МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток підтримки управління виконанням проектів в компанії»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-62 Волін Вадим Володимирович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «__» _____2020 р.

Науковий керівник _____ к.т.н., доц., Ващенко С. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Волін Вадим Володимирович

1 Тема роботи *Web-додаток підтримки управління виконанням проектів в компанії*
керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,
затверджені наказом по університету від « 14 » травня 2020 р. № 0576-III

2 Строк подання студентом роботи «1» червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку інформаційної системи

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування інформаційної системи, практична реалізація додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність роботи, мета та задачі, аналіз аналогів, порівняння додатків-аналогів, функціональні вимоги до інтерактивного додатку, засоби реалізації, структурно-функціональне моделювання роботи додатку, діаграма ВВ, діаграми діяльності, проектування бази даних, діаграма класів, навігація по системі, практична реалізація, висновки (всього 20 слайдів презентації)

6 Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	Ващенко С.М.		
Проектування інформаційної системи	Ващенко С.М.		
Практична реалізація додатку	Ващенко С.М.		

7.Дата видачі завдання_____01.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	01.03.20-11.03.20	
2	Постановка задачі та вибір методів розробки	12.03.20-20.03.20	
3	Постановка задачі та вибір методів розробки	12.03.20-20.03.20	
4	Проектування роботи	03.04.20-21.05.20	
5	Тестування	22.05.20-27.05.20	
6	Впровадження системи до загального доступу	28.05.20-09.06.20	

Студент _____ Волін В. В.
(підпис)

Керівник роботи _____ к.т.н., доц. Ващенко С. М.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підтримки управління виконанням проектів в компанії».

Кваліфікаційну роботу бакалавра присвячено розробці додатку для підтримки управління виконанням проектів.

В першому розділі проведено аналіз предметної області. У цьому розділі виконано: огляд останніх досліджень і публікацій, аналіз програмних продуктів – аналогів, постановку задачі, котра в свою чергу описує мету та задачі дослідження, які обов'язково повинні бути виконані, а також вибір засобів реалізації.

В другому розділі описано проектування інформаційної системи. У цьому розділі виконано: структурно-функціональне моделювання роботи додатку, в даному підрозділі наведено архітектуру інформаційної системи, IDEF модель головного процесу, котра показує функціональність додатку, далі було виконано моделювання варіантів використання, моделювання діаграми діяльності, діаграми послідовності, проектування бази даних та моделювання діаграми класів.

В третьому розділі описано практичну реалізацію додатку. У цьому розділі виконано: опис структурування програмного додатку, тобто опис прототипу додатку, який відображає функціонал сторінок додатку, та повідомлення про помилки, далі йде опис програмної реалізації та останнім підрозділом йде опис використання додатку.

Результатом проведеної роботи є розроблений web-додаток для підтримки управління виконанням проектів.

Кваліфікаційна робота містить 140 сторінок, 22 таблиці, 52 рисунків, список літератури 22 найменувань, 3 додатки.

Ключові слова: менеджер, користувач, додаток, проект, інтерфейс, файл, розробник, система, база даних, середовище розробки, запит, звіт.

ЗМІСТ

ВСТУП 6
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1. Огляд останніх досліджень і публікацій 8
1.2. Аналіз програмних продуктів – аналогів 10
1.3 Постановка задачі 18
1.3.1 Мета та задачі дослідження 18
1.3.2 Вибір засобів реалізації 20
2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 24
2.1. Структурно-функціональне моделювання роботи додатку 24
2.2. Моделювання варіантів використання 30
2.3. Моделювання діаграм діяльності 35
2.4. Проектування бази даних 30
2.5. Моделювання діаграми класів 30
3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ 41
3.1. Структура програмного додатку 41
3.2. Програмна реалізація 42
3.3. Використання програмного додатку 60
ВИСНОВКИ 75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 76
ДОДАТОК А 79
ДОДАТОК Б 97
ДОДАТОК В 111

ВСТУП

Управління проектами – діяльність, яка поєднує сучасні технології та знання з практичними навичками. Управління проектом є процес, протягом якого очікується ефективна робота з метою отримання максимального ефекту від реалізації. Управління проектами – це методологія планування та використання ресурсів протягом його життєвого циклу.

Для вдалого завершення проекту необхідна не тільки команда кваліфікованих співробітників, але і інструменти, за допомогою яких можна фіксувати виконані задачі та вдало управляти ресурсами. Серед чітко орієнтованих є рішення, які підходять для управління проектами різних видів. Проте, зазвичай, вони є дороговартісними і для компаній, які тільки знаходяться в стані становлення, це суттєві витрати.[14]

Тому метою даної роботи є розробка власної інформаційної системи для управління виконанням проектів.

Для досягнення поставленої мети потрібно виконати такі задачі.

1. Провести аналіз предметної області, а саме:
   - проаналізувати теперішній стан використання інформаційних технологій у області проектного менеджменту;
   - провести аналіз існуючих аналогів інформаційних систем для допомоги в управлінні проектів.
2. Визначити методи дослідження інструментів для швидкої та надійної роботи при створенні системи та спланувати роботу над проектом.
3. Провести проектування системи.
   - проаналізувати та обрати найкращий тип архітектури;
   - побудувати концептуальну діаграму бізнес логіки проекту та провести декомпозицію процесу роботи системи;
   - побудувати діаграму варіантів використання для деталізації функціональних етапів та залежностей;

— побудувати моделі діяльності основних модулів проекту;
— побудувати діаграму послідовності для деталізації передачі сигналів з одного процесу до іншого;
— спроектувати базу даних;
— побудувати діаграму класів для майбутнього проектування.

4. Виконати розробку інформаційної системи.

Практичне значення роботи полягає в тому, що отримано зручний продукт для автоматизації процесу управління процесом вокнання проектів.

Результати роботи доповідалися на щорічній науковій конференції «ІМА-2020» [22].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Предметна область - сукупність послуг та інструментів, створення яких має бути забезпечене в рамках здійснюваного проекту.

Питанням про те, чи варто застосовувати в компанії проектний менеджмент, задаються багато керівників. Цей інструмент вже впроваджено у багатьох сферах бізнесу, і зараз необхідно розуміти, як застосовувати технологію виходячи з планів, цінностей та завдань компанії, які інструменти використовувати для цього. [12][13]

На даний момент є великий вибір методологій управління виконанням проектів.

Всі доступні типи менеджменту можна розподілити на такі групи:

- традиційні послідовні методології (CPM, Waterfall);
- змінні методології(ECM, XPM);
- класичні методології (PMBOK);
- гнучкі методології (Agile, Scrum, Kanban і ін.);
- процесні методології (Lean, Process-Based Project Management);
- інші(Prince2, PRiSM).

Після вдало обраної методології, можна перейти до вибору інструментів для управління виконанням проекту. Обираючи такий інструмент користувач шукає продукт,який міг би складати діаграми Ганта, будувати графіки, розподіляти повноваження і завдання виконавцям з різними ролями.[9][10]

Ефективність проектного менеджменту була підтверджена багаторічним досвідом світових бізнесгігантів. Отже, необхідність обрання цієї форми управління стає все більш очевидною. Серед тих хто використовують цей підхід є передові компанії в самих різних напрямках:

- УКРСИББАНК BNP PARIBAS;
- ВОЛЯ КАБЕЛЬ;

— РАЙФФАЙЗЕН БАНК АВАЛЬ;
— МЕДІАХОЛДИНГ «ВІСТІ»;
— CORUM GROUP;
— COMFY TRADE;
— TERRA FOOD;
— МТС.

Практичне значення роботи полягає у розробці інструментарію для ефективного управління проектами. Розроблена веб-система є повноцінний інструмент для планування проектів та управління виконанням.[15]

Провівши аналіз існуючих інструментів для контролю управлінням виконання проектів та загальної предметної області було визначено такі характеристики:

— Взаємодія користувача з системою;
— Контроль процесу виконання проектів за допомогою графіків;
— Можливість управління командами проекту;
— Можливість зміни користувачем власних даних;
— Можливість завантаження файлів до інформації про проект;
— Наявність Kanban дошки як особистої для користувача, так і загальної для

проекту;

— Можливість планування майбутніх задач;
— Можливість контролю виконанням користувачів відповідних задач.

На даний момент більшість крупних компаній використовують проектний менеджмент як засіб для ефективного планування, управління проектами та їх виконанням.[16]

Тому для допомоги менеджерам, керівникам проектів з роботою створюється велика кількість інструментів, але всі ці засоби або не мають центрального модулю для поєднання в одну систему, або мають певні недоліки у використанні.[11][18]

З цього можна зробити висновок, що створення такої системи має перспективи у розвитку та масштабуванні у майбутньому. Метою даної роботи є, розробка інформаційної системи, яка буде повністю відповідати предметній області.

Вирішувати всі необхідні проблеми та відповідати поставленим вимогам. Система буде слугувати універсальним інструментом для управління виконанням проектів, які створювались з використанням різних методологій та відносяться до різних галузей.[20][17]

1.2 Аналіз програмних продуктів – аналогів

Системи для управління проектами поділяються на спеціалізовані та неспеціалізовані, в залежності від призначень та набору інструментів. [1]

Неспеціалізовані системи легші у освоєнні для базового користувача, тому що мають базовий інтерфейс с базовим функціоналом, але якщо потрібні більш широкі функції, то потрібно дивитись у бік спеціалізованих сервісів. [3][19]

Список систем управління проектами та їх короткий опис:

- Microsoft Project (або MSP),
- Spider Project,
- Trello,
- Jira,
- Bitrix24,
- ProjectMate,
- OpenPlan.

Microsoft Project – це доступний комплекс управління цілісними проектами та окремими внутрішніми процесами, сервіс пропонує оптимізувати бізнес-процеси з метою автоматизації та створення деяких видів діяльності. Реалізовані інструменти вже не одне десятиріччя привертають увагу як великих міжнародних організацій, так і представників місцевого малого та середнього бізнесу.

MS Project дуже простий в ідейному плані. Він оперує трьома сутностями завдання, ресурси, календар і зв'язки між ними. По суті - це база даних, призначена

для користувача, інтерфейс для створення і редагування сутностей і мінімальна, досить проста автоматизація. [1]

Trello – зручна система управління проектами, простий і універсальний онлайн-інструмент у вигляді веб-версії trello.com , мобільних додатків для Android і IOS, а також розширення для браузера Google Chrome. Сервіс реалізований за принципом канбан-дошки: користувач створює дошку, на ній списки, всередині яких картки. Простота, універсальність і багатофункціональність. За це його обирають професійні керівники проектів і продукт-менеджери. У цій системі управління проектами за замовчуванням немає діаграми Ганта, неможливо оцінити прогрес в часі.[2]

Тарифи Trello:

1 Безкоштовний:

— необмежену кількість персональних дошок, списків і карток в дошці;
— підключення 1 плагіна до дошки;
— вкладення в картку - до 10 Мб
— 10 групових дошок.

2 Business Class - $ 9.99 в місяць за 1 користувача:

— необмежену кількість персональних дошок, списків і карток в дошці;
— необмежену кількість плагінів до дошки;
— вкладення в картку - до 250Мб;
— необмежену кількість групових дошок;
— додатковий рівень доступу для користувачів без можливості редагувати, тільки коментувати;
— колекції дошок;

3 Enterprise - $ 20.83 в місяць за 1 користувача:

— необмежену кількість персональних дошок, списків і карток в дошці;
— необмежену кількість плагінів до дошки;
— вкладення в картку - до 250Мб;
— необмежену кількість групових дошок;
— додатковий рівень доступу для користувачів без можливості редагувати,

тільки коментувати;

— колекції дошок;

— додаткові можливості адміністрування дошок, прав доступу і індивідуальна адаптація співробітників.

Jira – це онлайн-сервіс або система управління проектами з веб-версією і мобільними додатками для Android і iOS. Заснований на методології Agile, являє собою дошку з картками і є найпопулярнішим серед команд розробки. Jira належить компанії Atlassian, яка також володіє Trello. [2]

Плюси Jira:

— простий і зрозумілий інтерфейс;

— канбан або scrum-дошки;

— можливість створення звітів;

Недоліки Jira:

— спочатку ця система управління проектами розрахована на спеціалізовані

— команди розробників, включає в себе можливості прив'язки до коду і не завжди необхідні для інших проектів опції.

Тарифи Jira:

Вартість залежить від кількості користувачів.

— Фіксований тариф для команди до 10 чоловік - $ 10 в місяць;

Гнучка модель:

— перші 11-100 користувачів - $ 7 на місяць за людину;

— наступні 150 користувачів - $ 5 в місяць за людину;

— після 250 чоловік - $ 1.10 в місяць за кожного наступного користувача.

Бітрікс24 – це найпопулярніший на пострадянському просторі CPM-сервіс, схожий з Мегаплан, дуже потужна система управління проектами. Сповідує принцип одного вікна, включає в себе практично все, що може знадобиться бізнесу. [2]

Переваги Бітрікс24:

— можливість налаштування наскрізний аналітики;

— можливість налаштування воронки продажів;

— комфортна інтеграція корпоративного порталу;

— месенджер і чат-боти;

— інтеграція з 1С і 1С-Бітрікс.

Недоліки Бітрікс24:

— багато в чому надлишковий інтерфейс;

— вимагає ресурсів на навчання та впровадження. [2]

ProjectMate – це Російська PSA-система автоматизованої роботи. Дослідження модуля управління має максимум функцій, необхідних у компаніях сфери консультаційних служб (юристи, адвокати та інші). [3]

На рисунку 1.1 зображено приклад інтерфейсу програми Spider Project.

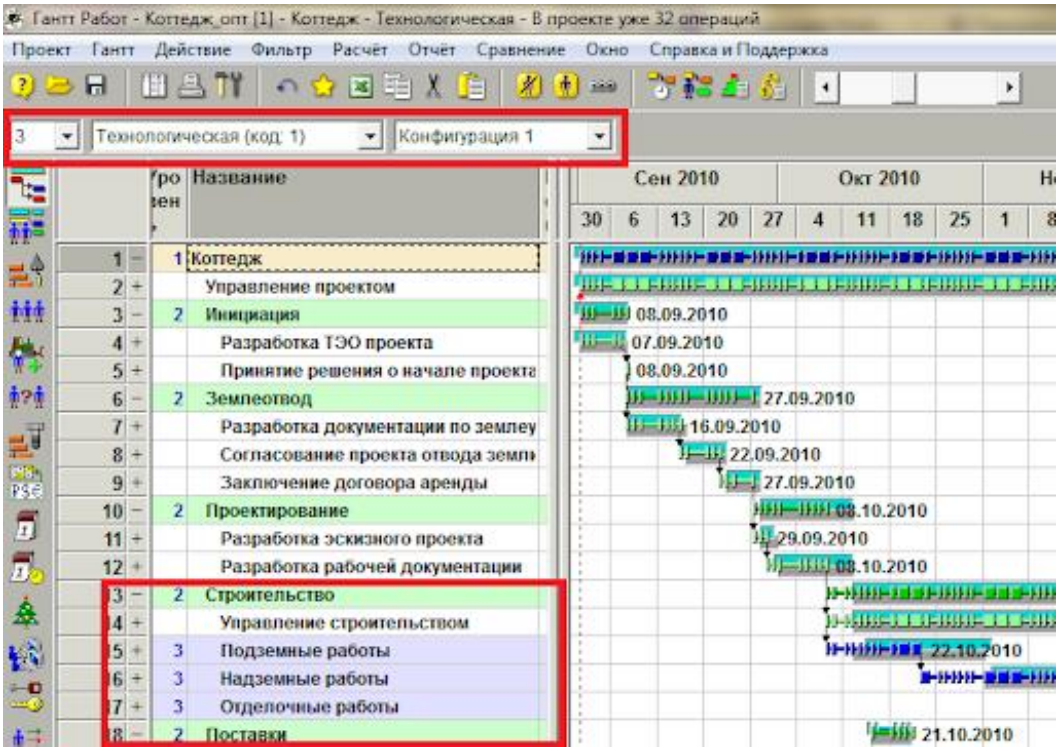

Рисунок 1.1 – Інтерфейс Spider Project

На рисунку 1.2 зображено приклад інтерфейсу програми Trello.

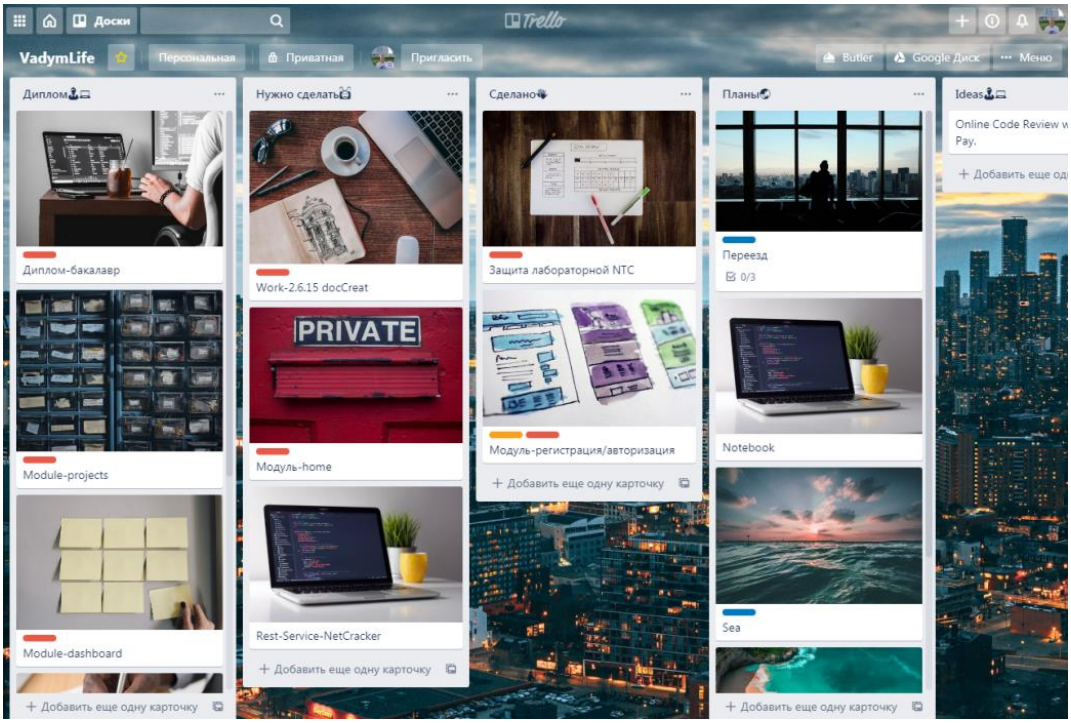

Рисунок 1.2 – Інтерфейс Trello

На рисунку 1.3 зображено приклад інтерфейсу програми Bitrix24.

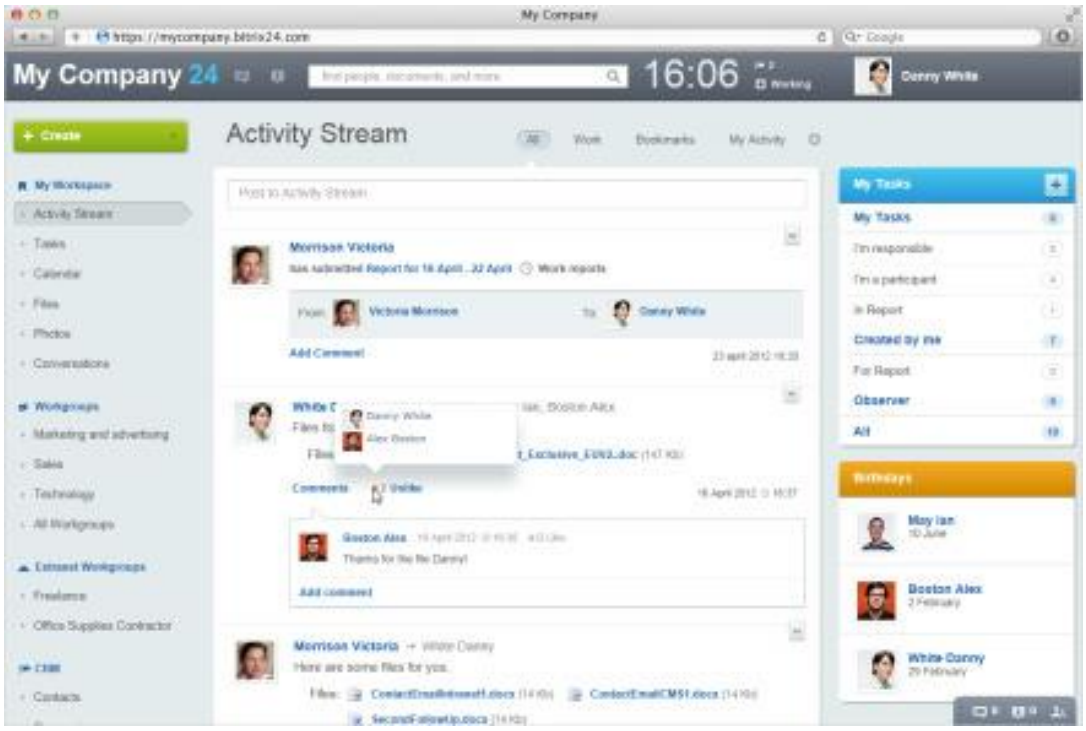

Рисунок 1.3 – Інтерфейс Bitrix24

На рисунку 1.4 зображено приклад інтерфейсу програми OpenPlan.

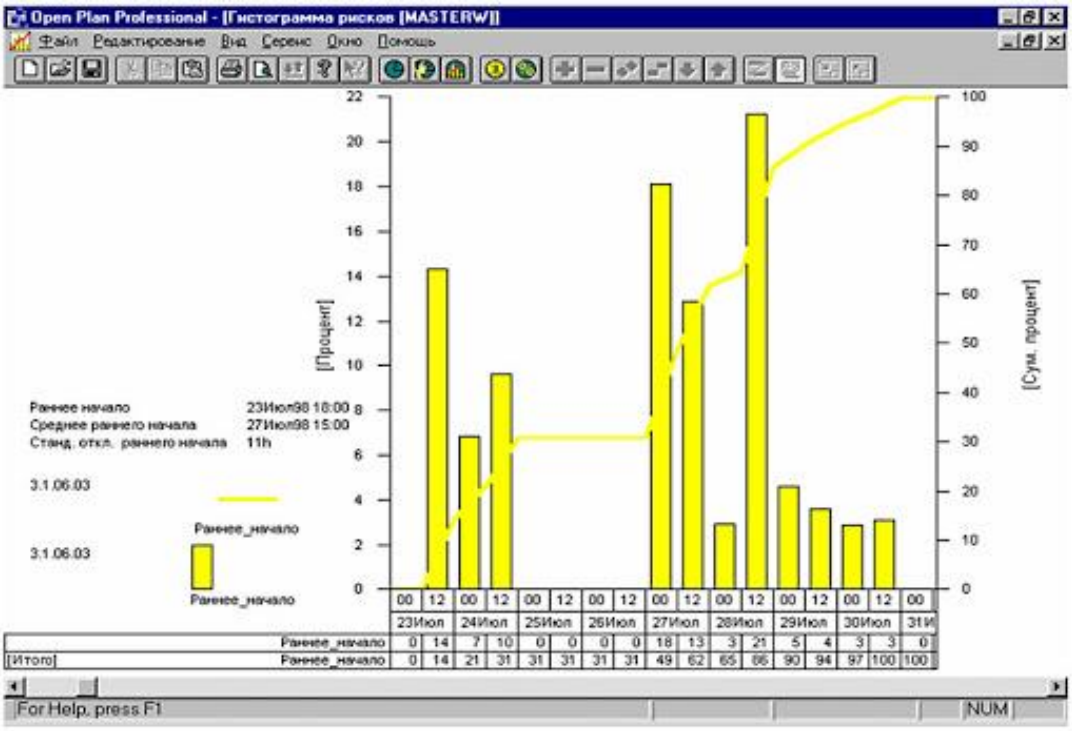

Рисунок 1.4 – Інтерфейс OpenPlan

На рисунку 1.5 зображено приклад інтерфейсу програми Jira.

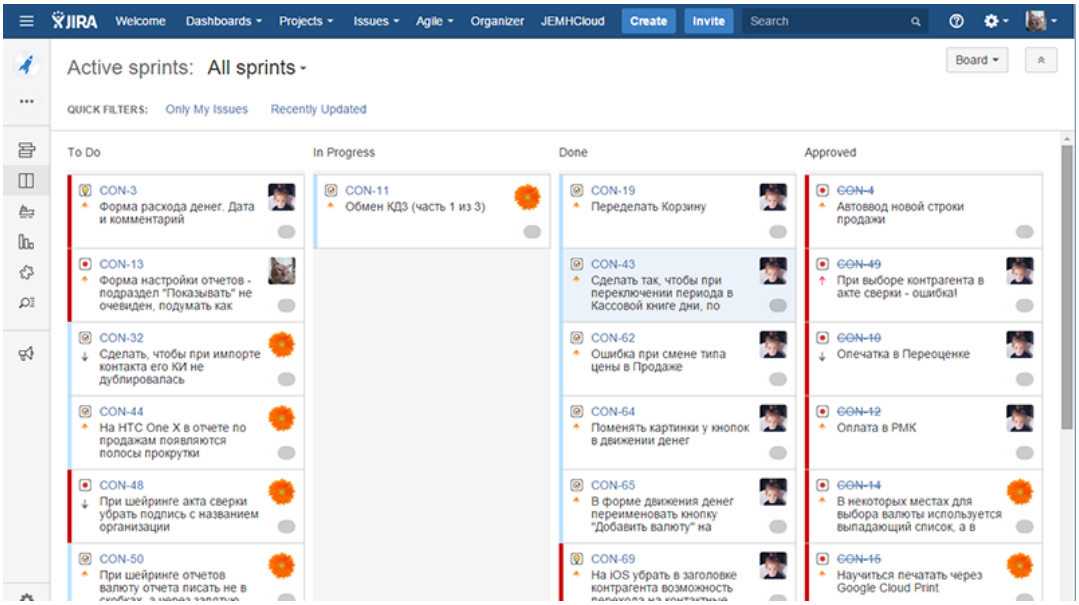

Рисунок 1.5 – Інтерфейс Jira

Проаналізувавши веб-сервіси управління проектами, було проведено порівняльний аналіз (табл.1.1), де було виявлено позитивні та негативні сторони, які при створені дипломного проекту треба уникати або обернути на користь системі.

Недоліки:

— наявність непотрібної реклами;
— недопрацьований контент;
— необхідність навчання для використання системи;
— важкий у розумінні інтерфейс;
— надмірний інтерфейс;
— платний функціонал;
— обмежений доступ до пробного періоду;
— онлайн платформа у платному тарифі.

До переваг сайтів-аналогів можна віднести:

— зручна панель навігації;
— дизайн платформ;
— наявність хмарного середовища;
— інтеграція зовнішніх модулів;
— функціональний набір.

Таблиця 1.1 – Аналіз декількох комерційних платформ для управління проектами

Критерії	**ManLuck**	Spider Project	Битрикс24	OpenPlan	Jira	Trello	Microsoft Project	Asana
Відсутність непотрібної реклами	+	+	+	+	+	+	+	+
Інструменти для побудови діаграм	-	+	+	-(лише у комерційній версії)	+	-	+	-
Можливість управління декількома проектами	+	-	-	-	+	+	-	+

Продовження табл.1.1.

Тарифи	-	45 000 руб. За одну ліцензію	Проект - безкоштовно 12 користувачів; 5Гб в хмарі. Проект + - 999 руб / місяць за користувача при оплаті річної передплати 24 користувача; 24Гб в хмарі. Команда - 4990 руб / місяць за користувача при оплаті річної передплати 50 користувачів; 100 ГБ в хмарі.	$ 9.99 в мі-сяць за	Фіксований для команди до 10 чоловік - $ 10 в місяць; Гнучка модель: - перші 11-100 користувачів - $ 7 на місяць за людину; - наступні 150 користувачів - $ 5 в місяць за людину; - після 250 чоловік - $ 1.10 в місяць за кожного наступного користувача.	Business Class - $ 9.99 в місяць за 1 користувача Enterprise - $ 20.83 в місяць за 1 користувача	Project (план 1) $10.00 за користувача на місяць (річне зобов'язання) Project (план 3) USD$30.00 за користувача на місяць (річне зобов'язання) Project (план 5) USD$55.00 за користувача на місяць (річне зобов'язання)	Premium - $ 9.99 per month per user
Зручна панель навігації	+	-	+	-	+	+	-	-

Продовження табл.1.1.

Дошка задач для власного користування	+	-	+	-	+	+	-	-
Дизайн платформи	+	-	-	-	+	+	-	+
Хмарне середовище	-	-	-	-	-	-	+	-
Безкоштовна	+	-	+/-	Безкоштовна версія з обмеженим функціоналом	-	+ є безкоштовний доступ - ширший функціонал платний	-	Безкоштовна версія Комерційна версія
Онлайн доступ	+	-	+	-	+	+	+	+
Інтуітивний дизайн	+	-	+	-	+	-	-	-
Розподілення доступу між командою	+	-	-	-	+	+	-	+
Засоби побудови звітів	+	+	+	+	+	-	+	-
Перегляд статистики	+	+	+	+	+	-	+	-
Кроссплатфрмна	+	-	+	-	+	+	-	+

1.3 Постановка задачі

1.3.1 Мета та задачі дослідження

Метою дипломної роботи є розробка web-додатка для управління виконанням проектів. Даний проєкт повинен забезпечити автоматизацію роботи, виконувану при управлінні виконанням проектів.

Для досягнення мети було визначено наступні задачі:

— провести аналіз існуючих додатків-аналогів;

— визначити послідовність дій створення веб-додатку та розмежувати процес створення додатку на етапи;

— проаналізувати та обрати тип архітектури веб-додатку та інструменти для швидкого та оптимального процесу побудови додатку;

— змоделювати функції додатку засобами uml;

— спроектувати та створити базу даних веб-додатку;

— розробка веб-додатку;

— тестування додатку.

Функціональні вимоги до веб-додатку.

— *Аутентифікація у системі*. Користувачі матимуть змогу пройти авторизацію чи реєстрацію у системі.

— *Зміна особистих даних користувача*. Користувачі матимуть змогу змінити особисті дані.

— *Створення проектів.* Користувач матиме змогу створювати проекти у системі.

— *Управління командою проекту*. Користувач матиме змогу додавати або видаляти учасників проекту.

— *Створення списків задач.* Користувач матиме змогу формувати списки задач для окремого проекту чи для власного використання.

— *Формування та завантаження звітів.* Користувач матиме змогу обрати на панелі проекту завантаження звіту у форматі PDF.

— *Завантаження файлів до проекту.* Користувач матиме змогу завантажувати файли до обраного проекту.

— *Перегляд статистики проекту.* Користувач матиме змогу переглядати статистику обраного проекту.

Інтерфейс веб-додатку повинен бути інтуїтивно зрозумілим навіть тим користувачам, які не мають ніякого відношення до предметної області проекту.

Даний проєкт буде у майбутньому буде розширений за допомогою створення мобільного додатку, формуванні модулю сповіщення та додавання нейронної мережі для розподілення задач у команді.

Повністю вимоги до розробки web-додатка для управління виконанням проектів наведені у технічному завданні (додаток А).

Проведено планування робіт, результати наведено в додатку Б.

1.3.2 Вибір засобів реалізації

Для розробки веб-додатку було використано середовище розробки Intellij IDEA, DataGrip JetBrains, Apache Tomcat, Java SE/EE.

JetBrains Intellij IDEA – інтегроване середовище розробки (IDE), в основному призначена для Java. Це середовище використовується спеціально для розробки програм. Він розроблений компанією під назвою JetBrains, яка формально називалася IntelliJ. Він доступний в двох редакціях: Community Edition, ліцензованої Apache 2.0, і комерційною версією, відомої як Ultimate Edition. Обидва вони можуть бути використані для створення програмного забезпечення, яке може бути продане. Що відрізняє IntelliJ IDEA від аналогів, так це простота використання, гнучкість і продуманий дизайн, також середовище розробки має розширені функції перевірки помилок, що дозволяє швидше і простіше перевірити помилки.

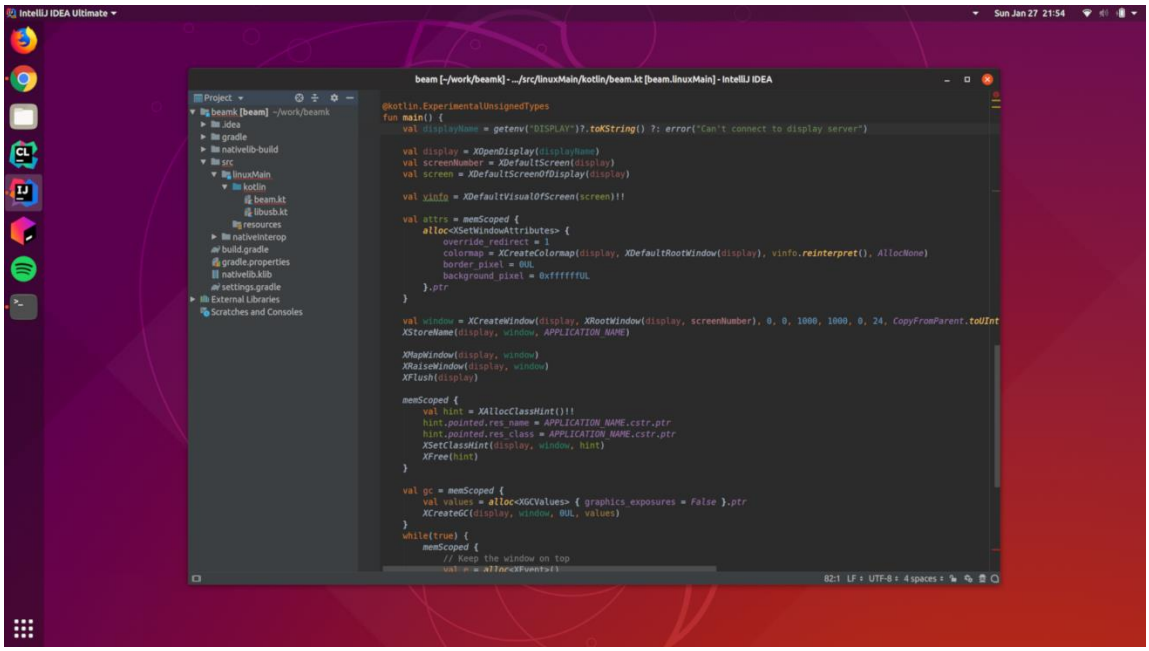

Рисунок 1.6 – Вікно редактора Intellij IDEA

Переваги Intellij IDEA:

— Розумне завершення;
— Завершення ланцюга;
— Статичне завершення членів;
— Аналіз потоку даних;
— Мовна ін'єкція;
— Багатомовне переробництво;
— Виявлення дублікатів;
— Перевірки та швидкі виправлення;
— Контроль версій;

DataGrip JetBrains – це вдосконалене середовище розробки інтегрованих баз даних багатомоторних систем, створене для професійних розробників SQL, щоб вони могли ефективно виконувати запити та виконувати навігацію за схемою. Розроблена JetBrains, система виділяє вбудовані драйвери для підтримки різних двигунів, таких як Exasol, Derby, AWS Redshift, DB2, HSQLDB, Microsoft Azure, H2, MySQL, Oracle, SQL Server, Sybase, SQLite та PostgreSQL. DataGrip пропонує безліч введення даних та деякі розширені функціональні можливості для полегшення перетворення та розвитку об'єктів, щоб відповідати підтримуваній мові.

