

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАНУВАННЯ І ОБЛІКУ
ВИКОНАННЯ РОБОЧИХ ЗАВДАНЬ
SOFTWARE FOR PLANNING AND TRACKING
THE PERFORMANCE OF TASKS

Виконав: студент(ка) 4 курсу, групи КНТ-137
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Дмитрієв І.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Дмитрієва Іллі Олеговича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення планування і обліку виконання робочих завдань
Software for Planning and Tracking the Performance of Tasks

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 20 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “10” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

_____ Дмитрієв І.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

_____ Льовкін В.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 80 с., 27 рис., 2 табл., 3 дод., 14 джерел.

РОБОЧИ ЗАВДАННЯ, ПЛАНУВАННЯ ЗАВДАНЬ, ОБЛІК ЗАВДАНЬ, ВЕБЗАСТОСУНОК, ASP.NET, MODEL-VIEW-CONTROLLER.

Об'єкт роботи – процес планування робочих завдань. Предмет роботи – програмне забезпечення планування і обліку виконання робочих завдань. Мета роботи – розробити програмне забезпечення планування і обліку виконання робочих завдань для підтримки командної роботи.

Матеріали, методи та технічні засоби: мова програмування C#, фреймворки веброзробки ASP.NET Core MVC, система керування базами даних Microsoft SQL Server.

Результати. Розроблено програмне забезпечення, побудоване у вигляді вебзастосунку на основі багатоваршівної архітектури, що включає шар доступу до даних, побудований на основі використання технології Entity Framework, шар бізнес-логіки, презентаційний шар, побудований на основі використання шаблону MVC. Створено базу даних, визначено користувацькі інтерфейси під всі режими роботи програми.

Висновки. Програмне забезпечення призначене для керування завданнями, що виникають в процесі індивідуальної та командної роботи, щодо їх планування і обліку. Програма надає можливість створення послідовностей завдань, підтримує роботу однієї компанії над завданнями, дозволяє працювати над завданнями у відрядженнях або в розподілено розташованих відділеннях компанії.

Галузь використання. Компанії різного профілю в частині управління робочими завданнями.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 27 figures, 2 tables, 3 appendixes, 14 sources.

WORK TASKS, TASK PLANNING, TASK TRACKING, WEB APPLICATION, ASP.NET, MODEL-VIEW-CONTROLLER.

The object of work is the process of planning work tasks. The subject of work is software for planning and tracking of work tasks. The purpose of the work is to develop software for planning and tracking of work tasks to support teamwork.

Materials, methods and technical means: C# programming language, ASP.NET Core MVC web development frameworks, Microsoft SQL Server database management system.

Results. Developed software built in the form of a web application based on a multilayer architecture, including a layer of data access based on Entity Framework technology, a layer of business logic, a presentation layer based on the use of the MVC template. A database has been created, user interfaces have been defined for all operating modes of the program.

Conclusions. The software is designed to manage tasks that arise in the process of individual and team work, its planning and tracking. The program provides the ability to create sequences of tasks, supports the work of one company on tasks, allows users to work on tasks on business trips or in distributed branches of the company.

Field of use. Companies of different profiles in terms of task management.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Основні принципи організації програмного забезпечення планування і обліку виконання робочих завдань.....	10
1.2 Огляд програмних аналогів.....	12
1.3 Висновки за розділом 1	18
2 Проєктування програмного забезпечення	19
2.1 Вибір технологій розробки	19
2.2 Вимоги до програмного забезпечення	21
2.3 Проєктування бази даних	25
2.4 Визначення архітектури програмного забезпечення.....	30
2.5 Висновки за розділом 2	31
3 Основні рішення щодо реалізації компонентів системи.....	32
3.1 Опис логіки роботи програми.....	32
3.2 Опис логіки реалізації.....	36
3.3 Висновки за розділом 3	40
4 Експлуатація, тестування та експериментальне дослідження програми	41
4.1 Призначення програми	41
4.2 Умови виконання програми	41
4.3 Тестування сценаріїв роботи з програмою.....	41
4.4 Висновки за розділом 4	50
Висновки	51
Перелік джерел посилення	52
Додаток А Технічне завдання	54
Додаток Б Текст програми	58
Додаток В Слайди презентації.....	74

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ASP.NET – Active Server Pages для .NET;

MVC – Model-View-Controller.

ВСТУП

Проблема управління власним часом є достатньо відомою, однак зі збільшенням активностей і кількості інформації, що виникає перед людиною, вона постає заново і заново і потребує ефективних інструментів її розв'язання. Одночасно разом з управлінням часом людини, що використовується в персональних цілях, існує проблема управління її робочими завданнями. Такі системи існують зазвичай паралельно з системами управління проєктами, адже потребують зокрема засобів, які дозволяють виявити дані не тільки всередині проєкту, а і загалом, консолідуючи всі існуючі завдання разом. Робочі завдання мають певну специфіку, адже зазвичай вони ставляться однією стороною, а виконуються іншою. Тобто можна розділити учасників на керівників та виконавців. Це вимагає певних засобів організації роботи програми.

Незважаючи на те, що систем управління завданнями існує достатньо багато, основна частина з них направлена на індивідуальні завдання, часом інтегруючи можливість поділитися переліками завдань з іншими користувачами.

Робочі завдання зазвичай існують в межах проєктів, але не завжди підпадають тільки під цей шаблон. Проєкти мають етапи, що впливають один на одного, відповідно врахування цих послідовностей важливо і для управління завданнями.

Існуючі на даний момент програмні засоби є універсальними рішеннями, а багато компаній потребують певної специфіки. Такі специфічні інструменти зазвичай частково існують в одній програмі, але відсутні в іншій. Для вирішення цієї проблеми як раз і розробляється програмне забезпечення в даній дипломній кваліфікаційній роботі. Тому проблему можна вважати актуальною.

Метою роботи в результаті було встановлено розробити програмне забезпечення планування і обліку виконання робочих завдань для підтримки командної роботи.

Об'єкт роботи – процес планування робочих завдань. Предмет роботи – програмне забезпечення планування і обліку виконання робочих завдань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні принципи організації програмного забезпечення планування і обліку виконання робочих завдань

Програмне забезпечення управління завданнями охоплює управління завданнями на різних етапах його життєвого циклу, включаючи планування, відслідковування, облік часу, облік виконання, тестування, створенні звітів про виконання. У роботі заплановано зробити акцент саме на плануванні завдань, а також обліку їх виконання, не включаючи в цей процес облік часу. Облік виконання необхідний для відслідковування роботи над завданнями, щоби можна було створювати переліки проєктів і виключати з них ті завдання, які вже було виконано. Облік часу роботи над завданнями може бути темою окремої роботи.

Програмне забезпечення планування і обліку виконання робочих завдань в якості основних своїх функціональних можливостей повинно забезпечувати:

– планування завдань: ефективне управління завданнями починається з відповідного планування, залежно від бюджету, ресурсів, обсягу та термінів, ефективне планування завдань вимагає розбиття роботи на етапи за допомогою структурованого підходу, за допомогою якісного програмного забезпечення для управління завданнями можна розбити свою роботу на простіші завдання, якими легко керувати, окрім того, для забезпечення планування повинна бути можливість організувати та розставити пріоритети завдань за потребою, до того ж ідея функцій програмного забезпечення для управління завданнями полягає в організації щоденної роботи, щоб зробити її більш впорядкованою, хороший інструмент управління повинен дозволяти створювати розклад, при плануванні щодня, щотижня або щомісяця стає складніше пропустити важливі завдання, при цьому допомагаючи контролювати прогрес, сповіщення та нагадування роблять розклад завдань

ще кращим, оскільки вони тримають користувача у курсі і можуть заощадити від перевитрати часу на одне завдання;

– повторювані завдання: інша можливість, яку мають найкращі інструменти управління – це можливість створити завдання та планувати його на декілька днів для різних цілей, деякі завдання потрібно повторювати щодня, окрім того одне завдання може часом служити багатьом проєктам, наприклад, завдання оновлення табелів обліку робочого часу до кінця тижня можна запланувати щодня, щотижня або щомісяця для декількох команд одночасно;

– пріоритет: коли в переліку багато завдань, природно заплутатися у розбірці завдань за пріоритетом, ефективні функції програмного забезпечення для управління завданнями можуть допомогти в цьому, дозволивши в якості одного з параметрів використовувати розстановку пріоритетів завдань, задачі, як правило, визначаються пріоритетними на основі рівня актуальності самого завдання або залежності інших завдань, хороший інструмент повинен мати можливість забезпечити механізм визначення таких пріоритетів разом із завданнями, це допомагає в першу чергу завершити термінову роботу та пришвидшити реалізацію критичних етапів проєкту;

– відстеження завдань: коли заплановано низку завдань, особливо в командах, відстеження всіх виконаних завдань може бути складним завданням, для того, щоб полегшити цей процес, доцільно застосувати інструмент, який допоможе точно простежити, які саме дії розпочаті, що виконуються та з яким результатом завершено, правильно визначаючи завдання з точки зору членів команди, на яких вони покладені, проєкту, на який вони потрапляють, та рівня їх актуальності, можна отримати додаткову перевагу пошуку та фільтрації завдань за будь-яким із критеріїв, коли можна відстежувати завдання, можна спростити завдання та планування проєктів, заощадивши час на непотрібному відтворенні завдань;

– командна робота: у командах співпраця йде паралельно з функціями програмного забезпечення для управління завданнями, тому завжди має використовуватися програмне забезпечення для управління завданнями для вдосконалення співпраці команд, складні або неефективні програми можуть сильно вплинути на комунікацію, що, в свою чергу, може негативно вплинути на управління завданнями, ефективний канал співпраці – це ключовий елемент, який потрібно шукати на платформі управління завданнями, команда повинна мати можливість обмінюватися інформацією, ділитися завданнями та отримувати оновлення завдань [1]-[2].

1.2 Огляд програмних аналогів

Asana [3] (рис. 1.1) – це інструмент для управління завданнями та іншої роботи, направлений на роботу з групами людей.

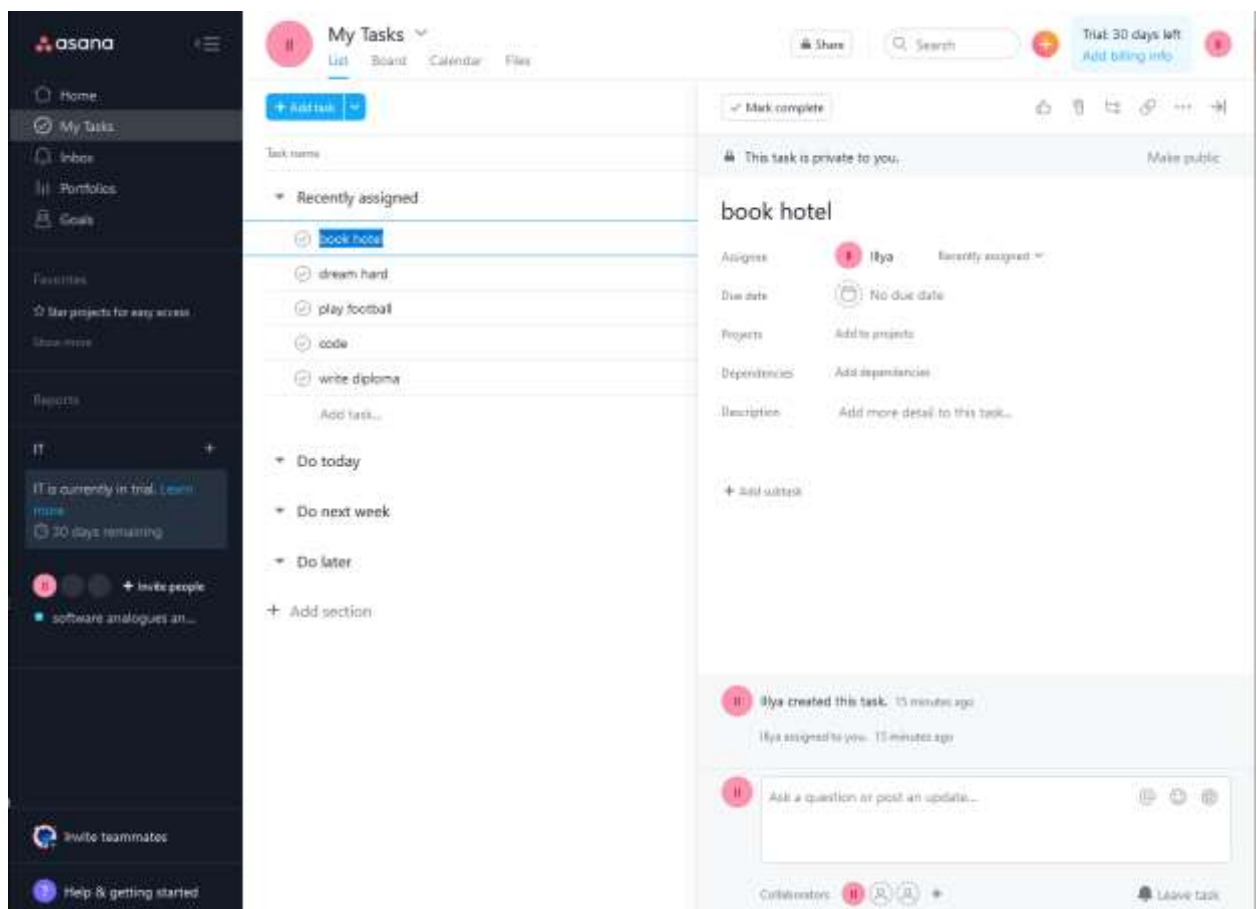


Рисунок 1.1 – Інтерфейс програми Asana

Asana відзначається наступним:

- відстежує, хто відповідає за якесь завдання, та всю інформацію, пов'язану з цим, включаючи всі необхідні кроки, терміни виконання тощо;
- однією з особливостей є гнучкість, тобто користувач вирішує, яку роботу слід відстежувати та як саме її використовувати
- Asana дозволяє працювати і з завданнями, що входять до складу проєктів, але відмінністю від програм для управління проєктами є те, що вони створені спеціально для обробки складних проєктів, керують графіками та ресурсами, а іноді і бюджетами, Asana вміє обробляти менш складні проєкти, де управління ресурсами та графіками не є невід'ємною частиною роботи (маркетингова кампанія, запуск продукту або збір коштів);
- кожне завдання може мати підзадачі, а також наступника, термін виконання, вкладення, коментарі, теги та послідовників: послідовники отримують оновлення щоразу, коли відбувається зміна завдання;
- будь-яке завдання, а також усі його підзадачі та налаштування можуть стати робочим процесом: наприклад, якщо загальним завданням команди є підписання нових клієнтів, підзадачі охоплюватимуть усі кроки, такі як підтвердження зацікавленості клієнта, надсилання їм необхідних документів, отримання повернених документів, надсилання їм вітального електронного листа тощо, за кожен крок можуть відповідати різні люди, і за допомогою Asana можна призначити кожен крок відповідній особі та додати відповідний термін виконання в потрібний час;
- Asana дозволяє перетворити завдання та всі його підзадачі на шаблон, щоб, коли створюється нове завдання в проєкті, було отримали його за замовчуванням;
- теги допомагають зробити завдання більш зручними для пошуку, наявна функція розширеного пошуку, за допомогою спеціальних полів можна додати деталь до завдання, яка є важливою для користувача і його команди, наприклад, пріоритет завдання (низький, середній, високий) або стан його виконання (очікує на розгляд, очікує на затвердження);

– ще одна корисна особливість – здатність програми створювати залежності між завданнями: якщо є завдання А і завдання В, а завдання В можна розпочати лише після завершення завдання А, тоді можна додати залежність між ними, щоб виразити цей зв'язок, для візуалізації таких залежностей стане в нагоді діаграма Ганта, яка буде отримана на шкалі часу, тут користувачі можуть побачити кожне своє завдання, викладене у форматі діаграми Ганта [4]-[5].

