МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
кафедра комп'ютерних наук

Система управління проєктами

Курсова робота
Рівень вищої освіти – перший (бакалаврський)

Виконав:
студент 3 курсу, 344а групи
Андронік В.І.
Керівник:
асистент **Ватаманіца Е.В**.

Чернівці – 2024

Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та комп'ютерних наук

Кафедра Комп'ютерних наук

Спеціальність Комп'ютерні науки

Освітній ступінь Бакалавр

Форма навчання денна курс 3 група 344а

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ Ушенко Ю.О.
(підпис) (ініціали, прізвище)
28 серпня 2023 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТА

Андроніка Веніаміна Ігоровича
(прізвище , ім'я, по батькові)

1. Тема роботи
Система управління проєктами

затверджена протоколом засідання кафедри від «28» серпня 2023 року № 1
2. Термін подання студентом закінченої роботи 22.05.2024
3. Вхідні дані до роботи
Visual Studio Code, MySQL Workbench

4. Зміст розрахунково-пояснювальної записки (перелік питань, які треба розробити)
Вступ
Розділ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ
1.1 Опис предметної області. обґрунтування необхідності використання пз в предметній області
1.2 Аналіз програм-аналогів
1.3 Постановка задачі на розробку ПЗ
РОЗДІЛ ІІ. РОЗРОБКА ПЗ ДЛЯ КУРСОВОЇ РОБОТИ
2.1 Вибір інструментарію для реалізації ПЗ для курсової роботи
2.2 Проектування та реалізація БД
2.3 Реалізація ПЗ для курсової роботи
2.4 Інструкція користувача
Висновки

Список використаних джерел

Додатки

5. Перелік графічного, наочного матеріалу

Скріншоти інтерфейсу

6. Консультант(и) курсової роботи

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КУРСОВОЇ РОБОТИ

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітки
1	Отримання завдання на курсову роботу	01.09.2023	виконано
2	Аналіз предметної області, дослідження літератури та матеріалів на задану тему	20.10.2023	виконано
3	Аналіз існуючих аналогів програмного забезпечення	08.11.2023	виконано
4	Постановка задачі за темою курсової роботи	10.11.2023	виконано
5	Вибір інструментальних засобів розробки системи	17.11.2023	виконано
6	Проєктування структури та алгоритму роботи розроблюваної системи	21.11.2023	виконано
7	Формування інструкції користувача	20.01.2024	виконано
8	Написання розділів пояснювальної записки	11.02.2024	виконано
9	Представлення закінченої роботи на перевірку	21.05.2024	виконано
10	Захист курсової роботи	24.05.2024	виконано

Студент _____ Андронік В.І.
(підпис)

Науковий керівник _____ Ватаманіца Е.В.
(підпис)

«___» ____ 2024 р.

ЗМІСТ

Вступ 5
Розділ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Опис предметної області 7
1.2 Аналіз програм-аналогів 8
1.3 Постановка задачі на розробку ПЗ для «Системи управління проєктами» 10
Висновок 12
Розділ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ «Системи управління проєктами» 13
2.1 Опис інструментів розробки 13
2.2 Проектування та реалізація бази даних 13
2.2.1 **Сутності**: 13
2.2.2 **Зв'язки**: 15
2.2.3 ER-Діаграма створеної БД: 16
2.3 Реалізація програмного продукту для Системи управління проєктами 16
2.4 Інструкція користувача 22
Висновок 25
ВИСНОВКИ 26
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 27
ДОДАТКИ 28
ДОДАТОК 1 28
ДОДАТОК 2 29

Вступ

Мета: розробити веб-додаток WorkWave для управління завданнями та проектами, який дозволить користувачам створювати, організовувати та відстежувати завдання у різних проектах.

Завдання:

a) Реалізація системи реєстрації та авторизації користувачів
   - Користувачі повинні мати можливість створювати облікові записи, використовуючи свої електронні адреси та паролі.
   - Забезпечення надійного захисту даних користувачів через механізми аутентифікації та авторизації.

b) Створення та управління проектами
   - Користувачі повинні мати можливість створювати нові проекти та призначати завдання для них.
   - Організація проектів та завдань з урахуванням статусів виконання завдань.

c) Призначення відповідальних та відстеження виконання завдань
   - Реалізація функціоналу призначення відповідальних за кожне завдання.
   - Можливість відстеження прогресу виконання завдань.

d) Коментування завдань та спільна робота
   - Забезпечення можливості користувачам залишати коментарі та нотатки до завдань.
   - Впровадження функціоналу для роботи в команді над спільними проектами.

e) Технічна реалізація
   - Реалізація бекенду на Node.js для обробки запитів користувачів та зберігання даних.
   - Використання бази даних для зберігання інформації про користувачів, їх проекти та завдання.

В сучасних умовах швидкого розвитку інформаційних технологій та зростання кількості проектів, які виконуються віддалено, виникає необхідність у використанні ефективних інструментів для управління проектами. WorkWave дозволяє спростити цей процес, забезпечуючи можливість командної роботи, організації та відстеження завдань. Це особливо актуально в умовах гнучкого графіка роботи та розподілених команд, де комунікація та координація між учасниками проекту є критично важливими.

Об'єктом дослідження є процес управління завданнями та проектами в командній роботі. Предметом дослідження є розробка та впровадження веб-додатку для ефективного управління цими процесами.

Практична значущість даної роботи полягає в розробці зручного та функціонального інструменту для управління проектами, що дозволить користувачам значно підвищити ефективність роботи, знизити витрати часу на координацію завдань та покращити загальний результат діяльності команди.

На сьогоднішній день існує багато рішень для управління проектами, таких як Jira, Trello та Worksection. Ці інструменти широко використовуються в різних галузях для планування, організації та відстеження завдань. Проте, кожен з них має свої обмеження та особливості. WorkWave пропонує унікальні можливості, які поєднують зручний інтерфейс, гнучку систему тегів та інтуїтивно зрозумілий дашборд, що робить його привабливим вибором для команд будь-якого розміру.

Розділ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасних умовах, коли проекти часто виконуються розподіленими командами, існує потреба в ефективних інструментах для координації дій, планування та контролю за виконанням завдань. Веб-додатки, такі як WorkWave, покликані вирішувати ці завдання, забезпечуючи зручну платформу для співпраці, моніторингу прогресу та комунікації між учасниками проекту.

