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

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток підтримки діяльності керівника проектної організації у КЗАПР»

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

Виконавець роботи: студентка групи ІТ-82-0 Медведєва Катерина Сергіївна

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. кафедрою ІТП

_____ В. В. Шендрик
«____»_____2022 р.

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

Медведєва Катерина Сергіївна

1 Тема роботи *Web-додаток підтримки діяльності керівника проектної організації у КЗАПР*

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

затверджені наказом по університету від «*27*» квітня 2022 р. №0301_VI

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

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

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та 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 6 жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Розробка планування робіт	02.05.2022 – 07.05.2022	
2	Написання технічного завдання	07.05.2022 – 14.05.2022	
3	Аналіз предметної області	14.05.2022 – 18.05.2022	
4	Проектування web-додатку	18.05.2022 – 25.05.2022	
5	Розробка web-додатку	25.05.2022 – 07.06.2022	
6	Тестування web-додатку	07.06.2022 – 08.06.2022	
7	Завантаження web-додатку на хостинг	08.06.2022 – 10.06.2022	
8	Оформлення пояснювальної записки	02.05.2022 – 12.06.2022	

Студент _____ Медведєва К.С.
(підпис)

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

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підтримки діяльності керівника проектної організації у КЗАПР».

Пояснювальна записка у своєму складі має вступ, три розділи, висновок, список використаних джерел із 30 найменувань, чотири додатки. Загальний обсяг пояснювальної записки складає 95 сторінка, у тому числі 43 сторінки основного тексту, 3 сторінки списку використаних джерел, 48 сторінки додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку підтримки діяльності керівника проектної організації у КЗАПР.

У першому розділі було розглянуто останні дослідження за тематикою даного проекту. Також проведено аналіз аналогів розроблюваного web-додатку, виділено їх переваги та недоліки. Крім того визначено засоби реалізації проекту, його мету та задачі.

У другому розділі було проведено структурно-функціональне моделювання, сформовані варіанти використання web-додатку та спроектовано базу даних. У результаті були розроблені діаграма IDEF0 та її декомпозиції, діаграма варіантів використання та схема бази даних.

У третьому розділі було описано процес розробки web-додатку, продемонстровано архітектуру та дизайн програмного продукту, а також проведено функціональне тестування даного програмного продукту.

Ключові слова: WEB-ДОДАТОК, ПРОЕКТНА ОРГАНІЗАЦІЯ, КЕРІВНИК, КОНТРАКТ, ПРОЕКТ, ПРОГРЕС, БАЗА ДАНИХ, JAVASCRIPT, REACTJS, MUI.

ЗМІСТ

ВСТУП

Використання сучасних інформаційних технологій (ІТ) дозволяє автоматизувати процес проектування задач задля прискорення та полегшення здійснення контролю над виконанням тих чи інших робіт.

Принципово необхідним планування є для багатьох керівників підприємств. Особливо для тих, які зобов’язані організовувати виконання виробничих завдань своїми підлеглими як у передбачуваних, так і екстремальних умовах. Наприклад, в умовах пандемії чи бойових дій тощо. Під час такого виду діяльності складно контролювати прогрес виконання завдання, оскільки працівник фактично відсутній в офісі. Здійснення комунікації засобами соціальних мереж також не є повним вирішенням питання, оскільки вони не забезпечують усіх необхідних функцій повідомлень, структуризації завдань і т.д.. Зі сторони керівника зручніше перевіряти виконання робіт у спеціальному створеному web-додатку, ніж зв’язуватися з кожним підлеглим щодо його досягнень та результатів по декілька разів на день.

Планування виробничих процесів, контроль над здійсненням завдань, перегляд статистики розробки, підписання контрактів – задачі, які реалізуються задля успішної колективної діяльності. Найбільший ефект досягається на єдиній інформаційній основі з використанням уніфікованого опрацювання усіх вирішуваних задач у комплексі засобів автоматизації проектувальних робіт (КЗАПР).

Отже, метою даного дослідження є розробка web-додатку для підтримки діяльності керівника проектної організації у КЗАПР та автоматизація таких процесів функціонування, як підписання контрактів, відображення статистики виконання проектів, назначення відповідальних над виконанням завдань та створення проектів.

Для досягнення мети проекту необхідно виконати наступні задачі:

− дослідити предмету область та цільову аудиторію, визначити актуальність роботи;

− виконати аналіз аналогів web-додатків і визначити їх плюси та мінуси;

- провести огляд останніх публікацій та літератури;
- спроектувати модель та структуру web-додатку;
- визначити технології для розробки web-додатку;
- розробити прототип web-додатку;
- відтворити структуру web-додатку;
- реалізувати функціонал web-додатку для підтримки діяльності керівника проектної організації у КЗАПР;
- виконати тестування web-додатку.

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

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Конкурентоспроможність сучасних підприємств прямо пропорційно залежить від впровадження інноваційних підходів у процеси управління технологічними, виробничими та фінансовими процесами [2]. Найчастішим результатом автоматизації процесу проектування в машинобудівній галузі є застосування систем автоматизації проектувальних робіт (САПР) [3]. Це є одним із найпопулярніших напрямків у даній сфері. САПР використовують для полегшення процесу виконання окремих проектувальних процедур. Наприклад, найпоширенішими є програмне забезпечення AutoCAD [4], CATIA [5], SolidWorks [6]. Вони надають функціонал для автоматизованого виконання креслення й проектування. Проте, їх сучасні розробки приділяють недостатньо уваги управлінню як цілим процесом проектування об’єктів машинобудування, так і підприємством загалом. Як зазначалося раніше, використання САПР дійсно спрощують виконання лише окремих проектувальних процедур. А функціонал для повноцінного управління організацією зі сторони її безпосереднього керівника відсутній. Це є одним із головних недоліків САПР.

Інструменти управління в масштабі всієї організації є потужними інструментами для досягнення конкурентоспроможності, зростання та розвитку будь-якого бізнесу, незалежно від галузі. Сучасні рішення в цій сфері в основному спрямовані на оптимізацію роботи керівництва компанії і відіграють важливу роль у підвищенні ефективності діяльності підприємства. [7]. Вирішенням питання автоматизації процесу управління організацією є створення web-орієнтованого додатку, який буде візуалізувати структуру підприємства, прогрес виконання проектів та контрактів, а також дозволить керівнику віддалено підписувати контракти та назначати відповідального проектного менеджера за проектом. Тому розробка web-додатку для підтримки діяльності керівника проектної організації є актуальною.

1.2 Аналіз існуючих продуктів-аналогів

У мережі Інтернет вже давно є доступними та функціонують web-додатки для розробки проектів. Вони забезпечують весь необхідний функціонал для планування завдань. Проте такі сервіси не забезпечують деякі специфічні функції для керування співробітниками підприємства машинобудівного профілю.

Для визначення вимог майбутнього програмного продукту було проведено дослідження існуючих аналогів web-додатків, а саме «MeisterTask», «Trello» та «Asana».

Першим було розглянуто сервіс «MeisterTask» [8]. Він слугує корисним інструментом для управління великими організаціями та групами в середині них. Як і типовий web-додаток для планування процесів «MeisterTask» має функціонал для створення проектів та завдань до них. На головній сторінці (рис. 1.1) зображені задачі у вигляді карток і відсортовані за статусом їх виконання. Також є можливість представити завдання у вигляді часової прямої.

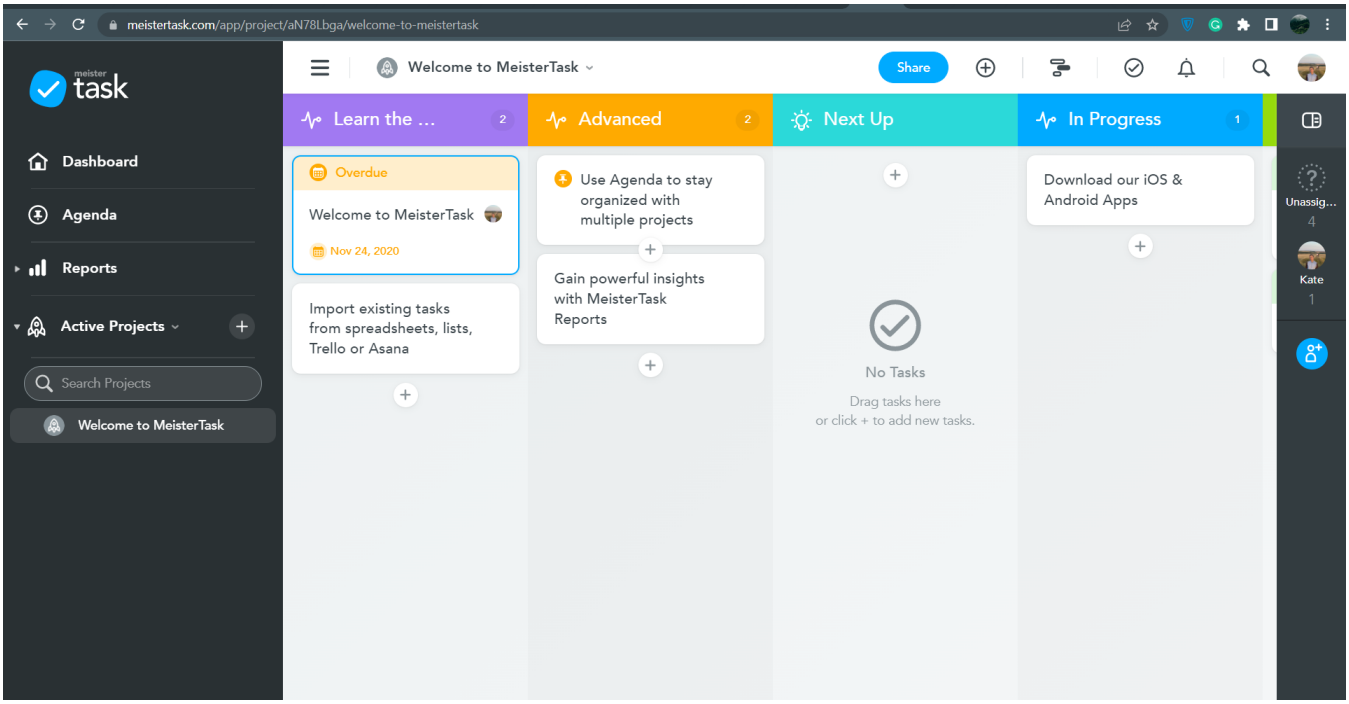


Рисунок 1.1 – Головна сторінка проекту у web-додатку «MeisterTask»

Сторінка статистики (рис. 1.2) містить гістограму з відсортованими за статусом задачами. Це дозволяє керівнику компанії легко оцінити прогрес та кількість виконаних завдань за певний часовий період.

Якщо коротко охарактеризувати дизайн web-додатку, то він відповідає останнім трендам web-дизайну та витриманий в одному стилі. Доступна світла та темна версія сайту, кольорова гамма витримана у відтінках Material Design [9]. Але цей додаток не забезпечує керівника функцією підписання та створення контракту.

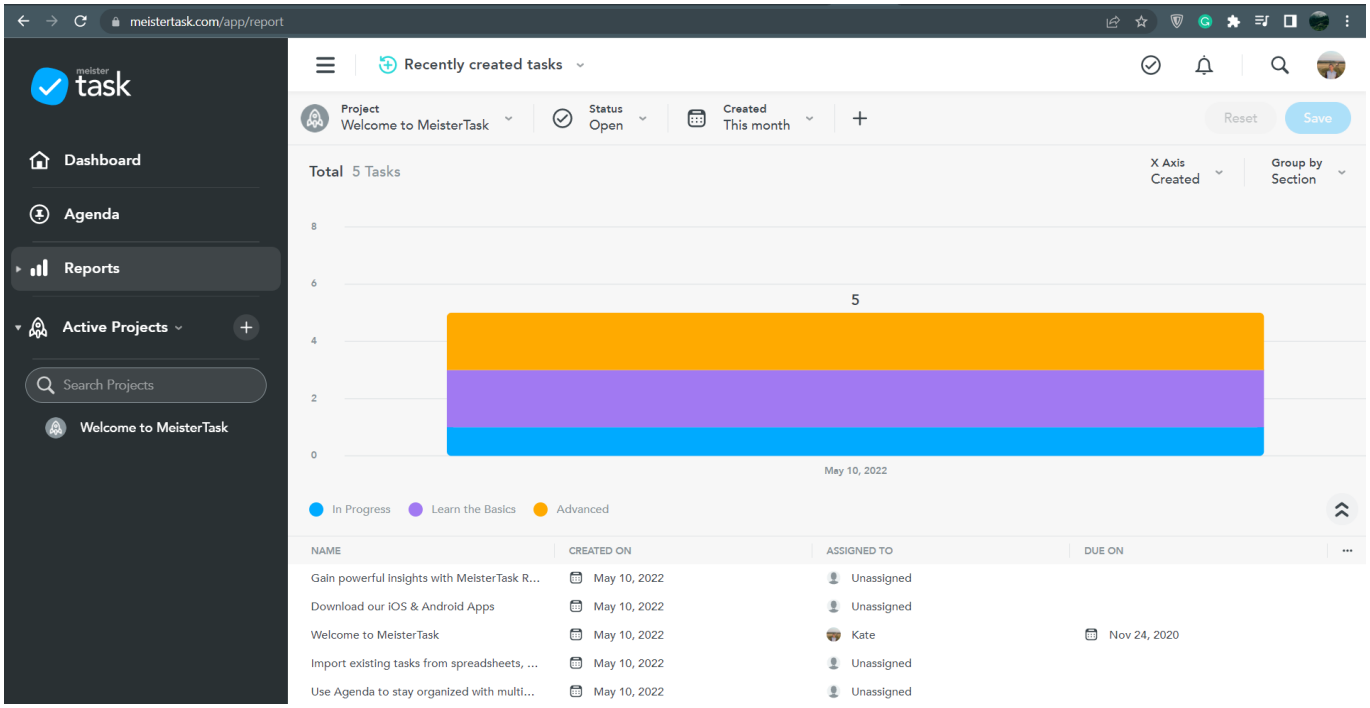


Рисунок 1.2 – Сторінка статистики web-додатку «MeisterTask»

Наступним було досліджено програмний продукт «Trello» [10], який є універсальним додатком для управління проектам та процесами. Він здатний підлаштовуватись до будь-якої сфери використання завдяки високому рівню власних можливостей адаптуватись до запитів користувачів. Його головна сторінка містить перелік проектів. Вона представлена на рисунку 1.3. Сторінка проекту (рис. 1.4) містить полотно для створення завдань із можливістю їх переведення у різні задані користувачем статуси. У додатку реалізована можливість запрошувати

співробітників до проекту за допомогою посилання. Проте відсутня можливість перегляду статистики за діями співробітників.

Дизайн даного web-додатку сучасний, кольорову палітру складають відтінки синього та блакитного. Він має досить продуманий UX дизайн [11]. Додатком можна користуватися без особливих навичок, розташування елементів є інтуїтивно зрозумілим.

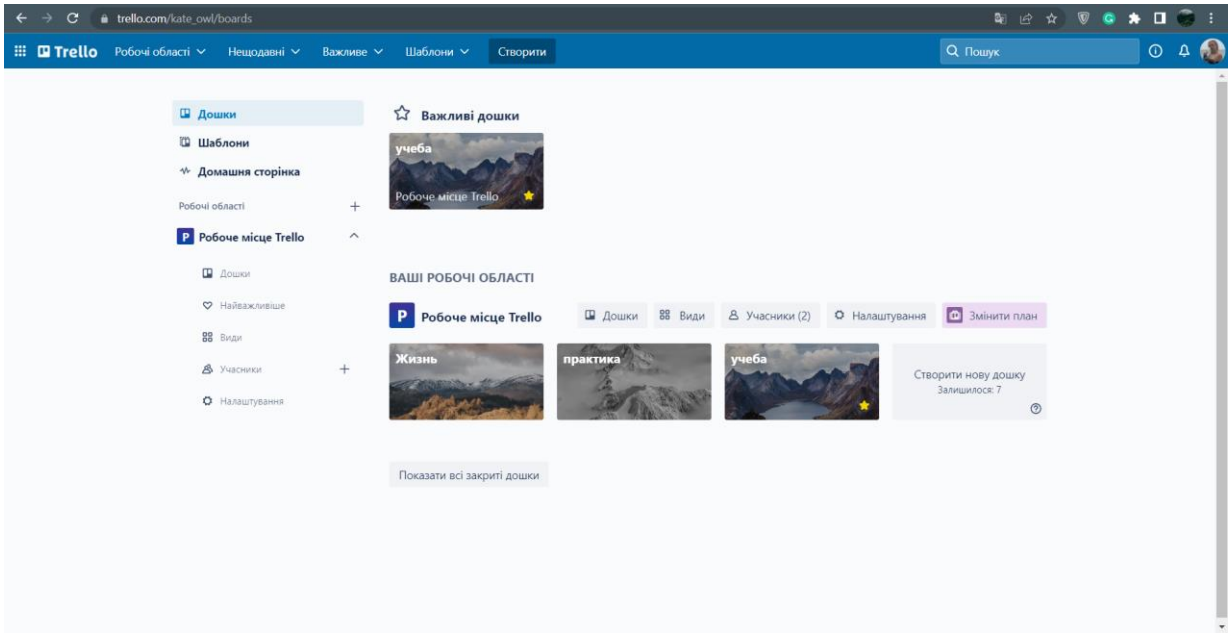


Рисунок 1.3 – Головна сторінка web-додатку «Trello»

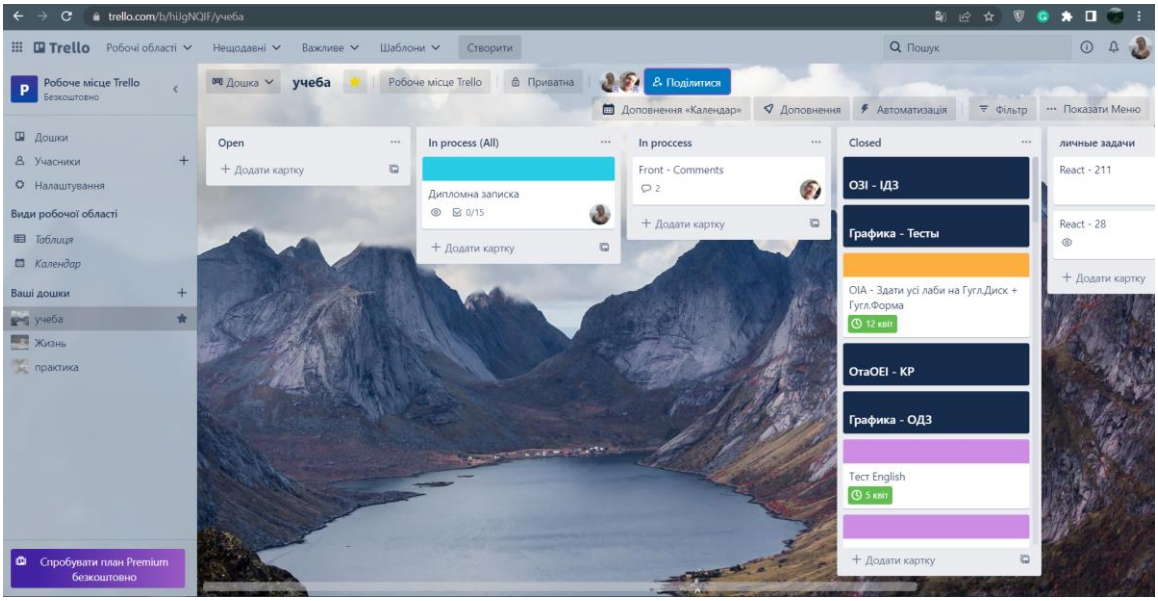


Рисунок 1.4 – Сторінка проекту web-додатку «Trello»

Також було проаналізовано web-додаток «Asana» [12]. На його головній сторінці (рис.1.5) представлено проект та пов’язані з ним задачі. Також є зручна можливість запросити співробітників до проекту за посиланням.