Asana має чотири рівні обслуговування:

- базовий (безкоштовний);
- преміум (13,49 доларів США на людину на місяць);
- бізнес (30,49 доларів США на людину на місяць);
- корпоративний, вартість якого встановлюється індивідуально, при цьому можна отримати деяку знижку за сплати одразу щорічно, проте можна відзначити, що раніше невеликі команди мали право на знижку на оновлення, але ця пропозиція більше не діє [4].

Todoist [6] є програмою управління завданнями, яка за зручністю роботи вважається на даний момент найкращим варіантом на ринку.

Todoist має три рівні обслуговування: безкоштовний, преміум (4 долари на місяць або 36 доларів на рік) та бізнес (6 доларів на людину на місяць або 60 доларів на людину на рік), компанія надає знижки студентам, викладачам та некомерційним організаціям, безкоштовний обліковий запис пропонує більш ніж достатній досвід, але він утримує кілька функцій, які можуть знадобитися (нагадування, мітки, фільтри, різні кольорові теми для інтерфейсу і можливість коментувати завдання та завантажувати файли у цій версії відсутні), при цьому за допомогою безкоштовного облікового запису можна співпрацювати з 5 людьми на проєкт і керувати до 80 проєктами [7].

Todoist відзначається наступним:

- налаштування програми є простим: за логікою роботи спочатку створюються проєкти, проєкти можуть мати кольорове кодування, щоб допомогти розрізнити їх, потім ви додаються завдання до проєктів і стільки

деталей до них, скільки потрібно, завдання можуть мати підзавдання, можна змінити порядок завдань та підзавдань, перетягуючи їх, завдання також можуть мати нагадування та мітки, які представляють собою теги, доступні сповіщення електронною поштою, push-сповіщення, нагадування на основі місцезнаходження (лише Premium-версія);

- під час перегляду завдань можна створювати власні фільтри – подання завдань, які відповідають певним критеріям: за замовчуванням користувач отримує перегляд завдань на сьогодні (усі завдання із терміном виконання сьогодні) та майбутні періоди (завдання з термінами на найближчі дні), наприклад, за допомогою спеціальних фільтрів можна створити представлення, яке відображатиме лише ті завдання, що виконуються сьогодні або завтра, що мають однаковий тег, незалежно від проєкту;

- командна співпраця: спочатку потрібно запросити співавторів до одного зі своїх проєктів, їм потрібно буде зареєструватися в обліковому записі Todoist, але їм не потрібно переходити на обліковий запис рівня преміум, після того, як співавтор прийме запрошення приєднатися до проєкту, можна призначити їм завдання, і співавтори також можуть призначити завдання користувачу, який їх запросив, можна завантажити файл і додати коментарі до будь-якого завдання, щоб поділитися інформацією зі своїми співавторами, можна налаштувати одержувані сповіщення, якщо не потрібно знати про кожну незначну зміну, яку вносять співавтори [7]-[9].

Інтерфейс програми наведено на рис. 1.2.

За результатами проведеного огляду було виконано порівняння програмних аналогів (Asana, Todoist) з програмою, що розробляється в роботі. Обидва аналоги є достатньо функціональними і адекватними рішеннями для використання у цій предметній області, окрім того це два різних, але популярних програмних засобів у цій сфері.

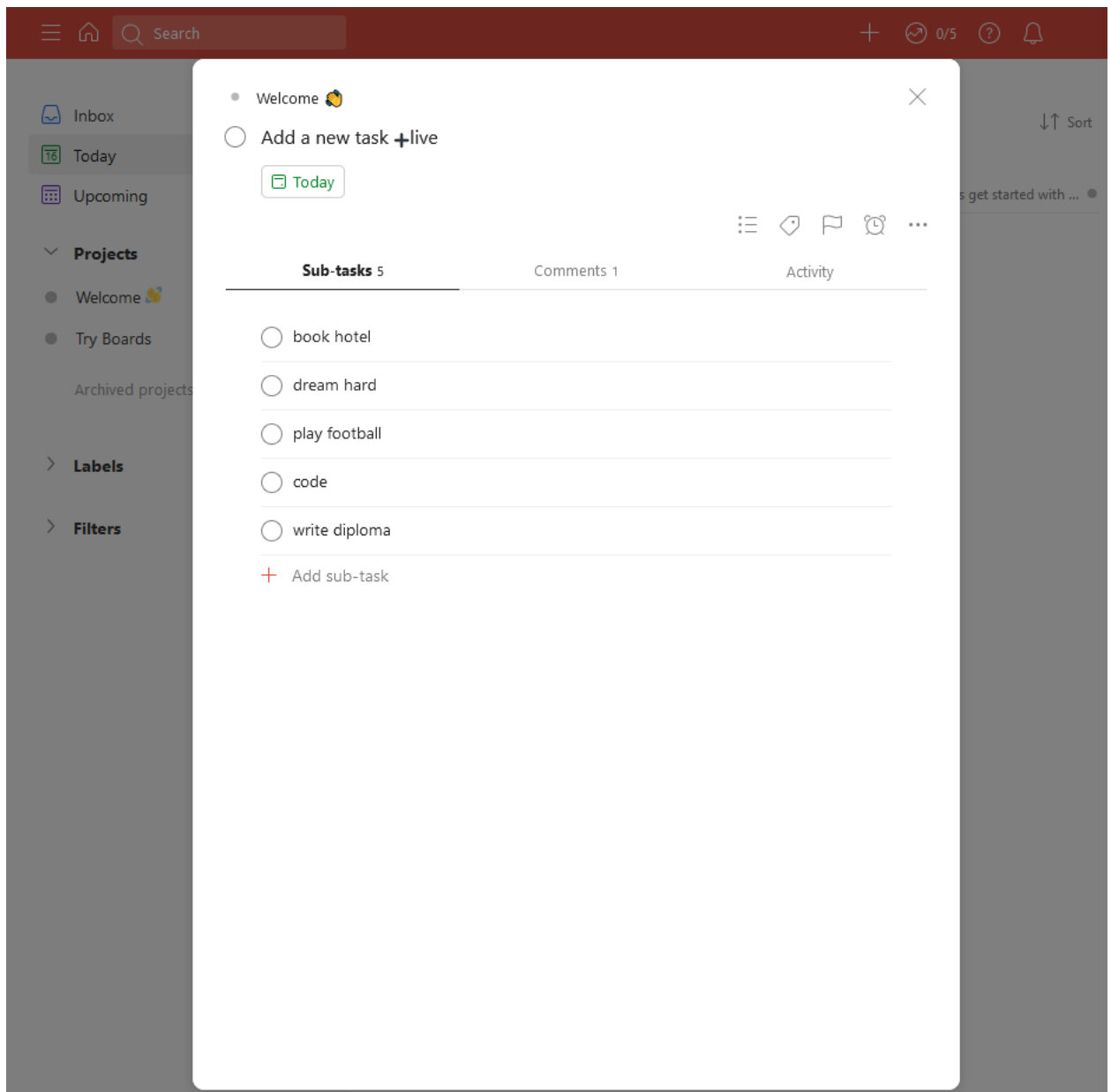


Рисунок 1.2 – Інтерфейс програми Todoist

Однією з головних особливостей програми Asana є створення проєктів на основі шаблонів. Дана функціональна особливість відсутня в Todoist і планується до впровадження в програмі, що розробляється.

Todoist має засоби командної роботи, але щодо розподілення між ними завдань проєкту має певні обмеження, зокрема не дозволяє розподіляти відсоткові співвідношення між учасниками. Дана особливість є в програмі Asana і планується до впровадження в програмі, що розробляється.

Особливістю Todoist, що відсутня в Asana, є планування на основі локацій, тобто прив'язування завдань до появи користувача у певному місці. Ця особливість є корисною на практиці, наприклад, для розподілення завдань між звичайним режимом роботи та відрядженнями. Вона також буде впроваджена в програмі, що розробляється.

Результати порівняння зведені в табл. 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

	Asana	Todoist	DmiManager (програма, що розробляється)
Планування завдань	+	+	+
Створення власних шаблонів проєктів	+	—	+
Розділення відповідальності між учасниками команди	+	—	+
Планування на основі локацій	—	+	+
Безкоштовність	—	—	+

Окрім того Todoist для професійного використання (робочі завдання як раз і є таким варіантом) потребує сплати за використання, що може бути незручно для невеликих команд, тільки створених.

Таким чином, за рахунок об'єднання ключових особливостей програмних аналогів буде досягнуто переваги програми, що розробляється.

1.3 Висновки за розділом 1

Виконаний аналіз проблеми дозволив виділити основні принципи організації програмного забезпечення планування і обліку виконання робочих завдань, проаналізувати і порівняти програмні аналоги, на основі чого виділити особливості програмного забезпечення, що розробляється.

Для того щоб досягти поставлену мету, в роботі було визначено наступні завдання:

- виконати аналіз предметної області, що було досягнуто в даному розділі;
- виділити вимоги до програмного забезпечення;
- виконати проєктування програмного забезпечення;
- виконати реалізацію програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір технологій розробки

У процесі вибору технологій розробки спочатку було прийнято рішення щодо вибору мови програмування. За результатами порівняння мов C# та Python було обрано C# [10], оскільки ця мова дозволяє створювати більш продуктивні програми в результаті, має більший потенціал розширення, зокрема за рахунок можливості створення застосунків для iOS, Android, а це є важливим фактором для даної предметної області, адже створення такого програмного забезпечення передбачає в подальшому розробку і мобільних застосунків також. Результати порівняння зведено до табл. 2.1.

Таблиця 2.1 – Вибір мови програмування

	C#	Python
Продуктивність програми	Більше	Менше
Створення вебзастосунків	Так	Так
Потенціал розширення	Більше (можна створити застосунки для iOS, Android)	Менше

На наступному етапі було виконано вибір технології веброботи. У роботі буде створено саме вебрішення, оскільки воно є універсальним і надає можливість роботи з сервером з різних пристроїв через веббраузер. Усі інші рішення призводять до того, що потрібно було створити застосунок, який би використовувався тільки на частині платформ (десктопних, одній з десктопних або одній з мобільних), що для початкового входження в галузь звужує кількість потенційних користувачів.

Веброзробка буде виконуватися на основі технології Active Server Pages для .NET (ASP.NET) Core [11] – це кросплатформний фреймворк з відкритим кодом для розробки Інтернет-орієнтованих сучасних вебзастосунків на базі хмарних технологій, що відзначається наступними перевагами:

- покращена продуктивність: найбільш очевидною і важливою перевагою платформи ASP.NET Core є її більш висока продуктивність, з новими вдосконаленнями, а також оновленнями, код стає набагато більш оптимізованим, що призводить до поліпшення продуктивності, компілятор ASP.NET Core оптимізує весь код при кожному повторному компілюванні із використанням платформи ASP.NET Core, фактична продуктивність ASP.NET Core в разі перевищує будь-яку з відомих реалізацій фреймворку;

- підтримка кросплатформності: коли справа стосується розробки вебзастосунків, дуже важливо переконатися, що програма насправді підтримує всі платформи, остання версія ASP.NET Core насправді є кросплатформною, що дозволяє легко створювати вебзастосунки, що працюють на Windows, Linux та Mac, використовуючи Xamarin, можна легко створити застосунок для iOS, а потім з часом використовувати той самий код для створення застосунка для Android;

- менше коду: нова технологія вимагає менше кодування, що означає, що розробники можуть легко оптимізувати структуру коду шляхом написання значно менших виразів, оскільки кодування менше, менше часу потрібно для того, щоб створити програму, що робить ASP.NET Core набагато економічнішим;

- простіше обслуговування: коли коду менше, його стає набагато простіше автоматично підтримувати, при цьому на створення вебпрограми потрібна не тільки менша кількість коду, але і набагато простіше ефективно керувати нею та підтримувати її, це покращує якість розробки застосунків Microsoft [12].

2.2 Вимоги до програмного забезпечення

Програма повинна дозволяти виконувати робочі завдання як визначені менеджером, так і власні. Такий підхід дозволяє працювати як індивідуально, так і командно. При цьому програма передбачає роботу однієї компанії з програмою.

Передбачається, що частина завдань може бути видимою тільки менеджеру і працівнику, якому воно призначено. Усі інші завдання видні всім працівникам, які належать до команди, що працює над проєктом.

Особливістю також є створення розкладу працівника за місцем виконання. Приєднання завдань до місця виконання дозволяє створити, наприклад, розклад для відряджень. Таким чином менеджер може визначити завдання для відрядження, а усі інші завдання в цей момент не будуть відображатися працівнику, як тільки він буде знаходитися в заданому місці. Таким чином можна ще створювати окремі завдання для реалізації в різних відділеннях компанії.

У результаті програмне забезпечення, що розробляється, повинно виконувати наступні функції:

- створення проєктів;
- визначення учасників проєкту;
- визначення робочого завдання;
- визначення стану робочих завдань;
- визначення строків виконання робочого завдання;
- визначення відсотку виконання робочого завдання;
- визначення кольору завдання;
- визначення місця виконання завдання;
- встановлення пріоритету робочому завданню;
- визначення підзавдань до завдання;
- редагування робочого завдання;
- вилучення робочих завдань;

- визначення завершення завдання;
- перегляд стану виконання проєкту;
- формування розкладу завдань для користувача;
- перегляд всіх власних завдань;
- перегляд власних завдань з найменшим відсотком виконання;
- перегляд завдань, не прикріплених до проєктів;
- формування розкладу завдань за місцем виконання;
- перегляд підзавдань завдання;
- пошук завдань;
- перегляд завдань користувача за проєктом;
- перегляд статистики виконання завдань за працівником;
- перегляд статистики виконання завдань за кольором;
- перегляд завдань за часовим проміжком;
- перегляд завдань з найбільшим відхиленням часу виконання;
- перегляд історії роботи над завданням;
- створення проєкту за шаблоном;
- авторизація користувача;
- реєстрація облікового запису працівника менеджером.

Для реалізації функціональних можливостей в межах описаної вище ідеї усі користувачі розділені на працівників та менеджерів.

Менеджерам дозволено керувати проєктами, відповідно і завданнями, які виконуються в межах цих проєктів.

Працівники можуть виконувати призначені для них завдання, а також визначати завдання без проєктів, відслідковуючи за потреби власну роботу над ними.

Сценарії роботи менеджера з програмою (рис. 2.1) включають:

- створення проєктів;
- визначення учасників проєкту;
- керування завданнями проєкту;

- пошук завдань;
- перегляд завдань;
- перегляд статистики;
- реєстрація облікового запису працівника.

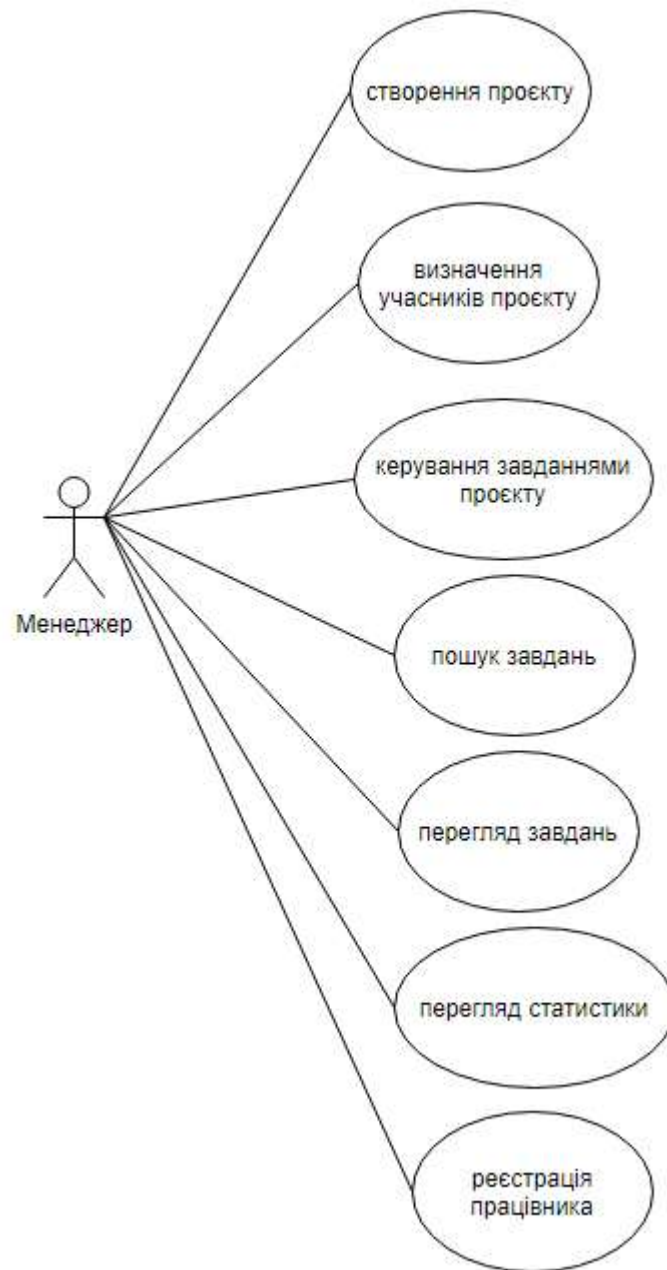


Рисунок 2.1 – Діаграма прецедентів менеджера

На рис. 2.2 представлено діаграму прецедентів для працівника.

