МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток супроводження діяльності співробітника проектної організації у КЗАПР»

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

Виконавець роботи: студент групи ІТ-82-0 Шелехов Денис Володимирович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2022 р.

Науковий керівник _____ к.т.н., доц., Антипенко В.П.
(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-2022

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

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«__» _____ 2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Шелехов Денис Володимирович

1 Тема роботи *Web-додаток супроводження діяльності співробітника проектної організації у КЗАПР*

керівник роботи *Антипенко Вікторія Петрівна, к.т.н., доцент*,

затверджені наказом по університету від «27» квітня 2022 р. _____

2 Строк подання студентом роботи «10» червня 2022 р.

3 Вхідні дані до роботи технічне завдання на розробку web-додатку супроводження діяльності співробітника проектної організації у КЗАПР

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування web-додатку, розробка web-додатку

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

_____

_____

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 05.10.2021 _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	15.03.2022-25.03.2022	
2	Оформлення технічного завдання	25.03.2022-10.04.2022	
3	Проведення аналізу предметної області	10.04.2022-20.04.2022	
4	Проведення проектування серверної частини web-додатку	20.04.2022-30.04.2022	
5	Розробка серверної частини web-додатку	01.05.2022-23.05.2022	
6	Тестування серверної частини web-додатку	23.05.2022-25.05.2022	
7	Оформлення пояснювальної записки	25.05.2022-31.05.2022	

Студент _____ Шелехов Д.В.
(підпис)

Керівник роботи _____ к.т.н., доц. Антипенко В.П.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток супроводження діяльності співробітника проектної організації у КЗАПР».

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел із 22 найменування, чотирьох додатків. Загальний обсяг пояснювальної записки складає 53 сторінки, у тому числі 29 сторінок основного тексту, 2 сторінки списку використаних джерел, 22 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено web-додатку супроводження діяльності співробітника проектної організації у КЗАПР.

У першому розділі проведено огляд останніх досліджень за тематикою даного проекту. Проаналізовано аналоги розроблюваного web-додатку, визначено їх переваги та недоліки. Також було сформовано мету й задачі проекту, обрано засоби реалізації.

У другому розділі проведено структурно-функціональне моделювання, визначено варіанти використання web-додатку та спроектовано базу даних. У результаті були змодельовані контекстна діаграма IDEF0 та її декомпозиції, діаграма варіантів використання.

У третьому розділі наведено архітектуру web-додатку супроводження діяльності співробітника проектної організації у КЗАПР й описано процес його розробки, яка є результатом проектування. Проведено тестування даного програмноо продукту.

Ключові слова: WEB-ДОДАТОК, БАЗА ДАНИХ, JAVA, СЕРВЕР, КЗАПР, SPRING BOOT, РОЗРОБКА.

ЗМІСТ

ВСТУП

Із розвитком цивілізації кількість задач, проектів та іншої діяльності пов’язаної з бізнесом тільки зростає. Тому виникає необхідність оптимізувати якомога більше процесів на виробництві. Це призведе до покращення перебігу справ всередині будь-якої організації. Найкращим способом такої оптимізації є автоматизація процесів виконання проектів. А з цією роллю найкраще справляються програмні продукти.

Успішність того чи іншого бізнесу залежить від кількості, якості та складності виконуваних проектів. Але при збільшенні числа останніх виникає необхідність найму додаткового персоналу. При цьому, через зростання кількості завдань становиться досить складно вручну відслідковувати всі процеси в організації. Тому підприємства шукають рішення подолання даної проблеми. А саме спосіб зменшити навантаження на процес моніторингу проектів. Зараз ми живемо в час розвитку інформаційних технологій. Тому, найкращим способом вирішення цієї проблеми є розробка такого додатку, який буде відслідковувати весь життєвий цикл проектів.

При створенні будь-якого програмного продукту, слід розуміти, що його замовник може бажати уникнення розповсюдження інформації серед сторонніх осіб. Тому, важливою частиною розробки таких додатків є максимальний захист даних у ході обміну ними між робітниками організації. Для цього необхідно коректно реалізувати механізм передачі завдань. При його використанні працівники будуть отримувати тільки ті дані, які необхідні для, власне, виконання певної задачі. І підприємства машинобудівного профілю не є виключенням.

Отже, метою цього дослідження є розробка механізму обміну та захисту даних при їх передаванні між співробітниками проектної організації та зберіганні у КЗАПР.

Для досягнення поставленої мети потрібно виконати наступні задачі:

– визначити актуальність роботи, дослідити предметну область та існуючі аналоги;

– спроектувати структуру модулю на сервері;

- проаналізувати та обрати технології та мову розробки;
- реалізувати структуру модулю на сервері;
- розробити функціонал проектного рішення;
- провести тестування розробленого модулю.

Результати роботи були апробовані на науково-практичній конференції ІМА-2022 у Сумському державному університеті [1].

Згідно вищевказаному дана робота є актуальною, оскільки проблема автоматизації проектувальних робіт досить гостро стоїть у сучасному світі. Важливою особливістю проекту є впровадження web-інтерфейсу до подібного комплексу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Для організації безперебійного та контрольованого процесу виробництва машинобудівних об’єктів є розробка комплексу засобів автоматизації проектувальних робіт. Розробка механізму обміну та захисту даних між співробітниками проектної організації є одним з його модулей. КЗАПР створюється на основі web-сервісу [2]. Останній – це комбінація стандартів та протоколів, які дають можливість спілкуватися між клієнтом та сервером, представлений сервером або програмним забезпеченням, який забезпечує передачу файлів, здебільшого в XML [3] або JSON [4] форматах між базою даних та web-ресурсом.

Згідно джерелу [5], найпопулярнішими типами web-сервісів на даний момент є REST [6] (передача стану репрезентації) та SOAP [7] (простий протокол доступу до об’єктів). Останній – це протокол передачі структурованих повідомлень в форматі XML. REST – архітектурний стиль для розробки web-сервісів. Він не залежить від жодного протоколу, але майже в кожній його службі для регулювання операцій над ресурсами використовуються дієслова HTTP [8] та немає жодних обмежень щодо формату представлення.

При порівнянні даних типів web-сервісів, можна визначити, що перевагою REST є простота розуміння та розробки, оскільки він є архітектурним стилем на відміну від SOAP. Також об’єм переданих даних, а, отже, і пропускна здатність за допомогою REST є меншою. Це тому, що відправляються тільки дані, без іншої корисної інформації. На відміну від REST, SOAP працює по-іншому. Даний протокол передає дані дотримуючись стилю конверту. У такому вигляді інформація є кориснішою, але має більший об’єм.

Перевагою SOAP можна вважати асинхронність запитів. Завдяки WS-Security даний протокол робить передачу даних безпечнішою, але додаткові методи безпеки можна встановити до обох типів web-сервісів у залежності від задач проекту.

REST менше пов’язаний з сервером. Це є його перевагою перед SOAP. REST легко обслуговувати на сервері. Натомість SOAP складно підтримувати на клієнті й у разі збою зв’язку він є надійнішим, оскільки має вбудовану логіку повторної спроби. REST такої логіки не має.

Для реалізації даного проекту було враховано переваги та недоліки обох представлених типів web-сервісів. Вирішено використовувати REST, оскільки він є легшим у використанні, швидшим та універсальним. Зважаючи на мету дослідження, а саме забезпечення безпеки передачі даних, важливішими є саме переваги обраного web-серверу.

1.2 Аналіз програмних продуктів-аналогів

Прикладом систем автоматизованого планування робіт можуть виступати краудсорсингові платформи.

Краудсорсинг [9] – це процес, який забезпечує спільну роботу певної кількості людей над тією чи іншою проблемою, задачею чи проектом. Він набув популярності завдяки мережі Інтернет, оскільки саме в ній легше за все координувати дії між групою виконавців. Важливою особливістю даного процесу є добровільність роботи, а не контрактний найм, що не зовсім стосується розроблюваного проекту. Але сама система забезпечення виконуваності завдань та їх структурованість відповідає вимогам КЗАПР.

Тому, було вирішено детальніше розглянути приклади краудсорсингових платформ. Оскільки їх використання відповідає вирішенню поставлених перед проектом завдань.

Для аналізу програмних продуктів було обрано такі дві платформи, як ClickWorker [10] та CrowdFlower [11]. Вони є досить популярними сьогодні.

ClickWorker – це одна з найбільших платформ для краудсорсингу, яка зосереджена на мікрозавданнях. Робітники можуть проживати будь-де та отримувати

оплату праці через Paypal [12]. Рекламується як «повній сервіс», тобто працюють з клієнтами протягом усього життєвого циклу завдань.

CrowdFlower – це платформа з функціями машинного навчання, де людина виконує задачі, які не вдається автоматизувати.

Основні критерії відмінності представлені наступними пунктами:

1. Механізми стимулювання відрізняються на даних платформах. У ClickWorker працівники отримують грошову винагороду за кожного запрошеного робітника, який заробляє від 10 доларів США. CrowdFlower використовує ігрове та змагальне заохочення. Це реалізовано у вигляді значків та рейтингової таблиці точності роботи. Перші відкривають можливість виконувати більш високооплачувані задачі, а за високу точність виконання завдань пропонуються грошові виплати.

2. Контроль якості платформи ClickWorker включає використання внутрішніх перевірок або задіяння іншого працівника для перевірки та виправлення виконаних завдань, перевірку на плагіат, випадкове вибіркове тестування персоналом платформи тощо. Безпосередній зв’язок між робітниками та запитувачами відсутній. Оплачується тільки виконана робота, а відхилена повинна мати докладне пояснення причини. У CrowdFlower контроль якості базується на золотих тестах (GTQ) [13].

Відмінні особливості платформ більш детально описані в дослідженні [14].

Результати порівняльного аналізу платформ ClickWorker та CrowdFlower представлені в таблиці 1.1

Таблиця 1.1 – Порівняльна характеристика платформ ClickWorker та CrowdFlower

Критерії	ClickWorker	CrowdFlower
Мультиплатформовість	+	–
Анонімність працівників	+	–
Визначення кваліфікації	+	+
Обмеження списку завдань	+	–
Механізми стимулювання	+	+

Продовження табл. 1.1.

Наявність онлайн-форуму	+	–
Контроль якості	+	+
RESTful API	+	+
Підтримка форматів	XML, JSON	RSS, Atom, XML, JSON
Підтримка спеціалізованих і складних завдань.	+	+

За результатами проведеного аналізу існуючих аналогів усі зазначені web-додатки мають недоліки. Наприклад, ClickWorker працює з меншою кількістю форматів, механізми стимулювання не направлені на виконання завдань, а лише на розповсюдження платформи. CrowdFlower не має мультиплатформовості та відсутній метод спілкування всередині платформи. Через описані недоліки необхідно створити власний web-додаток, який буде відповідати поставленим цілям завдання та подолає виявлені недоліки.

1.3 Постановка задачі

Метою дослідження є розробка механізму обміну та захисту даних при їх передаванні між співробітниками проектної організації та зберіганні у КЗАПР.

Основні вимоги до розроблюваного продукту є такими:

– організувати передачу даних від проектного менеджера до робітника;

– забезпечити захист приватної інформації замовника та деталі проекту від виконавця завдання;

– створити модуль на сервері для реалізації інших вимог.

Для досягнення поставленої мети потрібно виконати наступні задачі:

– визначити актуальність роботи, дослідити предметну область та існуючі аналоги;

– спроектувати структуру модулю на сервері;

– проаналізувати та обрати технології та мову розробки;

– реалізувати структуру модулю на сервері;

– розробити функціонал проектного рішення;

– провести тестування розробленого модулю.

Повне технічне завдання даного проекту наведене в Додатку А.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

2.1 Моделювання web-додатку супроводження програмного серверу КЗАПР в IDEF0 та IDEF1

Для повноцінного розуміння бізнес-процесів тієї чи іншої системи необхідно провести її функціональне моделювання. Для цього треба скористатися відповідними нотаціями IDEF0 [15]. Це методологія, що відображає структуру та функції будь-якої системи, а також потоки інформації та матеріальних об’єктів, які пов’язують ці функції. Представлена в графічній нотації. ЇЇ основними елементами є процес та стрілки. Останні можуть бути наступних чотирьох видів:

- Вхідні – ставлять певне завдання;
- Вихідні – виводять результат діяльності;
- Управління – механізми управління;
- Механізми – використовується для того, щоб зробити необхідну роботу.

У діаграмі IDEF0 фігурують такі дані:

- Вхідні – запит;
- Вихідні – виконана задача;
- Управління – технічне завдання, протокол передачі даних, протокол оцінювання виконання;
- Механізми – база даних, сервер, web-додаток.

Результат функціонального моделювання у вигляді IDEF0 для даної розробки представлений на рисунку 2.1.

