти та аналітика. Надає можливості створення звітів та аналітичних даних про прогрес проектів, продуктивність команди, витрати часу та інші показники. Це допомагає керівникам проектів отримувати уявлення про продуктивність та ефективність робочих процесів.

Загалом, Jira є потужним інструментом для керування проектами, спеціально розробленим для розробників програмного забезпечення. Він допомагає організовувати завдання, відстежувати прогрес, управляти помилками та сприяє ефективній спільній роботі в команді.

Також одним з важливих аспектів є можливість інтеграції з іншими системами, які вже використовуються на підприємстві, такими як системи документообігу, CRM-системи, системи обліку тощо.

### 1.3. Аналіз переваг і недоліків наявних рішень та програм

Використання програмних додатків для управління проектами та завданнями стало важливою частиною успішного ведення бізнесу. Вибір відповідного інструменту є ключовим для підвищення продуктивності та ефективності роботи.

Метою цього розділу є аналіз переваг та недоліків наявних на ринку рішень, що допоможе зрозуміти, які можливості вони пропонують, та які проблеми вони можуть вирішити. Це допоможе визначити, які функціональні та технічні характеристики має містити програмний додаток для управління виконанням завдань аутсорс-компанії, що планується розробити.

Для аналізу обрано такі популярні інструменти, як Asana, Trello та Jira, кожен з яких має свої унікальні особливості та характеристики.

*Asana*

Переваги даного інструменту включають такі аспекти:

– інтуїтивний і легкий у використанні. Має простий та зрозумілий інтерфейс, що полегшує користування і впровадження в команді;

– широкий функціонал. Пропонує багато функцій, таких як завдання, проекти, календарі, коментарі, звіти та інші, що забезпечують різноманітні можливості для організації та керування проектами;

– спільна робота та комунікація. Дозволяє командам спілкуватися, обговорювати завдання, обмінюватися файлами та координувати роботу в одному місці. Це сприяє ефективності командної роботи та зменшенню залежності від електронної пошти чи інших комунікаційних каналів;

– гнучкість та налаштування. Дозволяє налаштовувати проекти, завдання, терміни виконання та різні параметри відповідно до потреб команди.

Недоліки Asana:

– висока ціна. Може бути відносно дорогим в порівнянні з іншими аналогами. Вартість підписки може бути обмежуючим фактором для невеликих команд або приватних користувачів;

– обмежена функціональність безкоштовної версії. Безкоштовна версія має обмежені можливості та функції, і для доступу до деяких додаткових функцій може знадобитися платна підписка;

– складний пошук та фільтрація. Деяким користувачам може здатися, що пошук та фільтрація завдань в Asana не є настільки потужними або зручними, як в інших інструментах. Великі команди або проекти з великою кількістю завдань можуть знайти це обмеженням.

Отже, Asana є потужним інтуїтивно зрозумілим інструментом для керування проектами та спільної роботи в команді. Його широкий функціонал, легкість використання та можливості спільної комунікації роблять його

привабливим для багатьох користувачів. Однак, слід враховувати його високу ціну, обмежену функціональність без платної версії, обмежені можливості пошуку та фільтрації, а також складнощі зі синхронізацією з іншими інструментами.

*Trello*

Переваги Trello включають:

– простий та інтуїтивно зрозумілий інтерфейс. Відрізняється простотою використання і легкістю навігації, що дозволяє швидко розпочати роботу та мінімізує необхідність в навчанні користувачів;

– гнучкість в організації завдань. Заснований на концепції дошки з картками, що дозволяє користувачам гнучко структурувати та організовувати завдання відповідно до їх потреб та робочих процесів;

– візуалізація робочого процесу. Завдяки дошкам та колонкам, Trello дозволяє візуалізувати стан завдань та прогрес робочого процесу;

– проста спільна робота та комунікація. Надає можливість коментувати картки, взаємодіяти з командою та обмінюватися інформацією, сприяючи ефективній спільній роботі та комунікації;

– розширення та інтеграції. Підтримує різні розширення та інтеграції з іншими інструментами, такими як календарі, зберігання файлів, чат-програми та багато іншого.

Недоліки Trello:

– обмежена функціональність. У порівнянні з іншими інструментами для керування проектами, Trello може бути менш функціональним у плані деталей та можливостей, особливо для великих та складних проектів;

– відсутність розширених засобів звітності. Не надає вбудованих засобів для генерації детальних звітів та аналітики про прогрес проекту, продуктивність команди або витрати часу;

– відсутність розширених можливостей керування завданнями. Має обмежені можливості для керування завданнями, такі як відсутність спеціалізованих функцій для призначення пріоритетів, встановлення термінів, роботи зі залежностями та іншого;

– обмежена можливість керувати доступом. Відсутня деталізована система управління доступом, що може бути обмеженням для команд, які потребують більш жорсткої контролю над правами доступу до проектів та карток.

Отже, Trello - це простий та інтуїтивно зрозумілий інструмент для керування проектами, що пропонує гнучкість в організації завдань, візуалізацію робочих процесів та просту спільну комунікацію. Він має свої переваги у простоті використання та широкому розповсюдженні, але також має обмеження у функціональності, засобах звітності та керуванні завданнями для великих та складних проектів.

*Jira*

Переваги Jira включають наступне:

– потужний інструмент для керування проектами. Надає широкий функціонал для керування завданнями, відстеження прогресу, управління помилками та планування ресурсів. Він спеціально розроблений для потреб розробників програмного забезпечення та великих проектів;

– гнучкість та налаштування. Дозволяє гнучко налаштовувати проекти, завдання, ролі, права доступу та інші параметри відповідно до потреб команди. Це дозволяє вирішувати унікальні вимоги та налаштовувати робочі процеси.

– висока розширюваність. Має багато розширень та інтеграцій з іншими інструментами, що дозволяє розширити його функціонал та інтегрувати з існуючими робочими потоками;

– вбудована звітність та аналітика. Надає розширені засоби звітності та аналітики, які дозволяють генерувати звіти про прогрес проекту.

Недоліки Jira складаються із:

– висока складність для новачків. Складний для новачків та користувачів без технічного досвіду. Він має значну кількість функцій та може потребувати часу для вивчення та освоєння;

– висока ціна. Може бути витратним інструментом, особливо для невеликих команд або приватних користувачів. Ліцензія та підтримка можуть становити значну частку витрат на проект;

– можливість перевантаження. Зі зростанням обсягу даних та складності проектів, Jira може відчувати перевантаження. Великі команди або проекти з великою кількістю завдань можуть стикатися зі зменшенням продуктивності та швидкості роботи з системою.

Загалом, Jira є потужним інструментом для керування проектами з широким функціоналом, гнучкістю налаштування та вбудованою звітністю. Такий застосунок найбільш підходить для розробників ПЗ та великих проектів. Проте, варто враховувати складність для новачків, високу ціну та можливість перевантаження при роботі з великим обсягом даних. При виборі Jira важливо врахувати потреби та специфіку вашої команди.

Аутсорс-компанії, особливо малі та середні підприємства, часто стикаються з проблемою відсутності централізованого механізму для відслідковування та управління завданнями. Це призводить до втрати продуктивності, недосягнення важливих термінів і відсутності загального розуміння стану проекту.

На основі проведеного аналізу можна зробити висновок, що аутсорс-компанії потрібен інструмент, який би дозволив їм ефективно управляти завданнями. Такий інструмент має враховувати специфіку роботи конкретного підприємства, бути гнучким, простим у використанні та доступним для всіх співробітників.

### 1.4. Висновок до розділу 1

У цьому розділі було проведено детальний аналіз предметної області, зокрема огляд та аналіз основних існуючих рішень у сфері управління завданнями, таких як Asana, Trello, Jira. Ці системи представляють собою потужні інструменти для організації робочих процесів, але вони також мають свої обмеження, особливо щодо специфічних вимог аутсорс-компанії.

Проведено опис технічних та функціональних характеристик додатків та програм управління виконання завдань. Розглядаючи основні функціональні характеристики виявлено, що не всі потреби аутсорс-компаній можуть бути задоволені стандартними інструментами. Це стосується, зокрема, завдання, які потребують специфічного контролю, індивідуального підходу до управління та оптимізації процесів.

Зважаючи на це, необхідна потреба в розробці спеціалізованого програмного додатка, який би був адаптований до конкретних потреб аутсорс-компанії. Цей додаток має створити значні переваги, включаючи підвищення продуктивності роботи, зменшення помилок, покращення комунікації та координації, а також забезпечити більш гнучке управління завданнями. Результати цього аналізу є важливим вихідним пунктом для подальшої розробки та впровадження програмного додатка для контролю виконання завдань аутсорс-компанії.

# РОЗДІЛ 2

# ПРОБЛЕМАТИКА ТА ПОСТАНОВКА ЗАВДАННЯ

## 2.1. Визначення проблем, які вирішує розроблюваний програмний додаток для управління завданнями аутсорс-компанії

Аутсорс-компанії часто стикаються з викликами, пов'язаними з ефективним управлінням завданнями. Серед основних проблем, які мають бути вирішені за допомогою розроблюваного програмного додатка, можна визначити наступні:

– відсутність централізованого управління завданнями. Багато аутсорс-компаній продовжують користуватися випадковими або неструктурованими методами для відстеження завдань, такими як електронні таблиці, електронна пошта або навіть фізичні дошки. Це може призвести до втрати інформації, недостатньої прозорості та неефективності;

– погана видимість процесу. Нездатність відстежувати статус завдань в реальному часі може призвести до затримок, неефективного використання ресурсів і, в кінцевому результаті, втрати прибутку;

– неадекватне розподіл завдань та ресурсів. Без автоматизованого рішення для управління завданнями може бути складно розподіляти завдання та ресурси ефективно, що може призвести до перевантаження деяких членів команди, в той час як інші можуть бути недостатньо зайняті;

– відсутність зворотного зв'язку та аналізу продуктивності. Керівники команд часто не мають засобів для отримання вчасного зворотного зв'язку від своїх команд або для визначення, які методи роботи є найефективнішими.

Враховуючи ці проблеми, розроблюваний програмний додаток має на меті створити централізовану систему для ефективного управління завданнями. Така система повинна дозволити користувачам легко створювати, призначати та відстежувати завдання, забезпечуючи повну прозорість процесу. Крім того, програма повинна мати функції для аналізу продуктивності та отримання зворотного зв'язку, щоб команди могли постійно вдосконалювати свої методи роботи.

### 2.2. Опис способу вирішення проблеми та його обґрунтування

Управління проєктами є комплексною методологією, спрямованою на ефективне керування процесами організації, планування, управління та координації ресурсів протягом усього життєвого циклу проєкту. Його основна мета полягає у досягненні визначених цілей проєкту шляхом застосування сучасних методів, технік і технологій управління.

Ключовим чинником успіху будь-якого проєкту є професійне управління ним на кожній стадії його життєвого циклу. Життєвий цикл проєкту охоплює період з моменту, коли розпочинається вкладення коштів у його реалізацію, до моменту ліквідації проєкту. Управління проєктом на кожній фазі циклу є критично важливим для досягнення успіху.

Перед реалізацією програмного рішення необхідно детально розглянути етапи управління проєктами, зосереджуючись на плануванні та реалізації проєкту.

1. Планування проєкту, включає наступне:

– ідентифікація - це процес визначення основних характеристик проєкту, його мети та обсягу. На цьому етапі з'ясовується, які завдання потрібно виконати, які ресурси необхідні, а також визначаються потенційні ризики і вигоди проєкту;

– планування - це процес розробки детального плану дій для досягнення цілей проєкту. На цьому етапі визначаються конкретні кроки, ресурси, терміни і бюджет, необхідні для виконання проєкту.

2. Реалізація проєкту, цей етап включає наступне - організація - на цьому етапі здійснюється організаційна структура проєкту. Визначаються ролі та відповідальності учасників проєкту, забезпечується необхідне обладнання та матеріали, а також встановлюються процедури звітності та комунікації.

Ці етапи є ключовими для успішного управління проєктами, оскільки вони дозволяють систематично організувати та планувати роботу, що веде до досягнення поставлених цілей.

Кожна з фаз управління проєктами включає керування різними аспектами:

– управління змістом проєкту. Визначення всіх параметрів, пов'язаних з розробкою, реалізацією та завершенням проєкту;

– управління часом реалізації проєкту. Встановлення часових рамок для всіх видів робіт на кожному етапі проєкту;

– управління якістю проєкту. Встановлення вимог і стандартів щодо якості результатів проєкту, а також постійний моніторинг та контроль для їх досягнення;

– управління вартістю. Оцінка та визначення витрат, джерел фінансування, планування грошових потоків, прогнозування фінансових результатів проєкту та контроль над ними;

– управління персоналом. Підбір персоналу та формування команди проєкту;

– управління ризиком. Ідентифікація ризиків та розробка відповідних стратегій для їх управління;

– управління комунікаціями. Забезпечення ефективної взаємодії між учасниками проєкту, передача інформації та забезпечення відповідного комунікаційного середовища;

– управління закупівлями. Планування закупівель, укладання угод, проведення тендерів та контроль за процесом закупівель.

На рисунку 2.1 зображено схематичне уявлення про сутність управління проєктами.