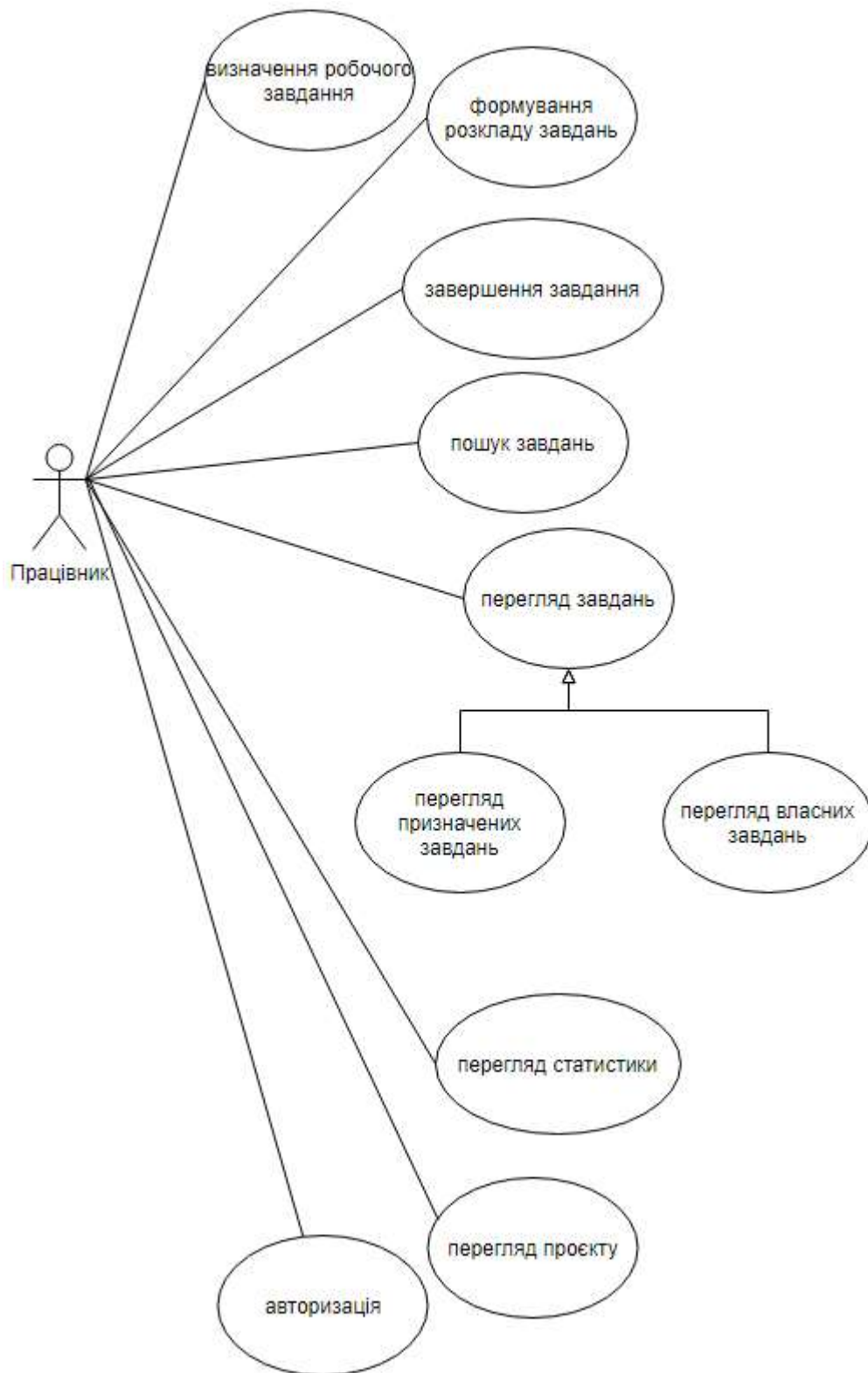


Рисунок 2.2 – Діаграма прецедентів працівника

Сценарії роботи працівника з програмою включають:

- визначення робочого завдання;
- визначення завершення завдання;

- формування розкладу завдань;
- перегляд власних завдань;
- перегляд статистики;
- перегляд призначених завдань;
- пошук завдань;
- перегляд проєкту;
- перегляд статистики;
- авторизація.

2.3 Проєктування бази даних

За результатами аналізу предметної області було виділено наступні основні сутності:

- Task – завдання, роботи з якими і є основою створення програми;
- Project – проєкт, який може об'єднувати завдання на виконання єдиної мети;
- User – користувач, тобто учасник системи, який виконує завдання, створює їх, організовує.

Таблиця User, характеризуючи користувача, включає поля:

- id – номер користувача;
- name – ім'я;
- surname – прізвище;
- position – посада в компанії;
- phone – номер телефону;
- login – логін;
- password – пароль;
- role – роль користувача.

Роль користувача в програмі може бути або менеджерською, або виконавською.

Таблиця Project, характеризуючи проєкт, включає поля:

- id – номер проєкту;
- title – назва проєкту;
- description – опис проєкту;
- started – дата і час початку проєкту;
- deadline – дата і час, до якого повинен завершитись проєкт;
- userId – менеджер проєкту;
- actual – фактичний час завершення проєкту.

Відношення між таблицями User і Project один-до-багатьох: один проєкт може мати тільки одного менеджера, але кожен менеджер може працювати з різними проєктами.

Таблиця ProjectParticipant, характеризуючи участь користувача в проєкті, включає поля:

- id – номер запису;
- projectId – номер проєкту;
- userId – номер користувача.

Таблиця ProjectParticipant реалізує зв'язок багато-до-багатьох між таблицями User і Project: один користувач може брати участь в різних проєктах, а кожен проєкт може мати команду учасників.

Відношення між таблицями Project і ProjectParticipant один-до-багатьох: один проєкт має багато учасників.

Відношення між таблицями User і ProjectParticipant один-до-багатьох.

Оскільки над кожним завданням може працювати одразу декілька учасників, то було виділено таблицю TaskUser для реалізації даної залежності. Для випадків, коли завдання можна розділити на конкретні підзавдання, і делегувати їх на виконання різним користувачам, це вирішується введенням підзавдань. Якщо ж таким чином вирішити цю проблему, не можна, тобто над завданням одночасно працюють декілька користувачів, то для цього було введено дану таблицю. Дана участь користувачів у виконанні завдання не співпадає з поняттям проєктної

команди, адже з учасників команди можуть бути виділені лише декілька для роботи над деяким підзавданням, тобто може бути створена підмножина учасників.

Таблиця TaskUser, характеризуючи участь виконавця у виконанні завдання, включає поля:

- id – номер запису;
- taskId – номер завдання (зовнішній ключ);
- userId – номер користувача, який повинен виконати це завдання (зовнішній ключ);
- description – опис особливостей виконання завдання для працівника.

Таблиця Task, характеризуючи завдання, включає поля:

- id – номер завдання;
- title – коротка назва завдання;
- projectId – проєкт, до якого належить завдання;
- userId – номер користувача, який створив завдання;
- description – опис завдання;
- priority – пріоритет завдання;
- percent – відсоток виконання завдання;
- color – колір завдання;
- orderstart – порядок початку виконання завдання в проєкті (якщо не встановлено конкретну дату початку, а потрібно розпочати виконання завдання після завершення деякого попереднього завдання);
- place – місце, де має виконуватися завдання;
- around – радіус дії місця;
- started – дата і час, коли має бути розпочато виконання завдання;
- deadline – дата і час, до якого має бути завершено виконання завдання;
- order – порядок завдання у проєкті;

- mode – режим роботи над завданням (дозволяє створювати індивідуальні робочі завдання, що не видні жодному іншому учаснику);
- actual – фактичний час завершення виконання завдання.

Відношення між таблицями User і Task один-до-багатьох: один користувач може створити багато завдань, але у кожного завдання може бути тільки один автор.

Оскільки кожне завдання може містити багато завдань, то було виділено таблицю SubTask для зберігання інформації про зв'язок між завданнями, одне з яких є підзавданням, а інше – його батьківським завданням.

Таблиця SubTask, характеризуючи підзавдання, включає поля:

- id – номер запису про підзавдання;
- taskParentId – батьківське завдання, що включає дане підзавдання (зовнішній ключ);
- taskChildId – номер підзавдання (зовнішній ключ);
- taskPercent – відсоток, якому відповідає виконання даного підзавдання, в успіху завдання;
- order – порядок даного підзавдання серед усіх підзавдань.

Таблиця TaskUpdate дозволяє зберігати оновлення станів завдань.

Таблиця включає поля:

- id – номер запису про підзавдання;
- taskId – номер завдання;
- userId – номер користувача;
- percent – відсоток виконання.

Відношення між таблицями Task і TaskUpdate один-до-багатьох: одне завдання може багато разів оновлюватися.

Відношення між таблицями User і TaskUpdate один-до-багатьох: один користувач може оновлювати різні завдання.

Схема бази даних представлена на рис. 2.3.

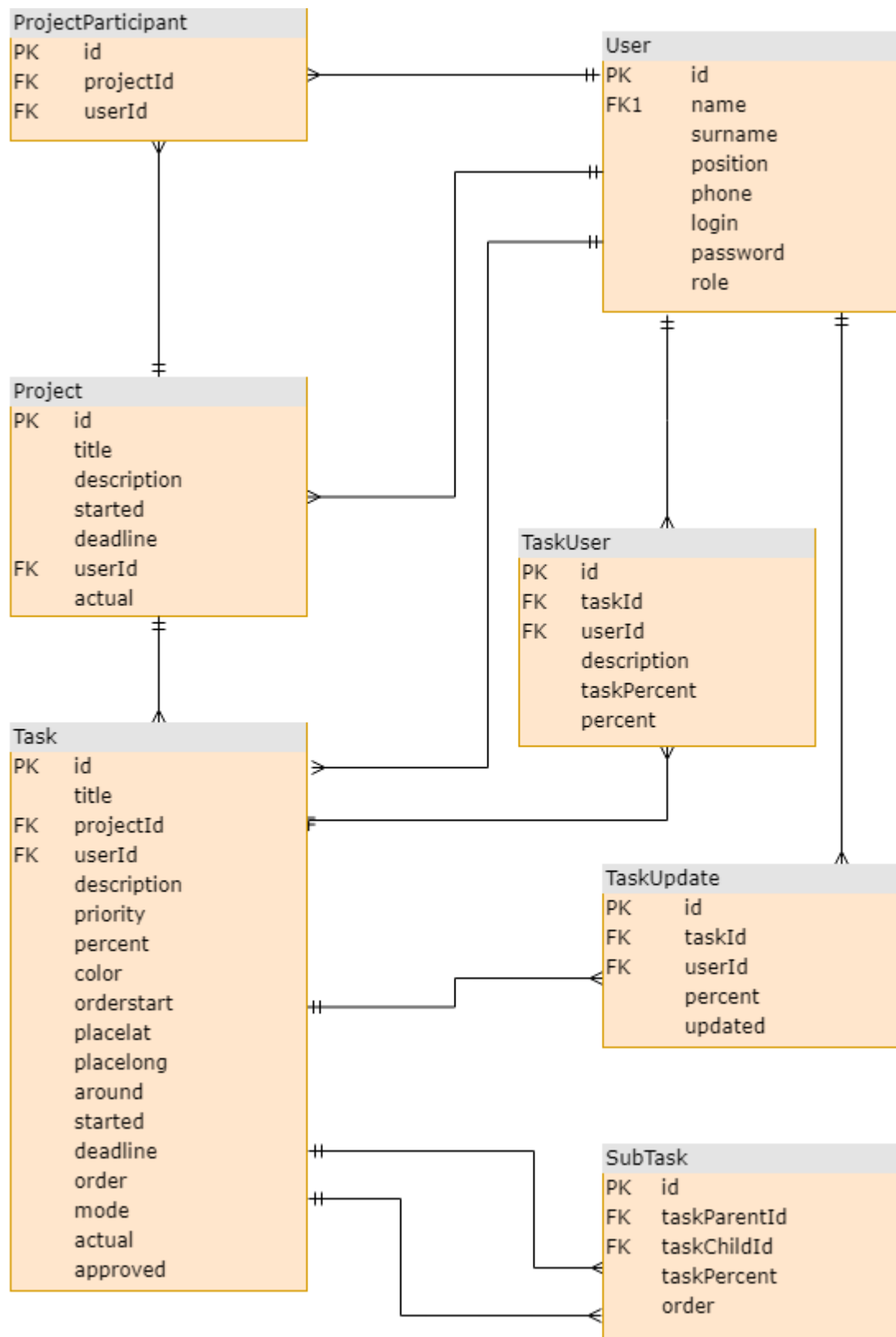


Рисунок 2.3 – Схема бази даних

2.4 Визначення архітектури програмного забезпечення

Архітектуру вебзастосунку побудовано на розділенні його на шари (рис. 2.4):

- презентаційний шар;
- шар бізнес-логіки;
- шар доступу до даних.

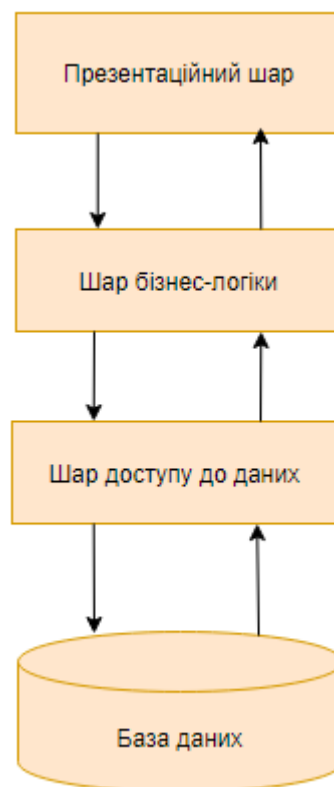


Рисунок 2.4 – Архітектура застосунку

Шар доступу до даних побудовано на основі використання об'єктно-орієнтованої технології для роботи з даними на основі фреймворку .NET, що лежить в основі ASP.NET, Entity Framework [13]. Дана технологія полегшує роботу з даними і забезпечує можливість уникати написання стандартних запитів до бази даних на роботу з таблицями.

Шар доступу до даних визначає контекст бази даних та класи відповідних сутностей.

Шар бізнес-логіки є проміжним шаром, який одночасно є надбудовою над шаром доступу до даних. Цей шар реалізовує особливості роботи з даними, зокрема для вебзастосунку планування і обліку робочих завдань такими особливостями є обчислення завершеності проєкту на основі обчислення завершеності виконання завдань, що входять в цей проєкт, а також обчислення завершеності завдань на основі обчислення завершеності підзавдань.