Незважаючи на наявність численних інструментів для управління проектами, багато з них мають суттєві недоліки, які обмежують їх ефективність та зручність для користувачів:

a) **Складність інтерфейсу**: Багато існуючих рішень мають надто складний або перевантажений інтерфейс, що ускладнює їх використання, особливо для нових користувачів.
b) **Обмежені можливості налаштування**: Деякі інструменти не дозволяють користувачам гнучко налаштовувати інтерфейс та функціонал відповідно до своїх потреб, що знижує їх універсальність.
c) **Відсутність інтеграції**: Багато рішень не підтримують інтеграцію зі сторонніми сервісами, що ускладнює роботу з іншими інструментами та платформами, які використовуються в проекті.
d) **Обмежені можливості спільної роботи**: Деякі інструменти не забезпечують ефективної підтримки командної роботи, що ускладнює комунікацію та координацію між учасниками проекту.
e) **Недостатня підтримка мобільних пристроїв**: Відсутність адаптованих мобільних версій або додатків обмежує можливість користувачів працювати з завданнями на ходу.

Розробка WorkWave покликана усунути ці недоліки та забезпечити користувачів більш зручним, гнучким та функціональним інструментом для управління проектами. WorkWave пропонує інтуїтивно зрозумілий інтерфейс, який спрощує процес навчання та використання додатку. Завдяки широким можливостям налаштування, користувачі можуть адаптувати інструмент відповідно до своїх потреб та вимог конкретного проекту.

Інтеграція зі сторонніми сервісами дозволяє легко взаємодіяти з іншими інструментами, такими як календарі, месенджери та системи зберігання даних, що підвищує ефективність роботи. WorkWave також забезпечує потужну підтримку командної роботи, включаючи можливість

коментування завдань, сповіщення та інші засоби для ефективної комунікації та координації.

Використання WorkWave є доцільним з кількох причин. По-перше, додаток забезпечує підвищену продуктивність роботи команд завдяки ефективній координації завдань та проектів. Це досягається за рахунок інтеграції всіх необхідних інструментів в одній платформі, що зменшує час на перемикання між різними додатками та спрощує управління проектами.

По-друге, інтуїтивний інтерфейс та широкий функціонал WorkWave роблять його зручним для користувачів різного рівня підготовки, що підвищує залученість та знижує опір до впровадження нового інструменту. Крім того, можливість адаптації та налаштування додатку відповідно до конкретних потреб команди забезпечує його універсальність та гнучкість.

По-третє, підтримка мобільних пристроїв дозволяє користувачам працювати з додатком будь-де та будь-коли, що є критично важливим у сучасному динамічному середовищі. Це забезпечує безперервний доступ до завдань та проектів, сприяє швидкому реагуванню на зміни та підвищує загальну ефективність роботи команди.

Нарешті, інтеграція зі сторонніми сервісами робить WorkWave важливим елементом екосистеми управління проектами, що забезпечує безшовну взаємодію з іншими інструментами та сервісами, які використовуються в проектній діяльності.

Таким чином, розробка та впровадження WorkWave не тільки усуває недоліки існуючих рішень, але й забезпечує ефективний, зручний та гнучкий інструмент для управління проектами та завданнями, що підвищує продуктивність та якість роботи команд.

1.2 Аналіз програм-аналогів

Для аналізу були обрані програми-аналоги: **Trello**, **JIRA**, **Worksection**. Кожна з них широко використовується в області управління проектами та завданнями.

Trello є одним з найпопулярніших інструментів для управління завданнями, який використовує підхід канбан. Він пропонує візуальний інтерфейс для організації завдань у вигляді дошок і карток, що дозволяє легко стежити за прогресом проекту.

JIRA - це потужний інструмент від Atlassian, який надає широкі можливості для управління проектами, особливо в сфері розробки програмного забезпечення. JIRA підтримує багаті функції для відстеження завдань, багів, планування спринтів та аналізу продуктивності команди.

Worksection - це український інструмент для управління проектами, який пропонує зручний інтерфейс для командної роботи, організації завдань та комунікації. Він також включає функції для відстеження часу, планування та створення звітів.

Таблиця 1.1

Параметр	Trello	JIRA	Worksection	WorkWave (планується)
Інтерфейс	Інтуїтивний, канбан-дошки	Складний, але потужний, з гнучкими налаштуваннями	Зручний, простий, підтримка різних видів подань	Планується
Функціональність	Базові функції управління завданнями	Розширені функції для розробки ПЗ, підтримка Agile	Базові та розширені функції для управління проектами	Базові та розширені функції для управління проектами
Інтеграції	Підтримка багатьох інтеграцій	Широкі інтеграції з інструментами Atlassian та сторонніми сервісами	Підтримка основних інтеграцій	Підтримка основних інтеграцій, можливість розширення
Мобільність	Доступні мобільні додатки	Доступні мобільні додатки	Доступні мобільні додатки	Планується мобільна версія
Командна робота	Спільні дошки, коментування	Потужні засоби для командної роботи, Agile методології	Спільні проекти, коментування	Спільні проекти, коментування
Підтримка	Багато навчальних матеріалів	Велика кількість документації та підтримки	Добре організована підтримка	Планується надання підтримки
Ціна	Безкоштовний план, платні функції	Платна, з різними планами	Платна, з безкоштовним пробним періодом	Безкоштовний план

Порівняльний аналіз трьох популярних програм для управління проектами та завданнями показує, що кожна з них має свої переваги та недоліки. Trello вирізняється своєю простотою та інтуїтивним інтерфейсом,

але обмежений у функціональності для великих проектів. JIRA пропонує потужні інструменти та розширені можливості для звітності, але має складний інтерфейс та високу вартість. Worksection є гнучким та простим у використанні інструментом, але має обмежену кількість інтеграцій та функціональність для складних проектів.

1.3 Постановка задачі на розробку ПЗ для «Системи управління проєктами»

Мета створення WorkWave полягає у розробці веб-додатку для ефективного управління завданнями та проектами. Застосунок повинен забезпечити користувачам можливість створювати, організовувати та відстежувати завдання, а також сприяти спільній роботі над проектами.

Вимоги до застосунку

Функціональні вимоги:

- Система реєстрації та авторизації користувачів з використанням електронної пошти та пароля.
- Можливість створення нових проектів та призначення завдань для кожного проекту.
- Додавання, редагування та видалення завдань з різними статусами (наприклад, "в роботі", "завершено").
- Призначення відповідальних за завдання та відстеження їх виконання.
- Система коментування та нотаток до кожного завдання для спільної роботи.
- Надання доступу до спільних проектів іншим користувачам, що дозволяє бачити і редагувати завдання.
- Інтеграція зі сторонніми сервісами для розширення функціоналу (наприклад, календарі, поштові сервіси).