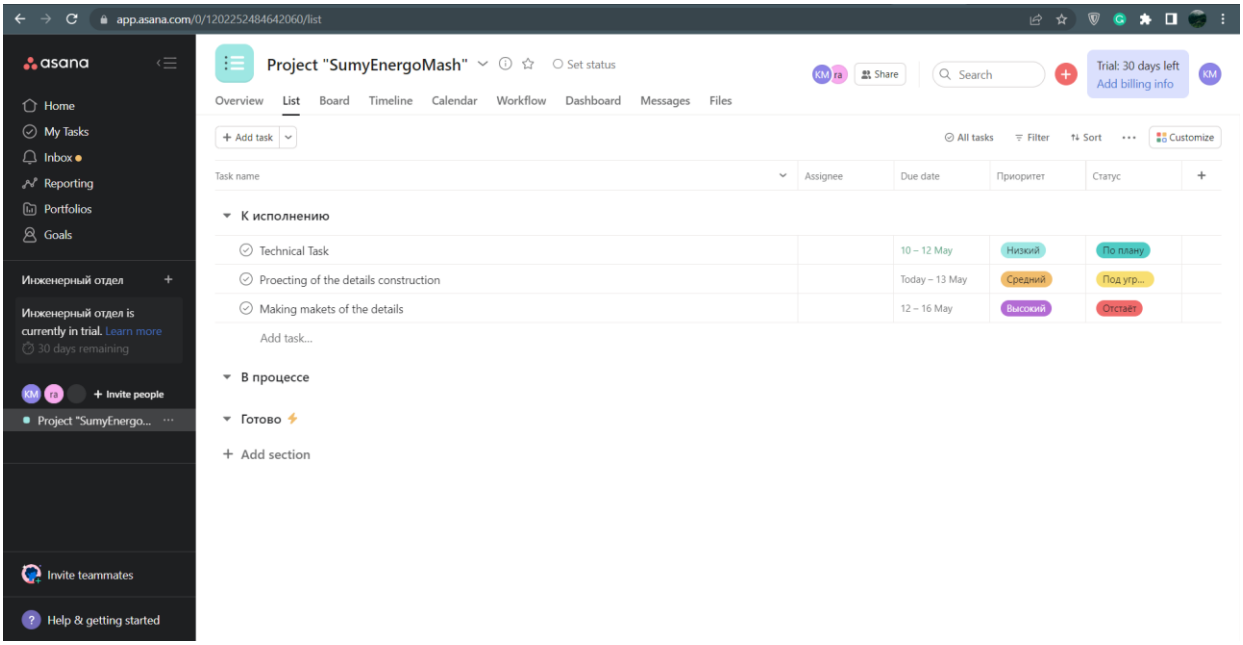


Рисунок 1.5 – Головна сторінка web-додатку «Asana»

Web-сторінка «Portfolio» містить приведену статистику стосовно прогресу виконання проектів (рис. 1.6).

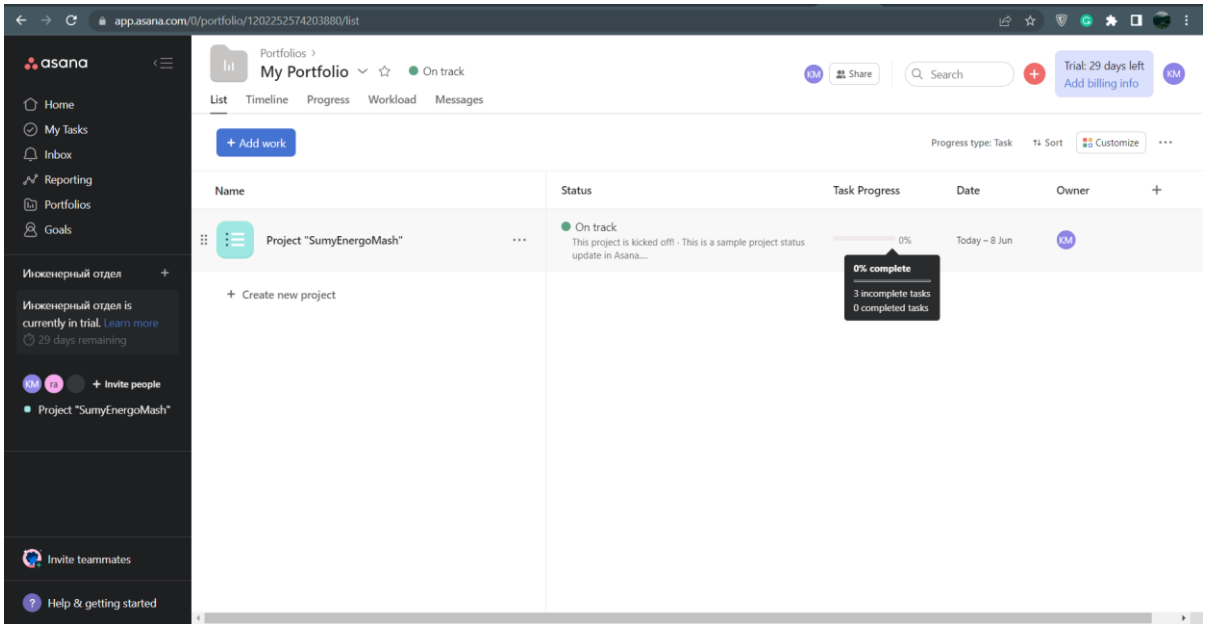


Рисунок 1.6 – Сторінка статистики web-додатку «Asana»

Дизайн сервісу «Asana» відповідає сучасним трендам web-дизайну та концепції Material design. Його кольорова гамма є приємною, переважають чорні, сині та білі кольори. Але даний програмний продукт не забезпечує керівника функціоналом створення та підписання контракту.

Проаналізувавши аналоги web-додатків для планування робіт і управління організацією було визначено їх плюси та мінуси. Його результати представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика web-додатків-аналогів

Характеристика/ Web-додаток	**«MeisterTask»**	**«Trello»**	**«Asana»**
Сучасний дизайн	+	+	+
Зручний інтерфейс	+	+	+
Інтерактивність	+	+	+
Функціональність	+	-	+
Навігація	+	+	+
Авторизація користувачів	+	+	+
Перегляд статистики	+	-	+
Можливість створення проекту	+	+	+
Можливість запрошувати співробітників до проекту	+	+	+
Можливість підписання контракту	-	-	-

У ході власної розробки дані з таблиці 1.1 надають змогу звернути увагу на необхідний функціонал, який треба реалізувати, долаючи виявлені недоліки розглянутих web-додатків. Створюваний програмний продукт повинен мати інтерактивну навігацію, сучасний дизайн, інтуїтивно-зрозумілий інтерфейс. Із функціонала варто реалізувати можливість підписання контракту, створення проекту та задач, запрошення співробітників до проекту, перегляд статистики та авторизацію користувачів.

1.3 Постановка задачі

Метою даного дослідження є розробка web-додатку для підтримки діяльності керівника проектної організації у КЗАПР та автоматизація таких процесів функціонування, як підписання контрактів, відображення статистики виконання проектів, назначення відповідальних над виконанням завдань та створення проектів. Він повинен надавати необхідний функціонал для управління персоналом та мати зручний інтуїтивно зрозумілий інтерфейс, який забезпечить легку організацію виконання відповідних процесів із управління підприємством за рахунок їх автоматизації.

Основні вимоги до створюваного програмного продукту є наступними:

- реалізувати можливість створення та підписання контракту;
- забезпечити керівника компанії статистикою виконання контракту;
- створити web-форми для створення проекту, контракту та підпису контракту;
- розробити сторінку для керівника компанії.

Щоб досягнути мету проекту необхідно виконати наступні задачі:

- дослідити предмету область та цільову аудиторію, визначити актуальність роботи;
- виконати аналіз аналогів web-додатків і визначити їх плюси та мінуси;
- провести огляд останніх публікацій та літератури;
- спроектувати модель та структуру web-додатку;
- визначити технології для розробки web-додатку;
- розробити прототип web-додатку;
- відтворити структуру web-додатку;
- реалізувати функціонал web-додатку для підтримки діяльності керівника проектної організації у КЗАПР;
- виконати тестування web-додатку.

Вимоги до структури web-додатку, видів забезпечення та його функціонування викладені в технічному завданні (Додаток А).

Для реалізації даного проекту було обрано такі технології, як Java-script [13] бібліотеку React [14] для створення інтерфейсів користувача, каскадні таблиці стилів CSS [15] для надання web-сторінці адаптивності та скриптову метамову Sass [16], яка інтерпретується в каскадні таблиці стилів, скриптову мову програмування JavaScript для асинхронної роботи з серверною частиною, а також бібліотеку MUI [17], яка пропонує повний набір інструментів інтерфейсу користувача. Для розробки бази даних було обрано СУБД PostgreSQL [18].

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

2.1 Структурно-функціональне моделювання

Характеризування системи за допомогою методології IDEF0 називається функціональною моделлю [19]. Її використовують для опису вже існуючих бізнес-процесів, в яких застосовується текстова та графічна мова.

Методологія IDEF0 означає реалізацію ієрархічної системи діаграм – одиничних описів фрагментів системи. Спочатку відбувається побудова контекстної діаграми, яка описую систему та її взаємодію з навколишнім світом. Після цього етапу будується функціональна декомпозиція, яка розбиває систему на підсистеми для докладного опису кожного елементу.

В основі IDEF0-методології існує 4 основних компоненти: функціональний блок, інтерфейсна дуга(стрілка), декомпозиція та глосарій.

Функціональне моделювання web-додатку підтримки діяльності керівника проектної організації у КЗАПР в IDEF0 представлене на рисунку 2.1.

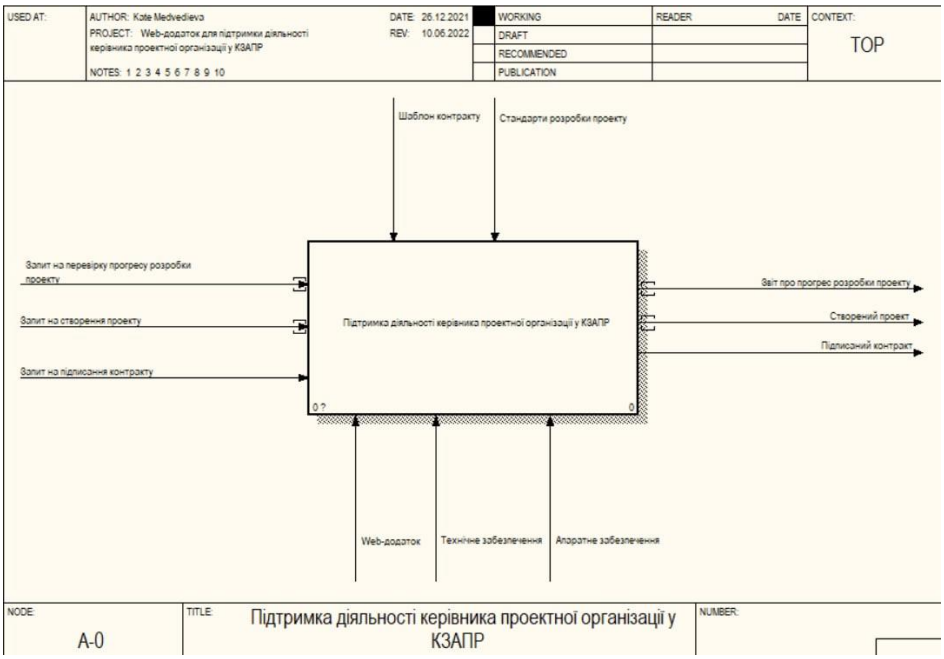


Рисунок 2.1 – Функціональна діаграма

Для деталізації внутрішніх процесів було виконано декомпозицію представленної діаграми IDEF0 (рис. 2.2).

Блоком для декомпозиції є підпис контракту на реалізацію проекту.

Для полегшення представлення інформації для кожного підпроцесу створюються таблиці введення та виведення даних (табл. 2.1).

Діаграма декомпозиції процесу підпису контракту на реалізацію проекту представлена такими підпроцесами:

- затвердження специфікацій до проекту;
- затвердження термінів та бюджету проекту;
- підписання контракту.

Таблиця 2.1 – Вхідні та вихідні дані для діаграми підписання контракту

Стрілка/ Підпроцес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Затвердження специфікацій до проекту	Інформація про специфікації проекту, Запит на підписання контракту	Стандарти розробки проекту	Керівник проекту	Затвердження специфікації
Затвердження термінів та бюджету проекту	Затвердження специфікації		Замовник, Керівник проекту	Затвердженні терміни розробки проекту, Затверджений бюджет проекту
Підписання контракту	Затвердженні терміни розробки проекту, Затверджений бюджет проекту	Шаблон контракту	Замовник, Керівник проекту, Web-додаток, Технічне забезпечення, Апаратне забезпечення	Підписаний контракт

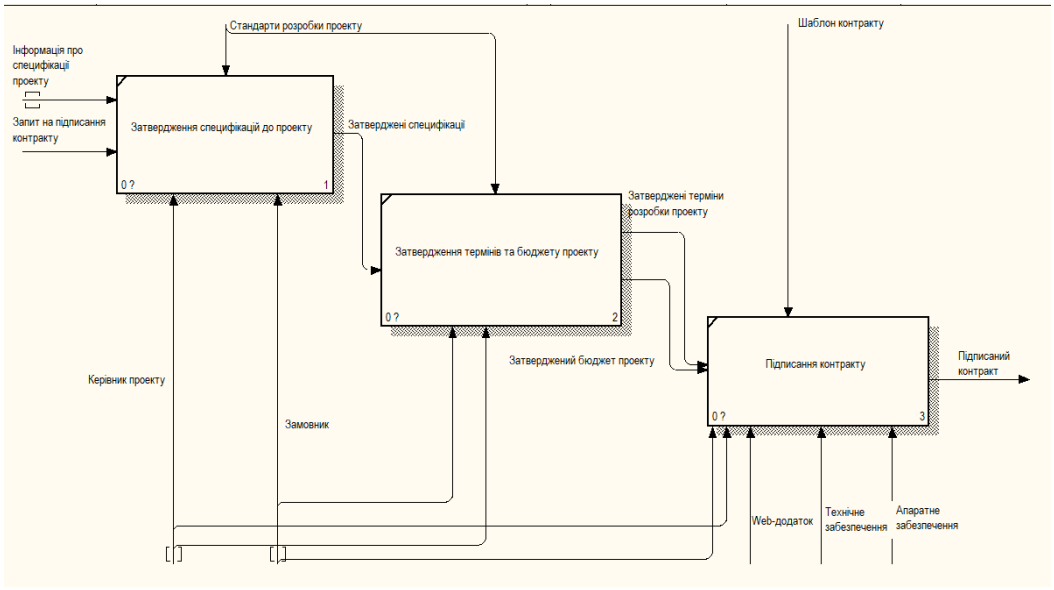


Рисунок 2.2 – Діаграма декомпозиції підпису контракту

2.2 Моделювання варіантів використання

Після проведення моделювання процесів та аналізу структури майбутнього web-додатку необхідно розробити діаграму варіантів використання.

Уніфікована мова моделювання (UML) [20], яка підходить для широкого класу проектованих програмних систем, описує об'єкт в одному чітко зазначеному синтаксисі.

За допомогою UML відбувається візуальне моделювання абстрактної концептуальної схеми, логічної та фізичної моделі. Спочатку здійснюється побудова діаграми варіантів використання, яка характеризує функціональне призначення у системи. Її застосовують для аналізу взаємодії останньої з учасниками. Деталі реалізації програмного продукту не беруться в такому випадку до уваги. У загальному сенсі use-case діаграма описує який функціонал необхідний кожній групі користувачів.

Для web-додатку підтримки керівника проектної організації варіанти використання є наступними:

- перегляд завдань;
- розбиття проекту на завдання;
- зміна статусу виконання завдання;
- написання ТЗ;
- ініціалізація проекту;
- керування репозиторієм;
- підписання контракту;
- перегляд прогресу виконання завдань;
- керування інформацією користувачів.

Акторами на діаграмі є виконавець, адміністратор, замовник, власник компанії, проектний менеджер.

Діаграма варіантів використання web-додатку підтримки діяльності керівника проектної організації у КЗАПР представлена на рисунку 2.3.

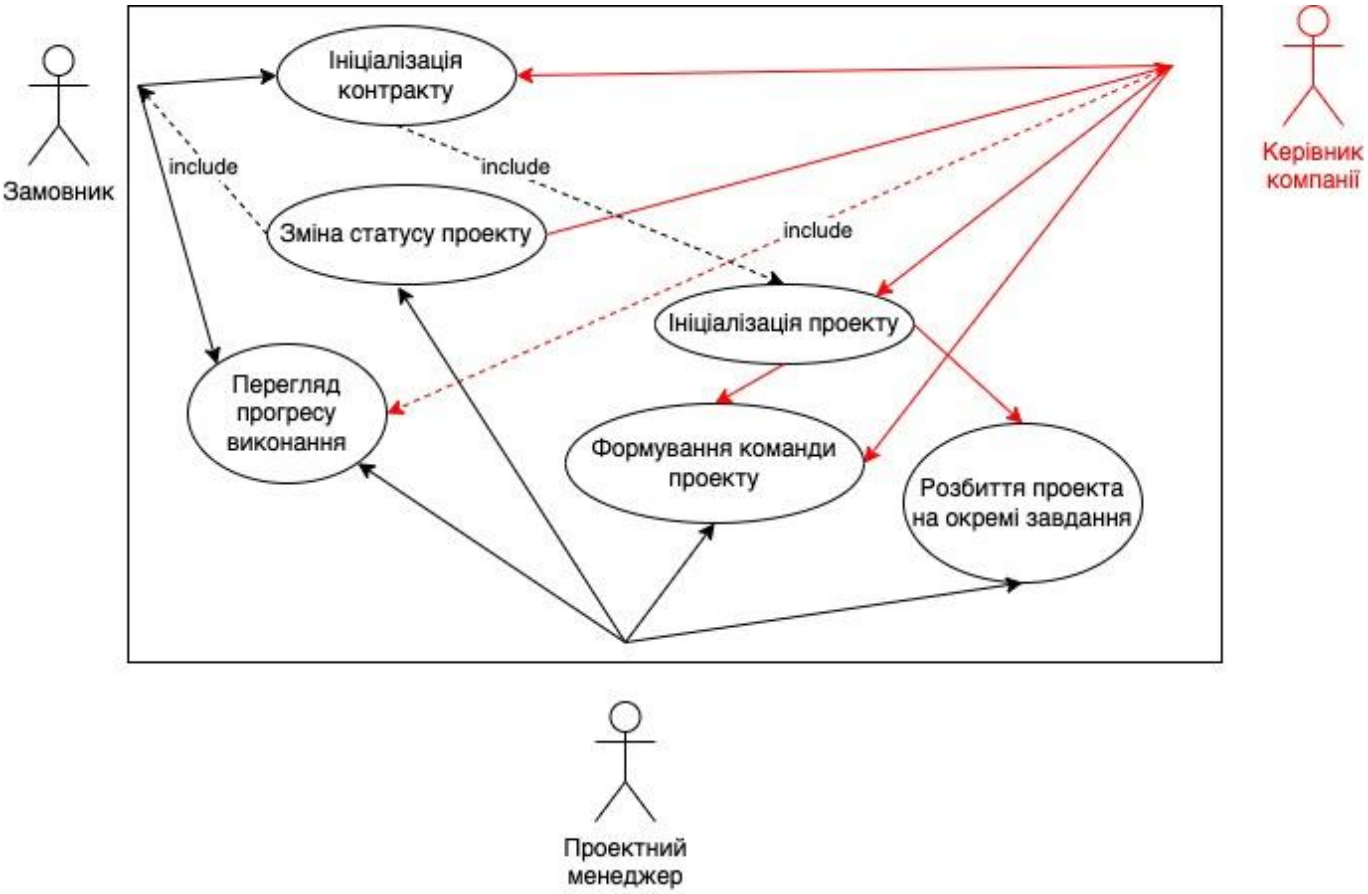


Рисунок 2.3 – Діаграма варіантів використання

2.3 Проектування бази даних

Для зручної маніпуляції з великою кількістю даних в web-додатку використовують систему управління базами даних (БД). У свою чергу БД можна представити як сховище однотипних даних.