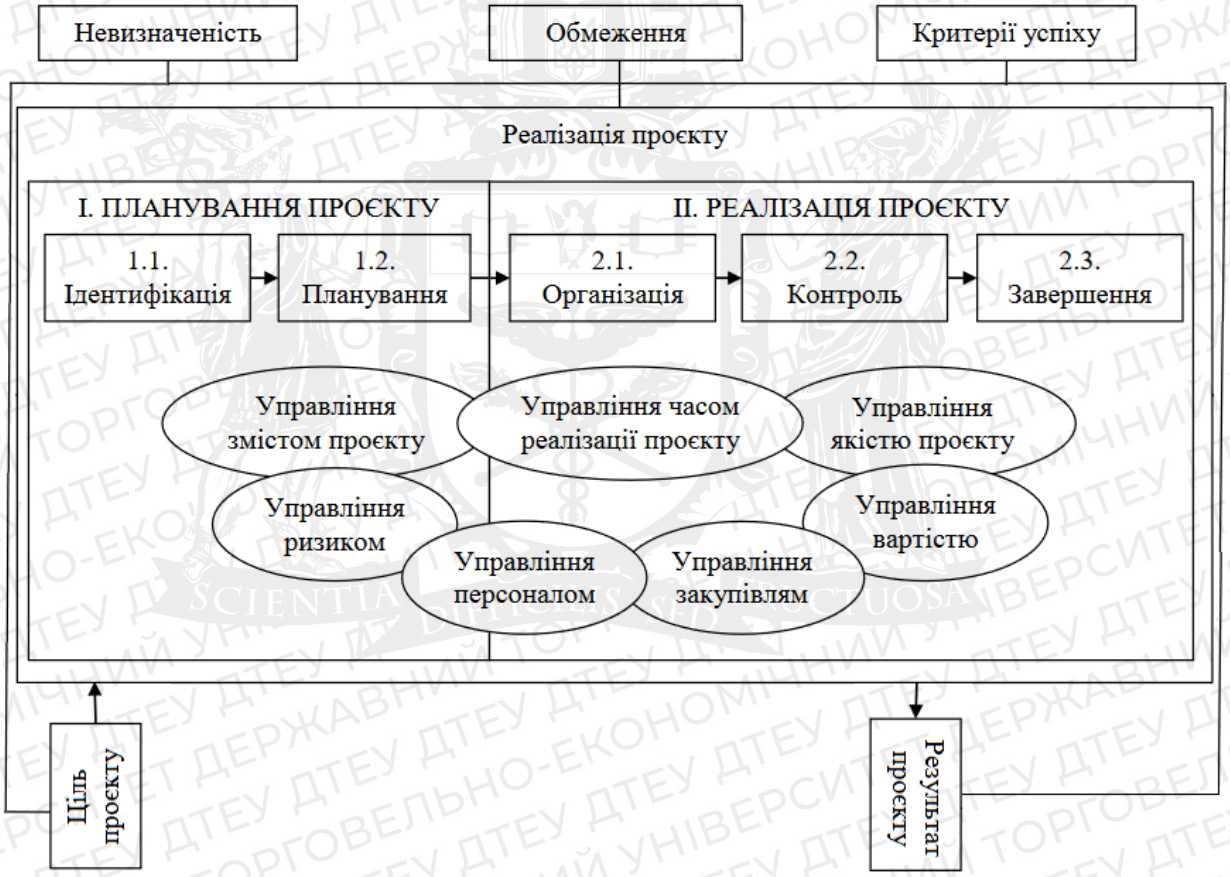


Рис. 2.1. Складові компоненти управління проєктами

*Джерело: побудовано автором*

Проєкт є системою, складеною з різних елементів, які визначають особливості його внутрішнього та зовнішнього середовища. Процес реалізації

проєкту пов'язаний з комунікацією всередині проєктної групи та між усіма елементами проєкту. Обмін інформацією між учасниками проєкту залежить від особливостей та елементів комунікаційного процесу.

Слід зазначити, що система управління проєктом включає два структурні елементи: об'єкт управління і суб'єкт управління або керуючу систему.

Об'єкт управління визначається відносинами між елементами системи, функціями, методами і процедурами виконання. Внутрішнє та зовнішнє середовище мають канали взаємодії на інформаційному рівні.

Суб'єкт управління проєктом має забезпечити досягнення прийнятної ефективності його реалізації, що можна характеризувати як досягнення стану рівноваги або стійкості.

Це схематичне зображення відображає ключові елементи та взаємозв'язки, які присутні в управлінні проєктами та сприяють досягненню його успішної реалізації.

Автоматизовані системи безсумнівно сприяють підвищенню якості реалізації проєкту, зокрема, шляхом прискорення обробки та передачі інформації та надання її у зручній формі. У сучасному розумінні, цифровий проєктний менеджмент охоплює процеси, що базуються на використанні віртуальної інфраструктури для планування, управління та контролю над діяльністю проєктної команди, яка може бути розподілена географічно та/або по часу.

Цифрові інструменти та системи управління проєктами дозволяють зберігати та обробляти великі обсяги даних, забезпечуючи швидкий доступ до актуальної інформації. Вони дозволяють командам спілкуватися та співпрацювати в режимі реального часу, незалежно від їх місцезнаходження. Крім того, цифрові інструменти можуть надавати графічну візуалізацію даних, що полегшує розуміння та аналіз інформації.

Цифровий проєктний менеджмент дозволяє ефективно координувати роботу розподіленої команди, забезпечувати зручний обмін даними та здійснювати ефективний контроль за виконанням завдань проєкту. Він допомагає знизити затримки, покращити комунікацію та підвищити продуктивність всієї команди, незалежно від її географічного розташування.

У сучасних дослідженнях цифровий проєктний менеджмент вивчається з різних напрямків, включаючи:

– інтелектуальне управління проєктами. цей напрямок використовує різні методи, такі як генетичні алгоритми, штучні нейронні мережі, метод опорних векторів, агентне моделювання, для ефективного управління проєктами;

– цифрове управління проєктами. Цей підхід в основному застосовується для створення або оптимізації веб-сайтів та мобільних додатків, і відрізняється від традиційного управління проєктами;

– віртуальне управління проєктами або «розподілена команда». Використовується, коли члени проєктної команди розташовані у різних географічних місцях, і дозволяє ефективно координувати їх роботу;

– автоматичне управління проєктами. Цей підхід використовується для створення звітів про управління проєктами для проєктів з програмного забезпечення. Він вимагає особливих вимог до представлення знань у середовищах програмного забезпечення;

– хмарне обчислення в управлінні проєктами. Використовуються різні моделі хмарних обчислень, такі як SaaS (програмне забезпечення як Сервіс), IaaS (інфраструктура як Сервіс), PaaS (платформа як Сервіс), для забезпечення потреб управління проєктами;

– електронне управління проєктами. Цей підхід дозволяє створити єдине сховище інформації про проєкт, що дозволяє членам команди отримувати доступ до даних в будь-якому місці за допомогою Інтернету.

Ці різні напрямки досліджень розширюють можливості управління проєктами за допомогою цифрових інструментів та технологій, сприяючи покращенню ефективності та результативності проєктів.

У практиці управління проєктами використовуються як універсальні, так і спеціалізовані програмні комплекси (див. рис. 2.2). Ці комплекси допомагають зберігати, оновлювати та аналізувати інформацію про проєкти, а також планувати ресурси, визначати залежності завдань, стежити за виконанням термінів та контролювати прогрес проєкту.

Кожен програмний комплекс має свої особливості та функціональні можливості, що дозволяють використовувати їх у різних галузях та в різних масштабах проєктів. Вибір конкретного програмного комплексу залежить від потреб і вимог конкретного проєкту та організації.

Використання таких програмних комплексів спрощує процес управління проєктами, полегшує планування та контроль, дозволяючи збільшити ефективність та досягати успіху в реалізації проєктів.

*Універсальні програмні комплекси*

Універсальні програмні комплекси управління проєктами є потужними інструментами, які надають широкий спектр функціональності для планування, виконання та контролю проєктів. Вони розроблені для застосування у різних галузях та для проєктів різних масштабів. Основна перевага універсальних програмних комплексів полягає в їх гнучкості та адаптивності до потреб користувача.

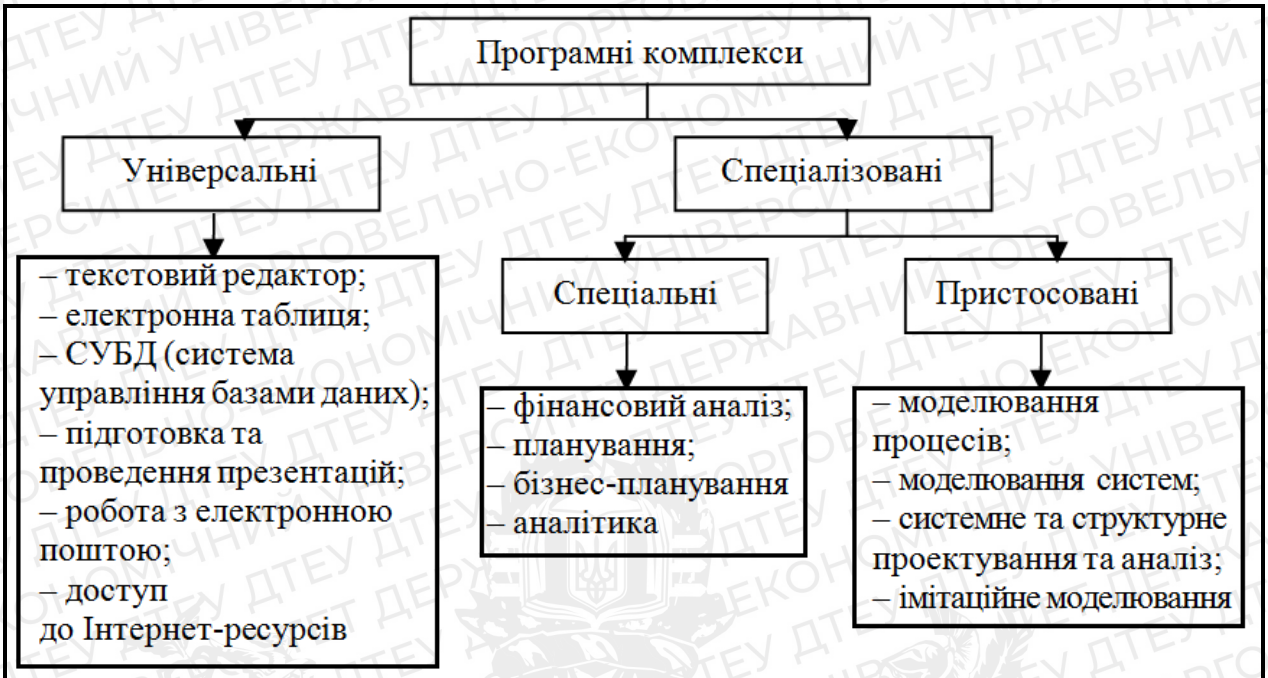


Рис. 2.2. Засоби управління проєктами

*Джерело: побудовано автором*

Ці програми надають можливість створювати та керувати розкладом проєкту, визначати послідовність завдань, призначати ресурси, відстежувати прогрес та контролювати виконання термінів. Вони також дозволяють створювати графіки, діаграми та звіти, що сприяє зрозумілому візуальному представленню інформації про проєкт.

Універсальні програмні комплекси часто мають можливості для спільної роботи над проєктом, спілкування команди, обміну документами та зберігання історії змін. Вони можуть підтримувати інтеграцію з іншими інструментами, такими як електронна пошта, календарі, системи управління завданнями тощо.

Популярні універсальні програмні комплекси управління проєктами включають Microsoft Project, Primavera P6, Wrike, Asana, Jira, Trello та багато інших. Кожен з них має свої особливості та можливості, і вибір залежить від потреб та вимог конкретного проєкту.

*Спеціалізовані програмні комплекси*

Спеціалізовані програмні комплекси управління проєктами призначені для конкретних галузей або типів проєктів, де вимагається специфічна функціональність та адаптація до особливостей процесів.

Ці програми надають спеціалізовані інструменти та функції, що відповідають вимогам конкретної галузі або виду проєктів. Вони можуть мати зручний інтерфейс та спеціальні шаблони для планування, виконання та контролю проєктів у цих галузях.

Наприклад, спеціалізовані програмні комплекси для будівництва та інженерних проєктів можуть мати функції для керування великими обсягами даних, планування ресурсів, відстежування стану будівельних робіт, координації підрядників та контролю за дотриманням нормативних вимог.

У специфічних галузях, таких як інформаційні технології або програмний розробка, використовуються спеціалізовані програми, що підтримують управління завданнями, відстеження прогресу розробки, контроль версій та спільну роботу команди розробників.

Спеціалізовані програмні комплекси також існують для управління проєктами в інших сферах, таких як маркетинг, фінанси, організація подій, логістика тощо. Вони надають функції, специфічні для відповідних галузей, наприклад, планування маркетингових кампаній, керування бюджетами, контроль логістичних операцій тощо.

Користувачі спеціалізованих програмних комплексів отримують переваги від налаштованої функціональності, що відповідає їхнім конкретним потребам та вимогам. Ці програми полегшують виконання специфічних завдань та забезпечують ефективне управління проєктами у відповідних галузях.

Отже, управління проєктами супроводжується використанням різноманітних програмних комплексів, які можуть бути спеціалізованими або

універсальними. Спеціалізовані програмні комплекси розроблені для конкретних галузей або типів проєктів, вони надають функціональність, специфічну для цих галузей. Універсальні програмні комплекси є більш універсальними і адаптуються до потреб різних проєктів. Вони надають широкий спектр функціональності, що дозволяє планувати, виконувати та контролювати проєкти на різних рівнях складності. Вибір між спеціалізованими та універсальними програмними комплексами залежить від конкретних потреб, характеристик проєкту та вимог замовника. Кожен з цих видів програмних комплексів має свої переваги та можливості, і важливо обрати той, який найкраще відповідає потребам та вимогам конкретної організації та проєкту.

### 2.3. Формулювання технічних вимог до розроблюваної програми

Перед розробкою застосунку необхідно провести аналіз вимог до системи, який заснований на вимогах користувачів та функціональних вимогах до системи. Для цього використовується use-case діаграма прецедентів, яка відображає взаємодію користувачів з системою та описує основні функціональні вимоги до неї.

Згідно із проведеного аналізу предметної області, був складений список вимог до системи, який наведений у таблиці 2.1. Цей список вимог враховує потреби користувачів та функціональні вимоги до системи.

Аналіз вимог є важливим етапом перед розробкою застосунку, оскільки він допомагає зрозуміти потреби користувачів та визначити основні функціональні вимоги до системи. Це забезпечує успішну розробку та задоволення потреб користувачів.

Таблиця 2.1.

**Функціональні вимоги до застосунку**

| ***Вимоги*** | ***Опис*** |
|---|---|
| REQ-1 | Можливість проведення реєстрації та автентифікації |
| REQ-2 | Ведення журналу подій застосунку |
| REQ-3 | Можливість додати та редагувати проєкти |
| REQ-4 | Можливість додати та редагувати завдання для проєктів |
| REQ-5 | Можливість додати та редагувати групи користувачів |
| REQ-6 | Можливість видачі задань користувача за допомогою канбан |
| REQ-7 | Можливість обговорення процесу вирішення завдань |

*Джерело: розроблено автором*

Наступним кроком є визначення нефункціональних вимоги до застосунку (див. табл. 2.2).

Таблиця 2.2.

