

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Державний вищий навчальний заклад  
«Нововолинський електромеханічний коледж»  
Комп'ютерно-економічне відділення  
Циклова комісія комп'ютерних дисциплін

## **Пояснювальна записка**

до дипломного проекту молодшого спеціаліста

на тему: «Розробка вебзастосунку для управління проектами (на прикладі організації діяльності ДВНЗ «Нововолинський електромеханічний коледж»)»

Виконала: студентка IV курсу, групи 1-КТ-17

Галузь знань:  
12 Інформаційні технології

Спеціальність:  
123 Комп'ютерна інженерія

«Обслуговування комп'ютерних  
систем і мереж»

**Ткачук С.О.**

Керівник

**Дзіковська Ю.М.**

Консультант з  
економічної  
частини  
Нормоконтроль  
Рецензент

**Камінська О.Ю.**

**Ткаченко Ю.І.**

м. Нововолинськ – 2021 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
 Державний вищий навчальний заклад  
 «Нововолинський електромеханічний коледж»  
 Комп'ютерно-економічне відділення  
 Циклова комісія комп'ютерних дисциплін  
 Освітньо-кваліфікаційний рівень «Молодший спеціаліст»  
 Галузь знань: 12 Інформаційні технології  
 Спеціальність: 123 Комп'ютерна інженерія  
 «Обслуговування комп'ютерних систем і мереж»

**ЗАТВЕРДЖУЮ:**

**Голова циклової комісії**

\_\_\_\_\_ Ю. П. Дзюбак  
 «1» червня 2021 р.

### **ЗАВДАННЯ НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ**

Ткачук Софії Олександрівні, гр. 1-КТ-17

1. Тема проекту «Розробка вебзастосунку для управління проектами (на прикладі організації діяльності ДВНЗ «Нововолинський електромеханічний коледж»)»

керівник проекту Дзіковська Юлія Миколаївна, к. т. н., викладач,

затверджені наказом вищого навчального закладу №\_\_\_\_-с від «   » \_\_\_\_\_ 2021 року.

2. Строк подання студентом проекту: 21 червня 2021 року.

3. Вихідні дані до проекту: робоча станція для розробки (ПК), середовище програмування Visual Studio Code, SQLite 3, СУБД SQLite Browser, СУБД DBaver.

4. Зміст розрахунково-пояснювальної записки:

#### **ВСТУП**

#### **1 СУЧАСНІ МЕТОДИ КЕРУВАННЯ ПРОЕКТНОЮ ДІЯЛЬНІСТЮ**

- 1.1 Обґрунтування доцільності використання PMS
- 1.2 Організація управління проектною діяльністю за допомогою вебресурсів
- 1.3 PMS для освітніх закладів

#### **2 ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ПРОЕКТАМИ**

- 2.1 Аналіз предметної області
- 2.2 Побудова концептуальної моделі бази даних
- 2.3 Вибір системи управління базами даними та середовища розробки програмної частини

#### **3 РЕАЛІЗАЦІЯ ПРОЕКТУ**

- 3.1 Створення логічної моделі бази даних
- 3.2 Розробка дизайну сторінок застосунку
- 3.3 Реалізація програмної частини застосунку
- 3.4 Розгортання вебзастосунку на сервері

#### **4 АНАЛІЗ РЕЗУЛЬТАТІВ**

- 4.1 Тестування вебзастосунку для керування проектною діяльністю
- 4.2 Створення супровідної документації до застосунку

#### **5 ЕКОНОМІЧНА ЧАСТИНА**

ВИСНОВКИ  
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

5. Практичне завдання: розробка вебзастосунку для управління проектами.

6. Консультанти розділів проекту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	Камінська О.Ю., викладач	01.06.2021	

7. Дата видачі завдання: «1» червня 2021 року.

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Оформлення завдання та структуризація теоретичного матеріалу	01.06-03.06	
2	Проектування бази даних та логіки застосунку	04.06-06.06	
3	Програмна реалізація застосунку	07.06-14.06	
4	Налаштування та тестування застосунку	15.06-16.06	
5	Оформлення документації, підготовка презентації та доповіді	17.06-19.06	

Студент \_\_\_\_\_ **С. О. Ткачук**  
( підпис )

Керівник проекту \_\_\_\_\_ **Ю. М. Дзіковська**  
( підпис )

## ЗМІСТ

<b>ВСТУП.....</b>	<b>5</b>
<b>1 СУЧАСНІ МЕТОДИ КЕРУВАННЯ ПРОЕКТНОЮ ДІЯЛЬНІСТЮ .....</b>	<b>7</b>
1.1 Обґрунтування доцільності використання PMS .....	7
1.2 Організація управління проектною діяльністю за допомогою вебресурсів	8
1.3 PMS для освітніх закладів .....	11
<b>2 ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЕКТАМИ .....</b>	<b>15</b>
2.1 Аналіз предметної області.....	15
2.2 Побудова концептуальної моделі бази даних .....	18
2.3 Вибір системи управління базами даних та середовища розробки програмної частини.....	20
<b>3 РЕАЛІЗАЦІЯ ПРОЕКТУ .....</b>	<b>27</b>
3.1 Створення логічної моделі бази даних .....	27
3.2 Розробка дизайну сторінок застосунку.....	30
3.3 Реалізація програмної частини застосунку .....	32
3.4 Розгортання вебзастосунку на сервері.....	37
<b>4 АНАЛІЗ РЕЗУЛЬТАТІВ .....</b>	<b>39</b>
4.1 Тестування вебзастосунку для керування проектною діяльністю.....	39
4.2 Створення супровідної документації до застосунку .....	42
<b>5 ЕКОНОМІЧНА ЧАСТИНА .....</b>	<b>43</b>
<b>ВИСНОВКИ .....</b>	<b>55</b>
<b>ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....</b>	<b>56</b>

## ВСТУП

Нині впровадження інформаційних технологій у всі сфери людської діяльності є звичною справою. Переважно такі нововведення покращують продуктивність праці та якість виконаної роботи, оскільки забирають на себе монотонні дії, а також допомагають тримати інформацію впорядкованою і легкодоступною.

Зважаючи на загальну тенденцію переведення діяльності в електронний простір, спрощення керування проектною діяльністю через вебресурси є неминучим процесом, а наявність такого засобу в особистій власності підприємства чи компанії збільшує їх престижність на ринку. Тому все частіше з'являються застосунки як для управління широким спектром цілей, так і вузьконаправлені, що орієнтуються на конкретну область.

Уведення системи керування проектами спрямовано на надання зручних інструментів для керування задачами, подіями, завданнями, ресурсами, а також отримання звітності в різних формах та варіантах щодо виконаної роботи, використаних фінансів, залучених кадрів тощо. Функціонал варіюється від того, в якій області виробник бажає запровадити цю систему.

Сьогодні створення такої системи для конкретних потреб є не лише доцільним та вигідним, а й відносно простим. Це пов'язано з тим, що із часом кількість розвинутих технологій та зручних засобів для створення програмного забезпечення стрімко зростає, це відповідно зменшує вартість виготовлення та кількість часу і ресурсів, що витрачені на реалізацію. Зважаючи на це, розробник може більше часу витратити на забезпечення зручної взаємодії користувача з інтерфейсом, простоти використання програми та іншого, щоб покращило її загальну спроможність та доцільність існування на ринку.

Метою даної дипломної роботи є створення вебзастосунку управління проектами для зручної організації проектної діяльності в межах підприємства або організації, що буде оснащений простими інструментами для розподілу завдань в межах групи або команди та відслідковування їх виконання.

Основними завданнями проекту є:

- дослідження застосування PMS в проектній та навчальній діяльності, аналіз предметної області;
- проектування концептуальної моделі бази даних та загального функціоналу застосунку;
- вибір СКБД та середовища розробки застосунку (IDE);
- розробка застосунку;
- розгортка застосунку на сервері;
- тестування, внесення корективів;
- створення супровідної документації до застосунку.

Впровадження даного вебзастосунку в керування проектною або навчальною діяльністю надасть переваги, що полягатимуть у зручному контролі вчасності виконання завдань та їх розподілу між людьми, що будуть згруповані відповідно до їх ролей.

# 1 СУЧАСНІ МЕТОДИ КЕРУВАННЯ ПРОЕКТНОЮ ДІЯЛЬНІСТЮ

## 1.1 Обґрунтування доцільності використання PMS

Нині будь-яка людина чи організація під час виконання тих чи інших робіт має перед собою поставленими конкретні завдання, вчасне та якісне виконання яких є необхідними для досягнення цілі.

Керування проектами – це застосування знань, досвіду, методів і засобів до робіт проекту для задоволення вимог і очікувань від його учасників. Щоб задовольнити їх, необхідно знайти оптимальне сполучення між цілями, термінами, витратами, якістю й іншими характеристиками проекту.

Оскільки тримати інформацію про всі завдання в паперовому вигляді із збільшенням кількості робіт або проектів, а також з огляду на швидкість доступу та безпеку є недоцільним, популярності набувають системи керування проектами (Project Management Systems, або ж PMS).

Система керування проектом є програмним засобом, основною задачею якого є надати інструмент для керування ресурсами всередині проекту, часом, завданнями та подіями, а також для формування різномірної звітності щодо успішності, витрат, завантаженості ресурсів.

Комбінації інструментів, що надаються такими системами, варіюються від спеціалізації, в якій розробник висуває свій продукт, – PMS, орієнтовані на керування великими компаніями, та ті, які виготовлені для задоволення цілей освітніх закладів, відрізнятимуться набором засобів для виконання перерахованої вище роботи.

Зважаючи на появу вузьконаправлених систем керування, можна зробити висновок, що користувач стає більш вимогливим до продукту та вибирає той, який задовольняє саме його потреби. Тому наявність засобів, які полегшують управління окремо в бізнесовій, навчальній, проектній сферах та особистих питаннях із часом стає все більш актуальним та необхідним.

## **1.2 Організація управління проектною діяльністю за допомогою вебресурсів**

У теперішній умовах ефективне керування проектами вимагає контроль над великою кількістю різноманітної роботи. Для того, щоб менеджеру було зручно керувати всіма аспектами, доцільним є використання вебресурсів (вебзастосунків) для організації цієї діяльності.

Вебзастосунки, що позиціонуються як повноцінні PMS, повинні надавати такий мінімальний набір можливостей:

- групування людей або ресурсів за спільними ознаками;
- призначення одноосібних завдань та відслідковування їх виконання;
- призначення подій для певної групи виконавців та відслідковування їх виконання;
- отримання звітності щодо виконання проекту, завдань, подій, успішності виконавців.

Також, окрім основного функціоналу, PMS можуть надавати додаткові засоби:

- спілкування між учасниками, що реалізується шляхом створення чату або надання можливості коментування завдань; сюди ж входить і зв'язок із керівництвом проекту за допомогою особистого листування або створення питань для загального обговорення;
- обмін файлами, що можливий конкретно під завданнями та подіями або в окремо відведеному місці;
- інструменти керування фінансами всередині проекту;
- розширене керування ресурсами – відслідковування завантаженості, витрат на них, часу на виконання роботи;
- організація онлайн-конференцій.

Якщо менеджер проекту використовує певну методологію, то для збільшення ефективності виконання проекту є доцільним вибрати той застосунок, інструменти якого поєднуються із стилем роботи команди. Найпопулярнішими методологіями,



які знайшли свою реалізацію в застосунках, є Scrum та Kanban.

Kanban – японська методологія, основний принцип якої полягає у зборі всіх завдань проекту в одне місце (дошку) та їх рух по етапах виконання (колонках). Більшість застосунків гнучко підтримують інтеграцію цієї методології в себе за допомогою оцифрування дошки (рис. 1.1).