Презентаційний шар застосунку створений для організації взаємодії з користувачем і побудований на основі архітектурного шаблону Model-View-Controller (MVC) з застосуванням технології ASP.NET Core MVC.

2.5 Висновки за розділом 2

Виконане проєктування програми дозволило забезпечити вибір мови програмування C# і фреймворку розробки вебзастосунків ASP.NET Core MVC для проєктування і розробки застосунку, промодельовати сценарії використання вебзастосунку менеджерами і працівниками, створити структуру бази даних, для реалізації якої використано систему управління базами даних Microsoft SQL Server, визначити архітектуру вебзастосунку на основі розділення її на шар доступу до даних, шар бізнес-логіки та презентаційний шар.

З ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Опис логіки роботи програми

Логіка роботи програми, виходячи з описаних сценаріїв роботи користувачів, передбачає її реалізацію у двох режимах, що значно відрізняються один від одного. Ці два режими відрізняються роллю, яку виконує користувач. Будь-який користувач має бути співвіднесений з певною роллю, тому для роботи з програмою в будь-якому випадку користувач має авторизуватися. Неавторизований користувач не може працювати з програмою. В залежності від ролі користувача виконується вибір режиму роботи після авторизації.

Якщо користувач має роль менеджера (рис. 3.1), то йому одразу відображається перелік проєктів, з якими він може працювати. Першими доступними одразу стають інструменти роботи з проєктами, які можна застосувати до обраного з переліку проєкту. З проєктом можна виконати його редагування або відобразити дані проєкту.

Далі менеджер може скористатися одним з режимів подальшої роботи стосовно створення проєкту, пошуку завдань, перегляду статистики виконання завдань, реєстрації працівників. Виконавши будь-яку з цих функцій або в процесі виконання такої функції, менеджер може повернутися до вибору режиму роботи заново. Цей процес буде повторюватися до моменту завершення роботи з програмою. Таким моментом є вихід менеджера з облікового запису.

Якщо користувач є звичайним працівником, то для нього спочатку формується розклад завдань. Розклад формується на основі параметрів завдань проєктів, а також власних завдань працівника. Перелік завдань є основою роботи працівника.

Далі працівник може працювати над завданнями за обраним проєктом, визначати власні завдання (не прив'язані до проєктів), переглядати

статистику за завданнями, а після виконання будь-якої з цих дій обирати наступну з даного набору або переходити до формування розкладу завдань.



Рисунок 3.1 – Схема функціонування програми в режимі роботи з менеджером

Схема функціонування програми в режимі роботи з працівником представлена на рис. 3.2.



Рисунок 3.2 – Схема функціонування програми в режимі роботи з працівником

Окремо потрібно розглянути процес формування розкладу завдань (рис. 3.3). На порядок розташування завдань у розкладі впливає їх пріоритет та фактор впорядкованості між завданнями в проєкті, тобто якщо в множині

завдань, з якої формується розклад, наявні два завдання з одного проєкту, одне з якого є послідовників іншого, то завдання-послідовник переноситься в кінець множини.

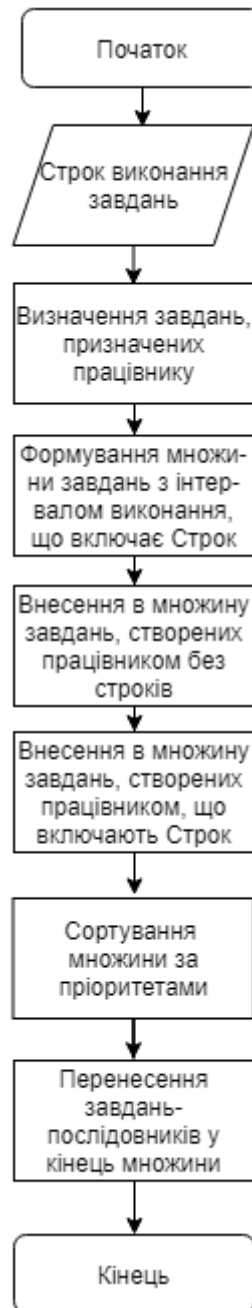


Рисунок 3.3 – Схема процесу формування розкладу завдань для працівника

До множини завдань включаються всі завдання, які були встановлені на виконання користувачу і при цьому відповідають встановленому інтервалу виконання, а також завдання, що не стосуються проєктів, створені

самим користувачем і при цьому такі, що за датами виконання попадають у заданий інтервал або не мають встановленої дати.

3.2 Опис логіки реалізації

У процесі реалізації програмного забезпечення було створено компоненти програмного забезпечення у відповідності з обраною архітектурою на основі фреймворку ASP.NET Core MVC.

Для реалізації шару роботи з даними створено класи моделей. Класи побудовано у відповідності зі структурою бази даних з додаванням необхідних засобів для організації зручного функціонування шару бізнес-логіки та представлення даних користувачу.

Клас Task окрім структури таблиці Task включає поля:

- string StartedText – текстове представлення дати і часу початку виконання завдання у відформатованому вигляді;
- string DeadlineText – текстове представлення дати і часу необхідного завершення завдання у відформатованому вигляді;
- string ActualText – текстове представлення дати і часу фактичного завершення виконання завдання у відформатованому вигляді;
- string PriorityText – текстове представлення пріоритету завдання, отримане шляхом перетворення з числового вигляду;
- string ModeText – текстове представлення режиму роботи з завданням, отримане шляхом перетворення з числового вигляду;
- string ApprovedText – текстове представлення підтвердження останнього стану завдання менеджером;
- string ColorCss – схема каскадної таблиці стилів, що використовується для відображення даного завдання (обирається з числового представлення кольору завдання);
- string ColorText – текстове представлення кольору завдання;

- `virtual User User` – визначає повні дані користувача, який є автором завдання, номер якого міститься в полі `UserId` таблиці `Task`;
- `virtual Project Project` – визначає повні дані проєкта, до якого належить завдання, номер проєкта міститься в полі `ProjectId`;
- `virtual ICollection<TaskUser> TaskUsers` – колекція користувачів, які працюють над даним завданням, що реалізує зв'язок один-до-багатьох;
- `virtual ICollection<TaskUpdate> TaskUpdates` – колекція оновлень завдань, що реалізує зв'язок один-до-багатьох;
- `virtual ICollection<SubTask> SubTasks` – колекція підзавдань завдання, що реалізує зв'язок один-до-багатьох.

Віртуальні поля використовуються для підвантаження повних даних відповідних пов'язаних об'єктів. Виконується дане підвантаження в контролері, наприклад, для підвантаження даних про автора, учасників проєкту та набір завдань виконується наступне звернення:

```

ViewData["Projects"] = bdata.Projects.Include(a => a.User).Include(a =>
a.ProjectParticipants).Include(a => a.Tasks).Where(a => a.User.login ==
User.Identity.Name);

```

Контролер `ProjectController` забезпечує роботу над проєктами і включає для цього наступні методи:

- `ProjectController (IBusinessService dbc)` – конструктор, визначає зв'язок з шаром бізнес-логіки;
- `Task<IActionResult> Index()` – метод заповнення представлення даними переліку проєктів;
- `Task<IActionResult> Details(int? id)` – метод заповнення даними проєкту з номером `id` представлення;
- `Task<IActionResult> Create([Bind("Id, Title, Description, Started, Deadline, UserId")] Project project)` – метод збереження результатів роботи з

представленням створення нового проєкту на основі прив'язування до полів форми;

- `Task<IActionResult> Edit(int id, [Bind("Id, Title, Description, Started, Deadline, UserId")] Project project)` – метод збереження результатів редагування проєкту у відповідному представленні;
- `bool ProjectExists(int id)` – метод перевірки існування проєкту з номером `id`;
- `Task<IActionResult> Duplicate([Bind("Id, Title, Description, Started, Deadline, UserId")] Project project, int projectid)` – метод створення нового проєкту і заповнення його шляхом дублювання структури проєкту з номером `projectid`, що реалізується через виклик відповідного методу шару бізнес-логіки.

Також було створено наступні контролери:

- контролер `ScheduleController` для роботи над складанням розкладу і його відображенням;
- контролер `StatisticsController` для роботи над статистичними даними;
- контролер `TaskController` для роботи з завданнями (створення, перегляд власних, проєктних, редагування, пошук, перегляд деталей, завершення);
- контролер `UserController` для роботи з обліковими записами.

Для реалізації представлень використано шаблон [14].

Усі представлення було згруповано за наступними папками:

- `project` – містить представлення для роботи з проєктами: `index.cshtml` для перегляду переліку проєктів, `create.cshtml` для створення нового проєкту, `edit.cshtml` для редагування проєкту, `details.cshtml` для перегляду повних даних проєкту, `duplicate.cshtml` для створення нового проєкту на основі існуючого в якості шаблону;

- `schedule` – містить представлення для роботи з розкладом завдань, включаючи `index.cshtml` для відображення звичайного розкладу працівнику та `local.cshtml` для відображення розкладу працівнику за місцем перебування;
- `shared` – містить спільні представлення, що використовуються у всіх інших, зокрема `layout.cshtml` створює меню, що відображається для навігації за сторінками, доступними відповідному користувачу;
- `statistics` – містить представлення перегляду статистичних показників, включаючи `index.cshtml` для перегляду статистики відхилень часу виконання за проєктом, `color.cshtml` для перегляду статистики за завданнями за кольорами, `undone.cshtml` для перегляду завдань з найменшим відсотком виконання;
- `task` – містить представлення для роботи над завданнями, включаючи зокрема `index.cshtml` для перегляду переліку власних завдань, `create.cshtml` для створення нового завдання або підзавдання, `edit.cshtml` для редагування завдання, `finish.cshtml` для завершення виконання завдання, `search.cshtml` для пошуку завдань, `details.cshtml` для перегляду повної інформації завдання, включаючи набір доступних підзавдань, якщо вони є;
- `user` – містить представлення для роботи з обліковими записами.

У представленні виконується заповнення розмітки вебсторінки даними, отриманими з контролера, наприклад, створення переліку проєктів для відображення на основній сторінці менеджера виконується наступним чином:

```
<div class="container">
  <div class="row">
    @foreach (var item in Model) {
      <div class="col-md-3 col-sm-6 text-center">
        <div class="wrap animated" data-animate="fadeInDown">
          <h3><a href="">@Html.DisplayFor(modelItem
item.Title)</a></h3>
=>
```

```

        <h5>@Html.DisplayFor(modelItem => item.ActualText) -
        @Html.DisplayFor(modelItem => item.DeadlineText)</h5>
        <p>@Html.DisplayFor(modelItem => item.Description)</p>
        <div class="col-md-12">
            <button class="btn btn-default submit">Змінити</button>
        </div>
    </div>
</div>
}
</div>
</div>

```

3.3 Висновки за розділом 3

Представлено послідовність функціонування програми в режимі роботи з менеджером та з працівником, для чого створено схеми функціонування, додатково описано схему формування розкладу завдань для працівника. Описано створені моделі, представлення і контролери програмного забезпечення.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Програмне забезпечення, що розробляється, призначене для керування завданнями, що виникають в процесі індивідуальної та командної роботи, щодо їх планування і обліку.

Програма надає можливість створення послідовностей завдань, підтримує роботу однієї компанії над завданнями, дозволяє працювати над завданнями у відрядженнях або в розподілено розташованих відділеннях компанії.

4.2 Умови виконання програми

Сервер має забезпечувати виконання наступних апаратних вимог:

- процесор з 2 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 4 Гб;
- 500 Мб доступного простору на жорсткому диску.

Обов'язковим є підключення до високошвидкісного інтернету (не менше ніж 100 Мбіт/с).

4.3 Тестування сценаріїв роботи з програмою

Тестування програми відбувалось шляхом розміщення вебзастосунку на вебсервері в локальній мережі.

Для роботи з програмою необхідно розмістити на вебсервері застосунок. Для кожної компанії, що працює з програмою потрібне окреме розміщення вебзастосунку.

Для роботи з програмою потрібно авторизуватися. Реєстрація облікових записів працівників виконується менеджером. Створити обліковий

запис менеджера може тільки системний адміністратор. Для цього необхідно звернутися з запитом до нього.

У випадку авторизації облікового запису менеджера відкривається стартова сторінка (основна сторінка), яка містить перелік проєктів компанії, менеджером яких є авторизований користувач, та меню (рис. 4.1).

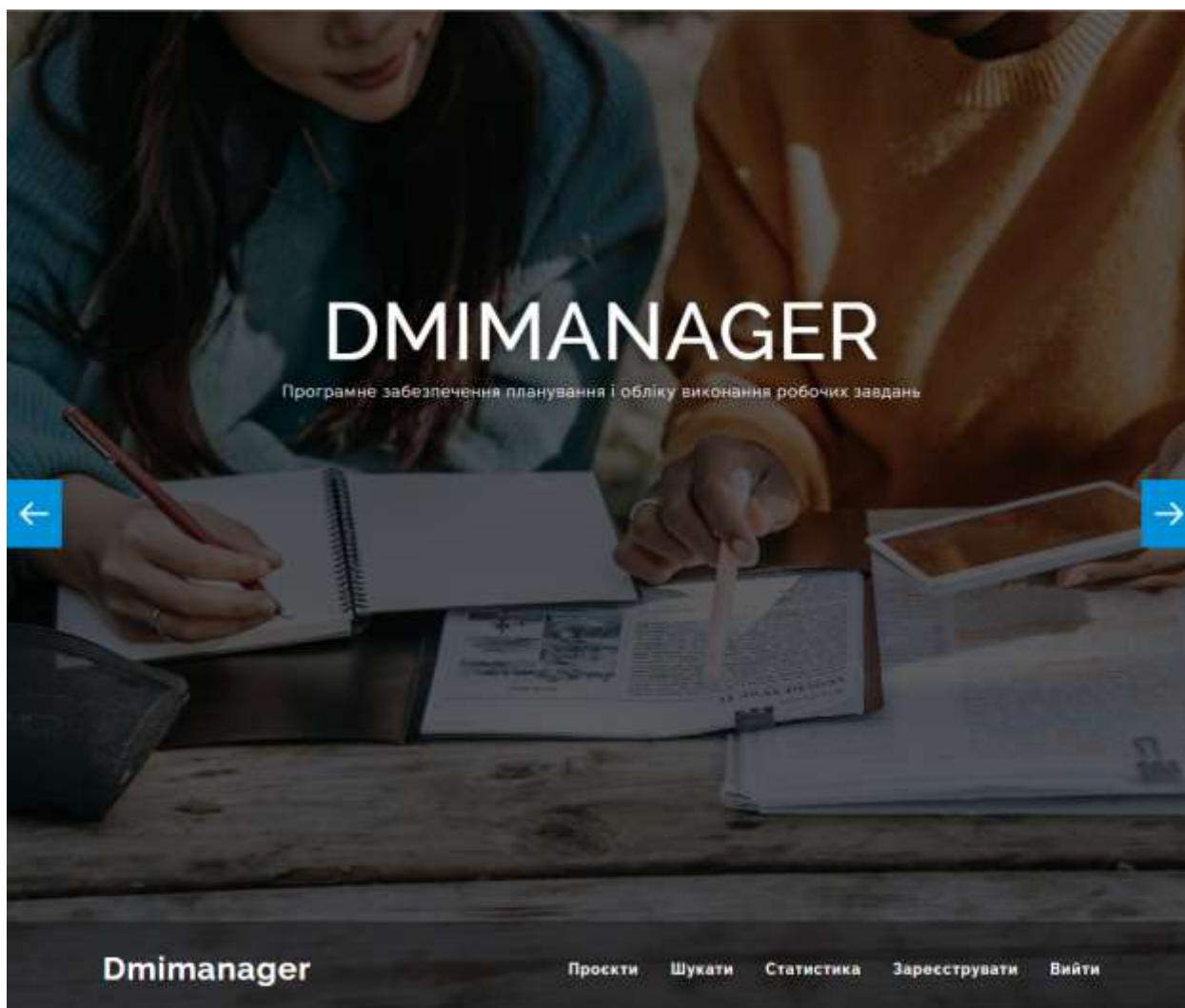


Рисунок 4.1 – Верхня частина стартової сторінки менеджера

Меню менеджера складається з пунктів:

- «Проекти», що представляє базову сторінку (рис. 4.2);
- «Шукати», що дозволяє виконувати пошук потрібних завдань, використовується для пришвидшення доступу для тих випадків, коли

менеджер потребує не переходу від завдань проєкту до підзавдань, а бажає знайти потрібне завдання у розгалуженій структурі проєкту;

- «Статистика», що дозволяє переглянути варіанти статистичних даних про роботу працівників над завданнями;
- «Зареєструвати», що дозволяє вносити облікові записи працівників;
- «Вийти».