Нефункціональні вимоги:

- Висока продуктивність та швидкодія системи, незалежно від кількості користувачів та завдань.
- Забезпечення безпеки даних користувачів через механізми авторизації та аутентифікації.
- Інтуїтивно зрозумілий та зручний інтерфейс для користувачів з різним рівнем технічних навичок.
- Можливість масштабування системи при зростанні кількості користувачів та даних.
- Висока надійність та відмовостійкість системи, мінімізація часу простою.

Опис користувачів застосунком

WorkWave планується зробити призначеним для використання різними категоріями користувачів:

- Проектні менеджери: створюють та організовують проекти, призначають завдання та відповідальних, відстежують прогрес.
- Члени команди: виконують призначені завдання, залишають коментарі та співпрацюють з іншими учасниками проектів.
- Адміністратори системи: керують користувачами, налаштовують права доступу, забезпечують безпеку та технічну підтримку.

Обмеження у створенні застосунку

Розробка WorkWave може стикнутися з такими обмеженнями:

- Технічні обмеження: необхідність підтримки різних браузерів та операційних систем, обмеження швидкості інтернет-з'єднання користувачів.
- Безпека: забезпечення високого рівня захисту даних користувачів, уникнення несанкціонованого доступу та атак.
- Ресурси: обмежений бюджет та час на розробку, можливість залучення кваліфікованих фахівців.

Короткий опис етапів реалізації ПЗ

Процес розробки WorkWave включає кілька основних етапів:

- Планування: визначення вимог до системи, створення технічного завдання та плану розробки.
- Розробка: програмування бекенду, інтеграція з базою даних та сторонніми сервісами.
- Тестування: перевірка функціональності, продуктивності та безпеки системи, виявлення та виправлення помилок.
- Впровадження: розгортання системи на сервері, налаштування та запуск в експлуатацію.
- Підтримка та оновлення: технічна підтримка користувачів, виправлення помилок, додавання нових функцій.

Оптимальні вимоги до апаратного та програмного забезпечення

Для ефективного функціонування WorkWave рекомендуються наступні вимоги:

Апаратне забезпечення:

- Сервер: процесор з високою продуктивністю (не менше 4 ядер), оперативна пам'ять від 8 ГБ, SSD-диск для швидкого доступу до даних.

- Клієнтські пристрої: комп'ютери та мобільні пристрої зі стабільним інтернет-з'єднанням, сучасними браузерами.

Програмне забезпечення:

- Сервер: операційна система Linux або Windows, веб-сервер Apache або Nginx, база даних MySQL або PostgreSQL.
- Бекенд: Node.js, бібліотеки для авторизації та аутентифікації, API для інтеграції зі сторонніми сервісами.

Висновок

У результаті виявлено, що додаток WorkWave є веб-додатком для управління завданнями та проектами, який дозволяє користувачам створювати, організовувати та відстежувати завдання у різних проектах. Основними функціями є реєстрація та авторизація користувачів, створення проектів, додавання та видалення завдань, призначення відповідальних, коментування завдань та доступ до спільних проектів.

Технічна реалізація передбачає фронтенд, бекенд та базу даних, що забезпечує відображення завдань та інтерфейсу користувача, обробку запитів користувачів та зберігання інформації про користувачів, їх проекти та завдання. Використання механізмів авторизації та аутентифікації, системи коментування, сповіщень та спільної роботи покликані забезпечити безпеку даних користувачів та ефективну співпрацю між учасниками проектів.

Порівняльний аналіз з програмами-аналогами, такими як Trello, JIRA та Worksection, показав, що WorkWave має схожий функціонал, але може бути більш привабливим завдяки інтуїтивно зрозумілому інтерфейсу та можливості спільної роботи з іншими учасниками проектів.

З урахуванням висновків розділу можна зробити висновок, що розробка проекту WorkWave є доцільною та може значно полегшити управління завданнями та проектами для широкого кола користувачів.

Розділ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ «Системи управління проєктами»

2.1 Опис інструментів розробки

Для реалізації проекту "WorkWave" планується використання наступного інструментарію:

Система управління базами даних (СУБД): MySQL

MySQL використовується для зберігання інформації про користувачів, їх проекти та завдання. Ця СУБД обрана через свою надійність, популярність та швидкодію роботи з даними.

Серверна частина: Node.js

Node.js буде використаний для обробки запитів користувачів, авторизації та зберігання даних про проекти та завдання. Використання Node.js сприятиме асинхронності роботи додатку та забезпечить його ефективну роботу.

2.2 Проектування та реалізація бази даних

2.2.1 **Сутності**:

- Користувачі (Users)
    - UserId: CHAR(38) (PRIMARY KEY)
    - Email: NVARCHAR(256) (UNIQUE NOT NULL)
    - PasswordHash: NVARCHAR(255) (NOT NULL)
    - UserName: NVARCHAR(80) (UNIQUE NOT NULL)
    - ProfileImage_Path: VARCHAR(1024)
    - CreateDate: DATETIME
    - UpdateDate: DATETIME
- Проекти (Projects)
    - ProjectId: CHAR(38) (PRIMARY KEY)
    - ProjectName: NVARCHAR(100) (NOT NULL)
    - ProjectDescription: NVARCHAR(1000) (NOT NULL)
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Стани (States)

    - StateId: CHAR(38) (PRIMARY KEY)
    - Title: NVARCHAR(20) (UNIQUE NOT NULL)
    - ProjectId: CHAR(38) (FOREIGN KEY REFERENCES
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Спринти (Sprints)
    - SprintId: CHAR(38) (PRIMARY KEY)
    - ProjectId: CHAR(38) (FOREIGN KEY REFERENCES Projects(ProjectId))
    - SprintNumber: INT (NOT NULL)
    - StartDate: DATETIME (NOT NULL)
    - EndDate: DATETIME (NOT NULL)
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Завдання (WorkItems)
    - WorkItemId: CHAR(38) (PRIMARY KEY)
    - Title: NVARCHAR(100) (NOT NULL)
    - Description: NVARCHAR(1000) (NOT NULL)
    - StateId: CHAR(38) (FOREIGN KEY REFERENCES States(StateId))
    - ProjectId: CHAR(38) (FOREIGN KEY REFERENCES Projects(ProjectId))
    - UserId: CHAR(38) (FOREIGN KEY REFERENCES Users(UserId))
    - SprintId: CHAR(38) (FOREIGN KEY REFERENCES Sprints(SprintId))
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Історія продуктивності (PerformanceHistories)
    - HistoryId: CHAR(38) (PRIMARY KEY)
    - ProjectId: CHAR(38) (FOREIGN KEY REFERENCES Projects(ProjectId))
    - UserId: CHAR(38) (FOREIGN KEY REFERENCES Users(UserId))
    - Date: DATETIME (DEFAULT(now()))