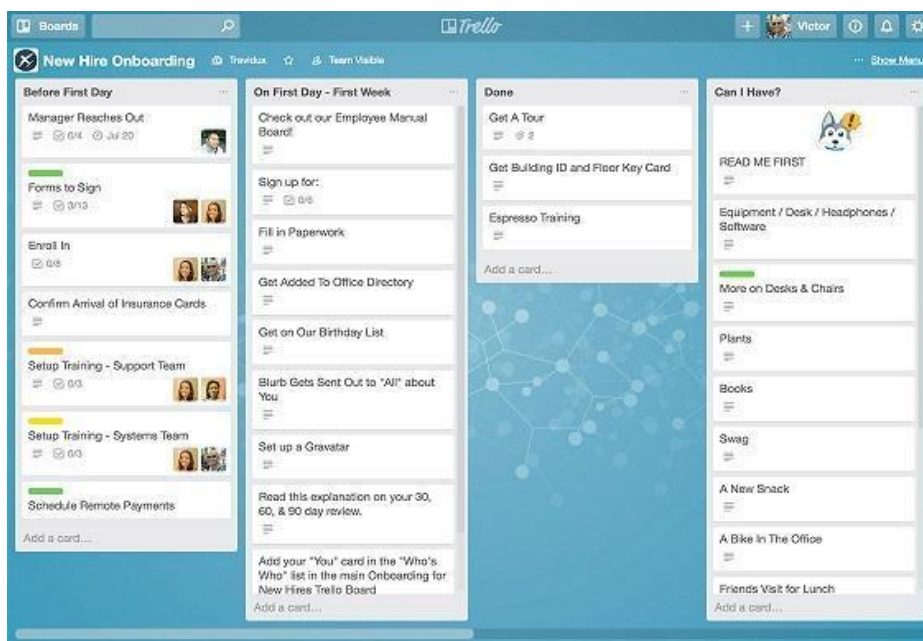


Рисунок 1.1 – Приклад дошки канбан від trello.com

Використання цього інструменту у застосунку несе за собою зручне керування завданнями та відслідковування їх статусу. Якщо ж в проекті ця методологія не застосована, то така дошка може стати місцем для групування одноосібних завдань за загальними спільними властивостями, а не статусом виконання.

Використання методології Scrum передбачає розбиття всього робочого процесу на спринти. Вони є короткотривалими проміжками часу, в межах яких мають бути виконаними певні дії. У застосунках це реалізується за допомогою розбиття загального проекту на менші підпроекти, функціонал в яких повністю повторює функціонал загального.

Також діджиталізація цієї методології переважно несе за собою встановлення конкретного порядку виконання завдань чи настання подій. З цього слідує, що



Використання діаграми Ганта під час організації проектної діяльності дає керівнику чітке розуміння щодо тривалості цілого проекту і його підзавдань, можливостей щодо збільшення чи зменшення часових витрат, загальної гнучкості і правильності планування.

Одним із популярних PMS застосунків є *Basecamp*. Він орієнтований в першу чергу на комунікацію в команді, реалізувавши в собі, окрім видачі завдань, загальний чат, невеликий форум для питань та періодичні автоматичні опитування учасників. Недоліком у цій програмі є відсутність можливості отримання загальної звітності, з чого слідує необхідність додаткових засобів під час керування проектом.

*Bitrix24* є ще одним застосунком, який сьогодні широко використовується в цій діяльності, проте несе в собі не лише стандартні засоби управління, а й інструменти для взаємодії з клієнтом, встановлення договорів та керування бізнесом.

*GanttPro* в свою чергу орієнтується не на взаємодії в команді, а на планування проекту керівником в рамках розподілу ресурсів та фінансування, календарного графіку, відслідковування завантаженості та іншого суміжного. В ньому не реалізована багатофункціональність, натомість він постачає менеджеру потужні інструменти для роботи під час керування проектом.

Оскільки для забезпечення взаємодії між учасниками проекту в рамках організації проектної діяльності, оптимальним є поєднання канбан дошки та діаграми Ганта, у застосунку, що створюється, буде використано канбан дошку із можливістю наступного впровадження діаграми Ганта.

### **1.3 PMS для освітніх закладів**

На даний момент існує велика кількість PMS, які призначені для керування освітніми закладами. Це пов'язано як із процесом діджиталізації, так і із масовим переведенням на дистанційне навчання у зв'язку з пандемією COVID-19, внаслідок чого ринок став наповнюватись відповідним програмним забезпеченням.

В освітніх закладах України вони ще не використовуються масово, водночас при використанні їх повний потенційний функціонал часто не розкривають, проте за кордоном ця практика набуває все більшого поширення і визнання, адже несе за собою значні переваги.

Система керування проектом, що орієнтована на керування навчальною діяльністю, містить в собі інструменти та засоби, що підлаштовані під особливості даної сфери. Всі застосунки цього спрямування можна розділити на дві категорії: ті, що призначені для керування навчальними дисциплінами в межах взаємодії викладача із студентом, та ті, що призначені для власне керування освітнім закладом і отримання відповідної звітності. Також є застосунки, які поєднують у собі ці дві властивості, проте їх використання переважно є платним, що є значним недоліком, зважаючи на наявність «частково обмежених за функціоналом» аналогів у вільному доступі, яким освітні заклади і віддають перевагу.

Наразі популярним додатком із першої категорії є *Google Classroom*. Він є сервісом Google, що призначений для поширення навчального матеріалу та завдань. Також він підтримує інтеграцію з іншими його сервісами: Disk, Meet тощо, що розширює його функціонал та можливості.

Взаємодія між учасниками в цьому застосунку відбувається у окремих групах (класах), до яких можна долучитись за спеціальним кодом або запрошенням. Всередині груп є дві основні ролі: викладачі та студенти.

Викладачі можуть опублікувати матеріал та завдання. За виконання завдань вони можуть виставляти оцінки, які можна переглядати у зведеній груповій таблиці. Особлива увага в цьому додатку приділяється комунікації між студентом та викладачем, що реалізується через приватні коментарі, коментарі до курсу, дописи в потоці та коментарі викладача до зданого матеріалу студента.

Перевага цього додатку над іншими полягає у його візуальній простоті та наявності мінімально необхідних інструментів для навчальної діяльності.

Застосунок *Edmodo* повністю наслідує Classroom у його функціоналі, проте розширює домашню сторінку із класами, підтягуючи на нього всю актуальну інформацію із тих класів, учасником яких є користувач, а також додаючи

загальнодоступні дописи, які дають інші користувачі. Таким чином Edmodo частково наслідуює особливості соціальної мережі.

Вдалою реалізацією щодо об'єднання всієї діяльності освітнього закладу в вебзастосунок є *My Class Campus*. Його метою є оцифрувати всі процеси, з якими стикається керівництво, та надати зручний інструмент для інших учасників цієї діяльності. Не зважаючи на багатофункціональність, кожен інструмент реалізований так, що користувач із будь-якою роллю зможе інтуїтивно його використовувати.

*My Class Campus* позиціонує себе як ERP (Enterprise Resource Planning) – системою для планування ресурсів підприємства. Значною відмінністю цього застосунку від попередніх є наявність інструменту для створення документів та їх шаблонів на основі наявних даних.

Іншим популярним застосунком, який орієнтований на отримання звітності щодо функціонування навчального закладу є *Eduwonka* (рис. 1.4). Необхідні доступні дані про освітню діяльність можуть отримувати всі охочі: керівники, викладачі, студенти, їх батьки.

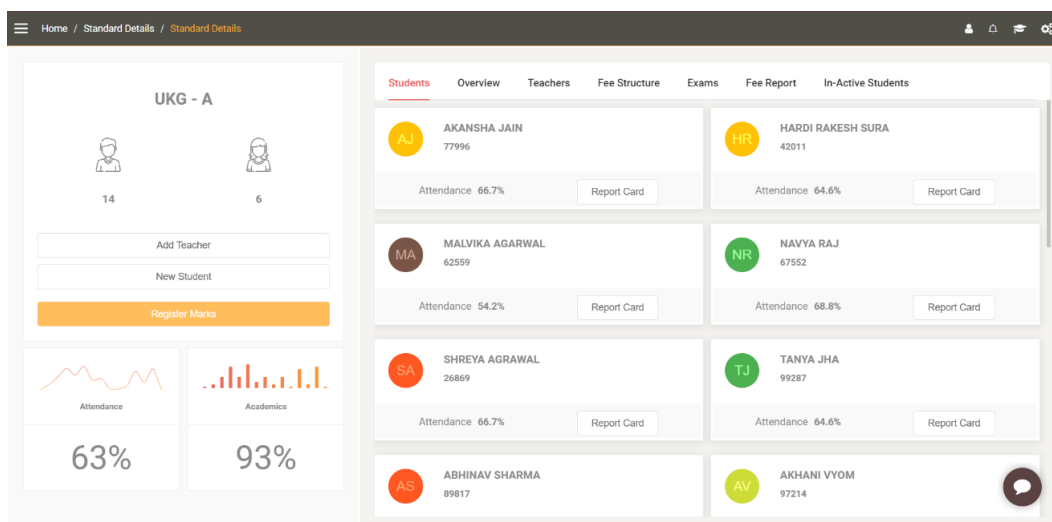


Рисунок 1.4 – Реалізація звітності групи студентів у eduwonka.com

При внесенні інформації уповноваженими людьми такої, як: відвідуваність студентів, успішність, нові зарахування, фінансові витрати та заборгованість,

система опрацьовує їх всі та видає у зручному вигляді на відповідних сторінках. Таким чином, використання цієї PMS дає переваги у структуризації та стандартизації всіх даних, яка в подальшому може бути записаною в архів чи переданою конкретній особі.

У результаті дослідження було виявлено велику кількість додатків для оптимальної взаємодії студента та викладача через видачу завдань, виставлення оцінок, спілкування й загальну організацію роботи. Проте кількість тих застосунків, які несуть в собі взаємодію між персоналом в рамках освітнього закладу, обмежена і зазвичай в них відсутня українська мова, що є незручним для більшості україномовних користувачів. Також наявні передбачають лише примітивну звітність або спілкування в чатах, що не несе вагомого функціонального навантаження.

Під час виконання практичної частини буде створено застосунок, який можна використати як для роботи викладачів із студентами, так і для організації взаємодії між різними групами працівників освітнього закладу. Також у кожного користувача буде можливість використати ці інструменти у власних цілях.

## 2 ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЕКТАМИ

### 2.1 Аналіз предметної області

Основним завданням застосунку для керування проектами є побудова оптимальної взаємодії між користувачами шляхом їх групування (команди та групи) та опису їх прав. Тому необхідно визначити форми та правила взаємодії всіх об'єктів, якими стануть і сутності *Команда* та *Група*.

Користувачі взаємодіють між собою у межах *команд* та *груп*.

Будь-хто може створити власну *команду* або приєднатися до існуючої за запрошенням, при чому на власність та членство не існує обмежень. При створенні користувач стає її *власником*, у якого є всі права на редагування (створення груп, видалення користувачів із команди та створення командних подій та завдань). Після цих дій можна запросити всіх людей, які мають бути членами цієї команди.

Запрошення відбувається за логіном користувача, який для кожного є унікальним. Підтвердивши таке запрошення, користувач приєднується до команди.

Наступним рівнем поділу команд є *групи*. Власник команди може створити безліч груп та має права на їх редагування (видалення, зміну керуючого та перейменування). Для кожної групи призначається *керуючий* із раніше запрошених користувачів, проте якщо власник не вкаже керуючого, то він сам ним стає. Він у будь-який момент може змінити керуючого групи. Керуючий отримує права обмеженого керування групою: запрошення до неї людей та розпорядження кінцевими виконавцями. Один користувач може бути у декількох групах – це дозволяє групувати людські ресурси як і за функціональними підрозділами, так і за проектами. Власник команди не може розпоряджатися ресурсами групи, це за нього робить керуючий. Це пов'язано із специфікою управління проектів, що дозволяє розбивати завдання на менші та призначати відповідальних за них, в роботу яких не варто вмішуватися, щоб не нашкодити процесу.

Останній рівень – це *кінцевий виконавець*. Він виконує поставленні йому доручення, не має прав на будь-які зміни в межах команд та груп і не може їх

покидати за власним бажанням. Виконавцю доступні лише перегляд подробиць завдань та склад людей у групах і командах.

На рисунку 2.1 представлена ієрархія основних об'єктів застосунку.