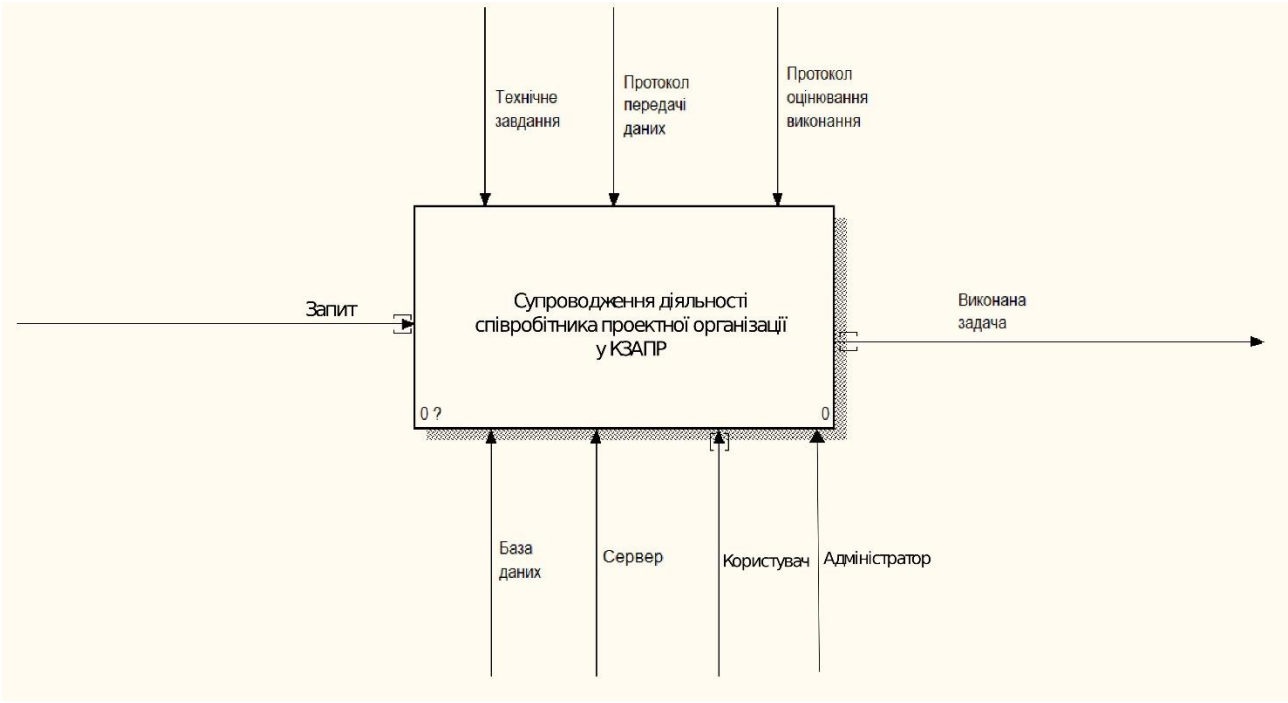


Рисунок 2.1 – IDEF0

Для більш деталізованої демонстрації внутрішніх процесів було виконано декомпозицію діаграми IDEF0 (рис. 2.2). Для зручного представлення інформації було розроблено таблицю підпроцесів (табл. 2.1).

Декомпозиція діаграми роботи модулю супроводження діяльності співробітника проектної організації в КЗАПР представлена такими підпроцесами:

- створення задачі;
- надсилання задачі виконавцю;
- виконання задачі;
- оцінювання виконання задачі.

Таблиця 2.1 – Вхідні та вихідні дані для діаграми оформлення замовлення

Стрілка/ Підпроцес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Створення задачі	Запит на розробку	Технічне завдання	Web-додаток, база даних, сервер	Створена задача
Надсилання задачі виконавцю	Створена задача	Протокол передачі даних	Web-додаток, сервер	Отримана задача

Продовження таблиці 2.1

Виконання задачі	Отримана задача	Технічне завдання	Web-додаток, база даних	Передана задача
Оцінювання виконання задачі	Передана задача	Технічне завдання, протокол оцінювання виконання	Web-додаток, база даних, сервер	Виконана задача

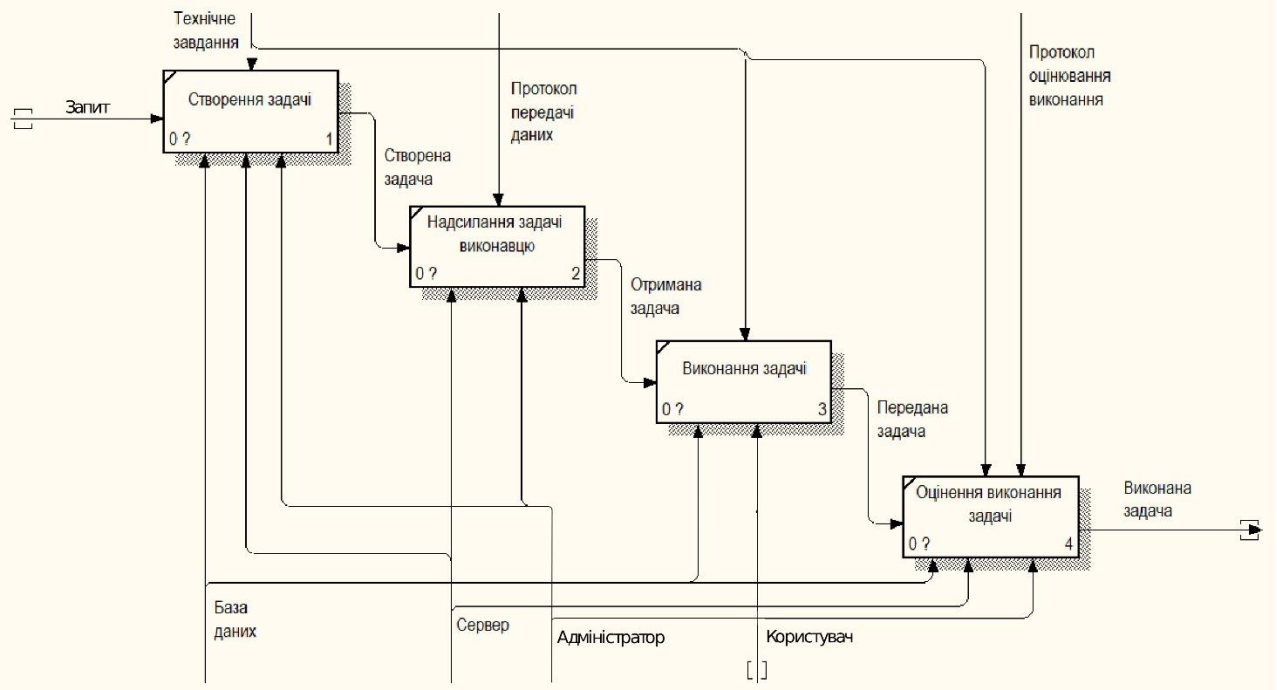


Рисунок 2.2 – Декомпозиція IDEF0

2.2 Структурно-функціональне моделювання

Аналіз предметної області та проектування є першими етапами в життєвому циклі створення програмного рішення. Одним із результатів цієї фази є діаграма варіантів використання (use-case) [16]. Вона описує основні групи об’єктів та суб’єктів та їх взаємодію з системою.

Головними елементами use-case діаграми є актори, функції, які ними використовуються, а також зв’язки між елементами. Останні поділяються на чотири типи: асоціації, включення, залежності та розширення.

Акторами діаграми в рамках даного проекту виступають робітник, проектний менеджер та база даних (БД). Варіанти використання проектного рішення, яке розробляється, є такими:

- Створення завдання;
- Редагування завдання;
- Видалення завдання;
- Відправка завдання;
- Перегляд доступної інформації про завдання;
- Виконання завдання;
- Відправка виконаного завдання;
- Перегляд виконаного завдання;
- Внесення відгуку про виконання.

Діаграма варіантів використання представлена на рисунку 2.3.

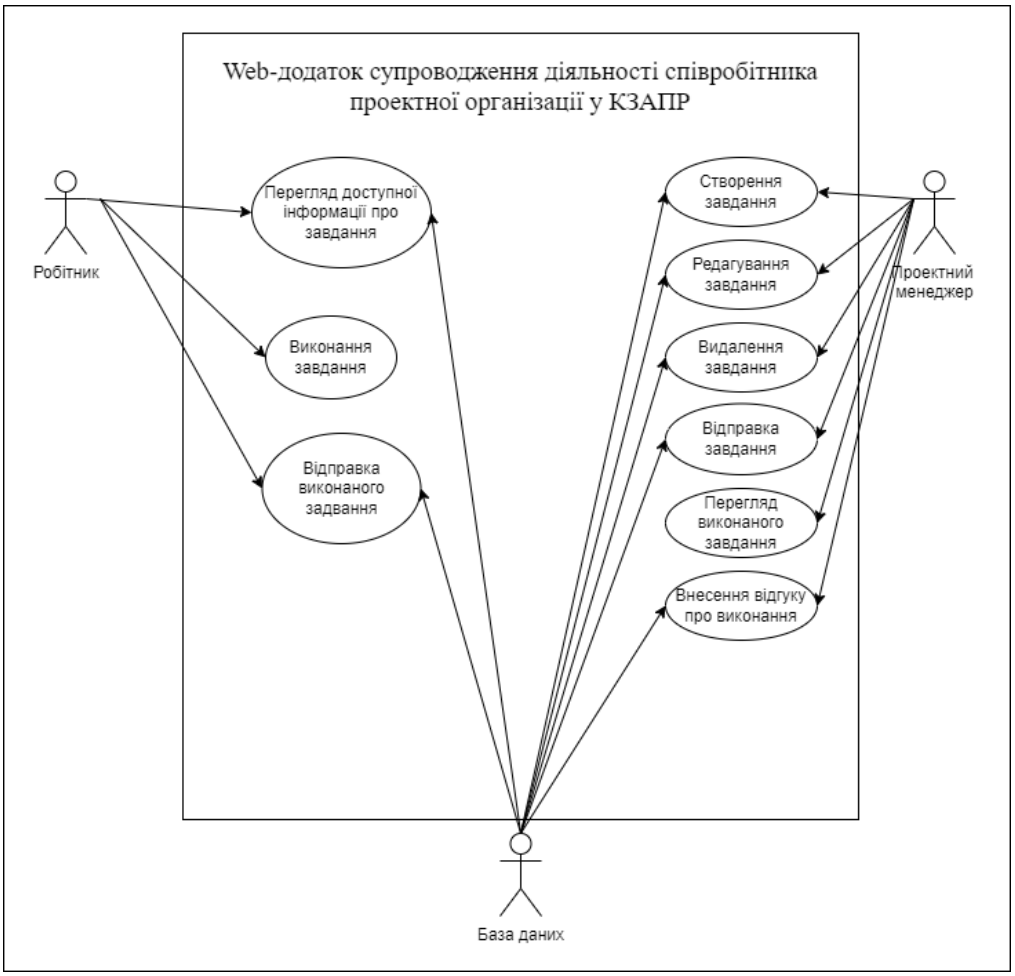


Рисунок 2.3 – Діаграма варіантів використання

2.3 Моделювання даних

На рисунку 2.4 представлена схема даних БД [17] web-додатку супроводження діяльності співробітника проектної організації у КЗАПР.

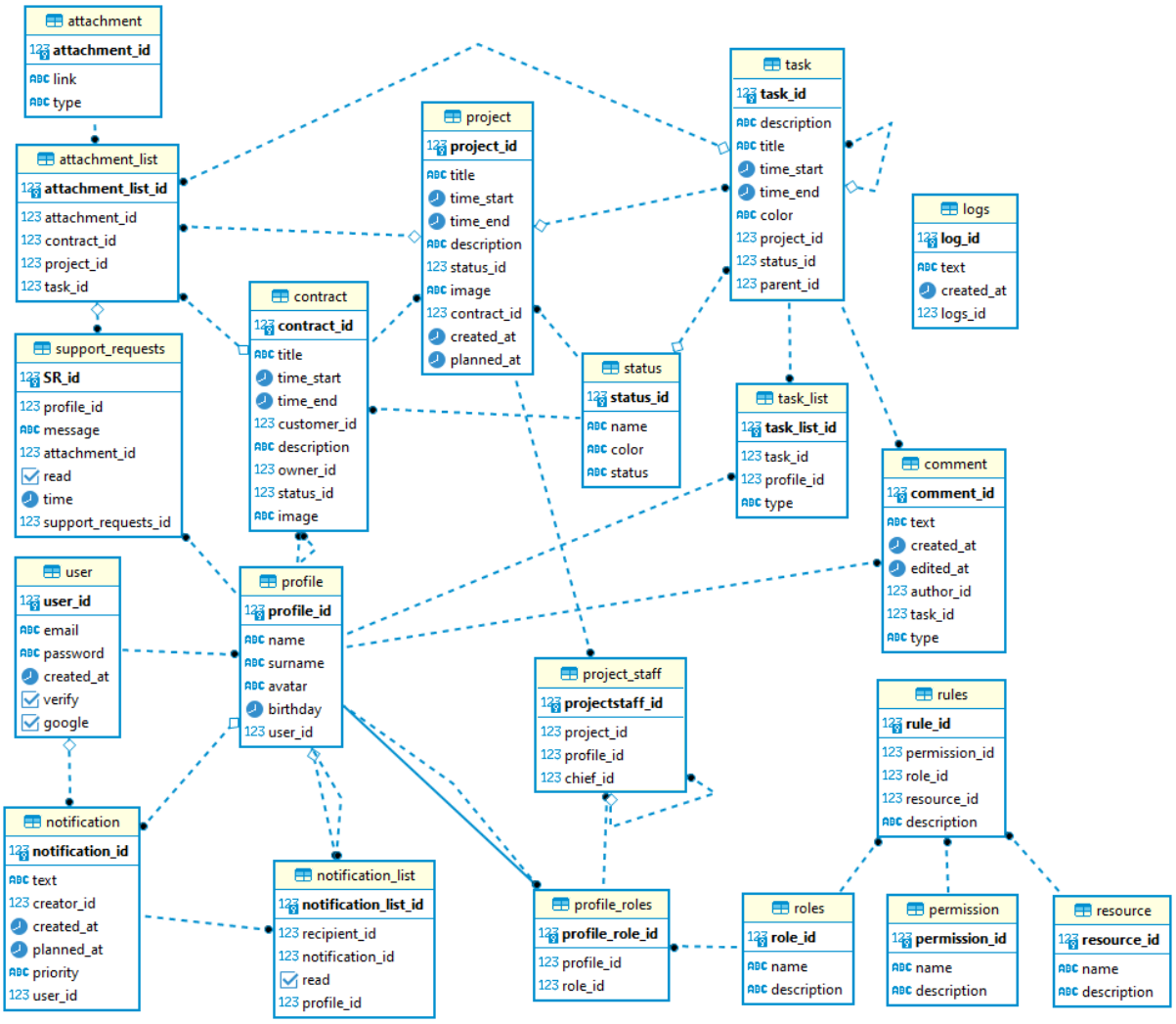


Рисунок 2.4 – Логічна модель бази даних

3 РОЗРОБКА WEB-ДОДАТКУ СУПРОВОДЖЕННЯ ДІЯЛЬНОСТІ СПІВРОБІТНИКА ПРОЕКТНОЇ ОРГАНІЗАЦІЇ У КЗАПР

3.1 Архітектура web-додатку

Перед розробкою web-додатку необхідно провести проектування його архітектури. Для цього необхідно розробити діаграму дизайну високого рівня (HLD) [18]. Вона описує взаємозв'язок між різними компонентами та функціями створюваного web-додатку та визначає логіку для кожного його модуля. У свою чергу, діаграма надає краще розуміння потоку програмного продукту з функціями та дизайном бази даних.