    - Description: NVARCHAR(1000) (NOT NULL)
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Теги (Tags)
    - TagId: CHAR(38) (PRIMARY KEY)
    - TagName: NVARCHAR(50) (NOT NULL)
    - TagColor: NVARCHAR(6) (NOT NULL)
    - WorkItemId: CHAR(38) (FOREIGN KEY REFERENCES WorkItems(WorkItemId))
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME
- Коментарі (Comments)
    - CommentId: CHAR(38) (PRIMARY KEY)
    - CommentText: NVARCHAR(1000) (NOT NULL)
    - CommentDate: DATETIME (DEFAULT(now()))
    - UserId: CHAR(38) (FOREIGN KEY REFERENCES Users(UserId))
    - WorkItemId: CHAR(38) (FOREIGN KEY REFERENCES WorkItems(WorkItemId))
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME

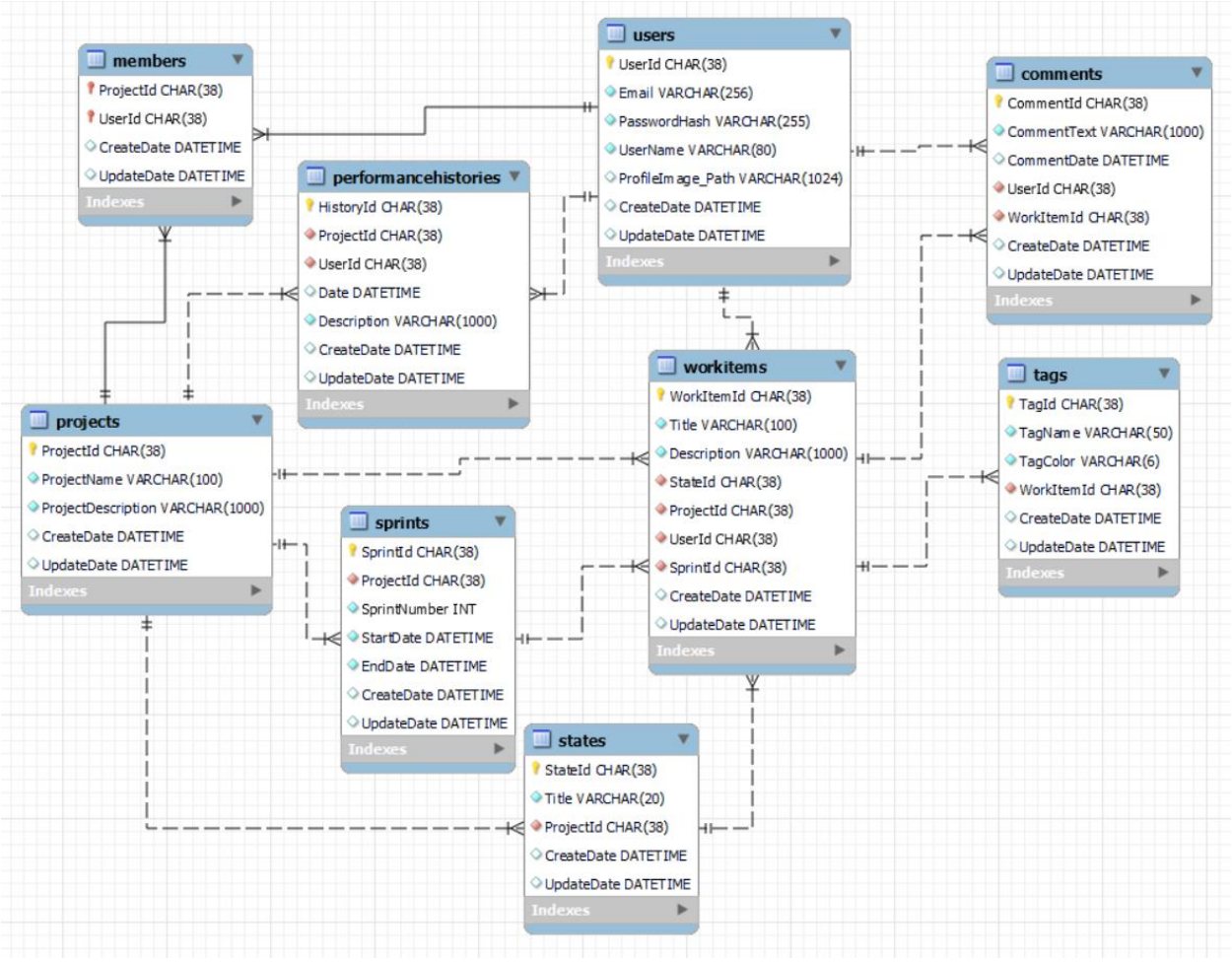
- Учасники проекту (Members)
    - ProjectId: CHAR(38) (FOREIGN KEY REFERENCES Projects(ProjectId))
    - UserId: CHAR(38) (FOREIGN KEY REFERENCES Users(UserId))
    - PRIMARY KEY (ProjectId, UserId)
    - CreateDate: DATETIME (DEFAULT(now()))
    - UpdateDate: DATETIME

2.2.2 **Зв'язки**:

- Користувач може мати багато проектів (1-N)
- Проект може мати багато станів (1-N)
- Проект може мати багато спринтів (1-N)
- Проект може мати багато завдань (1-N)

- Користувач може мати багато завдань (1-N)
- Завдання може мати багато коментарів (1-N)
- Завдання може мати багато тегів (1-N)
- Користувач може мати багато історій продуктивності (1-N)
- Проект може мати багато учасників (1-N)

2.2.3 ER-Діаграма створеної БД:



2.3 Реалізація програмного продукту для Системи управління проєктами

Процес розробки програмного забезпечення можна розділити на кілька основних етапів, які допомагають систематизувати та ефективно керувати проектом. Нижче наведено декілька основних етапів розробки програмного забезпечення для проекту "WorkWave":

Аналіз вимог та проектування

На етапі аналізу вимог та проектування здійснюється детальне вивчення потреб користувачів та визначення функціональних та нефункціональних вимог до програмного забезпечення. Основні завдання цього етапу включають:

1. Збір та аналіз вимог: Команда збирає вимоги від клієнтів або стейкхолдерів, розуміє їх потреби та вимоги до програмного забезпечення. Це може включати в себе проведення спеціальних сесій, спілкування з користувачами та аналіз конкурентів.
2. Формулювання вимог: На основі зібраних вимог формулюються документи вимог, які визначають функціональні та нефункціональні вимоги до програмного забезпечення. Це включає в себе створення специфікацій вимог, use case діаграм, сценаріїв та інших документів.
3. Визначення архітектури системи: Розробляється загальна архітектура програмного забезпечення, включаючи вибір підходів до розробки, розподіл компонентів та визначення системних інтерфейсів.
4. Вибір технологій та інструментів: На цьому етапі обираються технології, мови програмування, фреймворки та інші інструменти, які будуть використовуватися для реалізації програмного забезпечення.
5. Розробка плану проекту: Формується план реалізації проекту, в якому визначаються кроки, терміни та ресурси, необхідні для успішного завершення проекту.
6. Оцінка ризиків та аналіз вартості: Визначаються потенційні ризики проекту та відповідні стратегії їх управління. Також проводиться аналіз вартості проекту, включаючи витрати на розробку, тестування та підтримку програмного забезпечення.