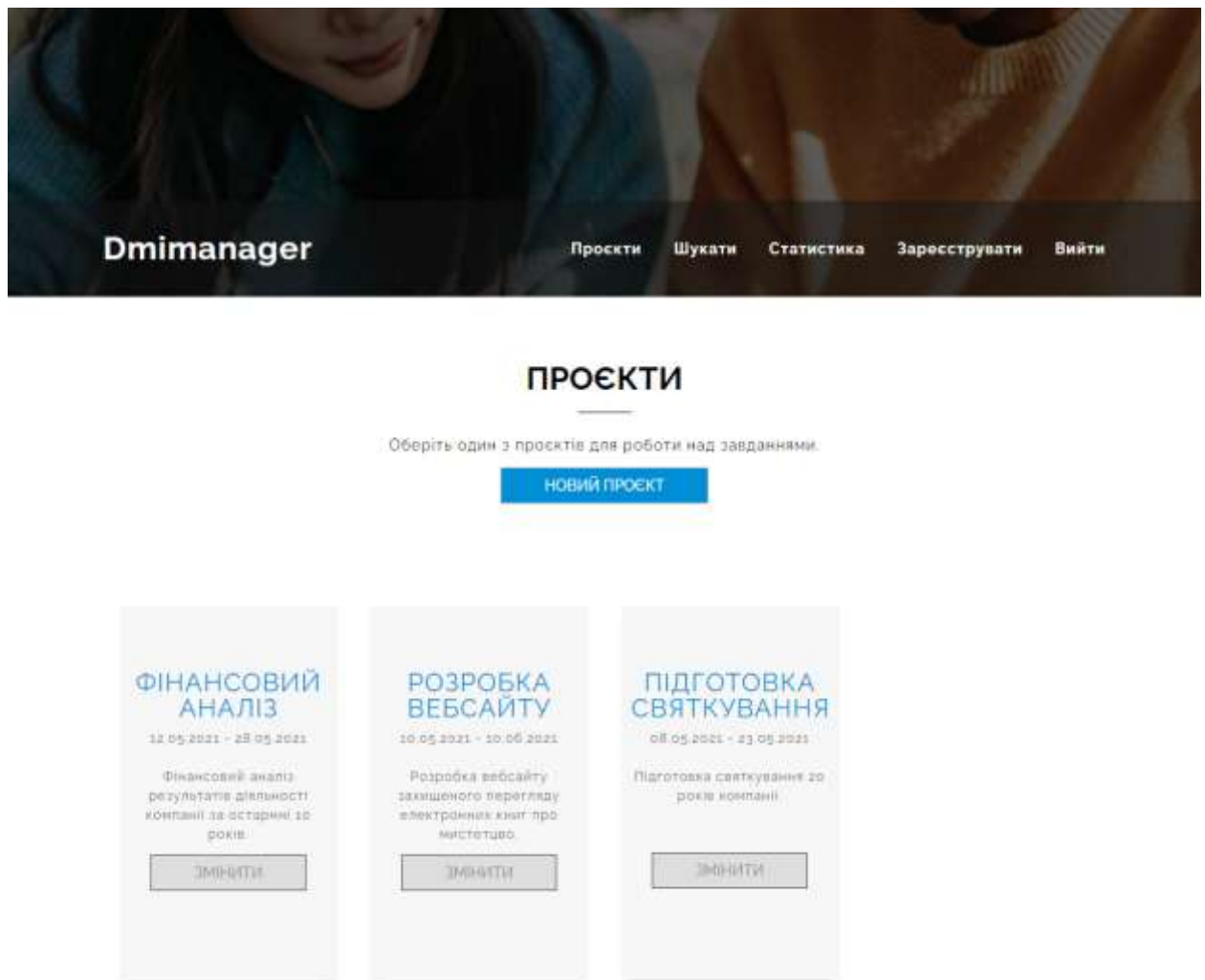


Рисунок 4.2 – Керування проєктами менеджером

На своїй базовій сторінці менеджер може обрати потрібний йому проєкт, натискаючи на його назву, після чого відбувається перехід на сторінку даного проєкту, де відображається повна інформація про нього,

набір завдань даного проєкту, за допомогою якого можна переходити до сторінок відповідних завдань з підзавданнями. Цей процес може бути будь-якого рівня вкладеності.

Сторінка перегляду детальної інформації про завдання містить його опис, дату і час початку роботи над ним, відсоток виконання, очікувану дату виконання, фактичну дату виконання або фактичну на даний момент затримку виконання завдання, факт підтвердження останньої зміни стану менеджером. Менеджер має підтверджувати внесені зміни працівником. Для кожного проєкту відображається також набір підзавдань. Окрім того для завдання відображається історія зміни його станів.

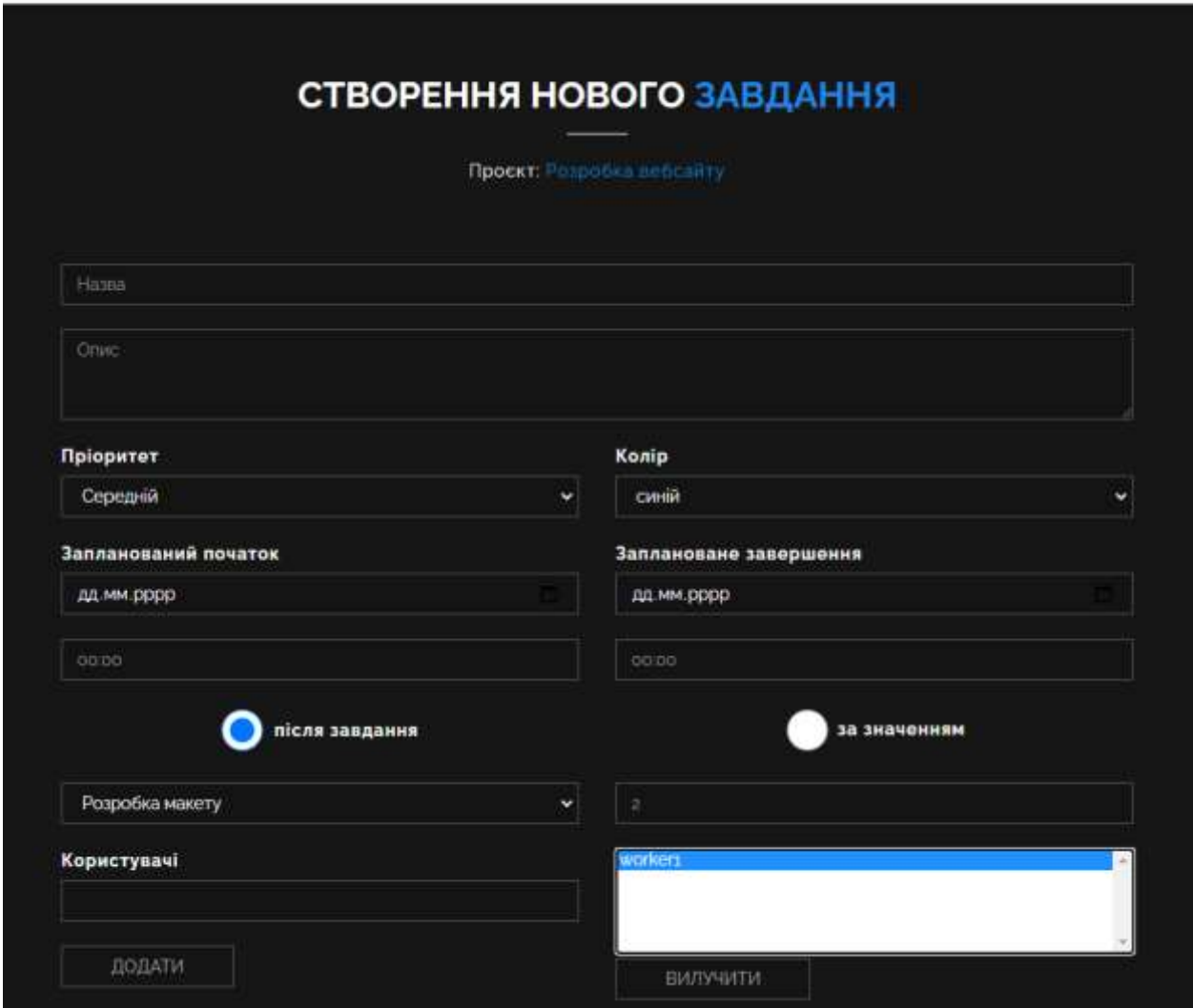
Окрім того менеджер може натиснути на кнопку «Змінити» на картці проєкту, що дозволить перейти до змінювання інформації про проєкт, а також вносити будь-які зміни в проєкт. Ці зміни охоплюють і перелік учасників, і завдання.

Функція створення завдання може використовуватися працівником для створення власних завдань, але в основному застосовується менеджером (рис. 4.3).

На сторінці створення завдання менеджер має задати:

- назву завдання;
- опис завдання;
- пріоритет завдання, що вибирається з переліку;
- колір завдання, який потім може змінити за потреби працівник;
- заплановану дату і час початку виконання завдання;
- заплановану дату і час завершення виконання завдання;
- порядок слідування завдання: використовується для проєктних завдань, можна обрати за допомогою радіокнопки варіант визначення порядку з завданням конкретної цифри або варіант з вибором завдання даного рівня в проєкті, що передує виконанню даного;
- перелік користувачів, які мають виконувати завдання;

- місце виконання завдання за потреби (потрібно встановити відповідний прапорець, обрати на мапі положення та задати радіус дії);
- режим роботи: чи буде завдання видно всім іншим учасникам команди, чим тільки працівнику, якому воно призначене.



Dmimanager Проєкти Шукати Статистика Зареєструвати Вийти

СТВОРЕННЯ НОВОГО ЗАВДАННЯ

Проєкт: Розробка вебсайту

Назва

Опис

Пріоритет **Колір**

Середній синій

Запланований початок **Заплановане завершення**

дд.мм.рррр дд.мм.рррр

00:00 00:00

☒ після завдання ☐ за значенням

Розробка макету 2

Користувачі

worker

ДОДАТИ ВИЛУЧИТИ

Рисунок 4.3 – Створення нового завдання

Менеджер може переглядати статистику виконання проєктів або узагальнено завдань. Спосіб відображення даних аналогічний тому, що буде нижче розглянуто для працівника. При цьому менеджеру не доступна кольорова схема розмітки. Цей варіант використовується саме працівником.

До меню працівника входять наступні розділи:

- «Розклад», що дозволяє сформувати розклад для роботи над завданнями;
- «Проекти», що дозволяє переглядати завдання за проектами;
- «Завдання», що дозволяє працювати з завданнями, які створив працівник, а також створювати такі завдання;
- «Шукати», що дозволяє виконувати пошук завдань за їх назвою;
- «Статистика», що дозволяє працювати з узагальненими власними даними щодо роботи над завданнями.

Розклад для користувача формується на основі логіки, яку було представлено у розділі 3.

Існують два варіанти розкладу.

Перший варіант (рис. 4.4) формує перелік з усіх завдань, призначених працівнику на заданий інтервал дат.

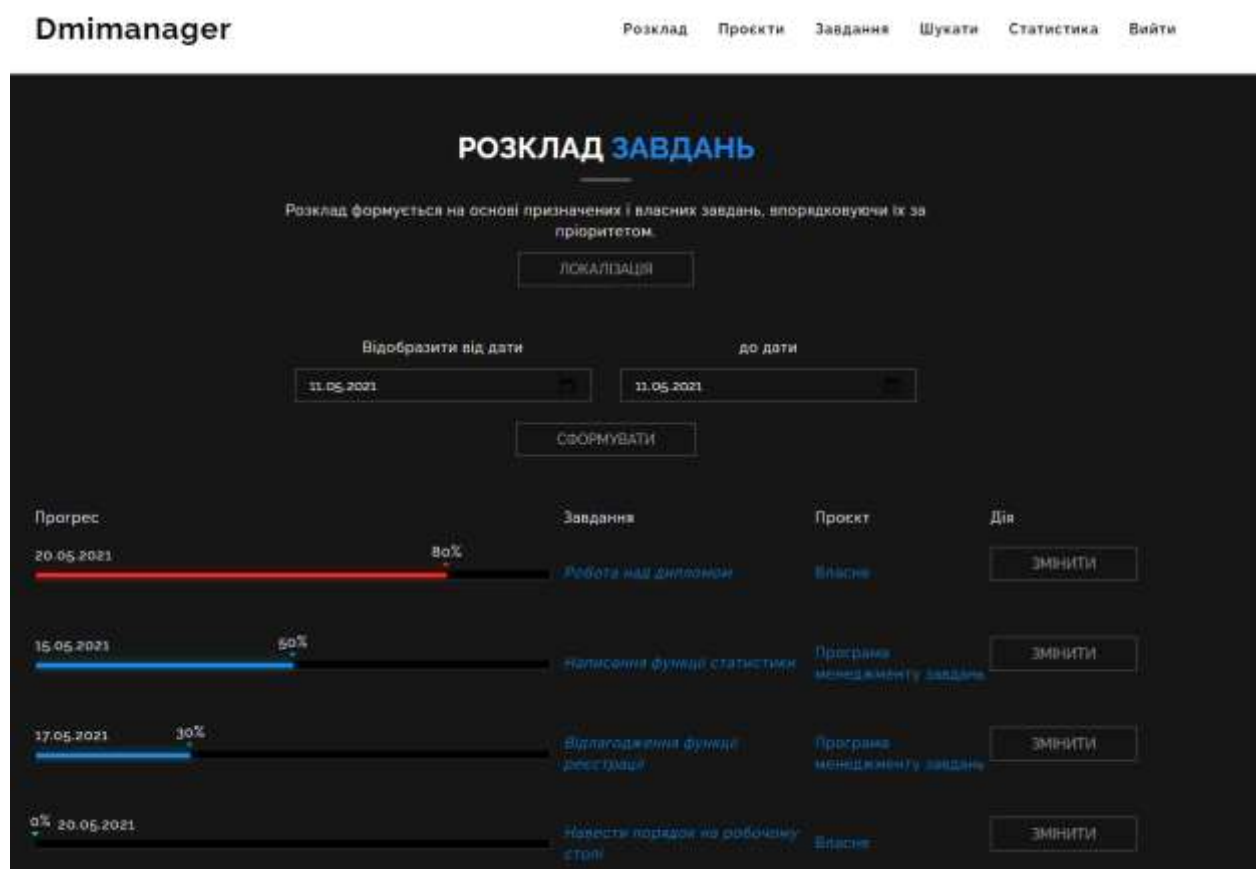


Рисунок 4.4 – Розклад виконання завдань для працівника

Для цього працівник має задати інтервал дати, коли буде реалізовуватися завдання, використовуючи у відповідних полях календарі, а потім натиснути на кнопку «Сформувати». За замовчуванням відображається поточна дата, тобто розклад формується на сьогодні. Тоді в таблиці буде відображено завдання, які відповідають даним критеріям.