Архітектура web-додатку супроводження діяльності співробітника проектної організації у КЗАПР складається з таких елементів:

- блок авторизації, визначає права користувача;
- об'єкти передачі даних, отримують або відправляють JSON об'єкт;
- контролер, відповідає за обробку запит;
- модель, об'єкти, що формуються на основі записів таблиць бази даних;
- репозиторій, відправляє запити до бази даних;
- базу даних, яка забезпечує модель даними;
- файлове сховище, зберігає логи та файли.

Клієнт надсилає запит до програми у вигляді JSON об'єкту. При цьому йому необхідно авторизуватися для надання прав доступу. Запит потрапляє до контролера, де він обробляється, щоб сформувати запис у вигляді моделі, який надходить до репозіторію, звідки у вигляді SQL [19] request надсилається до бази даних. Після цього контролер формує та надсилає відповідь користувачу. Діаграма високого рівня створюваного додатка представлена на рисунку 3.1

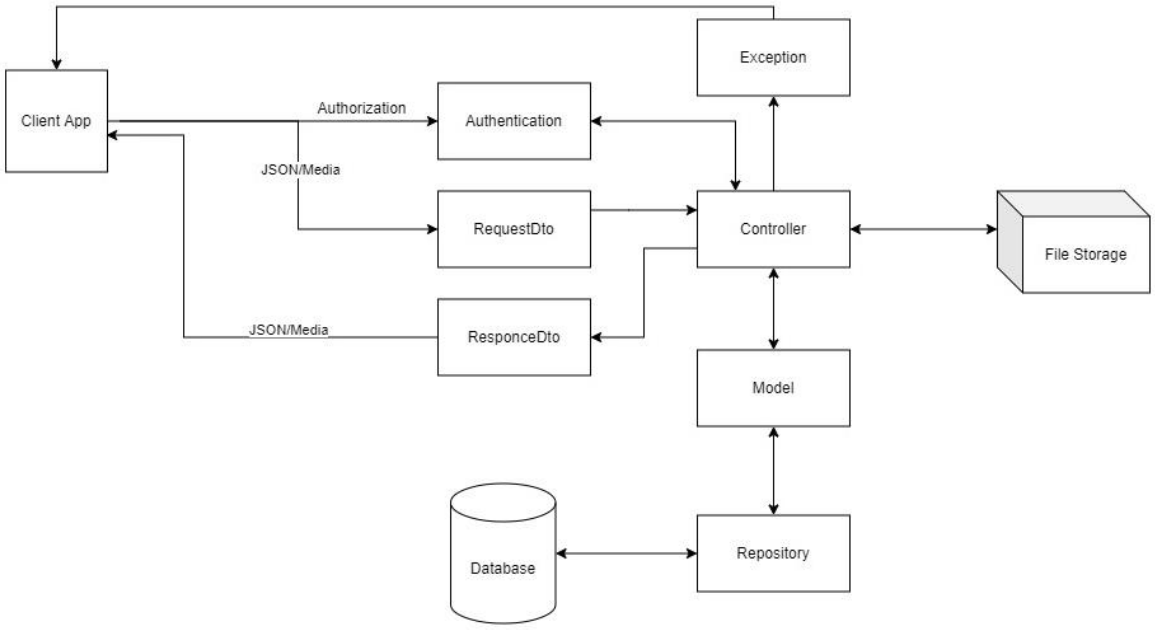


Рисунок 3.1 – Діаграма високого рівня

3.2 Програмна реалізація web-додатку

При розробці серверної частини web-додатку було розроблено декілька модулей. У даному досліджені були розроблені деякі з них, а саме:

- авторизація;
- об’єкти передачі даних;
- контролери;
- репозиторії.

У додатку існує дві ролі допуску до інформації: співробітник організації та адміністратор. У першого обмежений доступ до таких запитів, як: профілі, контракти, користувачі, додавання або видалення файлів, що відносяться до контрактів, додавання або редагування завдань. Адміністратор може використовувати будь-які запити. На рисунку 3.2 продемонстрований запит на отримання інформації до контрактів співробітником, а на рисунку 3.3 представлено той самий запит, але для адміністратора.

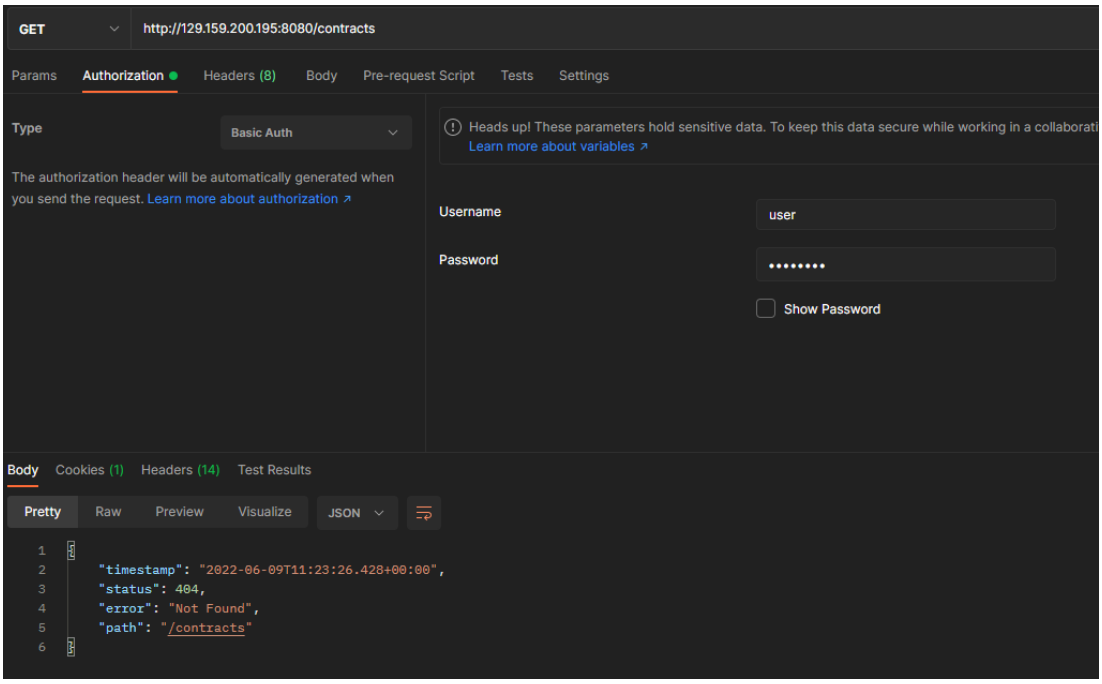


Рисунок 3.2 – Результат запиту на отримання списку контрактів користувачем, авторизованим як співробітник

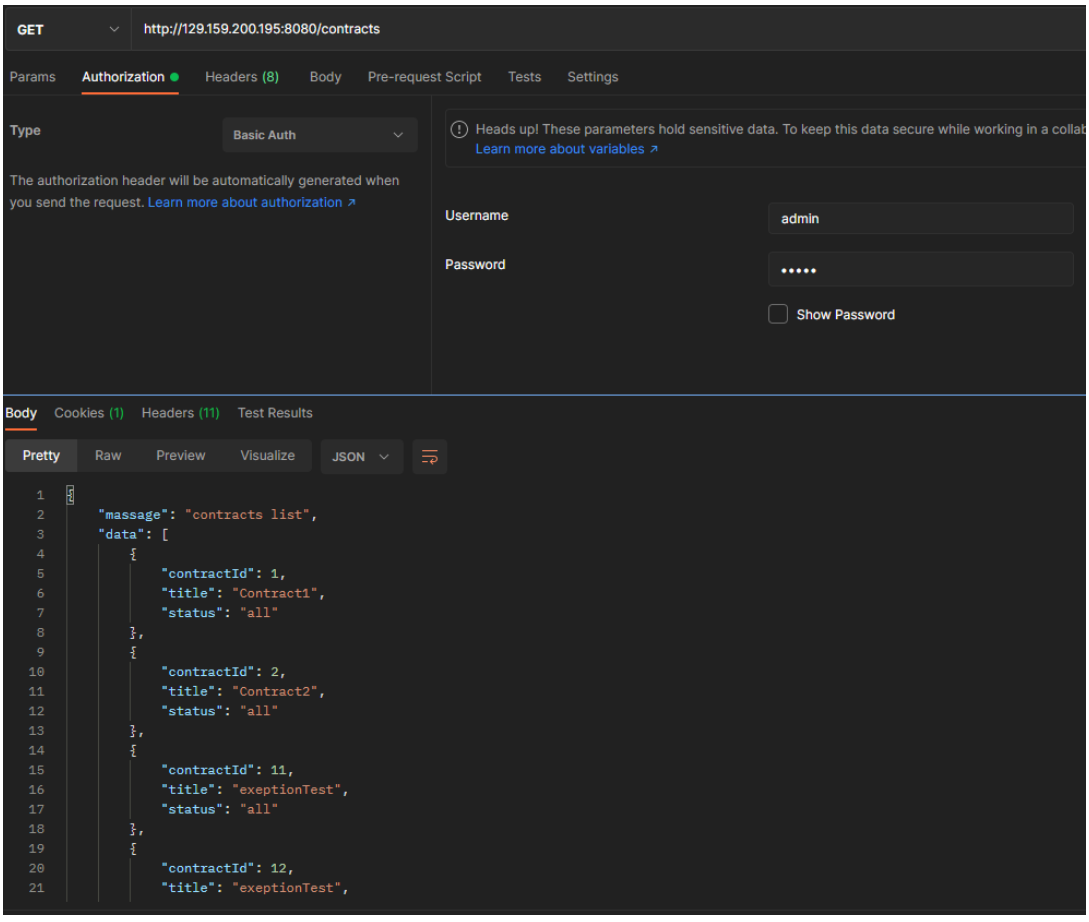


Рисунок 3.3 – Результат запиту на отримання списку контрактів користувачем, авторизованим як адміністратор

Є два типи об’єктів передачі даних – це запит (request) та відповідь (response). Перший приймає інформацію у вигляді JSON із web-частини додатку та використовується для обробки в контролері. Ці типи передачі даних використовуються в запитах на додавання чи оновлення записів в базі даних (рис. 3.4).

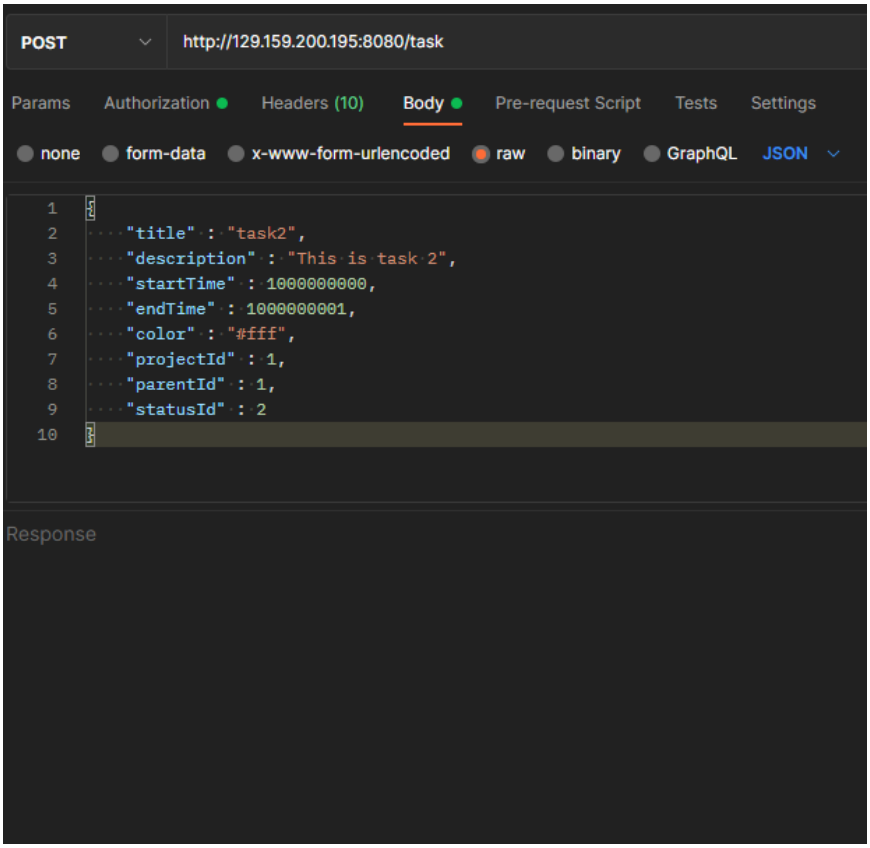


Рисунок 3.4 – Відправлення Post запиту на додавання нового завдання

Відповідь формується в контролері на відправляється до web-частини додатку. Хоча можна повертати тільки модель, але якщо необхідно більше корисної інформації, наприклад, повідомлення про успішність запиту, то буде доцільно використовувати об’єкт передачі даних – відповідь (рис. 3.5).