Першим кроком для створення бази даних є проектування. У результаті проектування БД визначаються сутності з їх аргументами. Також зазначається їх взаємозв’язок між собою (логічні зв’язки). Сутності представляють у вигляді таблиці, яка має певну структуру: зміст стовпців, типи даних, розмір атрибутів та первинний ключ.

Під час проектування даних було виділено наступні сутності для КЗАПР:

- Логи (Logs);
- Користувач (User);
- Профіль (Profile);
- Доступ (Permission);
- Ресурс (Resource);
- Ролі (Roles);
- Правила (Rules);
- Статус (Status);
- Контракт (Contract);
- Проект (Project);
- Співробітники проекту (ProjectStaff);
- Повідомлення (Notification);
- Список повідомлень (NotificationList);
- Роль профілю (ProfileRoles);
- Вкладення (Attachment);
- Список вкладень (AttachmentList);
- Завдання (Task);
- Коментар (Comment);
- Список завдань (TaskList);
- Запити в підтримку (Support_Requests).

На рисунку 2.6 зображена фізична модель бази даних web-додатку підтримки діяльності керівника проектної організації у КЗАПР.

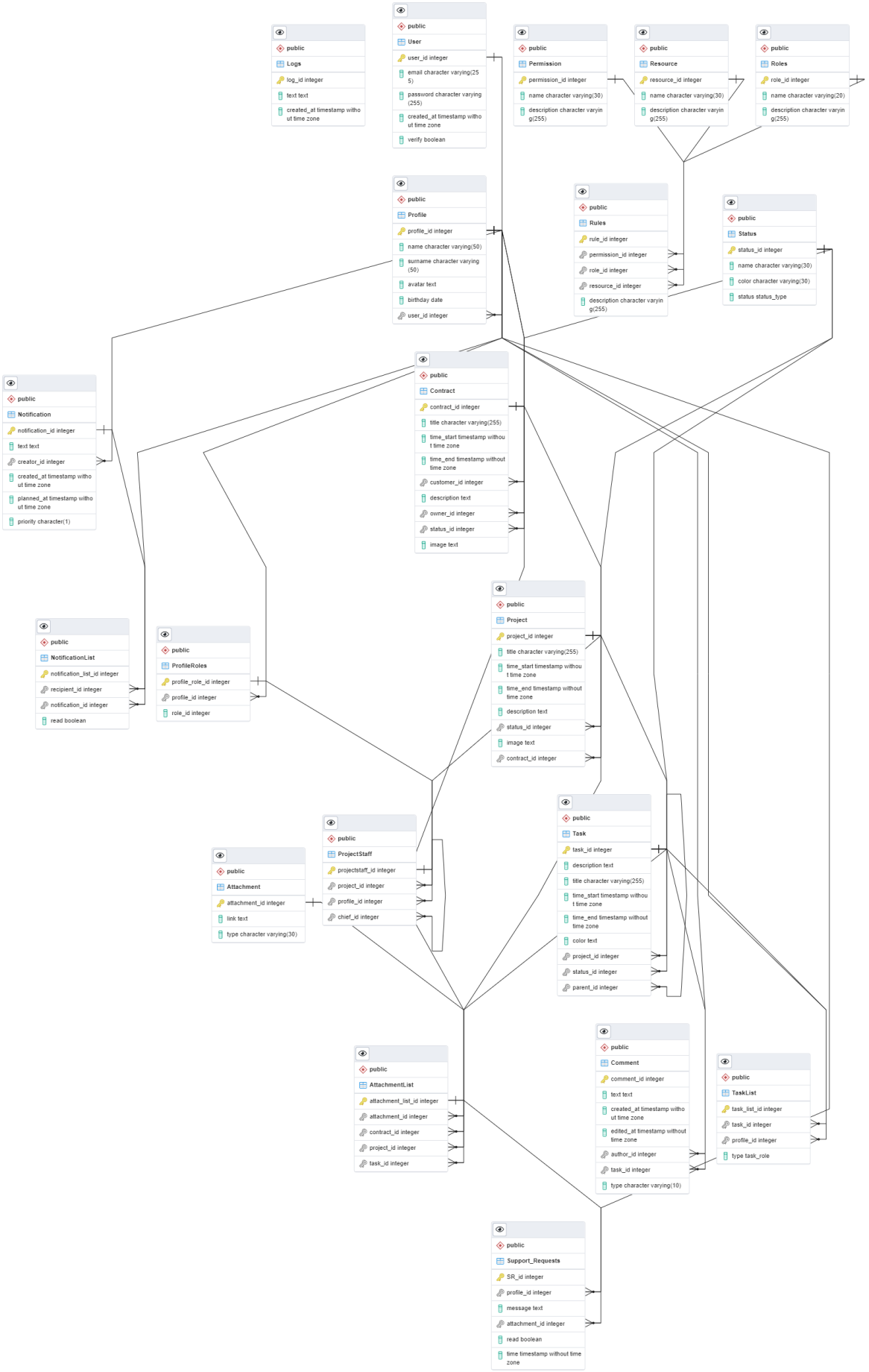


Рисунок 2.6 – Фізична модель розроблюваної бази даних

Для реалізації функціональних можливостей керівника проектної організації було виділено та створено наступні сутності:

- Контракт (Contract) – містить інформацію про контракт;
- Проект (Project) – містить інформацію про створений проект в контракті;
- Члени проекту (ProjectStaff) – інформація про співробітників проекту.

Таблиця Контракти (Contract) має наступні атрибути:

- contract_id (integer) – первинний ключ таблиці;
- title (character varying (255)) – назва контракту.
- time_start (timestamp without time zone) – дата початку дії контракту;
- time_end (timestamp without time zone) – дата кінця дії контракту;
- customer_id (integer) – foreign key замовника контракту;
- description (text) – опис контракту;
- owner_id (integer) – foreign key власника компанії;
- status_id (integer) – foreign key статусу контракту;
- image (text) – посилання на зображення.

Таблиця Проект(Project) має наступні атрибути:

- project_id (integer) – первинний ключ проекту;
- title (character varying(255)) – назва проекту;
- time_start (timestamp without time zone) – дата початку проекту;
- time_end (timestamp without time zone) – дата кінця проекту;
- description (text) – опис проекту;
- status_id (integer) – foreign key статусу проекту;
- image (text) – посилання на зображення;
- contract_id (integer) – foreign key контракту, до якого відноситься проект.

Таблиця Співробітників проекту (ProjectStaff) має наступні атрибути:

- projectstaff_id (integer) – первинний ключ таблиці Співробітників проекту;
- project_id (integer) – foreign key проекту;
- profile_id (integer) – foreign key профілю співробітника;
- chief_id (integer) – foreign key керівника співробітника.

3 РОЗРОБКА WEB-ДОДАТКУ ПІДТРИМКИ ДІЯЛЬНОСТІ КЕРІВНИКА ПРОЕКТНОЇ ОРГАНІЗАЦІЇ У КЗАПР

3.1 Архітектура web-додатку

Процес створення web-додатку починається з реалізації схеми його архітектури. Для майбутнього програмного продукту вона містить компоненти, які будуть використовуватися під час розробки. Web-додаток складається з клієнту, який надсилає запити на сервер; із серверу, який приймає та опрацьовує запит, та надсилає відповідь клієнту, а також із бази даних, яка надає інформацію для відповіді клієнту [21].

Схема архітектури даного програмного продукту зображена на рисунку 3.1

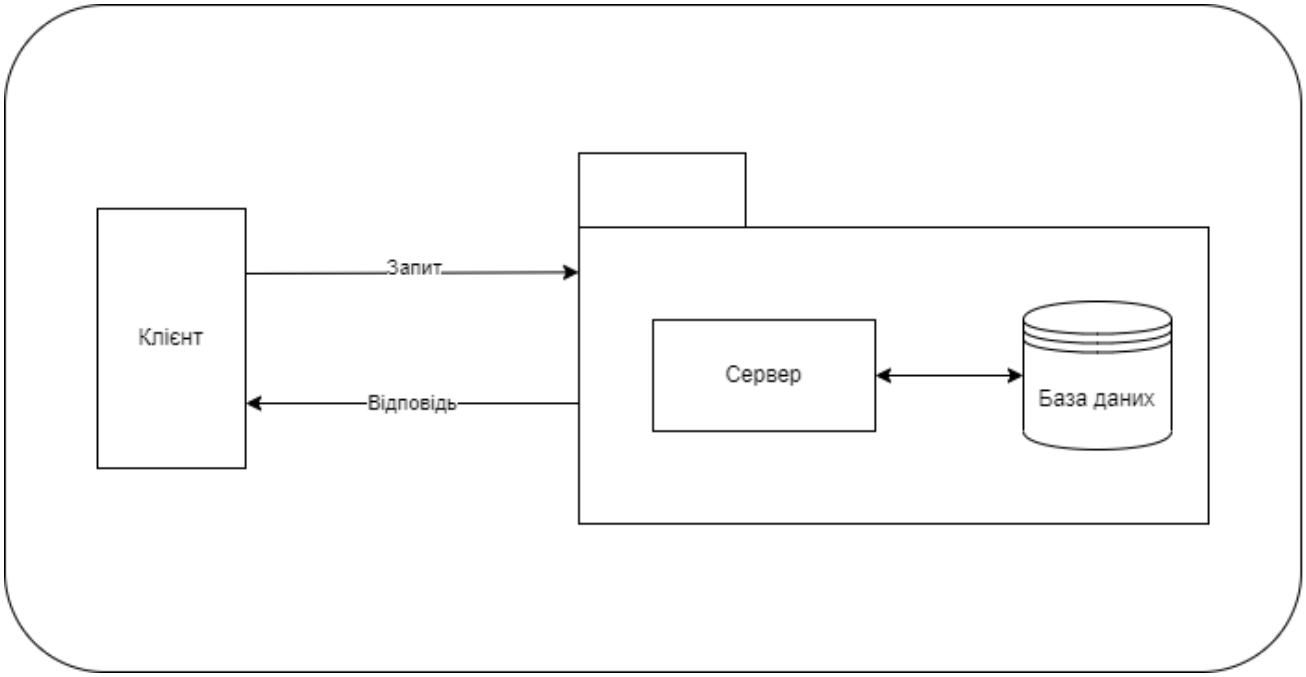


Рисунок 3.1 – Схема архітектури додатку

3.2 Розробка дизайну web-додатку

Після побудування схеми архітектури web-додатку було розпочато процес проектування макету дизайну. Для цього необхідно визначитися з структурою сторінок, розміщенням блоків в ньому, а також з кольоровим. Макет був розроблений, орієнтуючись на вимоги до програмного продукту, які описані у додатку А (рис. 3.2).

Рисунок 3.2 – Макет web-додатку

У результаті аналізу подібних систем управління проектами та персоналом компанії було розроблено дизайн сторінки керівника компанії, який зображений на рисунку 3.3, дизайн сторінки з контрактами (рис. 3.4) і дизайн сторінки з деталями контракту (рис. 3.5).

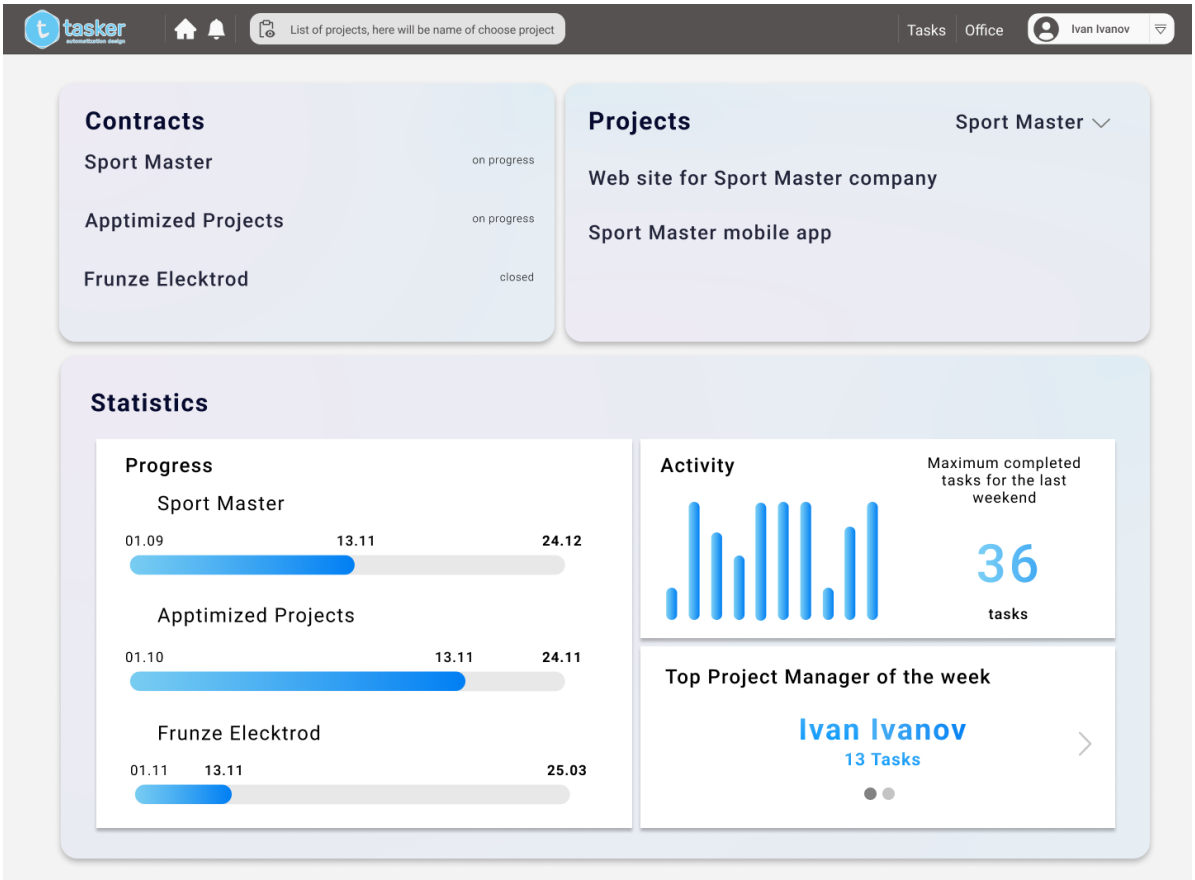


Рисунок 3.3 – Дизайн сторінки керівника компанії

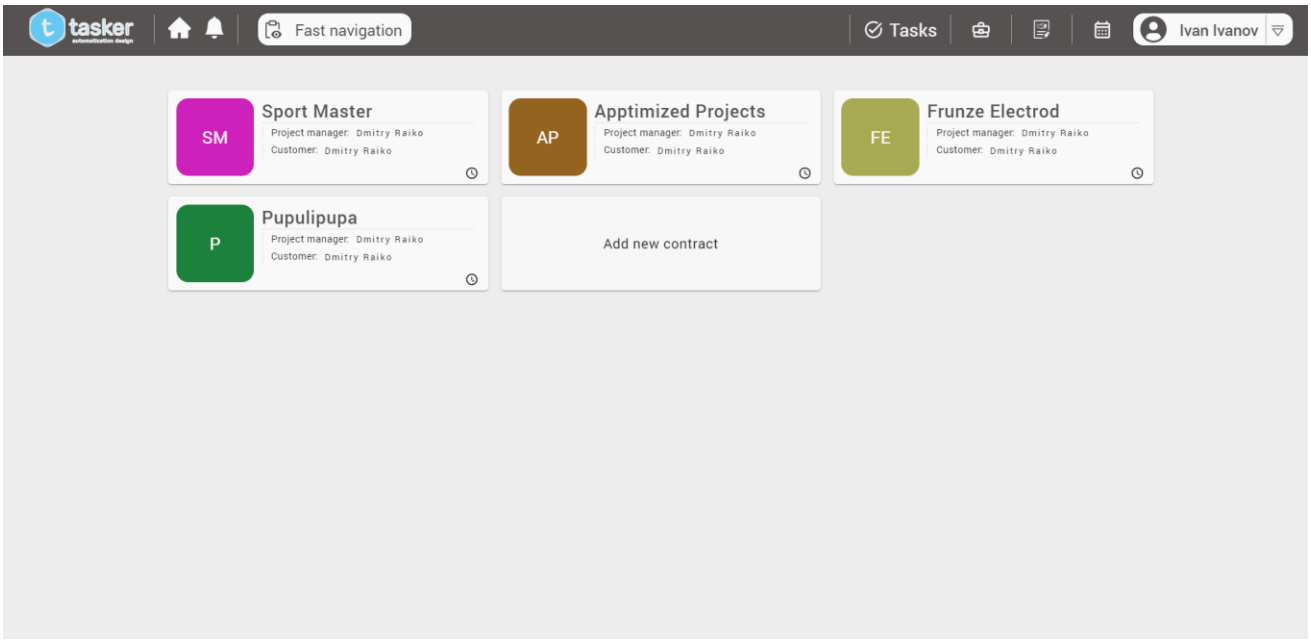


Рисунок 3.4 – Дизайн сторінки з контрактами

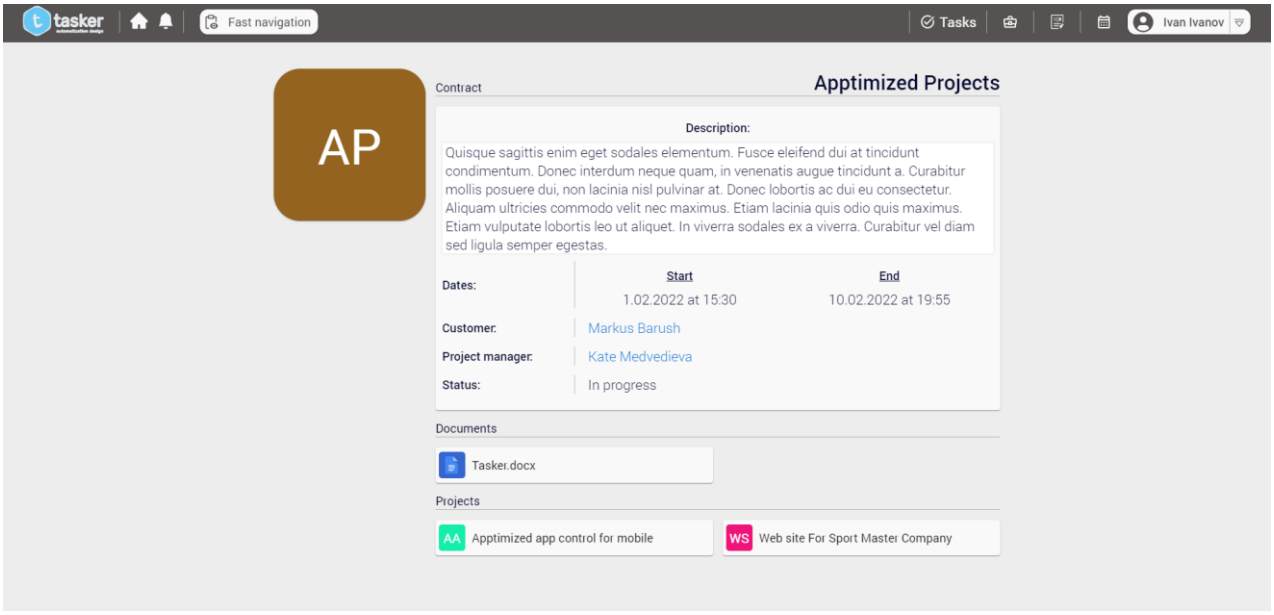


Рисунок 3.5 – Дизайн сторінки контракту

Для впізнання даного web-додатку було розроблено його логотип, використовуючи білий та блакитний кольори (рис. 3.6).