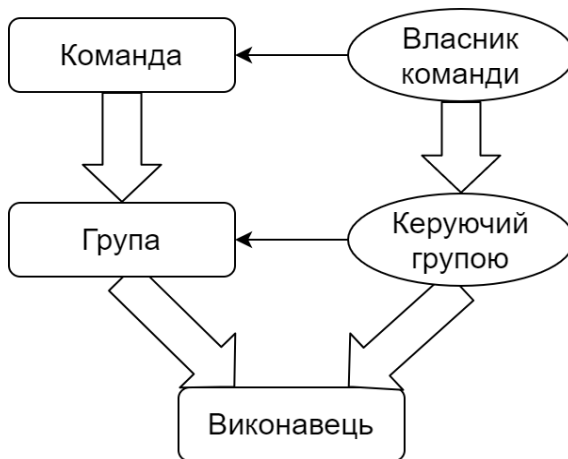


Рисунок 2.1 – Ієрархія об'єктів застосунку

У проєктній діяльності взаємодія зводиться до виконання окремих задач ресурсами (користувачами додатку), якими легко керувати та які є кінцевими одиницями роботи. У даному додатку існує два типи доручень – події та завдання (відповідно доцільно виділити дві окремі сутності *Подія* та *Завдання*).

*Подія* призначається одному або декільком кінцевим користувачам в межах команди або групи та не має дати початку, а лише дату настання. Проте власник команди здатен призначати події лише групам (насправді керуючому групи, який розділяє цю подію на менші події або завдання), а керуючий лише підпорядкованим йому виконавцям.

Творці подій і завдань здатні прикріплювати до них файли. В свою чергу виконавець також може прикріпити документи до доручення разом із відміткою про виконання.

Ще однією особливістю подій є те, що власник команди та керуючий групи у подробицях може бачити всіх користувачів, яким призначена подія, їх прикріплені файли та відмітку про виконання.



У свою чергу *завдання* призначається лише одному користувачеві, має початкову та кінцеву дати. Власник призначає завдання групі, а керуючий цієї групи ділить його на менші події та завдання між членами групи. Завдання відслідковуються за етапами виконання, тому поміщаються в *колонки*, описані методологією Kanban (для кожного рівня ієрархії вона своя), де можна їх відслідковувати.

*Колонки* прив'язуються до команд та груп, де формують Kanban-дошки. За такими дошками можна легко відслідковувати поточний етап виконання завдання. Проте, можна групувати завдання не етапами, а за певними ознаками, відповідно присвоївши ім'я колонкам та заблокувавши рух завдань виконавцями, якщо це необхідно. Це дозволяє гнучко використовувати дошки. Також в межах Команди існує дві дошки – в одній колонки передбачають відслідковування етапності виконання завдань, а в іншій – сортування завдань за групами, яким воно призначене. Це також допомагає слідкувати за ресурсами команд.

Окрім такого проектного способу використання додатку передбачено ще й особисте. Користувач може вести власні завдання та події і не звертати увагу на інший функціонал. У нього також є власна Kanban-дошка, якою він може розпоряджатися на свій розсуд. Якщо такого функціоналу йому буде мало, то він зможе створити особисту команду та керувати уявними людськими ресурсами й сортувати події та завдання за групами як забажає.

Таким чином людські ресурси у застосунку відображатимуться трьома сутностями – *команди*, *групи* та *кінцеві виконавці*, які дозволяють ефективно їх ділити, та розподіляти навантаження (рис. 2.2). Такий поділ реалізує форму поділу проектів на три рівні, коли власник ділить початкове завдання на менші та розподіляє їх між групами, а керуючі розділяють ці завдання на менші та призначають виконавцям. Не менш важливою складовою цієї системи є *завдання* та *події*, що дозволяють як виконувати завдання спільно, так і роздавати однакові завдання (події) певній групі людей відповідно. Це все забезпечує гнучкість цієї системи та дозволяє використовувати її як завгодно, та у будь-якій ситуації.

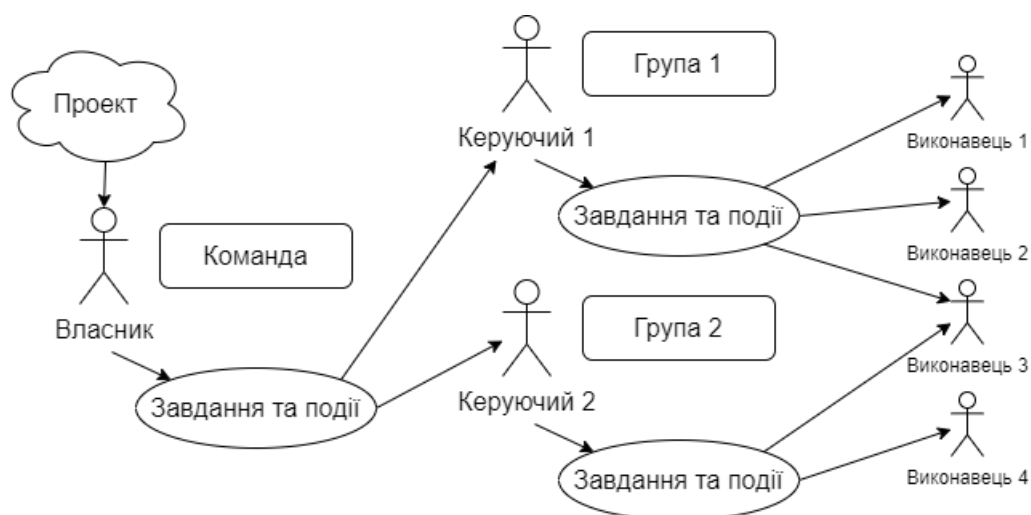


Рисунок 2.2 – Структура розподілу навантаження

## 2.2 Побудова концептуальної моделі бази даних

Відповідно до проаналізованих у попередньому підрозділі даних щодо опису предметної області, виділених основних інформаційних об'єктів та функціональних блоків моделі предметної області була спроектована концептуальна модель вебзастосунку (рис. 2.3).

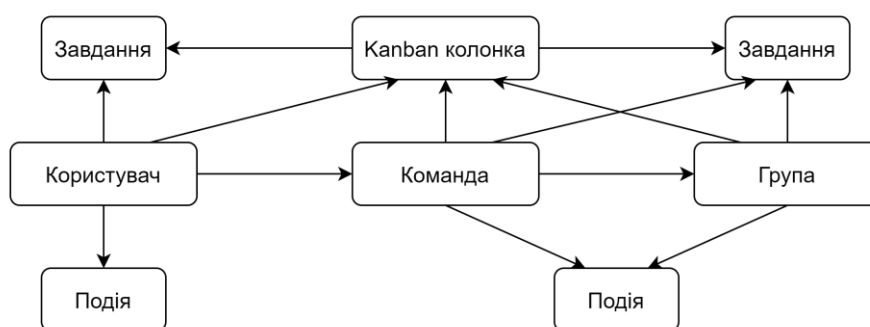


Рисунок 2.3 – Концептуальна модель бази даних застосунку

Створене уявлення про інформаційні об'єкти та функціональні блоки моделі предметної області разом із продуманою концептуальною моделлю інформаційної системи дають можливість приступити до наступного етапу, а саме проектування реляційної бази даних (далі БД).

На основі аналізу предметної області та концептуальної моделі можна

виділити такі специфічні моменти та основні обмеження:

1. Для збереження файлів, які прикріплюються до завдань та подій, потрібно було виділити місце в самій БД та створити поля типу BLOB (Binary Large Object / Бінарний великий об'єкт), які могли займати багато місця та збільшувати об'єми бази. Тому було вирішено розробити систему зберігання даних окремо на самому диску. Файли розділяються за ідентифікаторами (ID від identifier) об'єктів по репозиторіях із такими ж назвами. Це порушує нормалізацію БД та може вплинути на безпеку, проте дозволяє зберігати на диску файли без обмеження за розміром (що є перевагою в порівнянні з деякими конкурентами).

2. У цій структурі не доцільно виділяти дошку Kanban в окремий об'єкт, так як об'єкти Користувач, Команда, Група самі мають властивості цієї дошки (у додатку не передбачено створення декількох дошок). Проте колонки цієї дошки виділяються в окремі об'єкти.

3. Для можливості універсального використання дошок у межах груп дозволене блокування пересування завдань по дошці. Це дозволяє відслідковувати завдання не за етапами виконання, а, наприклад, сортувати їх за певними ознаками. При цьому втрачається можливість відслідковування за етапами, що може змусити створити нову дошку.

4. Для об'єктів Користувач, Команда, Група присутній зв'язок із Kanban-колонками, проте в схемі БД цих зв'язків немає. Насправді вони вказані неявно у властивостях самих об'єктів – для визначення порядку колонок вони відповідно записуються списком у це поле. Це полегшує структуру запитів до БД та пришвидшує їх обробку, проте порушує правила нормалізації баз. Через це на стороні сервера необхідно посилено перевіряти можливість внесення змін (що все ж виграє у швидкості обробки).

5. Для забезпечення поділу обов'язків для виконання поставлених цілей у застосунку передбачено обмеження сфер впливу користувачів та їх доступу до певних функцій. Власник команди створює групу та призначає їй керуючого. Власнику доступний повний контроль над налаштуваннями групи, проте керування її ресурсами (виконавці, завдання та події) йому заборонене. Це необхідно для

правильного розподілу обов'язків та попередження несанкціонованого змінення планів.

6. Також у процесі планування було вирішено розділити особисті завдання та події користувачів і завдання та події груп. Це зроблено для полегшення запитів – власник особистого завдання чи події і є їх виконавцем, а для командних і групових доручень – власниками у свою чергу є власник команди і керуючий групи, а виконавцями – їх підлеглі.

7. Для відображення виконання групою командного завдання до зв'язку команда-завдання додано властивість – група. Це пов'язано із тим, що один користувач може бути керівником багатьох груп і це унеможливорює чітке визначення того, якій групі призначено завдання.

### **2.3 Вибір системи управління базами даних та середовища розробки програмної частини**

При створенні додатку в навчальних цілях було вирішено використовувати полегшену реляційну систему керування БД SQLite. Доступ до неї здійснюється зверненням до єдиного файлу БД, що полегшує розгортку додатку на сервері.

*SQLite* втілена у вигляді бібліотеки, де реалізовано значну частину стандарту SQL-92 (стандарт SQL від 1992 року) (SQL, Structured query language – декларативна мова програмування (мова високого рівня, в якій програмістом не задається покроковий алгоритм рішення задачі, а деяким чином описується, що потрібно отримати як результат) для взаємодії користувача з базами даних). Програмний код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Із 2018 року SQLite, як й JSON та CSV, рекомендований Бібліотекою Конгресу США формат зберігання структурованого набору даних.

### Особливості SQLite:

- транзакції атомарні, послідовні, ізольовані і міцні (ACID) навіть після збоїв системи і збоїв живлення;
- встановлюється без конфігурації – не потребує ані установки, ані адміністрування;
- база даних зберігається в одному кросплатформовому файлі на диску;
- підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів;
- малий розмір коду: менше ніж 350Кб повністю налаштований і менш 200Кб з опущеними додатковими функціями;
- швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій;
- простий та легкий у використанні API;
- написана в ANSI C, включена прив'язка до TCL, доступні також прив'язки для десятків інших мов;
- доступний як єдиний файл програмного коду на ANSI C, який можна легко вставити в інший проект;
- автономність: немає зовнішніх залежностей;
- багатоплатформність: підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE), легко переноситься на інші системи;
- серцевий код вільно поширюється;
- поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

Візуальний інструмент для контролю БД не мав значення, тому було обрано полегшений додаток DB Browser for SQLite. Також для легкого відображення візуальної схеми БД було вибрано додаток DBeaver через відсутність такої функції у DB Browser for SQLite.

*DB Browser for SQLite* – це легкий багатоплатформний додаток із відкритим кодом, який дозволяє легко та швидко переглядати структуру та інформацію в БД. Також у додатку доступні функції створення та редагування баз, а ще у ньому

можна зручно писати SQL скрипти для їх тестування завдяки підсвітці синтаксису та виведення помилок.

*DBeaver* – SQL клієнт та інструмент управління базами даних. DBeaver має функції доповнення коду та підсвічування синтаксису, що забезпечує краще сприйняття. Він забезпечує архітектуру плагінів, яка дозволяє змінювати більшу частину роботи програми для підтримки специфічних для баз даних функцій, незалежних від бази даних.

DBeaver випускається в двох версіях DBeaver Community Edition (DBeaver CE) та DBeaver Enterprise Edition (DBeaver EE). DBeaver CE безкоштовний комп'ютерний додаток, написаний на Java, має відкритий вихідний код, поширюється під ліцензією Apache. DBeaver EE є платним та має розширені можливості – робота з noSQL (MongoDB і т. п.) базами даних.