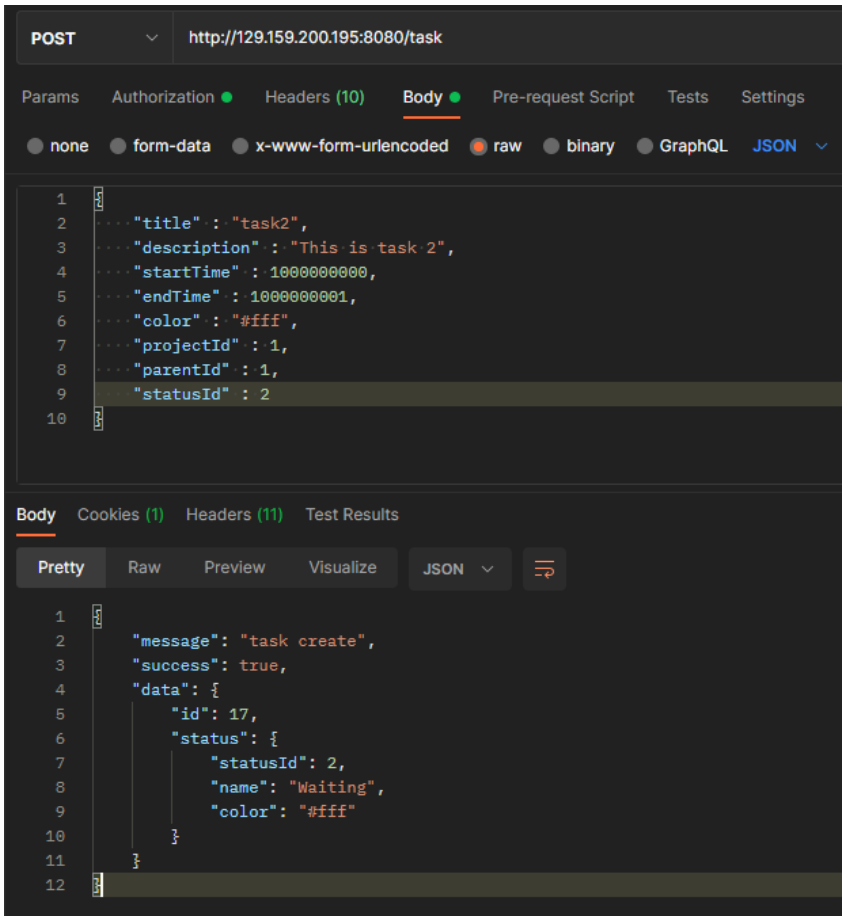
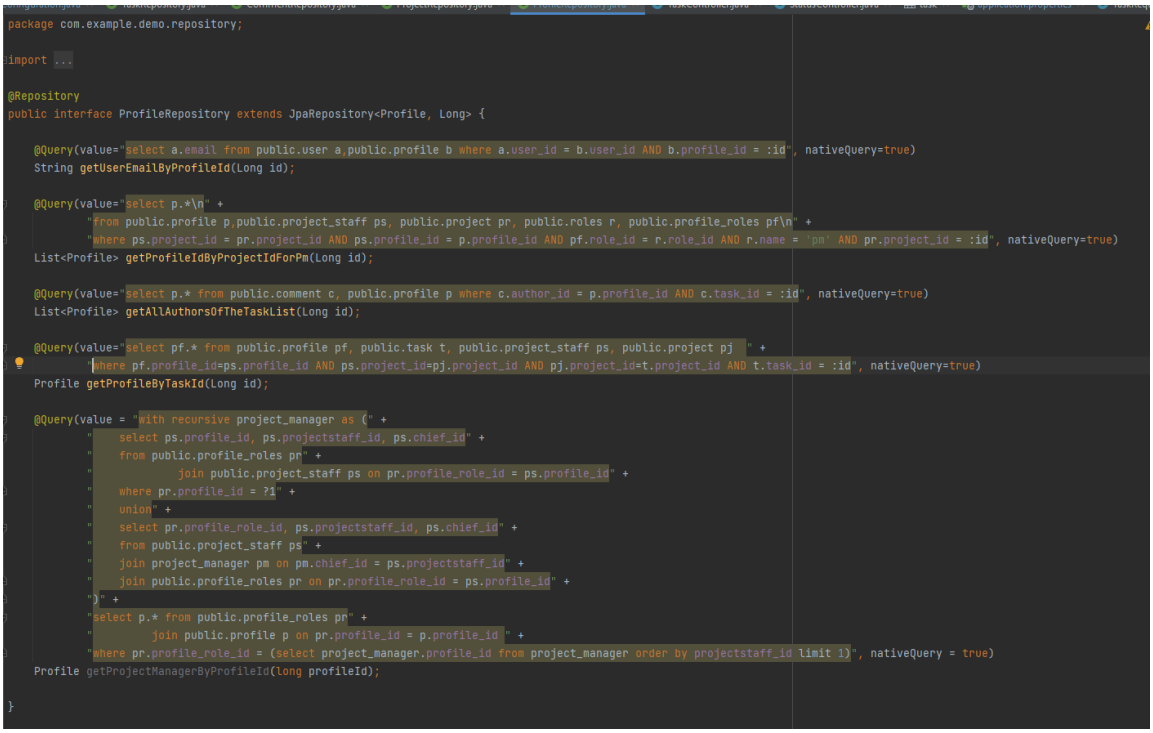


Рисунок 3.5 – Результат відправлення запиту на додавання нового завдання

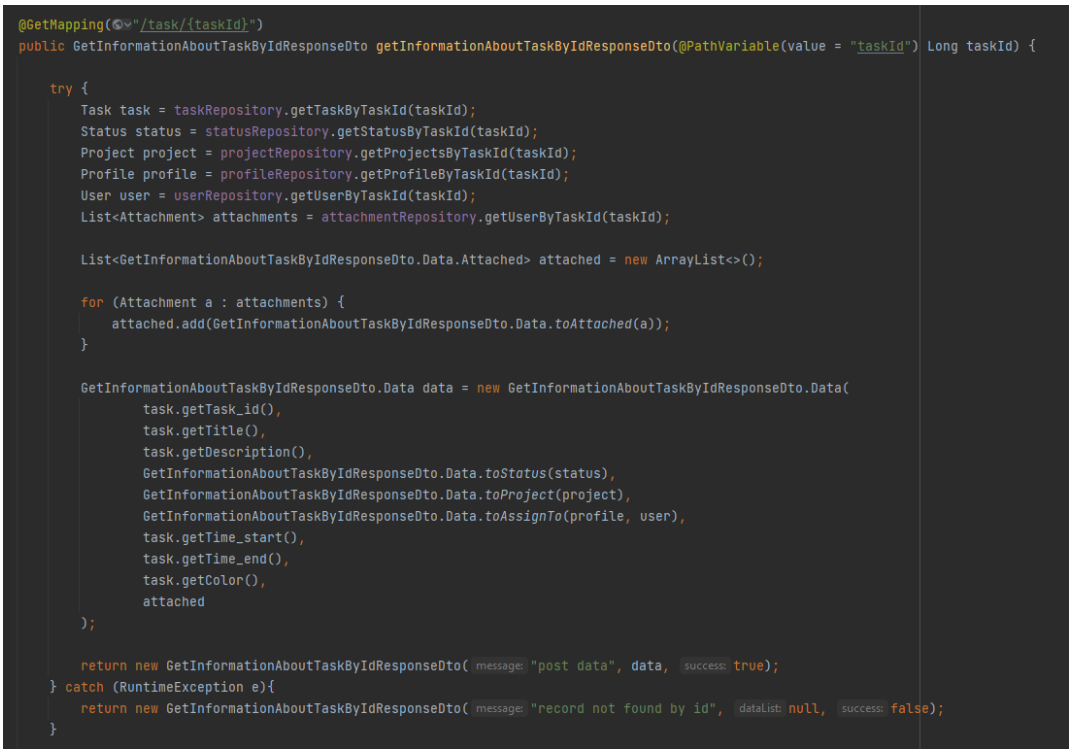
Для взаємодії контролера з базою даних використовуються репозиторії. За допомогою них формується запит до бази даних. Якщо необхідний стандартний request, наприклад, як знайти один запис таблиці по ключу, вивести всі записи таблиці, створити новий запис, оновити або видалити існуючий тощо, то використовується бібліотека Hibernate [20]. Вона має автоматично створені функції для формування таких запитів. Якщо необхідний більш складний request, то він прописується саме в репозиторії (рис. 3.6).



```
package com.example.demo.repository;

import ...

@Repository
public interface ProfileRepository extends JpaRepository<Profile, Long> {

    @Query(value="select a.email from public.user a,public.profile b where a.user_id = b.user_id AND b.profile_id = :id", nativeQuery=true)
    String getUserEmailByProfileId(Long id);

    @Query(value="select p.*\n" +
            "from public.profile p,public.project_staff ps, public.project pr, public.roles r, public.profile_roles pf\n" +
            "where ps.project_id = pr.project_id AND ps.profile_id = p.profile_id AND pf.role_id = r.role_id AND r.name = 'pm' AND pr.project_id = :id", nativeQuery=true)
    List<Profile> getProfileIdByProjectIdForPm(Long id);

    @Query(value="select p.* from public.comment c, public.profile p where c.author_id = p.profile_id AND c.task_id = :id", nativeQuery=true)
    List<Profile> getAllAuthorsOfTheTaskList(Long id);

    @Query(value="select pf.* from public.profile pf, public.task t, public.project_staff ps, public.project pj  " +
            "where pf.profile_id=ps.profile_id AND ps.project_id=pj.project_id AND pj.project_id=t.project_id AND t.task_id = :id", nativeQuery=true)
    Profile getProfileByTaskId(Long id);

    @Query(value = "with recursive project_manager as (" +
            "    select ps.profile_id, ps.projectstaff_id, ps.chief_id" +
            "    from public.profile_roles pr" +
            "             join public.project_staff ps on pr.profile_role_id = ps.profile_id" +
            "    where pr.profile_id = ?1" +
            "    union" +
            "    select pr.profile_role_id, ps.projectstaff_id, ps.chief_id" +
            "    from public.project_staff ps" +
            "    join project_manager pm on pm.chief_id = ps.projectstaff_id" +
            "    join public.profile_roles pr on pr.profile_role_id = ps.profile_id" +
            ")" +
            "select p.* from public.profile_roles pr" +
            "         join public.profile p on pr.profile_id = p.profile_id " +
            "where pr.profile_role_id = (select project_manager.profile_id from project_manager order by projectstaff_id limit 1)", nativeQuery = true)
    Profile getProjectManagerByProfileId(long profileId);

}
```

Рисунок 3.6 – Репозиторій із прописаними складними запитами

У контролері прописується логіка обробки запиту користувача, відправлення його до бази даних та формування відповіді для користувача (рис. 3.7).



```
@GetMapping("/task/{taskId}")
public GetInformationAboutTaskByIdResponseDto getInformationAboutTaskByIdResponseDto(@PathVariable(value = "taskId") Long taskId) {

    try {
        Task task = taskRepository.getTaskByTaskId(taskId);
        Status status = statusRepository.getStatusByTaskId(taskId);
        Project project = projectRepository.getProjectsByTaskId(taskId);
        Profile profile = profileRepository.getProfileByTaskId(taskId);
        User user = userRepository.getUserByTaskId(taskId);
        List<Attachment> attachments = attachmentRepository.getUserByTaskId(taskId);

        List<GetInformationAboutTaskByIdResponseDto.Data.Attached> attached = new ArrayList<>();

        for (Attachment a : attachments) {
            attached.add(GetInformationAboutTaskByIdResponseDto.Data.toAttached(a));
        }

        GetInformationAboutTaskByIdResponseDto.Data data = new GetInformationAboutTaskByIdResponseDto.Data(
                task.getTask_id(),
                task.getTitle(),
                task.getDescription(),
                GetInformationAboutTaskByIdResponseDto.Data.toStatus(status),
                GetInformationAboutTaskByIdResponseDto.Data.toProject(project),
                GetInformationAboutTaskByIdResponseDto.Data.toAssignTo(profile, user),
                task.getTime_start(),
                task.getTime_end(),
                task.getColor(),
                attached
        );

        return new GetInformationAboutTaskByIdResponseDto( message: "post data", data, success: true);
    } catch (RuntimeException e){
        return new GetInformationAboutTaskByIdResponseDto( message: "record not found by id", dataList: null, success: false);
    }
```

Рисунок 3.7 – Метод контролера, опис завдання після пошуку по ключу

3.3 Демонстрація роботи web-додатку

У серверній частині web-додатку існують такі методи взаємодії запитів [21] з базою даних:

- отримання даних (Get);
- додавання даних (Post);
- оновлення даних (Put);
- видалення даних (Delete).

Метод Get передає дані тільки через URL-адресу [21], тому через нього можна отримати список усіх записів (рис. 3.8) або знайти конкретний запис (рис. 3.9).