Рисунок 3.6 – Логотип web-додатку «Tasker»

Розташування блоків на web-сторінках було реалізовано за допомогою бібліотеки ReactJS та MUI. Також використовувалися каскадні таблиці стилів CSS, які були інтерпретовані за допомогою SASS. Розташування блоків контрактів на головній сторінці представлено на рисунку 3.7.

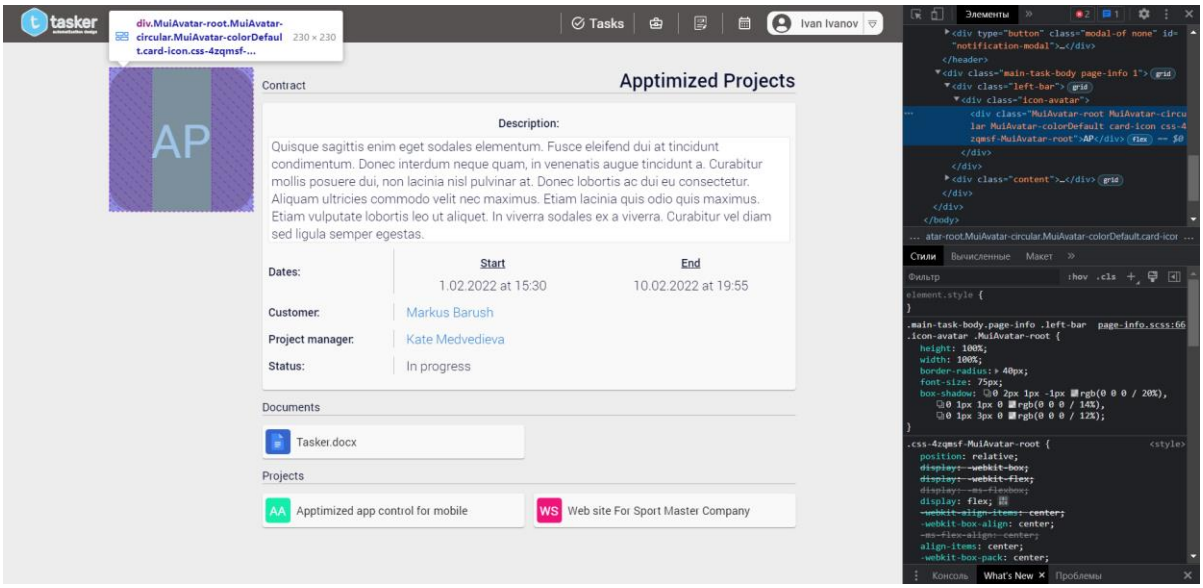


Рисунок 3.7 – Розташування блоків за допомогою бібліотеки MUI

3.3 Програмна реалізація web-додатку

Процес впровадження програмного забезпечення починається з підготовки середовища розробки. На персональному комп’ютері необхідно створити папку, в якій буде зберігатися майбутній проект, відкрити консоль, вказати шлях до директорії, увести команду для встановлення ReactJS на локальну машину – npx create-react-app «Tasker». Після ініціюємо клонування репозиторію на GitHub, увівши команду git init. На локальну машину встановлюємо інтерпретатор SASS для каскадних таблиць стилів. Для написання програмного коду було обрано IDE Visual Studio Code [22-23].

Головна сторінка web-додатку для підтримки діяльності керівника компанії містить перелік контрактів, проектів та статистику. Компонент для відображення контрактів представлено на рисунку 3.8.

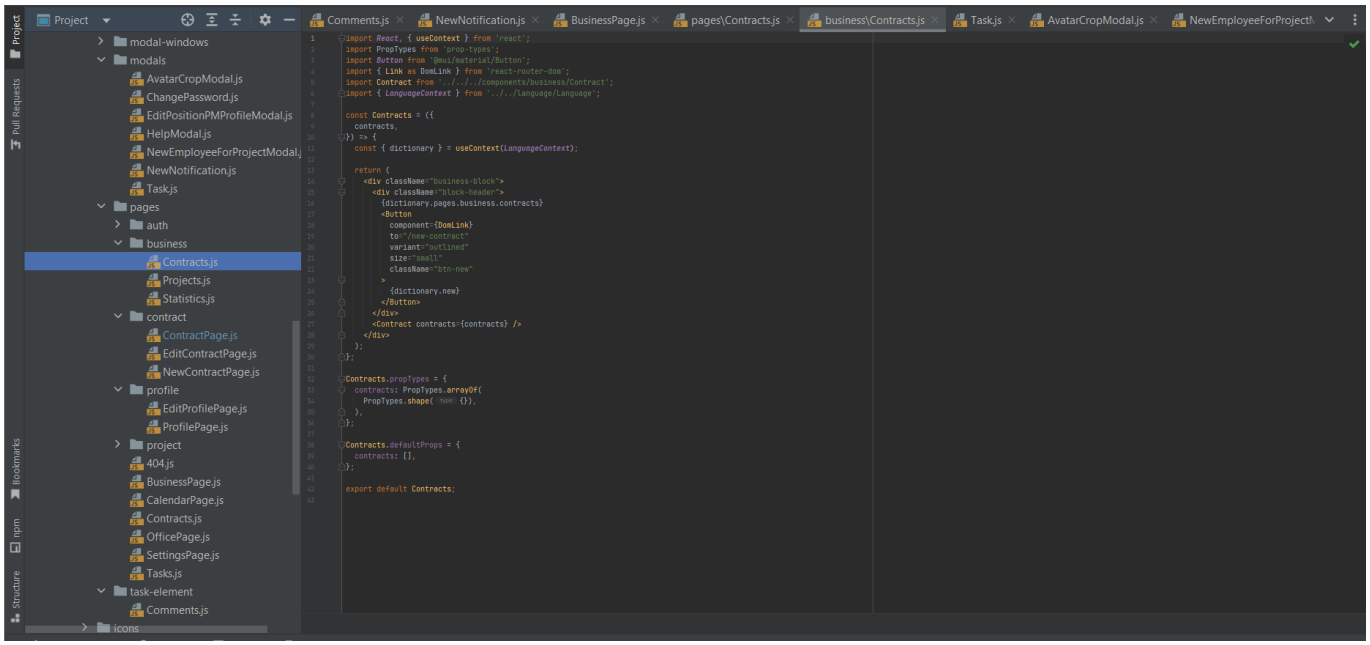


Рисунок 3.8 – Компонент для зображення контрактів
на сторінці керівника

Компонент для відображення проектів на сторінці керівника компанії надано на рисунку 3.9.

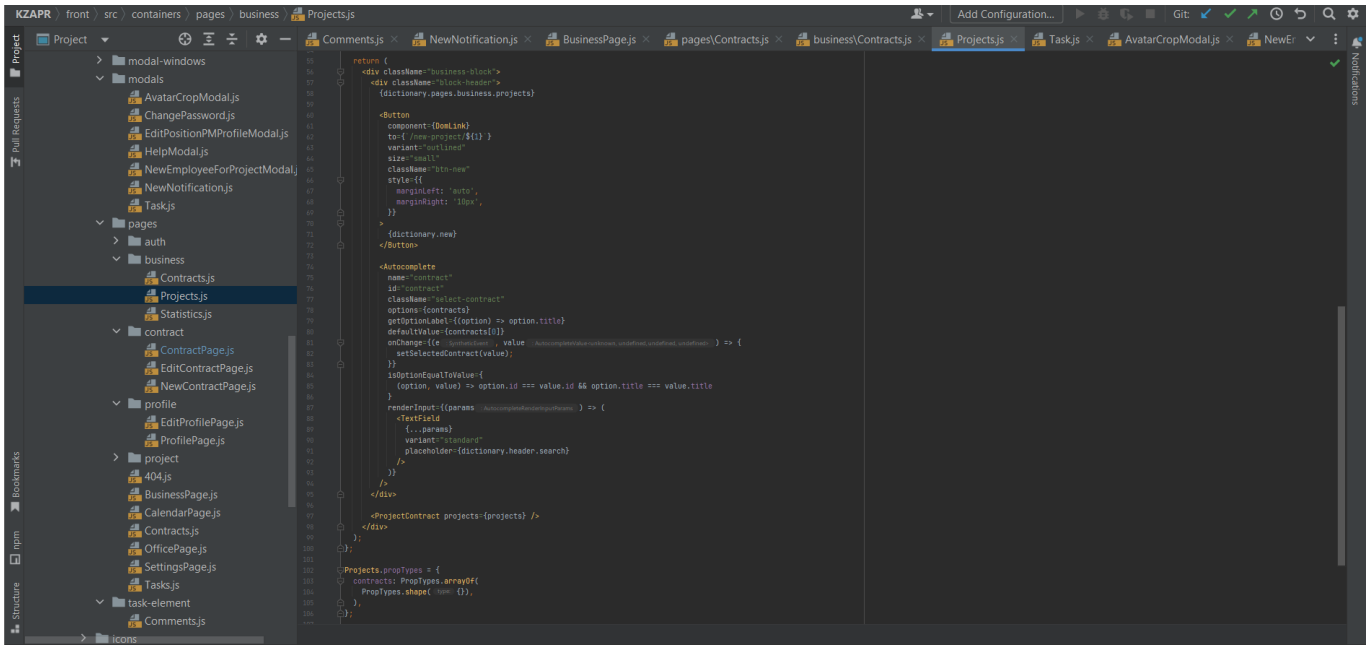


Рисунок 3.9 – Компонент для зображення проектів
на сторінці керівника компанії

Для додавання нового контракту необхідно заповнити форму, код якої можна переглянути на рисунку 3.10.

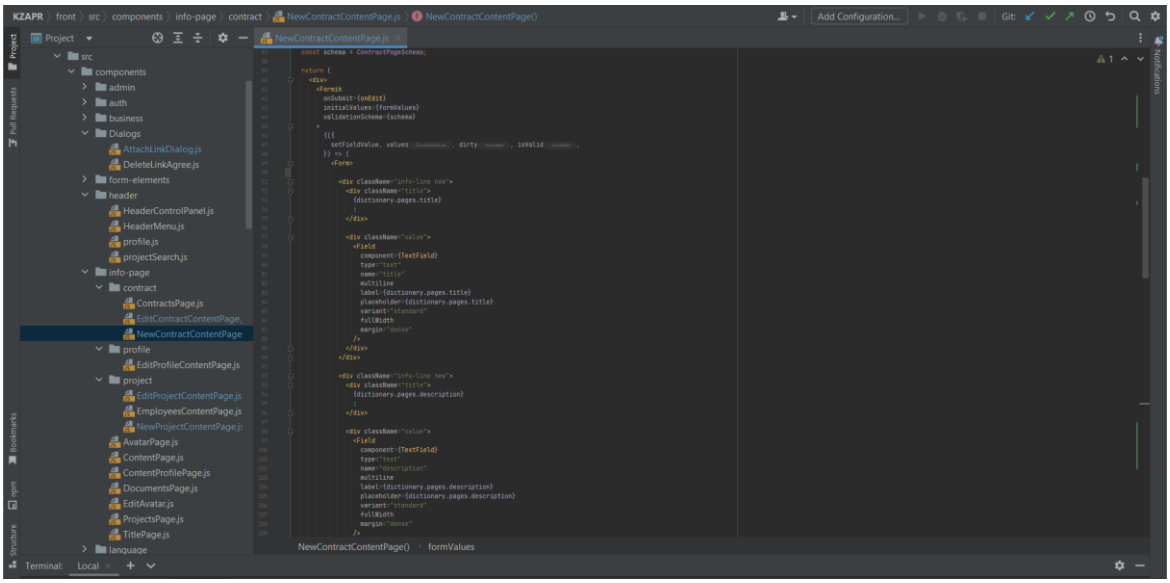


Рисунок 3.10 – Форма для створення нового контракту

Для перегляду самого контракту був розроблений компонент ContractPage (рис. 3.11).

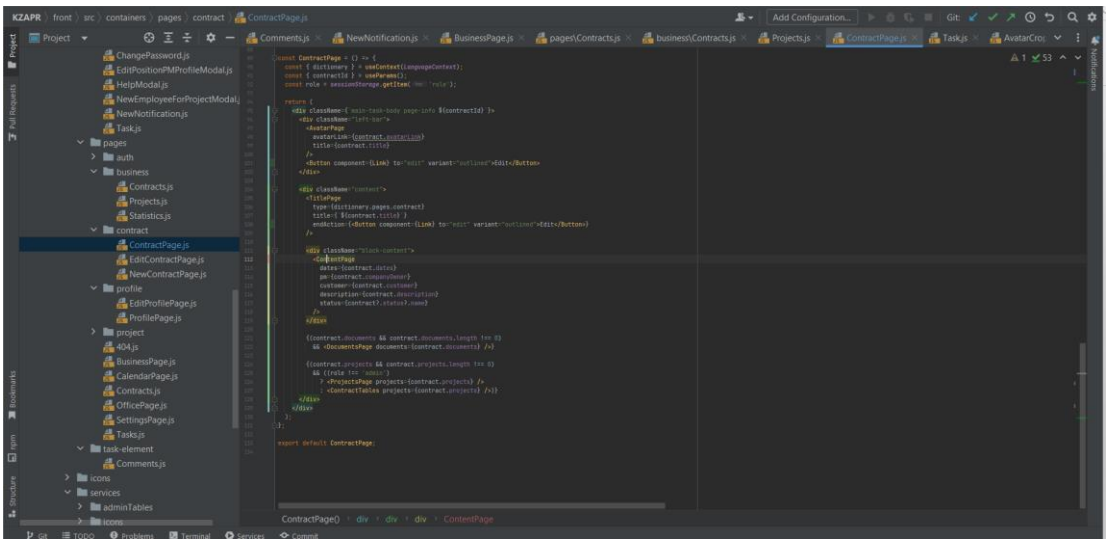


Рисунок 3.11 – Компонент ContractPage

Також була розроблена сторінка з співробітниками проекту, яка відображена на рисунку 3.12.

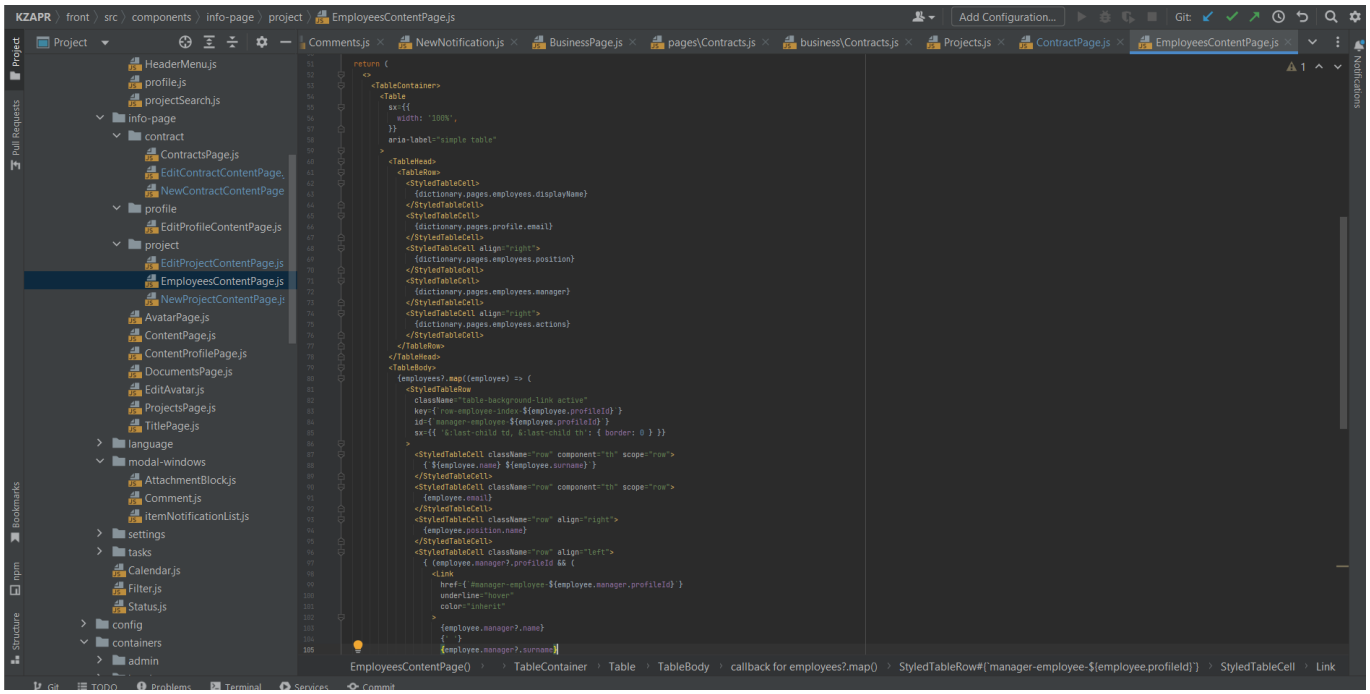


Рисунок 3.12 – Сторінка співробітників конкретного проекту

Програмний код основних модулів web-додатку підтримки діяльності керівника проектної організації у КЗАПР описаний у Додатку Г.

3.4 Демонстрація роботи web-додатку

Головна сторінка web-додатку для підтримки діяльності керівника проектної організації складається з п’яти частин. Це шапка з посиланням на сторінку з завданнями, на головну сторінку для керівника компанії, на календар та пошук по КЗАПР; блок з контрактами та їх статусом; проекти, які відносяться до певного контракту; блок із прогресом виконання проектів, статистикою активності виконання завдань співробітниками та блок з топ працівниками місяця або тижня (рис. 3.13).

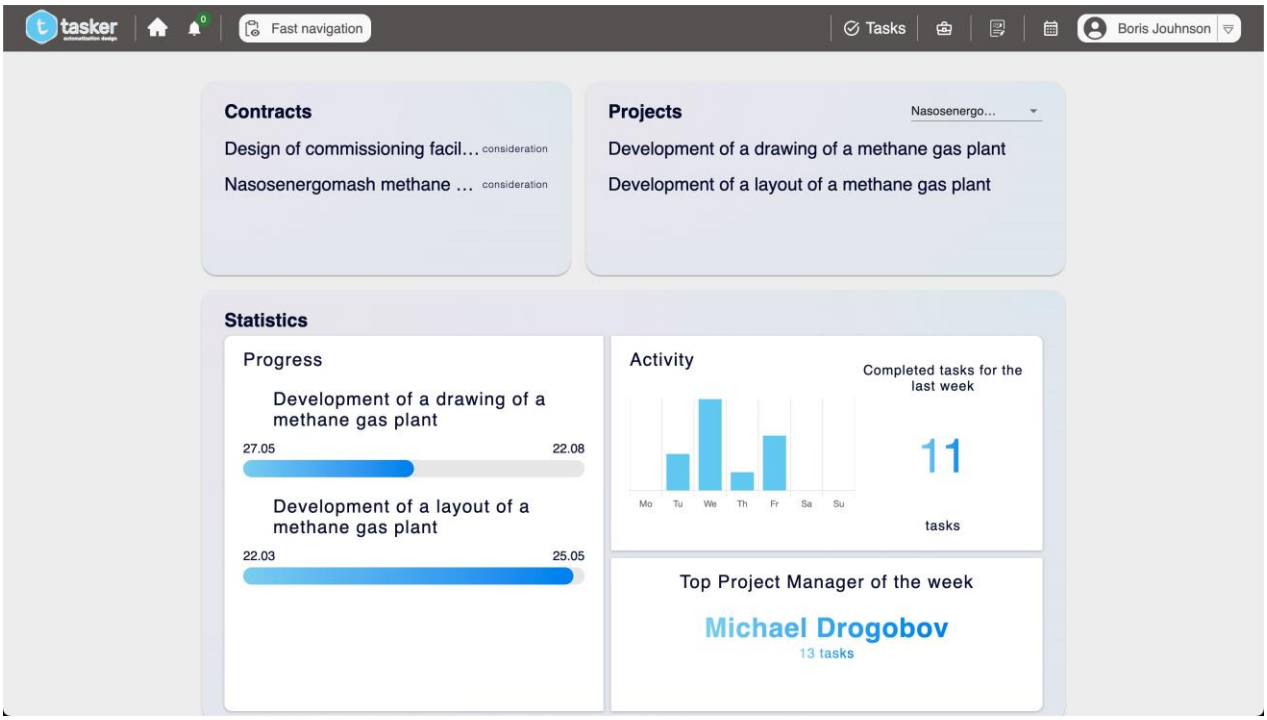


Рисунок 3.13 – Головна сторінка керівника компанії

На сторінці керівника компанії доступна функція додання нового контракту за допомогою кнопки «New» (рис. 3.14). Після цього відкривається відповідна форма. Вона містить такі поля, як назва та опис контракту, його часові рамки, поле для прикріплення технічного завдання, з ким підписується контракт та статус контракту (рис. 3.15).