Інтегрований інтерфейс передбачений для систем управління версіями, щоб запропонувати кращий досвід користувача з Perforce, Git, CVS, Subversion, GitHub, TFS, Perforce та Mercurial. Деякі основні функції включають редактор даних, об'єкти бази даних, розумний текстовий редактор, навігацію, заповнення коду, локальну історію, аналіз коду та швидкі виправлення, консоль запитів, підтримка VCS та переглядач різниці.

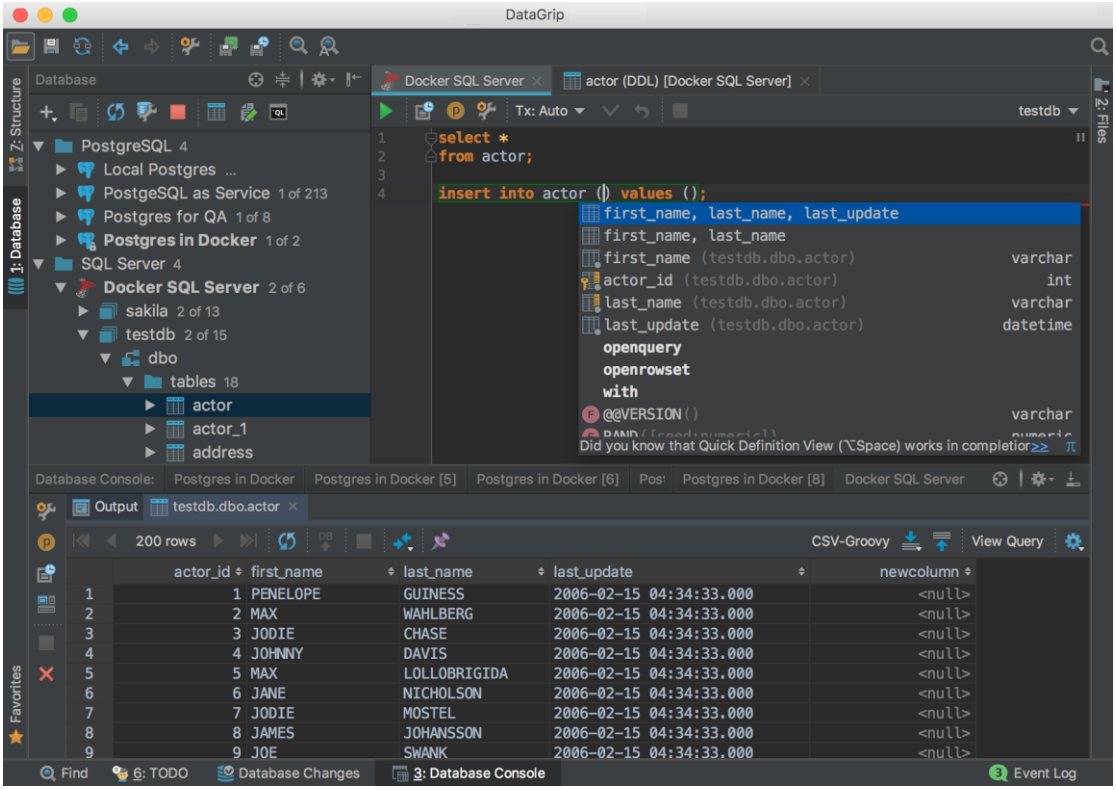

Рисунок 1.7 - Інтерфейс DataGrip

Apache Tomcat – веб-контейнер з відкритим кодом. Це не сервер прикладних програм, як JBoss, Glassfish тощо, які в основному необхідні для корпоративних веб-додатків. Зазвичай складається з EJB та інших важких компонентів J2EE. Tomcat – один з найбільш широко використовуваних веб-серверів розробниками Java. Його поточна версія - 9. Він використовується для запуску веб-додатків. Я використовував Tomcat для запуску таких програм:

— Програми JSP / Servlet.

— Прикладні програми Spring Framework.

— Struts 2 програми.

Maven та Spring Boot мають додатки для цього, які можна використовувати для запуску вбудованого Tomcat для розгортання додатків на локальних машинах.

Java – спочатку була мовою програмування створеною у компанії Sun Microsystems. Сьогодні Oracle володіє Sun, а тому і Java. Таким чином, Java є торговою маркою Oracle. З часом Java перетворилася на більше, ніж просто мову. Це

повна платформа з великою кількістю стандартних API, API з відкритим кодом, інструментами, велика спільнота розробників з мільйонами розробників тощо.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура додатку

Перед початком розробки інформаційної системи необхідно правильно побудувати архітектуру майбутньої системи та розробити логічну структуру для детального проектування системи.

Для базової роботи інформаційної системи потрібен сервер, на якому буде розміщено модулі нашої системи, для цього нас задовольнить звичайний безкоштовний хостинг, але для майбутнього масштабування системи та підтримки швидкості роботи при великих навантаженнях необхідно обрати якісний хостинг з потужними апаратними характеристиками.

Архітектура ІС приведена на зображенні (рис. 2.1) Для роботи з ІС користувачам потрібно відкрити веб-інтерфейс системи використовуючи веб-браузер. Архітектура системи побудована за принципом Клієнт-Сервер. [4]

Завдяки шаблону проектування Модель-Вигляд-Контролер ми маємо змогу відокремити взаємодію користувача безпосередньо з моделлю та передати управління контролеру, а контролер у свою чергу буде повертати необхідні дані користувачу. У такий спосіб ми мінімізуємо можливості проникнення до бази даних на сервері та зменшимо ризик втрати даних.

Користувач через браузер буде надсилати запити до контролеру, контролер у свою чергу буде перевіряти дані користувача на доступ до функціонали та у випадку успішної перевірки повертати необхідну відповідь з серверу.

Архітектура системи складається за таких компонентів:

— Користувачі;

— Веб-інтерфейс;

— API системи(контролери та сервіси роботи з базами даних);

— Бази даних.

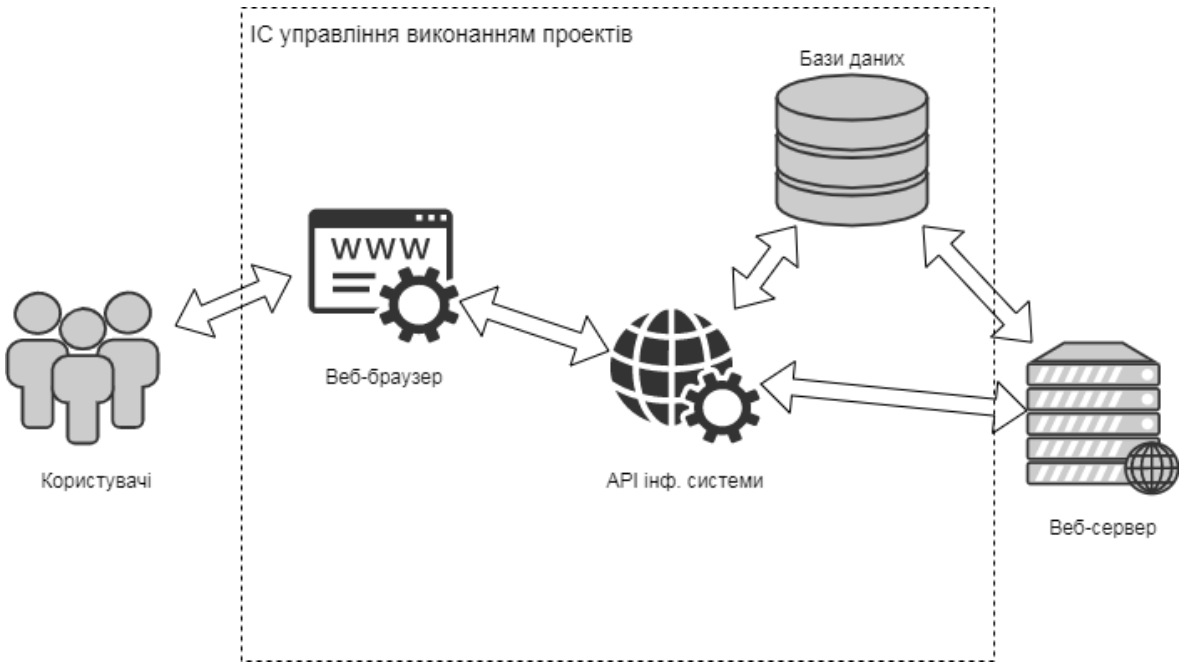

Рисунок 2.1 – Архітектура ІС

2.2 Структурно-функціональне моделювання роботи додатку

Для початку потрібно створити контекстну діаграму, IDEF0, функціональну модель, яка є центром побудови систем та поєднує бізнес процеси компанії.[5]

Центральною моделлю є функція, яка відображена функціональним блоком та розкриває основну логіку роботи системи.[5][21]

Вхідними даними до функції «Управління виконанням проектів» є:

— Інформація про користувача;

— Базова інформація про майбутній проект.

Вихідними даними є:

— Готовий проект як об’єкт для управління;

— Статистика по проекту;

— Звіт з обраного проекту.

Для контекстної діаграми є обов’язковим наявність хоча б одного елементу управління, який поступає згори моделі до функціонального блоку. На даній

діаграмі елементом управління є:

— Функції системи.

Механізмами моделі є:

— ІС;

— Менеджери;

— Розробник проекту;

— Замовник проекту.

На рисунку 2.2 зображено IDEF0 модель, головного процесу інформаційної системи управління виконанням проектів.

Рисунок 2.2 – Діаграма IDEF0

Для більш детального опису роботи інформаційної системи необхідно провести декомпозицію контекстної діаграми.[5]

В результаті аналізу системи мною було зроблено декомпозицію головного бізнес процесу.

Після виконаної роботи система була розділена на такі процеси:

— визначення доступу до системи;
— створення та управління проектом;
— управління дошкою завдань;
— аналіз виконаних задач;
— формування звіту з проекту;
— перегляд статистики проекту.

Процес «Визначення доступу до системи» має такі характеристики:

— Вхідні дані:
    — інформація про користувача.
— Вихідні дані:
    — надання послуг користувачу.
— Механізми:
    — ІС;
    — замовники;
    — менеджери;
    — розробники.

Процес «Створення та управління проектом» має такі характеристики.

— Вхідні дані:
    — файли та інформація проекту;
    — оновлені статистичні дані;
    — обрання проекту.
— Вихідні дані:
    — обраний проект;
    — проект як об’єкт управління.
— Елементи управління:
    — функції системи;
    — надання послуг користувачу.
— Механізми:
    — менеджери;

- ІС;
- розробник проекту.

Процес «Управління дошкою завдань» має такі характеристики.

- Вхідні дані:
  - інформація про задачі проекту.
- Вихідні дані:
  - інформація про задачі проекту.
- Механізми:
  - менеджери;
  - розробник проекту.

Процес «Аналіз виконаних задач» має такі характеристики.

- Вхідні дані:
  - інформація про задачі проекту.
- Вихідні дані:
  - оновлені статистичні дані.

Процес «Формування звіту з проекту» має такі характеристики.

- Вхідні дані:
  - обраний проект.
- Вихідні дані:
  - звіт з обраного проекту.
- Механізми:
  - менеджери.

Процес «Перегляд статистики проекту» має такі характеристики.

- Вхідні дані:
  - обраний проект.
- Вихідні дані:
  - Статистика по проекту.
- Механізми:
  - менеджери;
  - замовник проекту.

На рис. 2.3 зображено IDEF діаграму декомпозиції головного процесу.

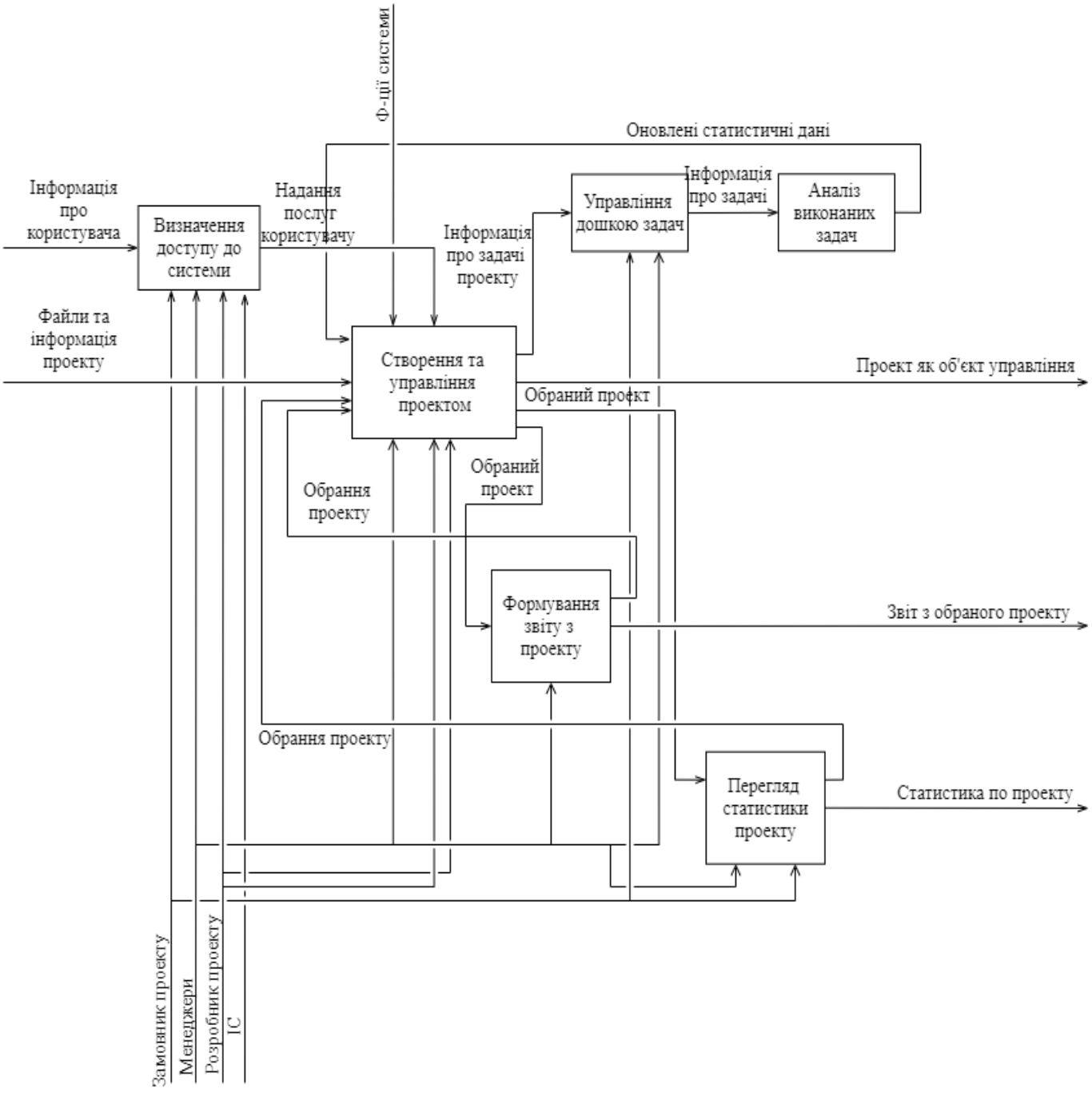

Рисунок 2.3 – Діаграма декомпозиції

2.3 Проектування бази даних

Для зберігання даних користувачів та оновлення чи видалення даних використовується база даних. Щоб продемонструвати структуру бази даних необхідно створити ER діаграму.

Діаграма ER показує взаємозв'язок між сутностями, де сутність - це таблиця або атрибут таблиці в базі даних, тому, показуючи взаємозв'язок між таблицями та їх атрибутами, ER-схема показує повну логічну структуру бази даних. [8]

Сутностями системи виступають:

— user – сутність для моделювання базового об’єкту користувача;

— city – сутність для зберігання назв міст;

— country – сутність для зберігання назв країн;

— role – сутність, яка зберігає назви рівнів доступу у системі;

— team – сутність для зберігання команди як об’єкту;

— project – сутність, яка зберігає об’єкт проекту.

На рисунку 2.4 приведено схему бази даних інформаційної системи управління проектами.

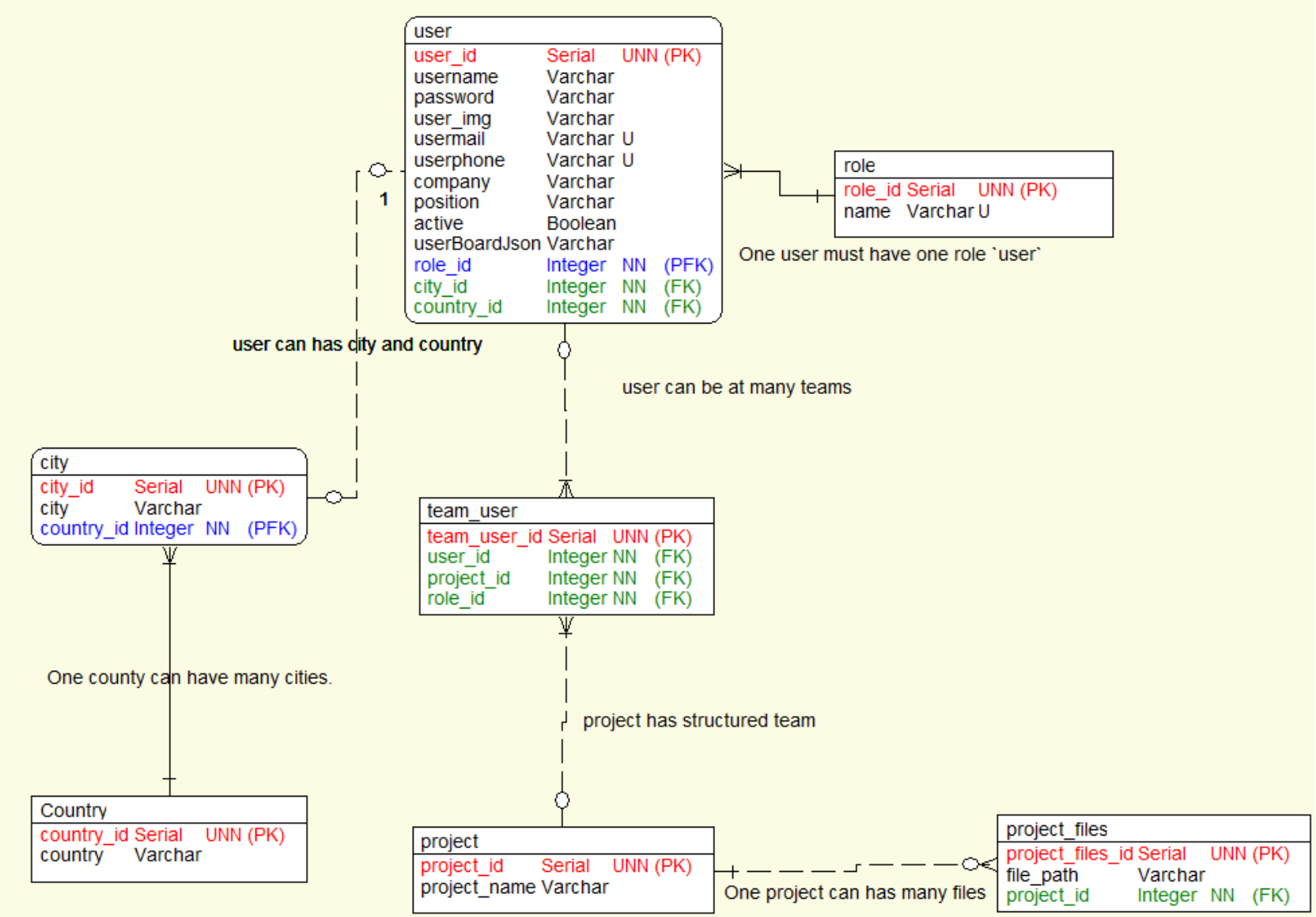

Рисунок 2.4 – ЕР діаграма бази даних додатку

2.4 Моделювання варіантів використання

Після моделювання базових бізнес процесів проекту я перейшов до створення діаграми варіантів використання, яка продемонструє послідовність дій при роботі з системою. [6]

UML є стандартизованою мовою моделювання, що складається з набору діаграм, розроблених, щоб допомогти розробникам у визначенні та візуалізації програмних систем. UML представляє графічні позначення для побудови архітектурного дизайну програмних продуктів.

Діаграму варіантів використання є сенс будувати під час вивчення технічного завдання, вона складається з графічної діаграми, яка описує дійові особи і прецеденти, а також специфікації, що представляє собою текстовий опис конкретних послідовностей дій (потоку подій), які виконує користувач при роботі з системою.[6]

Елементами зовнішнього зв’язку на діаграмі використання(рис. 2.5) виділено сутність для зовнішнього зв’язку у вигляді Бази даних, зв’язок з нею виникає за необхідністю користувача. До бази даних заноситься особиста інформація користувача, інформація про проекти, статистичні дані.

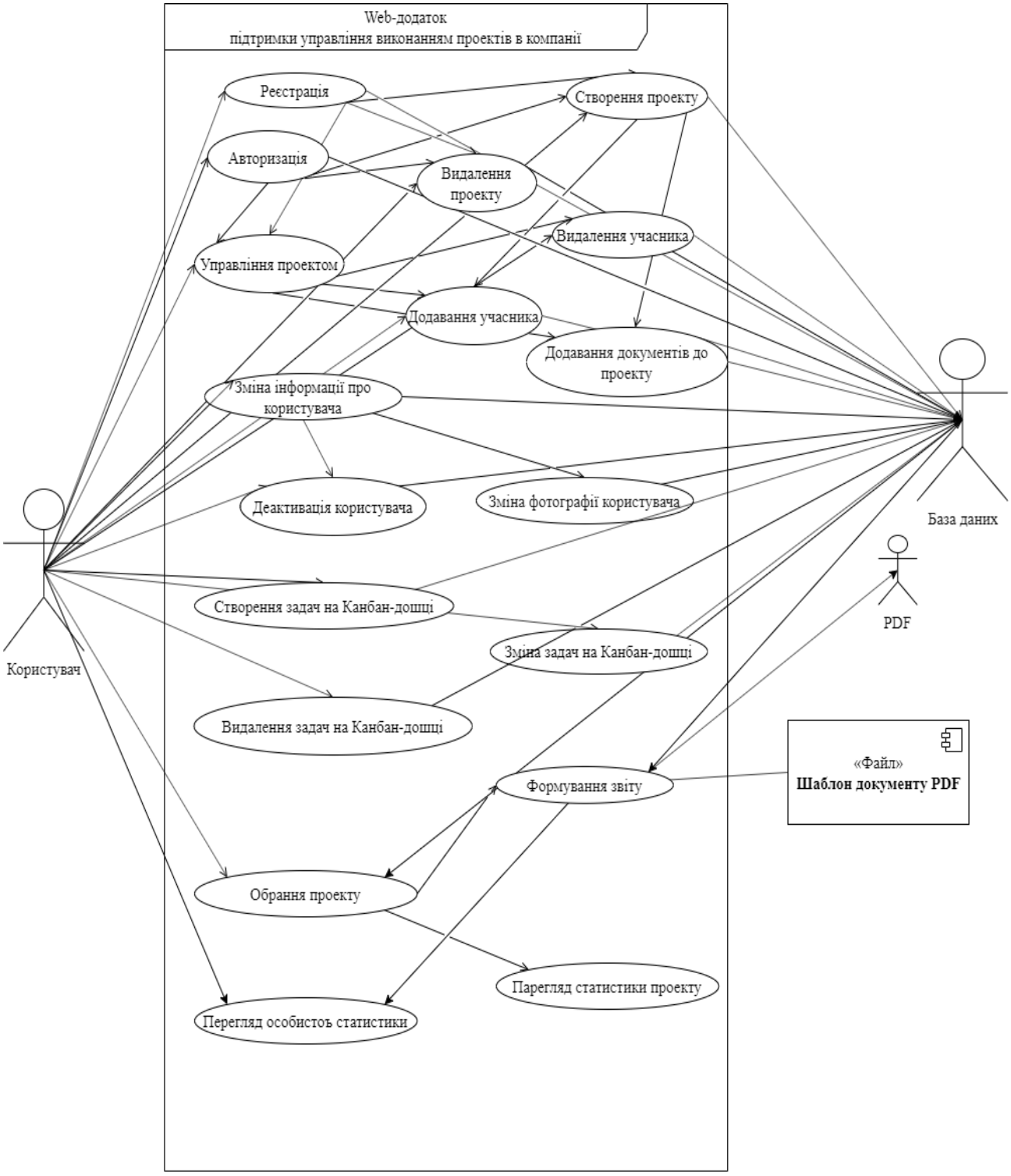

Рисунок 2.5 – Діаграма варіантів використання

Актором на діаграмі виступає користувач, який має можливість реєстрації або авторизації, якщо вже має акаунт, може створювати проекти, додавати інформацію про проект та змінювати учасників, в залежності від прав доступу на проекті. Користувач має змогу створювати задачі на панелі задач, видаляти та змінювати їх,

обравши проект може створити звіт з обраного проекту маючи права, також є можливість перегляду статистики по кожному з проектів або особистої статистики користувача.

Варіантами використання є:

— ВВ – Реєстрація – дозволяє користувачу зареєструватись у системі та отримати доступ до функціоналу;

— ВВ – Авторизація – дозволяє користувачу авторизуватись у системі та подовжити роботу під існуючим акаунтом;

— ВВ – Створення проекту – дозволяє створити проект та почати роботу;

— ВВ – Управління проекту – дозволяє змінювати інформацію про проект, якщо на це є права;

— ВВ – Видалення проекту – дозволяє користувачу видалити проект з системи та видалити всі існуючі файли та інформацію, якщо на це є права доступу;

— ВВ – Додавання учасника – дозволяє користувачу додавати учасників до команди, якщо є права на ці дії;

— ВВ – Видалення учасника – дозволяє видалити учасника з команди проекту та обмежити доступ до інформації проекту;

— ВВ – Додавання документів до проекту – користувач має змогу додавати файли та інформацію до проекту;

— ВВ – Зміна інформації про користувача – дозволяє користувачу оновлювати персональну інформацію про себе;

— ВВ – Зміна фотографії користувача – дозволяє користувачу оновлювати власну фотографію;

— ВВ – Деактивація користувача – функція дозволяє користувачу видалити власний акаунт;

— ВВ – Створення задач на Канбан-дошці – дозволяє створити задачу на дошці завдань;

— ВВ – Зміна задач на Канбан-дошці – дозволяє змінити інформацію задачі;

— ВВ – Видалення задач на Канбан-дошці – дозволяє видалити інформацію з дошки;

— ВВ – Обрання проекту – функція я виконується перед етапами формування звіту та статистики, дозволяє користувачу обрати проект для перегляду або формування необхідної інформації;

— ВВ – Формування звіту – формується звіт з обраного проекту;

— ВВ – Перегляд статистики проекту – дозволяє переглядати статистику з обраного проекту;

— ВВ – Перегляд особистої статистики – дозволяє переглядати власну статистику користувача.

2.5 Моделювання діаграм діяльності

Діаграма діяльності - це діаграма поведінки, яка показує процес роботи об’єкта з урахуванням умов потоку виконання. Дії у моделях діяльності можуть створюватись або через закінчення попередніх дій або через зовнішні чинники. [7]

На рисунку 2.6 зображено діаграму діяльності модулю авторизації.

Рисунок 2.6 – Діаграма діяльності модулю авторизації

На рисунку 2.7 зображено діаграму діяльності модулю реєстрації.

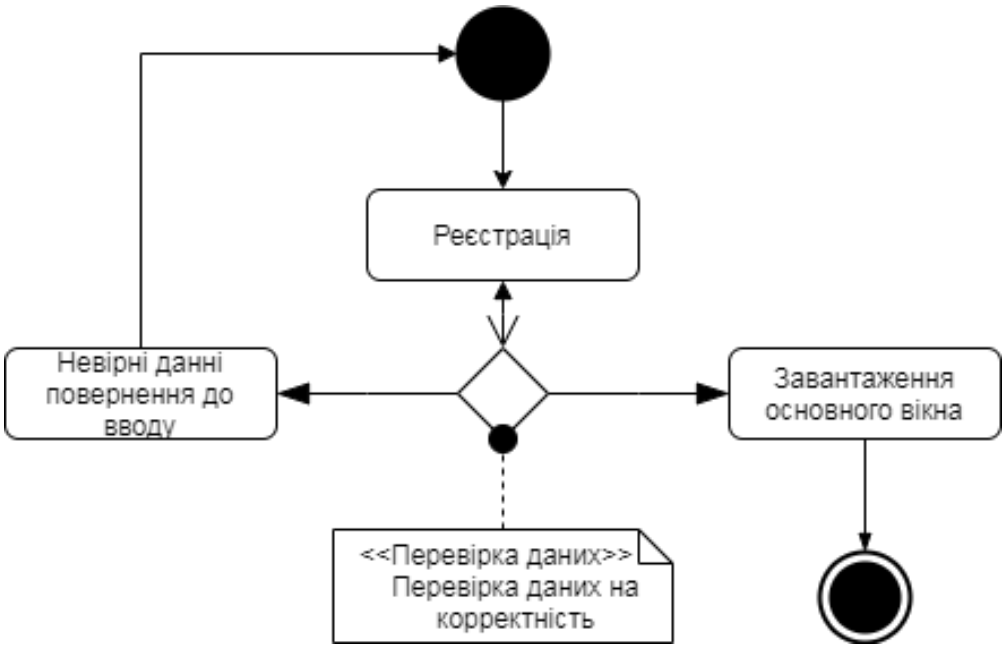

Рисунок 2.7 – Діаграма діяльності модулю реєстрації

На рисунку 2.8 зображено діаграму діяльності модулю зміни інформації користувача.

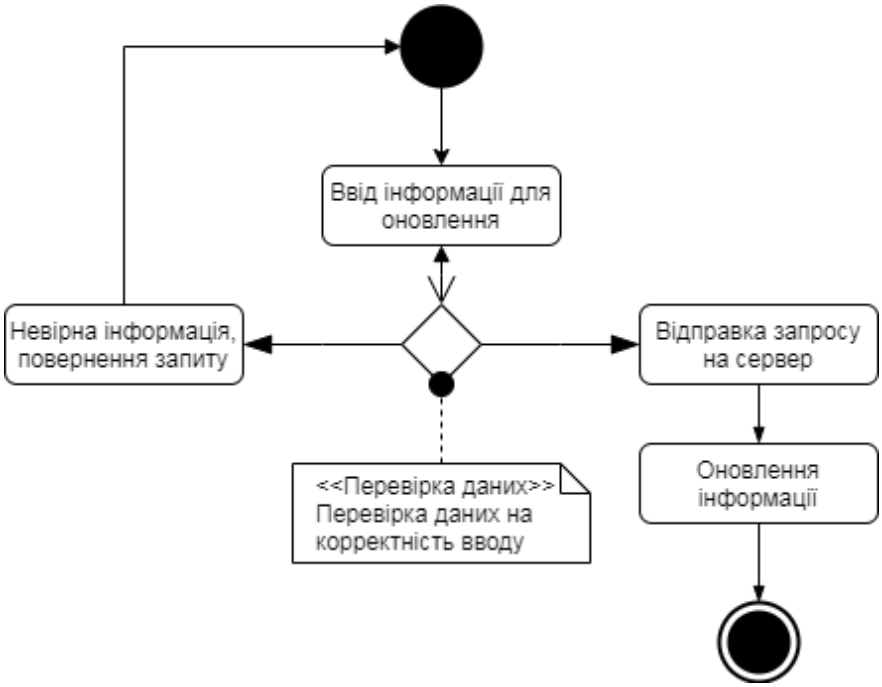

Рисунок 2.8 – Діаграма діяльності модулю зміни інформації користувача

На рисунку 2.9 зображено діаграму діяльності модулю управління проектами.