DBeaver є кросплатформовим інструментом, який працює на платформах, що підтримуються Eclipse (Windows, Linux, MacOS X, Solaris).

Як мову програмування для серверної частини було обрано Python, так як вона широко використовується у цих цілях. Також вагомим фактом була наявність попереднього досвіду роботи з цією мовою.

*Python* – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією (змінні та об'єкти не прив'язані строго до свого типу). Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. У мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);

- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв’язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;
- зручний для розв’язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

*Flask* – мікрофреймворк для веб-додатків, створений з використанням Python. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Поширюється відповідно до умов ліцензії BSD.

Flask називається мікрофреймворком, оскільки не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широкоживані функції за допомогою сторонніх бібліотек. Однак, Flask має підтримку розширень, які надають додаткові властивості таким чином, наче вони були доступні у ньому із самого початку. Існують розширення для встановлення об’єктно-реляційних зв’язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку. Розширення оновлюються частіше, ніж базовий код.

Властивості Flask:

- містить сервер для розробки та відлагоджувач;
- має вбудовану підтримку юніт-тестів;
- управляє запитами RESTful (підхід до архітектури мережеских протоколів);
- використовує шаблони Jinja2 (рушій шаблонів для мови програмування Python);
- має підтримку безпечних куків (сесії на стороні клієнта);
- 100% відповідність WSGI 1.0 (стандарт взаємодії між Python-програмою,

яка виконується на стороні сервера, і самим веб-сервером);

- має докладну документацію;
- має розширення для забезпечення бажаної поведінки.

*HTML* (англ. HyperText Markup Language – мова розмітки гіпертексту) – це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет. Браузери отримують HTML-документи з вебсервера або з локального сховища даних й перетворюють документи в мультимедійні вебсторінки. HTML описує структуру вебсторінки семантично і визначає зовнішній вигляд вебсторінки.

*CSS* (англ. Cascading Style Sheets, укр. Каскадні таблиці стилів) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript.

Найчастіше CSS використовують для візуального відображення сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

*JavaScript* (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Середовищем розробки програмної частини було обрано Visual Studio Code (далі VS Code).



*VS Code* – засіб для створення, редагування та відлагодження сучасних веб-застосунків і програм. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим крос-платформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js.

Редактор містить вбудований відлагоджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Середовище підтримує всі сучасні мови програмування.

Основною перевагою середовища є те, що воно не прив'язане до конкретної мови програмування, а гнучко налаштовується. У додатку є вбудований магазин із розширеннями, де розробники мов та спільнота може їх публікувати. Це дозволяє для себе обрати оптимальний набір розширень, що полегшують роботу.

Ще однією перевагою є те, що в додаток вбудована система контролю версій Git, котра полегшує та пришвидшує роботу з ним.

Додаток емулює рідну для системи консоль, хоча і це за потреби можна налаштувати.

Для спільної роботи необхідний інструмент для обміну наробками, об'єднання їх та контролю процесу розробки. Для цих цілей використовується технологія Git.

*Git* – розподілена система керування версіями файлів та спільної роботи. Проект створив Лінус Торвальдс для керування розробкою ядра Linux, а сьогодні підтримується Джуніо Хамано. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної

розробки, які базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Для роботи із Git використовувався вебсервіс *GitHub*, який є найбільшим та безкоштовним.

Увесь процес клопіткого та детального планування, вибору технологій, сервісів та програмного забезпечення дозволив приступити до розробки. Це допомогло уникнути маси помилок та проблем у подальшій розробці, що дало вести ефективну роботу над застосунком. Також через вільну форму поширення всього програмного забезпечення, грошові витрати були зведені до мінімуму, не потрібно було звертатися до піратських матеріалів та вирішувати проблеми, що могли виникнути через ліцензії, а через популярність та сучасність усіх згаданих вище технологій можна сказати, що додаток є технічно актуальним.

## 3 РЕАЛІЗАЦІЯ ПРОЕКТУ

### 3.1 Створення логічної моделі бази даних

За допомогою глибокого аналізу предметної області, дослідження процесу керування проектами та дослідження конкурентів шляхом вивчення, експериментів та проб було написано схему/скрипт БД SQLite.

Для кращого розуміння структури БД її зручно розділити та розглядати зв'язки окремих елементів.

Основою системи є *Користувач* (user), поля якого описані в таблиці users (рис. 3.1). На схемі видно, що користувач належить до команд та груп, які пов'язані проміжними таблицями (що реалізує зв'язок «багато-до-багатьох»). Проте користувачі ще й напряду є властивостями команд та груп через те, що у них відповідно є власники та керуючі, які є єдиними у цих об'єктів (реалізовує зв'язок «один-до-одного»).

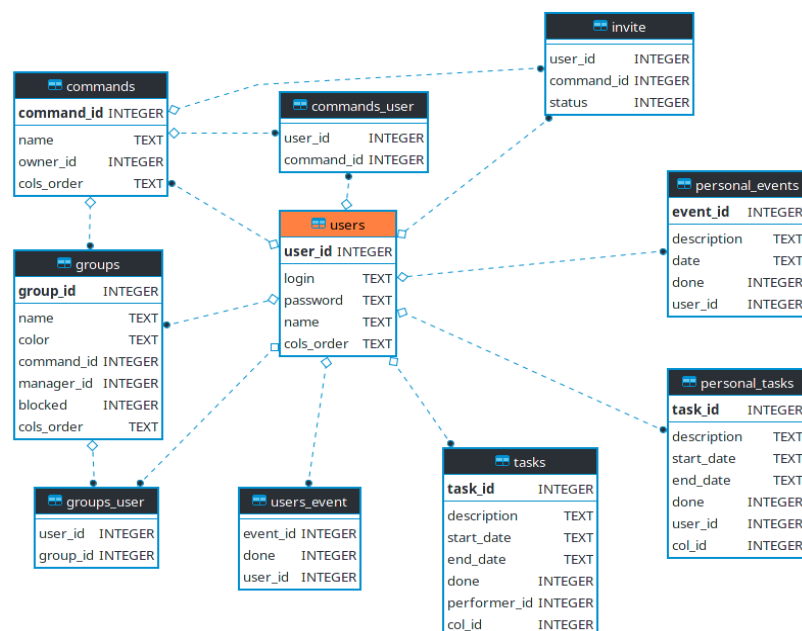


Рисунок 3.1 – Структура та зв'язки таблиці users

Також для зручності створені окремі таблиці із особистими Завданнями та Подіями користувача (personal\_tasks та personal\_events). Це дозволило їх відділити

від завдань команд та груп тому, що це дві подібні, але різні сутності, з якими зручно оперувати різними способами.

Першим рівнем ієрархії командної взаємодії у додатку є *Команда*, поля якої описано в таблиці `commands` (рис. 3.2). На схемі видно, що команди пов'язані із користувачами, які до неї входять (включаючи самого власника). Також вони пов'язані із завданнями та подіями, які їм належать. У кожній команді є єдиний власник. Це реалізовано прямим зв'язком користувачів із командою.

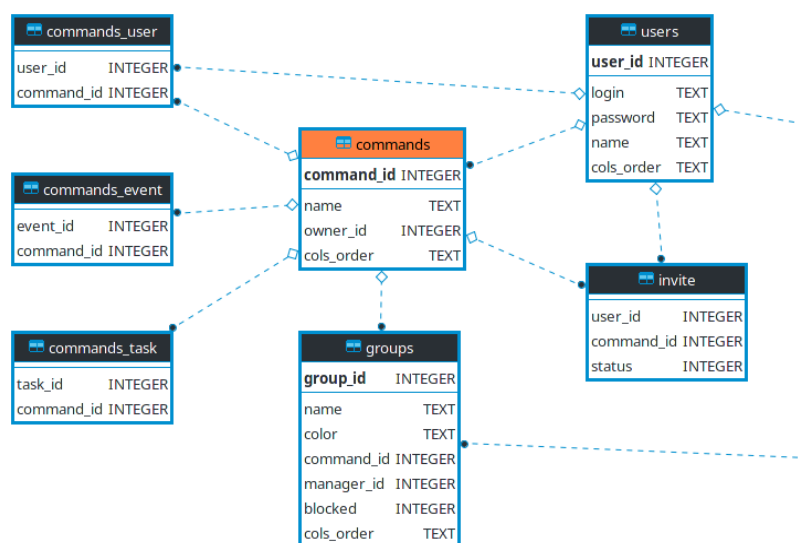


Рисунок 3.2 – Структура та зв'язки таблиці `commands`

*Запрошення* (`invite`) повністю пов'язані із командами. Вони пов'язують користувачів, яких запрошено, із командами, котрі запрошують. Статус (`status`) показує чи запрошення активне – 1, чи відхилене – 0. Якщо користувач приймає його, або власник команди видаляє його – запис із таблиці видаляється та проводяться необхідні дії.

Другим рівнем ієрархії є *Група*, поля якої описано в таблиці `groups` (рис. 3.3). На схемі видно, що група пов'язана із користувачами, які до неї входять (разом із керуючим та власником команди), завданнями та подіями, які їм належать. Групи пов'язані із командами напряму тому, що одна група може належати одній команді. Також у групі є керуючий, який також напряму пов'язаний з нею через те, що він єдиний для кожної групи.

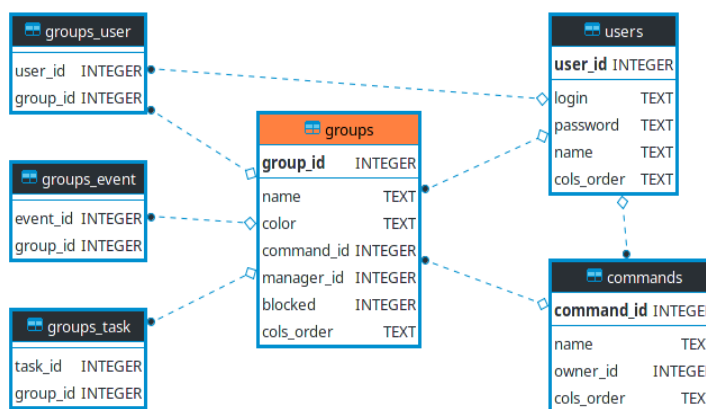


Рисунок 3.3 – Структура та зв'язки таблиці groups

*Колонки*, що описані в таблиці cols (рис. 3.4), за допомогою яких реалізується методологія Kanban, винесено в окрему сутність. Вони напряду пов'язані із завданнями команд і груп та особистими завданнями. Проте на схемі не відображено їх зв'язок із користувачами, командами та групами. Реалізацію цього зв'язку описано у розділі 2.2.

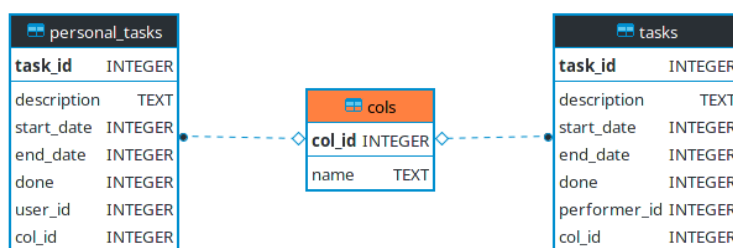


Рисунок 3.4 – Структура та зв'язки таблиці cols

*Завдання* описано в таблиці tasks (рис. 3.5). Вони напряду пов'язані із користувачами, але, на відміну від особистих завдань, користувач є лише виконавцем. Власником такого завдання є власник або керівник команди чи групи відповідно, яким воно належить.

У проміжній таблиці commands\_task додатково є зв'язок із групою для реалізації призначення завдань групі, що описано в розділі 2.2.

*Події* описано в таблиці events (рис. 3.6). Як і завдання, вони мають зв'язок із командами та групами. Проте особливістю є те, що у подій немає одного виконавця. Тому для зв'язку із користувачами є окрема проміжна таблиця, в якій

також і зазначено статус виконання (done) події.