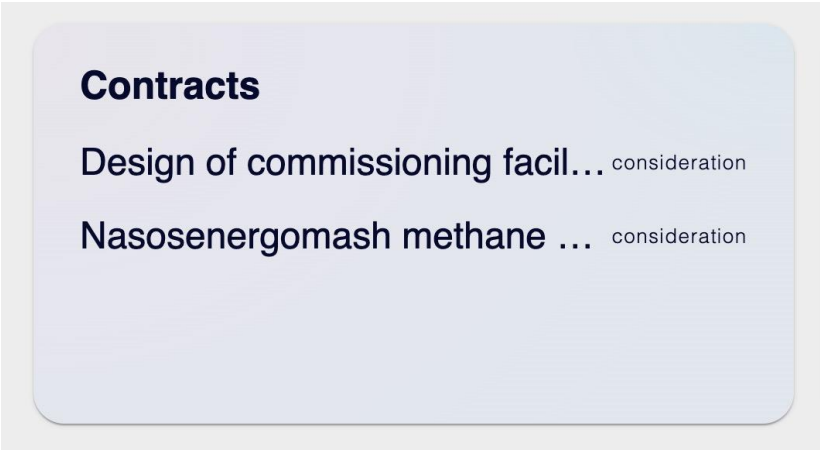


Рисунок 3.14 – Блок з контрактами з можливістю додавання нового

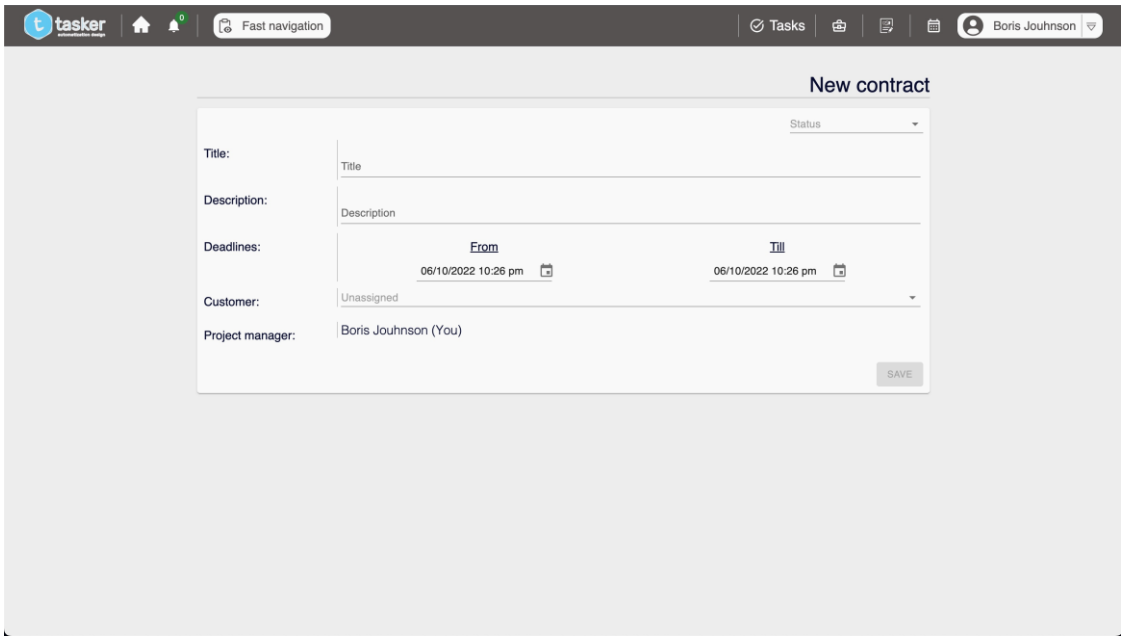


Рисунок 3.15 – Форма створення контракту

Також доступна функція створення проекту. Вона знаходиться у відповідному блоці. Для ініціалізації нового проекту необхідно обрати контракт, до якого він створюється, та натиснути на кнопку «New» (рис. 3.16). Після цього відбувається відкриття модального вікна. Саме воно надає можливість створення нового проекту. Поля ідентичні формі контракту. Різниця є тільки в полі «Contract». Воно відповідає за приналежність проекту до певного контракту (рис. 3.17).

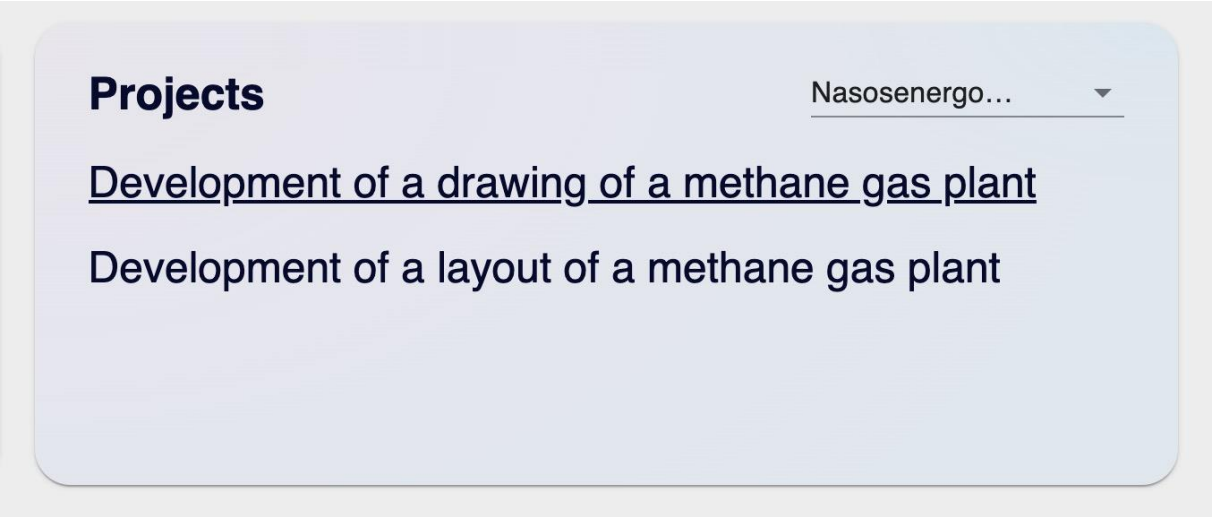


Рисунок 3.16 – Блок з проектами, який містить можливість додати новий проект до контракту

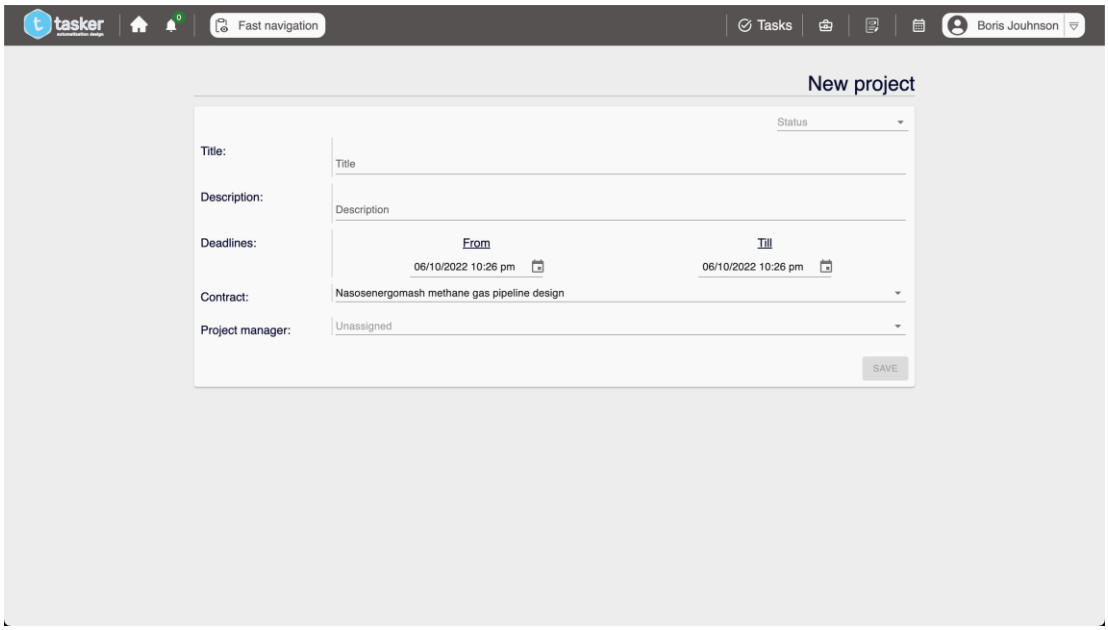


Рисунок 3.17 – Форма створення нового проекту

Для перегляду інформації про проект необхідно натиснути на його назву у блоці (рис. 3.16). Після відкривається відповідна сторінка (рис. 3.18). Вона містить логотип, назву, опис, дедлайни виконання, інформацію про відповідального за даний проект проектного менеджера, до якого контракту відноситься та який статус має.

Проект можна відредагувати, натиснувши на кнопку «Edit». Після цього відкриється сторінка для зміни контенту (рис. 3.19).

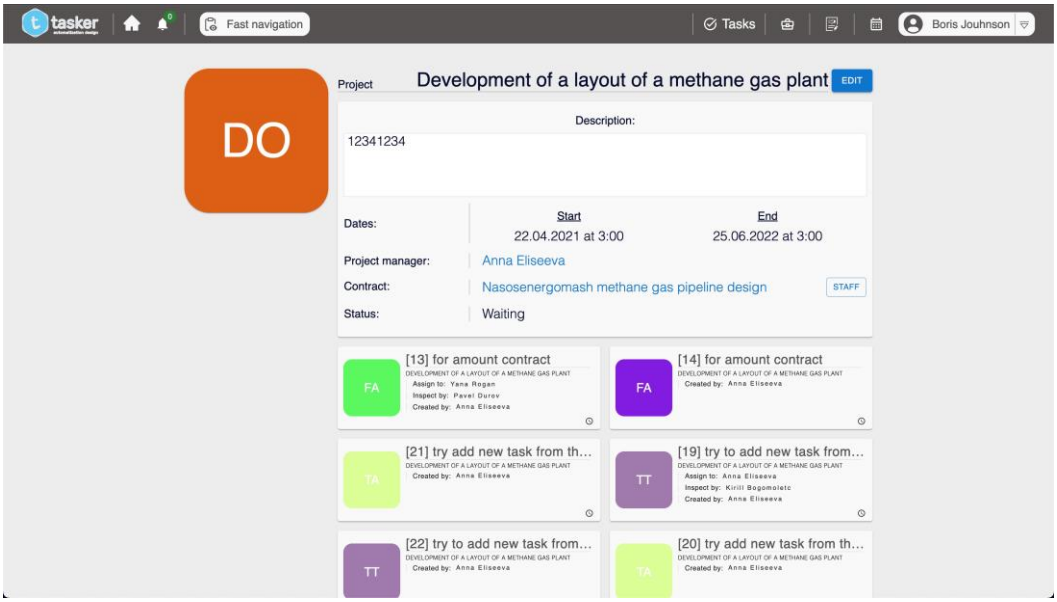


Рисунок 3.18 – Сторінка проекту

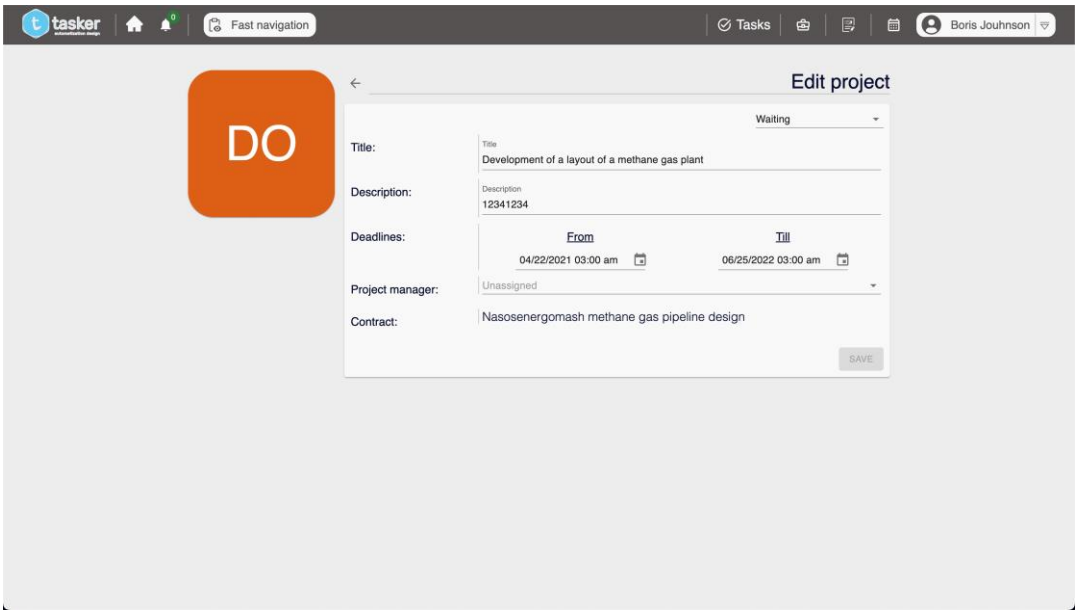


Рисунок 3.19 – Сторінка редагування проекту

Також на сторінці з проектом можна переглянути список учасників його виконання. Для цього необхідно натиснути на кнопку «Staff». Сторінка з командою проекту зображена на рисунку 3.20. Вона містить контактну інформацію співробітників, їх посаду та керівника. Дані можна редагувати у модальному вікні, натиснувши на кнопку «Edit» (рис. 3.21). За допомогою цієї можливості вносяться зміни щодо позицій персоналу та їх керівника.

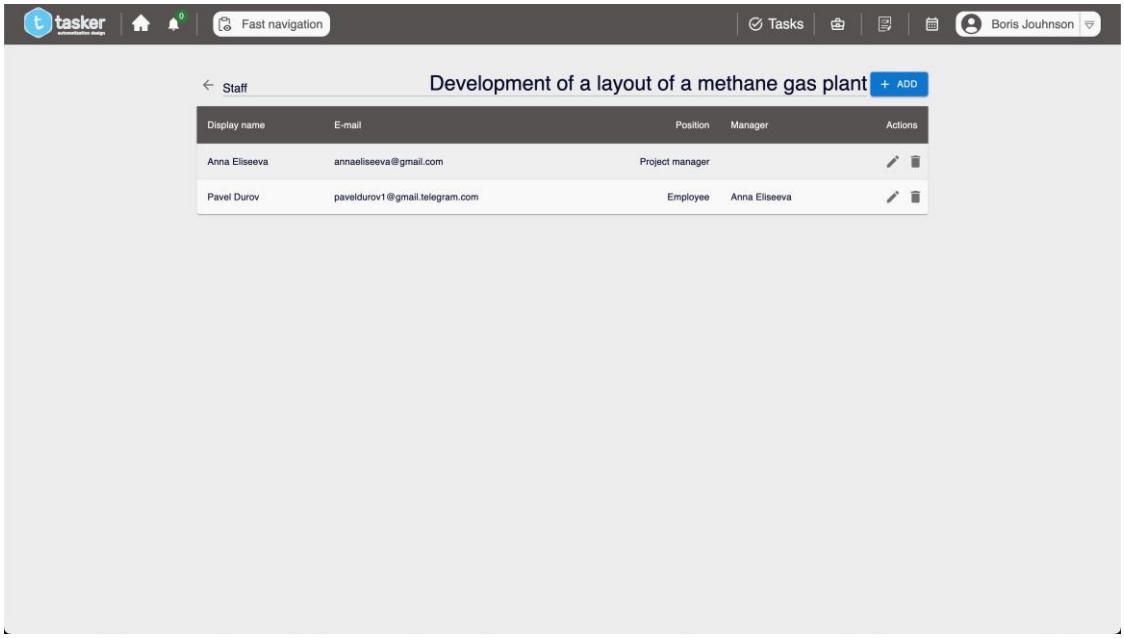


Рисунок 3.20 – Сторінка команди проекту

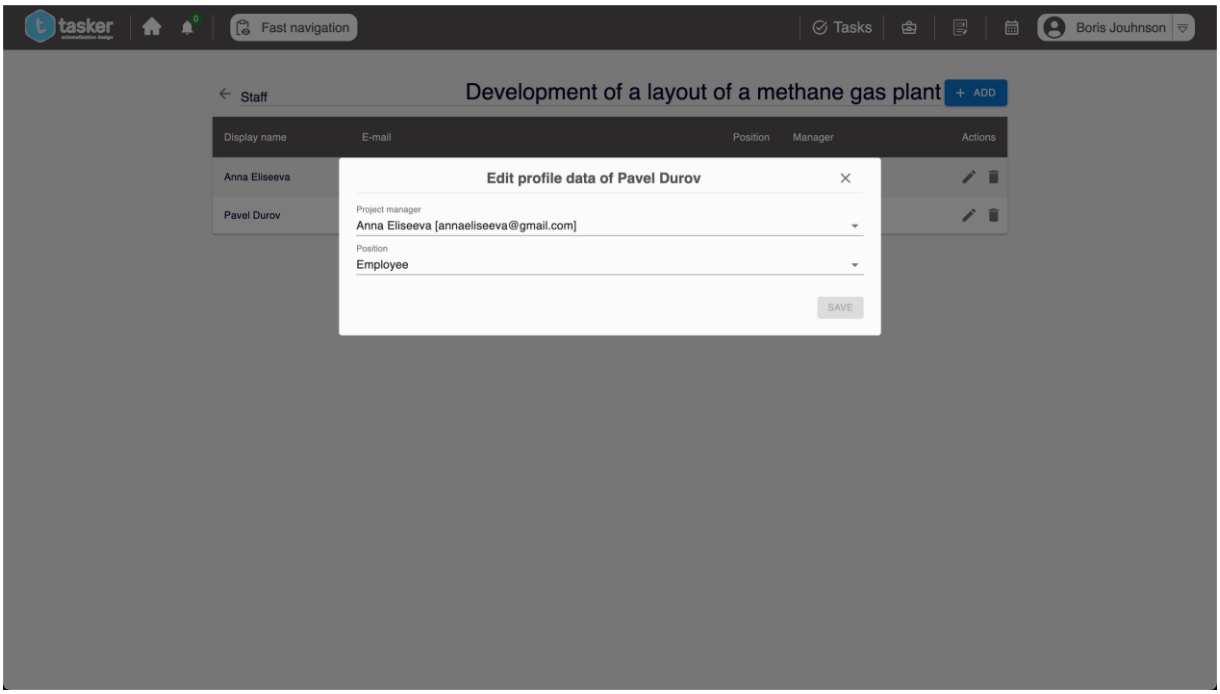


Рисунок 3.21 – Модальне вікно редагування персоналу

Також для керівника доступна функція додавання членів команди до проекту. Для цього необхідно натиснути на кнопку «Add». У доступному модальному вікні треба обрати з випадаючого списку працівників компанії, посаду співробітника, його керівника, також доступна можливість запросити співробітника через електронну пошту (рис. 3.22).