**Нефункціональні вимоги застосунку**

| ***Вимоги*** | ***Опис*** |
|---|---|
| REQ-8 | Система повинна бути стійкою та надійною, здатною забезпечити безперебійну роботу та уникнення втрати даних. |
| REQ-9 | Застосунок повинен забезпечувати належний рівень безпеки для захисту конфіденційності, цілісності та доступності даних |
| REQ-10 | Система повинна працювати швидко та ефективно, забезпечуючи мінімальні часові затримки та оптимальне використання ресурсів |
| REQ-11 | Система повинна бути легко масштабованою, здатною працювати з різною кількістю користувачів та об'ємами даних |

*Джерело: розроблено автором*

Для ідентифікації ключових учасників (акторів), які взаємодіють з системою, та їх основних цілей у контексті програмного додатку розроблено табл. 2.3. Кожен актор може мати одну або декілька цілей, які він намагається досягти за допомогою системи. Ці цілі, у свою чергу, стають основою для визначення основних варіантів використання (use-cases), що допомагають визначити функціональність та поведінку системи.

Таблиця 2.3.

**Актори та цілі застосунку**

| ***Актори*** | ***Цілі*** |
|---|---|
| **1** | **2** |
| Адміністратор | Головна мета адміністратора полягає в забезпеченні безпеки та належного функціонування облікових записів у системі, а також: груп, проєктів та завдань. Він відповідає за захист даних, контроль доступу та управління правами користувачів. Адміністратор забезпечує безперебійну роботу системи та реагує на потенційні загрози та проблеми безпеки |
| Користувач | Основна мета користувача полягає в ефективному та зручному користуванні системою. Користувач очікує, щоб система була легкою у використанні, зрозумілою та мала інтуїтивний інтерфейс |
| База даних | Головна мета бази даних полягає в зберіганні та організації необхідної інформації. Вона має забезпечувати надійність та цілісність даних, швидкий доступ до них та ефективне виконання запитів |

*Джерело: розроблено автором*

Табл. 2.4 представляє деталізований перелік сценаріїв використання системи різними акторами. Кожен варіант використання включає в себе ім'я,

яке однозначно ідентифікує його, опис, який висвітлює основну мету або результат використання, та послідовність кроків, яку актор повинен зробити для досягнення цієї мети.

Таблиця 2.4.

**Опис варіантів використання застосунку**

| ***Варіант використання*** | ***Ім'я*** | ***Опис*** |
|---|---|---|
| **1** | **2** | **3** |
| UC1 | Реєстрація користувача | Дозволяє користувачеві зареєструватися у системі, надаючи необхідну інформацію та створюючи обліковий запис для подальшого використання |
| UC2 | Автентифікація в системі | Дозволяє пройти процес автентифікації для підтвердження своєї особи та отримання доступу до системи |
| UC3 | Вивід каталогу користувачів | Доступна користувачеві з правами адміністратора і дозволяє йому переглянути повний список користувачів |
| UC4 | Додати користувача | Дозволяє адміністратору розширити кількість користувачів у системі шляхом створення нових облікових записів |
| UC5 | Редагувати користувача | Дозволяє адміністратору змінювати та оновлювати інформацію про користувача |
| UC6 | Вивід каталогу груп | Дає можливість користувачеві ознайомитись з повним списком груп створених у системі |
| UC7 | Додати нову групу | Дозволяє розширити кількість групу, що містяться в системі |

Продовження таблиці 2.4

| 1 | 2 | 3 |
|---|---|---|
| UC8 | Редагувати групу | Дозволяє змінювати та оновлювати інформацію про групу |
| UC9 | Управляння групами | Дозволяє керувати користувачами у групі |
| UC10 | Вивід каталогу проєктів | Дозволяє вивести каталог всіх проєктів, що створив користувач |
| UC11 | Додати проєкт | Дозволяє додати новий проєкт |
| UC12 | Редагувати проєкт | Дозволяє змінювати та оновлювати інформацію про вибраний проєкт |
| UC13 | Вивід каталогу завдань | Дозволяє вивести каталог всіх завдань у вибраному проєкту |
| UC14 | Додати завдання | Дозволяє додати нове завдання у вибраний проєкт |
| UC15 | Редагувати завдання | Дозволяє змінювати та оновлювати інформацію про вибране завдання |
| UC16 | Управління завданнями | Дозволяє провести процедуру приєднання користувачів для виконання завдань |
| UC17 | Виведення канбану | Дозволяє користувачеві бачити всі свої завдання, на які він приєднаний |
| UC18 | Обговорення завдань | Дозволяє користувачу залишати коментар у вибраному завданні |
| UC19 | Перегляд подій | Дозволяє переглядати події, що відбулися у системі |

*Джерело: розроблено автором*

На основі отриманих даних були створені use-case діаграми прецедентів для ролей «адміністратор» та «користувач». Use-case діаграма прецедентів розглядає взаємодію між акторами (користувачами) та системою, відображаючи основні функціональні можливості системи.

У діаграмі прецедентів для ролі «адміністратор» (див. рис. 2.3) представлені основні дії та взаємодії, які адміністратор може виконувати у системі. Це включає реєстрацію користувачів, управління обліковими записами, додавання та редагування користувачів, доступ до каталогу користувачів, а також виконання інших функцій, які залежать від специфіки системи.

Діаграма прецедентів для ролі «користувач» (див. рис. 2.4) показує дії та взаємодії, доступні користувачу. Це включає автентифікацію в системі, перегляд власного профілю, зміну особистої інформації та пароля, а також виконання інших функцій, які залежать від специфіки системи.


Рис. 2.3. Діаграма use-case із роллю «адміністратор»

*Джерело: побудовано автором*

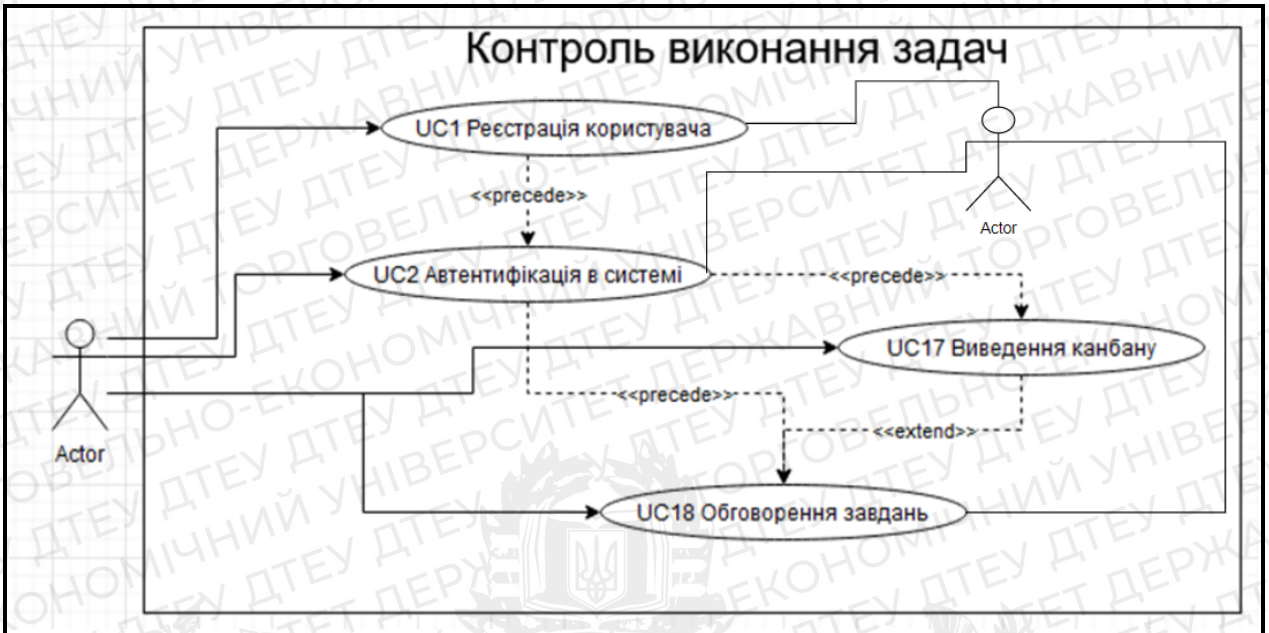


Рис. 2.4. Діаграма use-case із роллю «користувач»

*Джерело: побудовано автором*

Як можна побачити із рис. 2.3 системний адміністратор виконує роль, що дозволяє контролювати і управляти всіма аспектами програмного додатка. Адміністратор може реєструвати нових користувачів, проводити автентифікацію, виводити каталоги користувачів, груп, проєктів і задач. Він також має повноваження додавати та редагувати користувачів, групи, проєкти і задачі, які існують в системі. За допомогою інтерфейсу канбану, адміністратор може управляти процесом виконання задач, включаючи обговорення задач і перегляд подій. Ця роль має повний контроль над системою та її елементами.

Роль користувача (див. рис. 2.4) має менше повноважень, ніж системний адміністратор, але вона включає в себе ключові аспекти використання програмного додатка. Користувач може зареєструватися в системі, провести автентифікацію, вивести інтерфейс канбану для перегляду і управління задачами. Крім того, користувач може обговорювати задачі, що дозволяє учасникам команди вести діалог про специфіку задачі, її статус і потреби.

## 2.5. Висновок до розділу 2

У цьому розділі було проведено детальний аналіз предметної області, включаючи огляд основних існуючих рішень у сфері управління завданнями. Було виявлено, що стандартні інструменти не завжди забезпечують достатню гнучкість і можливості для задоволення специфічних потреб аутсорс-компаній, що акцентує на необхідності розробки спеціалізованого програмного додатка.

Також було зроблено акцент на визначенні проблем, які має вирішувати розроблюваний додаток. Описано способи вирішення цих проблем та їх обґрунтування, що підтверджує релевантність розроблюваного рішення.

Важливим етапом стало формулювання технічних вимог до розроблюваної програми. Це включає в себе створення специфікацій use-case, які відображають основні сценарії використання програми кінцевими користувачами. Це сприятиме більш ефективному проектуванню та розробці програмного забезпечення, що відповідатиме очікуванням користувачів.

Таким чином, аналіз предметної області та визначення проблематики, що покликана вирішити програма, стало фундаментом для подальшого проектування та розробки програмного додатка для контролю виконання завдань аутсорс-компанії.

# РОЗДІЛ 3
# ОПИС РОЗРОБКИ ТЕХНІЧНОГО ВИРІШЕННЯ ПРОБЛЕМАТИКИ

## 3.1. Концептуальна модель бази даних

Під час проектування бази даних для предметної області був використаний метод «сутність-зв'язок» (ER-метод). Цей метод передбачає розгляд завдань на нижчих рівнях перед тим, як переходити до розв'язання завдань більш високого рівня. Відповідно до цього підходу була створена логічна модель з використанням ER-діаграми, в якій були визначені наступні сутності та атрибути:

– групи: ідентифікатор групи, назва та опис;

– списки груп: ідентифікатор списку групи, ідентифікатор групи, ідентифікатор користувача, ідентифікатор завдання;

– коментар: ідентифікатор коментаря, ідентифікатор завдання, дата час коментаря, ідентифікатор користувача, назва користувача, опис коментаря;

– проєкти: ідентифікатор проєкту, ідентифікатор користувача, дата старту, дата закінчення, опис та статус виконання;

– завдання: ідентифікатор проєкту, ідентифікатор проєкту, дата виконання, опис завдання та статус виконання;

– списки завдань: ідентифікатор списку завдань, ідентифікатор завдання, ідентифікатор користувача;

– події: ідентифікатор події, дата події, ідентифікатор користувача та опис події;

– користувачі: ідент. користувача, прізвище, ім’я, назва облікового запису, пароль, ідентифікатор ролі, електронна адреса та додатковий опис.

ER-діаграма є важливим етапом проектування бази даних, оскільки дозволяє відображати зв'язки між сутностями та їх атрибутами. В результаті було створено ER-діаграму для бази даних аутсорс-компанії, яка наглядно відображає взаємозв'язки між таблицями та їх полями (див. рис. 3.1). Ця модель є важливою при подальшому проектуванні та розробці функціоналу додатку.

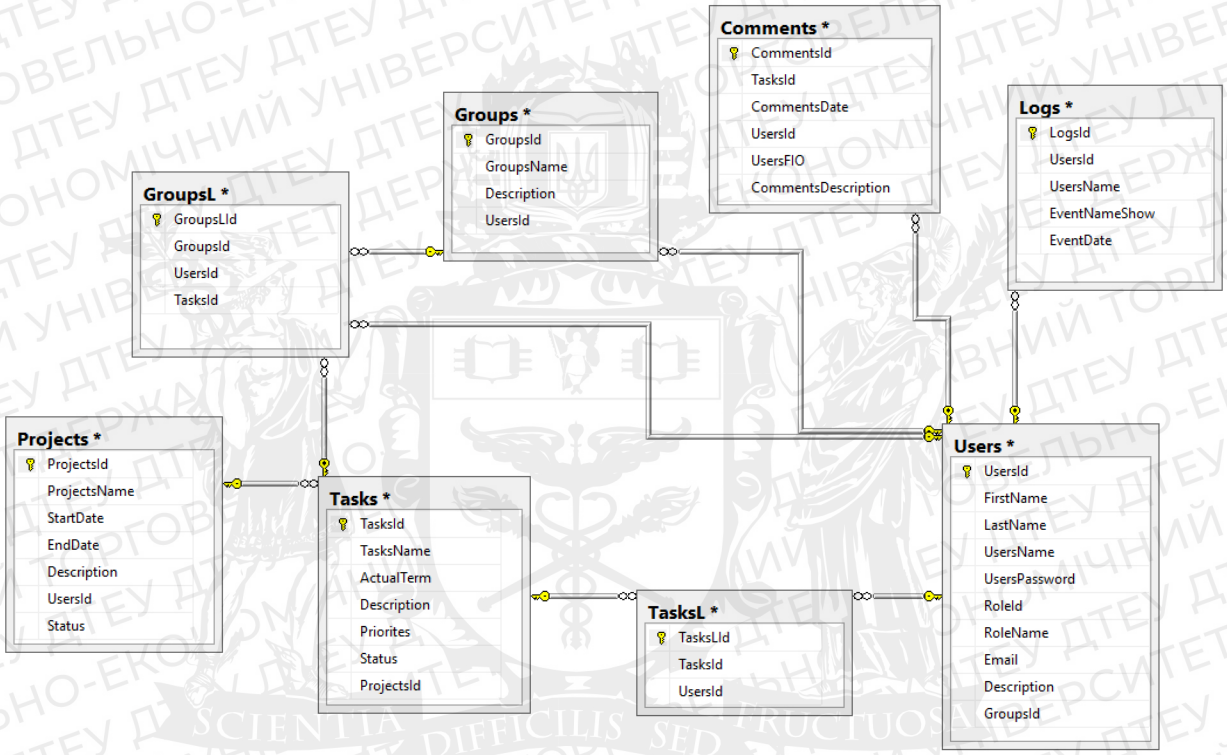


Рис. 3.1. Логічна модель бази даних

*Джерело: побудовано автором*

## 3.2. Макет програми

Перший модуль, з яким буде взаємодіяти будь-який користувач, є модуль реєстрації та авторизації. Цей модуль складається з двох окремих вікон: вікна авторизації та вікна реєстрації, які виконують різні завдання.