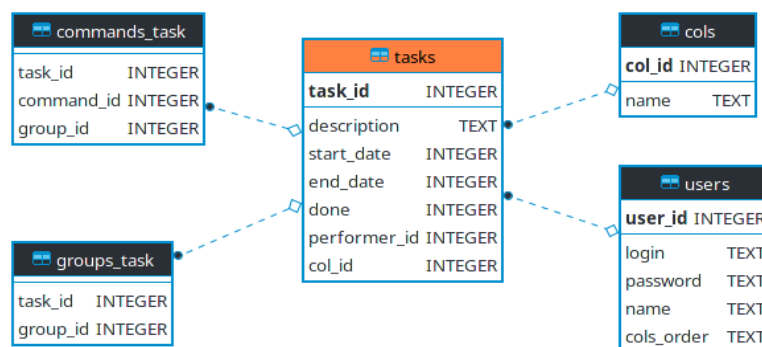


Рисунок 3.5 – Структура та зв'язки таблиці tasks

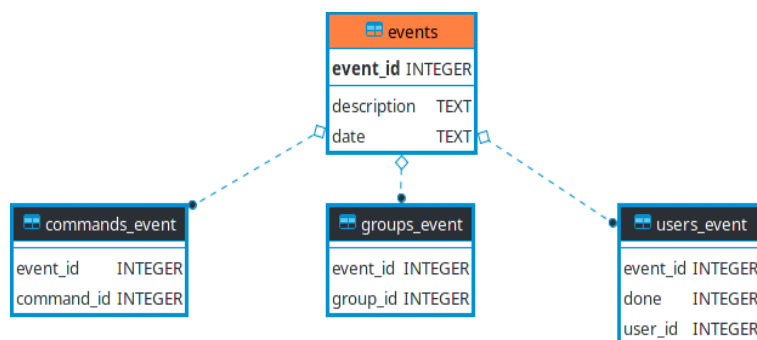


Рисунок 3.6 – Структура та зв'язки таблиці events

### 3.2 Розробка дизайну сторінок застосунку

При розробці дизайну застосунку основним завданням було створення простого та інтуїтивного інтерфейсу (рис. 3.7). Тому потрібно було уникати нагромадження функціоналу на одній сторінці та розділяти його за логічними блоками. Всі елементи великі, тому у них легко орієнтуватися. Активні елементи – посилання та кнопки – реагують на наведення, що поліпшує інтерактивність інтерфейсу.

Для самого інтерфейсу обрано спокійні блакитні кольори (рис. 3.8) та округлі форми елементів. Для дизайну застосунку обрано принцип матеріального дизайну, котрий популяризує Google. Так застосунок виглядає сучасно та дозволяє ще краще виділити елементи інтерфейсу та покращити рівень його «інтуїтивності».

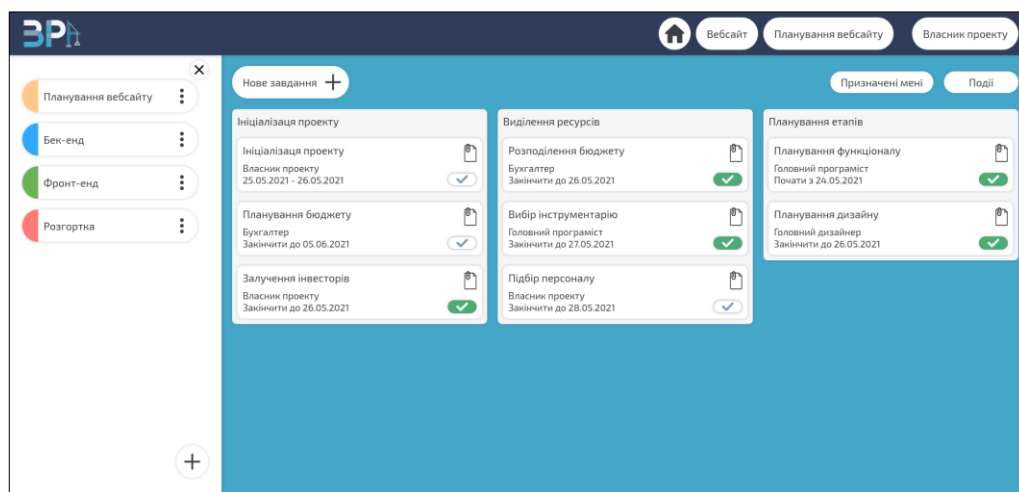


Рисунок 3.7 – Приклад дизайну інтерфейсу



Рисунок 3.8 – Основні та додаткові кольори інтерфейсу

Для формування інтерфейсу застосовується стандартний шаблонізатор Python – Jinja2. Він дозволяє використовувати сукупність однакових, власноруч написаних шаблонів інтерфейсу для формування подібних сторінок.

Для всіх сторінок існує єдиний шаблон із основними тегами HTML.

Для сторінок з основним функціоналом існує шаблон із блакитним фоном, шапкою та боковим меню, що можна ховати.

Для сторінок користувача, команд та груп зроблено три окремі шаблони із загальним функціоналом, а від них наслідуються вже окремі, закінчені сторінки.

Сторінки із завданнями та подіями (рис. 3.9) хоч і мають однакові принципи побудови, проте для кожної сутності відрізняються. Наприклад, на сторінці команд та груп групи у лівому меню виділені кольоровими індикаторами для їх візуального виділення. До того ж, на сторінці команд кольором виділені і завдання, що відображає їх належність окремим групам.

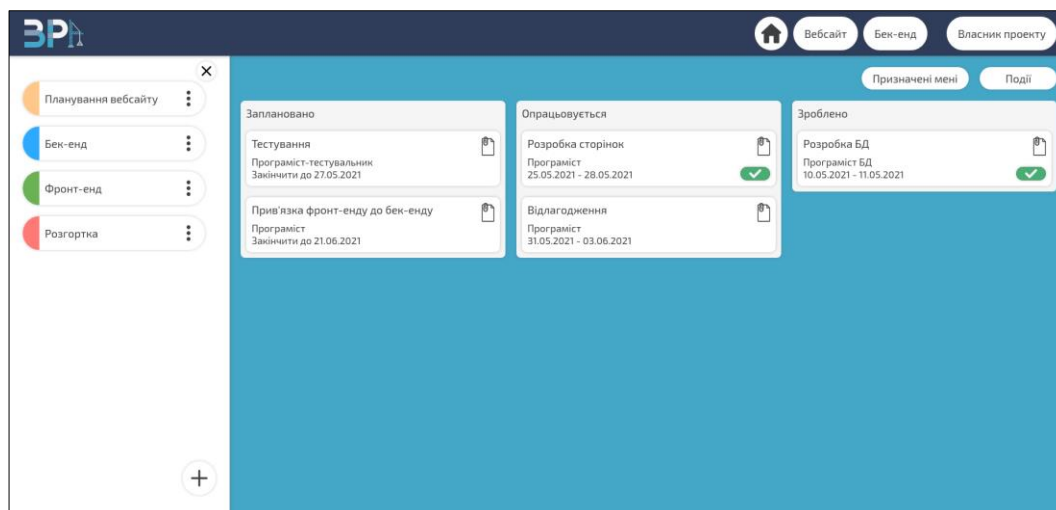


Рисунок 3.9 – Приклад сторінки із завданнями

Особливістю розробки дизайну сторінок налаштувань є те, що існує два шаблони для їх написання. Один шаблон дає просте біле тло для розміщення елементів, інший – біле тло із лівою панеллю для розміщення кнопок переходу між сторінками налаштувань.

Така особливість пояснюється тим, що, до прикладу, для відображення налаштувань завдання достатньо лише одної сторінки – сторінки із її редагуванням, в той час для налаштування команди необхідно кілька сторінок: редагування команди, редагування учасників та редагування Kanban-колонок. Тому для відображення редагування завдання буде використано один простий шаблон, а для команди – розширений, де на лівій панелі знаходитимуться посилання на перелічені вище сторінки, які стосуються редагування цієї сутності.

### 3.3 Реалізація програмної частини застосунку

Зважаючи на особливості роботи із фреймворком Flask, а також виходячи із зручності структура проекту є наступною:

- тека `db_file` містить схему БД та власне БД;
- тека `static` – допоміжні елементи для формування сторінки: стилі CSS та шрифти (папка `css`), зображення (папка `ico`), файли JavaScript (папка `script`);
- тека `templates` – HTML шаблони сторінок, написані за допомогою двигуна



шаблонів Jinja2;

- файл `__init__.py` – необхідний для того, щоб загальний вміст теки проекту розглядався компілятором Python як один цільний модуль;
- файл `config.py` – містить дані конфігурації, що використовуються під час компіляції застосунку;
- файл `create_db.py` – запускається для створення БД, дані про яку розміщені у теці `db_file` під назвою `shema.sql`;
- файл `db_work.py` – містить клас `db_work`, котрий призначений для роботи із базою даних;
- файл `main.py` – містить функції-обробники сторінок вебзастосунку, а також дані для його компіляції, призначений для обробки всіх запитів до серверу;
- файл `requirements` – містить дані про всі використані інструменти під розробки, а також їх версії;
- файл `wtform.py` – містить класи всіх форм застосунку, що написані за допомогою бібліотеки WTForms.

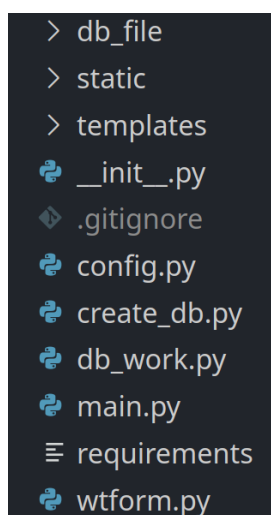


Рисунок 3.10 – Файлова структура додатку

Для зручності читання коду, а також розширення наявного функціоналу весь код супроводжується коментарями та поділений на логічні секції. До прикладу, у файлі стилів CSS усі стилі логічно посортовані у групи за їх призначенням, а також на початку кожної групи написаний коментар із її назвою. Таке сортування за

групами можна віднести і до файлів `main.py` та `db_work.py`, що суттєво полегшує ознайомлення із ними.

Крім того, всі функції та методи супроводжують документ-рядки, котрі у більшості сучасних середовищах програмування відображаються при наведенні на виклик функції чи методу.

При проектуванні основних сторінок користувача, команд та груп (рис. 3.11) потрібно було написати обробники цих сторінок та функції взаємодії із БД. Тому було дотримано декількох принципів для ефективності та зручності коду:

- ефективна обробка інформації, що забезпечує швидке завантаження сторінок;
- повторне використання коду (принцип програмування DRY / Don't repeat yourself / не повторяй своє), що у цьому випадку дозволяло легко змінювати код який використовується у багатьох місцях;
- використання простих конструкцій мови, що дозволяло зменшити об'єм та зрозумілість коду;
- логічне розділення функціоналу.

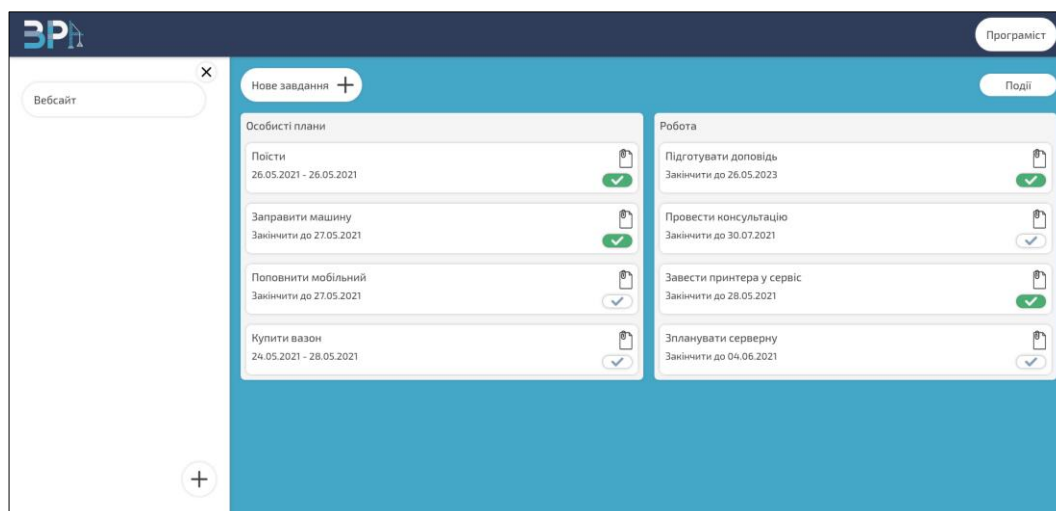


Рисунок 3.11 – Приклад основної сторінки вебзастосунку

Можна виділити такі основні етапи розробки окремих сторінок:

- розробка шаблонів сторінок;
- написання класів форм;

- написання методів для роботи із базою даних;
- написання функцій для обробки сторінок.