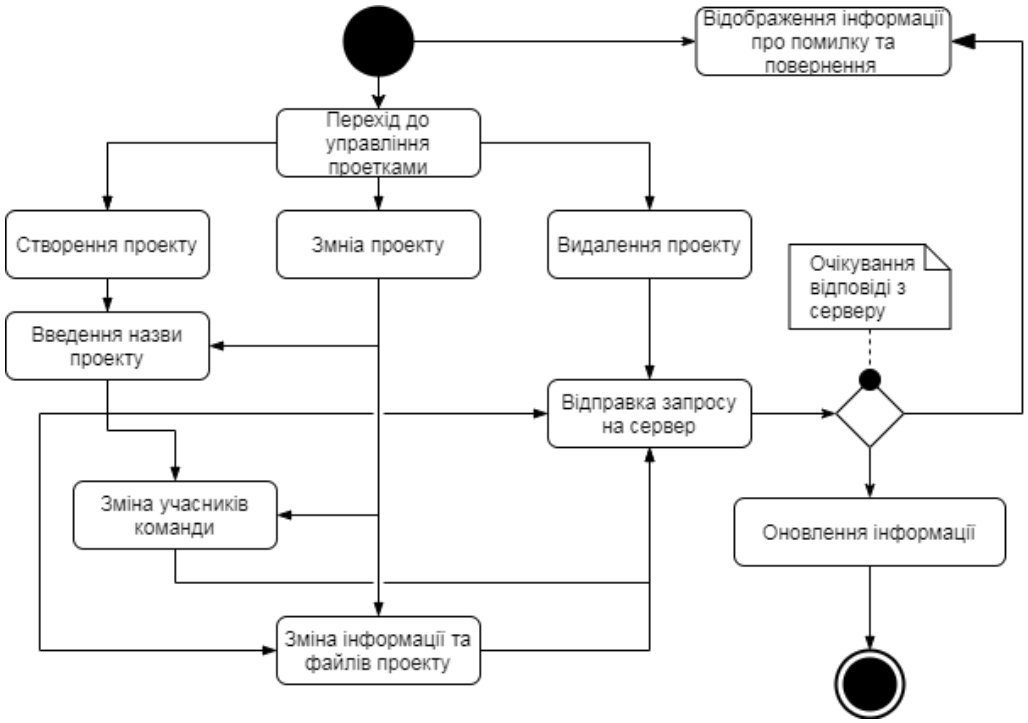

Рисунок 2.9 – Діаграма діяльності модулю управління проектами

На рисунку 2.10 зображено діаграму діяльності модулю створення звітів.

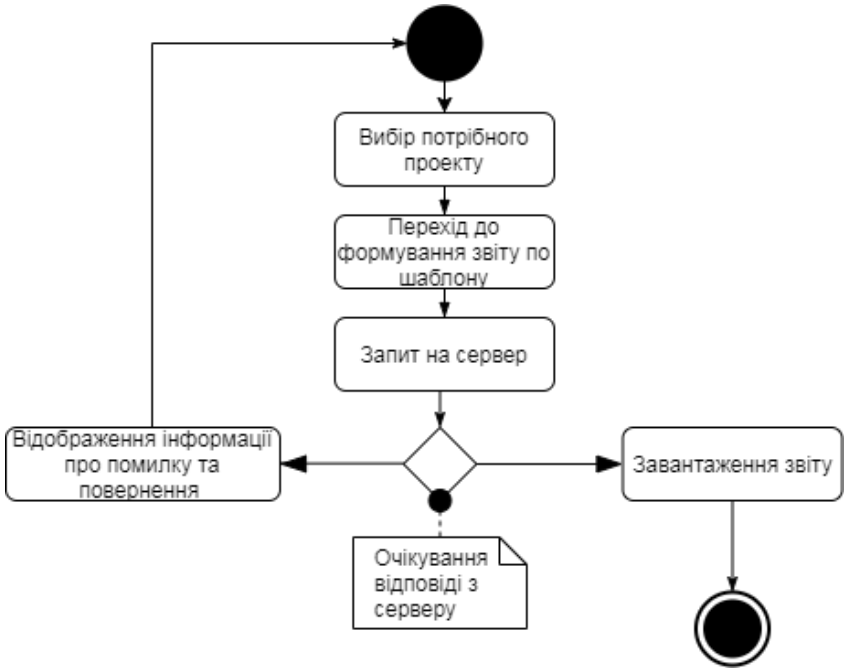

Рисунок 2.10 – Діаграма діяльності модулю створення звітів

На рисунку 2.11 зображено діаграму діяльності модулю перегляду статистики.

Рисунок 2.11 – Діаграма діяльності модулю перегляду статистики

2.6 Моделювання діаграми класів

Клас - це шаблон, який використовується для створення об’єкта. Клас визначає, що об’єкт може робити, дає огляд програмної системи шляхом відображення класів, атрибутів, операцій та їх взаємозв'язків. Ця діаграма включає в себе назву класу, атрибути та операції в окремих призначених відсіках.

Діаграма класів визначає типи об’єктів у системі та різні типи зв’язків, що існують серед них. Це дає перегляд програми на високому рівні. [9]

На рисунку 2.12 та 2.13 представлено діаграму класів системи, з детальними прикладами зв’язків між ними.

Рисунок 2.12 – Діаграма класів модулю роботи з базою даних

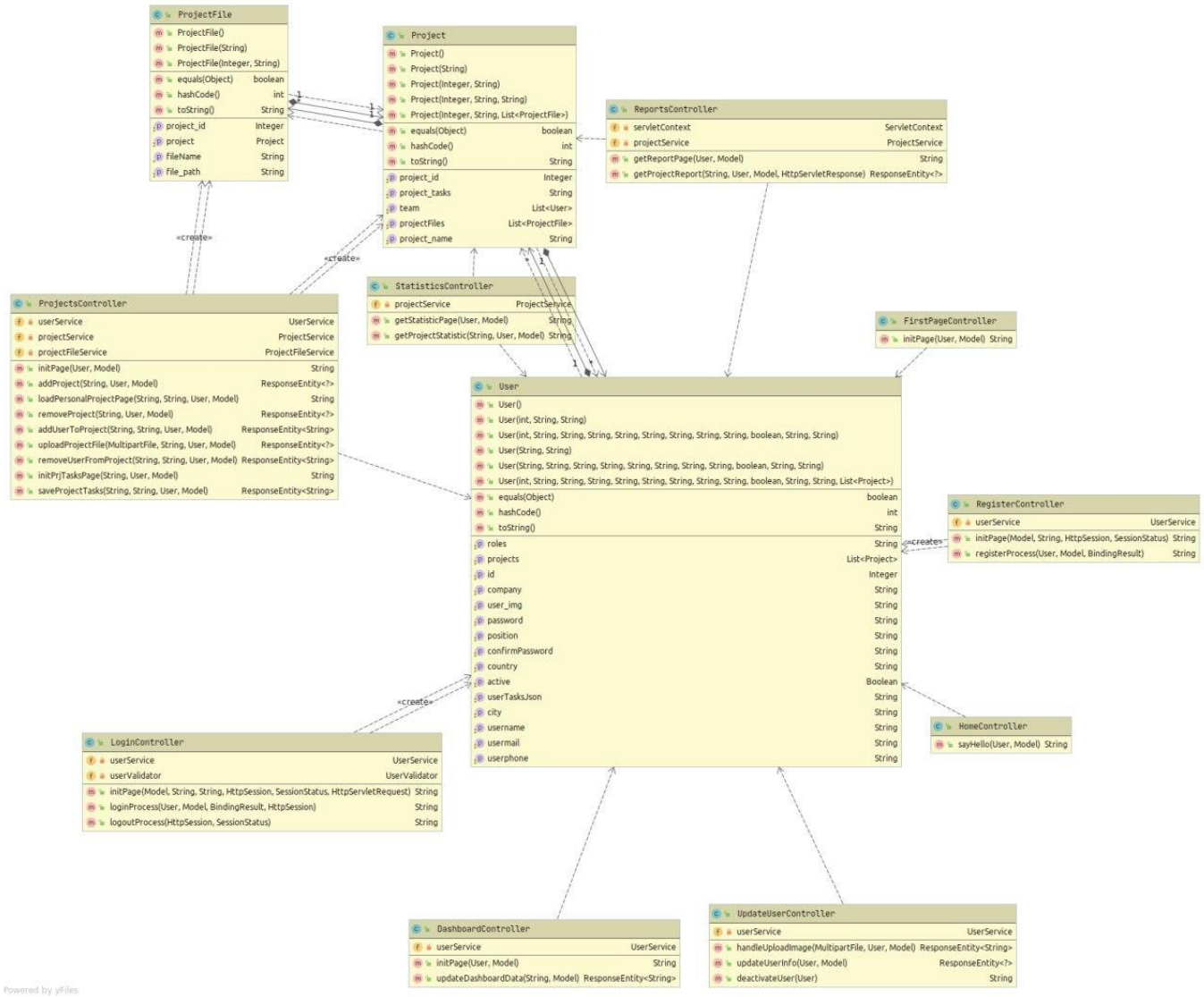

Рисунок 2.13 – Діаграма класів бізнес-логіки додатку

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Структура програмного додатку

Пройшовши етап моделювання та проектування додатку необхідно розпочати розробку структури програмного продукту.

Для побудови структури додатку було виділено два базові етапи, а саме: обрання найбільш вдалих технологій та побудова прототипу.

Згідно вибору, описаному в пункті 2.2 розробка продукту виконується мовою Java.

Для створення та обробки веб-запитів користувача був обраний Spring Framework для Java. Spring за допомогою модулів Spring Data, Spring MVC та Spring Security забезпечує можливість швидкої побудови веб-додатків, налаштування роботи з базою даних, та забезпечення перевірки вхідних даних.

Для зберігання даних було обрано PostgreSQL, ключовими показниками для обирання цієї бази даних були:

— безкоштовність;

— якість збереження даних;

— можливість гнучкого налаштування сховища.

Для розробки за локальний контейнер було обрано Apache Tomcat, використання якого створить імітацію роботи додатку на хостингу від Heroku Cloud.

На другому етапі було розроблено прототип майбутнього веб-додатку, або навігація по системі. При першому відвідуванні системи, користувач потрапляє на сторінку привітання, на сторінці привітання розташовується назва проекту, текст для залучення користувачів та дві кнопки для авторизації чи реєстрації. При натисканні на кнопку авторизації, користувач переходить на сторінку авторизації у системі, де знаходиться форма для введення електронної адреси та паролю користувача вже зареєстрованого у системі. При натисканні на кнопку реєстрації користувач перенаправляєть на сторінку реєстрації, де знаходиться форма для

створення акаунту користувача. Після обрання одного зі способів початку роботи з системою, користувач перенаправляєть до базової сторінки веб-додатку, на якій знаходиться частина статистичних даних користувача, та панель навігації по системі.

На панелі навігації розташовані панель меню для навігації та відображення зображення користувача та його ім’я у системі. Панель меню складається з посилань на ключові модулі системи:

— домашня сторінка;
— сторінка проектів;
— сторінка дошки завдань;
— сторінка формування звітів;
— сторінка перегляду статистики;
— вихід з акаунту користувача системи.

Домашня сторінка складається з форм, які зберігають у собі особисту інформацію користувача, ввівши нові дані, користувач матиме змогу змінити їх.

Сторінка проектів складається з кнопки для створення проектів та дошки вже існуючих проектів. Натиснувши на проект користувач перехожить до особистої сторінки проекту, на якій знаходить три блоки, а саме: дошка завдань, формування звіту з проекту, переглад статисти та блок додавання користувача чи файлу, чи управління командою проекту, у правій частині знаходиться панель з існуючими файлами.

На сторінках формування звіту чи перегляду статистики, користувач має змогу обрати завантаження звіту чи перегляд статистики по обраному проекту на панелі проектів.

3.2 Програмна реалізація

Програмна реалізація – це процес розробки додатку використовуючи обрані технології та засоби реалізації, по вже розробленому плану та структурі проекту.

На етапі програмної реалізації виконується процес підключення модулів додатку до бази даних, створення обробників запитів користувача до серверу та навігації по системі. На даному етапі буде описано ключові складові процесу реалізації веб-додатку, а саме:

— пакети для структурування тарозмежування модулів програми;
— класи, як основні складові розробки;
— проміжні файли для доповнення функціоналу системи;
— підключення та робота з базою даних;
— робота зі сховищем серверу.

Код додатку можна переглянути в додатку В.

Середовище Intellij IDEA використовувалося як основний текстовий редактор для розробки системи.

Для початку неоюхідно відобразити основні складові проекту, структуровані за допомогою пакетів та каталогів середовища розробки, у свою чергу це допоможе швидко знаходити компоненти програми та полегшить навігацію по проекту, дерево проекту зображено на рисунку 3.1. Основні частини додатку розташовані у каталогах та пакетах середовища, пакети – це каталог для зберігання .java файлів.

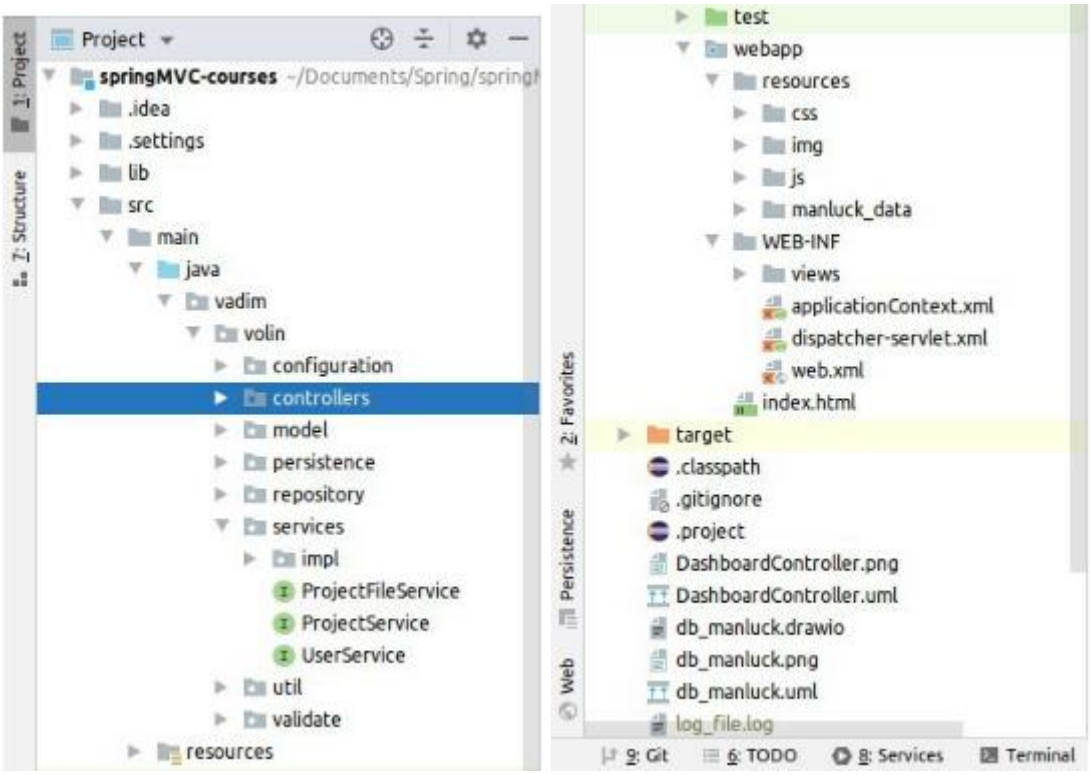

Рисунок 3.1 – Структура проекту

Пакет ***configuration*** – це пакет в якому знаходяться файли конфігурації додатку у інтернет мережі та розмежування рівню доступу.

Пакет ***controllers*** – цей пакет містить класи, які виконують роль обробників запитів користувача та містять основну логіку навігації та завантаження сторінок.

Пакет ***model*** – в цьому пакеті знаходяться класи, які відображають ключові логічні сутності системи.

Пакети ***persistence*** – в пакеті розміщені файли для підключення до бази даних.

Пакет ***repository*** – в пакеті знаходяться файли, які виконують базові функції роботи з базою даних та логічними сутностями.

Пакет ***services.impl*** – в пакетах розташовані інтерфейси та їх реалізації для виконання основної бізнес-логіки системи.

Пакет ***util*** – в даному пакеті знаходяться файли для виконання допоміжних процесів.

Пакет ***validate*** – в цьому пакеті розташовані файли, які використовуються для валідації даних.

Пакет ***webapp*** - в цьому каталозі розміщені каталоги та файли відображень.

Пакет ***resources.js*** – у цьому каталозі розташовані файли javascript, які виконують побічні бізнес процеси.

Пакет ***resources.css*** – в цьому пакеті розташовані файли, використовуються для стилізації відображень.

Пакет ***WEB-INF.views*** – в цьому пакеті розташовані файли відображень.

Для більшого розуміння компонентів програми необхідно детально описати файли каталогу проекту. На рисунку 3.2 зображено детальну структуру каталогу проекту з повним обсягом пакетів та каталогів описаних вище та файлів, які там зберігаються.

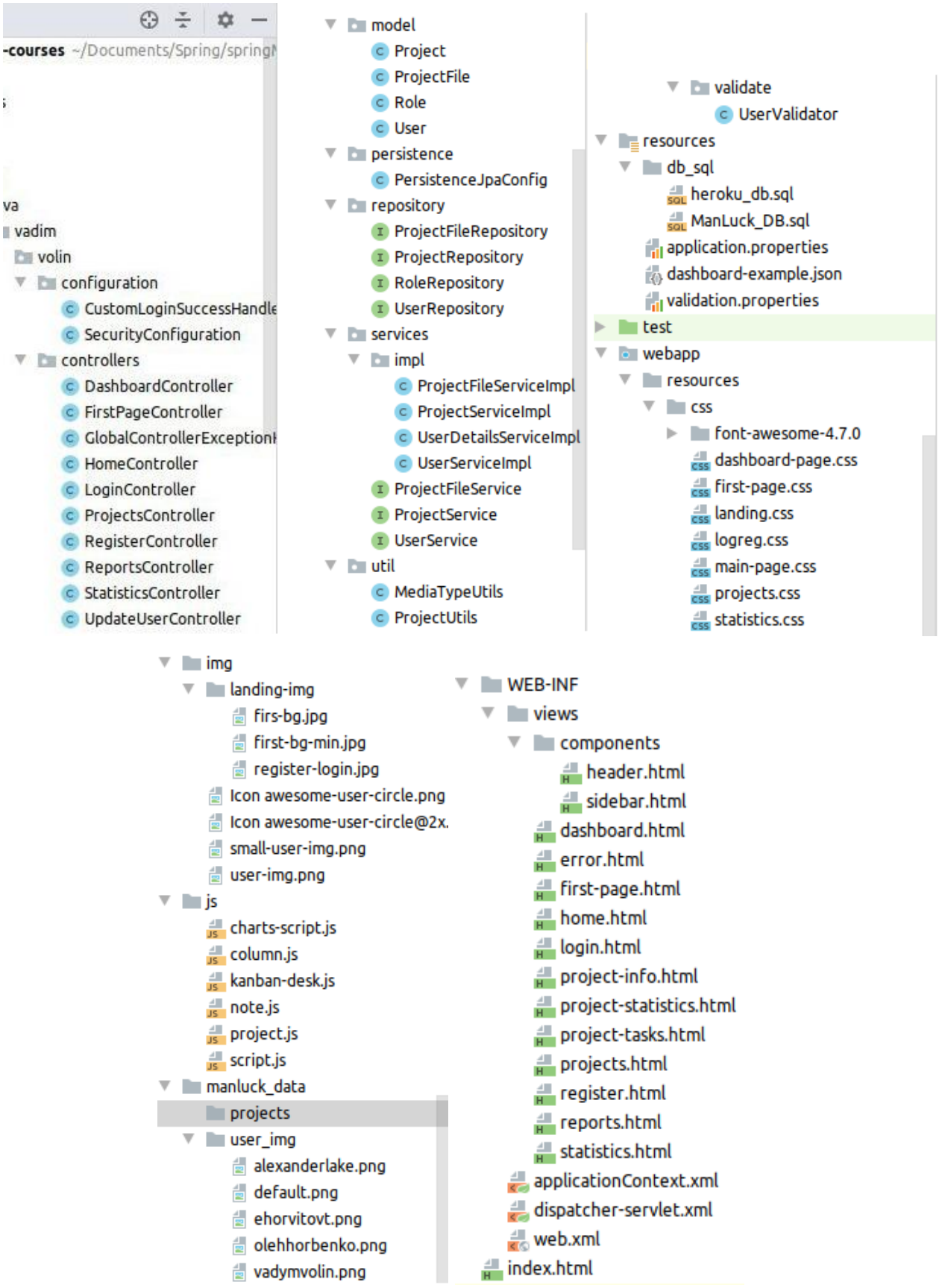

Рисунок 3.2 – Зображення файлів в пакетах

Таблиця 3.1 – Опис файлів, які знаходяться в пакеті configuration

Пакет (и)	Клас (и), файл (и)	Метод (и)
configuration	***CustomLoginSuccessHandler*** – класс для реагування на процес авторизації у системі.	***handle ()*** – метод викликається під час авторизації для перевірки та переадресації. ***determineTargetUrl ()*** – виконує процес валідації користувача.
	SecurityConfiguration – класс налаштувань перевірок та процесів реагування на запити користувачів.	***daoAuthenticationProvider ()*** – встановлення базового провайдера авторизації. ***userDetailsService ()*** – створює компонент деталізації даних для авторизації. ***bCryptPasswordEncoder ()*** – створює компонент для шифрування та дешифрування паролю. ***persistentTokenRepository ()*** – зберігає дані у пам’ять машини. ***configure ()*** – налаштування перевірок та розмежування доступу.

Таблиця 3.2 – Опис файлів, які знаходяться в пакеті controllers

Пакет (и)	Клас (и), файл (и)	Метод (и)
controllers	***DashboardController*** – клас контроллер для сторінки завдань.	***initPage(@ModelAttribute User user, Model model)***– метод ініціалізації сторінки. ***updateDashboardData(@RequestBody String jsonTasks, Model model)***– метод оновлення даних дошки задач.
	FirstPageController – клас контроллек для вхідної сторінки.	***initPage(@ModelAttribute User user, Model model)***– метод ініціалізації сторінки.
	GlobalControllerExceptionhandler – клас обробник базових помилок.	***handleHttpSessionRequiredException ()*** – метод обробки помилки сесії сторінки. ***handleNotFound()*** – метод обробки помилки сторінки. ***NotFoudPage(Model model)*** – метод обробки помилки сторінки.
	HomeController – клас для сторінки особистої інформації користувача.	***sayHello(@ModelAttribute User user, Model model)***– метод ініціалізації сторінки.

Продовження таблиці 3.2

controllers	***LoginController***– клас контроллер для модулю авторизації.	***initPage(Model model, String error, String logout, HttpSession httpSession, SessionStatus sessionStatus, HttpServletRequest request)***– метод ініціалізації сторінки. ***loginProcess(@ModelAttribute User user, Model model, BindingResult bindingResult, HttpSession httpSession)***– метод авторизації у системі. ***sayHello(@ModelAttribute User user, Model model)***– метод ініціалізації сторінки. ***logoutProcess(HttpSession httpSession, SessionStatus sessionStatus)***– метод виходу з системи.
	ProjectsController– клас контроллер для сторінки проектів.	***initPage(@ModelAttribute User user, Model model)*** **–** метод ініціалізації сторінки. ***addProject(@RequestParam(name = "project_name", defaultValue = "") String project_name, @ModelAttribute User user, Model model)*** **–** метод додавання проекту. ***loadPersonalProjectPage(@PathVariable String user_id, @PathVariable String project_id, @ModelAttribute User user, Model model)*** **–** завантаження сторінки проекту. ***removeProject(@RequestParam(name = "projectId", defaultValue = "") String projectId, @ModelAttribute User user, Model model*** ***)*** **–** видалення проекту.

Продовження таблиці 3.2

controllers		***addUserToProject(@RequestParam(name = "membermail", defaultValue = "") String membermail, @PathVariable String id, @ModelAttribute User user, Model model)*** – додавання користувача до проекту. ***uploadProjectFile(@RequestBody MultipartFile file, @PathVariable String id, @ModelAttribute User user, Model model)***– додавання файлу до проекту проекту. ***removeUserFromProject(@RequestParam(name = "projectId", defaultValue = "") String projectId, @RequestParam(name = "usermail", defaultValue = "") String usermail, @ModelAttribute User user, Model model)*** – видалення користувача з проекту. ***initPrjTasksPage(@PathVariable String project_id, @ModelAttribute User user, Model model)*** – завантаження дошки завдань проекту. ***saveProjectTasks(@RequestBody String jsonTasks, @PathVariable String project_id, @ModelAttribute User user, Model model)*** – збереження дошки завдань проекту.
	RegisterController – клас контроллер для модулю реєстрації.	***initPage(Model model, String error, HttpSession httpSession, SessionStatus sessionStatus)*** – метод , завантажує сторінку реєстрації. ***registerProcess(@ModelAttribute("user") User user, Model model, BindingResult bindingResult)*** – метод , реєстрації користувача у системі.

Продовження таблиці 3.2

controllers	***ReportsController*** – клас контроллер для сторінок звітів.	***getProjectReport(@PathVariable String project_id, @ModelAttribute User user, Model model, HttpServletResponse response)*** – метод , котрий розрахований для генерації звіту з проекту.
	StatisticsController– клас контроллер для сторінок статистики.	***getStatisticPage (@ModelAttribute User user, Model model)*** – метод , котрий розрахований для декількох кнопок, тобто анімація буде працювати для декількох кнопок. ***getProjectStatistic (@PathVariable String project_id, @ModelAttribute User user, Model model)*** – метод , котрий розрахований для декількох кнопок, тобто анімація буде працювати для декількох кнопок.
	UpdateUserController– клас контроллер для модулю зміни особистої інформації.	***handleUploadImage (@RequestBody MultipartFile file, @ModelAttribute User user, Model model)*** – метод , котрий розрахований для зміни зображення користувача. ***updateUserInfo (@ModelAttribute User user, Model model)*** – метод , котрий розрахований для зміни особистої інформації користувача. ***deactivateUser (@ModelAttribute User user)*** – метод , котрий розрахований для деактивації користувача у системі.

Таблиця 3.3 – Опис файлів, які знаходяться в пакеті model

Пакет (и)	Клас (и), файл (и), інтерфейс (и)	Метод (и)
model	***Project*** **–** логічна сутність проекту.	Стандартні методи об’єкту Pojo.
	ProjectFile – логічна сутність проектного файлу.	Стандартні методи об’єкту Pojo.
	Role – логічна сутність проектної ролі.	Стандартні методи об’єкту Pojo.
	User **–** логічна сутність користувача системи	Стандартні методи об’єкту Pojo.

Таблиця 3.4 – Опис файлів, які знаходяться в пакеті persistence

Пакет (и)	Клас (и)	Метод (и)
persistence	***PersistenceJpaConfig*** – клас, який описує підключення та логіку роботи з базою даних при підключенні	***entityManagerFactory()*** – метод повертає менеджер сутностуй для обробки логічних компонентів. ***transactionManager(LocalContainerEntity ManagerFactoryBean entityManagerFactoryBean)*** – метод, який повертає менеджер транзакцій фреймворку. ***exceptionTranslation()*** – метод, який помилки роботи з базою даних на фреймворк контекст. ***getHibernateProperties()*** – метод, який повертає властивості для сериалізування. ***dataSource()*** – метод, який повертає об’єкт підключення до бази даних.

Таблиця 3.5 – Опис файлів, які знаходяться в пакеті repository

repository	***ProjectFileRepository*** – інтерфейс, який відповідає за роботу з об'єктами та базою даних.	Немає методів.
	ProjectRepository – інтерфейс, який відповідає за роботу з об'єктами та базою даних.	Немає методів.
	RoleRepository – інтерфейс, який відповідає за роботу з об'єктами та базою даних.	Немає методів.
	UserRepository – інтерфейс, який відповідає за роботу з об'єктами та базою даних.	Немає методів.

Таблиця 3.6 - Опис файлів, які знаходяться в пакеті services

Пакет (и)	Клас (и), файл (и)	Метод (и)
services	***UserService*** – інтерфейс, який описує базові CRUD операції з логічною сутністю користувача.	***addUser(User user)*** – додати користувача до бази даних. ***delete(int id)*** – видалити користувача по ідентифікатору.

Продовження таблиці 3.6

services		***getByUserName(String username)*** – знайти користувача по імені. ***getByUserMail(String usermail)*** – знайти користувача по електронній адресі. ***editUser(User user)*** – змінити користувача. ***getAllUser()*** – взяти список користувачів.
	ProjectService– інтерфейс, який описує базові CRUD операції з логічною сутністю проекту.	***addProject(Project project)*** – додати проект до бази даних. ***removeProject (int id)*** – видалити проект по ідентифікатору. ***removeProject(Project project)*** – видалити проект. ***editProject(Project project)*** – змінити проект. ***getProject(int id)*** – взяти проект по ідентифікатору. ***getAll()*** – взяти список проектів.

Продовження таблиці 3.6

services	***ProjectFileService –*** інтерфейс, який описує базові CRUD операції з логічною сутністю проектного файлу.	***addProjectFile (ProjectFile projectFile)*** – додати проектний файл до бази даних. ***removeProjectFile (int id)*** – видалити проектний файл по ідентифікатору. ***removeProjectFile (ProjectFile projectFile)*** – видалити проектний файл. ***editProjectFile (ProjectFile projectFile)*** – змінити проектний файл. ***getProjectFile (int id)*** – взяти проектний файл по ідентифікатору. ***getAll()*** – взяти список проектів.

Таблиця 3.7 - Опис файлів, які знаходяться в пакеті util

Пакет (и)	Клас (и), файл (и)	Метод (и)
util	***MediaTypeUtils*** – клас, який відповідає за конвертацію медіа-файлів до веб-контенту.	***getMediaTypeForFileName(ServletContext servletContext, String fileName)***–метод який відповідає за конвертацію медіа-файлів до веб-контенту.

Продовження таблиці 3.7

util	***ProjectUtils*** – клас парсить рядок формату json та дістає потрібні дані.	***getTasksData(String JSON_TASKS)*** – метод повертає таблицю відповідностей кількості задач та назви колонки. ***getTaskCount(String JSON_TASKS)*** – метод повертає кількість задач у дошці.

Таблиця 3.8 - Опис файлів, які знаходяться в пакеті validate

Пакет (и)	Клас (и), файл (и)	Метод (и)
validate	***UserValidator*** - клас, який відповідає валідацію даних користувача.	***validate(Object o, Errors errors)*** – метод перевірки та реакції на введення даних користувача.

Наступним етапом було створено базу даних на сервісі heroku, сервіс дає змогу користувачам підєднатися до віддаленої бази даних, за допомогою посилання та даних користувача, який є адміністратором бази даних. Приклад створення таблиці наведений на рисунках 3.3 та 3.4.

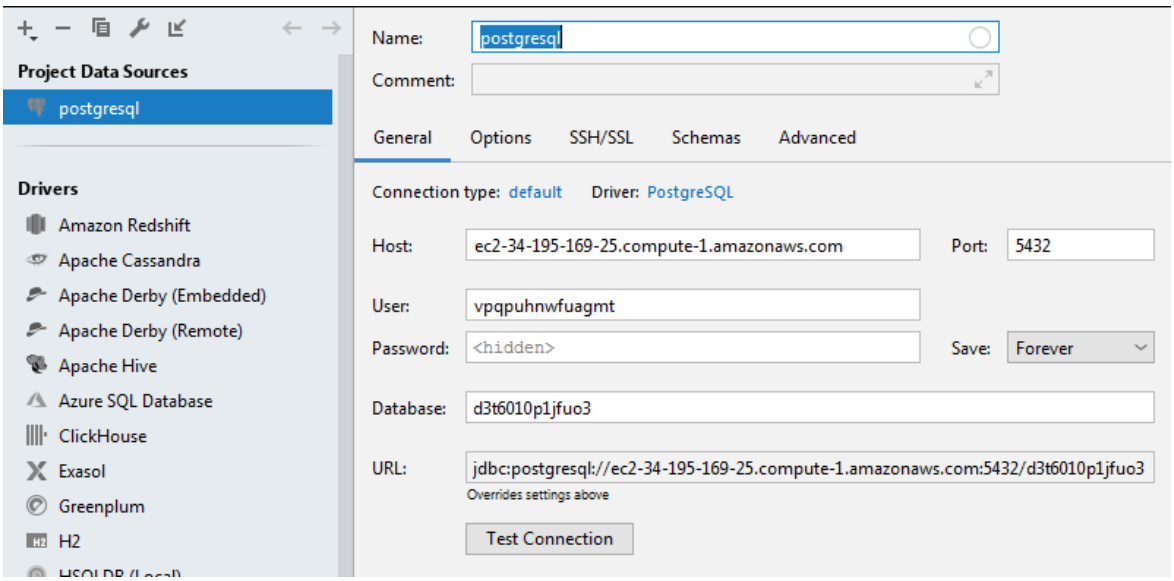

Рисунок 3.3 – Підключення до бази даних

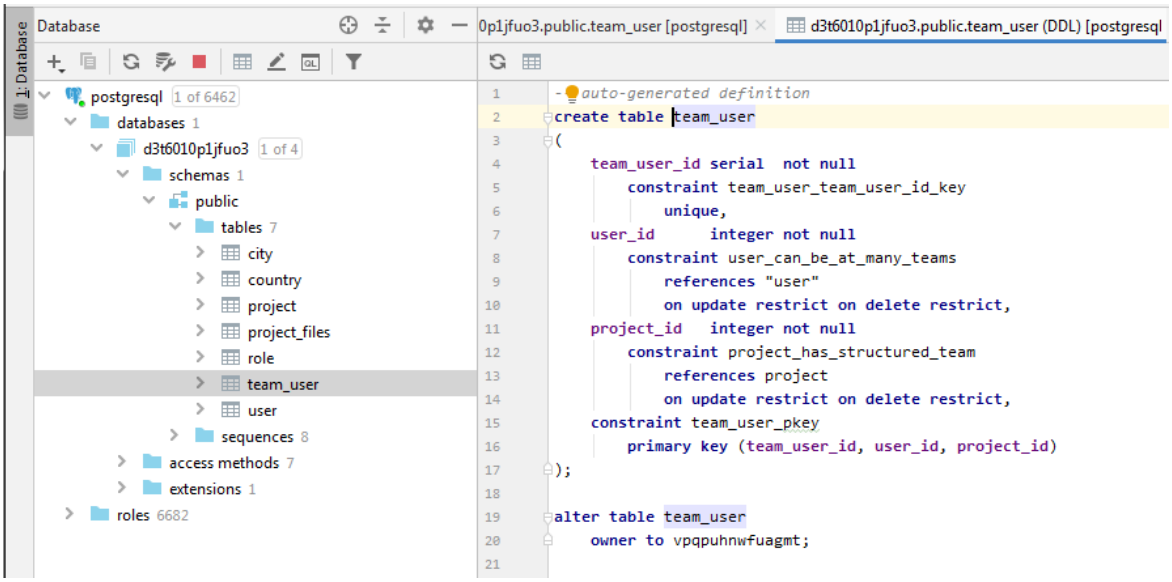

Рисунок 3.4 – Створення таблиці команди проекту

Після проектування та завантаження таблиць до сервісу, було створене підключення до віддаленої бази даних представленої на рисунку 3.5.