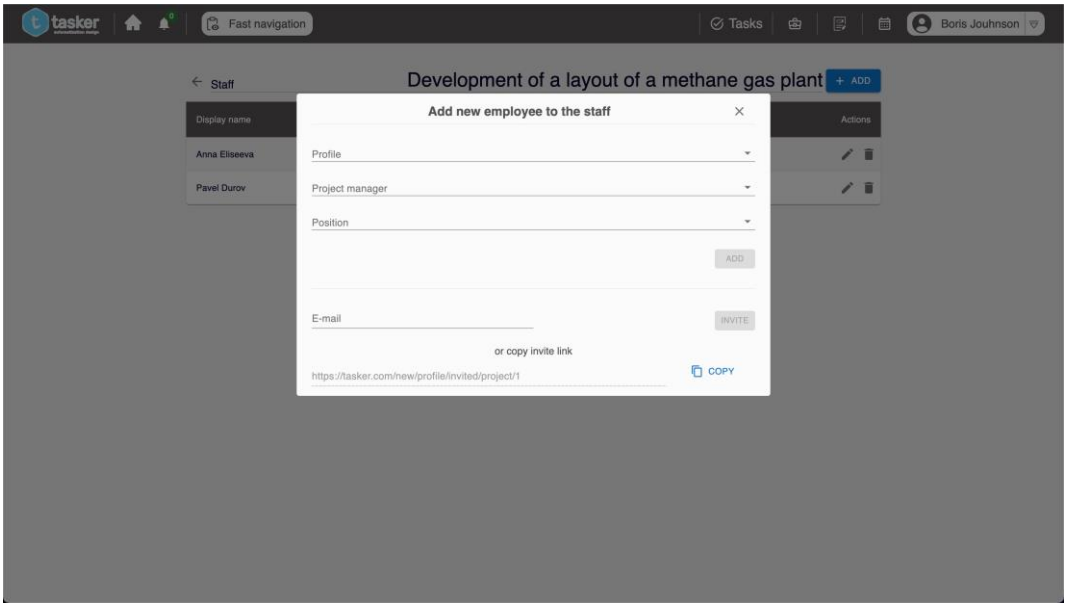


Рисунок 3.22 – Модальне вікно додання нового співробітника

Блок «Progress» демонструє стан залишку часу до дедлайну здачі проектів (рис. 3.23). Блок із «Activity» містить інформацію про максимальну кількість завдань виконаних за тиждень. Також є доступною певна гістограма. Вона відображає кількість завершених завдань за певний день (рис. 3.24). Блок «Top project manager» надає інформацію про найактивнішого проектного менеджера, під чим керівництвом було виконано найбільше задач за тиждень або місяць (рис. 3.25).

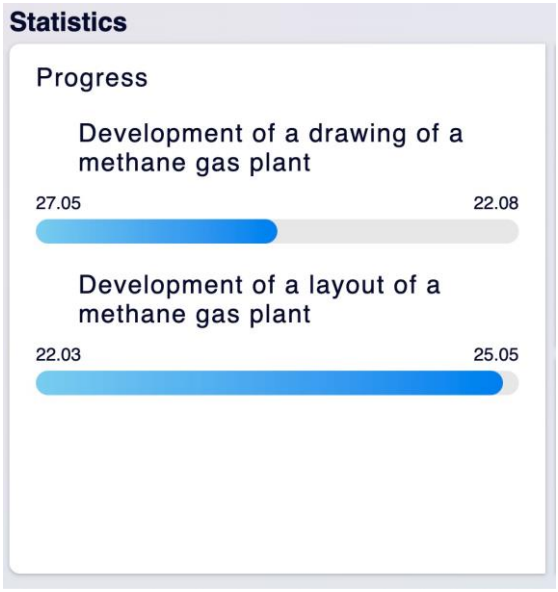


Рисунок 3.23 – Блок «Progress»

Рисунок 3.24 – Блок «Activity»

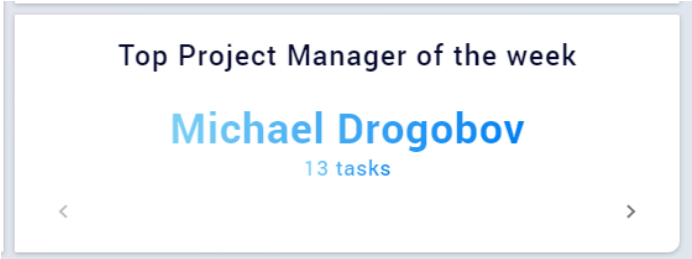


Рисунок 3.25 – Блок «Top project manager»

Наступною є сторінка присвячена контрактам. Вона містить їх повний список та кнопку для додавання нових (рис. 3.26). Якщо натиснути на кнопку «Add new contract» відкриється модальне вікно. Воно зображене на рисунку 3.15.

Для перегляду повної інформації про контракт необхідно натиснути на відповідний блок. Після цього відкривається сторінка з повним комплектом даних. А саме надається інформація про пов’язані з контрактом проекти, документи, які прикріплені до нього, а також статус, терміни виконання та замовник (рис. 3.27). Також на цій сторінці доступна функція редагування. Для цього треба натиснути на кнопку «Edit». Сторінка редагування контракту представлена на рисунку 3.28.

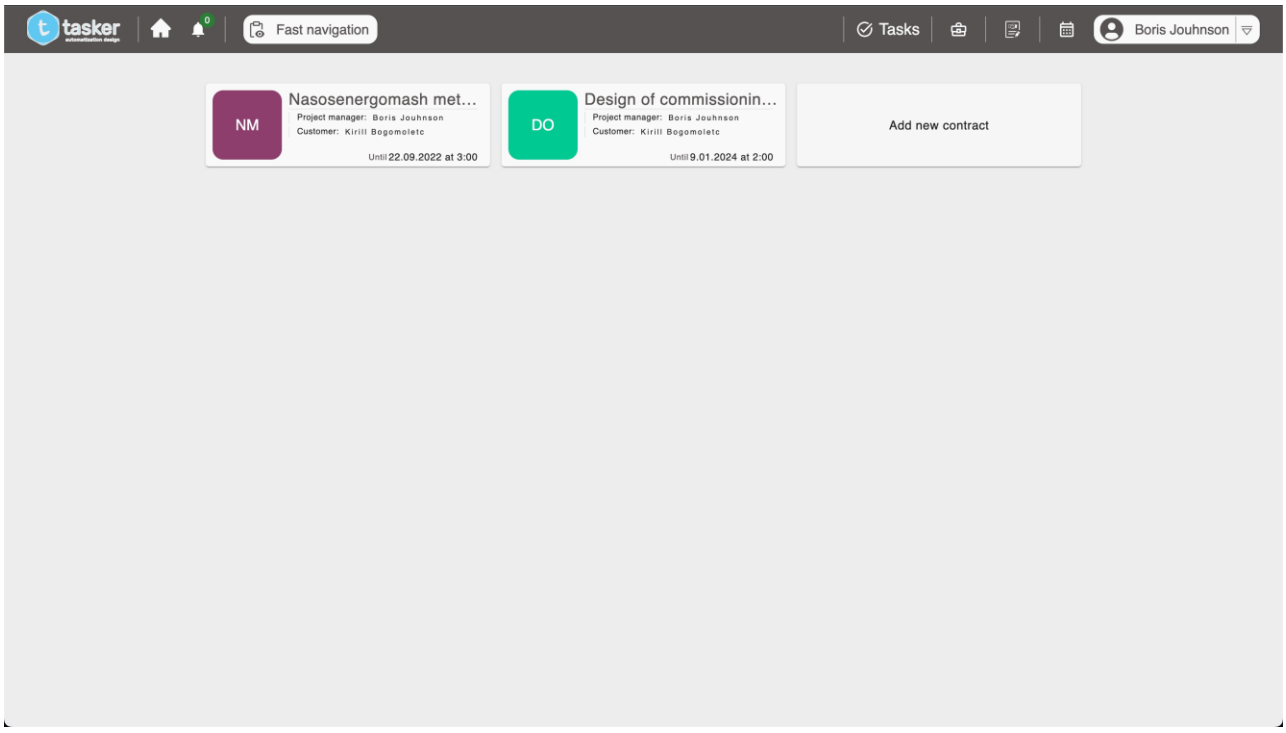


Рисунок 3.26 – Сторінка з контрактами

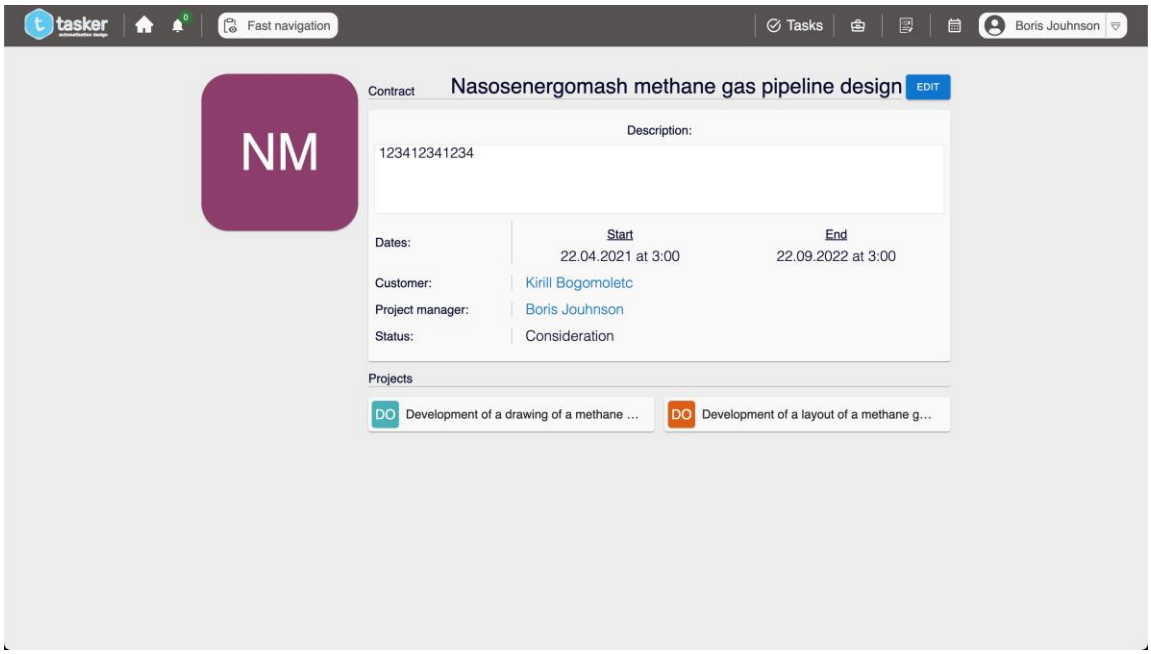


Рисунок 3.27 – Сторінка певного контракту

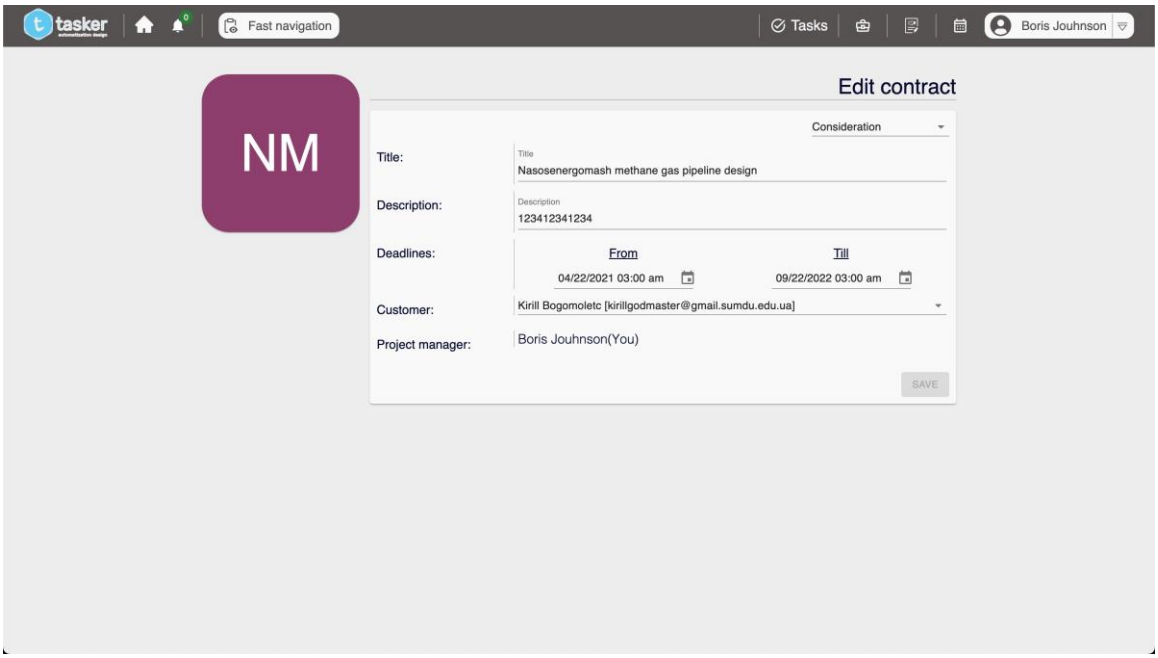


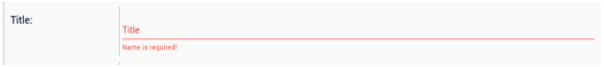
Рисунок 3.28 – Сторінка редагування відповідного контракту

3.5 Тестування web-додатку

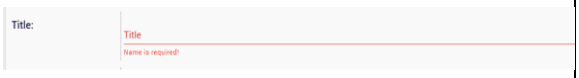
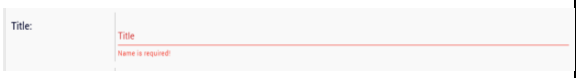
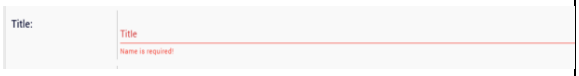
Для перевірки працездатності функцій web-додатку, які були реалізовані, необхідно провести функціональне тестування. Це включає в себе верифікацію вхідних даних форм створення та редагування контрактів та проектів. Також треба протестувати гіперпосилання та основний функціонал web-додатку.

Перш за все необхідно було перевірити роботу посилань у меню («Контракти» і «Сторінка керівника компанії») на те, чи відкривають вони правильні сторінки. Під час виконання даного процесу продемонстровано повну та коректну інформація. Проекти та контракти успішно додаються. Також вони повноцінно відображаються у списках на відповідних сторінках. Співробітники успішно додаються до команди того чи іншого проекту. Тестування роботи форм web-додатку було виконано за допомогою методу «black box». Це означає перевірку працездатності даної розробки без доступу до структури та коду [24]. Результати проведення тестування надано в таблиці 3.1.

Таблиця 3.1 – Тестування форм на валідні та невалідні дані

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
1	Перевірка форми створення проекту введенням коректних даних	Проект успішно створений.	Створений проект відображається у списку проектів на сторінці керівника компанії.	1
2	Перевірка форми створення проекту введенням некоректних даних	Не активна кнопка «Save», поле підсвічене червоним кольором та відображається текст повідомлення щодо неправильного введення даних.		1

Продовження таблиці 3.1.

3	Перевірка форми створення контракту введенням коректних даних	Контракт успішно створено.	Новий контракт відображається на сторінці контрактів та на сторінці керівника компанії у блоці «Contract».	1
4	Перевірка форми створення контракту введенням некоректних даних	Не активна кнопка «Save», поле підсвічене червоним кольором та відображається текст повідомлення щодо неправильного введення даних.		1
5	Перевірка форми редагування проекту введенням коректних даних	Проект успішно оновлено.	Оновлені дані проекту відображаються на сторінці проекту.	1
6	Перевірка форми редагування проекту введенням некоректних даних	Не активна кнопка «Save», поле підсвічене червоним кольором та відображається текст повідомлення щодо неправильного введення даних.		1
7	Перевірка форми редагування контракту введенням коректних даних	Контракт успішно оновлено.	Оновлені дані контракту відображаються на сторінці контракту.	1
8	Перевірка форми редагування контракту введенням некоректних даних	Не активна кнопка «Save», поле підсвічене червоним кольором та відображається текст повідомлення щодо неправильного введення даних		1

У результаті проведеного тестування не виявлено блокуючих дефектів. Увесь функціонал працює коректно.

ВИСНОВОК

Результатом виконання кваліфікаційної роботи бакалавра є розроблений web-додаток підтримки діяльності керівника проектної організації у КЗАПР.

Під час реалізації дипломного проекту було проведено дослідження та аналіз предметної області web-додатку підтримки діяльності керівника проектної організації у КЗАПР. Також проаналізовано його типові аналоги та здійснено огляд публікацій. У результаті визначено мету та задачі на реалізацію проекту «Web-додаток підтримки діяльності керівника проектної організації у КЗАПР». Зазначені засоби його створення. Складено відповідне технічне завдання (Додаток А) та виконано планування робіт даного проекту з дослідженням ризиків, які можуть виникнути під час розробки web-додатку (Додаток Б).

Другий розділ даної роботи було присвячено структурно-функціональному моделюванню та створенню use-case діаграми даного програмного продукту. У результаті проектування бази даних було виділено її сутності та їх атрибути.

У практичній частині дипломної роботи було розроблено схему архітектури web-додатку. Також виконано опис дизайну даного продукту та здійснено його програмну реалізацію. Продемонстровано роботу основного функціоналу. Тестування web-додатку для підтримки діяльності керівника проектної організації у КЗАПР пройшло успішно та не виявило жодних багів.

Використання даного програмного продукту дозволить керівнику проектної організації контролювати виконання завдань, автоматизувати підпис контрактів та створення проектів, а також дозволить проаналізувати досягнення співробітників за допомогою статистики.

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

Лістинг функціональних модулів реалізованого web-додатку розміщено у Додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web application to support the activities of the head of the designing organization in KZAPR // Матеріали та програма МІЖНАРОДНОЇ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ студентів та молодих учених / – Суми: МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ МІЖНАРОДНИЙ УНІВЕРСИТЕТ «АСТАНА», 2022. – С. 82.

2. Модернізація систем автоматизації управління підприємствами машинобудування [Електронний ресурс] – Режим доступу до ресурсу: http://phd.znu.edu.ua/page/dis/08_2019/dis_Nechepurenko.pdf.

3. Система автоматизованого проектування і розрахунку [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Система_автоматизованого_проєктування_і_розрахунку.

4. AutoCAD [Електронний ресурс] – Режим доступу до ресурсу: https://www.autodesk.com/products/autocad/overview?term=1-YEAR&tab=subscription.

5. CATIA [Електронний ресурс] – Режим доступу до ресурсу: https://www.3ds.com/ru/produkty-i-uslugi/catia/.

6. SolidWorks [Електронний ресурс] – Режим доступу до ресурсу: https://www.solidworks.com/ru.

7. Створення автоматизованої системи управління персоналом [Електронний ресурс] – Режим доступу до ресурсу: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/1969/1Автореферат%20401%20Щекотов%20Роман%20Вікторович.pdf.

8. MeisterTask [Електронний ресурс] – Режим доступу до ресурсу: https://www.meistertask.com.

9. Material Design [Електронний ресурс] – Режим доступу до ресурсу: https://material.io/design.

10. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com.

11. Що таке UX і чому користувацький досвід є важливим [Електронний ресурс] – Режим доступу до ресурсу:https://goldwebsolutions.com/uk/blog/shho-take-ux/

12. Asana [Електронний ресурс] – Режим доступу до ресурсу: https://app.asana.com.

13. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.javascript.com.

14. What is React? [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org.

15. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/Style/CSS/Overview.en.html.

16. Sass [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com.

17. MUI [Електронний ресурс] – Режим доступу до ресурсу: https://mui.com.

18. PostgreSQL: The World`s Most Advanced Open Source Relational Database [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/.

19. IDEF0 діаграма: приклади і правила побудови [Електронний ресурс] – Режим доступу до ресурсу: https://ukr.kagutech.com/3929706-idef0-diagram-examples-and-construction-rules.