Для реалізації основних сторінок також було розроблено декілька JavaScript скриптів. Вони як і реалізують просту взаємодію з інтерфейсом, так і асинхронну роботу із додатком. Так для зміни статусу *завдань* та *подій* розроблено скрипти, які дозволяють без перевантаження сторінок відправляти цю інформацію на сервер. У свою чергу на стороні користувача просто змінюється інтерфейс.

Також за допомогою JavaScript було власноруч реалізовано технологію drag-and-drop для перетягування завдань між колонками. При цій дії на сервер відправляється ідентифікатор завдання та нової колонки, а у користувача блок завдання просто переміщується у нову колонку.

Однією із задач реалізації програмної частини була розробка сторінок для створення та редагування сутностей (рис 3.12).

Рисунок 3.12 – Приклад сторінки редагування сутностей

Першочергово було створено два батьківські шаблони: `setting.html` та `setting_expand.html`. Вони, як і решта шаблонів, наслідують `base.html` та передбачають, що будуть наслідуваними сторінками налаштувань. Назви шаблонів власне сторінок починаються із «`add`», якщо це сторінка додачі об'єкту, та «`edit`», якщо редагування.

Класи форм додавання та редагування сутностей, а також відповідні методи

для звернення до бази даних описані у файлах `wtform.py` та `db_work.py`. Їх назви починаються із «add» та «edit» відповідно.

Функції для обробки сторінок написані у файлі `main.py`. Вони отримують необхідну інформацію для відображення сторінки та передають її клієнту. В них же знаходяться і обробники форм, які містять у собі сторінки.

Вартими уваги є форми видалення сутностей – вони з’являються в окремому вікні, коли відбувається натискання кнопки «Видалити» на окремій сторінці. Появу та зникнення цього вікна реалізовано за допомогою JavaScript.

Було спроектовано початкову сторінку (рис. 3.13), сторінку входу та реєстрації, для яких розроблено окремі шаблони.

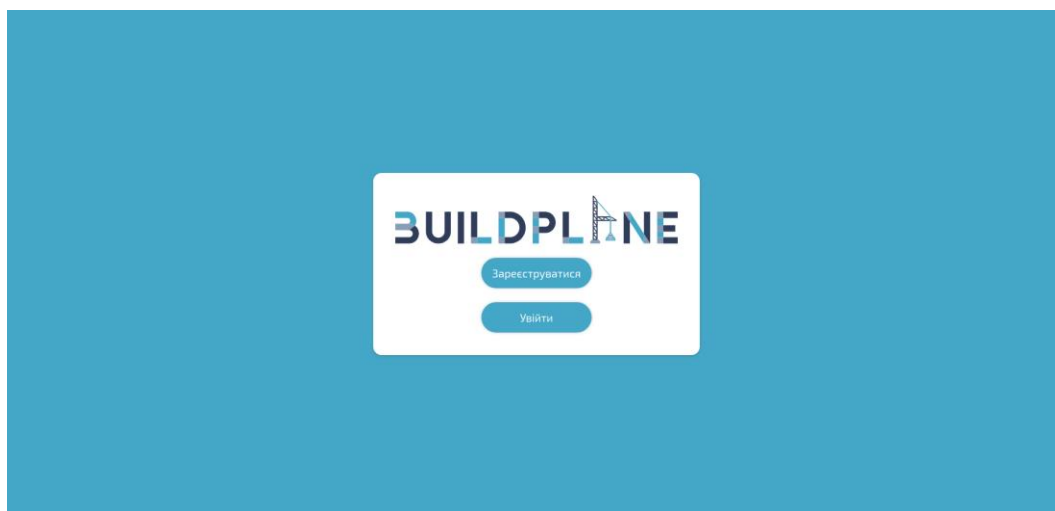


Рисунок 3.13 – Початкова сторінка

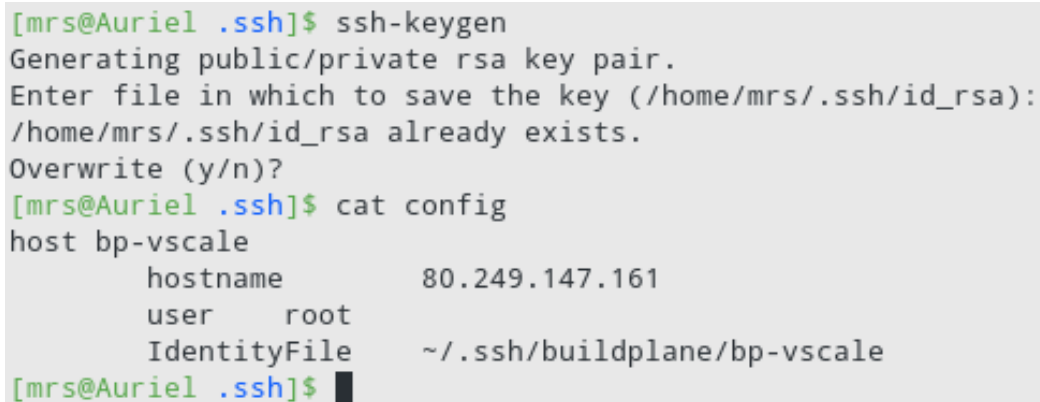
Також у вебзастосунку реалізовано певний рівень захисту інформації. Наприклад, неавторизований користувач не може користувати додатком поки не увійде у систему. При вході його ідентифікатор зберігається на сервері в базі даних Redis, що не дає користувачу якимось чином змінювати цей ідентифікатор. Користувач, що не належить до якоїсь команди чи групи, не зможе потрапити на їх сторінки. Для захисту користувацьких паролів проходить їх хешування при записі, що практично внеможливіює їх розшифрування.

### 3.4 Розгортання вебзастосунку на сервері

Для полегшення розгортки на сервері було прийняте рішення використовувати технологію Docker (програмне забезпечення для автоматизації розгортки й управління додатками у віртуальних середовищах). Це дозволяє не перейматися налаштуванням мережі та розгортати застосунок на різних серверах однаковим способом.

Для початку було орендовано VPS (Virtual Dedicated Server / віртуальний виділений сервер) на платформі vscale через низьку ціну та надійність. Створено віртуальний сервер на дистрибутиві Linux Ubuntu.

Наступним кроком було створення SSH ключа для під'єднання до сервера по захищеному протоколу SSH (Secure Shell Protocol) та налаштування з'єднання (рис. 3.14). Після створення приватного та публічного ключа публічний був доданий у профіль на хостингу.



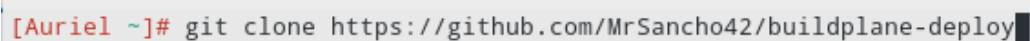
```
[mrs@Auriel .ssh]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mrs/.ssh/id_rsa):
/home/mrs/.ssh/id_rsa already exists.
Overwrite (y/n)?
[mrs@Auriel .ssh]$ cat config
host bp-vscales
    hostname      80.249.147.161
    user          root
    IdentityFile  ~/.ssh/buildplane/bp-vscales
[mrs@Auriel .ssh]$
```

Рисунок 3.14 – Генерація SSH ключа та конфігурування SSH з'єднання

Після цього було опубліковано застосунок на новому публічному репозиторію на GitHub. У застосунку заздалегідь створено файли Dockerfile та docker-compose.yaml необхідні для запуску контейнерів.

Після підготовчих етапів розпочалась розгортка застосунку. Відбулося підключення до віддаленого серверу. Після цього на нього було встановлено docker, docker-compose та git. Наступний крок – у домашню папку користувача root

було скопійовано застосунок із GitHub (рис. 3.15).

A terminal window with a light gray background. The prompt is [Auriel ~]#. The command being entered is git clone https://github.com/MrSancho42/buildplane-deploy. The cursor is at the end of the command.

```
[Auriel ~]# git clone https://github.com/MrSancho42/buildplane-deploy
```

Рисунок 3.15 – Копіювання застосунку з GitHub

Після копіювання застосунку було створено базу даних запуском скрипта create\_db.py.

Завершальним етапом розгортки є запуск команди docker-compose up -d.

## 4 АНАЛІЗ РЕЗУЛЬТАТІВ

### 4.1 Тестування вебзастосунку для керування проектною діяльністю

Тестування вебзастосунку виконувалось власне під час розробки після завершення певних логічних самостійних блоків а також після повного завершення розробки. Таким чином, виправляючи невеликі помилки одразу під час розробки, було зекономлено час на загальне тестування та виправлення тих несправностей, котрі були виявлені під час нього.

Під час тестування застосунку у різних браузерах була виявлена особливість тегу HTML dialog – він не сприймається браузером Firefox. Тому, коли у застосунку виникає діалогове вікно (переважно воно викликається для повторного підтвердження незворотньої дії), то у більшості браузерів воно відображатиметься окремим вікном поверх наявної сторінки на сірому тлі, а у Firefox – прямо на сторінці серед інших елементів.

Цю несправність можна виправити, якщо описувати діалогове вікно без застосування даного тегу і прописувати весь його функціонал за допомогою JavaScript, проте за браком часу на освоєння нових способів, особливість відображення цього елемента у різних браузерах була збережена та стилі діалогового вікна підібрані так, що навіть поміж інших елементів зберігається його естетичний вигляд та інтуїтивна зрозумілість.

Під час написання форм була виявлена проблема – за допомогою фреймворку Flask, а також бібліотеки wtforms неможливо створити обробник двох форм під однією функцією. Це пов'язано із особливістю написання методу класу форм `validate_on_submit`, котрий використовується для перевірки того, що форма була надіслана та всі її дані дійсні. Після цього мала б відбуватись обробка отриманих даних; натомість із кількома формами в одній функції цей метод конфліктує і після проходження перевірки надсилання форми перевірку щодо дійсності даних не пропускає. Тому для обробки кількох форм, окрім основної функції обробки сторінки, для обробки кожної форми створюється окрема функція. Так, до

прикладу, сторінку налаштувань команди описує функція `settings_command`, окремо форми редагування та видалення команди – `edit_command` та `del_command`. За допомогою декоратора `route` ці функції доступні за певним шляхом у застосунку – посилання на цей шлях є у описі форм в `html` документах. Проте за допомогою браузера чи інших інструментів обробити форму за допомогою шляху без натиснення кнопки підтвердження не вдасться, адже така небажана можливість була передбачена та заблокована.

Також фатальною помилкою, котра була виявлена під час тестування, стала «`sqlite3.programmingerror: recursive use of cursors not allowed.`». Вона виникає під час звернення до БД на сторінці, де використовується JS скрипт із функцією `fetch()` для звернення до обробника сторінки у файлі `main.py`. Причиною даної помилки є неможливість СКБД `SQLite` працювати у багатопотоковому режимі, а асинхронна функція `fetch`, вочевидь, на певній стадії виконання включає багатопотоковість.

Особливістю цієї помилки також є те, що її поява є необов'язковою під час виконання дій, котрі її викликають – часом вона виникає щоразу, часом через раз; також під час одного із тестувань було проведено більше ста дій, котрі б мали викликати цю помилку, проте застосунок працював повністю справно.

Оскільки можливості використаних під час розробки технологій не дозволяють відмовитись від функції `fetch`, було прийняте наступне рішення – оскільки навіть тоді, коли помилка виникає, всі необхідні дії із базою даних виконуються успішно у будь-якому випадку, в обробнику сторінки за допомогою конструкції `try-except` під час виникнення даної помилки переадресовувати користувача на ту ж сторінку. Таким чином, логіка програми є наступною: іде звернення до БД, запит до БД виконується успішно, виникає помилка, сторінка користувача перезавантажується. Даний метод використаний скрізь, де під час тестування була виявлена описана вище проблема.

При розробці інтерфейсу було створено ліве меню, яке при потребі можна приховати. Через прагнення розробити інтерфейс найпростішим способом та без застосування `JavaScript`, приховування лівого меню було реалізовано через `input checkbox`. Проте через неможливість впливати на будь-який елемент на сторінці



з'являлися неприємні обмеження інтерфейсу. Тому відбувся перехід на скрипт для зміни інтерфейсу.

У застосунку передбачається призначення завдань групі для ефективного управління. Проте при розробці БД було допущено помилку, через що завдання неможливо було призначити окремій групі, що при визначенні групи, якій призначалося завдання за її керівником, могло викликати помилку. Це пов'язано із тим, що у декількох груп може бути один і той же керівник.