Кожне завдання відображається своєю назвою (натискаючи на неї, можна перейти до перегляду повної інформації про завдання на окремій сторінці), назвою проєкту, до якого воно належить (натисканням можна перейти до завдань відповідного проєкту, а завдання, створені самим працівником відображаються як «Власне»), часовою шкалою, на якій відображається відсоток виконання завдання, та датою, до якої має бути завершено виконання завдання (датою і часом, якщо час не дорівнює 00:00). Якщо відсоток виконання завдання менше 20 %, то спочатку відображається відсоток на шкалі, а правіше – дата, до якої має бути завершено проєкт.

Для того, щоб визначити виконання завдання або встановити відсоток, потрібно натиснути на кнопку «Змінити», яка дозволить відкрити сторінку завдання і вказати на ній точний відсоток виконання завдання, відзначити початок роботи над завданням або відзначити завершення завдання. Після того, як працівник встановив відсоток виконання завдання, даний запис формується в історії оновлень завдання, а менеджер має підтвердити такий стан справ. Після підтвердження в завданні відзначається, що воно підтверджене. На сторінках повної інформації про завдання менеджеру і працівнику виводиться інформація про те, чи підтверджено даний стан справ менеджером.

Другий варіант (рис. 4.5) розкладу формує перелік завдань тільки з тих, які відповідають мітці геолокації, де він знаходиться в даний момент. Принцип визначення порядку завдань той самий, що і в попередньому варіанті, але при цьому множина завдань включає тільки ті, коло виконання яких (формується визначеним центром і радіусом, що зберігаються в даних завдання) включає поточну точку, де розташовується працівник.

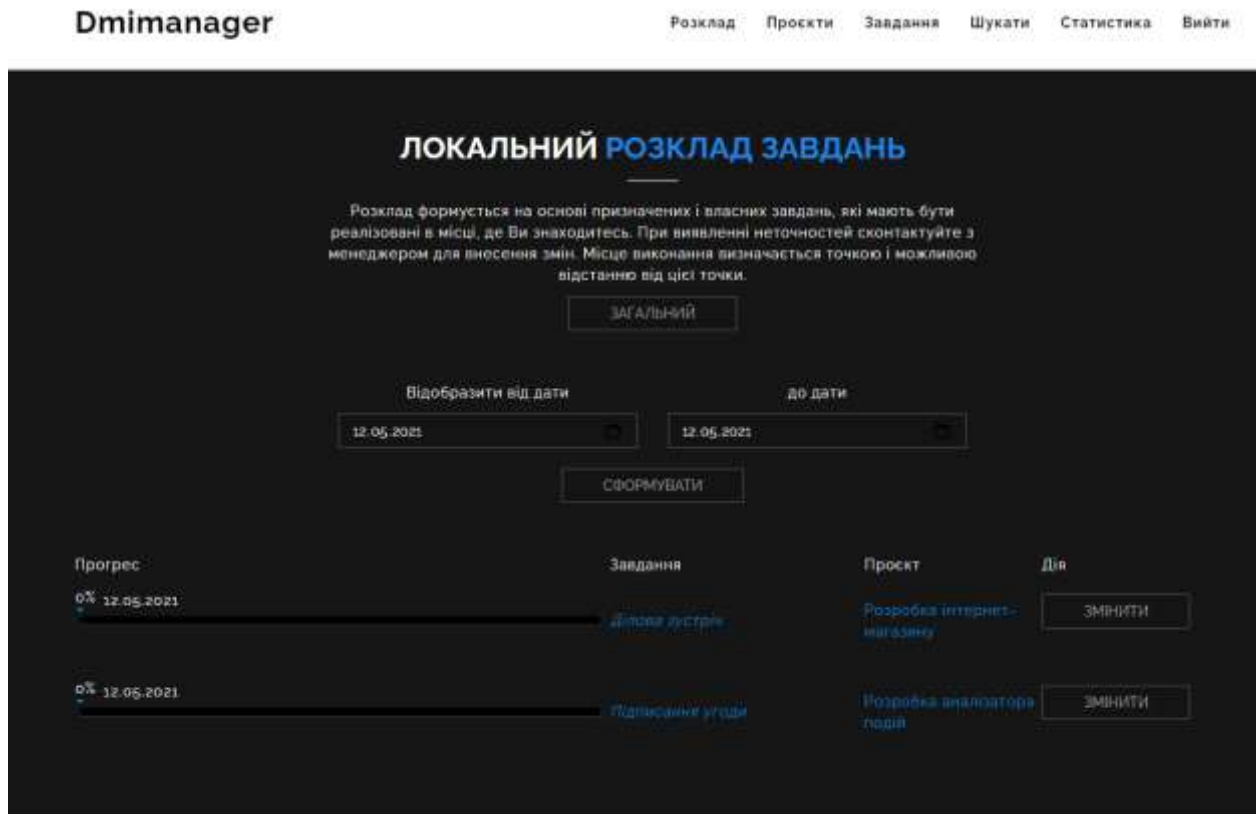


Рисунок 4.5 – Розклад виконання завдань для працівника за місцем розташування

Дана функціональність потрібна для роботи у відрядженнях або якщо в компанії є декілька підрозділів. Також таким чином можна підтримувати роботу працівників, що реалізується в динаміці, наприклад, кур'єрів.

Для завдання у такому випадку обов'язково має бути встановлено мітку виконання. Якщо таку мітку не встановлено, то воно в будь-якому випадку не буде відображено в даному переліку, потрібно буде повернутися до загального режиму, для чого призначено кнопку «Загальний», або зв'язатися з менеджером для внесення відповідних змін.

Пункт меню «Статистика» дозволяє отримати узагальнені дані про роботу працівника над завданнями. Працівнику за допомогою кнопок доступні наступні варіанти статистики:

- «Невиконані» для перегляду завдань за заданий проміжок часу, які мають найменший відсоток завершення;

- «Власні» для перегляду даних про затримку виконання завдань, які створив сам працівник;
- «За проєктом» для перегляду даних про затримку виконання завдань, що стосуються обраного з переліку проєкту;
- «Затримані» для перегляду завдань (з усіх завдань, з якими може працювати працівник), які було затримано на найбільший відсоток часу порівняно з запланованим завершенням;
- «Кольори» (рис. 4.6).

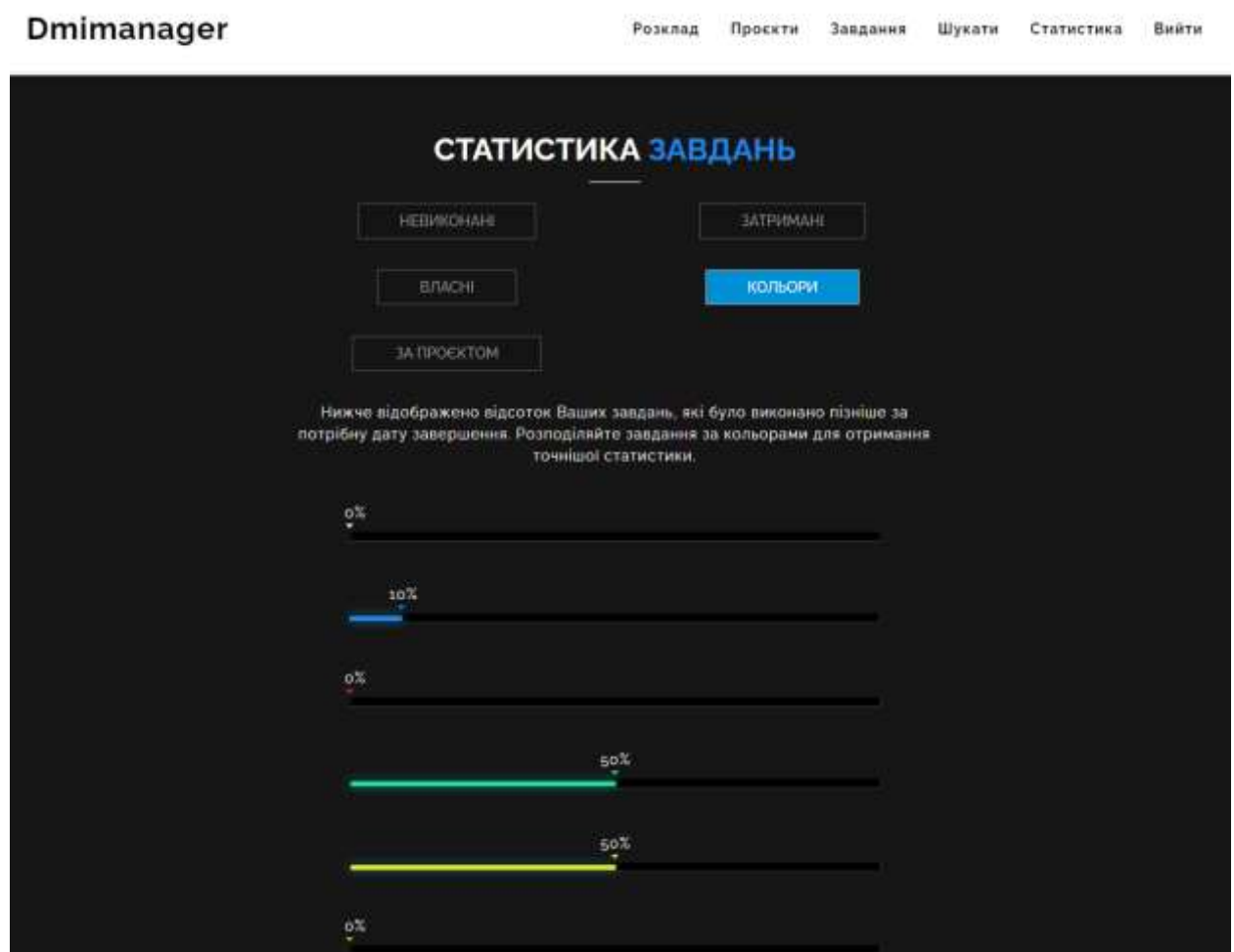


Рисунок 4.6 – Перегляд статистики виконання працівником завдань з затримкою

Обраний у даний момент варіант підсвічується синім кольором.

За кожним кольором завдань відображається відсоток проєктів, яким

працівник встановив у відповідність даний колір, які було виконано пізніше ніж запланований час завершення завдання, серед усіх завдань, над якими за весь час працював працівник.

Проведене тестування продемонструвало, що всі заплановані до реалізації функції програми, були успішно реалізовані.

4.4 Висновки за розділом 4

Сценарії роботи менеджера та працівника з програмою детально описані, представлені відповідні інтерфейси програми. Описано реалізовані функціональні можливості та те, як користувач може застосувати їх для виконання своїх завдань.

ВИСНОВКИ

Проблема розроблення програмного забезпечення планування і обліку виконання робочих завдань розглядається в роботі. Виконаний аналіз проблеми дозволив виділити основні принципи організації програмного забезпечення планування і обліку виконання робочих завдань, проаналізувати і порівняти програмні аналоги, на основі чого виділити особливості програмного забезпечення, що розроблялось.

Виконано проектування програми, результати якого дозволили забезпечити вибір мови програмування C# і фреймворку розробки вебзастосунків ASP.NET Core MVC для проектування і розробки застосунку, промодельовати сценарії використання вебзастосунку менеджерами і працівниками у вигляді діаграм прецедентів, створити структуру бази даних, для реалізації якої використано систему управління базами даних Microsoft SQL Server, визначити архітектуру вебзастосунку на основі розділення її на шар доступу до даних, шар бізнес-логіки та презентаційний шар.

Представлено послідовність функціонування програми в режимі роботи з менеджером та з працівником, для чого створено схеми функціонування, додатково описано схему формування розкладу завдань для працівника. Описано створені моделі, представлення і контролери програмного забезпечення.

Програмне забезпечення призначене для керування завданнями, що виникають в процесі індивідуальної та командної роботи, щодо їх планування і обліку. Програма надає можливість створення послідовностей завдань, підтримує роботу однієї компанії над завданнями, дозволяє працювати над завданнями у відрядженнях або в розподілено розташованих відділеннях компанії.

Усі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 16 Features That Make a Great Free Task Management Software [Electronic resource]. – Access mode : <https://www.ntaskmanager.com/blog/task-management-tool-features/>
2. Mowat, A. 12 Features Every Task Management Tool Should Have [Electronic resource] / A. Mowat. – Access mode : <https://www.prialto.com/blog/12-features-team-task-management-tool>
3. Manage your team's work, projects, & tasks online | Asana [Electronic resource]. – Access mode : <https://asana.com/>
4. Duffy, J. Asana Review | PCMag [Electronic resource]. – Access mode : <https://www.pcmag.com/reviews/asana>
5. Chappell, E. The 17 Best Task Management Software in 2021 (Free And Paid!) [Electronic resource] / E. Chappell. – Access mode : <https://clickup.com/blog/task-management-software/>
6. Todoist [Electronic resource] : The to do list to organize work & life. – Access mode : <https://todoist.com/>
7. Duffy, J. Todoist Review | PCMag [Electronic resource]. – Access mode : <https://www.pcmag.com/reviews/todoist>
8. Pot, J. The 9 best to do list apps of 2021 [Electronic resource] / J. Pot. – Access mode : <https://zapier.com/blog/best-todo-list-apps/>
9. Fear, N. Best task management apps of 2021 [Electronic resource] : organize and manage your workload / N. Fearn, B. Turner. – Access mode : <https://www.techradar.com/best/best-task-management-apps>
10. Фленов, М. Библия С# [Текст] / М. Фленов. – СПб. : БХВ-Петербург, 2020. – 512 с.
11. Введение в ASP.NET Core | Microsoft Docs [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0>
12. Benefits of ASP.NET Core to Develop robust Web Applications

[Electronic resource]. – Access mode : <https://www.cisin.com/coffee-break/technology/benefits-of-asp-net-core-to-develop-robust-web-applications.html>

13. Обзор Entity Framework Core – EF Core | Microsoft Docs [Электрон. ресурс]. – Режим доступа : <https://docs.microsoft.com/ru-ru/ef/core/>

14. Free CSS, Free CSS Templates, CSS Website Templates, Templates, Free Templates, CSS, Web Site Templates, Free CSS Menus [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page264/daraz>

ДОДАТОК А
Технічне завдання

Вступ

Особливостями програми мають стати можливість створення послідовностей завдань як у межах проєкту, так і у межах індивідуальних приватних завдань. Програма має обробляти робочі завдання щодо їх планування і обліку. Щодо обліку передбачається зміна станів завдання. Програма має підтримувати роботу однієї компанії над завданнями.

A.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Програмне забезпечення планування і обліку виконання робочих завдань» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення, що розробляється, призначене для керування завданнями, що виникають в процесі індивідуальної та командної роботи, щодо їх планування і обліку.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має надавати такі функціональні можливості:

- створення проєктів;
- визначення учасників проєкту;
- визначення робочого завдання;
- визначення стану робочих завдань;

- визначення строків виконання робочого завдання;
- визначення відсотку виконання робочого завдання;
- визначення кольору завдання;
- визначення місця виконання завдання;
- встановлення пріоритету робочому завданню;
- визначення підзавдань до завдання;
- редагування робочого завдання;
- вилучення робочих завдань;
- визначення завершення завдання;
- перегляд стану виконання проєкту;
- формування розкладу завдань для користувача;
- перегляд всіх власних завдань;
- перегляд власних завдань з найменшим відсотком виконання;
- перегляд завдань, не прикріплених до проєктів;
- формування розкладу завдань за місцем виконання;
- перегляд підзавдань завдання;
- пошук завдань;
- перегляд завдань користувача за проєктом;
- перегляд статистики виконання завдань за працівником;
- перегляд статистики виконання завдань за кольором;
- перегляд завдань за часовим проміжком;
- перегляд завдань з найбільшим відхиленням часу виконання;
- перегляд історії роботи над завданням;
- створення проєкту за шаблоном;
- авторизація користувача;
- реєстрація облікового запису працівника менеджером.

А.3.2 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп'ютером не висувається.

Для забезпечення роботи бази даних на сервері необхідні вміння системного адміністратора для підтримки роботи вебсервера, сервера системи керування базами даних Microsoft SQL Server.