У вікні реєстрації користувач буде заповнювати дані для авторизації, такі як логін і пароль. Після натискання кнопки підтвердження, введені дані

будуть перевірятися, і відповідно до результатів користувач буде перенаправлений до головного вікна програми або отримає повідомлення про помилку авторизації. У вікні авторизації також передбачена кнопка, яка дозволяє перейти до вікна реєстрації, якщо користувач ще не зареєстрований в системі. Прототип вікна авторизації наведено на рис. 3.2.

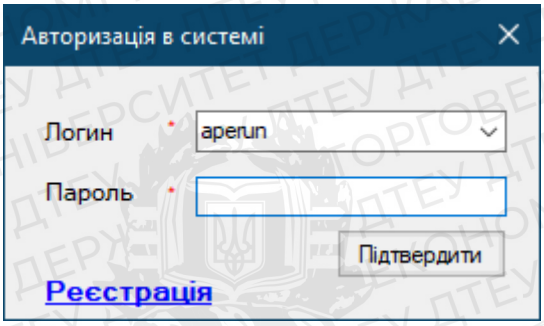


Рис. 3.2. Форма авторизації

*Джерело: побудовано автором*

Форма реєстрації вимагає введення всіх необхідних даних, таких як ім'я користувача, електронна пошта, пароль тощо. Після заповнення всіх даних користувач натисне кнопку підтвердження, що запустить процес запису його даних у базу даних. Відповідний прототип вікна реєстрації показано на рисунку 3.3.

Обидва ці модулі важливі для забезпечення безпеки та зручного користування системою, дозволяють користувачам реєструватися та авторизовуватися для доступу до функціональності програмного додатку.

Наступним кроком потрібна реалізація форми, що зможе керувати проєктами та завданнями, така функціональність потрібна для користувача, що матиме роль адміністратора. На рис. 3.4 та 3.5 зображено форми для додавання нових проєктів та завдань відповідно.

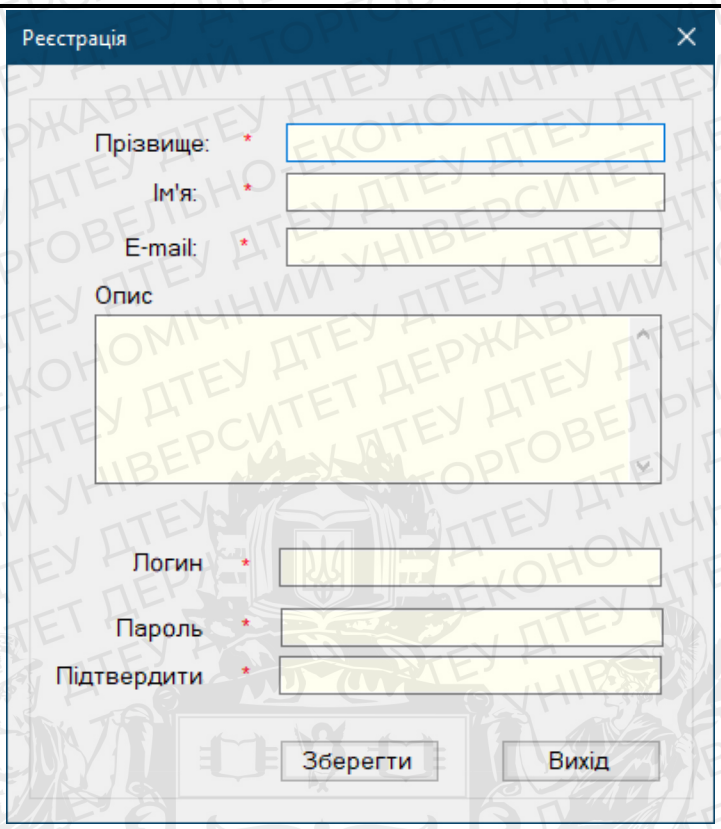


Рис. 3.3. Форма реєстрації

*Джерело: побудовано автором*

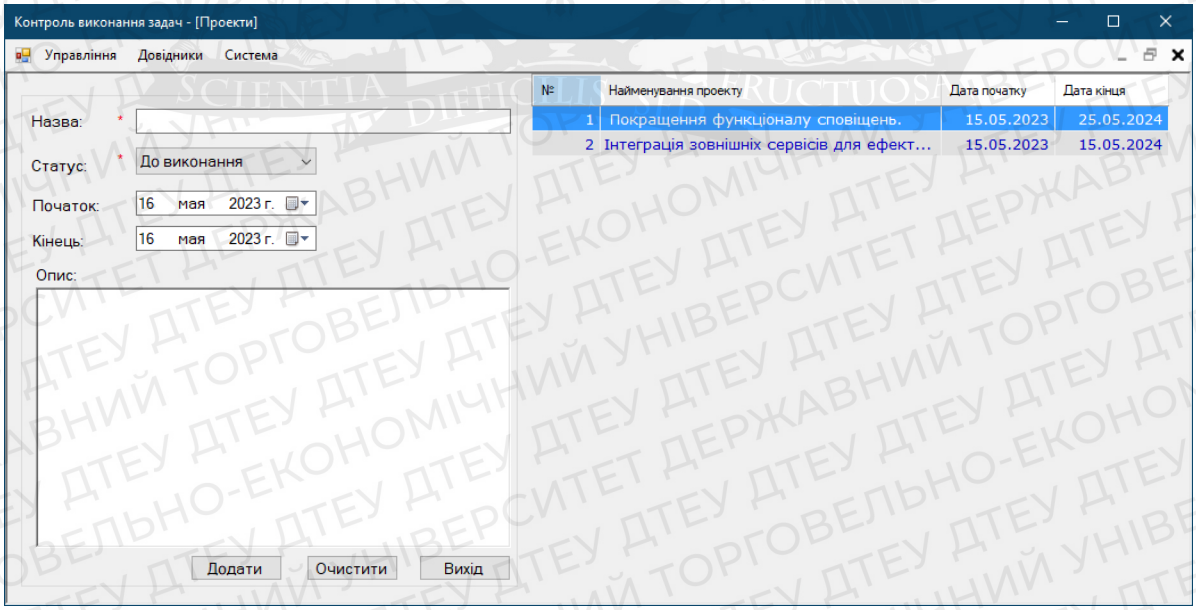


Рис 3.4. Форма опрацювання проєктів

*Джерело: побудовано автором*

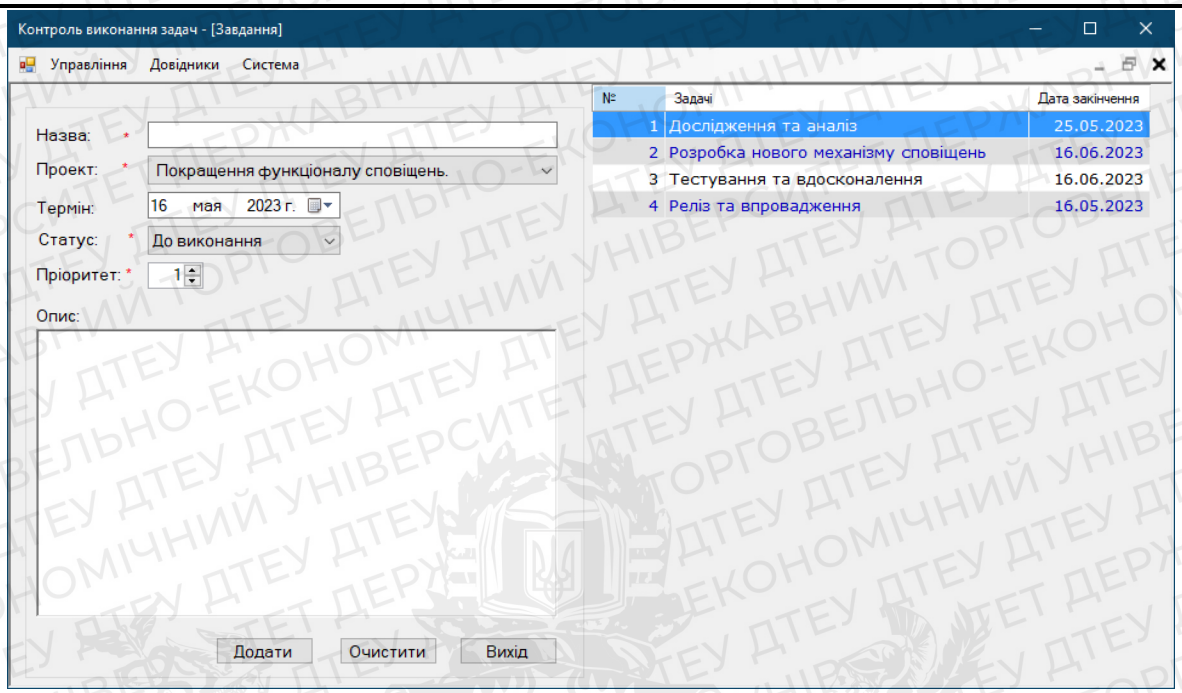


Рис. 3.5. Форма додавання опрацювання завдань

*Джерело: побудовано автором*

Також, важливим є створення форми для керування завданнями та відображенням дошки завдань (див. рис. 3.6-3.7).

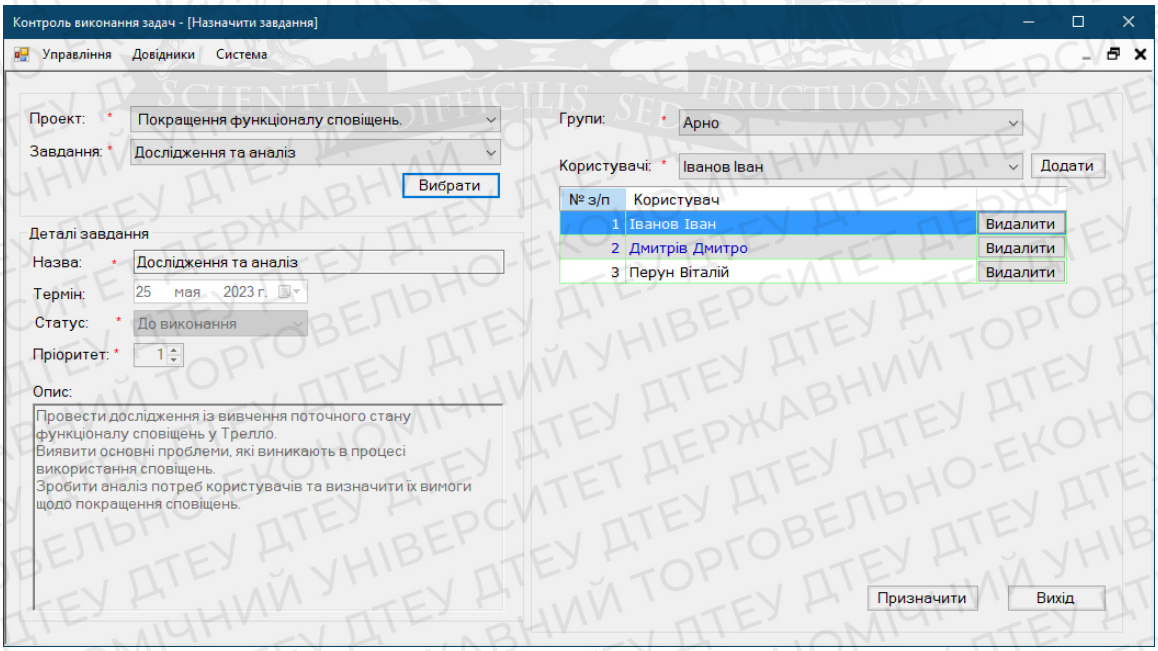


Рис. 3.6. Форма призначення завдань

*Джерело: побудовано автором*

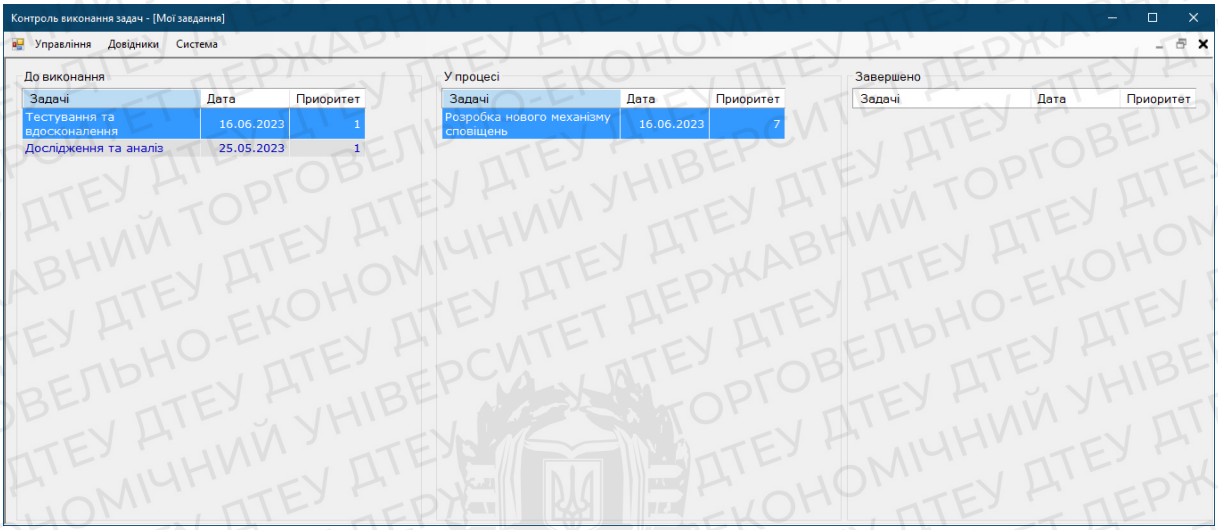


Рис. 3.7. Форма виведення поточних завдань

*Джерело: побудовано автором*

## 3.3. Опис архітектури програмного забезпечення

Архітектура проекту буде базуватися на трирівневій архітектурі, яка складається з наступних рівнів:

– рівень презентації - це верхній рівень, який відповідає за взаємодію користувача з системою;

– рівень бізнес-логіки - це середній рівень, який відповідає за обробку даних та реалізацію бізнес-логіки проекту. На цьому рівні будуть знаходитися класи, які забезпечують роботу з даними та виконання бізнес-логіки, що потрібна для розробки інформаційної системи підтримки прийняття рішень для інтернет продаж;

– рівень доступу до даних - це нижній рівень, який відповідає за зберігання та доступ до даних.