Для виправлення цієї помилки було додано додаткове поле проміжній таблиці `commands_task – group_id`, що вказує на групу якій призначено завдання.

При розробці сторінок потрібно було забезпечити обмеження доступу неавторизованих користувачів та доступ до команд та груп, учасником яких користувач не є. Це реалізувалося простою перевіркою перебування користувача в сесії та відповідністю записів у БД для команд та груп. Якщо користувач не належить до команди або групи, йому відображається помилка відмови у доступі (HTTP Error 403: Forbidden). А коли користувач не авторизований (його немає у сесії) його перенаправляє на сторінку входу (/login).

Проте не був передбачений виняток при завантаженні головної сторінки й сторінок реєстрації та входу. Це призводило до постійного зациклювання завантаження сторінки входу. Після створення винятку для цих сторінок помилка зникла. Однак це породило нову помилку – якщо користувач неавторизований, він може без проблем перебувати на головній сторінці та на сторінках входу та реєстрації, проте файли із зображеннями та стилями не завантажувалися. Виявилося, що ці файли завантажуються із сервера окремими запитами, які згаданим вище винятком перенаправлялися на сторінку входу, що не дозволяло їх отримати. Це вирішилося створенням ще однієї перевірки на вміст у запиті слова `static` (назва директорії, в якій містяться всі файли дизайну інтерфейсу). Із вирішенням цих помилок вдалося полегшити завантаження сторінок, ігноруванням згаданих вище перевірок.

При розробці застосунку, виявилося, що зберігання копії важливої інформації на стороні клієнта у сесіях (захищені cookie) є ненадійною та

недоречною практикою. Тому було вирішено перейти на сесії на стороні сервера із використанням Redis (розподілене сховище пар ключ-значення, які зберігаються в оперативній пам'яті).

## **4.2 Створення супровідної документації до застосунку**

Після завершення розробки для спрощення експлуатації, а також наявності довідкових матеріалів була створена супровідна документація у формі методичних вказівок. У ній зібрана найважливіша теоретична інформація, а також приклади та пояснення до практичного використання застосунку.

Методичні вказівки складаються із таких пунктів:

1. «Проектна діяльність» – пояснення суті проекту, теоретичні відомості щодо нього та роль застосунків в управлінні проектною діяльністю.
2. «Концепція застосунку» – опис логіки застосунку, його основних сутностей та ієрархії.
3. «Експлуатація застосунку на прикладі» – демонстрація функціоналу застосунку на прикладі «вживаної» сторінки користувача.
4. «Створення та редагування об'єктів» – демонстрація створення та редагування об'єктів із поясненням необхідних дій для цього.

Ці методичні вказівки будуть передані ДВНЗ «Нововолинський електромеханічний коледж» у циклову комісію комп'ютерних дисциплін для використання під час експлуатації. Не зважаючи на їх невеликий об'єм, вони є самодостатніми та несуть всю необхідну інформацію для того, щоб потенційний користувач міг ознайомитись із методами практичного використання застосунку, а також оцінити його потенціал на основі описаних можливостей та підходів використання.

## 5 ЕКОНОМІЧНА ЧАСТИНА

Метою даного розділу є обґрунтування доцільності надання послуги щодо створення вебзастосунку.

### 5.1 Обчислення баланс робочого часу працівника

Ефективний фонд часу розраховується за формулою:

$$\Phi_{\text{ef}} = \Phi_{\text{ном}} \times h, \quad (5.1)$$

де  $\Phi_{\text{ном}}$  – номінальний фонд робочого часу працівників, год.

$h$  – плановий коефіцієнт витрат часу, приймається в межах 2-5%.

Припускаємо втрати часу 5 %.

Номінальний фонд часу визначається за формулою:

$$\Phi_{\text{ном}} = (D_{\text{к}} - D_{\text{св}} - D_{\text{в}} - D_{\text{від}} - D_{\text{п.св}}) \times t_{\text{зм1}} + D_{\text{п.св}} \times t_{\text{зм2}}, \quad (5.2)$$

де  $D_{\text{к}}$  – кількість календарних днів у році, дн;

$D_{\text{св}}$  – кількість святкових днів у році, дн;

$D_{\text{в}}$  – кількість вихідних днів у році, дн;

$D_{\text{від}}$  – середня кількість днів у році наданих робітнику під відпустку, дн;

$D_{\text{п.св}}$  – передсвяткові, дн;

$t_{\text{зм1}}, t_{\text{зм2}}$  – тривалість зміни у звичайні та передсвяткові дні, год.

Для розрахунку наведеної формули її дані визначаємо за 2021 р. і вони становлять:

дні календарні – 365;

вихідні дні – 104;

святкові дні – 11;

дні відпустки – 24;

тривалість зміни – 8 годин, у передсвятковий день – 7 годин.

Звідси фонд номінальний дорівнює:

$$\Phi_{\text{ном}} = (365 - 104 - 11 - 11 - 24) \times 8 + 11 \times 7 = 1797, \text{ год.}$$

Фонд ефективний становить:

$$\Phi_{\text{еф}} = 1797 \times 0,95 = 1707,15, \text{ год.}$$

## 5.2 Розрахунок трудомісткості робіт з надання послуги створення вебзастосунку

Спочатку визначають трудомісткість робіт щодо 1 послуги, тобто час за формулою:

$$T_{\text{од}} = \sum t_{\text{н/год}}. \quad (5.3)$$

де  $T_{\text{од}}$  – трудомісткість робіт щодо 1 послуги, н/год.;

$t_{\text{н/год}}$  – загальна сума норми часу згідно технологічного процесу, год.

Трудомісткість обчислюємо у таблиці 5.1.

Таблиця 5.1 – Трудомісткість робіт

Найменування операції	Час на 1 операцію, год.
Планування проекту	32
Проектування бази даних	72
Проектування вебсторінок	88
Створення концептуальної моделі програмної частини	80
Розробка вебсторінок	160
Розробка бекенду	184
Під'єднання сторінок та БД до програми	28
Розгортка застосунку на сервері	32
Відлагодження	40
<b>РАЗОМ</b>	<b>716</b>

Далі визначаємо загальну трудомісткість робіт, тобто всю кількість робочого часу, яку необхідно витратити на планову річну кількість послуг за формулою:

$$T_{\text{посл}} = T_{\text{од}} \times N, \quad (5.4)$$

де  $T_{\text{посл}}$  – загальна трудомісткість робіт, н/год;

$T_{\text{од}}$  – трудомісткість 1 послуги, н/год.;

$N$  – річна кількість планових послуг, од.

Приймаємо кількість послуг створення вебзастосунку – 7 од. в рік.

Звідси:

$$T_{\text{посл.}} = 716 \times 7 = 5012, \text{ н/год.}$$

Припускаємо, що даний вид роботи не є єдиним на підприємстві і становить 29,95% від решти наданих послуг.

Отже на весь обсяг:

$$T_{\text{заг}} = 5012 / 0,2995 = 16734,56, \text{ н/год.}$$

### 5.3 Розрахунок необхідної кількості працюючих

Основних працівників підприємства визначаємо з розрахунку планової трудомісткості всіх наданих послуг фірми протягом планового року за формулою:

$$Ч_{\text{осн}} = T_{\text{заг.}} / (\Phi_{\text{еф}} \times k_{\text{вн}}), \quad (5.5)$$

де  $Ч_{\text{осн}}$  – чисельність основних робітників, ос.;

$T_{\text{заг}}$  – загальна трудомісткість робіт на підприємстві, н/год;

$\Phi_{\text{еф}}$  – ефективний фонд, год.;

$k_{\text{вн}}$  – коефіцієнт виконання норм, що приймаються рівним від 1 до 1,2. У

нашому випадку припускаємо перевиконання на рівні 9%, звідси  $k_{\text{вн}} = 1,09$ .

Обчислюємо чисельність основних робітників для надання даної послуги:

$$Ч_{\text{осн}} = 16734,56 / (1707,15 \times 1,09) = 9, \text{ ос.}$$

Крім того, приймаємо інші категорії персоналу згідно штатного розкладу у розмірі:

1. Директор – 1 ос.
2. Бухгалтер – 1 ос.
3. Допоміжний робітник – 1 ос.
4. Прибиральник – 1 ос.

#### **5.4 Обчислення витрат на оплату праці**

Заробітна плата основних робітників визначається за відрядною заробітною платою за формулою:

$$ЗП_{\text{ор}} = \sum P_{\text{зв}} \times N, \quad (5.6)$$

де  $ЗП_{\text{ор}}$  – заробітна плата основних робітників, грн;

$P_{\text{зв}}$  – це звичайний розцінок за 1 послугу створення вебзастосунку, грн.

$N$  – річна кількість послуг, од.;

Звичайний розцінок за 1 послугу створення вебзастосунку, що визначається за формулою:

$$P_{\text{зв}} = T_{\text{од}} \times T_{\text{стс}}, \quad (5.7)$$

де  $T_{\text{од}}$  – трудомісткість робіт, н/год.;

$T_{\text{стс}}$  – тарифна ставка робітників, що зайняті у наданні цієї послуги, грн.

Згідно з законодавством на 1.01.21 мінімальна заробітна плата встановлена

на рівні **6000** грн. Отже тарифну ставку 1 розряду приймаємо у сумі **36,11** грн. за 1 годину.

Середній коефіцієнт складності робіт з надання послуги приймаємо 1,57.

Визначаємо середню тарифну ставку працівника за формулою:

$$T_{\text{сгс}} = T_{\text{сг1}} \times K_{\text{с}}, \quad (5.8)$$

де  $T_{\text{сгс}}$  – середня тарифна ставка працівника, грн.;

$T_{\text{сг1}}$  – тарифна ставка 1 розряду, грн.;

$K_{\text{с}}$  – середній коефіцієнт складності робіт з послуги.

$$T_{\text{сгс}} = 36,11 \times 1,57 = 56,69, \text{ грн.}$$

Обчислюємо розцінок звичайний:

$$P_{\text{зв}} = 716 \times 56,69 = 40591,97, \text{ грн.}$$

Обчислюємо заробітну плату основних робітників:

$$ЗП_{\text{ор}} = 40591,97 \times 9 = 284143,81, \text{ грн.}$$

Отже прямий фонд оплати праці основних робітників, що надаватимуть послугу створення вебзастосунку за рік становить 284143,81 грн. А на весь обсяг робіт фірми:

$$ЗП_{\text{ор заг}} = 284143,81 / 0,2995 = 948727,25, \text{ грн.}$$

Заробітну плату іншим категоріям персоналу розраховуємо за формулою:

$$ЗП_i = O_i \times Ч_i \times M, \quad (5.9)$$

де  $ЗП_i$  – заробітна плата інших категорій персоналу, грн.;

$О_i$  – оклад одного працівника певної категорії, грн.;

$Ч_i$  – чисельність працівників, ос.;

$М$  – кількість місяців у році, од.

Премії визначаю за формулою:

$$П = ЗП \times К_d, \quad (5.10)$$

де  $П$  – премії, грн;

$ЗП$  – пряма заробітна плата працівників, грн;

$К_d$  – коефіцієнт премії.

Річний фонд оплати праці за кожною категорією визначаємо за формулою:

$$\Phi ЗП = ЗП + П \quad (5.11)$$

де  $\Phi ЗП$  – фонд оплати праці, грн;

$ЗП$  – пряма заробітна плата, грн;

$П$  – премія, грн.

Нарахування на соціальні потреби містять: відрахування на соціальне страхування, у пенсійний фонд, фонд зайнятості, та інші фонди. На сьогодні єдиний соціальний внесок становить 22%.

Розрахунки здійснюємо у зведеній відомості фонду оплати праці підприємства (табл.5.2).



Таблиця 5.2 – Зведена відомість фонду оплати праці

Категорія персоналу	Оклад, грн.	Прямий ФЗП, грн.	Премії		ФЗП, грн.	Сума нарахувань, грн
			%	грн.		
Основні робітники	-	948727,25	16%	151796,36	1100523,61	242115,20
Допоміжний працівник	7860	943200	7%	6602,40	100922,40	22202,93
Директор	15000	180000	16%	28800	208800	45936
Бухгалтер*	7680	46080	14%	6451,20	52531,20	11556,86
Прибиральниця*	6000	36000	7%	2520	38520	8474,40
<b>РАЗОМ</b>	-	1305,127,25		196169,96	1501297,21	330285,39

\*Бухгалтера та прибиральницю приймаємо на 0,5 ставки.