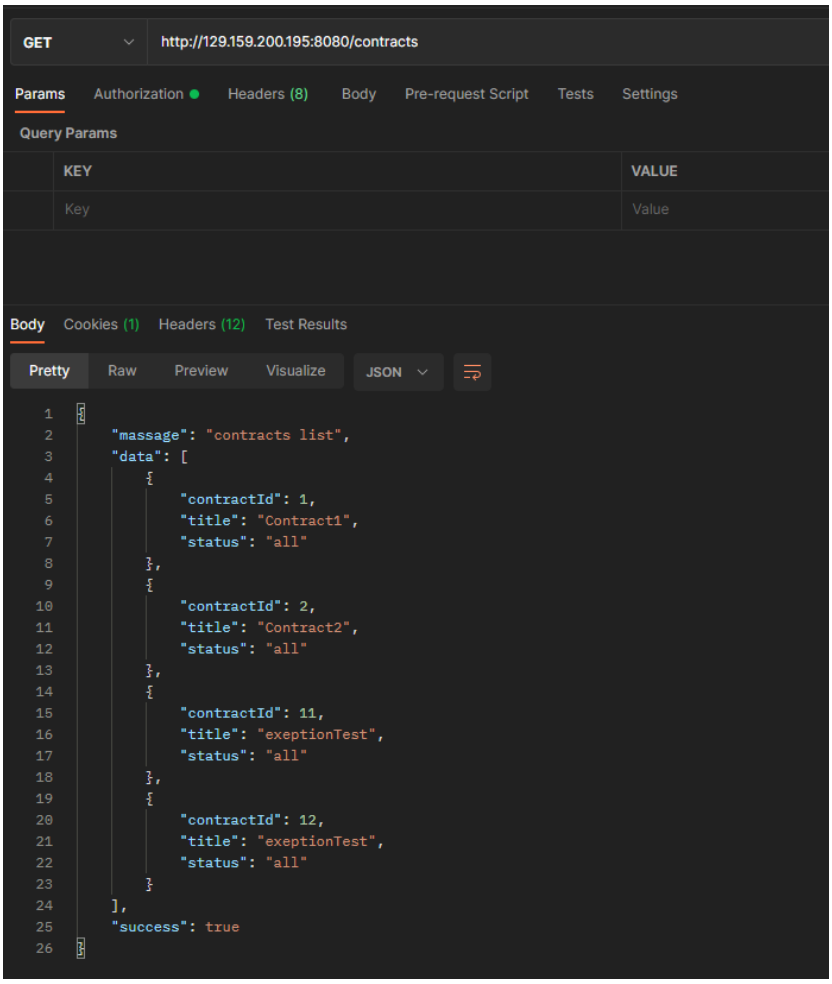


Рисунок 3.8 – Список контрактів

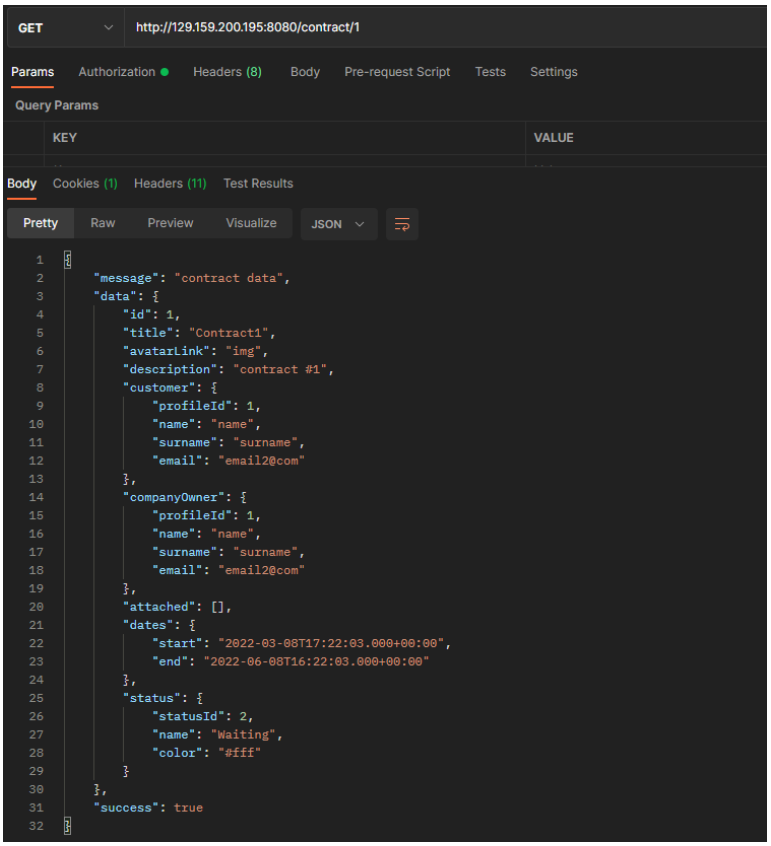


Рисунок 3.9 – Опис контракту

Метод Post містить в собі дані, за допомогою яких створюється новий запис таблиці в базі даних (рис. 3.10). У даному додатку дані передаються в форматі JSON.

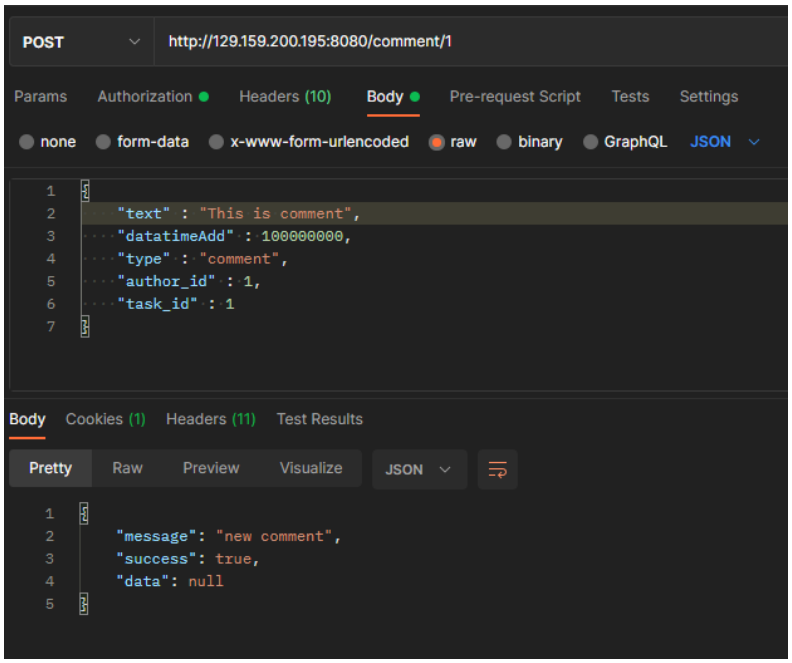


Рисунок 3.10 – Створення коментарю

Метод Put, як і Post містить дані, але вони використовуються для оновлення існуючого запису, тому в URL-адресі буде міститися значення унікального поля, здебільшого ключа, через яке повинен знаходитися запис для апдейту. Приклад продемонстровано на рисунку 3.11.

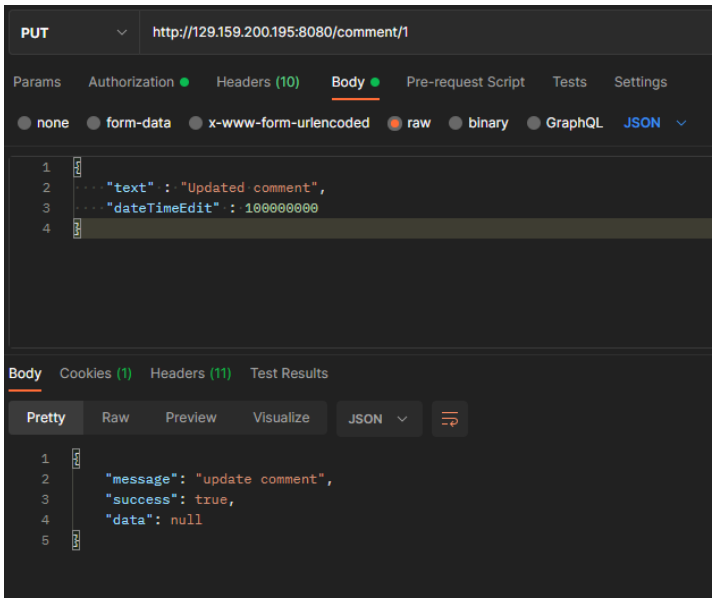


Рисунок 3.11 – Оновлення коментарю

Метод Delete не має тіла. В URL-адресі міститься ключ, по якому буде знайдено та видалено запис (рис. 3.12).

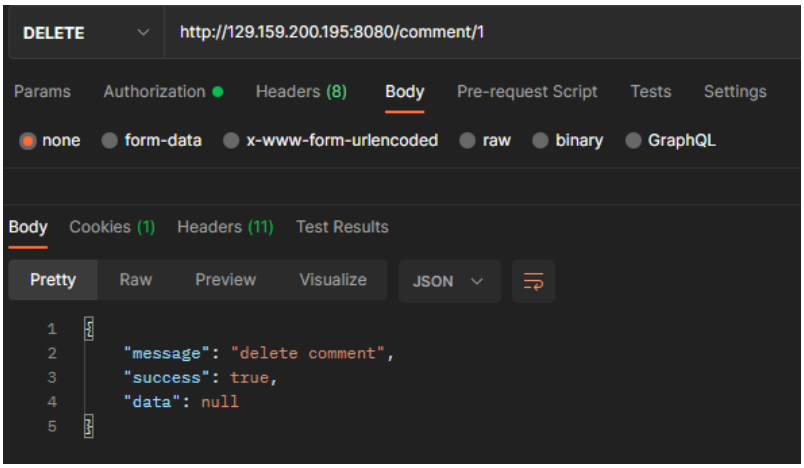


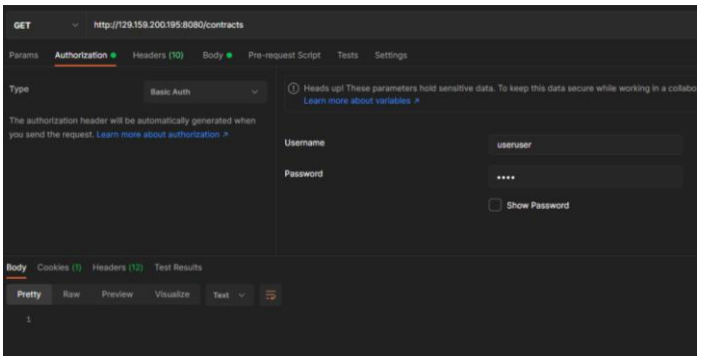
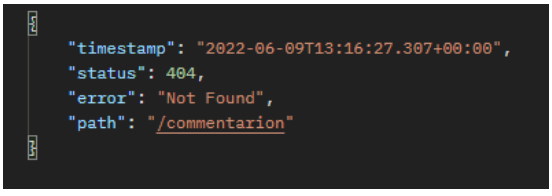
Рисунок 3.12 – Видалення коментарю

3.4 Тестування web-додатку

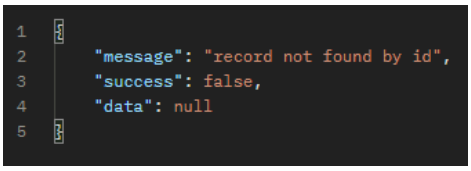
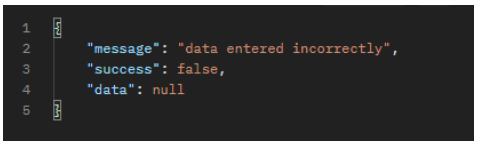
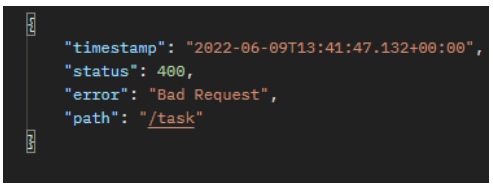
Для створення якісного програмного продукту, необхідно протестувати розроблену серверну частину web-додатку. Її функціональне тестування складається з перевірки реакції на коректність вхідних даних. Даний етап є заключним.

Було проведено тестування за допомогою системи покриття коду (Code coverage system) [22]. Це одна з форм методу «білої скриньки». Вона знаходить області запущеного програмного продукту, які не виконуються набором тестових випадків. Результати даної перевірки продемонстровано в таблиці 3.1.

Таблиця 3.1 – Тестування форм, які розміщені на клієнтській частині розробленого web-додатку

№	Назва	Очікуваний результат	Фактичний результат	0/1
1	Відсутність або некоректність введення імені та паролю акаунту	Неможливість виконання запиту		1
2	Введення некоректного запиту в URL-адресі	Повідомлення про відсутність даного запиту		1

Продовження таблиці 3.1

3	Введення неіснуючого ключа в URL-адресі	Повідомлення про відсутність даного запису в таблиці		1
4	Введення некоректної назви поля об’єкту в тіло запиту	Повідомлення про некоректно введені дані		1
5	Введення некоректного типу даних в тіло	Повідомлення про неправильно введені дані		1

За результатами тестування можна зробити висновок, що програмний продукт працює коректно, оскільки жодних багів виявлень не було.

ВИСНОВОК

Під час виконання кваліфікаційної роботи бакалавра було проведено дослідження двох методів створення web-сервісів: REST та SOAP, проаналізовані їх переваги та недоліки. Звертаючи увагу на специфіку проектного завдання, прийнято рішення обрати REST. Також проведено аналіз програмних продуктів-аналогів та визначено особливості платформ ClickWorker та CrowdFlower. Було сформульовано мету та задачі даної роботи, а також основні вимогами до розробки.

При проектуванні web-додатку була розроблена діаграма варіантів використання та виконано моделювання роботи розроблюваного програмного продукту, застосовуючи методологію IDEF0. Проведено декомпозиці даної діаграми. Спроектована схема даниї баз даних проекту, у котрій були визначені сутності та атрибути.

Розроблена архітектура та виконана програмна реалізація web-додатку. Проведене тестування програмного продукту.

У процесі виконання були сформульовані вимоги до web-додатку, які зазначені в Додатку А.

Також було сформовано планування робіт, досліджені ризики, які можуть виникнути під час розробки web-додатку, та реагування на них (Додаток Б).

Результати роботи були апробовані на науково-практичній конференції ІМА-2022 у Сумському державному університеті (Додаток В).

Лістинг основних типів класів зазначено в Додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. Shelekhov, V. Antypenko, V, Nenia. Web application to support the activities of employee of the project organization in KZAPR. Інформатика, математика, автоматика IMA :: 2022 : Матеріали та програма міжнародної наукової конференції молодих учених – Суми – Нур-Султан, 18–22 квітня 2022 року – Суми: Сумський державний університет , 2022 – С. 111.