Проектування бази даних

Під час проектування бази даних для бекенд частини додатку, головна мета - створити структуру даних, яка ефективно відображає вимоги функціональності додатку та забезпечує надійне зберігання та маніпулювання даними. Нижче наведено детальний опис етапів проектування бази даних:

1. Проектування схеми бази даних: На основі аналізу вимог створюється схема бази даних. Це включає визначення таблиць, їхніх полів та відносин між ними. Наприклад, для кожного об'єкта додатку (наприклад, користувачі, проекти, завдання) створюється відповідна таблиця.
2. Визначення первинних ключів та зовнішніх ключів: Кожна таблиця має мати первинний ключ, що однозначно ідентифікує кожен запис, а також зовнішні ключі, що встановлюють зв'язки між таблицями. Це дозволяє створити зв'язки між об'єктами додатку та забезпечити цілісність даних.
3. Визначення індексів: Для оптимізації швидкості пошуку та фільтрації даних встановлюються індекси на полях, які часто використовуються у запитах до бази даних. Це може включати

поля, які використовуються у запитах для фільтрації, сортування та об'єднання даних.

4. Оптимізація структури даних: Після створення схеми бази даних виконується її оптимізація. Це може включати розширення або скорочення кількості таблиць, нормалізацію бази даних для уникнення дублювання даних, а також розгляд можливості використання різних типів даних для зберігання інформації.
5. Створення бази даних: На основі схеми бази даних створюється сама база даних. Це може бути виконано за допомогою SQL-запитів або спеціалізованих інструментів для управління базами даних.

Розробка логіки застосунку

На цьому етапі реалізується логіка роботи застосунку, включаючи обробку запитів користувачів, керування сесіями, модулі для реєстрації та авторизації користувачів, обробку даних тощо. Для проекту "WorkWave" це означає створення функцій для створення, оновлення та видалення проектів, завдань, коментарів тощо.

На цьому етапі розробляються. Це включає створення запитів, обробку запитів авторизації, перевірку прав доступу тощо.

1. Розробка модуля авторизації:
   - Реалізація механізму реєстрації та аутентифікації користувачів. Це включає створення API для реєстрації нових користувачів (Додаток 1) з валідацією введених даних (Додаток 2), а також механізми аутентифікації, такі як перевірка логіну та пароля, які наведені нижче:

```
const user = await UserModel.findOne({ where: { email: req.body.email } });
        if (!user) {
            return res.status(400).json({
                message: 'Не вірний логін або пароль'
            });
        }
        const isValidPassword = await bcrypt.compare(req.body.password,
user.passwordHash);
        if (!isValidPassword) {
            return res.status(400).json({
```

```
                message: 'Не вірний логін або пароль'
            });
        }
```

- Захист доступу до ресурсів системи. Визначення прав доступу користувачів до функцій та даних в системі.

```
export default (req, res, next) => {
    const token = (req.headers.authorization || '').replace(/Bearer\s?/, '');
    if (token) {
        try {
            const decoded = jwt.verify(token,
'HL34HO20ACO020HBNDPQ103J8NCOP');
            req.userId = decoded.userId;
            next();
        }
        catch (err) {
            res.status(403).json({
                message: 'Немає доступа'
            });
        }
    }
    else {
        res.status(403).json({
            message: 'Немає доступа'
        });
    }
};
```

2. Розробка майбутньої взаємодії з клієнтською частиною
    - Створення API для взаємодії з фронтендом. Це включає реалізацію ендпоінтів для обробки запитів від клієнтської частини. Ендпоінти авторизації:

```
const authRouter = express.Router();
```

```
// authorization

authRouter.post('/login', loginValidation, handleValidationErrors,
UserController.login);

authRouter.post('/registration', registrationValidation,
handleValidationErrors, UserController.register);

authRouter.post('/reset', UserController.resetPassword);

authRouter.get('/me', checkAuth, UserController.getMe);

authRouter.get('/my-projects', checkAuth, UserController.getMyProjects);

authRouter.get('/count-active-users-today/:id', checkAuth,
UserController.getActiveUsersCountToday);

authRouter.get('/items-assigned-me', checkAuth,
UserController.getItemsAssignedMe);

authRouter.get('/completion-percentage/:id', checkAuth,
UserController.getCompletionPercentage);


export default authRouter;
```

- Форматування відповідей сервера у форматі, зрозумілому для фронтенда (наприклад, JSON):

```
    // Отримання ідентифікатора поточного користувача з запиту
        const userId = req.userId;
        // Запит до бази даних для отримання робочих елементів, призначених
поточному користувачеві
        const assignedWorkItems = await WorkItemModel.findAll({
            where: {
                userId: userId // Ідентифікатор поточного користувача
            }
        });
        // Підрахунок кількості робочих елементів, призначених поточному
користувачеві
        const assignedWorkItemCount = assignedWorkItems.length;
        res.json({
            success: true,
            assignedWorkItemCount,
            assignedWorkItems
        });
```

- Обробка помилок і винятків. Налаштування системи обробки помилок та повідомлень про помилки для зручного відлагодження та підтримки.

```
try {…}
catch (err) {
    console.error("Errors: ", err);

    res.status(500).json({
        success: false,
        message: "Не вдалось отримати робочі елементи, призначені вам"
    });
}

const projectId = req.params.id;
if (!projectId) {
    return res.status(404).json({
        success: false,
        message: 'Введіть ідентифікатор проєкта'
    });
}

const user = await UserModel.findOne({ where: { email: req.body.email } });
if (!user) {
    return res.status(400).json({
        message: 'Не вірний логін або пароль'
    });
}
```

3. Документування:
    - Підготовка документації до API, включаючи опис доступних ендпоінтів, формат вхідних та вихідних даних, інструкції з використання та приклади запитів і відповідей. Приклад одного із запитів зображено на рисунок 2.1.