Трирівнева архітектура дозволить розділити функціональні можливості проекту на окремі рівні, що дозволить підтримувати легкість модифікації, тестування та розширення проекту.

На початку було реалізовано шар доступу до даних для роботи з базою даних було реалізовано 8 класів: CommentsProvider, GroupsLProvider, GroupsProvider, LogsProvider, ProjectsProvider, TasksLProvider, TasksProvider, UsersProvider. Рис. 3.8 відображає діаграму класів з методами для роботи з базою даних.

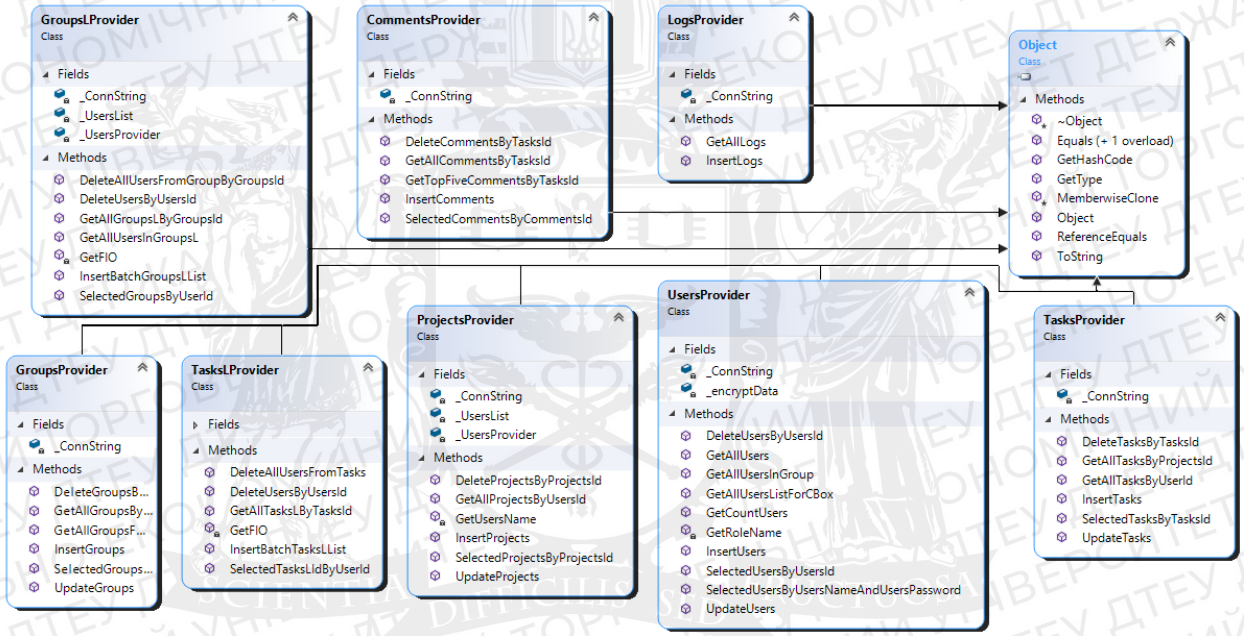


Рис. 3.8. Діаграма класів рівня даних

*Джерело: побудовано автором*

На рівні користувацького інтерфейсу було розроблено форми, які містять набір елементів керування, таких як кнопки, текстові поля та таблиці. Ці елементи дозволяють користувачам взаємодіяти з іншими рівнями системи шляхом обробки відповідних подій.

Користувач може використовувати цей рівень для введення необхідної інформації, здійснення пошуку та отримання результатів у зручному форматі.

Елементи керування взаємодіють з іншими компонентами системи, передаючи введені дані та отримуючи результати операцій.

Діаграма системи рівня користувацького інтерфейсу, яка зображена на рисунку 3.9, відображає взаємозв'язки між елементами і компонентами цього рівня. Це дозволяє розробникам системи керувати взаємодією з користувачами та забезпечувати їм зручний та ефективний досвід використання програмного додатку.

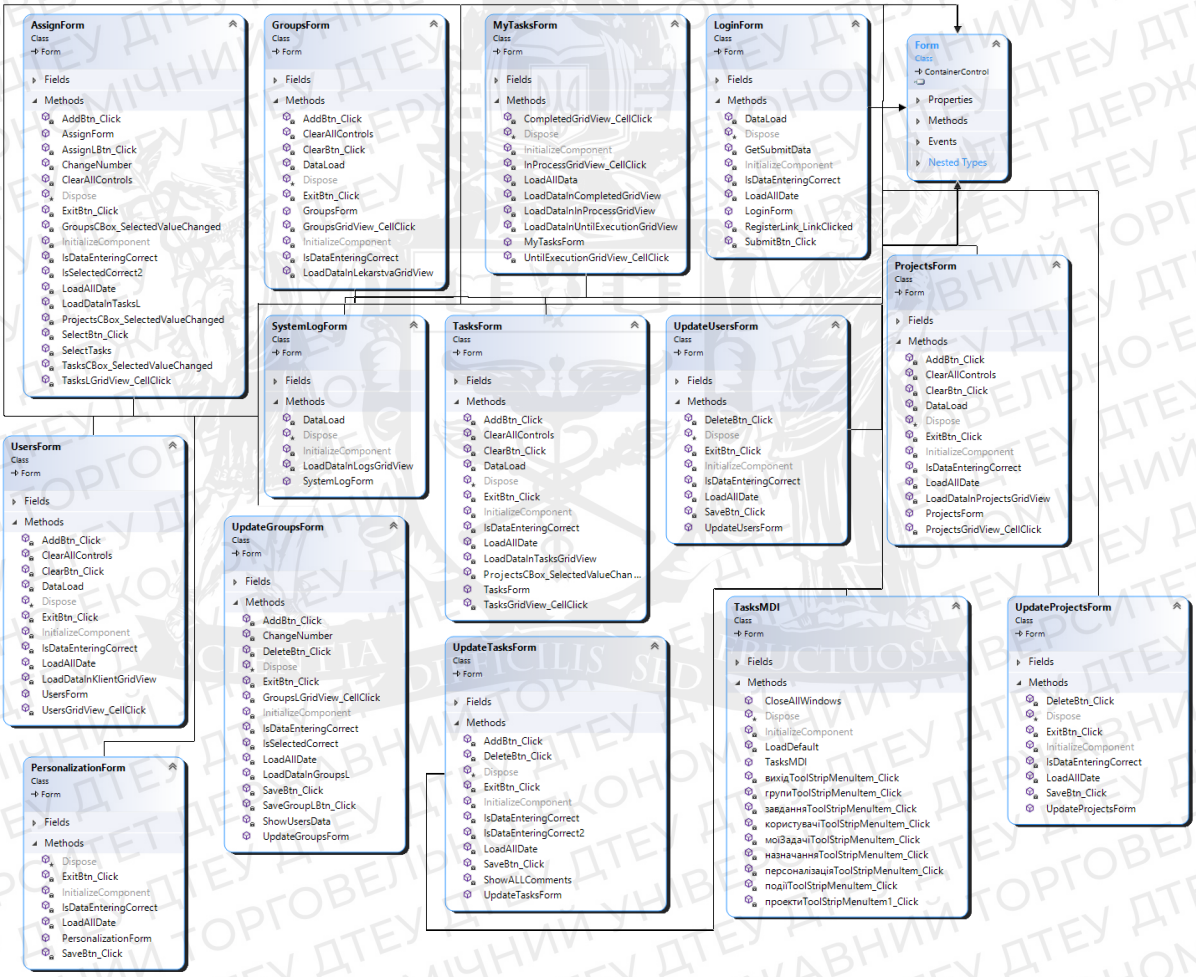


Рис. 3.9. Діаграма інтерфейсу системи

*Джерело: побудовано автором*

На діаграмі (рис. 3.9) відображено набір форм, що використовуються у системі для різних варіантів використання, що включають форми для керування завданнями (TasksForm, UpdateTasksForm, MyTasksForm), форми

для керування групами (GroupsForm, UpdateGroupsForm), форми для керування проектами (ProjectsForm, UpdateProjectsForm), форми для автентифікації користувача (LoginForm), форми для керування користувачами (UsersForm, UpdateUsersForm, AssignForm) та інші. Кожна форма на діаграмі пов'язана з відповідним варіантом використання, що допомагає забезпечити зрозумілість і логічність інтерфейсу для користувачів. Завдяки цьому користувачам надається зручний і ефективний досвід використання програмного додатку.

Останнім етапом реалізації архітектури було створення бізнес-логіки додатку, яке є ключовим в розробці програмного забезпечення. Бізнес-логіка відповідає за обробку бізнес-операцій і не залежить від бази даних або користувацького інтерфейсу, а забезпечує правильне функціонування додатку в цілому.

Створення бізнес-логіки дозволяє розбити функціонал додатку на логічні блоки, що сприяє модульності та гнучкості. Це означає, що окремі частини функціоналу можуть бути змінені без впливу на інші частини, що полегшує підтримку та розвиток проекту.

Розділення бізнес-логіки від бази даних та користувацького інтерфейсу дозволяє використовувати різні бази даних та інтерфейси без впливу на логіку додатку в цілому. Це забезпечує більшу гнучкість та адаптивність додатку до змін у вимогах бізнесу та технологіях.

Таким чином, створення бізнес-логіки є важливим кроком, який дозволяє розбити функціонал на логічні блоки, забезпечити модульність, гнучкість і незалежність від бази даних та інтерфейсу. Це сприяє підтримці, розвитку та адаптації додатку до змін у вимогах та технологіях.

Діаграма класів бізнес-логіки, представлена на рис. 3.10, відображає взаємодію між основними компонентами бізнес-логіки системи. Вона включає класи RoleApp, EncryptData, LogicBLL, TasksAll та ValidationMy, які

відповідають за різні аспекти виконання задач і функціональності системи. Наприклад, клас RoleApp управляє ролями в додатку, EncryptData забезпечує шифрування даних, LogicBLL виконує основну бізнес-логіку, TasksAll управляє всіма задачами в системі, а ValidationMy перевіряє коректність даних. Ці класи спільно забезпечують функціонування бізнес-логіки програмного додатку.

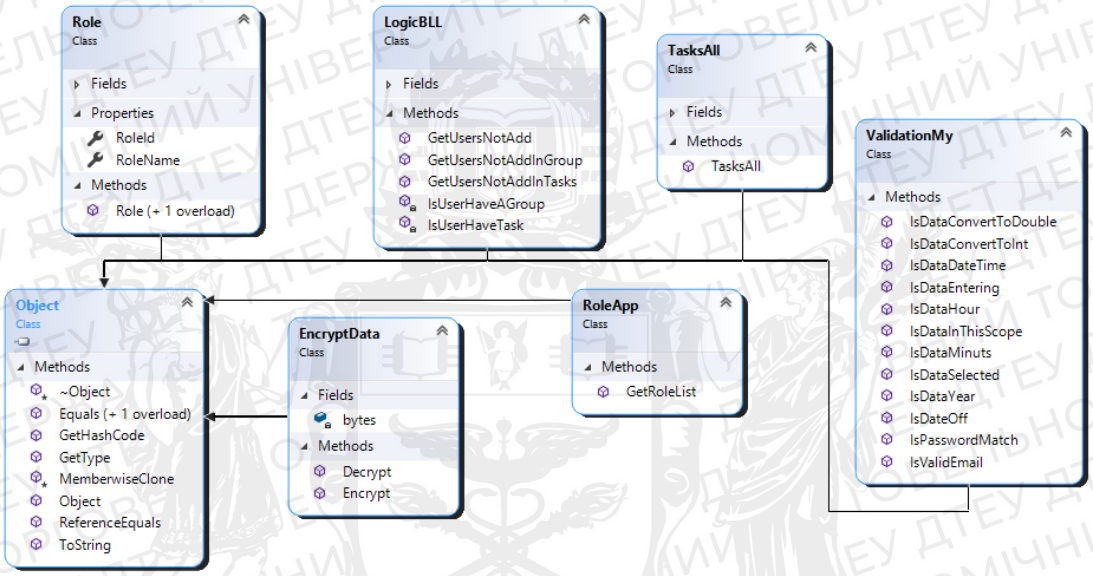


Рис. 3.10. Діаграма класів бізнес-логіки

*Джерело: побудовано автором*

## 3.4. Опис розробки та використання алгоритмів та методів для програмного додатку управління завдання аутсорс-компанії

Задача полягає в розробці простої та зручної у використанні платформи для контролю виконання завдань аутсорс-компанії. Для досягнення цієї мети необхідно розробити основні алгоритми, які забезпечать коректну роботу додатку та забезпечать користувачів необхідною інформацією та можливістю взаємодії з базою даних притулку тварин.

На рисунку 3.11 подана блок-схема, що ілюструє послідовність дій для додавання інформації про новий проєкт у базу даних. Починається процес з

введення користувачем необхідних даних про проєкт, включаючи назву проєкту, опис, дати початку та завершення. Наступний крок - це перевірка цих даних на відповідність вимогам, перевірка на наявність всіх обов'язкових полів, коректність дат, тощо. Якщо всі дані введені правильно, інформація зберігається в базі даних. В іншому випадку, користувачу представляється повідомлення про помилку із вказівками для виправлення. Весь цей процес призначений для забезпечення точного та ефективного зберігання інформації про проєкти.