2. WEB-СЕРВІСИ. URL: https://stud.com.ua/97613/informatika/servisi (дата звернення: 23.05.2022).

3. Робота із XML. URL: https://helpx.adobe.com/ua/incopy/using/xml.html (дата звернення: 23.05.2022).

4. Функція «JSON» у Power Apps. microsoft. URL: https://docs.microsoft.com/uk-ua/power-apps/maker/canvas-apps/functions/function-json (дата звернення: 23.05.2022).

5. Anshu Soni, Virender Ranga. API Features Individualizing of Web Services: REST and SOAP. International Journal of Innovative Technology and Exploring Engineering. 2019.

6. Основи REST і RESTful API Development. URL: https://ua.phhsnews.com/articles/coding/the-basics-of-rest-and-restful-api-development.html (дата звернення: 23.05.2022).

7. Що таке SOAP?. URL: https://uk.education-wiki.com/9679948-what-is-soap (дата звернення: 23.05.2022).

8. HTTP: Протокол, який повинен розуміти кожний веб-розробник. URL: https://code.tutsplus.com/uk/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177 (дата звернення: 23.05.2022).

9. Краудсорсинг: як спільними силами втілити ідею. URL: https://inspired.com.ua/ideas/crowdsourcing/ (дата звернення: 23.05.2022).

10. ClickWorker. URL: https://www.clickworker.com (дата звернення: 23.05.2022).

11. CrowdFlower. URL: https://visit.figure-eight.com/People-Powered-Data-Enrichment_T (дата звернення: 23.05.2022).

12. PayPal. URL: https://www.paypal.com/nl/home (дата звернення: 23.05.2022).

13. Validity and reliability of naturalistic driving scene categorization Judgments from crowdsourcing / Christopher D.D Cabralla et al. Accident Analysis and Prevention. 2018. P. 25–33.

14. Donna Vakharia, Matthew Lease. Beyond Mechanical Turk: An Analysis of Paid Crowd Work Platforms. Department of Computer Science and School of Information. 2015.

15. IDEF0 діаграма: приклади і правила побудови. URL: https://codoschool.ru/uk/services/idef0-diagramma-primery-i-pravila-postroeniya-metodologii-modelirovaniya.html (дата звернення: 23.05.2022).

16. Use Case Diagrams. URL: https://www.quality-assurance-group.com/use-case-diagrams/ (дата звернення: 23.05.2022).

17. Діаграми потоків даних DFD (Data Flow Diagrams). URL: http://um.co.ua/8/8-2/8-218941.html (дата звернення: 23.05.2022).

18. What is a High Level Design (HLD)?. URL: https://ipwithease.com/what-is-a-high-level-design-hld/ (дата звернення: 01.06.2022).

19. SQL Tutorial. URL: https://www.techonthenet.com/sql/index.php (дата звернення: 01.06.2022).

20. Hibernate. Everything data. URL: https://hibernate.org (дата звернення: 01.06.2022).

21. HTTP: Протокол, який повинен розуміти кожний веб-розробник. URL: https://code.tutsplus.com/uk/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177 (дата звернення: 01.06.2022).

22. Code Coverage Tutorial: Branch, Statement, Decision, FSM. URL: https://www.guru99.com/code-coverage.html (дата звернення: 01.06.2022).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на створення комп’ютерного додатку «Web-додаток супроводження діяльності співробітника проектної організації у КЗАПР»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Антипенко В.П.

Студент групи ІТ-82-0

_____ Шелехов Д.В.

Суми 2022

1 Призначення й мета web-додатку супроводження діяльності співробітника проектної організації у КЗАПР

1.1 Призначення модуля

Модуль призначений для забезпечення безпечної передачі інформації про завдання між адміністратором та виконавцем.

1.2 Мета створення проекту

Метою дослідження є розробка механізму обміну та захисту даних при їх передаванні між співробітниками проектної організації та зберіганні у КЗАПР.

1.3 Цільова аудиторія

Користувачами даного модулю виступають адміністратор та робітник.

2 Вимоги до проекту

2.1 Вимоги до проекту в цілому

2.1.1 Вимоги до структури й функціонування

Модуль повинен працювати на серверній частині розроблюваного комплексу, коректно опрацьовувати запити до бази даних та взаємодіяти з web-частиною.

Кінцевий продукт має бути модулем серверної частини, який буде повноцінно виконувати поставлені задачі.

2.1.2 Вимоги до персоналу

Персонал закладу не повинен мати особливих технічних навичок для роботи з web-додатком і його підтримкою. Єдиною вимогою є наявність навичок користування персональним комп'ютером та web-браузером.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у web-додатку повинна зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Розроблюваний web-додаток супроводження діяльності співробітника проектної організації у КЗАПР, повинен розмежувати користувачів за двома групами: користувач та адміністратор. Адміністратор має необмежений доступ до програмного серверу, а користувач продивлятися тільки доступну йому інформацію та відсилати виконане завдання.

2.2 Вимоги до видів забезпечення

2.2.1 Вимоги до лінгвістичного забезпечення

Інформація, яку буде використовувати проектне рішення має бути англійською мовою.

2.2.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи програмного серверу web-додатку КЗАПР, на віддаленому сервері повинна бути встановлено Java 1.8 та база даних MySQL.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Створення завдання	Адміністратор
UN-02	Редагування завдання	Адміністратор
UN-03	Надсилання завдання	Адміністратор, виконавець
UN-04	Перегляд бази даних	Адміністратор
UN-05	Перегляд інформації по завданню	Адміністратор, виконавець
UN-06	Виконання завдання	Виконавець
UN-07	Створення відгуку, щодо завдання	Адміністратор, виконавець
UN-08	Додавання та редагування контракту	Адміністратор
UN-09	Створення, редагування профілю	Адміністратор

2.3.2 Системні вимоги

Проаналізувавши потреби користувачів та персоналу закладу було визначено наступні вимоги:

- наявність авторизації;
- взаємодія з базою даних;
- механізм передачі даних;
- механізм захисту даних.

3 Склад і зміст робіт зі створення web-додатку супроводження діяльності співробітника проектної організації у КЗАПР

Детальний опис етапів створення web-додатку наведено в таблиці А.1.

Таблиця А.1 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки
1	Аналіз бази даних	3 дні
2	Розробка списку функціональних можливостей	7 днів
3	Розробка функціоналу	20 днів
4	Тестування модулю	7 днів
5	Налаштування взаємодії модуля з системою	5 днів
6	Тестування системи	10 днів
7	Написання супровідної документації	14 днів
	Загальна тривалість робіт	66 днів

4 Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Web-додаток має бути затверджено та розміщено на web-хостингу.

ДОДАТОК Б

Планування робіт

При розробці будь-якого програмного продукту, слід розуміти, що його замовник може бажати уникнення розповсюдження інформації серед сторонніх осіб. Тому, важливою частиною створення таких додатків є максимальний захист даних у ході обміну ними між робітниками організації. Для цього необхідно реалізувати механізм передачі задач. При його використанні працівники будуть отримувати тільки ті дані, які необхідні для, власне, виконання самого завдання.

Метою дослідження є розробка механізму обміну та захисту даних при їх передаванні між співробітниками проектної організації та зберіганні у КЗАПР

Деталізація мети проекту методом SMART. Для конкурентноспроможності проекту, необхідно визначити мету за допомогою методу SMART на концептуальному етапі. Результати деталізації SMART-методом вказані в таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific (конкретна)	Забезпечити супроводження діяльності співробітника проектної організації у КЗАПР
Measurable (вимірювана)	Розроблений модуль передачі та захисту даних між базою даних та інтерфейсом працівника.
Achievable (досяжна, узгоджена)	Для виконання проекту є необхідні знання об’єктно-орієнтованої мови програмування Java, баз даних MySQL. Наявні всі необхідні ресурси та навички - мета досяжна. Є затверджене технічне завдання.
Relevant (реалістична)	Для забезпечення безпечної передачі даних між web-частиною додатку та базою даних.
Time-framed (обмежена в часі)	Необхідно завершити проект до 10 червня 2022 року.

Структура декомпозиції робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) представляє схему, в якій завдання проекту відображають їхнє ставлення один до одного і до проекту в цілому. WBS основана на графічному підході, який допомагає менеджерам проектів передбачити результати, що базуються на різних сценаріях. Процес часто описується як структура відгалуження, що охоплює всі етапи проекту в організованому порядку. Менеджери використовують структуру декомпозиції, щоб структурувати та ділити проекти на легко керовані компоненти. Вони, своєю чергою, поділяються до того часу, поки вони не призначаються конкретному спеціалісту у команді. WBS також може бути представлена у вигляді табличного списку задач та елементів у плані розбивки робіт діаграм Ганта.

На рисунку Б.1 представлена ієрархічна структура робіт Web-додатку супроводження діяльності співробітника проектної організації у КЗАПР.

Планування структури виконавців. OBS (Organization Breakdown Structure) – це тимчасова організаційна структура, створена для підвищення якості управління та взаємодії у проекті шляхом визначення та візуалізації процесів взаємодії як між внутрішніми, і між зовнішніми учасниками проекту. Метою розробки OBS було визначення складу та розподіл обов'язків виконавців робіт, що входять до WBS-структури. Ця структура стосується лише внутрішньої організаційної структури проекту та не торкається відносин проектних груп або учасників з батьківськими організаціями.

На рисунку Б.2 представлена організаційна структура робіт Web-додатку супроводження діяльності співробітника проектної організації у КЗАПР.