### 5.5 Розрахунок матеріальних витрат на послугу

Розрахунок по елементам кошторису матеріальних витрат здійснюємо за формулами:

$$MB = \sum C_i \times n_i, \quad (5.12)$$

де MB – матеріальні витрати, грн.;

$C_i$  – ціна купованого виробу або матеріалу, грн.;

$n_i$  – кількість купованих виробів та матеріалів, од.

В нашому випадку матеріальні витрати відображені в табл. 5.3.

Таблиця 5.3 – Розрахунок матеріальних витрат

Найменування елементів	Джерело доступу	Ціна за одиницю, грн
Visual Studio Code	<a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a>	Умовно безкоштовна
DB Browser for SQLite	<a href="https://sqlitebrowser.org/dl/">https://sqlitebrowser.org/dl/</a>	Умовно безкоштовна
SQLite	<a href="https://www.sqlite.org/index.html">https://www.sqlite.org/index.html</a>	Умовно безкоштовна
Docker	<a href="https://www.docker.com/">https://www.docker.com/</a>	Умовно безкоштовна
Ubuntu Server	<a href="https://ubuntu.com/download/server">https://ubuntu.com/download/server</a>	Умовно безкоштовна
Хостинг	<a href="https://vscale.io/en/">https://vscale.io/en/</a>	348,22
<b>РАЗОМ</b>		348,22

## 5.6 Розрахунок загальновиробничих витрат

Підприємство планує орендувати приміщення з виробничою площею 55 м<sup>2</sup>. Вартість оренди в місяць приймаємо: 20 грн/м<sup>2</sup>. Загальну вартість оренди в рік визначаємо за формулою:

$$B_{\text{ор}} = S_{\text{заг.}} \times B_{\text{ор.1м}}^2 \times M, \quad (5.14)$$

де  $B_{\text{ор}}$  – загальна вартість оренди, грн;

$S_{\text{заг.}}$  – загальна площа орендованого приміщення підприємства, м<sup>2</sup>;

$B_{\text{ор.1м}}^2$  – вартість оренди 1 м<sup>2</sup> в місяць, грн;

$M$  – кількість місяців, од.

$$B_{\text{ор.}} = 55 \times 20 \times 12 = 13200, \text{ грн.}$$

Визначаємо суму комунальних платежів. Щоб визначити кількість енергії та освітлення використовуємо формулу:

$$E_{\text{осв}} = S_{\text{заг.}} \times \Phi_{\text{еф}} \times N_{\text{осв}} / (1000 \times \text{кВт}), \quad (5.15)$$

де  $E_{\text{осв}}$  – кількість енергії освітлення, кВт/год ;

$S_{\text{заг.}}$  – загальна площа орендованого приміщення підприємства, м<sup>2</sup>;

$\Phi_{\text{еф}}$  – ефективний фонд, год.;

$N_{\text{осв}}$  – це норма інтенсивності освітлення одного м<sup>2</sup> площі. Приймаємо – 40 Вт/м<sup>2</sup>;

кВт – коефіцієнт втрат електроенергії в мережі 0,92.

$$E_{\text{осв}} = (55 \times 1717,15 \times 40) / (1000 \times 0,92) = 3953, \text{ кВт/год.}$$

Вартість освітлення визначаємо за формулою:

$$B_{\text{осв}} = E_{\text{осв}} \times Ц1, \quad (5.16)$$

де  $B_{\text{осв}}$  – вартість освітлення, грн;

$Ц1$  – ціна однієї кВт години, яка з 1.01.2021 р. встановлена **1,68** грн. за кВт/год.

$$B_{\text{осв}} = 3953 \times 1,68 = 6641,71, \text{ грн.}$$

Витрати на опалення визначаються за формулою:

$$B_{\text{оп}} = S_{\text{заг.}} \times Ц2 \times M, \quad (5.17)$$

де  $B_{\text{оп}}$  – витрати на опалення, грн;

$S_{\text{заг.}}$  – загальна площа,  $\text{м}^2$ ;

$Ц2$  – вартість опалення одного  $\text{м}^2$  площі, грн. Повний сезонний тариф **42,9** грн.(з ПДВ) з **21.11.2018**;

$M$  – кількість опалюваних місяців у році, од.

$$B_{\text{оп}} = 55 \times 42,9 \times 6 = 14157, \text{ грн.}$$

Витрати на воду визначаються за формулою:

$$B_{\text{в}} = Ч \times Ц3 \times 0,5 \times M, \quad (5.18)$$

де  $B_{\text{в}}$  – витрати на воду, грн;

$Ч$  – чисельність персоналу, ос.;

$Ц3$  – вартість за витрачання одного  $\text{м}^3$  води, ос. Централізоване водопостачання та водовідведення – **33,94** грн. (з ПДВ);

$0,5 \text{ м}^3$  – орієнтовна норма витрачання води на 1 працівника в місяць;

$M$  – кількість місяців у році, од.

$$B_v = 13 \times 33,94 \times 0,5 \times 12 = 2647,32, \text{ грн.}$$

Знаходимо амортизацію на кожен елемент основних фондів за формулою:

$$A = (B_{\text{оф}} \times H_a) / 100, \quad (5.19)$$

де  $A$  – амортизація на кожен елемент основних фондів, грн;

$H_a$  – норма амортизації, %;

$B_{\text{оф}}$  – вартість основних фондів, грн.

Визначаємо амортизаційні відрахування за податковим методом у таблиці 5.4.

Таблиця 5.4 – Розрахунок амортизаційних відрахувань

Категорія основних фондів	Вартість ОФ, грн	% відрахувань	Амортизація, грн
Обладнання	100000	10	10000
Інструмент	1400	25	350
Інвентар	1900	27	513
<b>РАЗОМ</b>	103300	-	10863

Витрати на ремонт основних фондів приймаємо в розмірі 5 % від вартості основних фондів. Розраховуємо витрати на ремонт:

$$B_{\text{р.осн.ф}} = 103300 \times 0,05 = 5165, \text{ грн.}$$

Обчислюємо витрати на охорону праці, які приймаємо у розмірі 150 грн:

$$B_{\text{ох.пр.}} = 13 \times 150 = 1950, \text{ грн.}$$

Інші витрати приймаємо в сумі 4% від всіх загальновиробничих витрат. Усі елементи витрат зводимо у таблицю 5.5.

Таблиця 5.5 – Загальновиробничі витрати

Елементи витрат	Сума, грн.
1. Оренда	13200
2. Електроенергія	6641,71
3. Опалення	14157
4. Вода	2647,32
5. Амортизація	10863
6. Ремонт основних фондів	5165
7. Охорона праці	1950
Всього	54624,03
8. Інші витрати	2184,96
<b>РАЗОМ</b>	<b>56808,99</b>

### 5.7 Визначення собівартості та ціни послуги щодо створення вебзастосунку

До статей калькуляції відносяться: матеріальні витрати, витрати на транспортування, витрати на оплату праці, загальновиробничі витрати.

Калькуляцію собівартості здійснюємо у таблиці 5.6.

Таблиця 5.6 – Розрахунок собівартості 1 послуги

Статті калькуляції собівартості	Сума, грн	Примітки
1. Матеріальні витрати	348,22	З таблиці 5.3
2. Пряма ЗП	55840,80	З таблиці 5.2 з розрахунку на одну послугу
3. Премії	8393,27	
4. Нарахування	14131,50	
5. Загальновиробничі витрати	2430,61	З таблиці 5.5 з розрахунку на 1 послугу
6. Позавиробничі витрати	243,06	10% від пункту 5
Повна собівартість	81387,46	Сума пунктів 1-6

Обчислюємо прибуток згідно планової норми рентабельності за формулою:

$$\Pi = C \times k_p, \quad (5.20)$$

де  $\Pi$  – прибуток, грн;

$C$  – собівартість однієї послуги, грн;

$k_p$  – плановий коефіцієнт рентабельності.

Приймаємо плановий коефіцієнт рентабельності в розмірі 17%.

Здійснюємо розрахунки:

$$\Pi = 81387,46 \times 0,17 = 13835,87, \text{ грн.}$$

Обчислюємо ціну створення вебзастосунку (без ПДВ) за формулою:

$$\Pi_{\text{опт}} = C + \Pi, \quad (5.21)$$

Здійснюємо розрахунки:

$$\Pi_{\text{опт}} = 81387,46 + 13835,87 = 95223,33, \text{ грн.}$$

Виходячи з розрахунків, проведених в даному розділі, можна зробити висновок, що вартість послуги щодо створення вебзастосунку в середньому буде 95223,33 грн. Для того щоб окупити розробку даного продукту врахувавши його рентабельність, необхідно продати його 200 користувачам із річною підпискою вартістю 476,12 грн.

На ринку комп'ютерних послуг України ціна на річну підписку такої ж послуги може коливатись від 2331 до 11520 грн. Оскільки наша ціна є нижчою, то запровадження даного виду надання послуг нашої фірми є доцільним.

## ВИСНОВКИ

Метою даної роботи було створення вебзастосунку для управління проектами. Реалізація відбулася на основі бази даних SQLite, програмна реалізація на мові Python із використанням вебфреймворку Flask у середовищі Visual Studio Code. Для організації командної роботи над проектом було використано систему контролю версій Git та сервіс для управління репозиторіями GitHub.

За час виконання проекту було опрацьовано велику кількість матеріалу щодо розробки програмних продуктів та власне предметної області, а саме проаналізований функціонал подібних застосунків, виділено необхідні інструменти для організації керування проектною діяльністю. Також були здобуті практичні навички у використанні згаданих технологій, роботи у команді та загальній розробці програмних продуктів.

Даний вебзастосунок розроблявся максимально універсальним, тому він зможе задовільнити потреби замовника (зокрема, і коледжу) в організаційній діяльності, а також різноманітні потреби щодо планування власної та спільної діяльності інших людей. Завдяки простим інструментам та інтуїтивно зрозумілому інтерфейсу він буде легким до освоєння навіть тим, хто раніше не мав досвіду використання подібних застосунків.

Цей проект має можливість до розширення, а саме модифікації та перепису його певних функціональних елементів за бажанням замовника.

Сьогодні впровадження застосунків у керування проектною діяльністю є поширеним та, зазвичай, необхідним явищем, котре дозволяє тримати всю інформацію впорядкованою в одному місці та економити час, тому використання розробленого в межах дипломної роботи проекту є необхідним для підвищення ефективності як командної, так і особистої діяльності.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний вебсайт вебзастосунку Trello. Головна [Електронний ресурс]. – Режим доступу: <https://trello.com/>.
2. Офіційний вебсайт вебзастосунку Kendis. Головна [Електронний ресурс]. – Режим доступу: <https://kendis.io/>.
3. Офіційний вебсайт вебзастосунку Basecamp. Головна [Електронний ресурс]. – Режим доступу: <https://basecamp.com/>.
4. Офіційний вебсайт вебзастосунку Бітрікс24. Головна [Електронний ресурс]. – Режим доступу: <https://www.bitrix24.ua/>.
5. Офіційний вебсайт вебзастосунку Ganttpro. Головна [Електронний ресурс]. – Режим доступу: <https://ganttpro.com/>.
6. Офіційний вебсайт вебзастосунку Google Classroom. Сторінка користувача [Електронний ресурс]. – Режим доступу: <https://classroom.google.com/u/0/h>.
7. Офіційний вебсайт вебзастосунку Edmodo. Головна [Електронний ресурс]. – Режим доступу: <https://new.edmodo.com/>.
8. Офіційний вебсайт вебзастосунку My Class Campus. Головна [Електронний ресурс]. – Режим доступу: <https://myclasscampus.com/home>.
9. Офіційний вебсайт вебзастосунку Eduwonka. Головна [Електронний ресурс]. – Режим доступу: <https://www.eduwonka.com/>.
10. Server-side Sessions in Flask with Redis [Електронний ресурс]. – Режим доступу: <https://testdriven.io/blog/flask-server-side-sessions/>.
11. Ютуб-канал «Артем Матяшов». Відео «Основы Docker. Большой практический выпуск» [Електронний ресурс]. – Режим доступу: <https://youtu.be/QF4ZF857m44>.