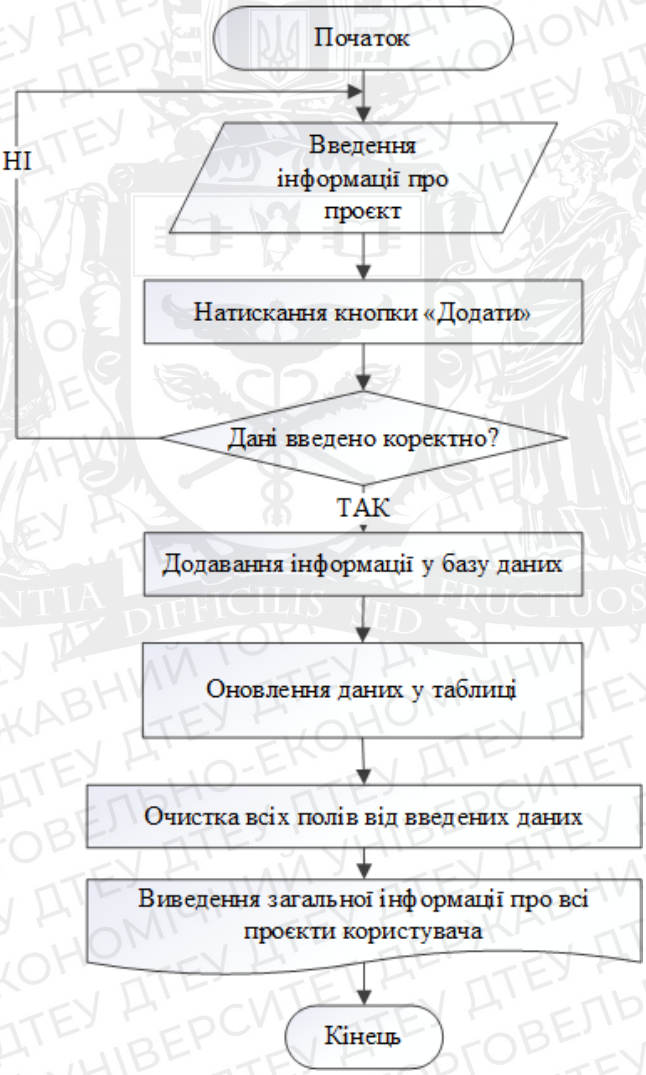


Рис. 3.11. Додавання інформації про новий проєкт

*Джерело: побудовано автором*

На рис. 3.12 показана блок-схема редагування інформації вибраного запису із таблиці та виведення інформації про всі проєкти, інформація про яких зберігається у базі даних.

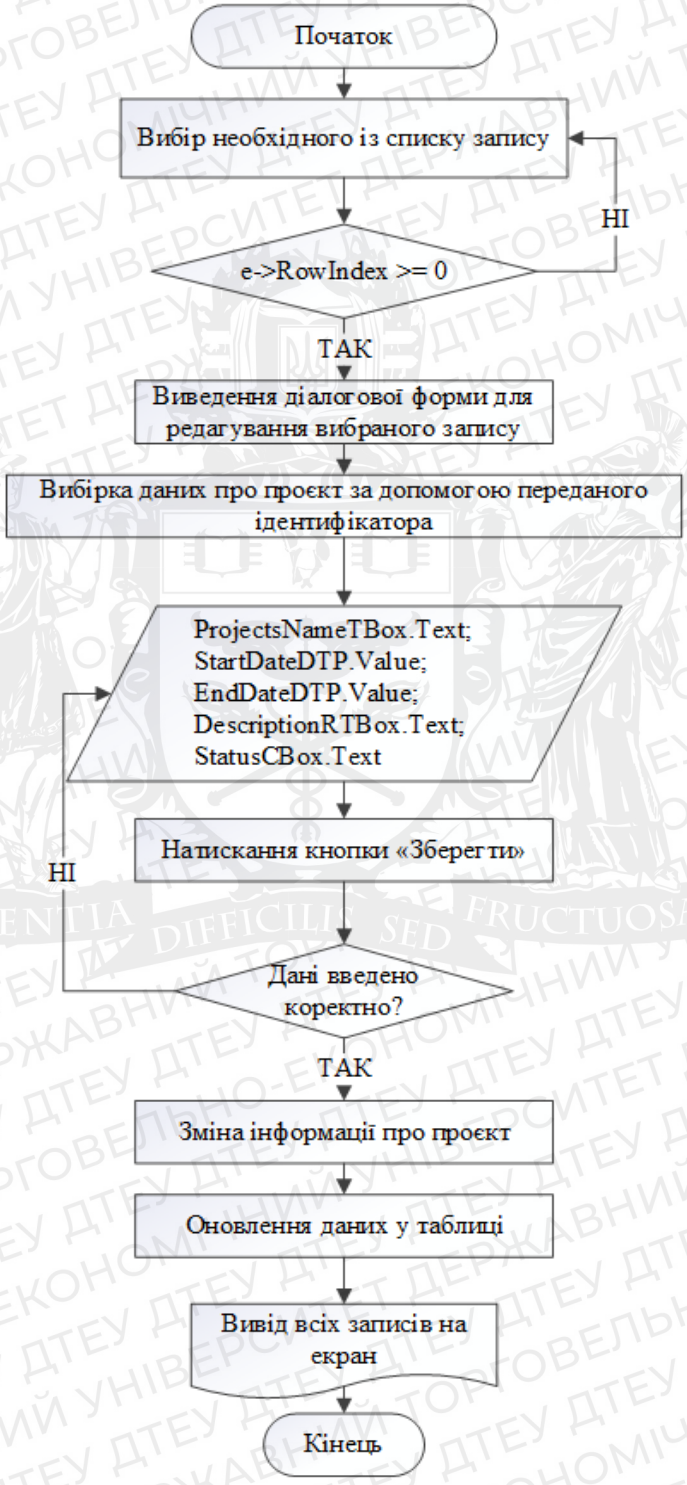


Рис. 3.12. Редагування інформації вибраного запису

*Джерело: побудовано автором*

Після проєктування бази даних та розробки алгоритмів, за допомогою Visual Studio 2019 на мові C# було розроблено код додатку. На початку за допомогою змінної «CONNECT» у файлі проекту «App.config» було здійснене підключення до бази даних та задано параметри, які зображено на рис. 3.13.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECT" value="Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\DB.mdf;
       Integrated Security=True" />
</appSettings>
```

Рис. 3.13. Змінна із параметрами налаштування бази даних

*Джерело: побудовано автором*

Для ефективної роботи з базою даних у проекті було використано простір імен System.Data.SqlClient. Цей простір надає набір класів, які дозволяють взаємодіяти з базою даних MS SQL Server

Для створення меню в проекті додано елемент menuStrip до головного вікна програми. Це дає можливість користувачам вибирати різні опції та викликати необхідні функції програми за допомогою простого та зрозумілого інтерфейсу (див. рис. 3.14). Меню дозволяє забезпечити зручну навігацію та доступ до функціоналу програми.

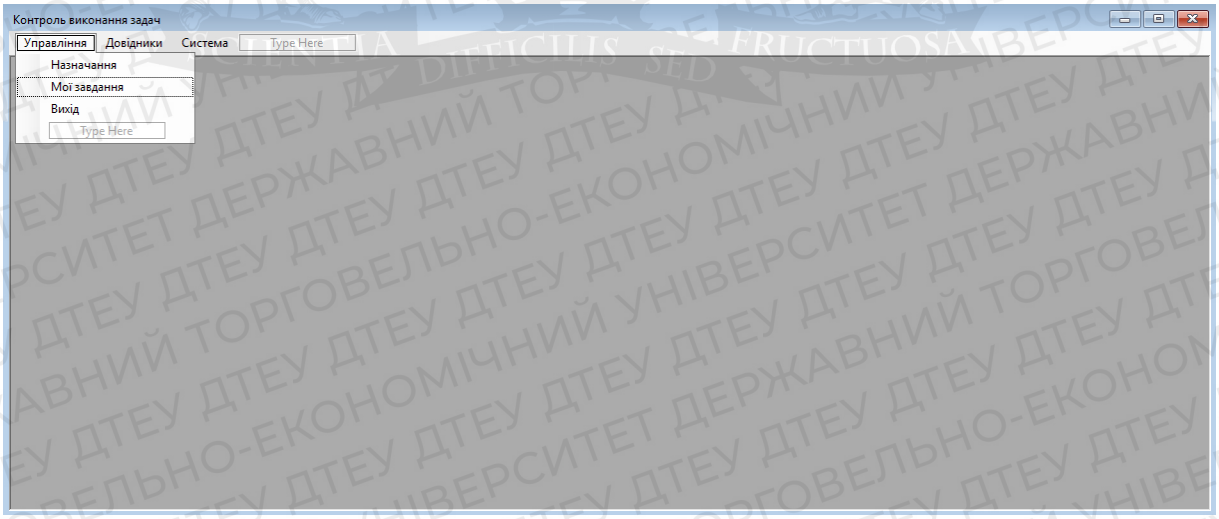


Рис. 3.14. Додавання головного меню

*Джерело: побудовано автором*

Для кожного пункту меню було додано відповідний код, який викликає створення екземпляру відповідної форми. Коли користувач вибирає певний пункт меню, відповідна форма відкривається, і її вікно стає активним для користувача.

Додатково, був доданий код, який забезпечує закриття попередньо відкритої форми перед відкриттям нової. Це гарантує правильне відображення вікон та уникнення можливих конфліктів між вікнами. Такий код застосовується для кожного пункту меню, забезпечуючи їх коректну роботу та взаємозв'язок з відповідними формами.

На рисунку 3.15 наведено приклад коду для одного з пунктів меню. Цей код відображає логіку, яка створює екземпляр форми та забезпечує її відкриття, а також закриття попередньо відкритого вікна. Такий підхід допомагає забезпечити належну роботу меню та зручну навігацію для користувача.

```
1 reference
private void завданняToolStripMenuItem_Click(object sender, EventArgs e) {
    if (LoginForm.CurrentUser.RoleId == 1) {
        CloseAllWindows();
        TasksForm tasksForm = new TasksForm();
        tasksForm.MdiParent = this;
        tasksForm.WindowState = FormWindowState.Maximized;
        tasksForm.Show();
    } else {
        MessageBox.Show(NamesMy.MessageBoxExaption.YouDontHavePermission);
    }
}
```

Рис. 3.15. Код пунктів меню

*Джерело: побудовано автором*

Після цього було розроблено класи, які відповідають за роботу з базою даних. Вони містять реалізації різних методів з детальним описом їх функціоналу.

Наприклад, для додавання інформації про новий проєкт був створений метод з назвою «InsertProjects». Цей метод відповідає за вставку даних про проєкт до бази даних. На рисунку 3.16 показано код даного методу.

```
1 reference
public void InsertProjects(string ProjectsName, DateTime StartDate, DateTime EndDate,
  string Description, int UsersId, string Status) {
  SqlConnection connection = new SqlConnection(_ConnString);
  string query = "INSERT into Projects (ProjectsName, StartDate, EndDate, Description, UsersId, Status) " +
    "VALUES (@ProjectsName, @StartDate, @EndDate, @Description, @UsersId, @Status)";
  SqlCommand command = new SqlCommand(query, connection);
  command.Parameters.AddWithValue("@ProjectsName", ProjectsName);
  command.Parameters.AddWithValue("@StartDate", StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
  command.Parameters.AddWithValue("@EndDate", EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
  command.Parameters.AddWithValue("@Description", Description);
  command.Parameters.AddWithValue("@UsersId", UsersId);
  command.Parameters.AddWithValue("@Status", Status);
  connection.Open();
  command.ExecuteNonQuery();
  connection.Close();
}
```

Рис. 3.16. Код методу «InsertProjects»

*Джерело: побудовано автором*

Метод InsertProjects відповідає за вставку нового запису про проект в базу даних. Він отримує декілька параметрів, які використовуються для заповнення полів таблиці проектів.

У першому рядку методу створюється об'єкт SqlConnection для підключення до бази даних за допомогою рядка з'єднання _ConnString, який ймовірно містить інформацію про сервер та назву бази даних. Створюється рядок запиту SQL, який містить команду INSERT INTO для таблиці проектів. Значення параметрів передаються за допомогою параметрів з кінцевою міткою @, що забезпечує безпеку від SQL-ін'єкцій.

Створюється об'єкт SqlCommand з рядком запиту та підключенням, і потім параметри запиту заповнюються значеннями з переданих аргументів методу. Далі відбувається відкриття з'єднання до бази даних за допомогою connection.Open(). Після цього виконується запит до бази даних за допомогою

command.ExecuteNonQuery(), який виконує команду INSERT і вставляє новий запис до таблиці проектів.

У кінці методу з'єднання з базою даних закривається за допомогою connection.Close() для звільнення ресурсів.

Для редагування даних про проєкт, також був розроблений метод, код якого представлено на рис. 3.17.