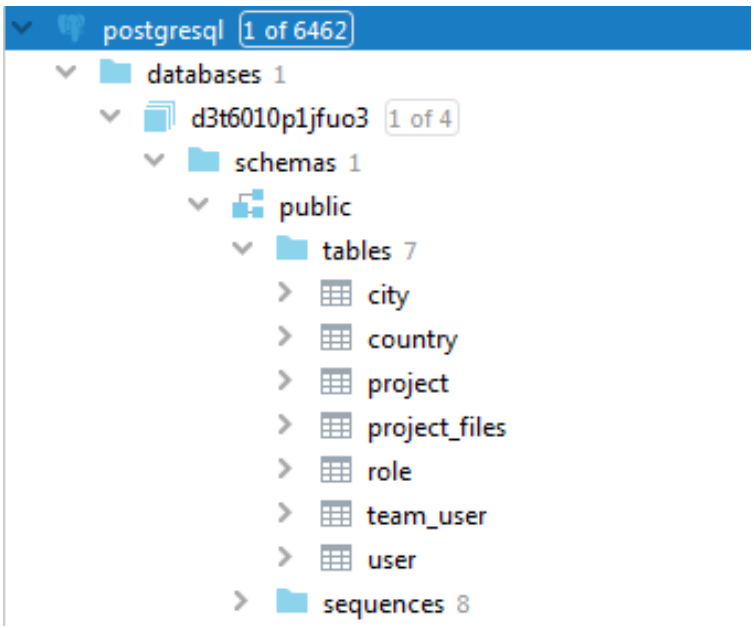

Рисунок 3.5 – База даних

Після шифрування даних сервісом heroku, отримано назву ***d3t6010p1jfuo3***. Таблиці мають назви: ***city, country, project, team_user, user, role, project_files***. Опишемо більш детально кожну із таблиць.

Таблиця 3.9 – Опис таблиці user

Назва таблиці	Опис даних таблиці
user – сутність користувача.	***user_id*** – ідентифікатор користувача.
	username – ім’я користувача.
	password – пароль користувача.
	user_img – шлях до зображення користувача.
	usermail – електронна адреса користувача.
	userphone – телефон користувача.
	company – компанія.
	position – посада користувача.
	active – чи активний акаунт.
	userBoardJson – дошка завдань.

Таблиця 3.10 – Опис таблиці team_user

Назва таблиці	Опис даних таблиці
team_user – збереження даних про запитання.	***team_user_id*** – ідентифікатор команди.
	user_id – ідентифікатор користувача.
	project_id – ідентифікатор проекту.
	role_id – ідентифікатор ролі.
	city_id – ідентифікатор міста.
	county_id – ідентифікатор країни.

Таблиця 3.11 - Опис таблиці country

Назва таблиці	Опис даних таблиці
country – сутність країни.	***country_id*** – ідентифікатор країни.
	country – назва країни.

Таблиця 3.12 – Опис таблиці role

Назва таблиці	Опис даних таблиці
***role* –** сутність ролі.	***role_id*** – ідентифікатор ролі.
	name – назва ролі.

Таблиця 3.13 – Опис таблиці city

Назва таблиці	Опис даних таблиці
***city* –** сутність ролі.	***city_id*** – ідентифікатор міста.
	city – назва міста.
	country_id – ідентифікатор країни.

Таблиця 3.14 – Опис таблиці project

Назва таблиці	Опис даних таблиці
project – таблиця, в якій зберігаються шляхи до відео фрагментів.	***project _id*** – ідентифікатор проекту.
	project_name– назва проекту.

Таблиця 3.15 – Опис таблиці project_files

Назва таблиці	Опис даних таблиці
***project_files* –** сутність ролі.	***project_files _id*** – ідентифікатор файлу.
	file_path – шлях до файлу.
	project_id – ідентифікатор проекту.

Після реалізації бази даних необхідно завантажити створений веб-додаток на сервер. За хостинг та віддалену базу даних було обрано Service Heroku. Сервіс дозволяє безкоштовно на початку створити та використовувати базу даних PostgreSQL, та контейнер веб-додатку на базу Apache.

Знайти додаток можна за посиланням https://manluck.herokuapp.com/.

Завантажений веб-додаток на сервер представлений на рисунку 3.6.

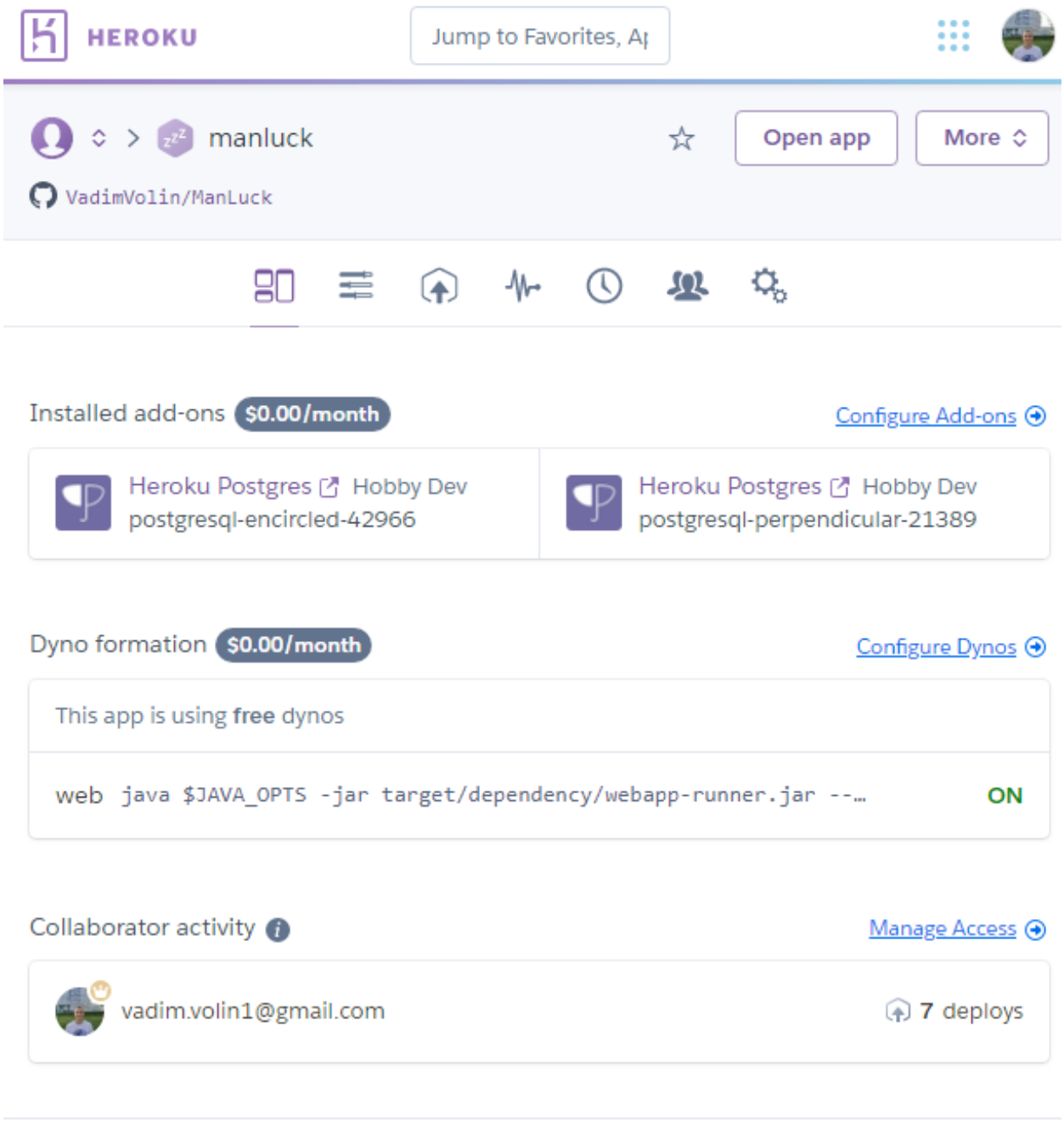

Рисунок 3.6 – Веб-додаток на сервері heroku

3.3 Використання програмного додатку

На даному етапі буде продемонстровано процес роботи програми, навігацію по системі та використання функціональних можливостей системи.

При першому відвідуванні системи користувач потрапить на сторінку привітання. На рисунку 3.7 можна спостерігати дану сторінку. В цьому вікні користувач обирає спосіб входу до системи.

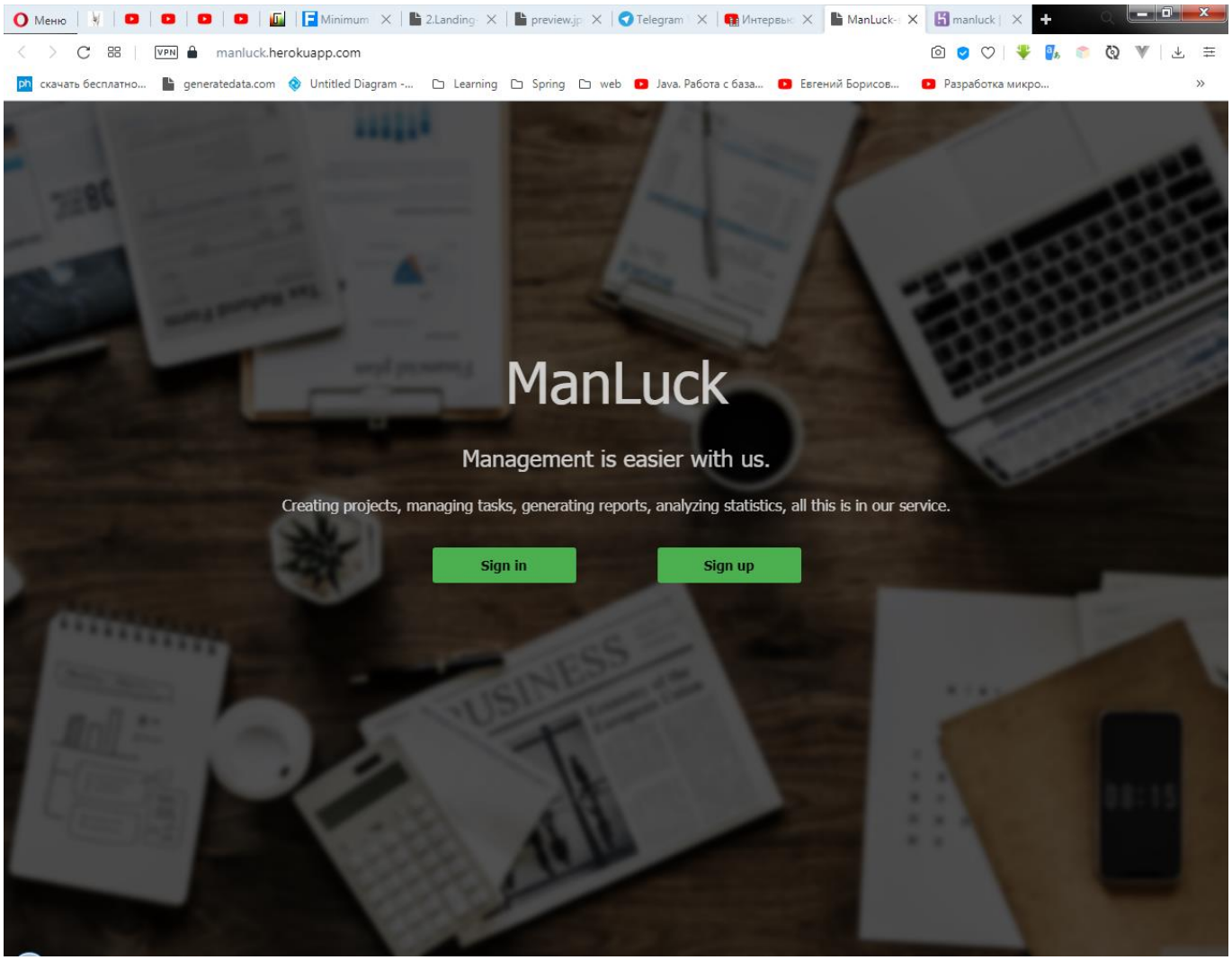

Рисунок 3.7 – Сторінка привітання

Якщо, користувач натиснув на кнопку авторизації, то він буде перенаправлений на сторінку авторизації. На рисунку 3.8 представлена ця сторінка.

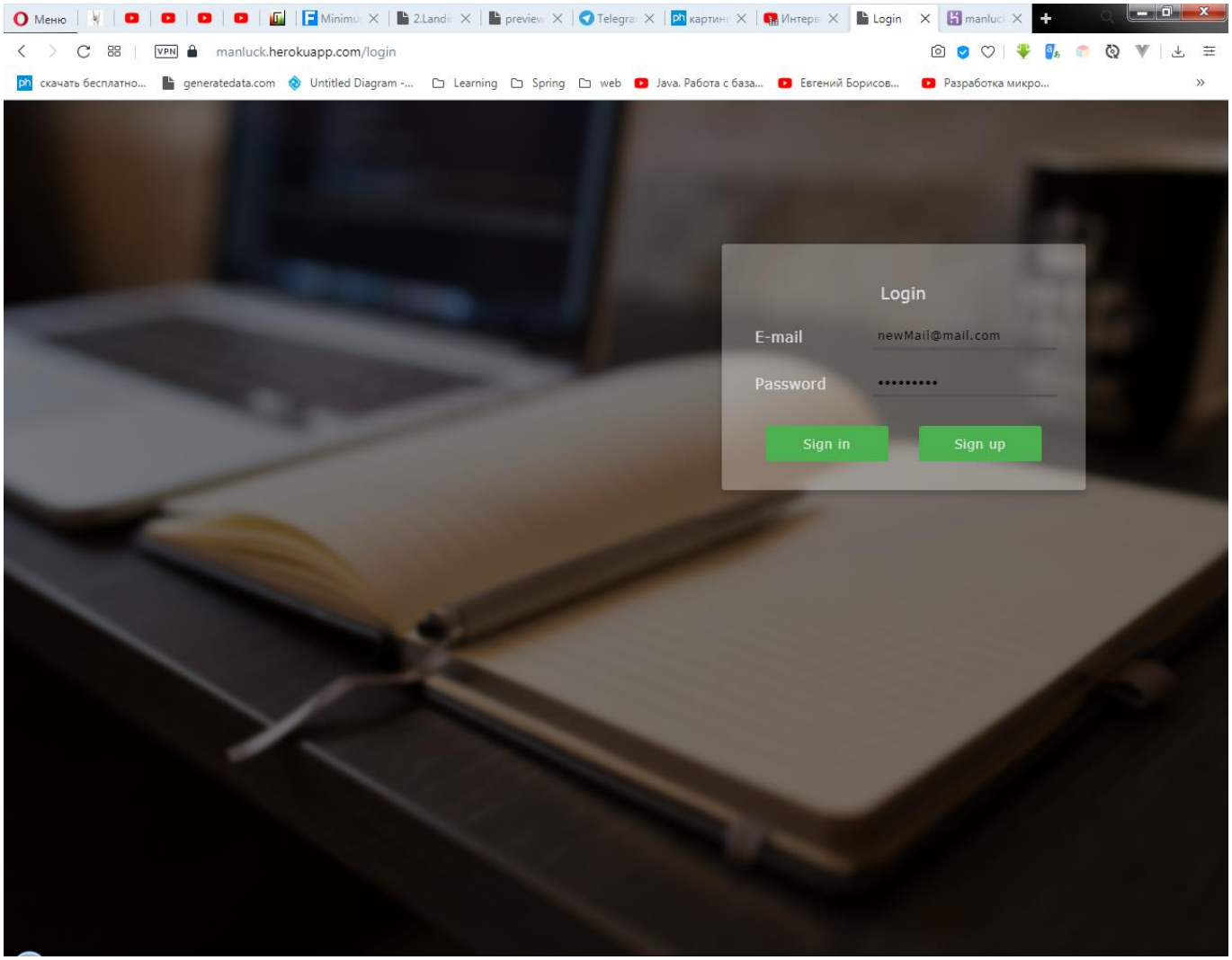

Рисунок 3.8 – Вікно авторизації

Якщо користувач ввів невірні дані або залишив обов’язкове поле пусти, то він отримає попередження. На рисунку 3.9 зображено вікно попередження.

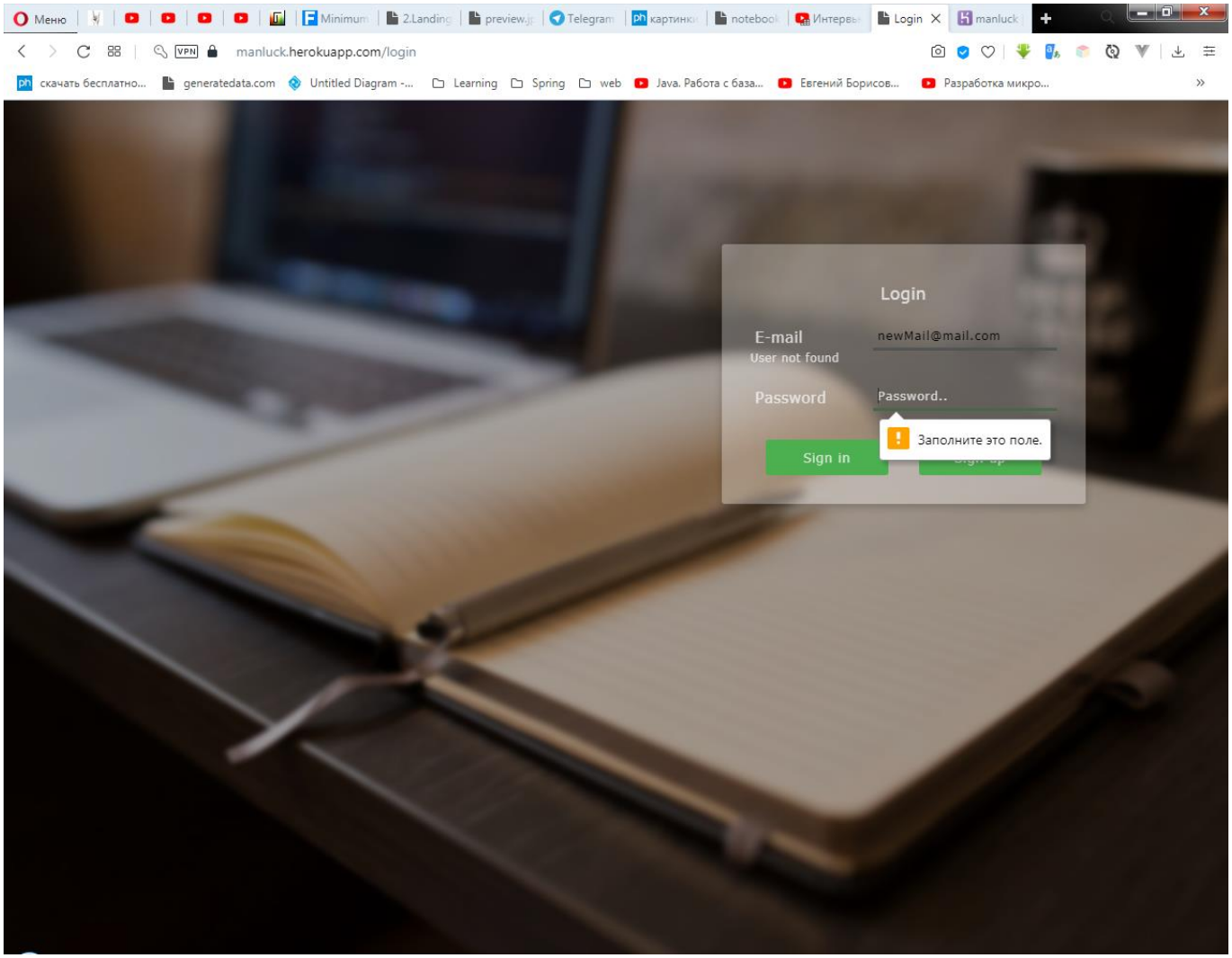

Рисунок 3.9 – Вікно попередження

Якщо користувач обрав реєстрацію, то він буде перенаправлений на сторінку реєстрації, яку зображено на рисунку 3.10.

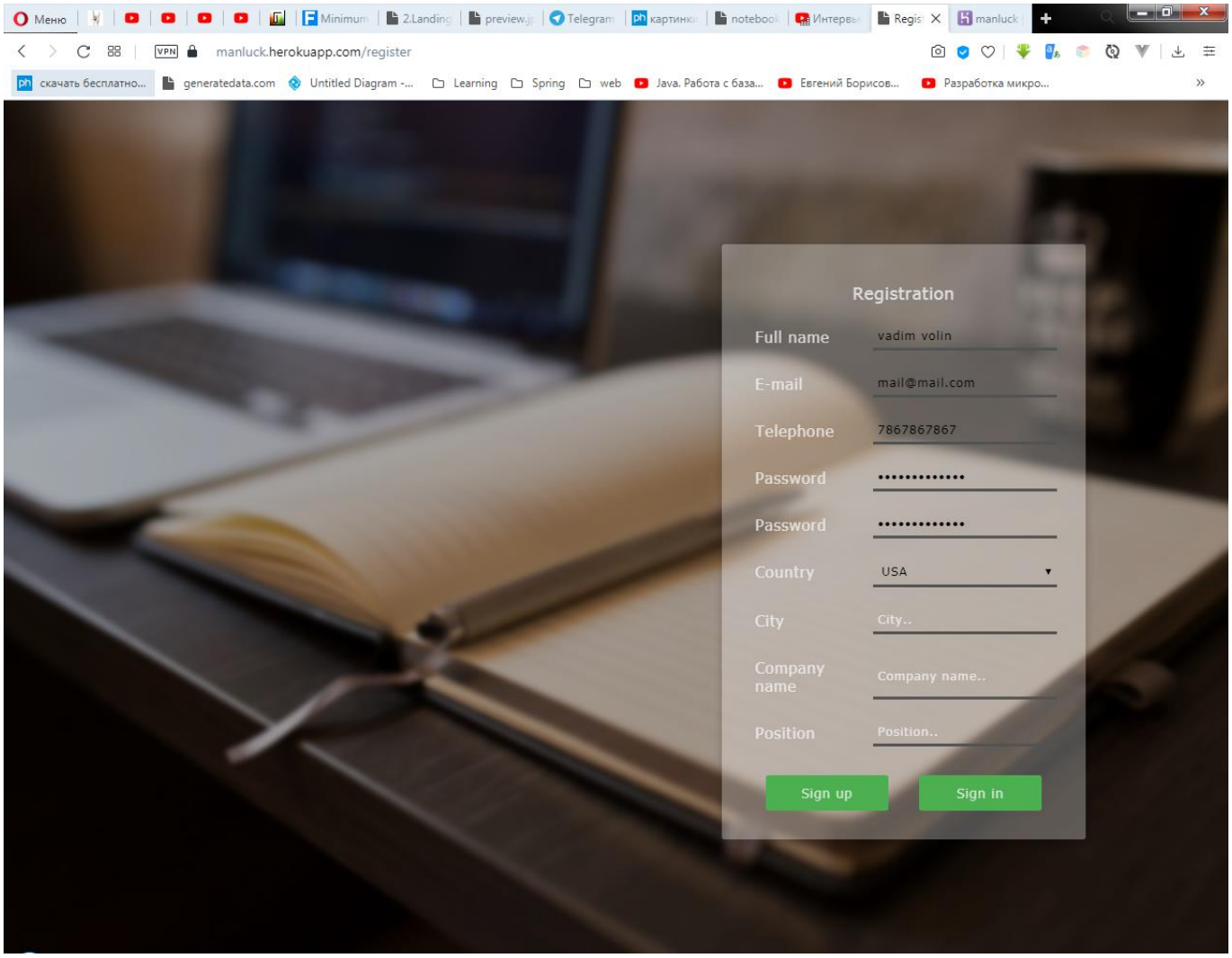

Рисунок 3.10 – Вікно реєстрації

Авторизувавшись чи зареєструвавши акаунт користувач буде перенаправлений на сторінку зі статистикою та базовою інформацією користувача, яку зображено на рисунку 3.11.

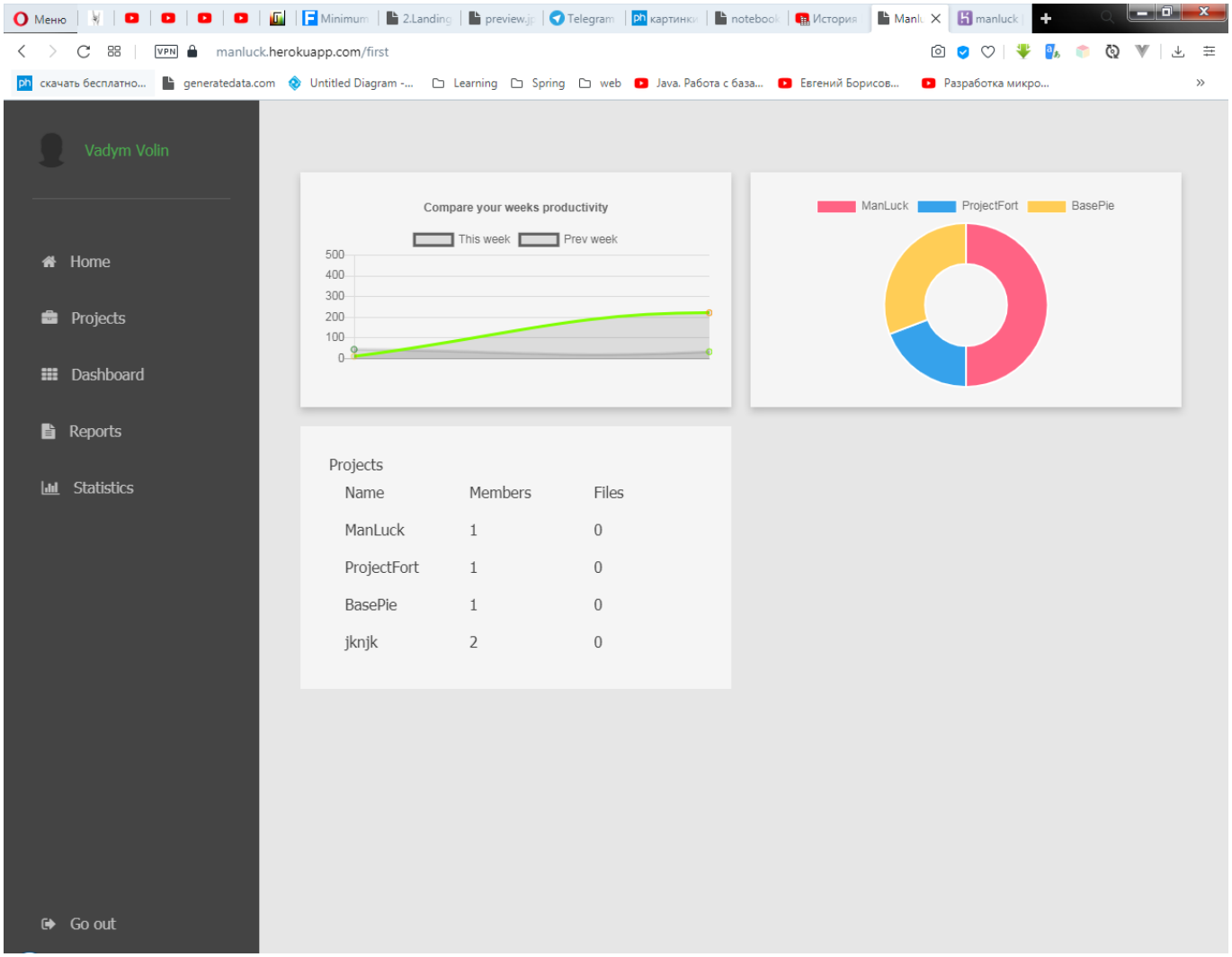

Рисунок 3.11 – Початкова сторінка

На домашній сторінці зображено форми для зміни даних користувача чи його фото. Змінивши дані у полях для введення, відразу на сервер буде відправлено запит, для швидкої зміни інформації. Сторінку зображено на рисунку 3.12.

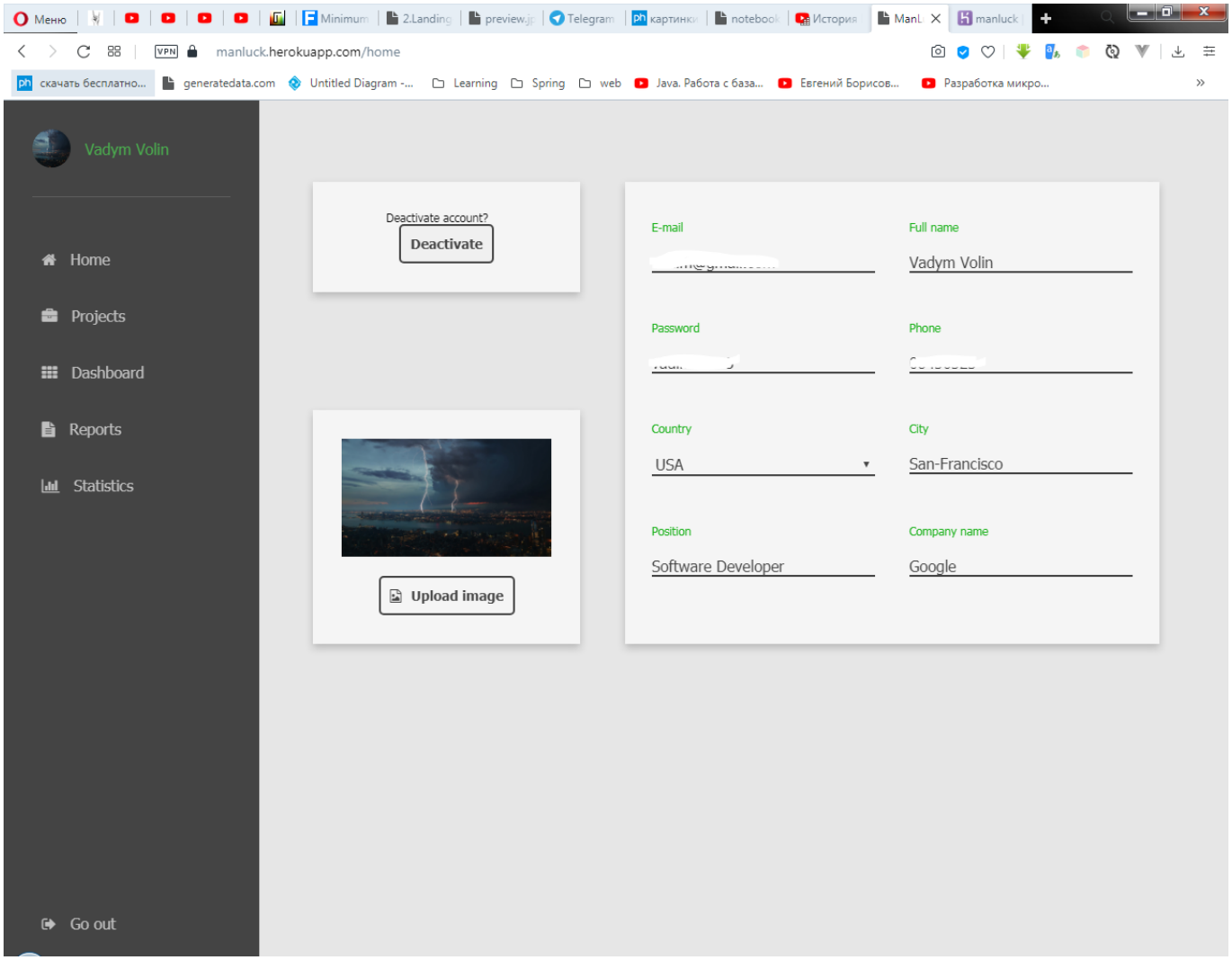

Рисунок 3.12 – Домашня сторінка

Перейшовши на панелі навігації на сторінку проектів, користувача перенаправить на сторінку з плиткою проектів. На сторінці знаходить плитка проектів та кнопка додавання проекту до плитки у горі, видалити проект з власного акаунту можна натиснувши на кнопку у правому куті проекту, якщо проект не матиме учасників команди, то він буде видалений з системи, сторінку зображено на рисунку 3.13.

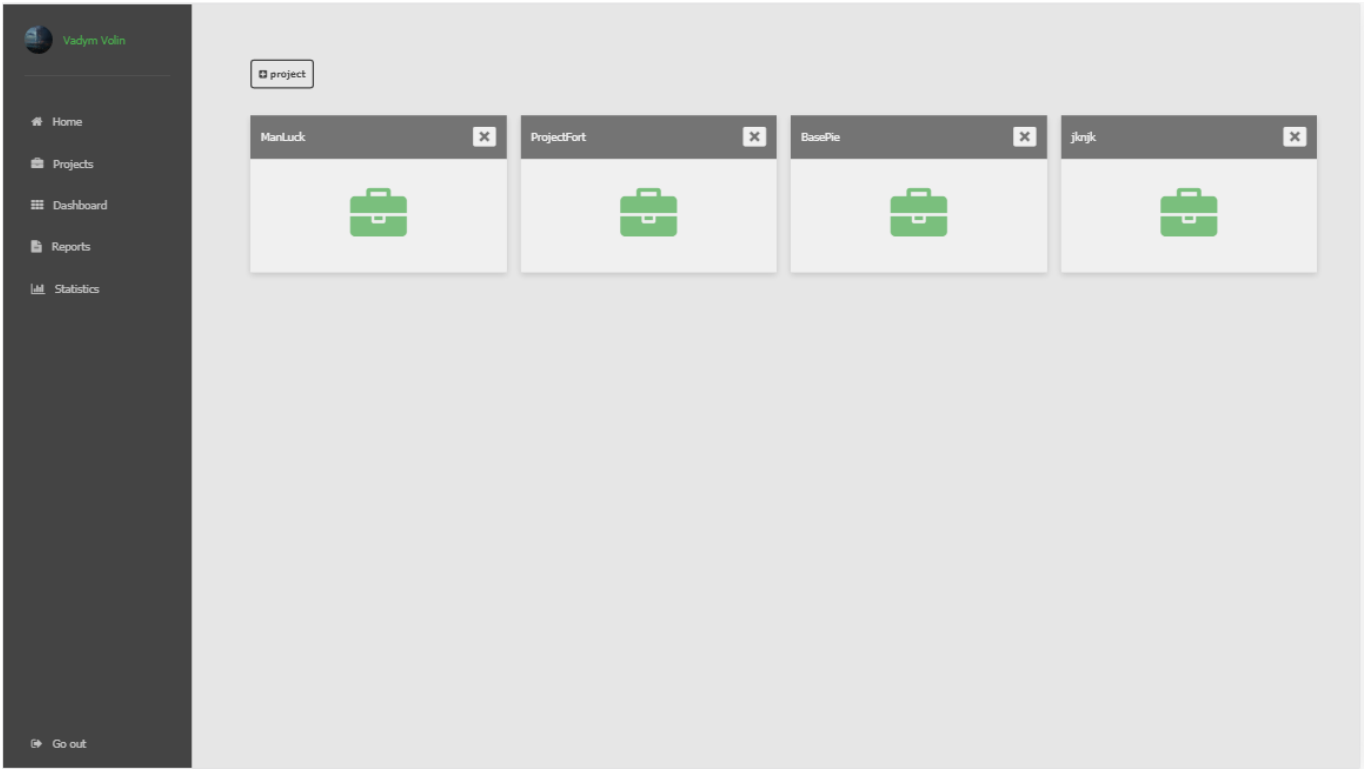

Рисунок 3.13 – Проектна сторінка

Обравши один з проектів, користувач буде перенаправлений на детальну сторінку проекту. На верхній панелі знаходяться три плитки, а саме панель задач проекту, формування звіту, перегляд статистики проекту відповідно. Додати користувача чи файл можна натиснуваши відповідну кнопку. Видалити користувача можна натиснувши кнопку хрест на відповідному користувачеві, сторінку зображено на рисунку 3.14.