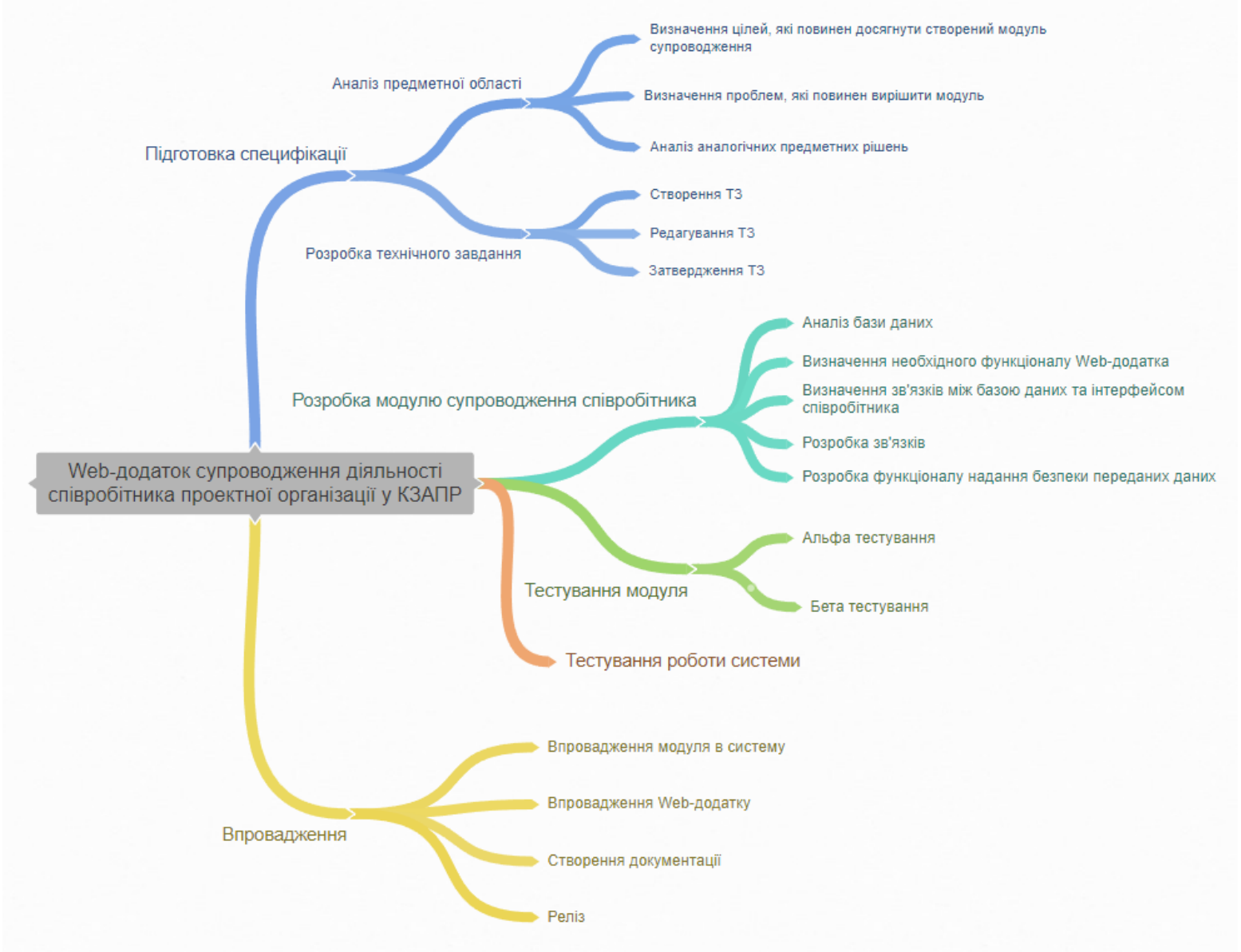


Рисунок Б.1 – WBS-структура робіт проекту

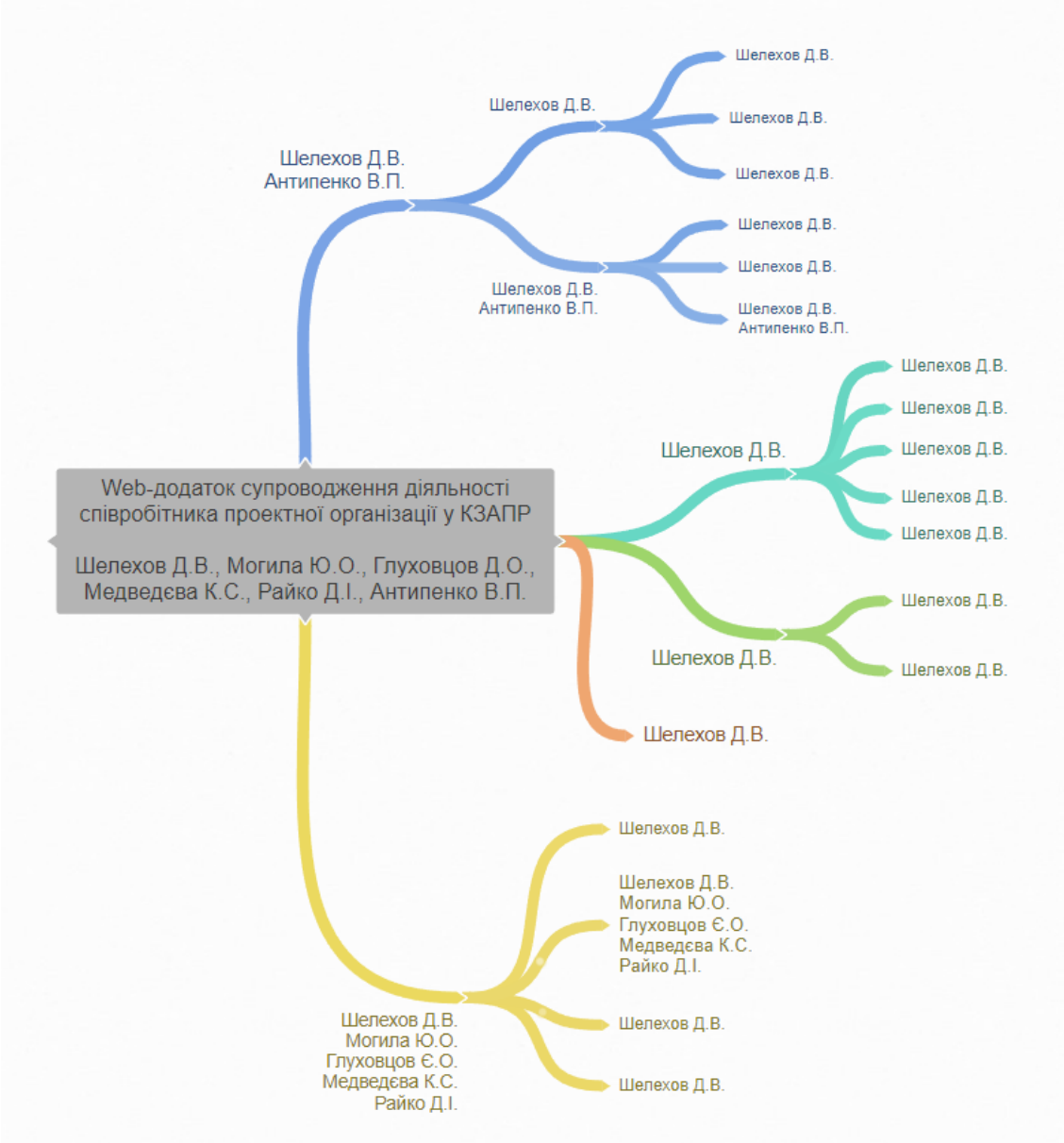
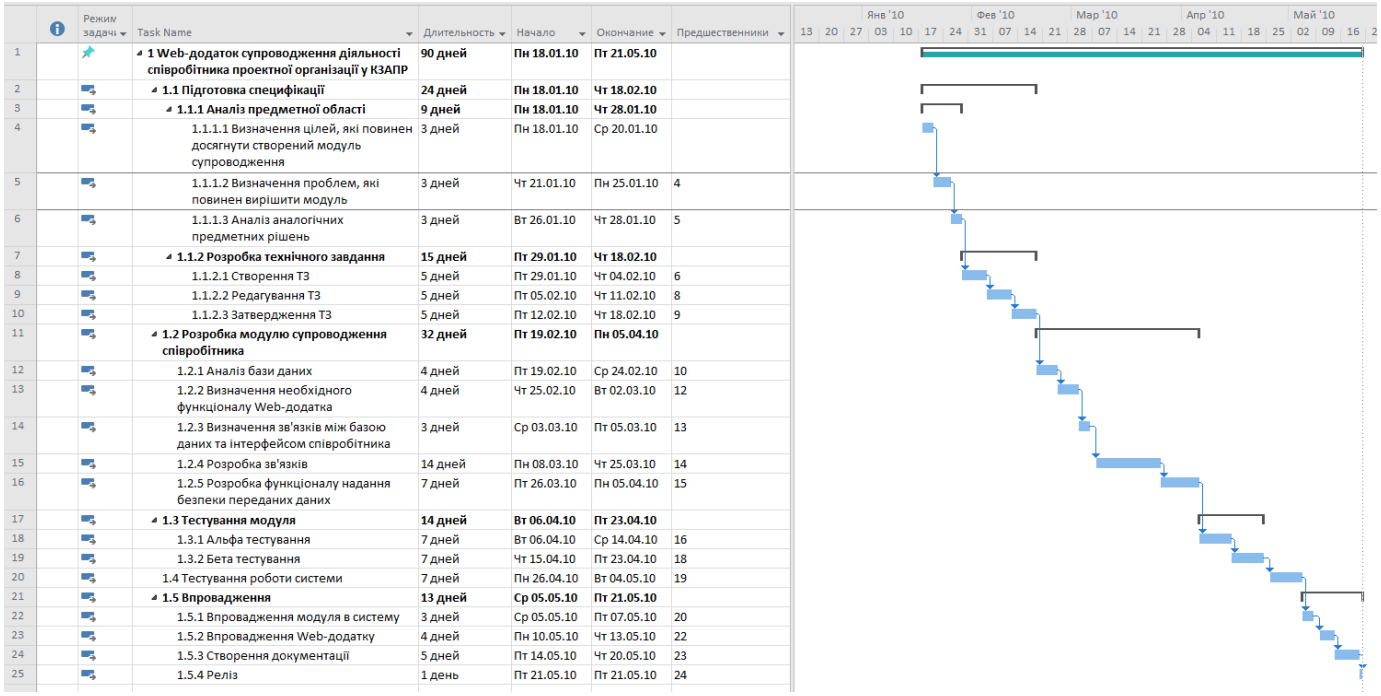


Рисунок Б.2 – OBS-структура робіт проекту

Таблиця Б.2 – Виконавці проекту

Роль	**Ім’я**	**Проектна роль**
Розробник	Шелехов Д.В.	Розробляє модуль супроводження робітника
Розробник	Могила Ю.О.	Впроваджує цілісну систему Web-додатку
Розробник	Глуховцов Є.О.	Впроваджує цілісну систему Web-додатку
Розробник	Медведєва К.С.	Впроваджує цілісну систему Web-додатку
Розробник	Райко Д.І.	Впроваджує цілісну систему Web-додатку
Тестувальник	Шелехов Д.В.	Тестує модуль на наявність некоректних результатів та безпечність
Керівник проекту	Антипенко В.П.	Формує тему проектного завдання
Менеджер проекту	Шелехов Д.В.	Відповідає за розподілення роботи та ресурсів між учасниками, проводить аналіз даних, слідкує за виконанням термінів.

Діаграма Ганта. Це інструмент, що дозволяє візуалізувати та управляти проектами, структурувати їх виконання та бачити загальну картину завдань, як особистих, так і організації. Зазвичай вона складається із двох частин: у лівій частині наведено список завдань, а у правій — тимчасова шкала зі смугами, що зображають роботу. Діаграма Ганта також може включати дати початку та завершення завдань, контрольні точки, залежності між завданнями та виконавцями. Календарний графік проекту представлено на рисунку Б.3.



	Режим задач	Task Name	Длительность	Начало	Окончание	Предшественники
1		**1 Web-додаток супроводження діяльності співробітника проектної організації у КЗАПР**	**90 дней**	**Пн 18.01.10**	**Пт 21.05.10**	
2		**1.1 Підготовка специфікації**	**24 дней**	**Пн 18.01.10**	**Чт 18.02.10**	
3		**1.1.1 Аналіз предметної області**	**9 дней**	**Пн 18.01.10**	**Чт 28.01.10**	
4		1.1.1.1 Визначення цілей, які повинен досягнути створений модуль супроводження	3 дней	Пн 18.01.10	Ср 20.01.10	
5		1.1.1.2 Визначення проблем, які повинен вирішити модуль	3 дней	Чт 21.01.10	Пн 25.01.10	4
6		1.1.1.3 Аналіз аналогічних предметних рішень	3 дней	Вт 26.01.10	Чт 28.01.10	5
7		**1.1.2 Розробка технічного завдання**	**15 дней**	**Пт 29.01.10**	**Чт 18.02.10**	
8		1.1.2.1 Створення ТЗ	5 дней	Пт 29.01.10	Чт 04.02.10	6
9		1.1.2.2 Редагування ТЗ	5 дней	Пт 05.02.10	Чт 11.02.10	8
10		1.1.2.3 Затвердження ТЗ	5 дней	Пт 12.02.10	Чт 18.02.10	9
11		**1.2 Розробка модулю супроводження співробітника**	**32 дней**	**Пт 19.02.10**	**Пн 05.04.10**	
12		1.2.1 Аналіз бази даних	4 дней	Пт 19.02.10	Ср 24.02.10	10
13		1.2.2 Визначення необхідного функціоналу Web-додатка	4 дней	Чт 25.02.10	Вт 02.03.10	12
14		1.2.3 Визначення зв'язків між базою даних та інтерфейсом співробітника	3 дней	Ср 03.03.10	Пт 05.03.10	13
15		1.2.4 Розробка зв'язків	14 дней	Пн 08.03.10	Чт 25.03.10	14
16		1.2.5 Розробка функціоналу надання безпеки переданих даних	7 дней	Пт 26.03.10	Пн 05.04.10	15
17		**1.3 Тестування модуля**	**14 дней**	**Вт 06.04.10**	**Пт 23.04.10**	
18		1.3.1 Альфа тестування	7 дней	Вт 06.04.10	Ср 14.04.10	16
19		1.3.2 Бета тестування	7 дней	Чт 15.04.10	Пт 23.04.10	18
20		1.4 Тестування роботи системи	7 дней	Пн 26.04.10	Вт 04.05.10	19
21		**1.5 Впровадження**	**13 дней**	**Ср 05.05.10**	**Пт 21.05.10**	
22		1.5.1 Впровадження модуля в систему	3 дней	Ср 05.05.10	Пт 07.05.10	20
23		1.5.2 Впровадження Web-додатку	4 дней	Пн 10.05.10	Чт 13.05.10	22
24		1.5.3 Створення документації	5 дней	Пт 14.05.10	Чт 20.05.10	23
25		1.5.4 Реліз	1 день	Пт 21.05.10	Пт 21.05.10	24

Рисунок Б.3 – Календарний графік проекту

Управління ризиками проекту. При проведенні якісної оцінки , необхідно визначити ризики, які можуть бути критичними для проекту. Класифікація ризиків відбувається за такими критеріями, як ймовірність виникнення та вплив ризику на проект. Шкала класифікації ризиків за впливом та ймовірністю виникнення вказана в таблиці Б.3.

Таблиця Б.3 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу.

Оцінка	**Ймовірність виникнення**	**Вплив ризику**	**Тип ризику**
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

На кожний ризик повинен бути розроблений план по усуненню або мінімізації негативного вплину ризику на проект. Відбувається оцінка наслідків ризику та ефективності розробки проекту за показниками таблиці Б.3. Будується

матриця ймовірності та впливу згідно проекту, зеленим кольором позначені прийнятні ризик, жовтим – виправдані, червоним показані недопустимі. У результаті розробки плану реагування отримано матрицю показану в таблиці Б.4.

Таблиця Б.4 – Матриця ймовірності та впливу згідно проекту

Ймовірність ризику(Й)		Вплив загрози(ризику)		
		Низький	Середній	Високий
		1	2	3
Низька	1		1,4	3,7,10
Середня	2	6,13	8,9,11	2,12
Висока	3			5

За отриманим індексом впливу, ризик підпадає під один з типів представлених в таблиці Б.5.

Таблиця Б.5 – Шкала оцінювання за рівнем ризику.

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$1 \leq R \leq 3$	1,4,6,13
2	Виправдані	$R=4$	3,7,8,9,10,11
3	Недопустимі	$4 < R \leq 6$	2,5,12

Ризики, ймовірність виникнення, вплив та стратегії реагування на них продемонстровані в таблиці Б.6.

Таблиця Б.6 – Ризики та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	3	1. Створити документацію плану розробки, в якому буде вказане бачення проекту в зрозумілому для замовника вигляді 2. Продемонструвати замовнику план. 3. Прийди до консенсусу в спірних питаннях.	Прийняття	
RS_2	Відкритий	Нечітке завдання на розробку	Середня	Високий	5	1.Ясно і однозначно обговорити із замовником усі види вимог. 2.Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів. 3.Періодичний контроль замовником етапів роботи.	Попередження	При виявленні невідповідностей деяких характеристик продукту заявленим вимогам потрібно уважно та чітко окреслити те, що було виконано невірно та зробити правки.
RS_3	Відкритий	Виникнення проблем із програмним забезпеченням користувачів	Низька	Високий	4	1.Розробка проекту з урахуванням вимог до програмного забезпечення користувачів проекту. 2.Модифікація проекту з урахуванням різних версій програмного забезпечення, яке буде застосовуватися.	Прийняття	

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Низька кваліфікація розробників	Низька	Середній	3	1.Підвищити кваліфікацію персоналу. 2. Переглянути онлайн-ресурси для підвищення рівня знань.	Пом’якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки.
RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	6	Провести аналіз актуальності найважливіших процесів та робіт. Звернути особливу увагу на правильність розподілу часу. Правильно визначити пріоритети виконання робіт. Чітко дотримуватися календарного плану.	Пом’якшення	Змінити порядок пріоритетів робіт. Знайти способи оптимізації роботи з вже існуючою розстановкою. Обговорити варіанти внесення поправок до термінів реалізації із замовником.
RS_6	Відкритий	Часте внесення змін у ТЗ	Середня	Низький	3	1.Виділити всі необхідні параметри проекту 2.Чітко описати вимоги до проекту. 3.Обговорити всі технічні засоби виконання проекту та умови реалізації.	Перенесення	Узгодити всі положення з замовником, у разі потреби внести необхідні зміни та поправки.