А.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби серверної сторони повинні включати щонайменше:

- процесор з 2 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 4 Гб;
- 500 Мб доступного простору на жорсткому диску, адже зокрема на сервер мають завантажуватися фотографії моделей автомобілів.

А.4 Порядок контролю та приймання

Принципи контролю виконання роботи визначаються календарним планом, розташованим у завданні на дипломну кваліфікаційну роботу, де вказано назви етапів, строки їх виконання та результат.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Task.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class Task
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public int Priority { get; set; }
        public int ProjectId { get; set; }
        public int UserId { get; set; }
        public int Percent { get; set; }
        public int Color { get; set; }
        public int Orderstart { get; set; }
        public int Placelat { get; set; }
        public int Placelong { get; set; }
        public float Around { get; set; }
        public DateTime Started { get; set; }
        public DateTime Deadline { get; set; }
        public int Order { get; set; }
        public bool Mode { get; set; }
        public DateTime Actual { get; set; }
        public bool Approved { get; set; }

        [NotMapped]
        public string StartedText
        {
            get
            {
                bool flag = this.Started == null;
                if (flag)
                {

```

```

        return "не визначено";
    }
    else
    {
        return
((DateTime)this.Started).ToString("dd.MM.yyyy hh:mm");
    }
}

[NotMapped]
public string DeadlineText
{
    get
    {
        bool flag = this.Deadline== null;
        if (flag)
        {
            return "не визначено";
        }
        else
        {
            return
((DateTime)this.Deadline).ToString("dd.MM.yyyy hh:mm");
        }
    }
}

[NotMapped]
public string ActualText
{
    get
    {
        bool flag = this.Actual == null;
        if (flag)
        {
            return "не визначено";
        }
        else
        {
            return
((DateTime)this.Actual).ToString("dd.MM.yyyy hh:mm");
        }
    }
}

```

```

    }
}

[NotMapped]
public string PriorityText
{
    get
    {
        if (this.Priority == 0)
        {
            return "незначний";
        }
        else if (this.Priority == 1)
        {
            return "низький";
        }
        else if (this.Priority == 2)
        {
            return "помірний";
        }
        else if (this.Priority == 3)
        {
            return "високий";
        }
        else if (this.Priority == 4)
        {
            return "критичний";
        }
    }
}

```

```

[NotMapped]
public string ModeText
{
    get
    {
        if (this.Mode)
        {
            return "приватний";
        }
        else return "командний";
    }
}

```

```
[NotMapped]
public string ApprovedText
{
    get
    {
        if (this.Mode)
        {
            return "підтверджений";
        }
        else return "непідтверджений";
    }
}
```

```
[NotMapped]
public string ColorCss
{
    get
    {
        if (this.Color == 0)
        {
            return "line-black";
        }
        else if (this.Color == 1)
        {
            return "line-blue";
        }
        else if (this.Color == 2)
        {
            return "line-red";
        }
        else if (this.Color == 3)
        {
            return "line-green";
        }
        else if (this.Color == 4)
        {
            return "line-yellow";
        }
        else if (this.Color == 4)
        {
            return "line-orange";
        }
    }
}
```

```

}

[NotMapped]
public string ColorText
{
    get
    {
        if (this.Color == 0)
        {
            return "білий";
        }
        else if (this.Color == 1)
        {
            return "синій";
        }
        else if (this.Color == 2)
        {
            return "червоний";
        }
        else if (this.Color == 3)
        {
            return "зелений";
        }
        else if (this.Color == 4)
        {
            return "жовтий";
        }
        else if (this.Color == 5)
        {
            return "помаранчевий";
        }
    }
}

public virtual User User
{ get; set; }
public virtual Project Project
{ get; set; }
public virtual ICollection<TaskUser> TaskUsers
{ get; set; }
public virtual ICollection<TaskUpdate> TaskUpdates
{ get; set; }
public virtual ICollection<SubTask> SubTasks

```

```

        { get; set; }
    }
}

```

Б.2 Текст файла User.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class User
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Position { get; set; }
        public string Phone { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }
        public int role { get; set; }

        [NotMapped]
        public string RoleText
        {
            get
            {
                if (this.Role == 0)
                {
                    return "працівник";
                }
                else if (this.Role == 1)
                {
                    return "менеджер";
                }
            }
        }
    }
}

```

```

    }
}
}

```

Б.3 Текст файла Project.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class Project
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public DateTime Started { get; set; }
        public DateTime Deadline { get; set; }
        public int UserId { get; set; }
        public DateTime Actual { get; set; }

        [NotMapped]
        public string StartedText
        {
            get
            {
                bool flag = this.Started == null;
                if (flag)
                {
                    return "не визначено";
                }
                else
                {
                    return
((DateTime)this.Started).ToString("dd.MM.yyyy hh:mm");
                }
            }
        }
    }
}

```

```

    }
}

[NotMapped]
public string DeadlineText
{
    get
    {
        bool flag = this.Deadline == null;
        if (flag)
        {
            return "не визначено";
        }
        else
        {
            return
((DateTime)this.Deadline).ToString("dd.MM.yyyy hh:mm");
        }
    }
}

[NotMapped]
public string ActualText
{
    get
    {
        bool flag = this.Actual == null;
        if (flag)
        {
            return "не визначено";
        }
        else
        {
            return
((DateTime)this.Actual).ToString("dd.MM.yyyy");
        }
    }
}

public virtual User User
{ get; set; }
public virtual ICollection<User> ProjectParticipants
{ get; set; }

```

```

        public virtual ICollection<Task> Tasks
        { get; set; }
    }
}

```

Б.4 Текст файла TaskUpdate.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class TaskUpdate
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public int TaskId { get; set; }
        public int UserId { get; set; }
        public int Percent { get; set; }
        public DateTime Updated { get; set; }

        [NotMapped]
        public string UpdatedText
        {
            get
            {
                bool flag = this.Updated == null;
                if (flag)
                {
                    return "не визначено";
                }
                else
                {
                    return
((DateTime)this.Updated).ToString("dd.MM.yyyy hh:mm");
                }
            }
        }
    }
}

```

```

    }

    public virtual User User
    { get; set; }
    public virtual Task Task
    { get; set; }
}
}

```

Б.5 Текст файла TaskUser.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class TaskUser
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public int TaskId { get; set; }
        public int UserId { get; set; }
        public string Description { get; set; }
        public int TaskPercent { get; set; }
        public int Percent { get; set; }
        public virtual User User
        { get; set; }
        public virtual Task Task
        { get; set; }
    }
}

```

Б.6 Текст файла SubTask.cs

```

using System;

```

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace DmiManager.Models
{
    public class SubTask
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public int TaskParentId { get; set; }
        public int ChildId { get; set; }
        public int TaskPercent { get; set; }
        public int Order { get; set; }
        public virtual Task Child
        { get; set; }
        public virtual Task TaskParent
        { get; set; }
    }
}

```

Б.7 Текст файла ProjectController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using DmiManager.Business;
using DmiManager.Models;

namespace DmiManager.Controllers
{
    public class ProjectController: Controller
    {

```

```

private readonly IBusinessService bdata;

public ProjectController (IBusinessService dbc)
{
    bdata = dbc;
}

[Authorize]
public async Task<IActionResult> Index()
{
    ViewData["Projects"] = bdata.Projects.Include(a =>
a.User).Include(a => a.ProjectParticipants).Include(a =>
a.Tasks).Where(a => a.User.login == User.Identity.Name);
    return View();
}

[Authorize]
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var project = await bdata.Projects.Include(a =>
a.User).Include(a => a.ProjectParticipants).Include(a =>
a.Tasks).FirstOrDefaultAsync(b => b.Id == id);
    if (project == null)
    {
        return NotFound();
    }
    return View(project);
}

[Authorize]
public IActionResult Create()
{
    ViewData["Users"] = new SelectList(bdata.Users,
    "Id", "Login");
    ViewData["Projects"] = new
    SelectList(bdata.Projects, "Id", "Title");
    return View();
}

```

```

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,
Title, Description, Started, Deadline, UserId")] Project project)
        {
            if (ModelState.IsValid)
            {
                bdata.Projects.InsertOnSubmit(project);
                await bdata.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["User"] = new SelectList(bdata.Users, "Id",
            "Id", project.UserId);
            return View(project);
        }

        [Authorize]
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var project = await bdata.Projects.FindAsync(id);
            if (project == null)
            {
                return NotFound();
            }

            ViewData["Users"] = new SelectList(bdata.Users,
            "Id", "Login");
            return View(project);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,
Title, Description, Started, Deadline, UserId")] Project project)
        {
            if (id != project.Id)
            {
                return NotFound();
            }

```

```

        if (ModelState.IsValid)
        {
            try
            {
                bdata.Update(project);
                await bdata.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ProjectExists(project.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["User"] = new SelectList(bdata.Users, "Id",
        "Id", project.UserId);
        return View(automodel);
    }

    private bool ProjectExists(int id)
    {
        return bdata.Projects.Any(a => a.Id == id);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Duplicate([Bind("Id,
    Title, Description, Started, Deadline, UserId")] Project project, int
    projectid)
    {
        if (ModelState.IsValid)
        {
            bdata.Projects.InsertOnSubmit(project);
            await bdata.SaveChangesAsync();
            bdata.Duplicate(project.id, projectid);
            return RedirectToAction(nameof(Index));
        }
    }

```