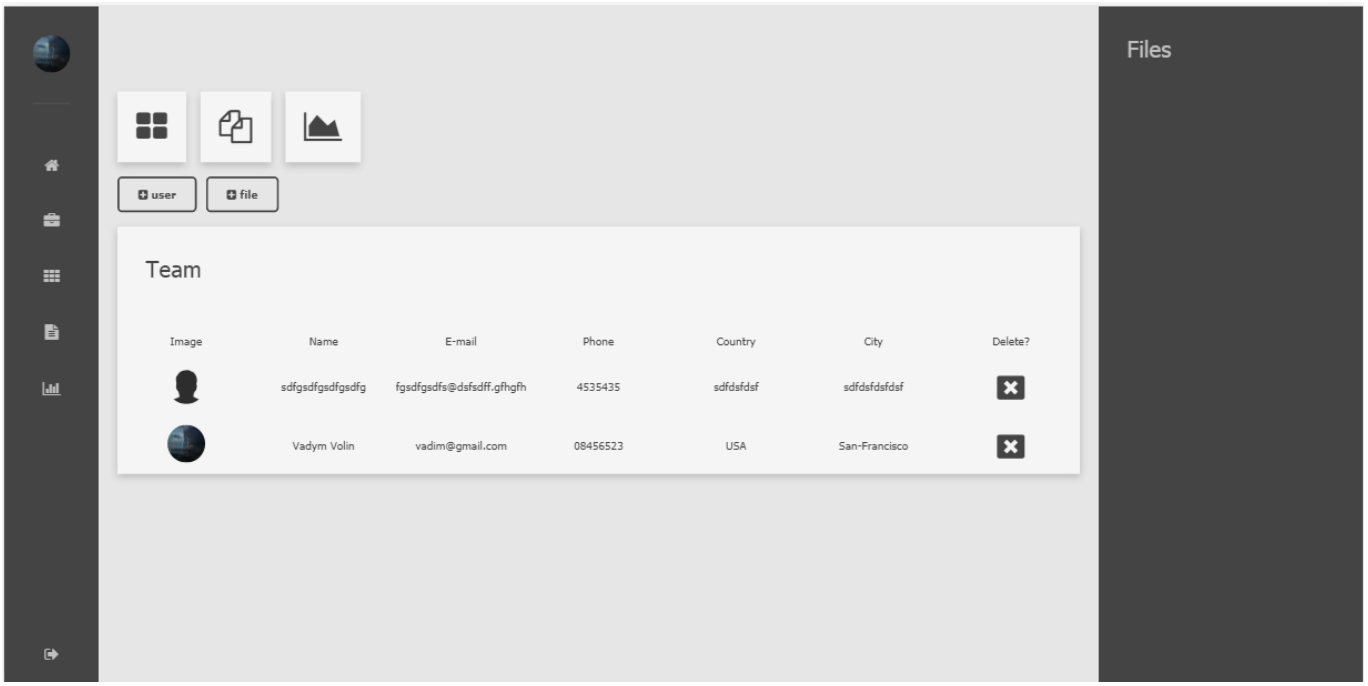

Рисунок 3.14 – Детальна сторінка проекту

Натиснувши на панель задач, користувача буде перенаправлено на сторінку з списками задач проекту, додавати списки можна натиснувши на відповідну кнопку, видалити натиснувши на кнопку хрест у правому верхньому куті списку, додати задачу можна натиснувши на кнопку додати запис, видалити можна просто перетягнувши запис у вільне місце. Запис можна перетягувати у інший список, списки також можна змінювати, сторінку задач зображено на рисунку 3.15.

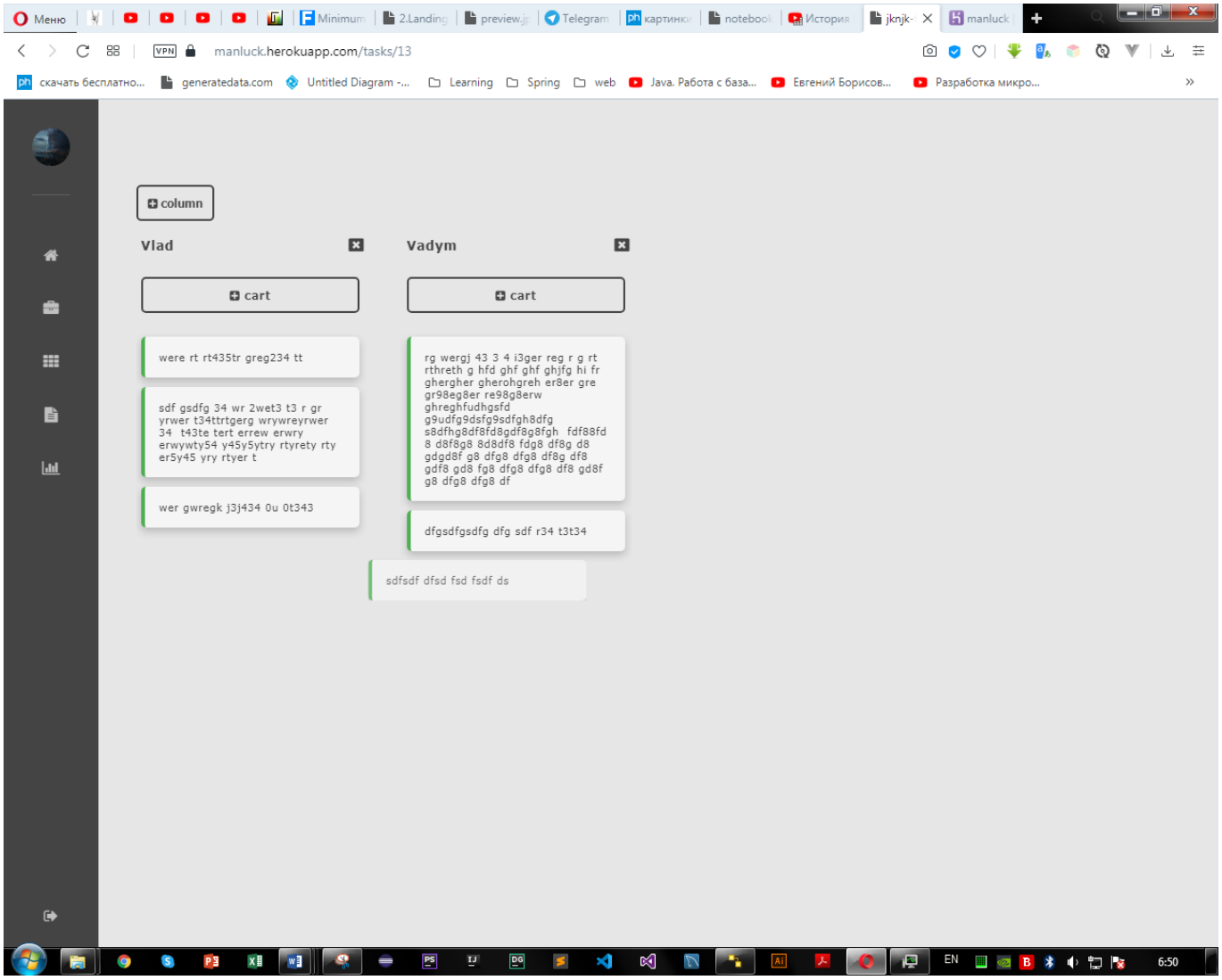

Рисунок 3.15 – Сторінка задач проекту

Обравши панель статистики, користувач матиме змогу переглянути статистику проекту, з продуктивністю команди та процесом розвитку проекту за рахунок підрахунку к-сті задач ствоюравних за час проекту, та обігу файлів, сторінку зображено на рисунку 3.16.

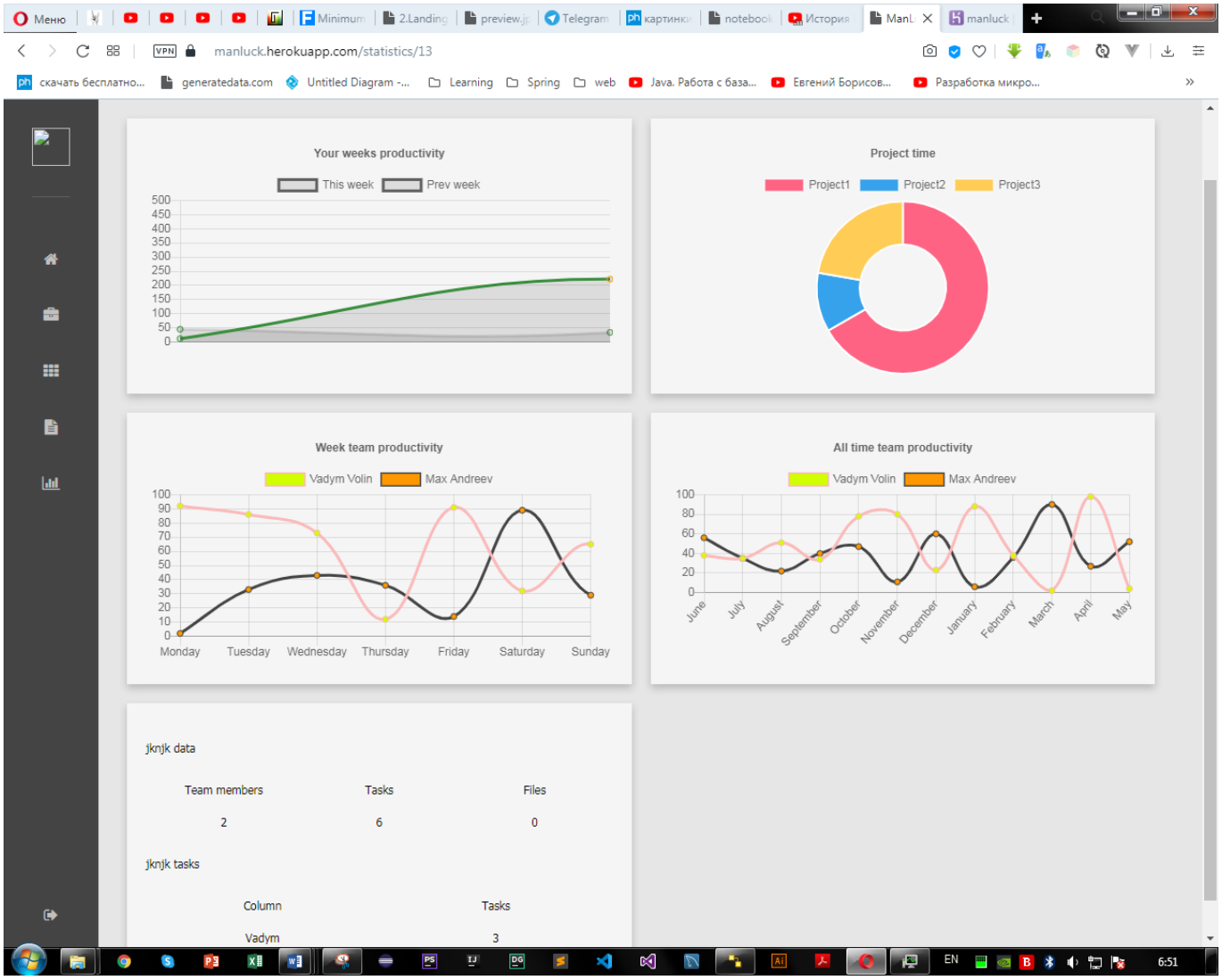

Рисунок 3.16 – Сторінка статистики проекту

Створення проекту проходить по натисненню на кнопку додати проект, з’являється форма, у якій користувач матиме змогу ввести назву проекту, натиснувши кнопку створити проект буде створений у системі, процес зображено на рисунку 3.17.

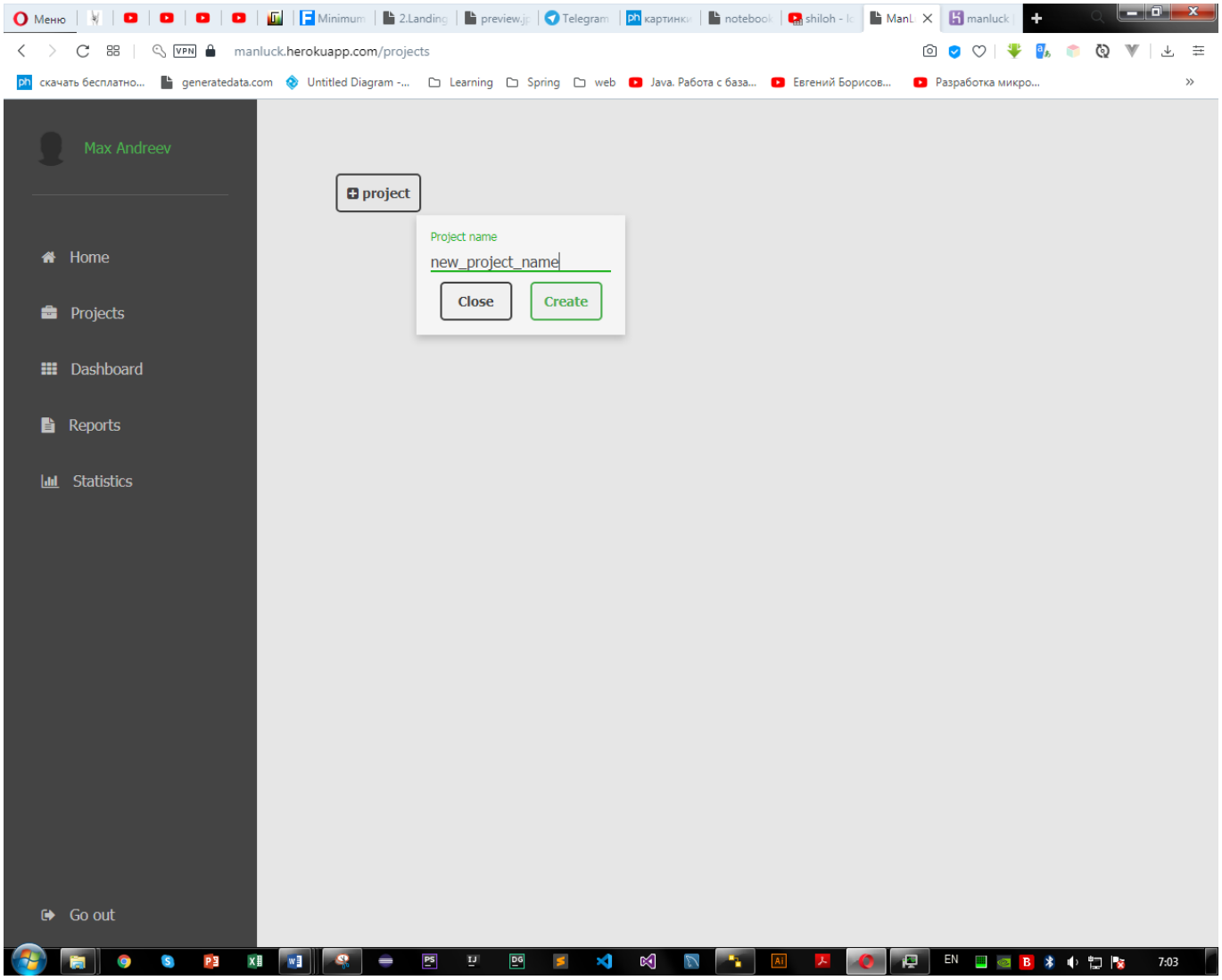

Рисунок 3.17 – Процес створення проекту

Якщо проект успішно створено, то з’являється повідомлення,приклад повідомлення зображено на рисунку 3.18 та 3.19.

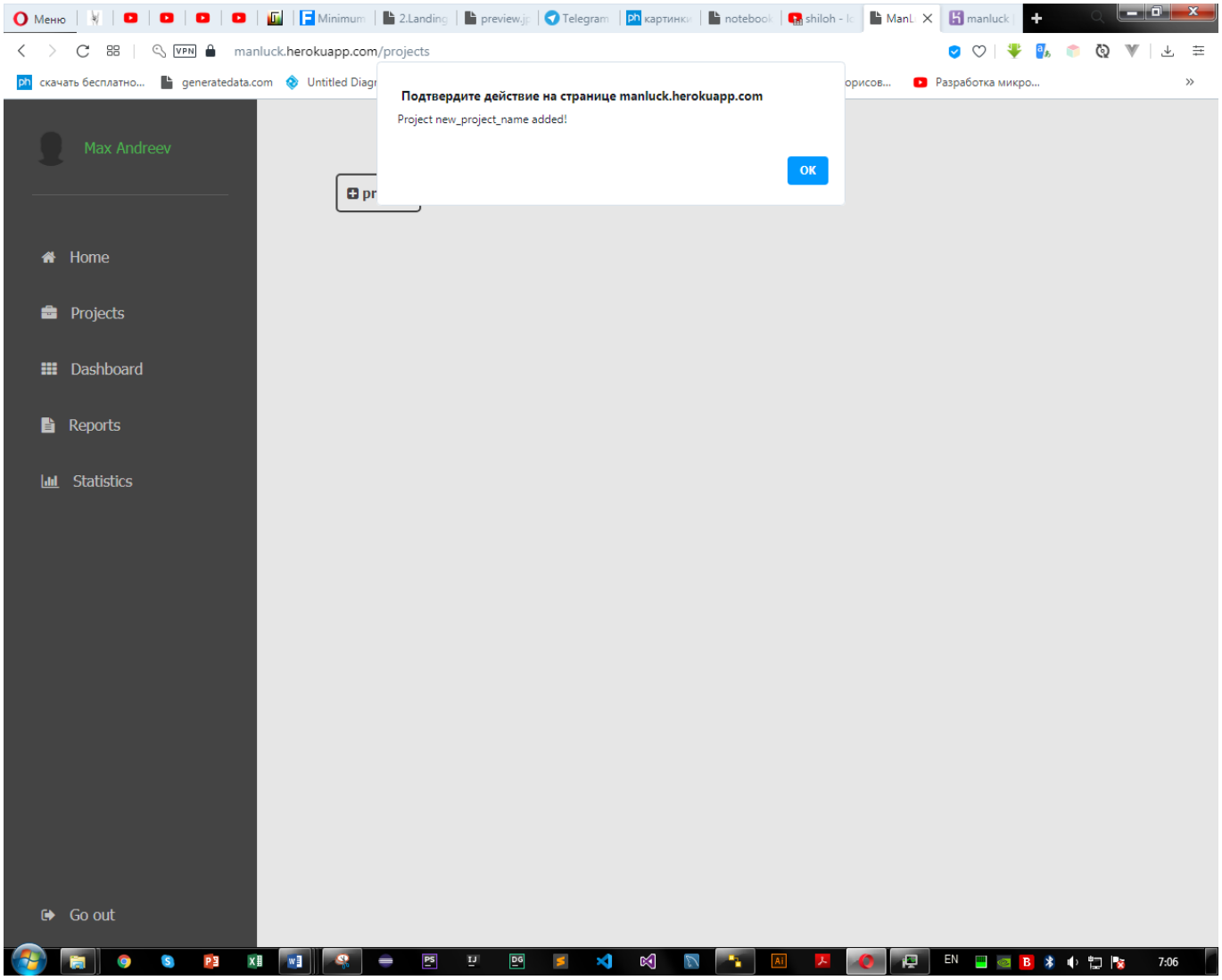

Рисунок 3.18 – Детальна сторінка проекту

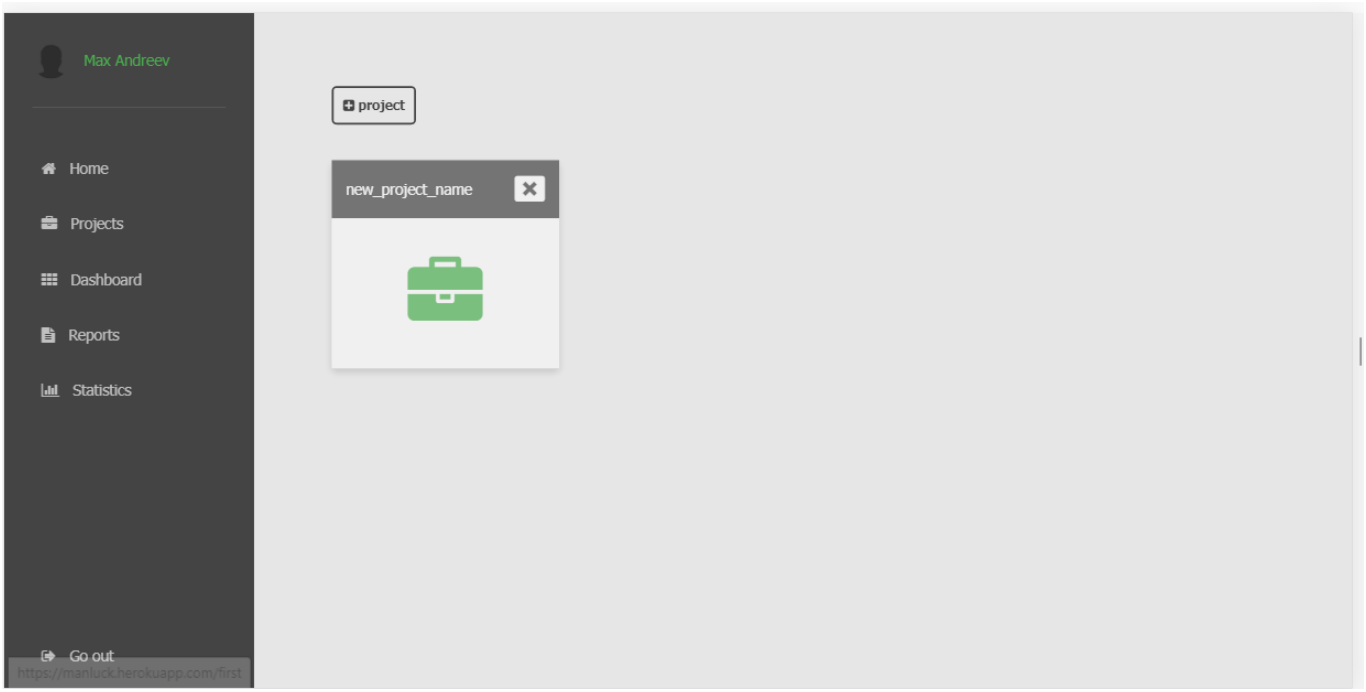

Рисунок 3.19 – Новоутворений проект

Обравши на панелі навігації сторінку звітів, користувач перенаправляється на список проектів, де має змогу обрати звіт для конкретного проекту, завантажити .pdf звіт можна натиснувши на файл у правій частині проекту, процес зображено на рисунку 3.20.

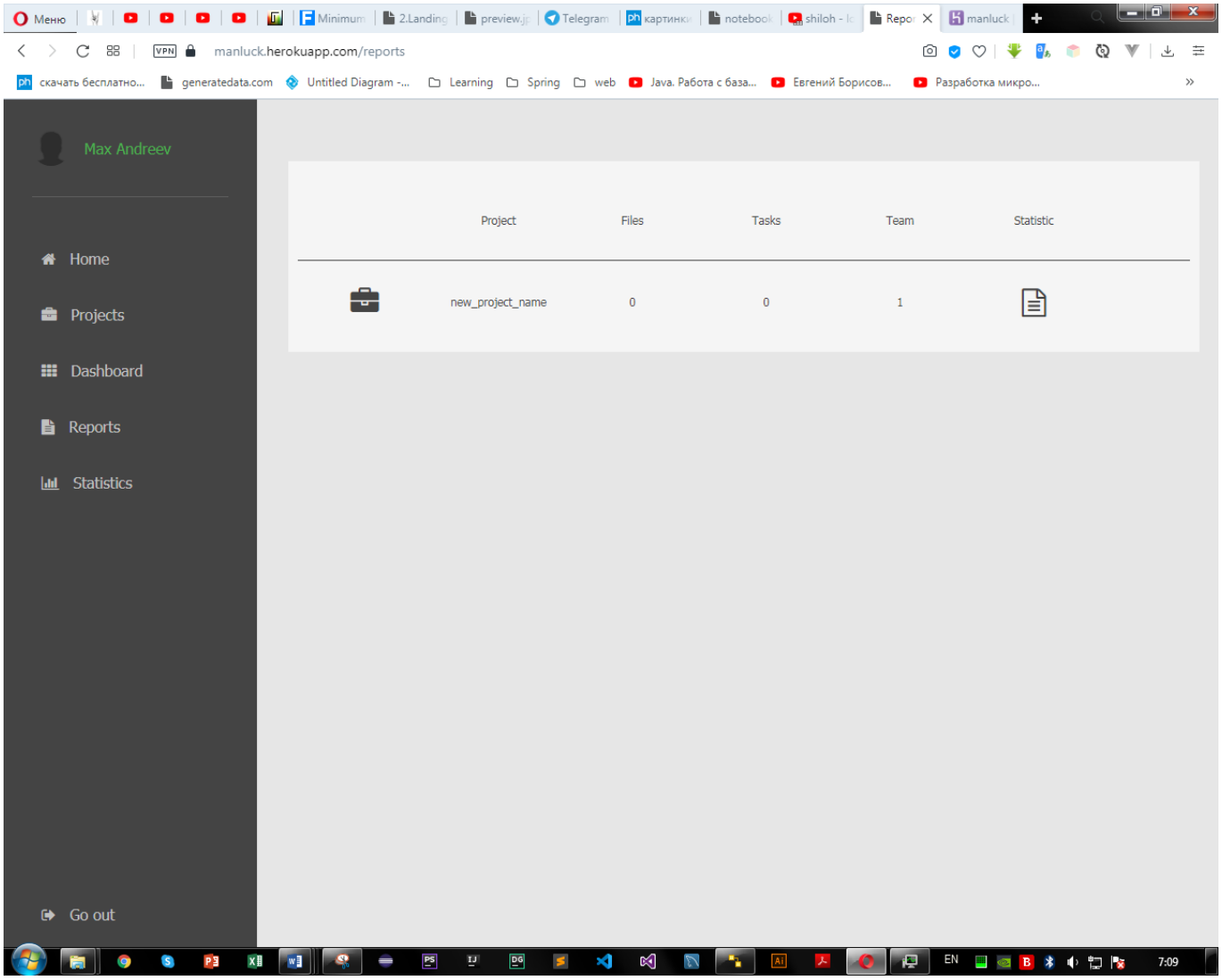

Рисунок 3.20 – Детальна сторінка проекту

ВИСНОВКИ

При виконанні дипломної роботи мною було проведено аналіз останніх досліджень в області проектного управління та проаналізовано предметну область проекту.

На наступному етапі було знайдено аналоги інформаційної системи та проведено дослідження їх функціоналу, знайдені переваги та недоліки цих сервісів перед нашою системою, в результаті цього етапу було доведено доцільність створення власної системи.

Після аналізу сервісів з аналогічними можливостями було виконано формування постановки задачі, описано базовий функціонал системи та обрано інструменти для реалізації системи.

Для налагодження процесу побудови системи управління виконанням проектів було детально описано процес проектування системи.

Для початку було проведено структурно-функціоналтне моделюваня та створено контекстну діаграму та діаграму декомпозиції з метою деталізованого огляду бізнес-логіки системи.

Для збереження інофрмації змоделбовано і розроблено власну базу даних.

Наступним етапом було створення діаграми варіантів використання, яка продемонструє послідовність дій при роботі з системою.

Після побудови діаграми варіантів використання було створено базові діаграми діяльності, які є діаграмами поведінки, що показують процес роботи об’єкта з урахуванням умов потоку виконання.

Наступним етапом була створена діаграма класів системи, яка визначає типи об’єктів у системі та різні типи зв’язків, що існують серед них. Це дозволяє переглядати систему на високому рівні.

За отриманими моделями виконано програмну реалізацію веб-додатку, який повністю відповідає технічному завданню та виконую уві висунуті вимоги.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Управління проектами та системи управління проектами. Поняття управління проектами: реферат. URL: https://ua-referat.com/Управління_проектами_та_системи_управління_проектами
2. Системи управління проектами. Кращі системи управління проектами: стаття. URL: https://www.fewskills.com/workflow-project-app/
3. Управління проектами та системи управління проектами. Системи управління проектами: реферат. URL: https://ukrbukva.net/page,3,1455-Upravlenie-proektami-i-sistemy-upravleniya-proektami.html
4. Архітектура клієнт-сервер на основі додатка відділу аспірантури та докторантури ОНАХТ. архітектура клієнт-сервер: реферат. URL: https://journals.onaft.edu.ua/index.php/atbp/article/view/974
5. Чи актуально на сьогодні моделювання в IDEF0? Projectimo. Моделювання. URL: http://projectimo.ru/biznes-processy/idef0.html
6. Основи UML — діаграми використання. Блог розробника. URL: https://pro-prof.com/archives/2594
7. Activity Diagrams. UML diagrams. URL: https://www.uml-diagrams.org/activity-diagrams.html
8. Entity Relationship Diagram – ER Diagram in DBMS. Be-ginnersBook. URL: https://beginnersbook.com/2015/04/e-r-model-in-dbms/
9. UML Class Diagram Tutorial with Examples. Guru99. URL: https://www.guru99.com/uml-class-diagram.html
10. Н.М. Абдікєєв, Н.З. Ємельянова Партика Т.Л., В.П. Романов. Проектування автоматизованих інформаційних систем (сучасні методи та технології): КОС-ІНФ., Рос. екон. акад., 2004. - 140 с.
11. Саак, А.Е. Інформаційні технології управління: Підручник для вузів, СПб.: Питер, 2005. - 320 с.

12. Адванта – онлайн система управління проектами URL: http://www.advanta-group.ru/
13. Андрєєва Т. Є. Проектний менеджмент як засіб досягнення мети підприємства. Вісник економіки транспорту і промисловості. – 2011. – № 34. – С. 364-370.
14. Батенко Л. П.,О. А. За-городніх, В. В. Ліщинська Управління проектами: навч. посіб. – К.: КНЕУ, 2003. – 231 с.
15. Безверхнюк Т. М. Проектно-орієнтований підхід як нова філософія організації управління державними програмами і проектами. Науковий вісник Академії муніципального управління : збірник наукових праць. – Вип. 3/2011. – Київ: Академія муніципального управління, 2011. – С. 17-24.
16. Безверхнюк Т. М. Публічна сфера як предметна область проектного менеджменту: збірник наукових праць Херсонського національного технічного університету. – №2(3). – 2010. URL: http://www.nbuv.gov.ua/old_jrn/Soc_Gum/Apdup/2010_2/2-3-2.pdf
17. Юраш А. Методичні вказівки до написання наукових робіт для студентів факультету журналістики. — Львів: ЛНУ, 2002.
18. Застосування інформаційних систем для управління проектами на підприємствах малого та середнього бізнесу. URL: https://otherreferats.allbest.ru/management/00625670_0.html#text
19. Бурков В. Н. Как управлять проектами / В.Н. Бурков., Д.А. Новиков. – М.: Синтег, 1997. – 188 с.
20. Бушуев С. Д.,С. Д. Бушуев, В. В. Морозов. Динамічне лідерство в управлінні проектами: Монографія. Українська асоціація управління проектами. – К., 1999. – 312 с.
21. Бушуев С. Д., С. Д. Бушуев, Н. С. Бушуева. Управління проектами: стан та перспективи: Матеріали IV міжнародної науково-технічної конференції. – Миколаїв: НУК, 2008. – С. 26-27.
22. Волін В.В., Ващенко С.М. Web-додаток підтримки управління виконанням проектів в компанії. / В.В. Волін, С.М. Ващенко С.М. // Інформатика,

математика, автоматика: матеріали та програма науково-технічної конференції, м. Суми, 20 – 24 квітня 2020 р. – Суми: Сумський державний університет, 2020. – С. 108.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку Web-додаток підтримки управління виконанням проектів в компанії

Суми 2020

1. Призначення й мета створення додатку

1.1.Призначення web - додатку

Web-додаток повинен представляти платформу управління виконанням проектів в компанії.

1.2.Мета створення web - додатку

Підтримка управління виконанням проектів в компанії, прискорення роботи менеджера, покращення управління ресурсами проекту.

1.3.Цільова аудиторія

У цільовій аудиторії web - додатку можна виділити наступні групи:

1. Проектні менеджери.
2. Розробники.
3. Персонал техпідтримки.

Менеджери та розробники мають доступ до загальнодоступної частини додатку.

Персонал техпідтримки виконує обов’язки збору поштових повідомлень про можливі помилки та обробку помилок у продакшені.