```
public void UpdateProjects(string ProjectsName, DateTime StartDate, DateTime EndDate, string Description,
  int UsersId, string Status, int ProjectsId) {
  using (SqlConnection con = new SqlConnection(_ConnString)) {
    using (SqlCommand command = new SqlCommand("UPDATE Projects SET ProjectsName=@ProjectsName,  " +
      "StartDate = @StartDate, EndDate=@EndDate, " +
      " Description = @Description,  " +
      "UsersId=@UsersId, Status=@Status " +
      " WHERE ProjectsId = @ProjectsId", con)) {
      command.CommandType = CommandType.Text;
      command.Parameters.AddWithValue("@ProjectsName", ProjectsName);
      command.Parameters.AddWithValue("@StartDate", StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@EndDate", EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@Description", Description);
      command.Parameters.AddWithValue("@UsersId", UsersId);
      command.Parameters.AddWithValue("@Status", Status);
      command.Parameters.AddWithValue("@ProjectsId", ProjectsId);
      con.Open();
      int rowsAffected = command.ExecuteNonQuery();
      con.Close();
    }
  }
}
```

Рис. 3.17. Код методу «UpdateProjects»

*Джерело: побудовано автором*

Метод UpdateProjects відповідає за оновлення запису проекту в базі даних. Він приймає кілька параметрів, які використовуються для оновлення відповідних полів таблиці проектів.

Після розробки всіх класів рівня даних було розроблено форми для взаємодії користувача із програмою. Наприклад, розроблена форма для опрацювання даних приєднання користувачів на певне завдання(див. рис. 3.18).

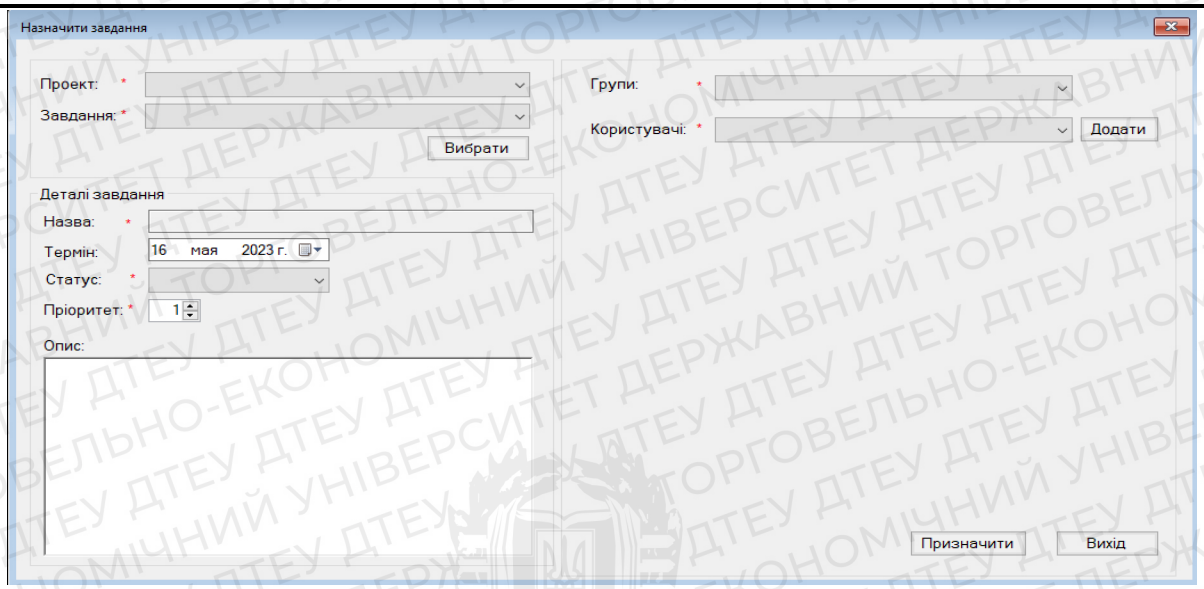


Рис. 3.18. Розробка форми для приєднання користувачів на завдання

*Джерело: побудовано автором*

Під час завантаження форми у її конструкторі викликається метод «LoadAllDate», який завантажує дані про статуси, проєкти, групи, завдання та користувачів у випадаючі списки (див. рис. 3.19).

```
private void LoadAllDate() {
    _StatusList = _StatusAppProvider.GetStatusList();
    StatusCBox.DataSource = _StatusList;
    StatusCBox.ValueMember = "StatusId";
    StatusCBox.DisplayMember = "StatusName";

    _ProjectsList = _ProjectsProvider.GetAllProjectsByUsersId(LoginForm.CurrentUser.UsersId);
    ProjectsCBox.DataSource = _ProjectsList;
    ProjectsCBox.ValueMember = "ProjectsId";
    ProjectsCBox.DisplayMember = "ProjectsName";

    SelectTasks();
    _GroupsList = _GroupsProvider.GetAllGroupsByUsersId(LoginForm.CurrentUser.UsersId);
    GroupsCBox.DataSource = _GroupsList;
    GroupsCBox.ValueMember = "GroupsId";
    GroupsCBox.DisplayMember = "GroupsName";
    _IsGroupsLoad = true;

    GroupsCBox_SelectedValueChanged(GroupsCBox, EventArgs.Empty);
    _IsProjectsLoad = true;
}
```

Рис. 3.19. Код методу «LoadAllDate»

*Джерело: побудовано автором*

У розроблених формах, для забезпечення коректності введених даних, було створено методи перевірки під назвою «IsDataEnteringCorrect». Кожна форма має свій власний метод, який виконує перевірку обов'язкових полів даних.

Один з таких методів, «IsDataEnteringCorrect» для форми «ProjectsForm», відображений на рисунку 3.20. Цей метод виконує перевірку коректності введених даних у всіх обов'язкових полях форми «ProjectsForm». У методі використовуються різні умови та перевірки, що забезпечують валідацію даних. Наприклад, для кожного обов'язкового поля виконується перевірка на наявність значення та його відповідність певним критеріям. Якщо дані не задовольняють критеріям, генерується повідомлення про помилку.

Цей метод дозволяє переконатися, що всі обов'язкові поля форми містять коректні дані перед подальшою обробкою або збереженням.

```
2 references
private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (Convert.ToInt32(ProjectsCBox.SelectedValue)> 0) {
    ProjectsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ProjectsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (Convert.ToInt32(TasksCBox.SelectedValue) > 0) {
    TasksValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    TasksValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}
```

Рис. 3.20. Код методу для перевірки коректності введених даних

*Джерело: побудовано автором*

Кожне поле має свій відповідний Label, де буде відображено результат перевірки. Якщо всі поля заповнені правильно, метод повертає true, інакше, метод повертає false.

Отже, в даному підрозділі було проведено частковий опис реалізації системи.

### 3.5. Висновок до розділу 3

У даному розділі було подано технічне вирішення проблематики, виходячи з ідентифікованих в розділі 2 проблем і потреб.

Спочатку, була створена концептуальна модель бази даних на базі MS SQL Server. Ця модель визначає структуру даних, яка буде використовуватися у програмному додатку, і забезпечує збереження, впорядкування та відновлення даних про завдання та проекти.

Далі було розроблено макети програми, які відображають основні екрани та елементи користувацького інтерфейсу. Це дозволило визначити структуру користувацького інтерфейсу та надало уявлення про те, як програмний додаток буде виглядати та яким чином буде взаємодіяти з користувачем.

Також було описано архітектуру програмного забезпечення, зокрема використання трьохрівневої архітектури. Ця модель включає в себе рівень даних, рівень бізнес-логіки та рівень представлення, що сприяє більшій модульності, гнучкості та зручності управління проектом.

Закінчується розділ описом розробки та використання алгоритмів і методів для програмного додатку, реалізованого за допомогою мови програмування C#. Це включає процеси обробки та аналізу даних, використання алгоритмів для оптимізації роботи з завданнями, а також використання різноманітних методів для забезпечення зручності та ефективності роботи програми.

# ВИСНОВКИ ТА ПРОПОЗИЦІЇ

На основі проведених досліджень та розробки, можна зробити наступні висновки:

1. Проведено аналіз існуючих програм та додатків для управління завданнями (Asana, Trello, Jira). Виокремлені їх основні переваги та недоліки, що дозволило відобразити потребу в новому програмному додатку, орієнтованому на унікальні потреби аутсорс-компанії.

2. В якості основних технологій для розробки додатку було обрано мову програмування C# та базу даних MS SQL Server, які забезпечують гнучкість, масштабованість та надійність системи.

3. Проведений аналіз проблематики аутсорс-компанії та ідентифіковані основні проблеми, які має вирішити розроблюваний програмний додаток. Зокрема, це питання ефективного управління завданнями, контролю їх виконання та спілкування між учасниками процесу.

4. В процесі виконання проєкту було розроблено концептуальну модель бази даних, макети програми, описано архітектуру програмного забезпечення та використання алгоритмів.

5. Розроблена система повністю задовольняє технічні вимоги та очікування користувачів. Вона спрощує процес управління завданнями, покращує комунікацію та співпрацю між учасниками процесу.

Як пропозицію на майбутнє, варто розглянути можливість інтеграції з іншими системами, такими як поштові клієнти, календарі, системи управління проектами, що дозволить ще більше розширити можливості використання додатку.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Керзнер Г. Управління проектами: системний підхід до планування, планування графіку та контролю / Г. Керзнер. - К.: Видавництво «Професіонал», 2022. - 912 с.

2. Аллен Д. Отримання речей зроблено: Мистецтво безстресової продуктивності / Д. Аллен. - Львів: Видавництво «Старого Лева», 2020. - 352с.

3. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Sixth Edition. - Project Management Institute, 2017. - 589 p.

4. Гусєв, О. В. Програмне забезпечення для управління проектами: погляд на Asana та Trello / О. В. Гусєв, В. Я. Костюк // Управління проектами та розвиток виробництва. - 2019. - № 4 (68). - С. 42-53.

5. Мельник, Б. Інтеграція інструментів управління проектами Asana і Trello в системи управління якістю IT-компаній / Б. Мельник, Л. Стефанович, А. Козак // Комп'ютерні науки і інженрія. - 2020. - № 1. - С. 65-73.

6. Murphy, B. Trello for Project Management: A Comprehensive Guide to Using Trello for Team Collaboration / B. Murphy. – USA: Amazon Digital Services LLC, 2021. – 80 с.

7. Сердюк, В. М. Особливості використання Trello в управлінні проектами / В. М. Сердюк // Управління проектами та розвиток виробництва. - 2020. - № 1 (73). - С. 89-99.

8. Мартинюк, О.С. Застосування Jira для оптимізації роботи IT-компаній / О.С. Мартинюк // Сучасні інформаційні системи та технології. - 2019. - № 2. - С. 74-80.


| | | | | | *ДТЕУ 121 06-20.БР* | | | |
|---|---|---|---|---|---|---|---|---|
| *Зм.* | *Аркуш* | *№ докум.* | *Підпис* | *Дата* | | | | |
| Зав. каф. | | Криворучко О.В. | | 23.12.22 | Програмний додаток контролю виконання завдань команди аутсорс-компанії | *Стадія* | *Аркуш* | *Аркушів* |
| Керівник | | Хорольська К.В. | | 23.12.22 | | *ВП* | *60* | *61* |
| Гарант | | Рзаєва С.Л. | | 23.12.22 | | Факультет інформаційних технологій *4 курс, 6 група* | | |
| Розробив | | Руденко В.Ю. | | 23.12.22 | *Висновки та пропозиції* | | | |

9. Asana . [Електронний ресурс] – Режим доступу: https://asana.com/ (дата звернення 14.04.2023)

10. Trello. [Електронний ресурс] – Режим доступу: https://trello.com/en (дата звернення 15.04.2023)

11. Jira. [Електронний ресурс] – Режим доступу: https://www.atlassian.com/jira (дата звернення 15.04.2023)

# ДОДАТКИ

ДОДАТОК А

## Код класу AssignForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TasksApp.AppCode;
using TasksApp.Forms.Systems;
using TasksApp.Provider;
using TasksApp.Providers;

namespace TasksApp.Forms.Controls {
 public partial class AssignForm : Form {
  private ValidationMy _validation = new ValidationMy();
  private ProjectsProvider _ProjectsProvider = new ProjectsProvider();
  private List<Projects> _ProjectsList = new List<Projects>();
  private StatusApp _StatusAppProvider = new StatusApp();
  private List<Status> _StatusList = new List<Status>();
  private TasksProvider _TasksProvider = new TasksProvider();
  private List<Tasks> _TasksList = new List<Tasks>();
  private Tasks _SelectedTask = new Tasks();
  private GroupsProvider _GroupsProvider = new GroupsProvider();
  private List<Groups> _GroupsList = new List<Groups>();
  private bool _IsGroupsLoad = false;
  private bool _IsProjectsLoad = false;

  private GroupsLProvider _GroupsLProvider = new GroupsLProvider();
  private List<GroupsL> _GroupsLList = new List<GroupsL>();
  private TasksLProvider _TasksLProvider = new TasksLProvider();
  private List<TasksL> _TasksLList = new List<TasksL>();

  public AssignForm() {
   InitializeComponent();
   LoadAllDate();
  }

  private void SelectBtn_Click(object sender, EventArgs e) {
_StatusList = _StatusAppProvider.GetStatusList();

StatusCBox.DataSource = _StatusList;

StatusCBox.ValueMember = «StatusId»;

StatusCBox.DisplayMember = «StatusName»;
```

```
    _ProjectsList = _ProjectsProvider.GetAllProjectsByUsersId(LoginForm.CurrentUser.UsersId);
    ProjectsCBox.DataSource = _ProjectsList;
    ProjectsCBox.ValueMember = «ProjectsId»;
    ProjectsCBox.DisplayMember = «ProjectsName»;
        SelectTasks();
    _GroupsList = _GroupsProvider.GetAllGroupsByUsersId(LoginForm.CurrentUser.UsersId);
    GroupsCBox.DataSource = _GroupsList;
    GroupsCBox.ValueMember = «GroupsId»;
    GroupsCBox.DisplayMember = «GroupsName»;
    _IsGroupsLoad = true;
    GroupsCBox_SelectedValueChanged(GroupsCBox, EventArgs.Empty);
    _IsProjectsLoad = true;
  }
  private void SelectTasks() {
    _TasksList = _TasksProvider.GetAllTasksByProjectsId(Convert.ToInt32(ProjectsCBox.SelectedValue));
    TasksCBox.DataSource = _TasksList;
    TasksCBox.ValueMember = «TasksId»;
    TasksCBox.DisplayMember = «TasksName»;
    _SelectedTask = _TasksProvider.SelectedTasksByTasksId(Convert.ToInt32(TasksCBox.SelectedValue));
    if (_SelectedTask.TasksId > 0) {
     TasksNameTBox.Text = _SelectedTask.TasksName;
     ActualTermDTP.Value = _SelectedTask.ActualTerm;
     StatusCBox.Text = _SelectedTask.Status;
     PrioritesNUD.Value = _SelectedTask.Priorites;
     DescriptionRTBox.Text = _SelectedTask.Description;
     _TasksLList = _TasksLProvider.GetAllTasksLByTasksId(_SelectedTask.TasksId);
     LoadDataInTasksL(_TasksLList);
    } else {
     ClearAllControls();
    }
  }
```

```
private void ClearAllControls() {
 TasksNameTBox.Text = String.Empty;
 ActualTermDTP.Value = DateTime.Now;
 PrioritesNUD.Value = 1;
 DescriptionRTBox.Text = String.Empty;
 _TasksLList.Clear();
 LoadDataInTasksL(_TasksLList);
}

private bool IsDataEnteringCorrect() {
 bool isCorrect = true;
 if (Convert.ToInt32(ProjectsCBox.SelectedValue)> 0) {
  ProjectsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
 } else {
  ProjectsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
  isCorrect = false;
 }
 if (Convert.ToInt32(TasksCBox.SelectedValue) > 0) {
  TasksValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
 } else {
  TasksValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
  isCorrect = false;
 }
 return isCorrect;
}
private void ProjectsCBox_SelectedValueChanged(object sender, EventArgs e) {
 if (_IsProjectsLoad) {
  SelectTasks();
 }
}
private void TasksCBox_SelectedValueChanged(object sender, EventArgs e) {
```

```
        if (_IsGroupsLoad && Convert.ToInt32(GroupsCBox.SelectedValue) > 0) {
            _SelectedTask = _TasksProvider.SelectedTasksByTasksId(Convert.ToInt32(TasksCBox.SelectedValue));
            if (_SelectedTask.TasksId > 0) {
                TasksNameTBox.Text = _SelectedTask.TasksName;
                ActualTermDTP.Value = _SelectedTask.ActualTerm;
                StatusCBox.Text = _SelectedTask.Status;
                PrioritesNUD.Value = _SelectedTask.Priorites;
                DescriptionRTBox.Text = _SelectedTask.Description;
                _TasksLList = _TasksLProvider.GetAllTasksLByTasksId(_SelectedTask.TasksId);
                LoadDataInTasksL(_TasksLList);
            } else {
                ClearAllControls();
            }

        }
    }
}
```