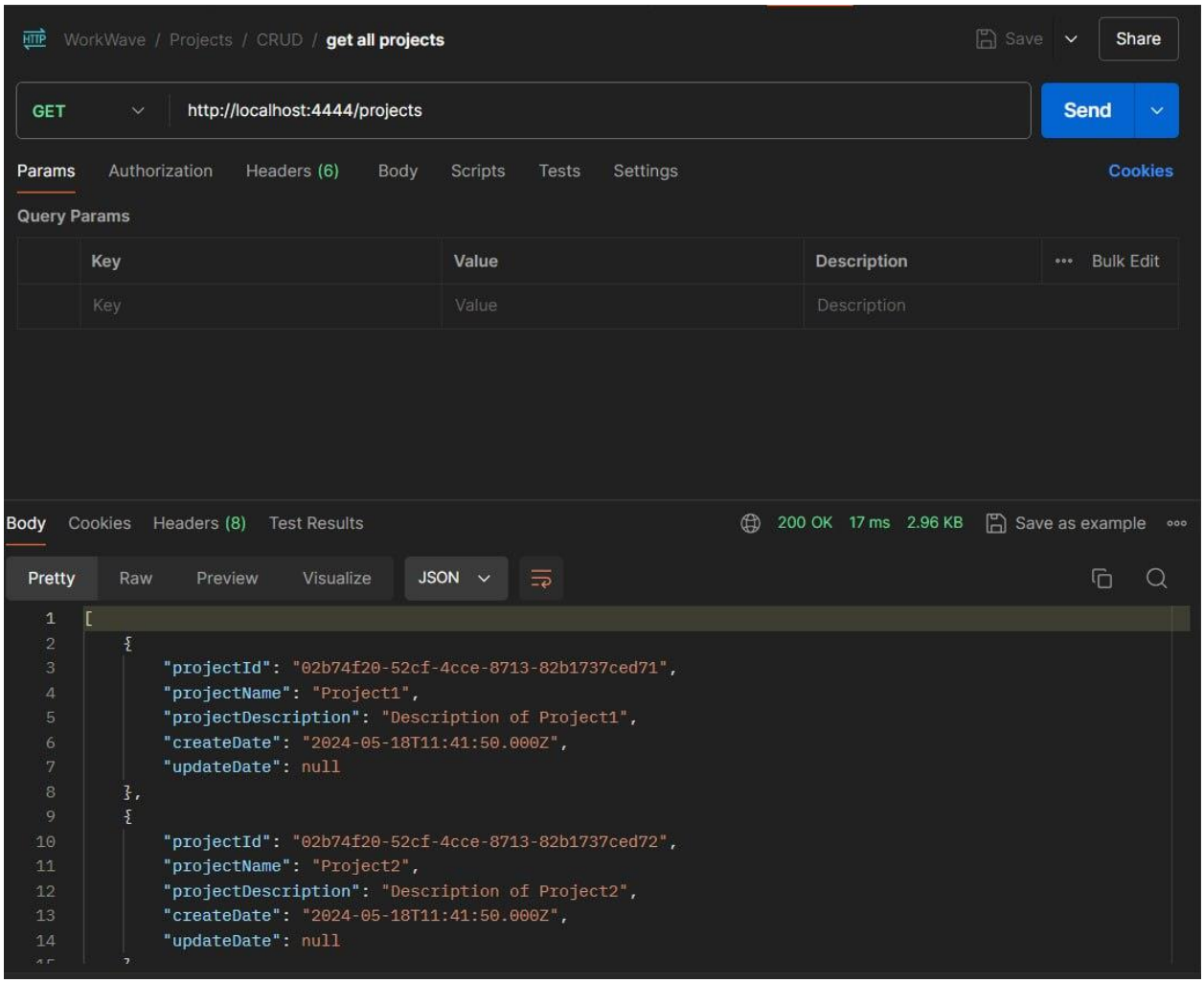


Рисунок 2.1

2.4 Інструкція користувача

Користувачі можуть використовувати створений застосунок для управління своїми проектами та завданнями, спільної роботи з колегами та ефективного виконання робочих завдань. Ось опис того, як користувачі можуть взаємодіяти із створеним застосунком:

Реєстрація та авторизація:

Користувачі реєструються у системі, вводячи свою електронну адресу та пароль, Рисунок 2.2

Після реєстрації користувачі можуть авторизуватися, використовуючи свої облікові дані, Рисунок 2.3.