2. Вимоги до web-додатку

2.1.Вимоги до web - додатку в цілому

2.1.1. Вимоги до структури й функціонування web - додатку

Інформаційна система повинна бути реалізована у вигляді сайту, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2. Вимоги до персоналу

Для підтримки сайту від персоналу техпідтримки не повинно вимагатися спеціальних технічних навичок, знання технологій або програмних продуктів, за винятком загальних навичок роботи з персональним комп'ютером і стандартним веб-браузером (наприклад, MS Internet Explorer 7.0 або вище).

2.1.3. Вимоги до збереженні інформації

У системі керування сайтом повинен бути передбачений механізм резервного копіювання структури й умісту бази даних. Процедура резервного копіювання повинна проводитися співробітником, відповідальним за підтримку сайту, не рідше 1 разу на місяць. Резервне копіювання графічного вмісту повинне здійснюватися вручну.

2.1.4. Вимоги до розмежування доступу

Інформація, розташовувана на основних сторінках, не є загальнодоступною.

Користувач отримує доступ до базового функціоналу сайту після реєстрації.

Загальнодоступними є лише сторінки реєстрації, автентифікації та початкова сторінка.

2.2.Вимоги до функцій, виконуваних сайтом

2.2.1. Основні вимоги

2.2.1.1. Структура сайту

Сайт повинен складатися з наступних розділів:

- Реєстрація – виводиться сторінка сторінка з формою для реєстрації.
- Автентифікація – виводиться сторінка з формою для автентифікації.
- Головна – виводиться головна сторінка зі статистичними графіками і інформацією по проектам.
- Додому – виводиться сторінка для зміни особистої інформації.
- Проекти – виводиться сторінка з інформацією про проекти.

— Панель задач – виводиться сторінка зі списками задач користувача.
— Звіти – виводиться сторінка зі списком проектів та можливістю створити звіт про необхідну складову проекту.
— Статистика – виводиться сторінка с загальною статистикою користувача по витраченим ресурсам на кожен проект та роботу загалом.

2.2.1.2. Навігація

Користувацький інтерфейс сайту повинен забезпечувати наочне, інтуїтивно зрозуміле представлення структури розміщеної на ньому інформації, швидкий і логічний перехід до розділів і сторінок. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту: посилання на сторінки повинні мати заголовок, умовні позначки відповідати загальноприйнятим. Графічні елементи навігації повинні мати альтернативний підпис.

Система повинна забезпечувати навігацію по всіх доступних користувачеві ресурсам і відображати відповідну інформацію. Для навігації повинна використовуватися панель навігації. Меню повинне являти собою текстовий блок (список гіперпосилань) у лівій колонці.

2.2.1.3. Наповнення сайту

Сторінки всіх розділів сайту повинні формуватися програмним шляхом на підставі інформації з бази даних на сервері.

Вміст деяких розділів корегується користувачем, в залежності від потрібної інформації, файлів та медіа даних.

2.2.1.4. Система навігації

Взаємозв'язок між розділами й підрозділами сайту (карта сайту) представлено на рисунку А.1.

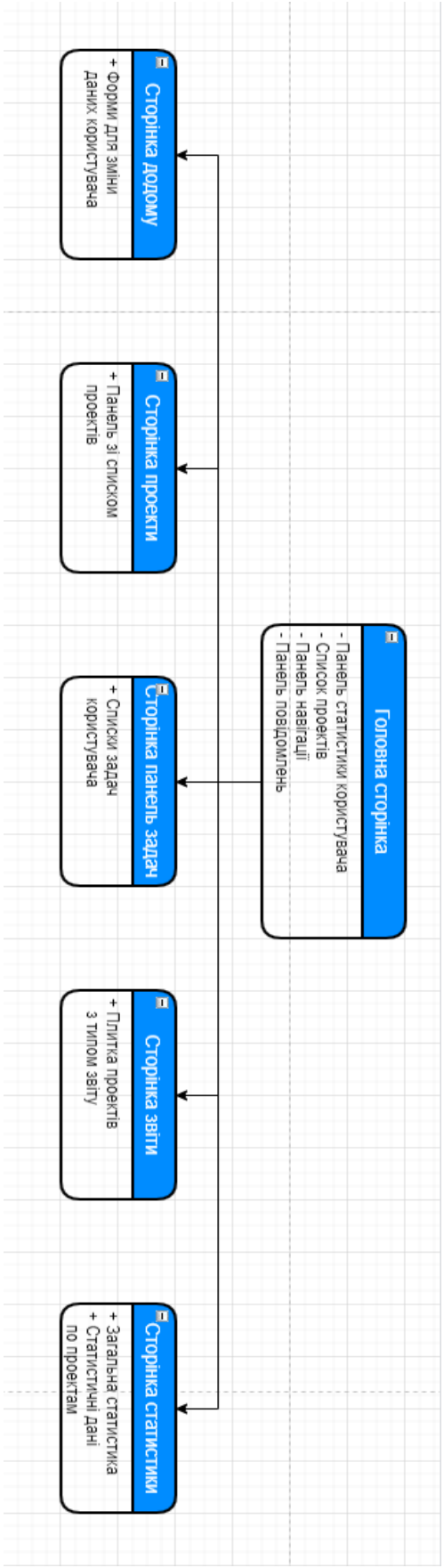

Рисунок А.1– Карта сайту

2.2.2. Вимоги до функціональних можливостей

Система керування наповненням сайту повинна надавати можливість файлів, медіа контенту користувачами, обмежувати цей функціонал для користувачів без прав доступу.

2.2.2.1. Функціональні можливості розділів

На сторінці привітання є можливість перейти до реєстрації або авторизації користувача.

На сторінці реєстрації користувач має форму для заповнення інформації необхідної для реєстрації нового акаунту користувача у системі, користувач повинен мати унікальну електронну адресу та унікальне ім’я користувача системи, щоб бути зареєстрованим у системі, інформація повинна відповідати необхідним вимогам та проходити валідацію форми, якщо дані невірні, то користувачу повідомляється про помилку.

На сторінці авторизації користувач має два поля, електрону адресу користувача системи та пароль, який необхідний для авторизації акаунту, інформація повинна відповідати необхідним вимогам та проходити валідацію форми, якщо дані невірні, то користувачу повідомляється про помилку.

На головній сторінці представлено панель навігації для переходу до необхідного розділу та головна панель зі статистикою та швидким доступом до активних проектів, обрати проект можна натиснувши на обраний зі списку.

На домашній сторінці є панель з формами для зміни особистої інформації, фото користувача та кнопка для виходу з акаунту.

На сторінці проекти, користувач має змогу переглянути всі проекти, де він є учасником та перейти до особистої сторінки проекту натиснувши на обраний проект. На особистій сторінці проекту користувач може обрати перегляд статистики, медіа файлів проекту, переглянути команду та зробити звіти натиснувши на відповідну кнопку.

На сторінці панелі задач, користувач має змогу переглядати та створювати списки завдань для роботи. Створити задачу можна натиснувши кнопку додати на списку. Створити список можна натиснувши на кнопку додати список, яка

знаходиться угорі сторінки. Задачі можна перетягувати зі одного списку до іншого, також, можна перетягувати списки, міняючи їх місцями.

На сторінці звітів користувач обирає проект натиснувши на нього, далі має змогу обрати тип звіту для завантаження обравши необхідний пункт.

На сторінці статистики користувач обирає проект з переліку натиснувши на обраний, далі він перенаправляється на сторінку зі статистикою проекту.

Угорі розташована панель повідомлень, де знаходиться кнопка, натиснувши на яку можна перейти на сторінку де переглянути повідомлення про дії на ваших проектах. На панелі знаходиться назва кожного розділу

2.2.2.2. Загальні вимоги

Стиль сайту можна описати як мінімалістично-діловий. У якості фонового кольору рекомендовано обирати легкі та стримані кольори.

Оформлення не повинне заважати інформативності, але на сайті не повинно бути досить багато графіки, він повинен бути зручний користувачам у плані навігації та довготривалої роботи.

Розташування елементів на сторінці реєстрації схематично показано на рис. А.2.

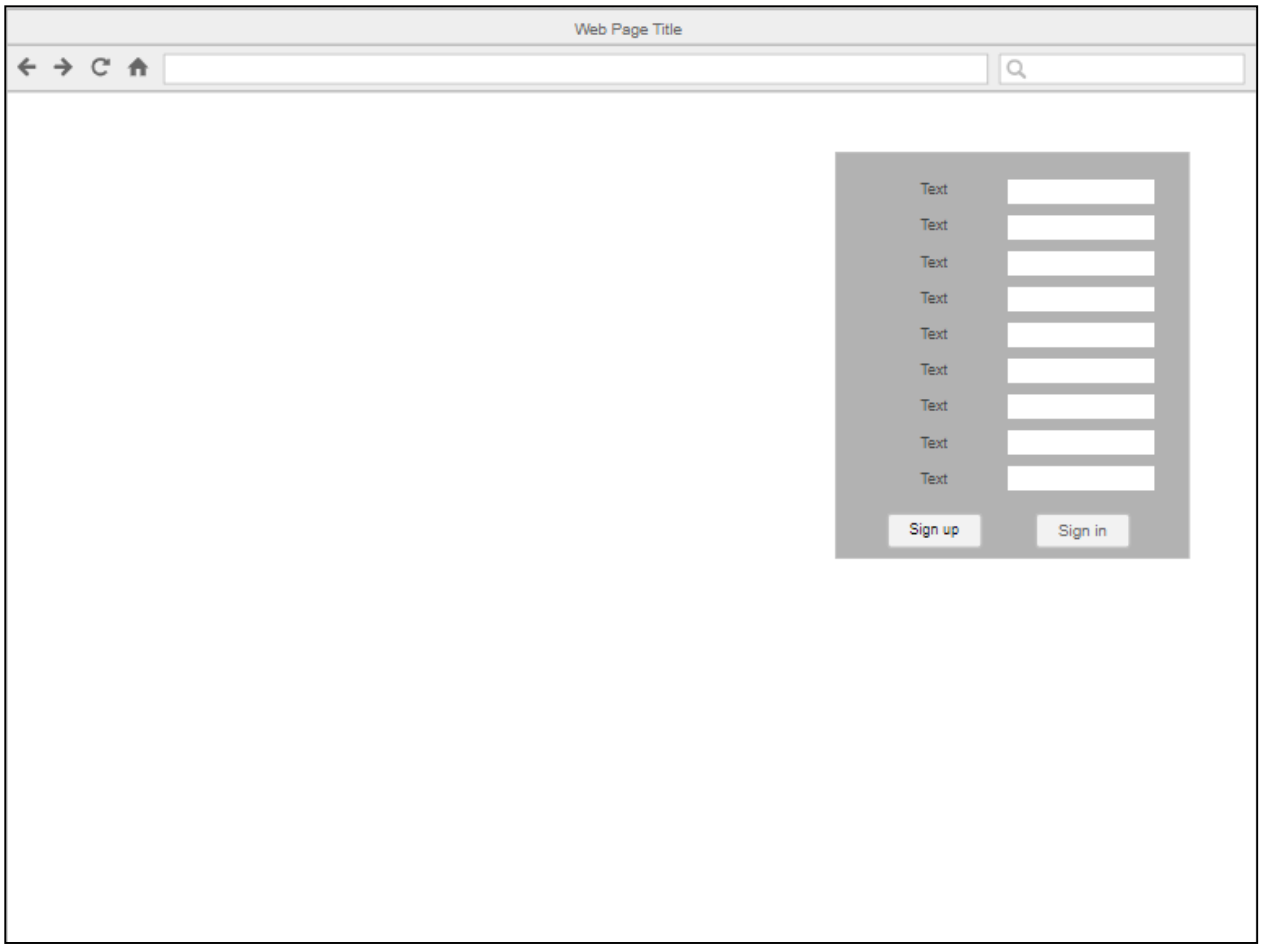

Рисунок А.2 – *Типова сторінка реєстрації*

Розташування елементів на сторінці авторизації схематично показано на рис. А.3.

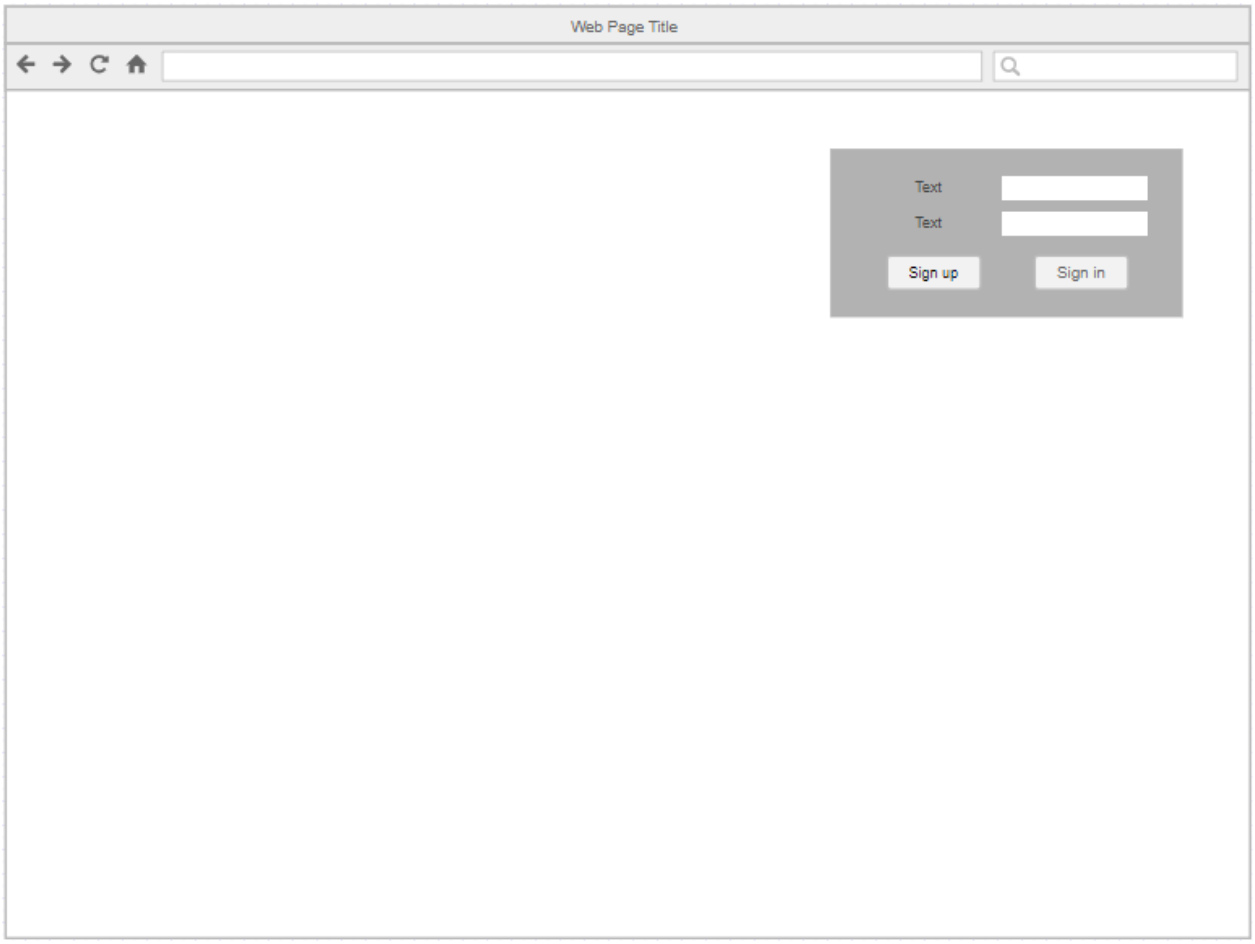

Рисунок А.3 – *Типова сторінка авторизації*

Розташування елементів на головній сторінці схематично показано на рис. А.4.

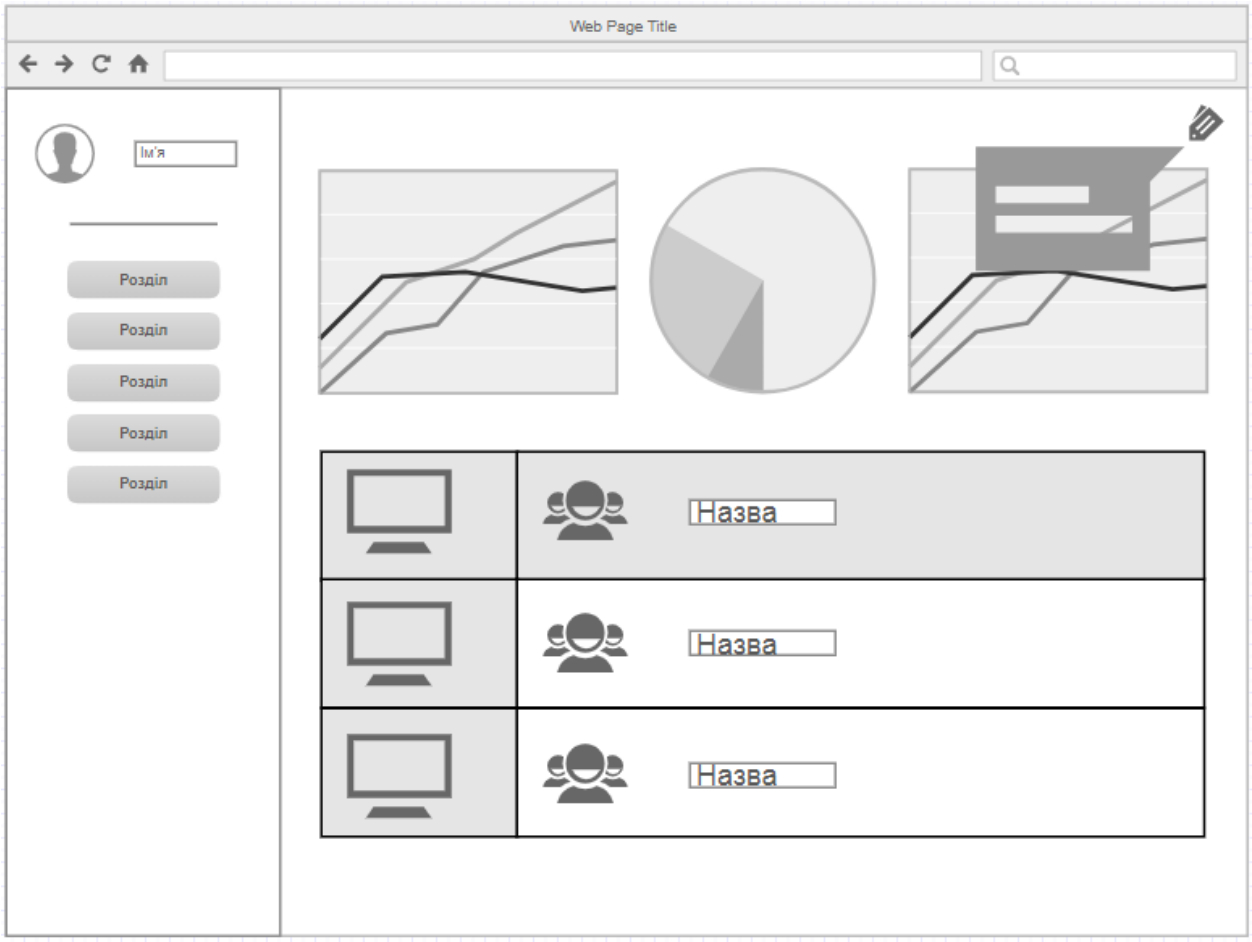

Рисунок А.4 – *Типова сторінка головна*

Розташування елементів на сторінці додому схематично показано на рис. А.5.

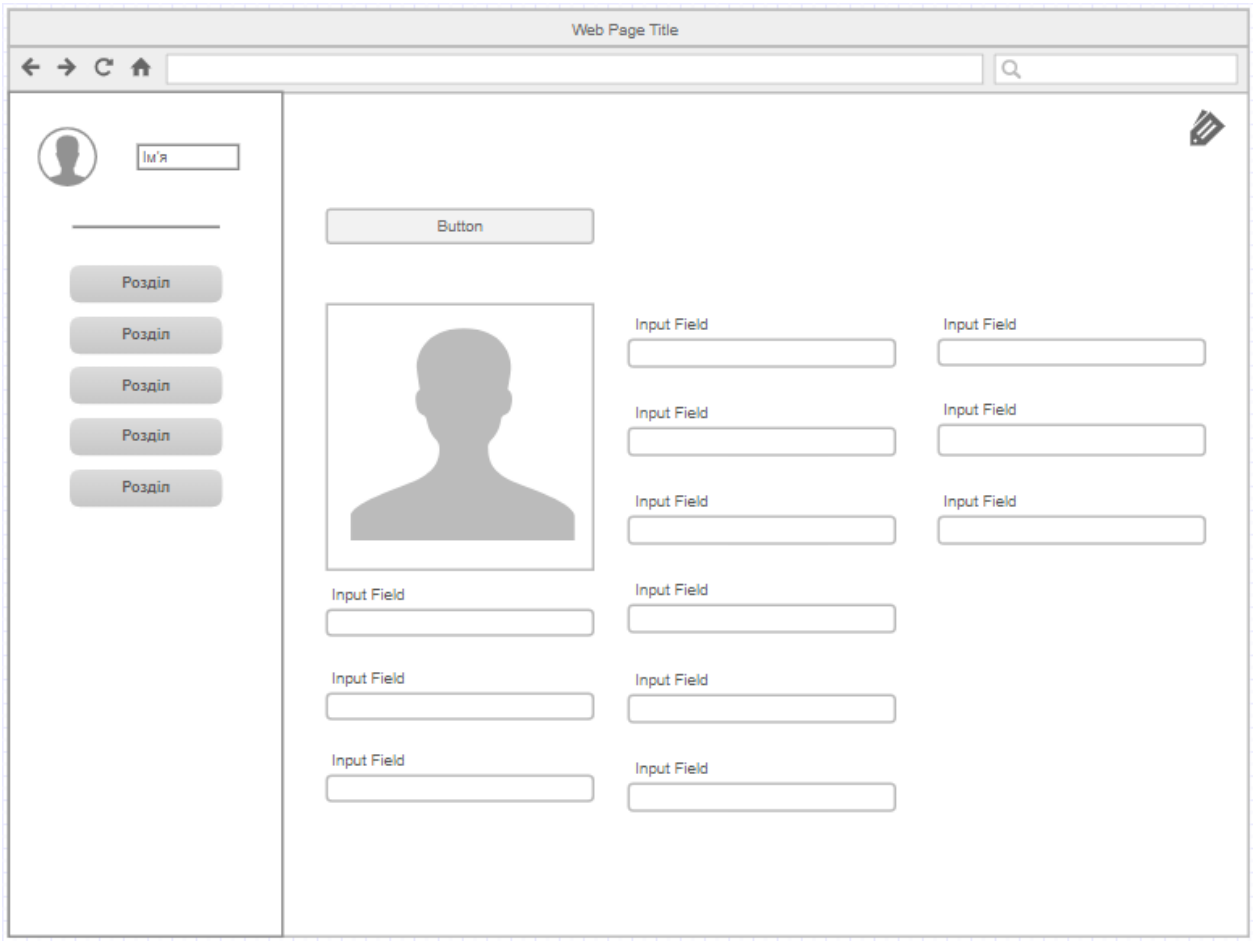

Рисунок А.5 – *Типова сторінка додому*

Розташування елементів на сторінці проекти схематично показано на рис. А.6.

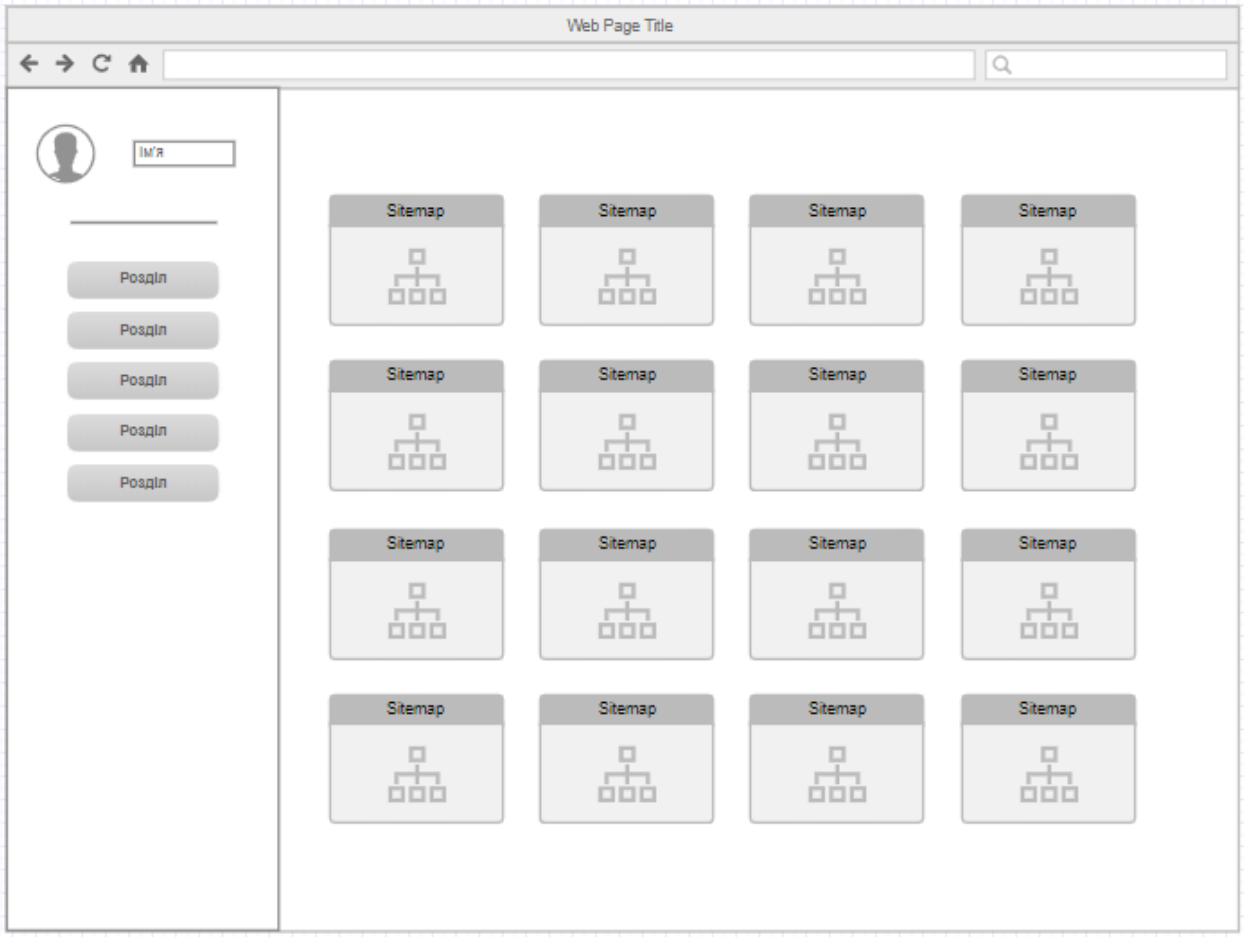

Рисунок А.6 – *Типова сторінка проекти*

Розташування елементів на сторінці панелі задач схематично показано на рис. А.7.

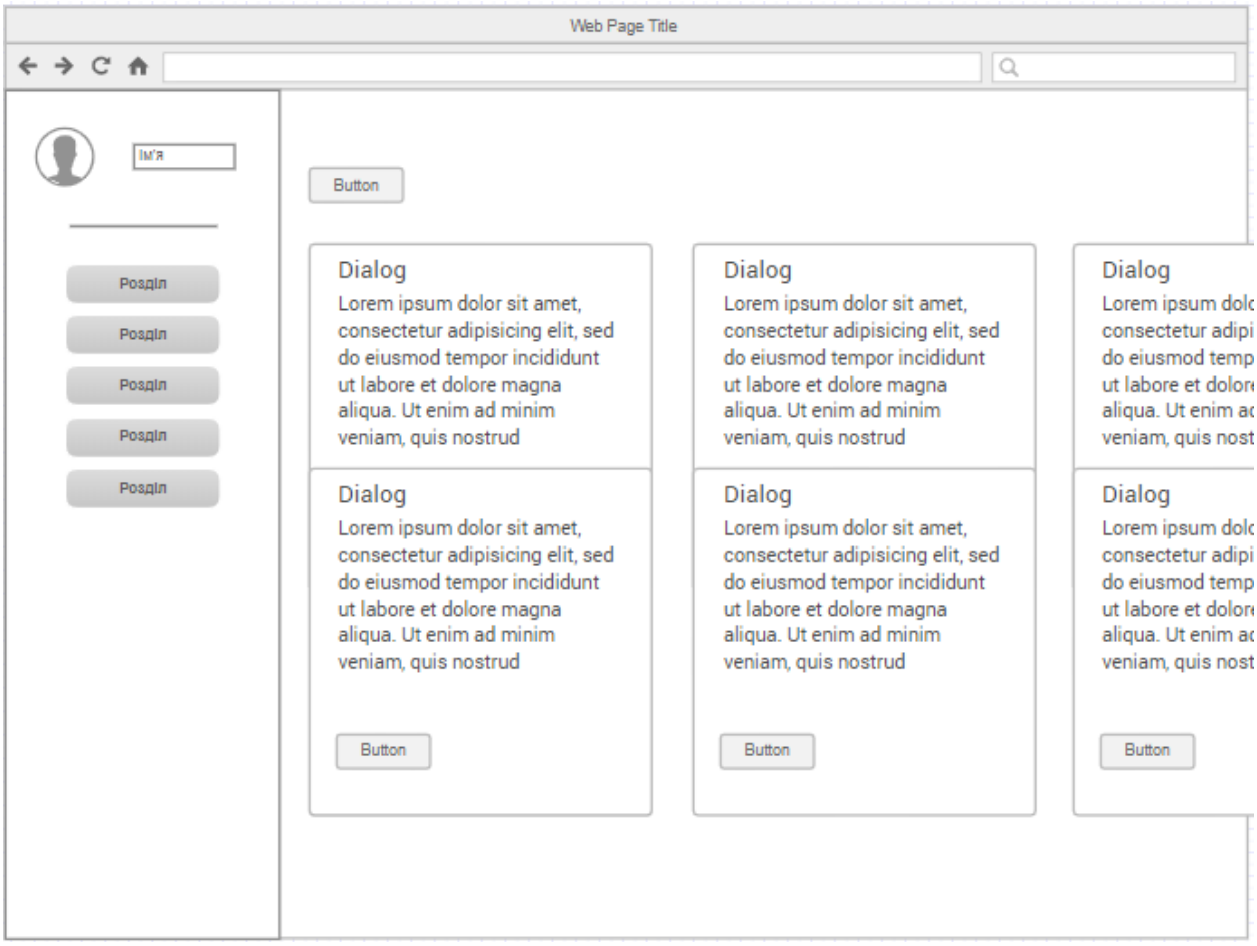

Рисунок А.7 – *Типова сторінка панелі задач*

Розташування елементів на сторінці звітів схематично показано на рис. А.8.

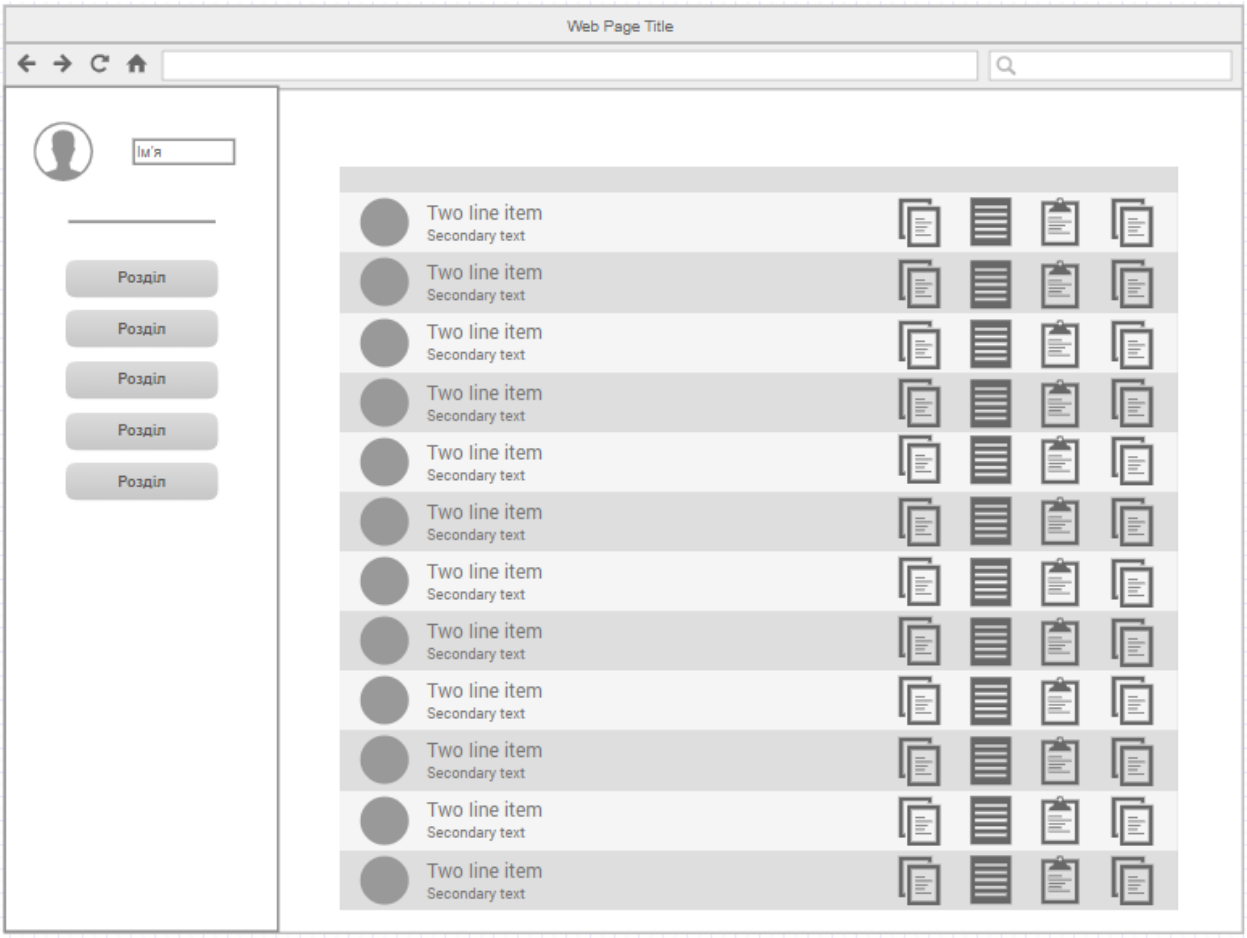

Рисунок А.8 – *Типова сторінка звітів*

Розташування елементів на сторінці статистики схематично показано на рис. А.9.