```
        ViewData["User"] = new SelectList(bdata.Users, "Id",  
"Id", project.UserId);  
        return View(project);  
    }  
}  
}
```

ДОДАТОК В
Слайди презентації

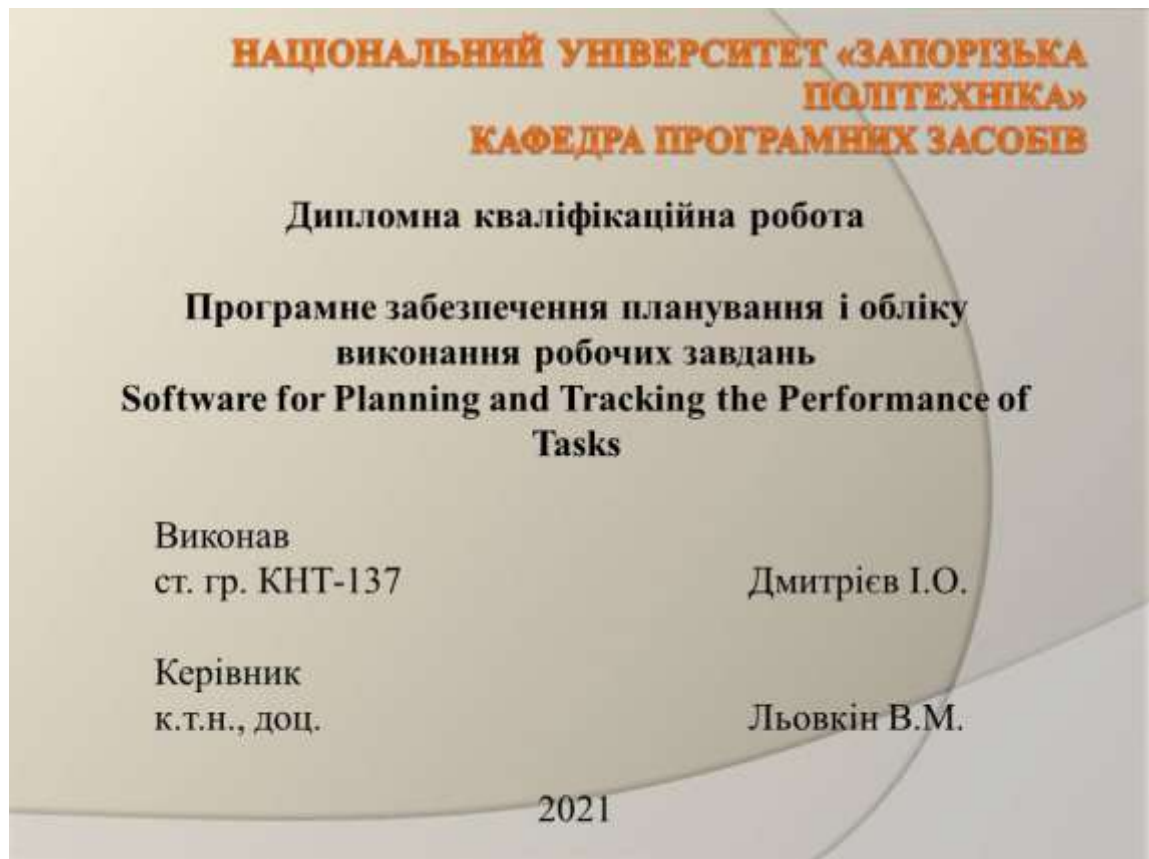


Рисунок В.1 – Слайд № 1

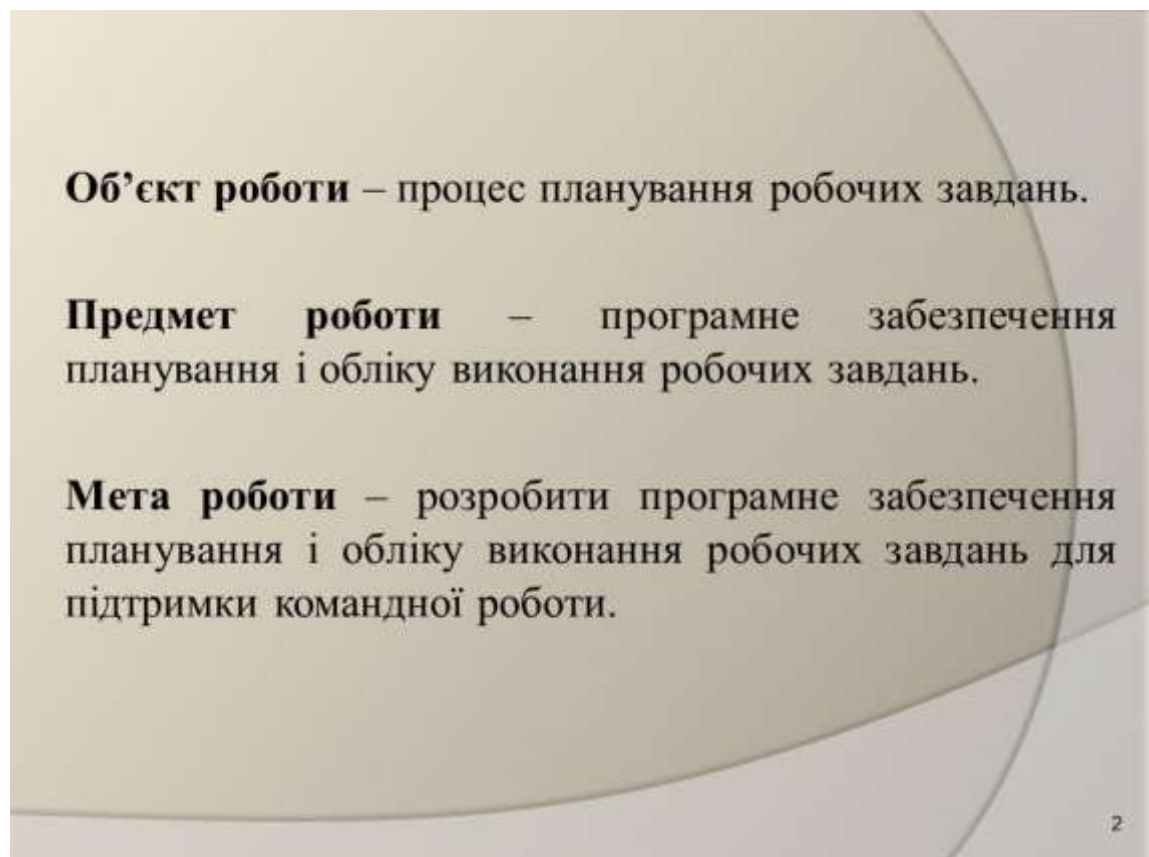


Рисунок В.2 – Слайд № 2

Завдання дипломної кваліфікаційної роботи

- виконати аналіз предметної області;
- виділити вимоги до програмного забезпечення;
- виконати проєктування програмного забезпечення;
- виконати реалізацію програмного забезпечення.

3

Рисунок В.3 – Слайд № 3

Порівняння програмних аналогів

	Asana	Todoist	DmiManager (власна розробка)
Планування завдань	+	+	+
Створення власних шаблонів проєктів	+	–	+
Розділення відповідальності між учасниками команди	+	–	+
Планування на основі локацій	–	+	+
Безкоштовність	–	–	+

4

Рисунок В.4 – Слайд № 4

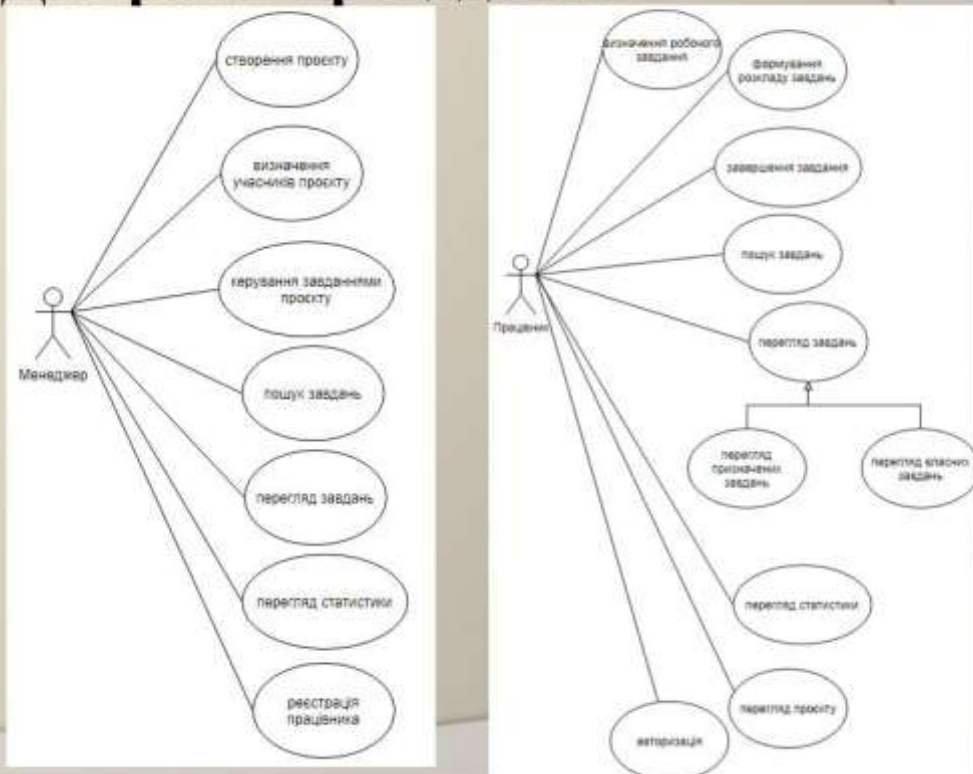
Вибір мови програмування

	C#	Python
Продуктивність програми	Більше	Менше
Створення вебзастосунків	Так	Так
Потенціал розширення	Більше (можна створити застосунки для iOS, Android)	Менше

5

Рисунок В.5 – Слайд № 5

Діаграми прецедентів



6

Рисунок В.6 – Слайд № 6

Схема бази даних



Рисунок В.7 – Слайд № 7

Архітектура програмного забезпечення



Рисунок В.8 – Слайд № 8

Схеми функціонування програми в режимах роботи з менеджером та працівником

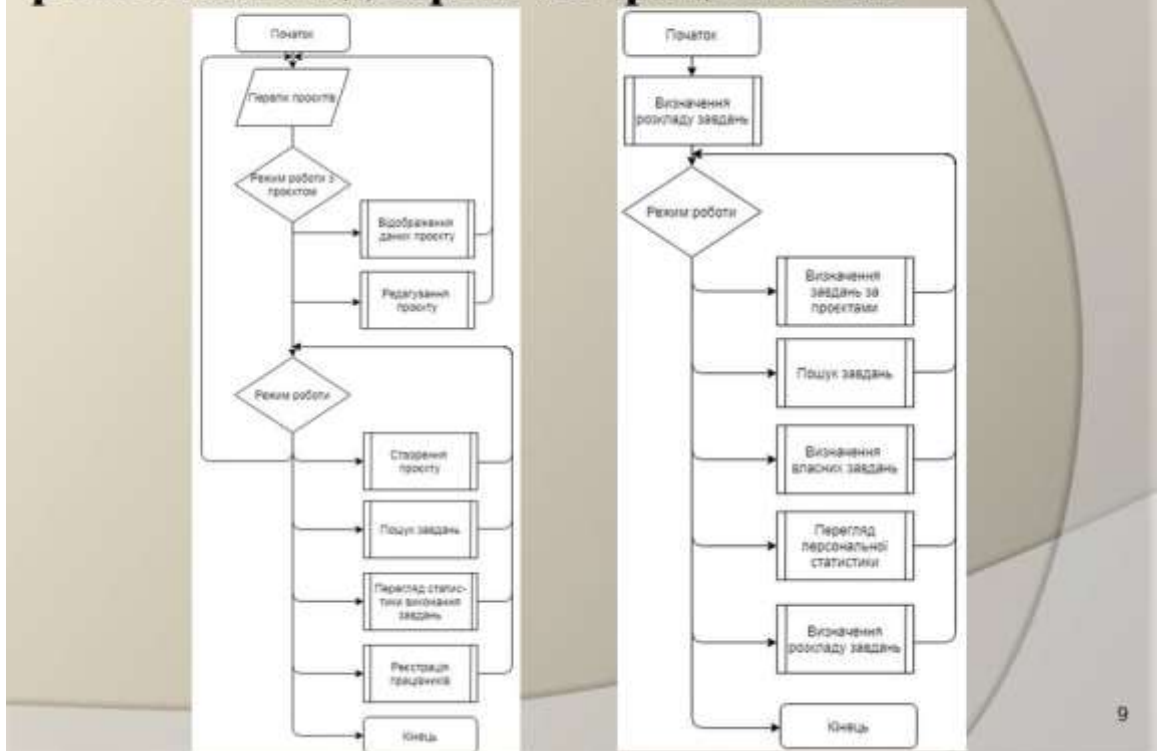


Рисунок В.9 – Слайд № 9

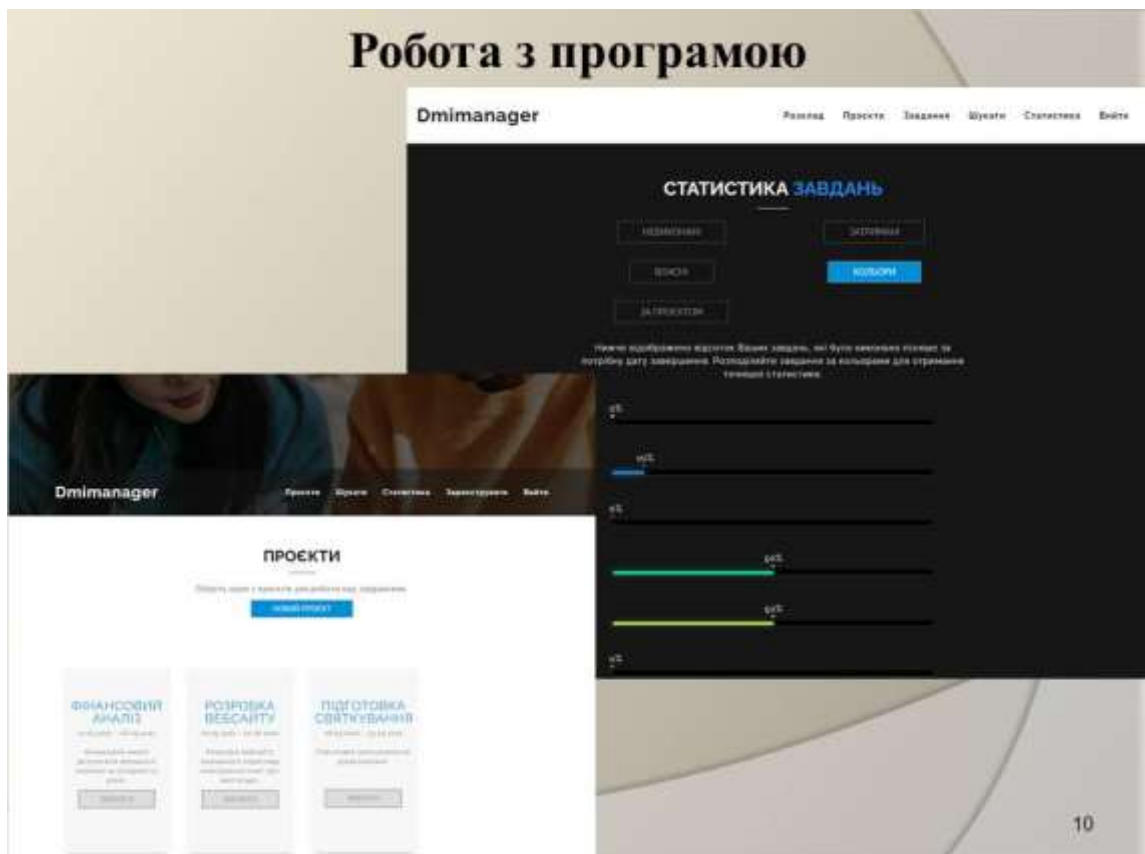


Рисунок В.10 – Слайд № 10

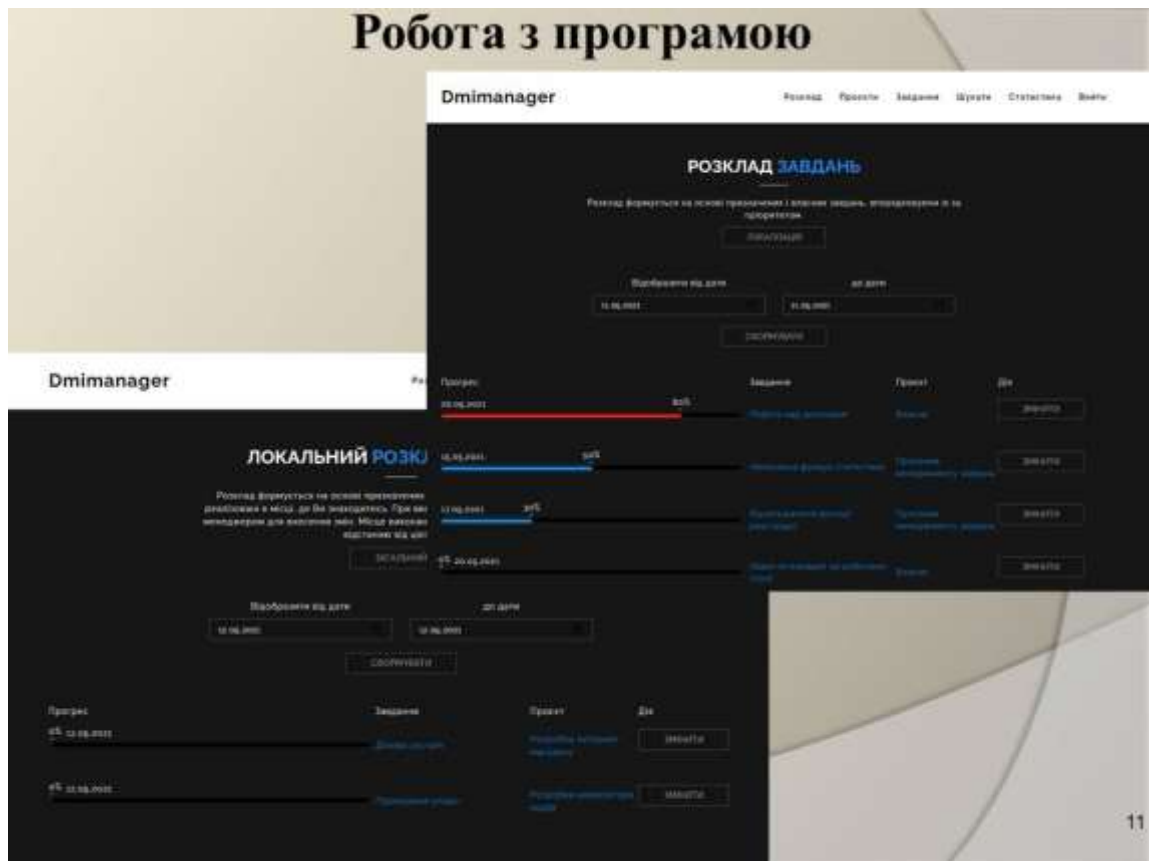


Рисунок В.11 – Слайд № 11

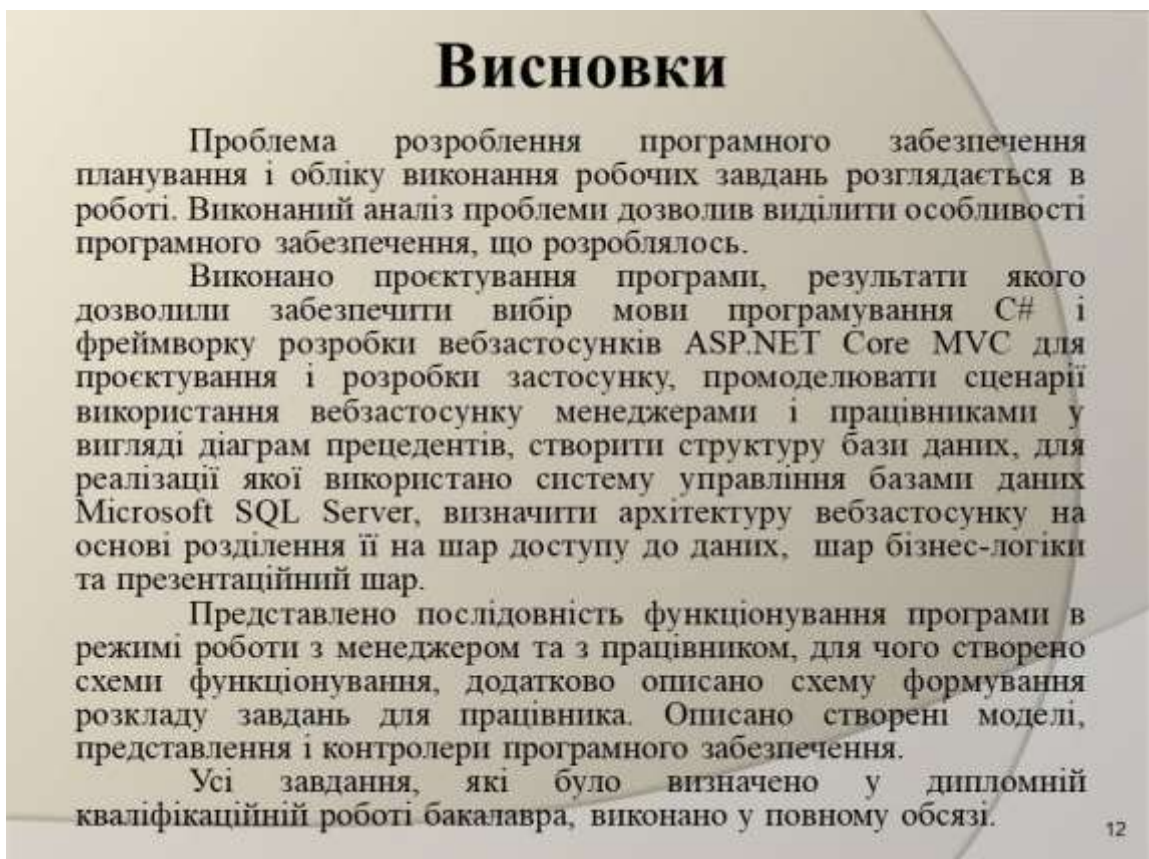


Рисунок В.12 – Слайд № 12