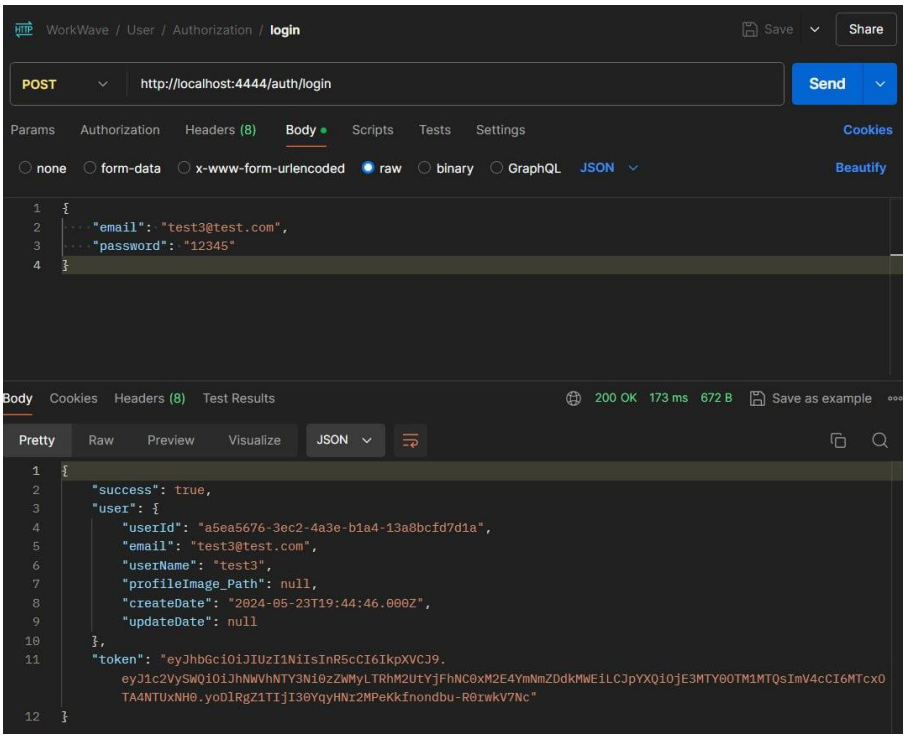


Рисунок 2.2

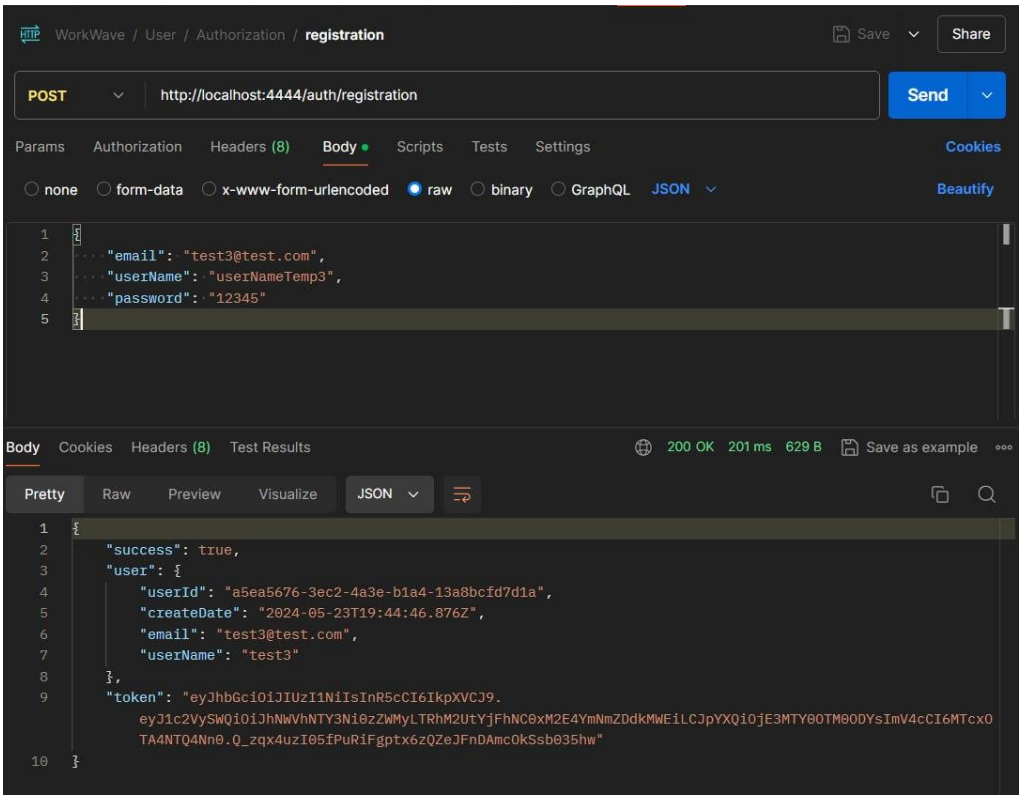


Рисунок 2.3

Після реєстрації користувачі можуть переглядати свої особисті дані, використовуючи свої облікові дані, Рисунок 2.4.

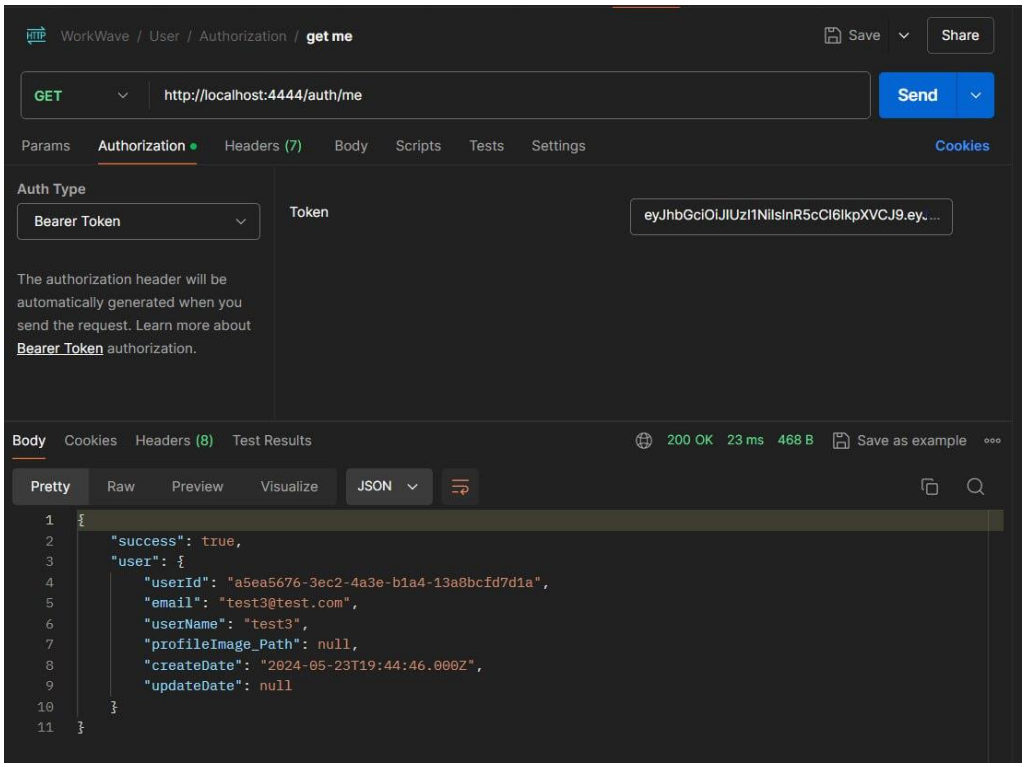


Рисунок 2.4

Якщо користувач не ввійшов в свій аккаунт то він не зможе переглянути свої особисті дані, Рисунок 2.5