20. Діаграми UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/uml-diagrams.html.

21. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://buklib.net/books/23148/.

22. What is and integrated development environment? [Електронний ресурс] – Режим доступу до ресурсу: https://www.veracode.com/security/integrated-development-environment.

23. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/.

24. Тестування методом чорної скриньки [Електронний ресурс] – Режим доступу до ресурсу: https://uk.myservername.com/black-box-testing-an-depth-tutorial-with-examples.

25. Управління проектами в механічній інженерії: практикум [Електронний ресурс] – Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/38184/1/Upravl_proekt_v_mekhan_inzhenerii.pdf.

26. Базові Git-команди [Електронний ресурс] – Режим доступу до ресурсу: https://vyspiansky.gitbook.io/introduction-to-web-development/git/git-commands

27. Моделі та інформаційні технології проектування і управління в складних системах [Електронний ресурс] – Режим доступу до ресурсу: https://essuir.sumdu.edu.ua/bitstream-download/123456789/84041/1/Kuznietsov_%201632.pdf.

28. Support of Project Management Methods by Project Management Information System [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S1877042815056803.

29. Сучасні методи управління проектами [Електронний ресурс] – Режим доступу до ресурсу: http://dspace.wunu.edu.ua/bitstream/316497/42997/Заставнюк.pdf.

30. Organization Breakdown Structure (OBS) [Електронний ресурс] – Режим доступу до ресурсу: https://uplandsoftware.com/psa/resources/glossary/organization-breakdown-structure-obs/.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

«Web-додаток підтримки діяльності керівника проектної організації у КЗАПР»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Антипенко В.П.

Студентка групи ІТ-82-0

_____ Медведєва К.С.

Суми 2022

1. Призначення й мета web-додатку для підтримки діяльності керівника проектної організації у КЗАПР

1.1 Призначення web-додатку

Web-додаток призначений для підтримки діяльності керівника проектної організації у КЗАПР.

1.2 Мета створення web-додатку

Головна мета проекту – це створення web-додатку для підтримки діяльності керівника проектної організації, використання якого забезпечить автоматизацію процесів заключення контрактів, створення проектів, а також контроль над діяльністю співробітників організації.

1.3 Цільова аудиторія

Цільовою аудиторією даного проекту є керівник проектної організації та потенційні партнери компанії.

2 Вимоги до проекту

2.1 Вимоги до проекту в цілому

2.1.1 Вимоги до структури й функціонування

Web-додаток для підтримки діяльності керівника проектної організації у КЗАПР повинен бути розроблений за допомогою web-інструментів та забезпечувати визначений набір функціоналу.

Кінцевий програмний продукт даного проекту має бути представлений web-додатком, який має відповідне та якісне інформаційне наповнення та графічні матеріали.

2.1.2 Вимоги до персоналу

Основною вимогою до користувачів є наявність навичок користування web-браузером та персональним комп’ютером.

2.1.3 Вимоги до збереження інформації

Уся інформація, що міститься у web-додатку, повинна бути збережена у базі даних, яка реалізована засобами системи управління базами даних PostgreSQL.

2.1.4 Вимоги до розмежування доступу

Web-додаток має обмежений доступ та реалізовується для внутрішнього використання в організаціях за допомогою підключення через мережу Інтернет.

Керівник підприємства назначає проектних менеджерів на проекти та створює їх, слідкує за статистикою, підписує контракти.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

На головній сторінці користувач має інформацію про завдання, в яких він відповідальний за виконання. Для отримання докладної інформації про завдання – необхідно натиснути на нього. Сторінка «Calendar» містить інформацію про завдання в проекті відсортовану за датою та часом виконання. Сторінка керівника організації містить статистику, список контрактів та проектів. Для створення проекту та контракту треба натиснути на відповідні кнопки та заповнити поля форми.

2.2.2 Навігаційне меню

Для зручної навігації між сторінками повинно бути передбачене меню, яке дозволить користувачу швидко переміщуватись по всім доступним сторінкам web-додатку. Меню має бути закріплене і розташовуватися у шапці сторінки.

2.2.3 Управління контентом

Управління контентом web-додатку здійснюється користувачем, в залежності від привілеїв. Інформаційне наповнення web-додатку повинно міститися у базі даних та на сервері.

2.2.4 Дизайн web-додатку

Дизайн web-додатку має бути виконаний у сучасному стилі. У кольоровій гаммі мають переважати блакитні відтінки, білий та сірий кольори.

Додаток має бути витриманий в одному стилі. Текст повинен мати один шрифт та бути комфортним для читання. Шаблон майбутнього програмного продукту зображено на рисунку А.1.

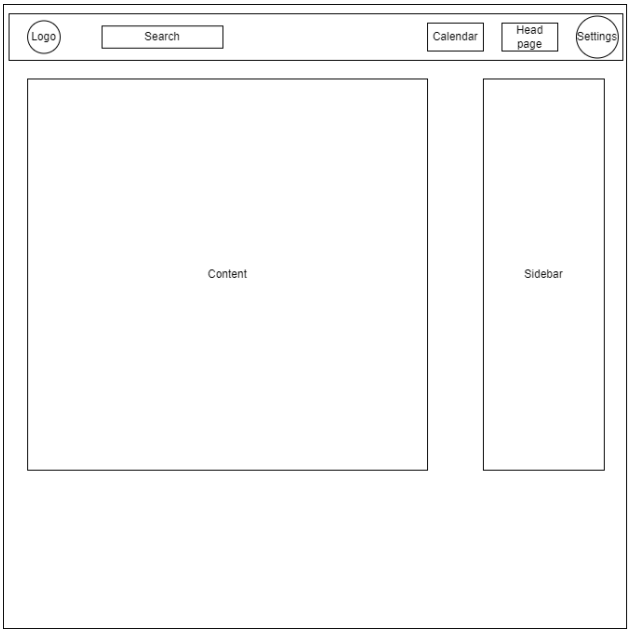


Рисунок А.1 – Схема головної сторінки

2.2.4 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.2.

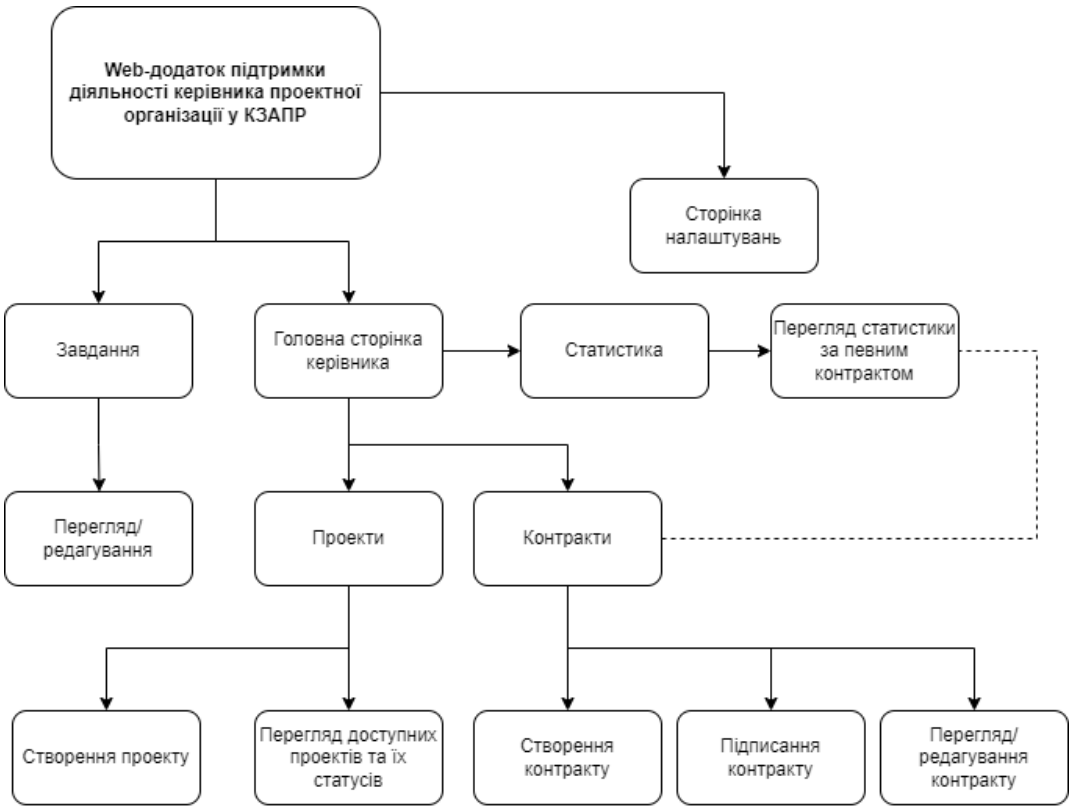


Рисунок А.2 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Весь текст у web-додатку має бути виконаний англійською мовою.

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи web-додатку web-браузер має бути Internet Explorer 11 і вище, або Safari 10 і вище, або Opera 10 і вище, або Firefox 5 і вище, або Chrome 2 і вище, або Microsoft Edge.

2.4 Вимоги до функціонування системи

2.4.1 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, видалення, редагування контракту	Адміністратор, Керівник
UN-02	Створення, видалення, редагування проекту	Адміністратор, Керівник
UN-03	Додавання, редагування, видалення завдань	Проектний менеджер, Керівник
UN-04	Назначення співробітника на виконання завдання	Проектний менеджер, Керівник
UN-05	Додавання, редагування ролі, видалення співробітника	Адміністратор
UN-06	Перегляд завдань в проекті	Адміністратор, Проектний менеджер, Керівник, Співробітник
UN-07	Оновлення статусу завдань	Співробітник, Керівник, Проектний менеджер
UN-08	Створення повідомлень для осіб, які виконують завдання	Проектний менеджер, Керівник
UN-09	Видалення застарілої інформації	Адміністратор
UN-10	Можливість звернення до підтримки	Усі користувачі
UN-11	Надання відповіді на звернення до підтримки	Адміністратор

2.4.2 Системні вимоги

Проаналізувавши потреби користувачів було визначено наступні вимоги:

- наявність реєстрації та авторизації користувачів.
- створення, редагування, видалення проектів та контрактів;
- підписання контракту;
- назначення проектного менеджера на проект;
- перегляд статистики за користувачами;
- перегляд статистики за виконанням контрактів;
- можливість налаштування профілю користувача.

3 Склад і зміст робіт зі створення web-додатку для підтримки керівника проектної організації у КЗАПР

Детальний опис етапів створення web-додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки
1	Створення шаблону web-додатку	2 дні
2	Створення верстки сторінок web-додатку	14 днів
3	Розробка модулю з проектами	10 днів
4	Розробка модулю з контрактами	10 днів
5	Розробка модулю з статистикою	7 днів
6	Розробка бази даних	7 днів
7	Додавання записів до бази даних	3 днів
8	Beta-тестування	4 днів
9	Alpha-тестування	4 днів

Продовження табл. А.2.

10	Розміщення на хостингу	1 день
11	Перевірка працездатності	2 дні
12	Написання супровідної документації	3 дні
13	Реліз web-додатку	2 день
	Загальна тривалість робіт	69 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

Попит у сфері організації роботи компанії, а саме комплексу засобів автоматизованого проектування задач, стрімко зріс за останні роки, у зв’язку з пандемією та міжнародними зв’язками в роботі компаній, які передбачають роботу людей на великій відстані. Також популярним стало управління власним часом, тому що життя людей стало більш насиченим, а розбиття свого повсякдення на задачі дозволяє економити час та швидше досягати своїх цілей.

Вдосконалення процесів управління компаніями та власним часом позитивно вплине не тільки на продуктивність роботи, а й на самопочуття людей, завдяки комфортному інтерфейсу та збільшенню вільного часу.

Деталізація мети методом SMART. Щоб проект був успішним та конкурентоспроможним треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. Його суть закладена в абревіатурі назви: конкретна (Specific), вимірювана (Measurable), досяжна (Achievable), актуальна (Relevant), обмежена у часі (Time-framed).

S – конкретність, специфічність. Створити web-додаток для підтримки діяльності керівника проектної операції у КЗАПР.

M – вимірюваність. Розробити web-додаток, функціонал якого дозволяв автоматизувати такі процеси, як підписання контрактів, відображення статистики виконання проектів, назначення відповідальних над виконанням завдань та створення проектів.

A – досяжність. Для розробки проекту потрібні знання HTML, CSS, мови програмування JavaScript, PHP, MySQL та навичок написання документації. Є ТЗ

на розробку даного проекту. Проаналізувавши доступні ресурсні можливості та обмеження можна зробити висновок, що мета є такою, яку можливо досягнути.

R – актуальність. Створюваний web-додаток підтримки діяльності керівника проектної організації у КЗАПР полегшить управління підприємством за рахунок автоматизації певних процесів.

T – обмеженість в часі. Ціль має часове обмеження. Термін досягнення мети проекту визначено за навчальним планом до червня 2022 року.

Планування змісту робіт. Розбиття роботи на менші завдання – це зручний метод підвищення продуктивності розробки, який використовується для того, щоб зробити роботу більш доступною та керованою. Для проектів ієрархічна структура робіт(WBS) є інструментом, який використовує цей метод і є одним з найважливіших документів з управління проектами, організації командної роботи, забезпечує ретельну оцінку термінів та контролю графіків роботи. Елементами декомпозиції можуть бути дані, продукти, послуги.

На першому рівні розміщений продукт проекту. На другому рівні декомпозиції розташовані основні дії та заходи, що забезпечують досягнення мети проекту. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними.

Елементарні роботи – це дії, які в результаті їх виконання мають однозначний чіткий результат, на які призначена одна особа, для якої можна обчислити тривалість виконання та витрати на працю. На рисунку Б.1 представлено WBS з розробки Web-додатку підтримки діяльності керівника проектної організації з КЗАПР.

Планування структури виконавців. Після декомпозиції процесів необхідно розробити організаційну структуру виконавців або OBS, яка визначається як графічна структура відповідальних осіб або учасників виконання робіт, які беруть участь у втіленні проекту.

Співробітники виступають у ролі відповідальних осіб, які відповідальні за виконання елементарної роботи та організацію роботи, що зазначена у WBS. Кожну елементарну роботу треба розглянути як окремий проект.

Планування структури організації. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення відповідальних осіб або осіб, які беруть участь у розробці проекту.

У ролі відповідальних осіб виступають співробітники, які відповідальні за організацію та виконання елементарної роботи, яка приведена у WBS. Елементарну роботу слід розглядати як окремий проект.

На рисунку Б.2 представлено організаційну структуру планування проекту. Список виконавців, що працюють над проектом описано в таблиці Б.1.

Таблиця Б.1 – Виконавці проекту

Роль	**Ім’я**	**Проектна роль**
Розробник	Медведєва К.С.	Виконує front-end та back-end розробку.
Проектувальник	Медведєва К.С.	Розроблює структуру web-додатку та виконує проектувальні роботи.
Тестувальники	Райко Д.І., Глуховцов Д.О.	Відповідальні за тестування функціоналу та дизайну web-додатку.
Керівник проекту	Антипенко В.П.	Делегує розробкою та складає завдання на розробку проекту.
Менеджер проекту	Медведєва К.С.	Відповідальна за виконання термінів, розподіл завдань та ресурсів між учасниками.