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_7	Відкритий	Вибір не ефективної технології розробки	Низка	Високий	4	1.Проаналізувати методи та засоби, для виконання проекту. 2.Обрати технологію, яка краще всього підходить для виконання поставлених задач	Пом'якшення	Виділити час та ресурсі на пошуки покращення обраної технології. Застосувати допоміжні ресурси.
RS_8	Відкритий	Неправильна оцінка в масштабі проекту	Середня	Середній	4	1.Провести детальний аналіз проекту. 2.Визначити основні етапу проекту, розподілити час на їх виконання. 3.Проаналізувати масштаби проекту на основі додаткових джерел.	Пом'якшення	Переоцінка масштабів проекту. Перебудова стратегії реалізації проекту.
RS_9	Відкритий	Помилки проектування	Середня	Середній	4	На етапі проектування тісно співпрацювати із замовником та на певних етапах демонструвати поточні результати.	Пом'якшення	Здійснювати проміжний контроль результатів в ході виконання проекту.

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_10	Відкритий	Збої в роботі програмного забезпечення	Низька	Високий	4	1.Підготувати резерв програмних засобів. 2.Залучити спеціаліста для усунення збоїв.	Попередження	Замінити програмне забезпечення.
RS_11	Відкритий	Відсутність резервних копій даних	Середня	Середній	4	1.Налаштувати автоматичне збереження даних. 2.Зберігати дані на різних носіях інформації.	Попередження	Робити копію даних після кожного виконаного етапу.
RS_12	Відкритий	Зміна вимог замовника в процесі розробки проекту	Середня	Високий	5	Узгодити всі питання на початкових етапах, щоб мінімізувати кількість змін під час розробки.	Пом’якшення	Переоцінка проекту, кожного разу, коли вимоги змінюються
RS_13	Відкритий	Невиконання моніторингу проекту	Середня	Низький	3	Здійснювати проміжний контроль результатів в ході виконання проекту. Здійснювати моніторинг проекту працівниками.	Перенесення	Здійснювати моніторинг проекту замовником. Надання проміжних результатів виконання проекту після кожного етапу.

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Web application to support the activities of employee of the project organization in KZAPR

Denys Shelekhov, *student;*
Viktoriia Antypenko, *associate professor;*
Viktor Nenia, *associate professor*
Sumy State University, Sumy, Ukraine

There are many different applications in the world today. A lot of them are used to simplify one or another activity in most areas of life. Examples include calculators to reduce settlement time, news web-applications that provide access to the latest events, both urban and global, which would require a lot of time to find etc.. However, there are those who differ in their own purpose from others. These are applications designed to simplify the creation of all the above developments. These can be, for instance, frameworks for programming languages.

The tools complex of the automation the designing works (KZAPR) is a software product which is created within the applied scientific research works of Sumy State University. Its using should automate the process of organizing the designing of engineering objects. It is important to develop a module to support the activities of the employee of the project organization.

KZAPR Core is a Task Manager, where such functions as creating, sending, receiving and verifying tasks in accordance with access rights are available. The executor is warned in advance about the content of the task and is provided with specialized development tools. Project documents are provided automatically. Collective peer review is a subject to special permission. This ensures that the data protection mechanism works.

The user interacts with the KZAPR core through the web-application. The software is based on an object-oriented programming methodology. Java has been chosen as the development programming language. Communication with the server is provided only by the offered means using only the commands provided in advance. All commands pass only through a specially configured filter, which prevents the destruction or modification of software and / or information.

Data exchange is provided using JSON text format and self-presentation.

ДОДАТОК Г

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ WEB-ДОДАТКУ

TaskController.java

```
@RestController
public class TaskController {
    @Autowired
    TaskRepository taskRepository;

    @Autowired
    ProjectRepository projectRepository;

    @Autowired
    StatusRepository statusRepository;

    @Autowired
    CommentRepository commentRepository;

    @Autowired
    ProfileRepository profileRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    AttachmentRepository attachmentRepository;

    // Получить все записи
    @GetMapping("/task")
    public List getAllNotes() {
        return taskRepository.findAll();
    }

    @PostMapping("/task")
    public ResponseDto createNote(@RequestBody TaskRequestDto taskRequestDto) {
        Task task = taskRequestDto.toTask();
        if (taskRequestDto.getProjectId() != null)

projectRepository.findById(taskRequestDto.getProjectId()).ifPresent(task::setProje
ct);
        if (taskRequestDto.getParentId() != null)

taskRepository.findById(taskRequestDto.getParentId()).ifPresent(task::setTask);
        if (taskRequestDto.getStatusId() != null)

statusRepository.findById(taskRequestDto.getStatusId()).ifPresent(task::setStatus)
;
        taskRepository.save(task);
        return new ResponseDto("task create", true,
                new ResponseDto.DataDTO(task.getTask_id(),
                        new
ResponseDto.DataDTO.Stat(task.getStatus().getStatus_id(),
                                task.getStatus().getName(),
task.getStatus().getColor())));
    }

    @PutMapping("/task/{taskId}")
    public ResponseDto updateTask(@RequestBody TaskRequestDto taskRequestDto,
@PathVariable(value = "taskId") Long taskId) {
        try {
            Task task = taskRepository.findById(taskId).get();
            task.setTitle(taskRequestDto.getTitle());
            task.setDescription(taskRequestDto.getDescription());
```

```
            task.setTime_start(taskRequestDto.getStartTime());
            task.setTime_end(taskRequestDto.getEndTime());
            task.setColor(taskRequestDto.getColor());
            if (taskRequestDto.getProjectId() != null)

projectRepository.findById(taskRequestDto.getProjectId()).ifPresent(task::setProje
ct);
            if (taskRequestDto.getParentId() != null)

taskRepository.findById(taskRequestDto.getParentId()).ifPresent(task::setTask);
            if (taskRequestDto.getStatusId() != null)

statusRepository.findById(taskRequestDto.getStatusId()).ifPresent(task::setStatus)
;
            taskRepository.save(task);
            return new ResponseDto("task updated", true, null);
        } catch (RuntimeException e){
            return new ResponseDto("data entered incorrectly", false, null);
        }
    }

    @DeleteMapping("/task/{taskId}")
    public ResponseDto deleteTask(@PathVariable(value = "taskId") Long taskId){
        taskRepository.deleteById(taskId);
        return new ResponseDto("task deleted", true, null);
    }

    @GetMapping("/task/{taskId}/comments")
    public GetAllCommentsOfTheTaskResponseDto
getAllComentsOfTheTask(@PathVariable(value = "taskId") Long taskId) {

        List<Comment> comments =
commentRepository.getAllCommentsOfTheTaskList(taskId);
        List<Profile> profiles =
profileRepository.getAllAuthorsOfTheTaskList(taskId);
        List<GetAllCommentsOfTheTaskResponseDto.Data> data = new ArrayList<>();

        for(int i = 0; i < comments.size(); i++) {
            data.add(new GetAllCommentsOfTheTaskResponseDto.Data(
                    comments.get(i).getComment_id(),
                    comments.get(i).getText(),

GetAllCommentsOfTheTaskResponseDto.Data.getAuthors(profiles.get(i)),
                    comments.get(i).getCreated_at(),
                    comments.get(i).getEdited_at(),
                    comments.get(i).getType()));
        }

        return new GetAllCommentsOfTheTaskResponseDto("get post comments",
data,true);
    }

    @GetMapping("/task/{taskId}")
    public GetInformationAboutTaskByIdResponseDto
getInformationAboutTaskByIdResponseDto(@PathVariable(value = "taskId") Long
taskId) {

        try {
            Task task = taskRepository.getTaskByTaskId(taskId);
            Status status = statusRepository.getStatusByTaskId(taskId);
            Project project = projectRepository.getProjectsByTaskId(taskId);
            Profile profile = profileRepository.getProfileByTaskId(taskId);
            User user = userRepository.getUserByTaskId(taskId);
            List<Attachment> attachments =
attachmentRepository.getUserByTaskId(taskId);

            List<GetInformationAboutTaskByIdResponseDto.Data.Attached> attached =
new ArrayList<>();

            for (Attachment a : attachments) {

attached.add(GetInformationAboutTaskByIdResponseDto.Data.toAttached(a));
```

```
            }

            GetInformationAboutTaskByIdResponseDto.Data data = new
GetInformationAboutTaskByIdResponseDto.Data(
                    task.getTask_id(),
                    task.getTitle(),
                    task.getDescription(),
                    GetInformationAboutTaskByIdResponseDto.Data.toStatus(status),

GetInformationAboutTaskByIdResponseDto.Data.toProject(project),

GetInformationAboutTaskByIdResponseDto.Data.toAssignTo(profile, user),
                    task.getTime_start(),
                    task.getTime_end(),
                    task.getColor(),
                    attached
            );

            return new GetInformationAboutTaskByIdResponseDto("post data", data,
true);
        } catch (RuntimeException e){
            return new GetInformationAboutTaskByIdResponseDto("record not found by
id", null, false);
        }

    }

}
```

TaskRequestDto.java

```
@AllArgsConstructor
@Getter
@ToString
public class TaskRequestDto {

    private String title;
    private String description;
    private Timestamp startTime;
    private Timestamp endTime;
    private String color;
    private Long projectId;
    private Long parentId;
    private Long statusId;

    public Task toTask() {
        Task task = new Task();
        task.setTitle(title);
        task.setDescription(description);
        task.setTime_start(startTime);
        task.setTime_end(endTime);
        task.setColor(color);
        return task;
    }

}
```

ResponseDto.java

```
@AllArgsConstructor
@Getter
@ToString
public class ResponseDto {

    private String message;
    private Boolean success;
    private ResponseDto.DataDTO data;

    @AllArgsConstructor
    @Getter
    @ToString
```

```
    @NoArgsConstructor
    public static class DataDTO {
        private Long id;
        private ResponseDto.DataDTO.Stat status;

        @AllArgsConstructor
        @Getter
        @ToString
        public static class Stat {
            private Long statusId;
            private String name;
            private String color;
        }
    }

}
```

Task.java

```
@Data
@NoArgsConstructor
@Entity
@Table(name = "task")
public class Task {

    @Id
    @GeneratedValue
    private Long task_id;

    @NotBlank
    private String title;

    @NotBlank
    private String description;

    private Timestamp time_start;

    private Timestamp time_end;

    private String color;

    @OneToOne
    @JoinColumn(referencedColumnName = "task_id",name = "parent_id")
    private Task task;

    @ManyToOne
    @JoinColumn(name = "project_id")
    private Project project;

    @ManyToOne
    @JoinColumn(name = "status_id")
    private Status status;

    @OneToMany(mappedBy = "task")
    private List<Comment> comments;

    @Override
    public String toString() {
        return "Task{" +
                "task_id=" + task_id +
                ", title='" + title + '\'' +
                ", description='" + description + '\'' +
                ", time_start=" + time_start +
                ", time_end=" + time_end +
                ", color='" + color + '\'' +
                '}';
    }
}
```

ProfileRepository.java

```
@Repository
```

```
public interface ProfileRepository extends JpaRepository<Profile, Long> {

    @Query(value="select a.email from public.user a,public.profile b where
a.user_id = b.user_id AND b.profile_id = :id", nativeQuery=true)
    String getUserEmailByProfileId(Long id);

    @Query(value="select p.*\n" +
            "from public.profile p,public.project_staff ps, public.project pr,
public.roles r, public.profile_roles pf\n" +
            "where ps.project_id = pr.project_id AND ps.profile_id = p.profile_id
AND pf.role_id = r.role_id AND r.name = 'pm' AND pr.project_id = :id",
nativeQuery=true)
    List<Profile> getProfileIdByProjectIdForPm(Long id);

    @Query(value="select p.* from public.comment c, public.profile p where
c.author_id = p.profile_id AND c.task_id = :id", nativeQuery=true)
    List<Profile> getAllAuthorsOfTheTaskList(Long id);

    @Query(value="select pf.* from public.profile pf, public.task t,
public.project_staff ps, public.project pj  " +
            "where pf.profile_id=ps.profile_id AND ps.project_id=pj.project_id AND
pj.project_id=t.project_id AND t.task_id = :id", nativeQuery=true)
    Profile getProfileByTaskId(Long id);

    @Query(value = "with recursive project_manager as (" +
            "    select ps.profile_id, ps.projectstaff_id, ps.chief_id" +
            "    from public.profile_roles pr" +
            "             join public.project_staff ps on pr.profile_role_id =
ps.profile_id" +
            "    where pr.profile_id = ?1" +
            "    union" +
            "    select pr.profile_role_id, ps.projectstaff_id, ps.chief_id" +
            "    from public.project_staff ps" +
            "    join project_manager pm on pm.chief_id = ps.projectstaff_id" +
            "    join public.profile_roles pr on pr.profile_role_id =
ps.profile_id" +
            ")" +
            "select p.* from public.profile_roles pr" +
            "         join public.profile p on pr.profile_id = p.profile_id " +
            "where pr.profile_role_id = (select project_manager.profile_id from
project_manager order by projectstaff_id limit 1)", nativeQuery = true)
    Profile getProjectManagerByProfileId(long profileId);

}
```