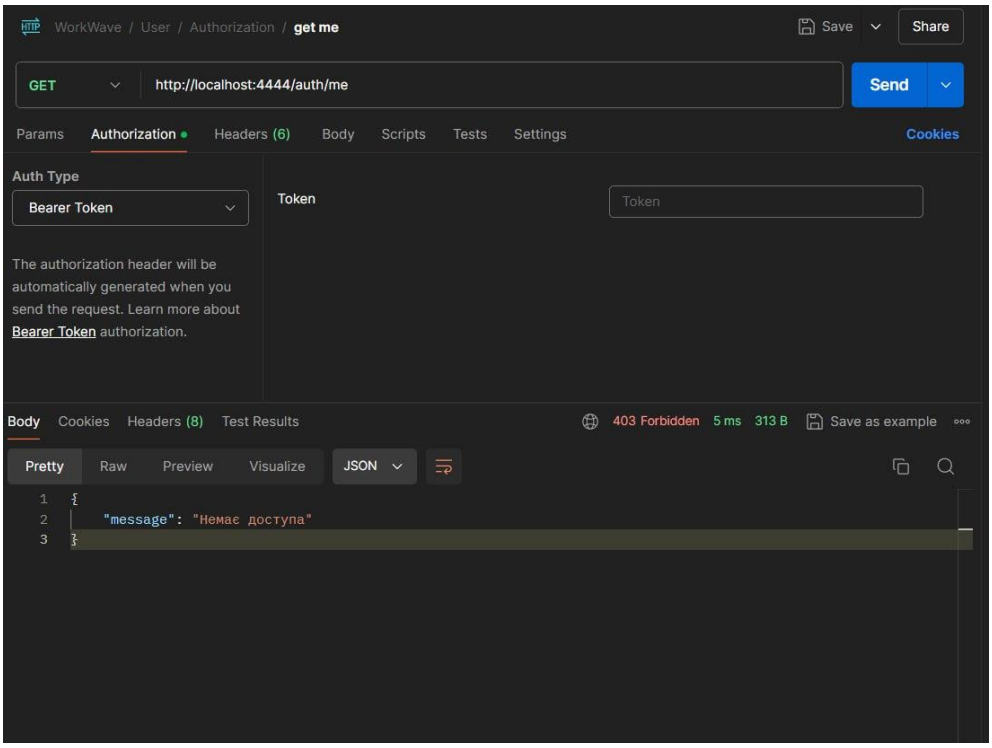


Рисунок 2.5

Висновок

Було проведено детальне проектування та реалізація програмного забезпечення WorkWave, зосереджуючись на серверній частині та базі даних.

Було виконано критичний аналіз інструментарію для реалізації ПЗ. Обрані інструменти, зокрема MySQL для бази даних, Node.js як серверна платформа, та фреймворк Express, мають свої переваги і недоліки. MySQL відзначається надійністю і продуктивністю, але може мати обмеження при масштабуванні великих обсягів даних. Node.js забезпечує швидку та ефективну обробку запитів, але потребує ретельного підходу до управління асинхронними процесами.

Розглянуто проектування та реалізацію бази даних. Були розроблені концептуальна, логічна та фізична моделі бази даних, що включали таблиці для зберігання даних користувачів, проектів, завдань, станів та інших сутностей.

Розглянуто процес розробки ПЗ, розділений на чотири основні етапи: розробка модуля авторизації, проектування інтерфейсу, розробка логіки застосунку, підключення бази даних і реалізація основних функцій. Логіка застосунку включала реалізацію основних операцій з даними: створення, оновлення, видалення та отримання інформації про проекти і завдання. Підключення бази даних забезпечило інтеграцію серверної частини з базою даних, що дозволило зберігати та обробляти дані.

Було зроблено детальний опис роботи користувача із створеним застосунком. Користувач може реєструватися, входити до системи, створювати і керувати проектами, додавати завдання, призначати відповідальних, коментувати завдання та відстежувати їхній стан.

Таким чином, реалізація WorkWave забезпечила створення надійного інструменту для управління проектами та завданнями. Використані технології та інструменти дозволили створити продуктивний і безпечний бекенд, що задовольняє основні потреби користувачів у управлінні робочими процесами. Дальший розвиток проекту може включати розширення функціоналу та інтеграцію зі сторонніми сервісами для ще більшої зручності і ефективності використання.

ВИСНОВКИ

У висновку курсової роботи на тему розробки веб-додатку WorkWave для управління завданнями та проектами можна підсумувати основні досягнення та результати, а також визначити їхнє практичне значення. Під час написання роботи було здійснено глибокий аналіз предметної області, визначено вимоги до програмного забезпечення, спроектовано базу даних, а також розроблено модуль авторизації та основну логіку застосунку. Проектування бази даних охопило створення таблиць для зберігання інформації про користувачів, проекти, завдання та інші ключові елементи системи, забезпечуючи необхідні зв’язки між даними та підтримуючи цілісність даних.

Практичне значення розглянутих питань полягає в тому, що WorkWave надає користувачам зручний інструмент для організації та управління робочими процесами. Це особливо важливо для команд, що працюють над спільними проектами, оскільки застосунок забезпечує можливість координації завдань, призначення відповідальних осіб та відстеження виконання завдань. Функціонал коментування та сповіщення сприяє покращенню комунікації всередині команди, що, в свою чергу, підвищує ефективність роботи та якість кінцевих результатів.

На основі виконаного дослідження та розробки можна рекомендувати декілька пропозицій щодо подальшого використання та розвитку застосунку. Зокрема, варто розширити функціонал WorkWave, додавши можливість планування та призначення термінів виконання завдань, створення детальних звітів та аналітики щодо продуктивності роботи команди. Інтеграція зі сторонніми сервісами, такими як календарі чи інструменти для комунікації, може ще більше підвищити зручність використання та забезпечити комплексний підхід до управління проектами. Також варто звернути увагу на забезпечення високого рівня безпеки даних користувачів, впровадивши додаткові механізми аутентифікації та шифрування.

Загалом, розробка WorkWave стала значущим кроком у створенні ефективного інструменту для управління завданнями та проектами, що може знайти широке застосування у різних сферах діяльності, сприяючи оптимізації робочих процесів та підвищенню продуктивності команд.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Приклад бекенд застосунку на Node.js: https://github.com/Archakov06/mern-blog-backend
2. Документація Node.js: https://nodejs.org/docs/latest/api/
3. Документація MySQL: https://dev.mysql.com/doc/

ДОДАТКИ

ДОДАТОК 1

```
export const register = async (req, res) => {
    try {
        const password = req.body.password;
        const salt = await bcrypt.genSalt(10);
        const hash = await bcrypt.hash(password, salt);

        const user = await UserModel.create({
            email: req.body.email,
            passwordHash: hash,
            userName: (req.body.email + "").split('@')[0],
            profileImage_Path: req.body.profileImage_Path
        });

        const token = jwt.sign(
            { userId: user.userId },
            "HL34HO20ACO020HBNDPQ103J8NCOP",
            { expiresIn: "30d" },
        );

        const userWithoutPassword = user.toJSON();
        delete userWithoutPassword.passwordHash;

        res.json({
            success: true,
            user: userWithoutPassword,
            token
        });
    }
    catch(err) {
        console.error("Errors: ", err);

        res.status(500).json({
            success: false,
            message: "Не вдалось зареєструвати користувача"
```

```
        });
    }
};
```

ДОДАТОК 2

```
export default (req, res, next) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json(errors.array());
    }

    next();
}
export const registrationValidation = [
    body('email', 'Не вірний формат електронної адреси').isEmail(),
    body('password', 'Пароль повинен бути мінімум 5 символів').isLength({
min: 5 }),
    body('userName', 'Вказіть ім\'я користувача').optional().isLength({ min:
5 }),
    body('profileImage_Path', 'Не вірне посилання на
аватарку').optional().isURL(),
];
```