## Код класу MyTasksForm

```
private bool IsSelectedCorrect2() {
  bool isCorrect = true;
  if (Convert.ToInt32(GroupsCBox.SelectedValue) > 0) {
   GroupsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
   GroupsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
   isCorrect = false;
  }
  if (Convert.ToInt32(UsersCBox.SelectedValue) > 0) {
   UsersValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
   UsersValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
   isCorrect = false;
  }
  for (int i = 0; i < _TasksLList.Count; i++) {
   if (_TasksLList[i].UsersId == Convert.ToInt32(UsersCBox.SelectedValue)) {
    isCorrect = false;
    MessageBox.Show(«Вибраного користувача вже приєднано до завдання», «Увага!»);
    break;
   }
  }

  return isCorrect;
 }

 private void GroupsCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsGroupsLoad && Convert.ToInt32(GroupsCBox.SelectedValue) > 0) {
   _GroupsLList = _GroupsLProvider.GetAllGroupsLByGroupsId(Convert.ToInt32(GroupsCBox.SelectedValue));
   UsersCBox.DataSource = _GroupsLList;
   UsersCBox.ValueMember = «UsersId»;
   UsersCBox.DisplayMember = «FIO»;
  }
 }



 private void ChangeNumber() {
  for (int i = 0; i < _TasksLList.Count; i++) {
   _TasksLList[i].Number = i + 1;
  }
 }


 private void LoadDataInTasksL(List<TasksL> AllTasksLList) {
  ChangeNumber();
  TasksLGridView.DataSource = null;
  TasksLGridView.Columns.Clear();
```

```
  TasksLGridView.AutoGenerateColumns = false;
  TasksLGridView.RowHeadersVisible = false;

  TasksLGridView.DataSource = AllTasksLList;

  if (AllTasksLList.Count > 0) {
   DataGridViewColumn UsersIdColumn = new DataGridViewTextBoxColumn();
   UsersIdColumn.DataPropertyName = «UsersId»;
   UsersIdColumn.Name = «UsersId»;
   TasksLGridView.Columns.Add(UsersIdColumn);
   TasksLGridView.Columns[0].Visible = false;

   DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
   numberColumn.HeaderText = «№ з/п»;
   numberColumn.DataPropertyName = «Number»;
   numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
   numberColumn.Width = NamesMy.SizeOptins.NumberSize;
   TasksLGridView.Columns.Add(numberColumn);

   DataGridViewColumn FIOColumn = new DataGridViewTextBoxColumn();
   FIOColumn.HeaderText = «Користувач»;
   FIOColumn.DataPropertyName = «FIO»;
   FIOColumn.Name = «FIO»;
   FIOColumn.Width = 320;
   TasksLGridView.Columns.Add(FIOColumn);

   DataGridViewButtonColumn DeleteBtn = new DataGridViewButtonColumn();
   DeleteBtn.Text = «Видалити»;
   DeleteBtn.UseColumnTextForButtonValue = true;
   DeleteBtn.ToolTipText = «Видалити»;
   DeleteBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
   TasksLGridView.Columns.Add(DeleteBtn);

   for (int i = 0; i < TasksLGridView.Columns.Count; i++) {
    TasksLGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
   }
  }
 }

 private void AddBtn_Click(object sender, EventArgs e) {
  if (IsSelectedCorrect2() && IsDataEnteringCorrect()) {
   TasksL selTasksL = new TasksL();
   selTasksL.TasksId = _SelectedTask.TasksId;
   selTasksL.UsersId = Convert.ToInt32(UsersCBox.SelectedValue);
   selTasksL.FIO = UsersCBox.Text;
   _TasksLList.Add(selTasksL);
   LoadDataInTasksL(_TasksLList);
  }
 }
 private void TasksLGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
  if (e.ColumnIndex == 3 && TasksLGridView[0, e.RowIndex].Value.ToString() != _GroupsLList[0].Message) {
   int usersId = Convert.ToInt32(TasksLGridView[0, e.RowIndex].Value.ToString());
```

```
    for (int i = 0; i < _TasksLList.Count; i++) {
     if (usersId == _TasksLList[i].UsersId) {
      _TasksLList.RemoveAt(i);
      break;
     }
    }
    LoadDataInTasksL(_TasksLList);
   }
  }

  private void AssignLBtn_Click(object sender, EventArgs e) {
   _TasksLProvider.InsertBatchTasksLList(_TasksLList, _SelectedTask.TasksId);
   MessageBox.Show(«Список користувачів збережено для цього завдання!», «Зберігання»);
  }

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TasksApp.AppCode;
using TasksApp.Forms.Dictionary;
using TasksApp.Forms.Systems;
using TasksApp.Provider;

namespace TasksApp.Forms.Controls {
 public partial class MyTasksForm : Form {
  TasksProvider _TasksProvider = new TasksProvider();
  List<Tasks> _TasksList = new List<Tasks>();
  List<Tasks> _UntilExecutionList = new List<Tasks>();
  List<Tasks> _InProcessList = new List<Tasks>();
  List<Tasks> _CompletedList = new List<Tasks>();
  StatusApp _StatusApp = new StatusApp();
  List<Status> _StatusList = new List<Status>();
  public MyTasksForm() {
   InitializeComponent();
   LoadAllData();
  }

  private void LoadAllData() {
   _StatusList = _StatusApp.GetStatusList();
   _TasksList = _TasksProvider.GetAllTasksByUserId(LoginForm.CurrentUser.UsersId);
   _UntilExecutionList.Clear();
   _InProcessList.Clear();
   _CompletedList.Clear();
   for (int i = 0; i < _TasksList.Count; i++) {
    if (_TasksList[i].Status == _StatusList[0].StatusName) {
     _UntilExecutionList.Add(_TasksList[i]);
    } else if (_TasksList[i].Status == _StatusList[1].StatusName) {
```

```
    _InProcessList.Add(_TasksList[i]);
   } else {
    _CompletedList.Add(_TasksList[i]);
   }
  }
  LoadDataInUntilExecutionGridView(_UntilExecutionList);
  LoadDataInInProcessGridView(_InProcessList);
  LoadDataInCompletedGridView(_CompletedList);
 }


 private void LoadDataInUntilExecutionGridView(List<Tasks> UntilExecutionList) {
  UntilExecutionGridView.DataSource = null;
  UntilExecutionGridView.Columns.Clear();
  UntilExecutionGridView.AutoGenerateColumns = false;
  UntilExecutionGridView.RowHeadersVisible = false;
  UntilExecutionGridView.DefaultCellStyle.WrapMode = DataGridViewTriState.True;
  UntilExecutionGridView.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
  UntilExecutionGridView.DataSource = UntilExecutionList;

  if (UntilExecutionList.Count > 0) {
   if (UntilExecutionList[0].Message == NamesMy.NoDataNames.NoDataInTasks) {
    DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
    messageColumn.DataPropertyName = «Message»;
    messageColumn.Width = UntilExecutionGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
    UntilExecutionGridView.Columns.Add(messageColumn);
   } else {
    DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
    DetailIdColumn.DataPropertyName = «TasksId»;
    UntilExecutionGridView.Columns.Add(DetailIdColumn);
    UntilExecutionGridView.Columns[0].Visible = false;

    DataGridViewColumn TasksNameColumn = new DataGridViewTextBoxColumn();
    TasksNameColumn.HeaderText = «Задачі»;
    TasksNameColumn.DataPropertyName = «TasksName»;
    TasksNameColumn.Width = 200;
    UntilExecutionGridView.Columns.Add(TasksNameColumn);

    DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
    StartDateColumn.HeaderText = «Дата закінчення»;
    StartDateColumn.DataPropertyName = «ActualTerm»;
    StartDateColumn.DefaultCellStyle.Format = «dd/MM/yyyy»;
    StartDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    StartDateColumn.Width = 100;
    UntilExecutionGridView.Columns.Add(StartDateColumn);

    DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
    numberColumn.HeaderText = «Приоритет»;
    numberColumn.DataPropertyName = «Priorites»;
    numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    numberColumn.Width = 85;
    UntilExecutionGridView.Columns.Add(numberColumn);
   }
```

```
    for (int i = 0; i < UntilExecutionGridView.Columns.Count; i++) {
     UntilExecutionGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
   }
  }

  private void LoadDataInInProcessGridView(List<Tasks> InProcessList) {
   InProcessGridView.DataSource = null;
   InProcessGridView.Columns.Clear();
   InProcessGridView.AutoGenerateColumns = false;
   InProcessGridView.RowHeadersVisible = false;
   InProcessGridView.DefaultCellStyle.WrapMode = DataGridViewTriState.True;
   InProcessGridView.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
   InProcessGridView.DataSource = InProcessList;

   if (_UntilExecutionList.Count > 0) {
    if (_UntilExecutionList[0].Message == NamesMy.NoDataNames.NoDataInTasks) {
     DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
     messageColumn.DataPropertyName = «Message»;
     messageColumn.Width = InProcessGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
     InProcessGridView.Columns.Add(messageColumn);
    } else {
     DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
     DetailIdColumn.DataPropertyName = «TasksId»;
     InProcessGridView.Columns.Add(DetailIdColumn);
     InProcessGridView.Columns[0].Visible = false;

     DataGridViewColumn TasksNameColumn = new DataGridViewTextBoxColumn();
     TasksNameColumn.HeaderText = «Задачі»;
     TasksNameColumn.DataPropertyName = «TasksName»;
     TasksNameColumn.Width = 200;
     InProcessGridView.Columns.Add(TasksNameColumn);

     DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
     StartDateColumn.HeaderText = «Дата закінчення»;
     StartDateColumn.DataPropertyName = «ActualTerm»;
     StartDateColumn.DefaultCellStyle.Format = «dd/MM/yyyy»;
     StartDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
     StartDateColumn.Width = 100;
     InProcessGridView.Columns.Add(StartDateColumn);

     DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
     numberColumn.HeaderText = «Приоритет»;

     numberColumn.DataPropertyName = «Priorites»;
     numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
     numberColumn.Width = 85;
     InProcessGridView.Columns.Add(numberColumn);
    }
    for (int i = 0; i < InProcessGridView.Columns.Count; i++) {
     InProcessGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
   }
```

```
}

private void LoadDataInCompletedGridView(List<Tasks> CompletedList) {
 CompletedGridView.DataSource = null;
 CompletedGridView.Columns.Clear();
 CompletedGridView.AutoGenerateColumns = false;
 CompletedGridView.RowHeadersVisible = false;
 CompletedGridView.DefaultCellStyle.WrapMode = DataGridViewTriState.True;
 CompletedGridView.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
 CompletedGridView.DataSource = CompletedList;

 if (_UntilExecutionList.Count > 0) {
  if (_UntilExecutionList[0].Message == NamesMy.NoDataNames.NoDataInTasks) {
   DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
   messageColumn.DataPropertyName = «Message»;
   messageColumn.Width = CompletedGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
   CompletedGridView.Columns.Add(messageColumn);
  } else {
   DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
   DetailIdColumn.DataPropertyName = «TasksId»;
   CompletedGridView.Columns.Add(DetailIdColumn);
   CompletedGridView.Columns[0].Visible = false;

   DataGridViewColumn TasksNameColumn = new DataGridViewTextBoxColumn();
   TasksNameColumn.HeaderText = «Задачі»;
   TasksNameColumn.DataPropertyName = «TasksName»;
   TasksNameColumn.Width = 200;
   CompletedGridView.Columns.Add(TasksNameColumn);

   DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
   StartDateColumn.HeaderText = «Дата закінчення»;
   StartDateColumn.DataPropertyName = «ActualTerm»;
   StartDateColumn.DefaultCellStyle.Format = «dd/MM/yyyy»;
   StartDateColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
   StartDateColumn.Width = 100;
   CompletedGridView.Columns.Add(StartDateColumn);

   DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
   numberColumn.HeaderText = «Приоритет»;
   numberColumn.DataPropertyName = «Priorites»;

   numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
   numberColumn.Width = 85;
   CompletedGridView.Columns.Add(numberColumn);
  }
  for (int i = 0; i < CompletedGridView.Columns.Count; i++) {
   CompletedGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
  }
 }
}

private void UntilExecutionGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
```

```
            if (e.RowIndex >= 0 && UntilExecutionGridView[0, e.RowIndex].Value.ToString() != _UntilExecutionList[0].Message) {
              UpdateTasksForm updateTasksForm = new UpdateTasksForm(Convert.ToInt32(UntilExecutionGridView[0,
e.RowIndex].Value.ToString()));
              updateTasksForm.ShowDialog();
              LoadAllData();
            }
           }

           private void InProcessGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
            if (e.RowIndex >= 0 && InProcessGridView[0, e.RowIndex].Value.ToString() != _InProcessList[0].Message) {
              UpdateTasksForm updateTasksForm = new UpdateTasksForm(Convert.ToInt32(InProcessGridView[0,
e.RowIndex].Value.ToString()));
              updateTasksForm.ShowDialog();
              LoadAllData();
            }
           }

           private void CompletedGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
            if (e.RowIndex >= 0 && CompletedGridView[0, e.RowIndex].Value.ToString() != _CompletedList[0].Message) {
              UpdateTasksForm updateTasksForm = new UpdateTasksForm(Convert.ToInt32(CompletedGridView[0,
e.RowIndex].Value.ToString()));
              updateTasksForm.ShowDialog();
              LoadAllData();
            }
           }
          }
         }
```