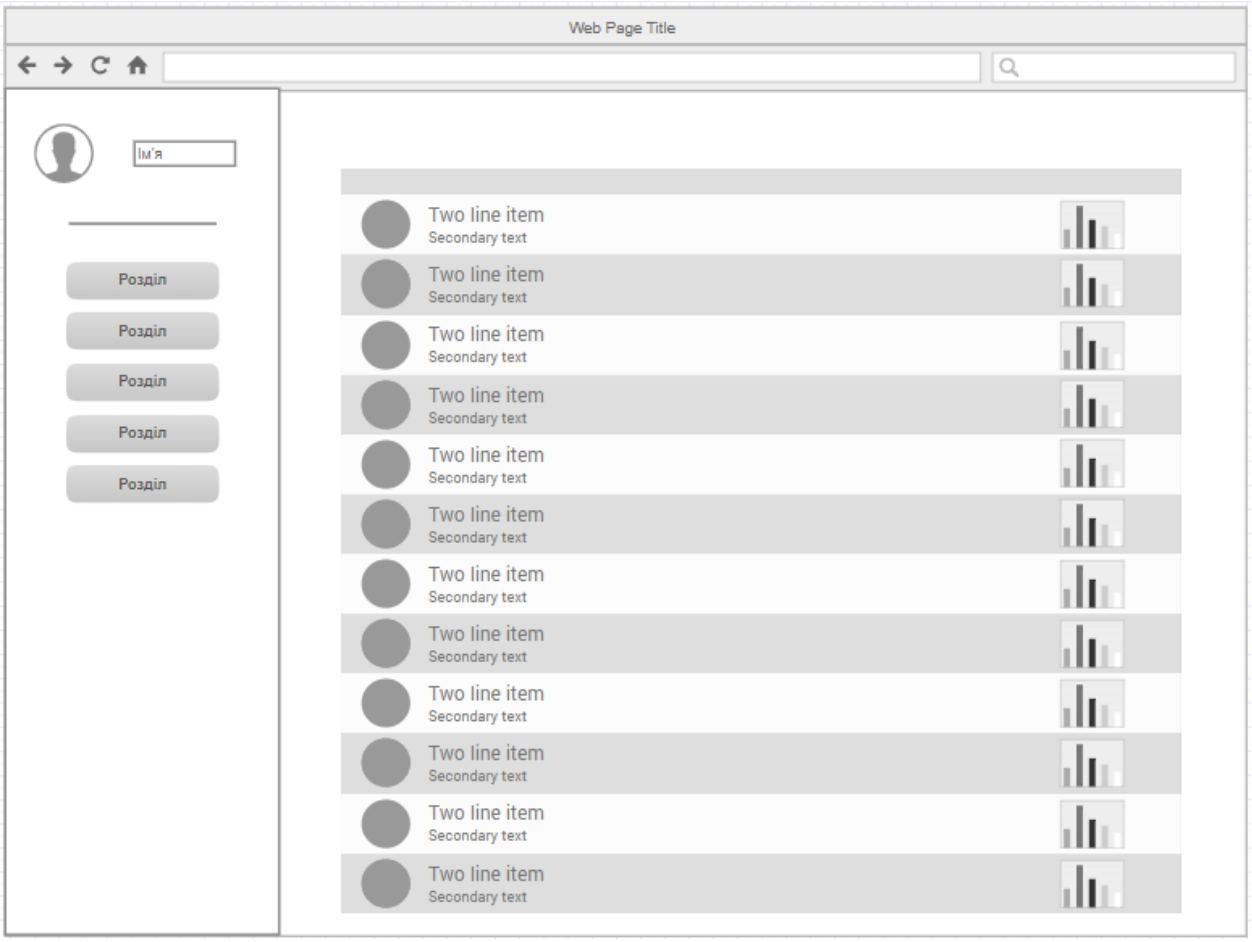

Рисунок А.9 – *Типова сторінка статистики*

2.2.2.3. Типові навігаційні й інформаційні елементи

— Шапка сайту

— Бокове меню

— Основне поле контенту

2.2.2.4. Шапка сайту

Шапка сайту повинна містити фото користувача і його ім’я. Блок є посиланняи на головну сторінку сайту.

2.2.2.5. Бокове меню

Бокове меню повинне розташовуватися в лівій частині вікна (залежно від затвердженого дизайн-макета) і містити посилання на всі розділи першого рівня.

2.2.2.6. Основне поле контенту

Основне поле контенту повинне розташовуватися в центрі сторінки. У цьому полі відображається основний зміст обраного розділу. Стильове оформлення матеріалів і їх елементів (посилань, заголовків, основного тексту, зображень, форм, таблиць і т.п.) повинне бути єдиним для всього веб-сайту.

2.3. Вимоги до видів забезпечення

2.3.1. Вимоги до інформаційного забезпечення

Реалізація сайту відбувається з використанням:

- Apache Tomcat 9.0.32
- Spring Framework 5.2.2
- Java 12
- PostgreSQL 11.0
- Linux Ubuntu 18.04

2.3.2. Вимоги до лінгвістичного забезпечення

Сайт повинен бути виконаний англійською мовою.

2.3.3. Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Веб-браузер: Internet Explorer 11.0 і вище, або Firefox 50.0 і вище, або Opera 9.5 і вище, або Safari 9 і вище, або Chrome 60.0.3112 і вище;
- Включена підтримка javascript, cookies.

2.3.4. Вимоги до апаратного забезпечення

Апаратне забезпечення серверної частини повинне задовольняти наступним вимогам:

- Веб-сервер Apache Tomcat з модулем MOD_REWRITE;
- База даних PostgreSQL вище 11 версії;
- Не менш 500 МБ вільного місця на диску.

Апаратне забезпечення клієнтської частини повинне забезпечувати підтримку програмного забезпечення клієнтської частини, зазначеного в п. 2.3.1.

3. Склад і зміст робіт зі створення сайту

Докладний опис етапів роботи зі створення сайту наведено в табл. А.1.

Таблиця А.1 – Етапи створення сайту

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка каркасу: Проектування розмітки та наповнення сайту	10 днів
2	Авторизація: Розроблення та реалізація блоку Авторизації на веб-сайті	4 дні
3	Зміна інформації: Створення модулю зміни особистих даних	2 дні
4	Модуль проектів: Розробка модулю управління проектами	8 днів
5	Модуль панелі задач: Розробка модулю панелі задач	12 днів
6	Модуль звітів: Розробка модулю управління та створення звітів	4 днів
7	Модуль статистики: Розробка модулю статистки	4 дні
	Загальна тривалість робіт (з урахуванням резервного строку на налагодження й виправлення помилок) і строк закінчення проекту	44 днів

4. Вимоги до складу й змісту робіт із введення сайту в експлуатацію

Для створення умов функціонування, при яких гарантується відповідність створюваного сайту вимогам сьогодення ТЗ і можливість його ефективної роботи, в організації Замовника повинен бути проведений певний комплекс заходів.

Для переносу сайту на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ. На хостинг переноситься програма (сайт), представлення клієнтської частини, серверна частина системи та база даних з контентом.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Мета проекту: створити програмний додаток, який дозволить автоматизувати та покращити процес та методи управління проектами. Мета є досяжною, підґрунтям створення додатку є навички отримані на курсі з дисципліни Java, ООР, веб-розробка. Проект буде виконано вчасно, що підтверджується календарним планом проекту. Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити програмний додаток, який дозволить автоматизувати та покращити процес та методи управління проектами.
Measurable (вимірювана)	Результатом роботи проекту є робоча система з виконаними поставленими вимогами.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки Intellij IDEA 2020.1, мови програмування Java, бази даних PostgreSQL, фреймворку Spring, технології Ajax.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були описані в календарному плані. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − представлення проекту, виконане у вигляді ієрархічної структури робіт, що досягається за допомогою послідовної декомпозиції. Інструмент спрямований на детальне планування, оцінку вартості, визначення та розподіл персональної відповідальності виконавців та інші - тобто, на основні роботи і результати, що визначають зміст проекту.

Як правило, на верхньому рівні вказується сам проект, під ним (на першому рівні) - основні результати, кожен з яких, в свою чергу, деталізується, тобто наступний рівень завжди менше попереднього за обсягом робіт і, як правило, включає 2 і більше пакетів робіт. При цьому в різних гілках WBS може бути різна кількість рівнів в залежності від потрібного ступеня деталізації. Діаграма WBS зображена на рис. Б.1.

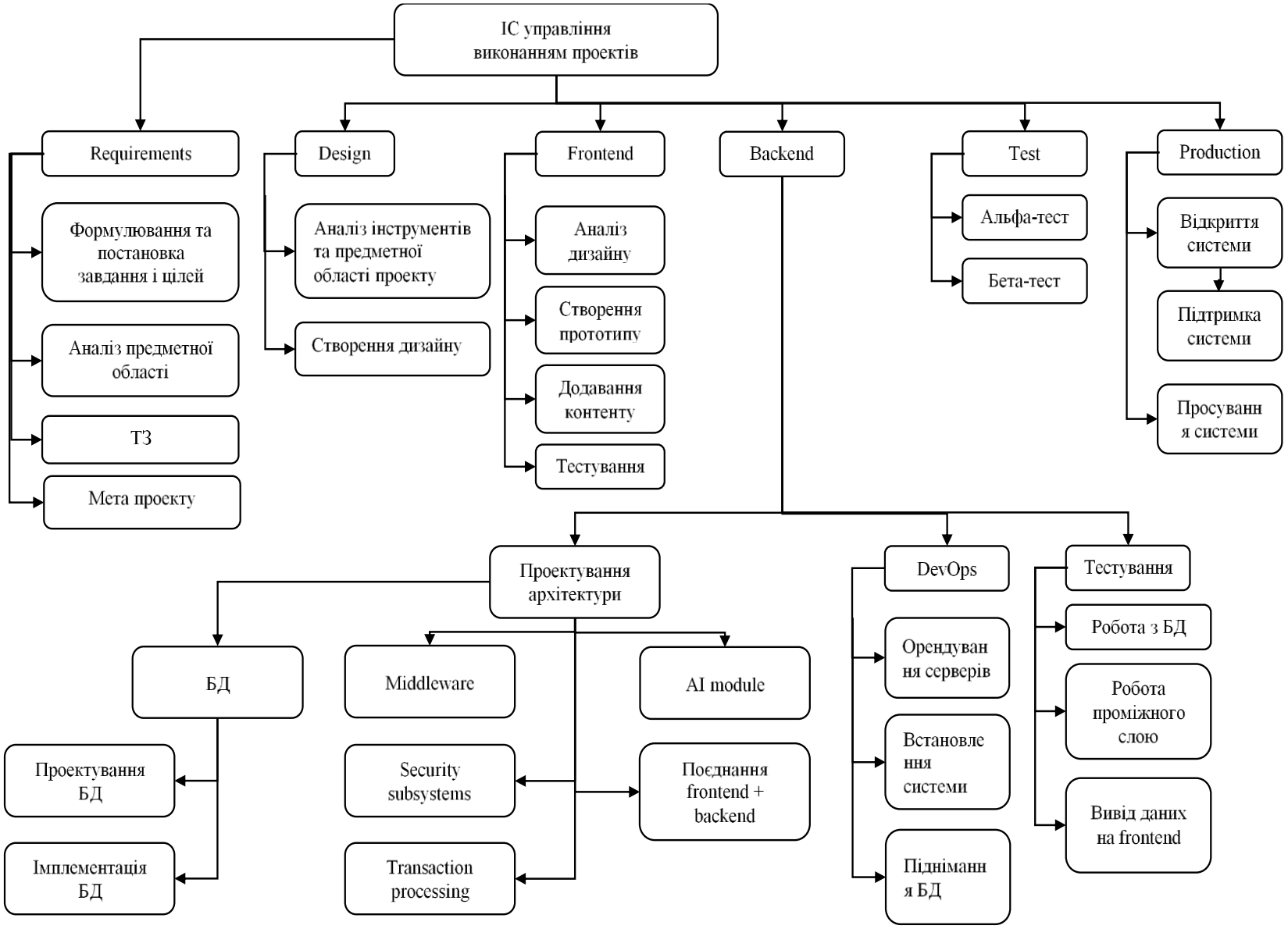

Рисунок Б.1 – WBS-структура проекту

Після того, як була побудована WBS структура проекту наступним етапом є розроблення OBS - склад, підпорядкованість, взаємодія і розподіл робіт по підрозділах і органам управління, між якими встановлюються певні відносини з приводу реалізації владних повноважень, потоків команд і інформації. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Список виконавців, що функціонують в проекті представлений в таблиці Б.2 та на рисунку Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Виконавець	Волін В.В.	Виконує розробку функціоналу системи та її інтерфейс.
Керівник проекту	Ващенко С.М.	Формує завдання на виконання проекту, корегує процес створення проекту та перевіряє виконання.

Діаграма OBS представлена на рисунку Б.2.

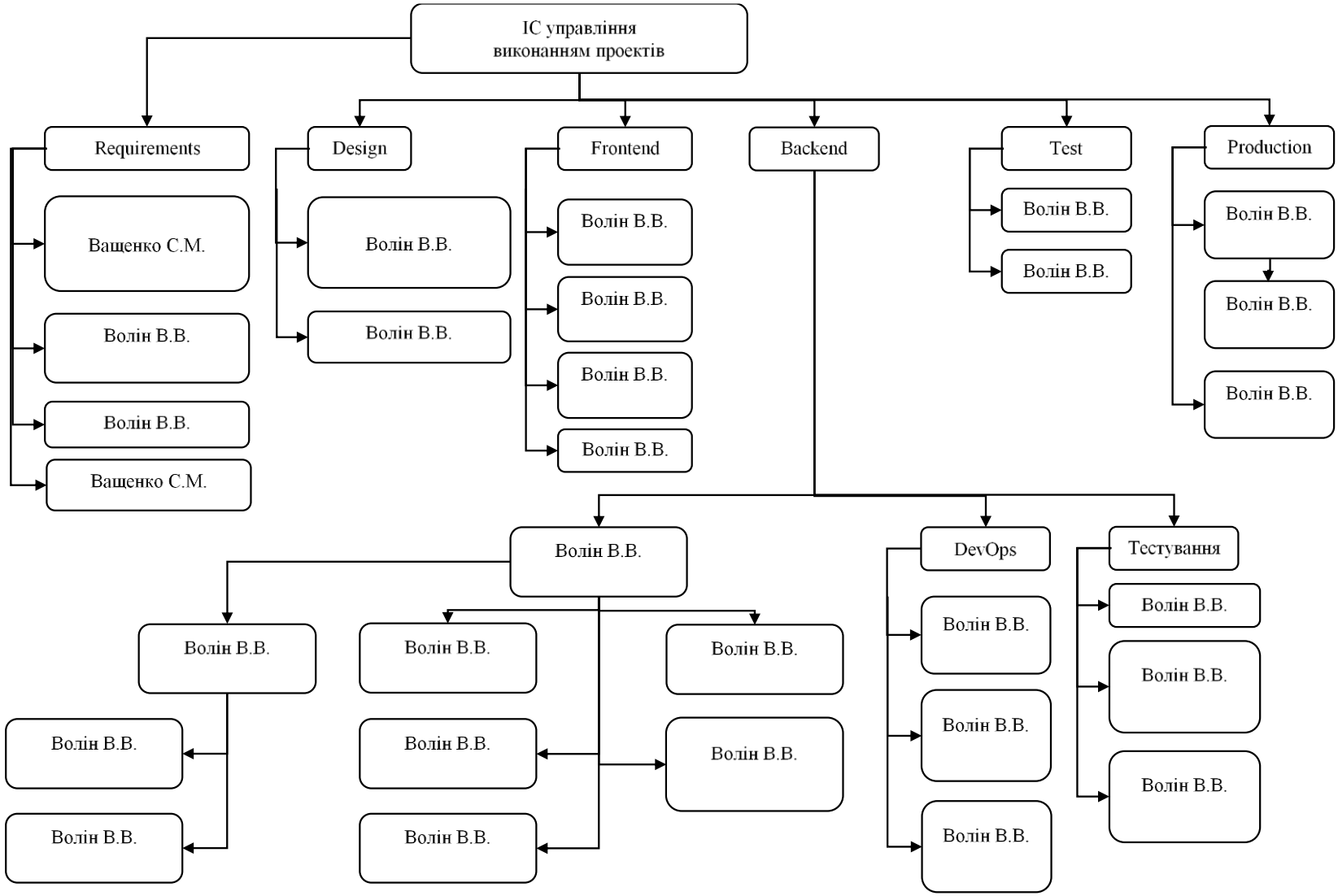

Рисунок Б.2 – OBS структура проекту

Діаграма Ганта. Діаграма Ганта вважається якісним інструментом для відображення цілей та задач. Основна відмінність цих інструментів - в способі демонстрації і відображення інформації.

На відміну від PDM – мережі, що пропонує мережеву модель, управління проектами з діаграмами Ганта засноване на форматі гістограм. Це допомагає відслідковувати відсоток робіт, виконаних по кожному завданню. Керівникам проектів дуже важливо правильно розподілити завдання і бути впевненими в тому, що проект буде завершений вчасно.

Основна увага діаграм Ганта зосереджено на процентному завершенні кожного завдання. Крім того, діаграми Ганта краще для проектів з невеликою кількістю взаємопов'язаних завдань.

Завдяки засобам програмного продукту Smartsheet була розроблена діаграма Ганта, яка у вигляді гістограми відображає тривалість кожного процесу, що був визначений на етапі формування WBS. Діаграма Ганта представлена на рисунку Б.3.

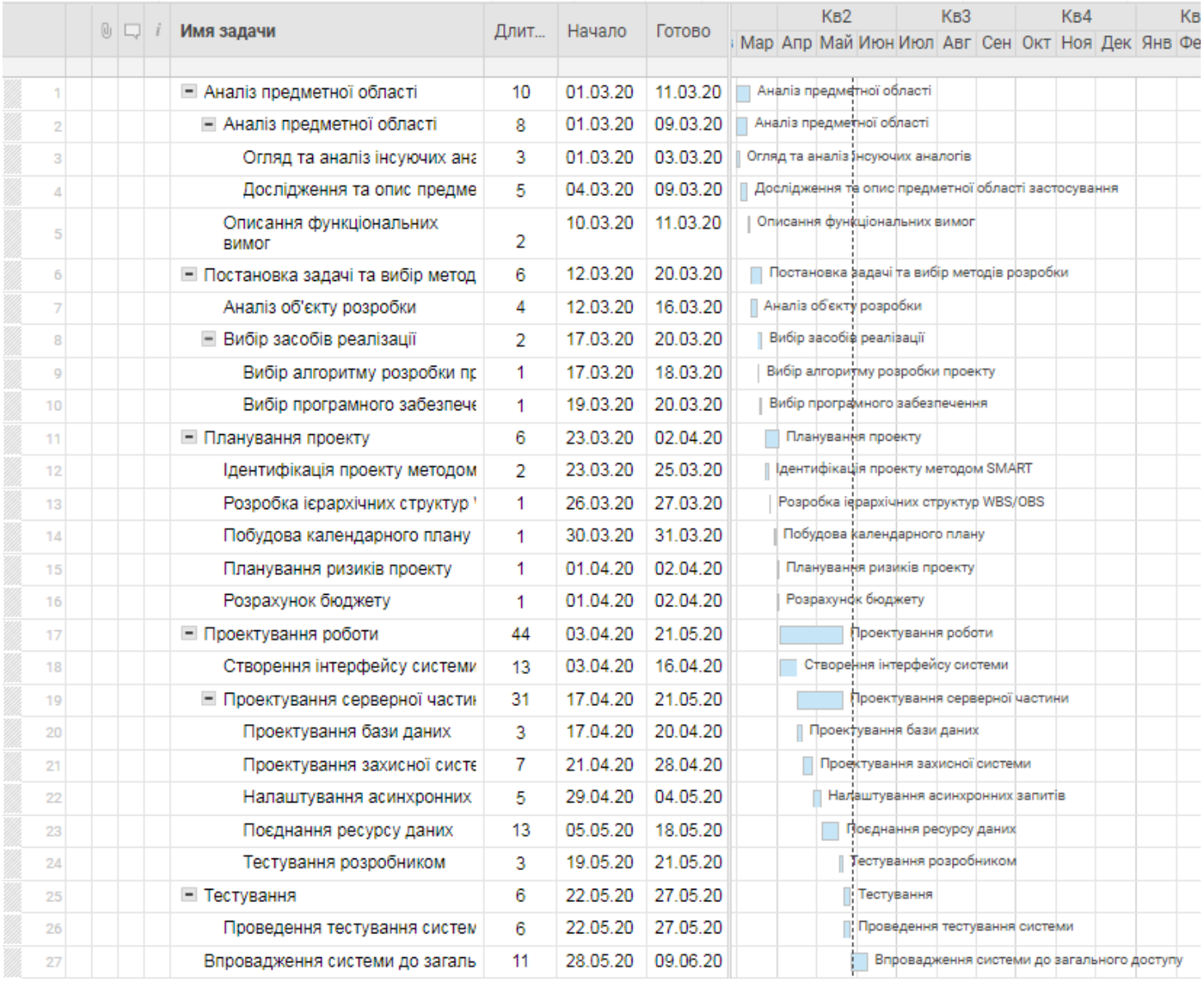

Рисунок Б.3 – Діаграма Ганта проекту

Аналіз ризиків. Ризик – ймовірнісна подія, яка може позитивно чи негативно вплинути на проект. Причиною виникнення ризиків є невизначеності, існуючі в кожному проекті. Ризики можуть бути «відомі» - ті, які визначені, оцінені, для яких можливе планування. Ризики «невідомі» - ті, які не ідентифіковані і не можуть бути прогнозовані. Хоча специфічні ризики і умови їх виникнення не визначені, але більшу частину ризиків можна передбачити.

Ідентифікація ризиків - визначення ризиків, здатних вплинути на проект, і документування їх характеристик.

Ідентифікація ризиків визначає, які ризики здатні вплинути на проект, і документує характеристики цих ризиків. Ідентифікація ризиків не буде ефективною, якщо вона не буде проводитися регулярно протягом реалізації проекту.

Ідентифікація ризиків повинна залучати якомога більше учасників: менеджерів проекту, замовників, користувачів, незалежних фахівців.

Класифікація ризиків:

1. За імовірністю виникнення:
   - слабо ймовірнісні (Negligible) ;
   - мало ймовірнісні (Low);
   - імовірні (Medium);
   - досить імовірні (High);
   - майже імовірні (Critical).
2. За величиною впливу:
   - Мінімальна (Negligible);
   - Низька (Low);
   - Середня (Medium);
   - Висока (High);
   - Максимальна (Critical).

На основі цих даних була проведена класифікація ризиків для даного проекту, що наведена в таблиці Б.3.

Таблиця Б.3 – Класифікація ризиків

№	Назва ризику	Ймовірність	Величина втрат	Рівень ризку	План дій
1	Зміна ТЗ на етапі розробки	2	4	Середній	Проведення переаналізу проектування, стрімке впровадження змін у проекті
2	Недотримання термінів календарного плану	2	3	Середній	
3	Технологічні ризики, пов'язані з розробкою і налагодженням технології для розповсюдження завдань серед команди	3	5	Високий	Детальне проектування системи, налагодження роботи алгоритму обмінами задач
4	Збій в роботі готового продукту	3	3	Середній	Дотримання планування розробки та технічного завдання, аналіз пропозицій керівника.

Продовження табл. Б.3

5	Виникнення незапланованих робіт по проекту	3	4	Середній	Аналіз необхідних робіт та виділення резерву з ресурсів для можливих майбутніх додаткових робіт
6	Висока залежність від ключових співробітників (Недостатня кваліфікація для створення чи використання технологій)	1	5	Середній	Відбір якісних співробітників або вивчення нових технологій для вдалого використання
7	Залежність від ключових співробітників (хвороба)	2	2	Низький	Підтримка емоційного стану та здоров’я учасників проекту за рахунок фізичних вправ та часу на відпочинок

Продовження табл. Б.3

8	Проблеми з хостингом чи серверами	1	2	Низький	Проведення аналізу існуючих хостингів та серверів на коефіцієнт помилки та к-сть помилок при роботі.
9	Втрата складових проекту через технічні проблеми	2	5	Середній	Проведення копіювання системних даних та версіонування за рахунок систем контролю версій
10	Звуження термінів здачі проекту	1	3	Низький	Розподілення часу для вдалого управління ресурсами проекту

Використовуючи дану класифікацію, була побудована матриця ризиків, що представлена в таблиці Б.4.

Таблиця Б.4 - Матриця ризиків

5					
4					
3			**4**	**5**	**3**
2		**7**	**2**	**1**	**9**
1		**8**	**10**		**6**
Ймовірність / Величина втрат	1	2	3	4	5

Формування бюджету. Завершальним етапом планування робіт є етап формування бюджету для даного проекту.

В управлінні проектами використовують декілька термінів, пов’язаних з фінансуванням проекту: кошторис проекту, бюджет проекту, план фінансування продукту проекту. При цьому їх можуть застосовувати стосовно:

— всього життєвого циклу проекту;

— моменту отримання продукту проекту;

— фази реалізації проекту.

Кошторис продукту проекту – це загальні майбутні витрати, які необхідні безпосередньо для створення продукту проекту. Тобто це витрати на фінансування всіх робіт, передбачених WBS - структурою проекту.

Бюджет продукту проекту – це кошторис продукту проекту, розподілений в часі на основі календарного плану реалізації робіт або за окремими WBS елементами. План фінансування – це кошторис продукту проекту в розрізі основних джерел фінансування робіт з проекту. Розрахуємо бюджет заробітної плати.

На початку реалізації даного процесу необхідно визначити учасників проектів, які безпосередньо приймали участь в етапах реалізації задачі. На основі структури OBS було виділено таких працівників\виконавців:

— розробник

— менеджер проекту

— консультант проекту

— тестувальник

В таблиці Б.5 приведена заробітна плата кожного учасника проекту з урахуванням різної кількості робочих годин кожного з них.

Таблиця Б.5 – Заробітна плата учасників проекту

Проектування бази даних	15 000 грн.
Створення дизайну інтерфейсу	12 000 грн.
Створення візуального інтерфейсу	15 000 грн.
Проектування та реалізація серверної частини системи та всіх необхідних шарів.	20 000 грн.
Ціна бізнес-хостингу для системи	0 грн/міс.
Орендування серверів	9 000 грн/міс.
Всього	62 000 грн. + (9000 грн/міс)

ДОДАТОК В

В.1 Налаштування захисту

```
package vadim.volin.configuration;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.authentication.rememberme.InMemoryTokenRepositoryImpl;
import org.springframework.security.web.authentication.rememberme.PersistentTokenRepository;
import vadim.volin.services.impl.UserDetailsServiceImpl;

@Configuration
@EnableWebSecurity
@ComponentScan("vadim.volin")
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    private AuthenticationProvider authenticationProvider;

    @Autowired
    @Qualifier("daoAuthenticationProvider")
    public void setAuthenticationProvider(AuthenticationProvider authenticationProvider) {
        this.authenticationProvider = authenticationProvider;
    }

    @Autowired
    public void configureAuthManager(AuthenticationManagerBuilder authenticationManagerBuilder) {
        authenticationManagerBuilder.authenticationProvider(authenticationProvider);
    }

    @Bean
    public DaoAuthenticationProvider daoAuthenticationProvider() {
```

```
    DaoAuthenticationProvider daoAuthenticationProvider = new DaoAuthenticationProvider();
    daoAuthenticationProvider.setPasswordEncoder(bCryptPasswordEncoder());
    daoAuthenticationProvider.setUserDetailsService(userDetailsService());
    return daoAuthenticationProvider;
}

@Bean
public UserDetailsService userDetailsService() {
    return new UserDetailsServiceImpl();
}

@Bean
public BCryptPasswordEncoder bCryptPasswordEncoder() {
    return new BCryptPasswordEncoder(10);
}

@Bean
public PersistentTokenRepository persistentTokenRepository() {
    return new InMemoryTokenRepositoryImpl();
}

@Bean
public CustomLoginSuccessHandler customLoginSuccessHandler() {
    return new CustomLoginSuccessHandler();
}

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable();
    http.authorizeRequests()
            .antMatchers("/", "/login", "/logout", "/register").permitAll();

    http.authorizeRequests()
            .antMatchers("/first**", "/home**", "/dashboard**", "/reports**", "/statistics**", "/projects**")
            .access("hasAnyRole('ROLE_USER')");

    http.authorizeRequests().and().exceptionHandling().accessDeniedPage("/403");

    http.authorizeRequests().and().formLogin()//
            .loginProcessingUrl("/login") // Submit URL
            .loginPage("/login")//
            .defaultSuccessUrl("/first")
            .successHandler(customLoginSuccessHandler())
            .failureUrl("/login?error")
            .usernameParameter("usermail")
            .passwordParameter("password")
            .and().logout().logoutUrl("/logout")
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID")
            .logoutSuccessUrl("/login?logout=success");

    http.headers().frameOptions().disable();
```

```
        http.authorizeRequests().and() //
                .rememberMe().tokenRepository(persistentTokenRepository())
                .tokenValiditySeconds(24 * 60 * 60 * 60);

    }

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/resources/**", "/css/**", "/js/**");
    }

}
```

В.2 Проектний контроллер

```
package vadim.volin.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import vadim.volin.model.Project;
import vadim.volin.model.ProjectFile;
import vadim.volin.model.User;
import vadim.volin.services.ProjectFileService;
import vadim.volin.services.ProjectService;
import vadim.volin.services.UserService;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

@Controller
@SessionAttributes({"user"})
public class ProjectsController {

    @Autowired
    private UserService userService;

    @Autowired
```

```
    private ProjectService projectService;

    @Autowired
    private ProjectFileService projectFileService;

    @GetMapping("/projects")
    public String initPage(@ModelAttribute User user, Model model) {
        if (user.getRoles() == null || user == null || !user.getRoles().contains("ROLE_USER")) {
            return "redirect:/login";
        }
        user = userService.getByUserMail(user.getUsermail());
        model.addAttribute("pageName", "Projects");
        model.addAttribute("user", user);
        return "projects";
    }

    @PostMapping(value = "/projects/add", produces = {MediaType.APPLICATION_JSON_VALUE})
    @ResponseBody
    public ResponseEntity<?> addProject(@RequestParam(name = "project_name", defaultValue = "")
String project_name, @ModelAttribute User user, Model model) {
        if (user == null || project_name == null || project_name.equals("")) {
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
        }
        Project project = new Project(project_name);
        user.getProjects().add(project);
        User update = userService.editUser(user);
        project = update.getProjects().stream().filter(project1 ->
project1.getProject_name().equals(project_name)).findFirst().get();
        model.addAttribute("user", update);
        return new ResponseEntity(String.valueOf(project.getProject_id()), HttpStatus.OK);
    }

    @GetMapping(value = "/projects/{user_id}/{project_id}")
    public String loadPersonalProjectPage(@PathVariable String user_id, @PathVariable String
project_id, @ModelAttribute User user, Model model) {
        if (user == null || user.getRoles() == null || !user.getRoles().contains("ROLE_USER")
                || user_id == null) {
            return "redirect:/login";
        }
        if (project_id == null) {
            return "redirect:/projects";
        }
        user = userService.getByUserMail(user.getUsermail());
        boolean hasProject = false;
        Project project = null;
        List<Project> projectList = user.getProjects();
        for (int i = 0; i < projectList.size(); i++) {
            if (projectList.get(i).getProject_id().equals(Integer.parseInt(project_id))) {
                hasProject = true;
                project = projectList.get(i);
                model.addAttribute("pageName", project.getProject_name());
                model.addAttribute("project", project);
```

```
            model.addAttribute("user", user);
            break;
         }
      }
      if (hasProject) {
         return "project-info";
      } else {
         return "redirect:/projects";
      }
   }

   @PostMapping(value = "/projects/remove", produces =
{MediaType.APPLICATION_JSON_VALUE})
   @ResponseBody
   public ResponseEntity<?> removeProject(
         @RequestParam(name = "projectId", defaultValue = "") String projectId, @ModelAttribute User
user, Model model
   ) {
      if (user == null || projectId == null || projectId.equals("")) {
         return new ResponseEntity(HttpStatus.BAD_REQUEST);
      }
      Project project = null;
      for (int i = 0; i < user.getProjects().size(); i++) {
         if (user.getProjects().get(i).getProject_id().equals(Integer.parseInt(projectId))) {
            project = user.getProjects().get(i);
            user.getProjects().remove(project);
            User update = userService.editUser(user);
            if (project.getTeam().isEmpty()) {
               projectService.removeProject(project);
            }
            model.addAttribute("user", update);
            break;
         }
      }
      return new ResponseEntity("Deleted!", HttpStatus.OK);
   }

   @PostMapping(value = "/projects/{id}/add/user", produces =
{MediaType.APPLICATION_JSON_VALUE})
   @ResponseBody
   public ResponseEntity<String> addUserToProject(
         @RequestParam(name = "membermail", defaultValue = "") String membermail, @PathVariable
String id, @ModelAttribute User user, Model model
   ) {
      if (user == null || membermail == null || membermail.equals("")) {
         return new ResponseEntity(HttpStatus.BAD_REQUEST);
      }
      User futureUser = userService.getByUserMail(membermail);
      if (futureUser == null) {
         return ResponseEntity.badRequest().body("User not found");
      }
      for (int i = 0; i < user.getProjects().size(); i++) {
```

```
            if (user.getProjects().get(i).getProject_id().equals(Integer.parseInt(id))) {
                if (user.getProjects().get(i).getTeam().contains(futureUser)
                        && futureUser.getProjects().contains(user.getProjects().get(i))) {
                    return ResponseEntity.badRequest().body("User already added!");
                } else {
                    user.getProjects().get(i).getTeam().add(futureUser);
                    futureUser.getProjects().add(user.getProjects().get(i));
                    futureUser = userService.editUser(futureUser);
                    projectService.editProject(user.getProjects().get(i));
                    model.addAttribute("user", user);
                    return ResponseEntity.ok("User added!");
                }
            }
        }
        return ResponseEntity.badRequest().body("Try later!");
    }

    @PostMapping("/projects/{id}/upload/file")
    @ResponseBody
    public ResponseEntity<?> uploadProjectFile(
            @RequestBody MultipartFile file, @PathVariable String id, @ModelAttribute User user, Model
model
    ) {
        if (user.getRoles() == null || user == null || !user.getRoles().contains("ROLE_USER")) {
            return ResponseEntity.badRequest().body("Try later");
        }
        if (file.isEmpty()) {
            model.addAttribute("message", "Please, choose file!");
            return ResponseEntity.badRequest().body("Please, select image for uploading!");
        }
        try {
            Project project = user.getProjects().stream()
                    .filter(project1 -> (
                            project1.getProject_id().equals(Integer.valueOf(id))
                    ))
                    .findFirst()
                    .get();

            byte[] bytes = file.getBytes();
            Path path = Paths.get(
                    "/home/vadim/Documents/Spring/springMVC-courses/target/springMVC-
courses/manluck_data/projects/"
//                  "/manluck_data/projects/"
                            + id + "/"
                            + file.getOriginalFilename()
            );
            Files.createDirectories(path.getParent());

            Files.write(path, bytes);
            ProjectFile projectFile = new ProjectFile("/manluck_data/projects/" + id + "/" +
file.getOriginalFilename());
            if (project.getProjectFiles().contains(projectFile)) {
```

```
            return ResponseEntity.badRequest().body("File already upload");
        } else {
            project.getProjectFiles().add(projectFile);
            projectFile.setProject(project);
            projectFile = projectFileService.addProjectFile(projectFile);
            model.addAttribute("user", user);
        }
    } catch (IOException e) {
        e.printStackTrace();
        return ResponseEntity.badRequest().body("Please, select file for uploading!");
    }
    return ResponseEntity.ok("Successfully upload!");
}

@PostMapping(value = "/projects/remove/user", produces =
{MediaType.APPLICATION_JSON_VALUE})
@ResponseBody
public ResponseEntity<String> removeUserFromProject(
        @RequestParam(name = "projectId", defaultValue = "") String projectId, @RequestParam(name
= "usermail", defaultValue = "") String usermail, @ModelAttribute User user, Model model
) {
    if (user == null || projectId == null || projectId.equals("") || usermail == null || usermail.equals("")) {
        return ResponseEntity.badRequest().body("Try later");
    }
    for (int i = 0; i < user.getProjects().size(); i++) {
        if (user.getProjects().get(i).getProject_id().equals(Integer.parseInt(projectId))) {
            User removedUser = userService.getByUserMail(usermail);
            if (removedUser == null) {
                return ResponseEntity.badRequest().body("User not found!");
            }

            boolean removeProject = false;
            for (int j = 0; j < removedUser.getProjects().size(); j++) {
                if (removedUser.getProjects().get(j).getProject_id().equals(Integer.parseInt(projectId))) {
                    removeProject = removedUser.getProjects().remove(removedUser.getProjects().get(j));
                    userService.editUser(removedUser);
                }
            }
            boolean removeUser = false;
            for (int j = 0; j < user.getProjects().get(i).getTeam().size(); j++) {
                if (user.getProjects().get(i).getTeam().get(j).equals(removedUser)) {
                    removeUser = user.getProjects().get(i).getTeam().remove(removedUser);
                    projectService.editProject(user.getProjects().get(i));
                }
            }
            if (user.getProjects().get(i).getTeam().isEmpty()) {
                projectService.removeProject(user.getProjects().get(i));
            }
            model.addAttribute("user", user);
            if (removeUser & removeProject) {
                return ResponseEntity.ok("Deleted!");
            }
```

```
            }
        }
        return ResponseEntity.badRequest().body("Try again");
    }


    @GetMapping(value = "/tasks/{project_id}")
    public String initPrjTasksPage(@PathVariable String project_id, @ModelAttribute User user, Model
model) {
        if (user.getRoles() == null || user == null || !user.getRoles().contains("ROLE_USER")) {
            return "redirect:/login";
        }
        if (project_id == null || project_id.equals("")) {
            return "redirect:/projects";
        }

        user = userService.getByUserMail(user.getUsermail());
        for (int i = 0; i < user.getProjects().size(); i++) {
            if (user.getProjects().get(i).getProject_id().equals(Integer.parseInt(project_id))) {
                model.addAttribute("pageName", "" + user.getProjects().get(i).getProject_name() + "-tasks");
                model.addAttribute("project", user.getProjects().get(i));
                model.addAttribute("user", user);
                return "project-tasks";
            }
        }
        model.addAttribute("user", user);
        return "redirect:/projects";
    }
    //    TODO: sync updates tasks for team members;
    @PostMapping(value = "/projects/{project_id}/save/tasks", produces =
{MediaType.APPLICATION_JSON_VALUE})
    @ResponseBody
    public ResponseEntity<String> saveProjectTasks(@RequestBody String jsonTasks, @PathVariable
String project_id, @ModelAttribute User user, Model model) {
        if (user.getRoles() == null || user == null || !user.getRoles().contains("ROLE_USER")) {
            return ResponseEntity.badRequest().body("Try later");
        }
        for (int i = 0; i < user.getProjects().size(); i++) {
            if (user.getProjects().get(i).getProject_id().equals(Integer.parseInt(project_id))) {
                model.addAttribute("pageName", "" + user.getProjects().get(i).getProject_name() + "-tasks");
                if (jsonTasks != null) {
                    if (!jsonTasks.equals(user.getProjects().get(i).getProject_tasks())) {
                        user.getProjects().get(i).setProject_tasks(jsonTasks);
                        projectService.editProject(user.getProjects().get(i));
                        user = userService.getByUserMail(user.getUsermail());
                        model.addAttribute("user", user);
                        return ResponseEntity.ok("Update!");
                    }
                    return ResponseEntity.status(HttpStatus.NOT_MODIFIED).body("Updated!");
                }
            }
        }
```

```
        model.addAttribute("user", user);
        return ResponseEntity.badRequest().body("Try later"); }
}
```

В.3 Контроллер звітів

```
package vadim.volin.controllers;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDPage;
import org.apache.pdfbox.pdmodel.PDPageContentStream;
import org.apache.pdfbox.pdmodel.font.PDFont;
import org.apache.pdfbox.pdmodel.font.PDType1Font;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.SessionAttributes;
import vadim.volin.model.Project;
import vadim.volin.model.User;
import vadim.volin.services.ProjectService;
import vadim.volin.util.MediaTypeUtils;
import vadim.volin.util.ProjectUtils;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletResponse;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Map;

@Controller
@SessionAttributes({"user"})
public class ReportsController {

    @Autowired
    private ServletContext servletContext;

    @Autowired
    private ProjectService projectService;

    @GetMapping(value = "/reports")
```

```
    public String getReportPage(@ModelAttribute User user, Model model) {

        model.addAttribute("user", user);
        model.addAttribute("pageName", "Reports");
        return "reports";
    }

    @GetMapping(value = "/reports/{project_id}")
    public ResponseEntity<?> getProjectReport(@PathVariable String project_id, @ModelAttribute User
user, Model model, HttpServletResponse response) {
        // Прежде всего стоит проверить, если необходимо, авторизован ли пользователь и имеет
достаточно прав на скачивание файла. Если нет, то выбрасываем здесь Exception
        //Авторизованные пользователи смогут скачать файл
        if (project_id == null || project_id.equals("")) {
            return ResponseEntity.badRequest().body("Project not found!");
        }
        Project project = projectService.getProject(Integer.parseInt(project_id));
        if (project == null) {
            return ResponseEntity.badRequest().body("Project not found!");
        }
        PDDocument pdfDocument = null;
        String fileName = "/home/vadim/Documents/Spring/springMVC-courses/target/springMVC-
courses/manluck_data/projects/reports/" +
//        String fileName = "/manluck_data/projects/reports/" +
                project_id + "/" + project.getProject_name() + "_report.pdf";
        File pdfFile = new File(fileName);
        try {
            Files.createDirectories(pdfFile.toPath().getParent());
        } catch (IOException exception) {
            exception.printStackTrace();
        }

        try {
            pdfDocument = new PDDocument();
            PDPage page = new PDPage();

            PDPageContentStream contents = new PDPageContentStream(pdfDocument, page);
            contents.beginText();
            PDFont font = PDType1Font.TIMES_BOLD;
            contents.setFont(font, 20);
            contents.newLineAtOffset(200, 720);
            contents.showText("Project-" + project.getProject_name());
            contents.endText();

            contents.beginText();
            font = PDType1Font.TIMES_ROMAN;
            contents.setFont(font, 14);
            contents.newLineAtOffset(40, 660);
            contents.newLine();
            contents.showText("Team: " + project.getTeam().size() + " members.");
            contents.endText();
```

```
            contents.beginText();
            contents.newLineAtOffset(40, 640);
            contents.newLine();
            contents.showText("Process:");
            contents.endText();

            int ty = 640;
            Map<Integer, String> columnData = ProjectUtils.getTasksData(project.getProject_tasks());
            for (Map.Entry<Integer, String> mapEntry : columnData.entrySet()) {
                contents.beginText();
                ty = ty - 20;
                contents.newLineAtOffset(60, ty);
                contents.newLine();
                contents.showText("" + mapEntry.getValue() + "-" + mapEntry.getKey() + " cards");
                contents.endText();
            }
            contents.close();

            pdfDocument.addPage(page);
            pdfDocument.save(pdfFile);
        } catch (IOException exception) {
            exception.printStackTrace();
        } finally {
            try {
                if (pdfDocument != null) {
                    pdfDocument.close();
                }
            } catch (IOException exception) {
            }
        }

        MediaType mediaType = MediaTypeUtils.getMediaTypeForFileName(this.servletContext,
fileName);

        File file = new File(fileName);
        InputStreamResource resource = null;
        try {
            resource = new InputStreamResource(new FileInputStream(file));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        return ResponseEntity.ok()
                // Content-Disposition
                .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=" + file.getName())
                // Content-Type
                .contentType(mediaType)
                // Contet-Length
                .contentLength(file.length()) //
                .body(resource);
    }
}
```

В.4 Контроллер зміни інформації користувача

```
package vadim.volin.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import vadim.volin.model.User;
import vadim.volin.services.UserService;

import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

@Controller
@SessionAttributes("user")
public class UpdateUserController {

   @Autowired
   private UserService userService;

   @PostMapping("/update/user/img")
   @ResponseBody
   public ResponseEntity<String> handleUploadImage(@RequestBody MultipartFile file,
@ModelAttribute User user, Model model) {
      if (user.getRoles() == null || user == null || !user.getRoles().contains("ROLE_USER")) {
         return new ResponseEntity("Error!", HttpStatus.BAD_REQUEST);
      }
      if (file.isEmpty()) {
         model.addAttribute("message", "Please, choose file!");
         return new ResponseEntity("Please, select image for uploading!", HttpStatus.BAD_REQUEST);
      }
      try {
         byte[] bytes = file.getBytes();
         String filename = user.getUsername().toLowerCase().replaceAll("\\s+", "") + ".png";
         Path path = Paths.get("/home/vadim/Documents/Spring/springMVC-courses/target/springMVC-
courses/manluck_data/user_img/"
//            Path path = Paths.get("/manluck_data/user_img/"
               + filename);
         Files.createDirectories(path.getParent());

         Files.write(path, bytes);
         user.setUser_img("/manluck_data/user_img/" + filename);
```

```
            userService.editUser(user);
            model.addAttribute("user", user);
        } catch (IOException e) {
            model.addAttribute("message", "Server error, please, try again!");
            return new ResponseEntity("Error: " + e.getMessage(), HttpStatus.BAD_REQUEST);
        }
        return new ResponseEntity("Successfully upload!", HttpStatus.OK);
    }

    @PostMapping(value = "/update/user", produces = {MediaType.APPLICATION_JSON_VALUE})
    @ResponseBody
    public ResponseEntity<?> updateUserInfo(@ModelAttribute User user, Model model) {
        if (user == null) {
            return new ResponseEntity(HttpStatus.BAD_REQUEST);
        }
        User update = userService.editUser(user);
        model.addAttribute("user", update);
        return new ResponseEntity("Successfully upload!", HttpStatus.OK);
    }

    @RequestMapping(value = "/update/deactivate", method = RequestMethod.POST)
    public String deactivateUser(@ModelAttribute User user) {
        if (user == null) {
            return "Error on deactivate";
        }
        user.setActive(false);
        userService.editUser(user);
        return "redirect:/login?logout=ok";
    }

}
```

В.5 База даних

```
drop table "team_user" restrict;
drop table "project_files" restrict;
drop table "project" restrict;
drop table "city" restrict;
drop table "country" restrict;
drop table "role" restrict;
drop table "user" restrict;



/* create domains */



/* create tables */
```

```
create table "user"
(
	"user_id" serial not null unique,
	"username" varchar,
	"password" varchar,
	"user_img" varchar,
	"usermail" varchar unique,
	"userphone" varchar unique,
	"company" varchar,
	"position" varchar,
	"active" boolean,
	"userboardjson" varchar,
	"role_id" integer not null,
	"city_id" integer not null,
	"country_id" integer not null,
 primary key ("user_id","role_id")
) without oids;


create table "role"
(
	"role_id" serial not null unique,
	"name" varchar unique,
 primary key ("role_id")
) without oids;


create table "country"
(
	"country_id" serial not null unique,
	"country" varchar,
 primary key ("country_id")
) without oids;


create table "city"
(
	"city_id" serial not null unique,
	"city" varchar,
	"country_id" integer not null,
 primary key ("city_id","country_id")
) without oids;


create table "project"
(
	"project_id" serial not null unique,
	"project_name" varchar,
 primary key ("project_id")
) without oids;
```

```
create table "project_files"
(
        "project_files_id" serial not null unique,
        "file_path" varchar,
        "project_id" integer not null,
 primary key ("project_files_id")
) without oids;


create table "team_user"
(
        "team_user_id" serial not null unique,
        "user_id" integer not null,
        "project_id" integer not null,
        "role_id" integer not null,
 primary key ("team_user_id")
) without oids;



/* create tab 'others' for selected tables */


/* create alternate keys */



/* create indexes */



/* create foreign keys */

alter table "team_user" add  foreign key ("user_id","role_id") references "user" ("user_id","role_id") on update restrict on delete restrict;

alter table "user" add  foreign key ("role_id") references "role" ("role_id") on update restrict on delete restrict;

alter table "city" add  foreign key ("country_id") references "country" ("country_id") on update restrict on delete restrict;

alter table "user" add  foreign key ("city_id","country_id") references "city" ("city_id","country_id") on update restrict on delete restrict;

alter table "project_files" add  foreign key ("project_id") references "project" ("project_id") on update restrict on delete restrict;

alter table "team_user" add  foreign key ("project_id") references "project" ("project_id") on update restrict on delete restrict;
```

В.7 Лістинг коду частини відображень

В.7.1 Сторінка привітання

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>

<head>
        <title>ManLuck-service</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="resources/css/landing.css" />
</head>

<body>
        <div class="first">
                <div class="content">
                        <div class="container">
                                <div class="service-name">ManLuck</div>
                                <div class="titles">
                                        <p class="main-talk">Management is easier with us.</p>
                                        <p class="opportunities">Creating projects, managing tasks,
generating reports, analyzing statistics, all this is in our service.</p>
                                </div>
                                <div class="actions">
                                        <a href="/login" class="action-btn">Sign in</a>
                                        <a href="/register" class="action-btn">Sign up</a>
                                </div>
                        </div>
                </div>
        </div>
</body>

</html>
```

В.7.2 Сторінка проектів

```
<html xmlns:th="http://www.w3.org/1999/xhtml">
<head>
   <title>ManLuck-projects</title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" type="text/css" href="../../resources/css/font-awesome-4.7.0/css/font-
awesome.min.css">
   <link rel="stylesheet" type="text/css" href="../../resources/css/main-page.css">
   <link rel="stylesheet" type="text/css" href="../../resources/css/projects.css">
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
   <script src=".../../resources/js/project.js"></script>
</head>
```

```
<body>
<!--<div th:replace="components/header :: header"></div>-->
<main class="main">
   <div th:replace="components/sidebar :: sidebar"></div>
   <section class="content">
      <div class="row">
         <span id="action"><i class="fa fa-plus-square" aria-hidden="true"></i> project</span>
         <div id="project-add" hidden>
            <form action="#" th:action="@{/project/add/new}" id="project-form"
enctype="multipart/form-data"
                  method="POST" name="create_project">
               <div class="project-row">
                  <label for="project_name">Project name</label>
                  <input id="project_name" type="text" name="project_name" pattern="(\D{2,})">
               </div>
               <div class="project-row">
                  <input id="close" type="button" value="Close">
                  <input id="send_new" type="submit" value="Create">
               </div>
            </form>
         </div>
      </div>
      <div class="projects" th:if="${user.projects} != null">
         <div class="load" th:each="project : ${user.projects}" hidden>
            <script th:inline="javascript">
               loadProjects([[${project.project_id}]], [[${project.project_name}]], [[${user.id}]]);
            </script>
         </div>
      </div>
   </section>
</main>
<script th:inline="javascript" type="text/javascript">
   $(document).ready(function (e) {
      document.getElementById('action').addEventListener('click', function () {
         document.getElementById('project-add').hidden = false;
         document.getElementById('project_name').focus();
         document.getElementById('project_name').setAttribute('required', 'true');
      });
      document.getElementById('close').addEventListener('click', function () {
         document.getElementById('project-add').hidden = true;
         document.getElementById('project_name').removeAttribute('required');
      });
      $('form[name=create_project]').submit(function (e) {
         const proj_name = document.getElementById('project_name').value;
         $.ajax({
            type: "POST",
            global: false,
            url: '/projects/add',
            data: $('form[name=create_project]').serializeArray(),
            dataType: 'text',
            timeout: 600000,
            success: function (data, status) { // в случае успешного завершения
```

```
                let proj = data;
                let projectElement = new Project(proj, proj_name, [[${user.id}]]);
                alert('Project ' + proj_name + ' added!');
            },
            error: function (data, status) { // в случае провала
                alert('Try later!');
            }
        });
        document.getElementById('project-add').hidden = true;
        e.preventDefault();
    });
});
</script>
</body>

</html>
```

В.7.3 Приватна сторінка проекту

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="windows-1251">
    <title th:text="'ManLuck-' + ${project.project_name}"></title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="../../resources/css/font-awesome-4.7.0/css/font-
awesome.min.css">
    <link rel="stylesheet" type="text/css" href="../../resources/css/main-page.css">
    <link rel="stylesheet" type="text/css" href="../../resources/css/dashboard-page.css">
    <link rel="stylesheet" type="text/css" href="../../resources/css/projects.css">

</head>

<body>
<!--<div th:replace="components/header :: header"></div>-->
<main class="main">
    <div th:replace="components/sidebar :: sidebar"></div>
    <section class="content central">
        <div class="row">
            <div class="project-tools">
                <a th:href="@{'/tasks/' + ${project.project_id}}">
                    <div class="project-action"><i class="fa fa-th-large" aria-hidden="true"></i></div>
                </a>
                <a th:href="@{'/reports/' + ${project.project_id}}" target="_blank" download="true">
                    <div class="project-action"><i class="fa fa-files-o" aria-hidden="true"></i></div>
                </a>
                <a th:href="@{'/statistics/' + ${project.project_id}}">
                    <div class="project-action"><i class="fa fa-area-chart" aria-hidden="true"></i></div>
                </a>
            </div>
```

```
    </div>

    <div class="row">
      <div class="project-tools">
        <span id="add-user" class="action"><i class="fa fa-plus-square" aria-hidden="true"></i>
user</span>
        <form id="file_upload" name="file_upload" enctype="multipart/form-data"
          th:action="@{'/projects/' + ${project.project_id} + '/upload/file'}" method="post">
          <label for="add-prj-file" id="add-file">
            <span class="action">
              <input id="add-prj-file" type="file" name="file">
              <i class="fa fa-plus-square" aria-hidden="true"></i> file
            </span>
          </label>
        </form>
        <div id="user-add" hidden>
          <form action="#" th:action="@{'/projects/' + ${project.project_id} + '/add/user'}" id="user-
form"
            enctype="multipart/form-data"
            method="POST" name="add-user">
            <div class="user-add-row">
              <label for="usermail">E-mail</label>
              <input id="usermail" type="email" name="membermail"
                pattern="^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-
]+)*$"
              >
            </div>
            <div class="user-add-row">
              <input id="close" type="button" value="Close">
              <input id="send_new" type="submit" value="Add">
            </div>
          </form>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="team">
        <div class="row team-title">Team</div>
        <div class="user-row">
          <div class="user-data-set">
            <div class="user-data-elem">Image</div>
            <div class="user-data-elem">Name</div>
            <div class="user-data-elem">E-mail</div>
            <div class="user-data-elem">Phone</div>
            <div class="user-data-elem">Country</div>
            <div class="user-data-elem">City</div>
            <div class="user-data-elem">Delete?</div>
          </div>
        </div>
        <div class="user-row" th:each="user : ${project.team}">
          <div class="user-data-set team-data">
            <div class="user-data-elem">
```

```
                    <img src="#"
                        th:src="(${user.user_img} eq null) ?
@{/resources/manluck_data/user_img/default.png} : @{${user.user_img}}">
                </div>
                <div class="user-data-elem" th:text="${user.username}"></div>
                <div class="user-data-elem" th:text="${user.usermail}"></div>
                <div class="user-data-elem" th:text="${user.userphone}"></div>
                <div class="user-data-elem" th:text="${user.country}"></div>
                <div class="user-data-elem" th:text="${user.city}"></div>
                <div class="user-data-elem"
                    th:onclick="removeMember([[${project.project_id}]], [[${user.usermail}]]);">
                    <i class="fa fa-window-close" aria-hidden="true"></i>
                </div>
            </div>
        </div>
    </div>
</section>
<section id="filebar">
    <div class="row file-title">Files</div>
    <div class="row" th:if="${project.projectFiles} != null">
        <div class="files">
            <div class="project-file" th:each="file : ${project.projectFiles}">
                <a th:href="@{${file.file_path}}" target="_blank" download="true">
                    <span class="file-part-1">
                        <i class="fa fa-file-text-o" aria-hidden="true"></i>
                    </span>
                    <span class="file-part-2" th:text="${file.getFileName()}"></span>
                </a>
            </div>
        </div>
    </div>
</section>
</main>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script th:inline="javascript" type="text/javascript">
    function removeMember(projectId = null, usermail = null) {
        $.ajax({
            type: "POST",
            global: false,
            url: '/projects/remove/user',
            data: {projectId: projectId, usermail: usermail},
            dataType: 'text',
            timeout: 600000,
            success: function (data, status) { // â ñëó÷àå óñïåøíîãî çàâåðøåíèÿ
                let message = data;
                $(".team").load(window.location.href + " .team");
                alert(message);
            },
            error: function (data, status) { // â ñëó÷àå ïðîâàëà
                let message = data;
                alert(message);
```

```
        }
    });
}

$(document).ready(function (e) {
    document.getElementById('add-user').addEventListener('click', function () {
        document.getElementById('user-add').hidden = false;
        document.getElementById('usermail').focus();
        document.getElementById('usermail').setAttribute('required', 'true');
    });
    document.getElementById('close').addEventListener('click', function () {
        document.getElementById('user-add').hidden = true;
        document.getElementById('usermail').removeAttribute('required');
    });

    $('form[name=file_upload]').change(function (e) {
        e.preventDefault(); // äåëàåì îòìåíó äåéñòâèÿ áðàóçåðà è ôîðìèðóåì ajax
        var formData = new FormData($('#file_upload')[0]);
        console.log(formData.keys())
        $.ajax({
            type: 'POST', // òèï çàïðîñà
            url: $(this).attr('action'), // êóäà áóäåì îòïðàâëÿòü, ìîæíî ÿâíî óêàçàòü
            data: formData, // äàííûå, êîòîðûå ïåðåäàåì
            cache: false, // êýø è ïðî÷èå íàñòðîéêè ïëàòü èìåííî òàê (äëÿ ôàéëîâ)
            contentType: false, // íóæíî óêàçàòü òèï êîíòåíòà false äëÿ êàðòèíêè(ôàéëà)
            processData: false, // äëÿ ïåðåäà÷è êàðòèíêè(ôàéëà) íóæíî false
            timeout: 600000,
            success: function (data, status) { // â ñëó÷àå óñïåøíîãî çàâåðøåíèÿ
                let message = data;
                $(".files").load(window.location.href + " .files");
                alert(message);
            },
            error: function (data, status) { // â ñëó÷àå ïðîâàëà
                let message = data;
                alert(message);
            }
        });
    });
    $('form[name=add-user]').submit(function (e) {
        e.preventDefault();
        $.ajax({
            type: "POST",
            global: false,
            url: $(this).attr('action'),
            data: $('form[name=add-user]').serializeArray(),
            dataType: 'text',
            timeout: 600000,
            success: function (data, status) { // â ñëó÷àå óñïåøíîãî çàâåðøåíèÿ
                let message = data;
                $(".team").load(window.location.href + " .team");
                alert(message);
            },
```

```
            error: function (data, status) { // â ñëó÷àå ïðîâàëà
                let message = data;
                alert(message);
            }
        });
        document.getElementById('user-add').hidden = true;
    });
});
</script>
</body>
</html>
```

В.8 Каскадна таблиця стилів для сторінки привітання

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    text-decoration: none;
    outline: none;
    font-family: 'Tahoma', 'Geneva', sans-serif;
    -webkit-font-smoothing: antialiased;
}

html, body {
    height: 100%;
    width: 100%;
}

.first {
    background: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url(../img/landing-img/first-bg-
min.jpg);
    background-position: center;
    background-repeat: no-repeat;
    background-size: cover;
    height: 100%;
    width: 100%;

}

.content {
    backdrop-filter: blur(3px);
    background-color: rgba(0, 0, 0, 0.37);
    -webkit-backdrop-filter: blur(3px);
    height: 100%;
    width: 100%;
}

.container {
    padding: 20% 0;
    margin: 0 auto;
```

```
    width: 940px;

    display: flex;
    flex-flow: column;
    -ms-align-items: center;
    align-items: center;
}

.service-name {
    font-size: 60px;
    color: hsla(0, 0%, 96%, .8);
    margin-bottom: 30px;
}

.main-talk,
.opportunities {
    font-size: 24px;
    color: hsla(0, 0%, 96%, 0.8);
    text-align: center;
    margin-bottom: 25px;
}

.opportunities {
    font-size: 16px;
    margin-bottom: 35px;
}

.actions {
    width: 50%;
    display: -webkit-flex;
    display: -moz-flex;
    display: -ms-flex;
    display: -o-flex;
    display: flex;
    justify-content: space-around;
}

.action-btn {
    text-align: center;
    width: 150px;
    padding: 10px;
    background-color: #4caf50;
    transition: 1s;
    color: unset;
    font-size: 14px;
    font-weight: bold;
    border-radius: 3px;
    -webkit-backface-visibility: hidden;
    -moz-backface-visibility: hidden;
    -ms-backface-visibility: hidden;
    backface-visibility: hidden;
    -ms-transform: translateZ(0); /* IE 9 */
```

```
    -webkit-transform: translateZ(0); /* Chrome, Safari, Opera */
    transform: translateZ(0);
}

.action-btn:hover {
    transition: 1s;
    background-color: #1db31d;
    -webkit-transform: scale(1.05);
    transform: scale(1.05);
}

.action-btn:focus {
    transition: .6s;
    background-color: #199017;
}
```

В.9 Каскадна таблиця стилів для сторінки проектів

```
* /*-------------------------------------------------------*/

.content {
    margin: 90px 50px 0 340px;
}

.central {
    margin: 90px 300px 0 120px;
}

#project-add, #user-add {
    position: absolute;
    margin: 15px 0 0 95px;
    background-color: #f5f5f5;
    width: 220px;
    padding: 10px;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    z-index: 10;
}

.user-add-row,
.project-row {
    display: flex;
    flex-flow: column;
    padding: 5px;
}

.user-add:last-child,
.project-row:last-child {
    flex-flow: row wrap;
    justify-content: space-around;
```

```
}

.user-add-row label,
.project-row label {
    margin-bottom: 10px;
    color: #1db31d;
}

#project_name {
    background-color: unset;
    color: #444444;
    border-bottom: 2px solid #444444;
    font-size: 16px;
}

#project_name:hover {
    border-bottom-color: #4caf50;
}

#project_name:active,
#project_name:focus {
    border-bottom-color: #1db31d;
}

#send_new, #close {
    cursor: pointer;
    padding: 10px;
    text-align: center;
    font-size: 14px;
    font-weight: bold;
    user-select: none;
    border: 2px solid #444444;
    color: #444444;
    border-radius: 5px;
    width: 40%;
}

#send_new:hover, #close:hover {
    border: 2px solid #4caf50;
    color: #4caf50;
}

#send_new:focus, #close:focus {
    border: 2px solid #1db31d;
    color: #1db31d;
}

#send_new:active, #close:active {
    border: 2px solid #1db31d;
    color: #1db31d;
}
```

```
.projects {
    display: -webkit-flex;
    display: -moz-flex;
    display: -ms-flex;
    display: -o-flex;
    display: flex;
    flex-flow: row wrap;
    align-content: center;
    align-items: center;
}

.project {
    width: calc(25% - 20px);
    display: flex;
    flex-flow: column;
    margin: 10px;
    border-radius: 5px;
    border: 0px solid #000;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    opacity: .7;
}

.project:hover {
    -webkit-transform: scale(1.05);
    -ms-transform: scale(1.05);
    transform: scale(1.05);
    transition: .7s;
    opacity: 1;
}

.title {
    display: flex;
    flex-flow: row wrap;
    align-items: center;
    background-color: #444444;
    justify-content: space-between;
}

#project-remove {
    /*width: 10%;*/
    padding: 15px;
    font-size: 32px;
    color: #f5f5f5;
    z-index: 100;
}

#project-remove:hover {
    color: #1db31d;
    transition: color 1.2s;
}
#project-remove:not(hover) {
    transition: color 1.2s;
```

```
}

.project-title {
    white-space: nowrap;
    text-overflow: ellipsis;
    overflow: hidden;
    padding: 15px;
    text-align: center;
    font-size: 16px;
    color: #f5f5f5;
}

.project:hover
.project-img {
    color: #1db31d;
    transition: color 1.2s;

}

.project:hover
.project-title {
    background-color: #444444;
    transition: background-color 1.2s;
    color: #f5f5f5;
}

.project:not(hover) {
    transition: .7s;
}

.project:not(hover)
.project-img {
    transition: color 1.2s;
}

.project:not(hover)
.project-title {
    transition: background-color 1.2s;
}

.project-img {
    text-align: center;
    vertical-align: middle;
    font-size: 80px;
    background-color: #f5f5f5;
    color: #4caf50;
    padding: 40px 0;
    width: 100%;
    bottom: 0;
}

.project-tools {
```

```
    display: flex;
    flex-flow: row wrap;
}

.project-action {
    background-color: #f5f5f5;
    color: #444444;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
    margin-right: 15px;
    font-size: 35px;
    padding: 20px 20px;
    align-content: center;
    align-items: center;
    display: flex;
    justify-content: center;
    align-self: center;
}

.project-action:hover {
    transition: .6s;
    color: #1db31d;
    transform: scale(1.05);
}

.project-action:not(hover) {
    transition: .6s;
}

#action {
    margin: 10px;
}

.action {
    padding: 10px 20px;
    margin: 0 10px 0 0;
}

.team {
    display: flex;
    flex-flow: column;
    background-color: #f5f5f5;
    color: #444444;
    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
}

.team-title {
    padding: 30px;
    font-size: 24px;
}
.user-data-set {
    display: flex;
    flex-flow: row wrap;
```

```
    padding: 10px 0;
    align-content: center;
    align-items: center;
    justify-content: space-around;
}

.user-data-elem {
    height: 100%;
    text-align: center;
    width: 14%;
    white-space: nowrap;
    text-overflow: ellipsis;
    overflow: hidden;
    transition: .6s;
}

.user-data-elem img {
    width: 40px;
    height: 40px;
    border-radius: 50%;
}

.team-data > .user-data-elem:last-child {
    font-size: 30px;
}

.team-data > .user-data-elem:last-child:hover {
    transition: .6s;
    color: #1db31d;
    -webkit-transform: scale(1.1);
    -ms-transform: scale(1.1);
    transform: scale(1.1);
}

#filebar {
    height: 100%; /* Full-height: remove this if you want "auto" height */
    position: fixed; /*Fixed Sidebar (stay in place on scroll) */
    z-index: 3; /* Stay on top */
    top: 0; /* Stay at the top */
    right: 0;
    overflow-x: hidden; /* Disable horizontal scroll */
    background-color: #444444;
    color: #c0c0c0;
    display: -webkit-flex;
    display: -moz-flex;
    display: -ms-flex;
    display: -o-flex;
    display: flex;
    flex-flow: column;
    width: 280px;
}
```

```
.file-title {
    padding: 30px 0 10px 30px;
    font-size: 24px;
}

.files {
    padding: 0 25px;
    display: flex;
    flex-flow: row wrap;
}

.project-file {
    font-size: 40px;
    margin: 5px 20px 5px 5px;
    width: calc(25% - 10px);
    text-align: center;
    transition: .6s;
}

.project-file a {
    text-decoration: none;
    color: #4caf50;
}

.project-file:hover,
.project-file:active {
    transition: .6s;
    color: #1db31d;
    -webkit-transform: scale(1.2);
    -ms-transform: scale(1.2);
    transform: scale(1.2);
}

#add-prj-file {
    width: 0.1px;
    height: 0.1px;
    opacity: 0;
    overflow: hidden;
    position: absolute;
    z-index: -1;
}

.file-part-2 {
    display: flex;
    white-space: nowrap;
    text-overflow: ellipsis;
    overflow: hidden;
    font-size: 16px;
    text-align: center;
}
```