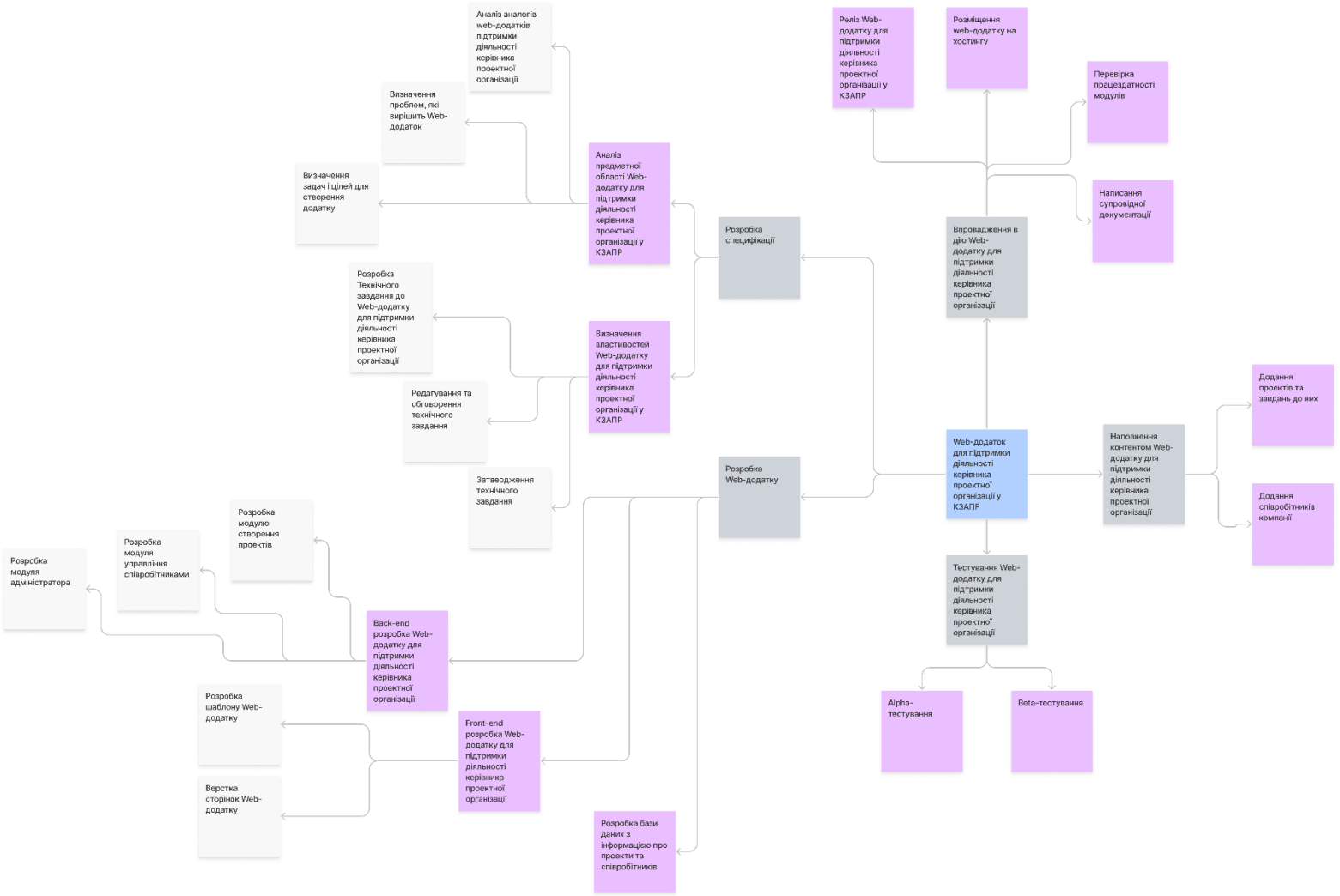


Рисунок Б.1 – WBS-структура робіт проекту

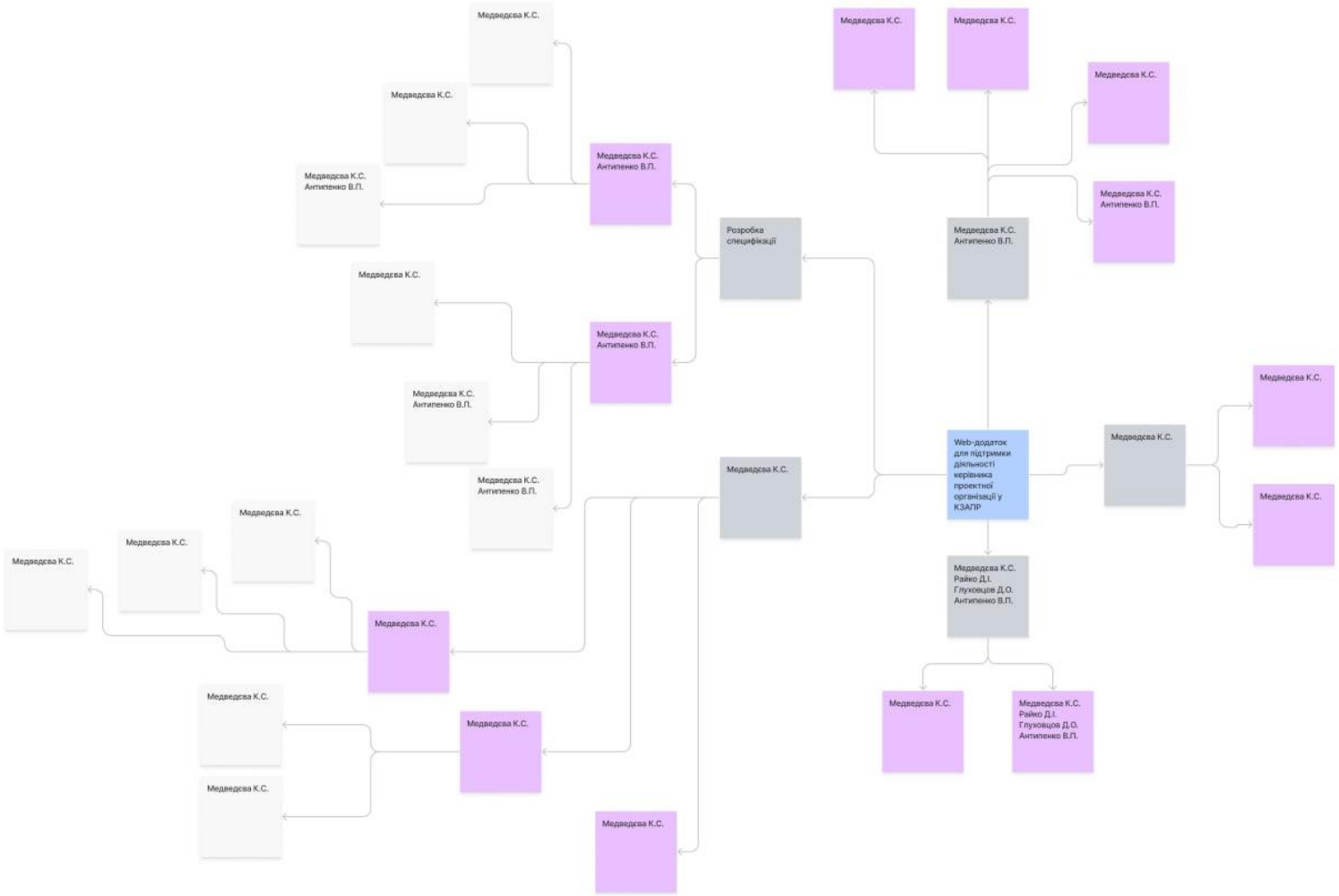


Рисунок Б.2 – OBS-структура робіт проекту

Діаграма Ганта. Діаграма Ганта – це діаграма, яка застосовується для демонстрації плану, графіка робіт за будь-яким проектом. Побудова діаграми Г

анта є одним з важливих етапів планування проекту, що виглядає як розклад виконання робіт з реальним розподілом дат. Цей розподіл дозволяє отримати уявлення про тривалість кожного з процесів, з урахуванням обмежень у ресурсах, вихідних та святкових днів.

Графічне представлення розподілу робіт даного проекту у часі представлено на рисунках Б.3-Б.6.

Web-додаток для підтримки діяльності керівника проектної організації у КЗАПР

	Имя	Кому назначена	Начало	Окончание	%...
1	Web-додаток для підтримки діяльності керівника проектної організації у КЗАПР		24.1.2022	10.6.2022	0
2	**Підготовка специфікації**		24.1.2022	4.2.2022	0
3	**Аналіз предметної області Web-додатку для підтримки діяльності керівника проектної організації у КЗАПР**		24.1.2022	31.1.2022	0
4	Визначення проблем, які вирішить Web-додаток		24.1.2022	25.1.2022	0
5	Визначення задач і цілей для створення додатку		26.1.2022	26.1.2022	0
6	Аналіз аналогів додатків для підтримки діяльності керівника проектної організації		27.1.2022	31.1.2022	0
7	**Визначення властивостей Web-додатку для підтримки діяльності керівника проектної організації у КЗАПР**		1.2.2022	4.2.2022	0
8	Розробка Технічного Завдання до Web-додатку для підтримки діяльності керівника проектної організації у...		1.2.2022	2.2.2022	0
9	Редагування та обговорення Технічного завдання		3.2.2022	3.2.2022	0
10	Затвердження Технічного Завдання		4.2.2022	4.2.2022	0
11	**Розробка Web-додатку**		7.2.2022	19.4.2022	0
12	**Розробка Front-End**		7.2.2022	24.2.2022	0
13	Розробка шаблону web-додатку		7.2.2022	15.2.2022	0
14	Верстка сторінок web-додатку		16.2.2022	24.2.2022	0
15	**Розробка Back-End**		25.2.2022	19.4.2022	0
16	Розробка модуля для створення задач/проектів		25.2.2022	16.3.2022	0
17	Розробка модуля для керування штатом компанії		17.3.2022	30.3.2022	0
18	Розробка модулю для керівника компанії		31.3.2022	19.4.2022	0
19	Розробка Бази даних з інформацією про проекти та працівників компанії		25.2.2022	10.3.2022	0
20	**Наповнення контентом Web-додатку для підтримки діяльності керівника проектної організації у КЗАПР**		20.4.2022	27.4.2022	0
21	Додання інформації про проекти		20.4.2022	26.4.2022	0
22	Додання працівників компанії		27.4.2022	27.4.2022	0
23	Розбиття процесу проектування на підзадачі		27.4.2022	27.4.2022	0
24	**Тестування Web-додатку для підтримки діяльності керівника проектної організації у КЗАПР**		28.4.2022	3.5.2022	0
25	Beta-тестування		28.4.2022	2.5.2022	0

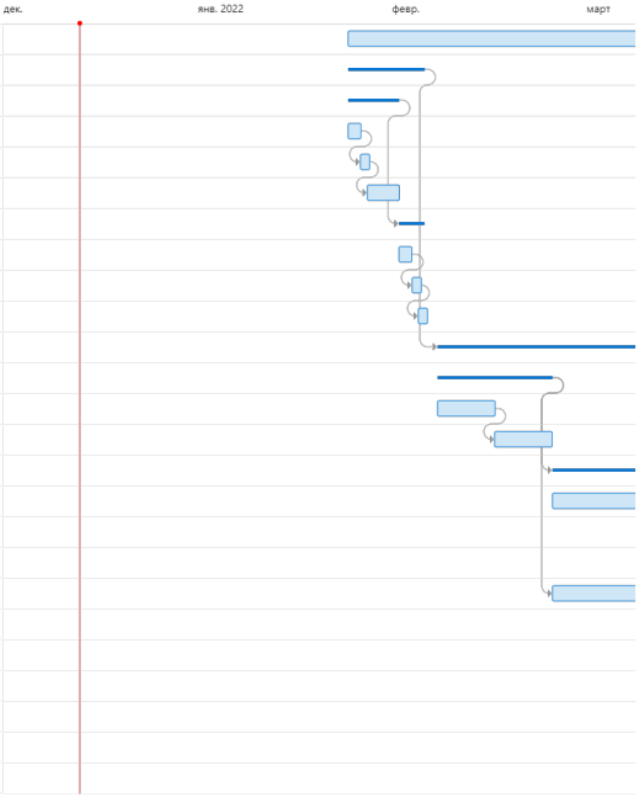


Рисунок Б.3 – Діаграма Ганта. Частина 1

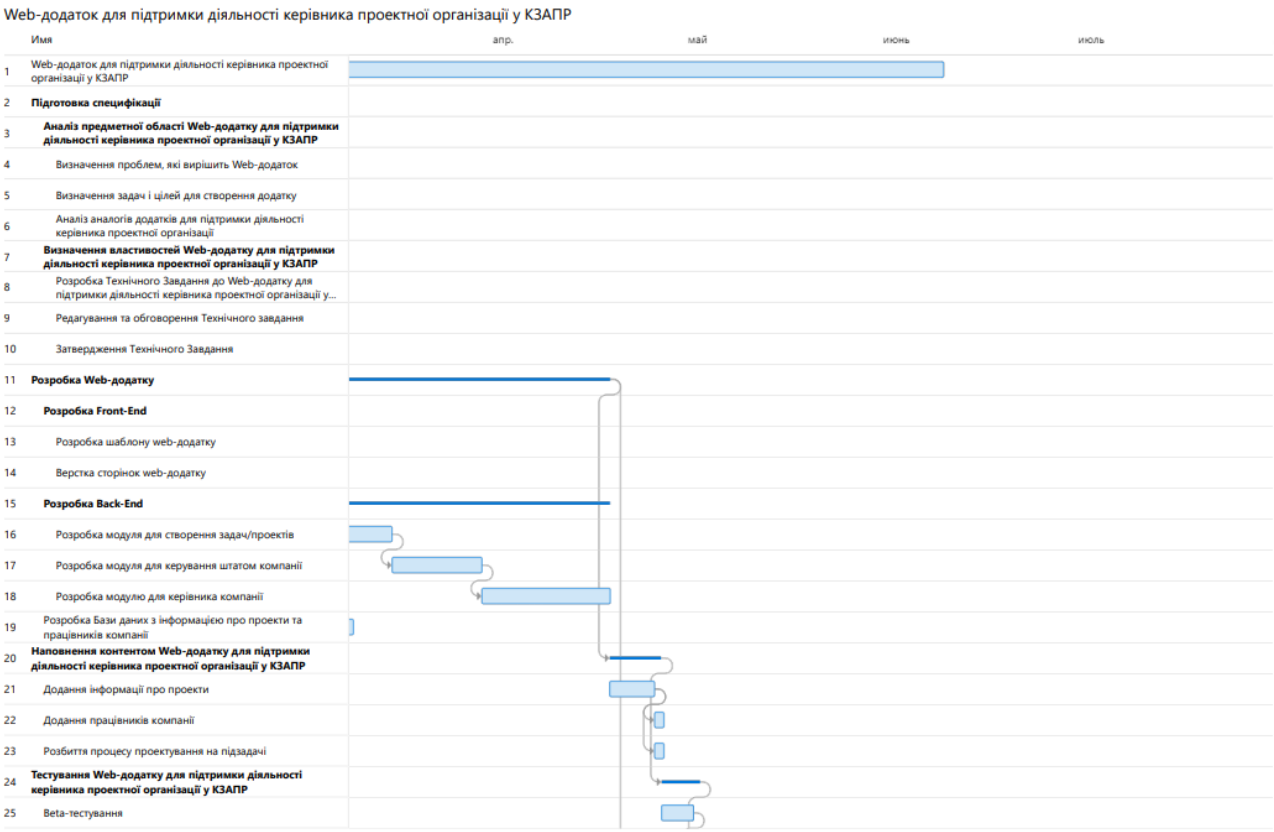


Рисунок Б.4 – Продовження діаграми Ганта. Частина 2

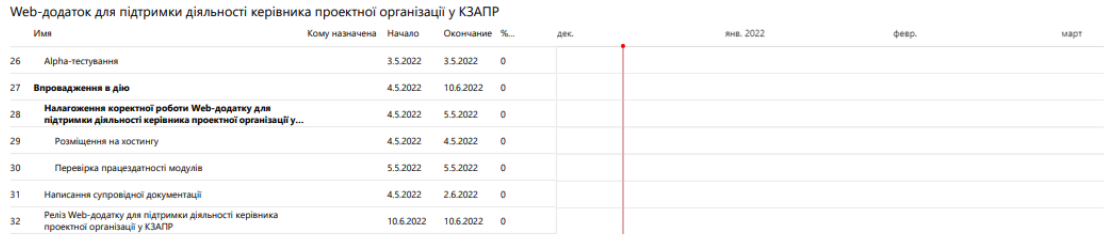


Рисунок Б.5 – Продовження діаграми Ганта. Частина 3

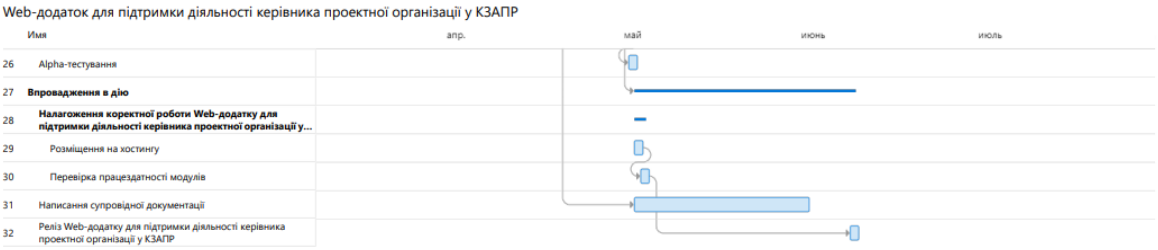


Рисунок Б.6 – Продовження діаграми Ганта. Частина 4

Управління ризиками проекту. Управління ризиком – це процес реагування на виникаючі події та зміни ризиків у процесі розробки проекту. Під час виконання якісної оцінки ризиків виникає необхідність визначити ті ризики, як будуть усунені якнайшвидше. Реагування на різний ступінь важливості ризику буде відповідне. Наступним етапом буде виконання кількісного оцінювання ризиків. Якісне та кількісне оцінювання можуть бути виконані як окремо, так і одночасно. Це залежить від ступеня забезпечення проекту. Таблиця Б.2 показує шкалу для класифікації ризиків за величиною впливу на проект та ймовірність виникнення ризиків.

Таблиця Б.2 – 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

Для зниження негативного впливу ризиків на проект необхідно виконати планування редагування на них. До планування входить: оцінка наслідків впливу на проект та визначення ефективності розробки. Оцінка виконується за показниками, що описані в таблиці Б.2. Результатом планування реагування є матриця ймовірності виникнення ризиків та впливу ризиків, що зображена на рисунку Б.7. Зелений колір відповідає за прийнятні ризики, жовтий за виправдані, а червоний за недопустимі.

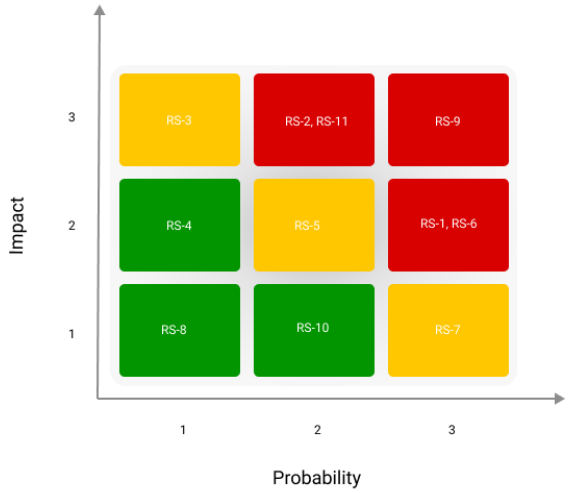


Рисунок Б.7. – Матриця ймовірності

Таблиця Б.3 демонструє класифікацію ризиків за рівнем, відповідно до отриманого значення індексу. Таблиця Б.4 описує ризики та стратегії реагування на кожен з ризиків.

Таблиця Б.3 – Шкала оцінювання за рівнем ризику.

№	**Назва**	**Межі**	**Ризики, які входять(номера)**
1	Прийнятні	1≤R≤2	RS-8, RS-4, RS-10
2	Виправдані	3≤R≤4	RS-3, RS-5, RS-7
3	Недопустимі	6≤R≤9	RS-2, RS-11, RS-9, RS-1, RS-6

Таблиця Б.4 – 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	Низька кваліфікація розробників	Середня	Високий	6	1. Підвищити кваліфікацію розробників. 2. Надати необхідні ресурси або курси для підвищення знань.	Пом’якшення	Видати необхідну літературу, дослідити онлайн заняття, додати час на саморозвиток.
RS-2	Відкритий	Зміна вимог замовником	Середня	Високий	6	1. Домовитись про всі вимоги на початкових етапах проектування.	Пом’якшення	Переоцінка вимог до проекту, коли вимоги від замовника змінились.

Продовження таблиці Б.4 – Ризики та стратегії реагування

RS-3	Відкритий	Проблеми з роботою хостингу	Низька	Високий	3	Підбір надійного хостингу на початку проектування процесів.	Прийняття	Зміна хостингу на більш надійний.
RS-4	Відкритий	Недотримання обсягів фінансування	Низька	Середній	3	Підписати контракт з замовником, домовитись про певний об’єм фінансування.	Пом’якшення	Переглянути контракт, обґрунтувати необхідність додаткового фінансування на проект.
RS-5	Відкритий	Недостатня підтримка проекту з боку замовника	Низька	Високий	4	Налагодження гарних відносин між розробником та керівником, створення комфортних умов для роботи та співпраці.	Пом’якшення	Влаштувати перемови з замовником, щодо владнання проблеми недостатнього фідбеку між розробником та замовником.

Продовження таблиці Б.4 – Ризики та стратегії реагування

RS-6	Відкритий	Хвороба розробника	Середня	Високий	6	Забезпечення комфортних умов праці.	Прийняття	Виділити необхідні ресурси для допомоги розробнику та знайти заміну розробнику.
RS-7	Відкритий	Відсутність механізму резервного копіювання	Низька	Високий	3	Забезпечити налаштування автоматичного збереження даних та створити додаткові копії на різних носіях інформації.	Попередження	Створення хмарних копій після кожного виконаного етапу розробки.
RS-8	Відкритий	Реалізація непотрібного функціоналу	Низька	Низький	1	Домовитись з замовником про можливу реалізацію додаткового функціоналу	Використання	Проаналізувати всі вигоди та збитки після реалізування додаткового функціоналу.

Продовження таблиці Б.4 – Ризики та стратегії реагування

RS-9	Відкритий	Неправильне розпорядження часом розробника	Висока	Високий	9	Проаналізувати найважливіші процеси та роботи, визначити пріорітети, дотримуватися календарного плану.	Пом’якшення	Переглянути порядок пріорітетів робіт, застосувати додатки для планування часу розробника, наприклад «Trello».
RS-10	Відкритий	Поява альтернативного продукту	Низька	Середній	2	Провести дослідження існуючих аналогів, додати унікальний функціонал	Прийняття	
RS-11	Відкритий	Помилки під час проектування проекту	Середня	Високий	6	На ранніх етапах визначити всі вимоги з замовником; демонструвати результати на проміжних етапах розробки.	Пом’якшення	Здійснювати проміжний контроль результатів розробки під час виконання проекту.

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Web application to support the activities of the head of the designing organization in KZAPR

Kateryna Medvedeva, *Student of IT-82*;
Viktoriia Antypenko, *Associate Professor*

Department "Information Technology"
Sumy State University, Sumy, Ukraine

Nowadays the use of modern information technologies (IT) allows to automate the process of tasks designing in order to speed up and facilitate control over the implementation of planned work. Today it really is a topical issue considering a modern situation in the world.

Planning is essential for business leaders and managers of enterprises, who are obliged to organize carrying out the manufacturing tasks by their subordinates in both predictable and extreme conditions, such as pandemics or hostilities and so on. During this type of activity, it is difficult to monitor the task's progress because the employee is absent at the office. Implementing communication through social networks is also not a complete solution because they do not provide all the necessary functions of messages, structuring tasks etc.. It is more convenient for the manager to check the performance of work using a specially created web-application than to contact each team member about his or her achievements and results several times a day.

Planning of manufacturing processes, control over the tasks implementation, reviewing development statistics, signing contracts are functions which must be implemented for successful collective action. The most excellent effect is achieved on a single information basis using unified processing of all tasks through the tools complex of automation the designing works – KZAPR.

The integration of this information technology is performed using components such as HTML5 for markup, Express, which is a web applications framework for Node.js, for the server part, JavaScript React library for the client part, SASS metalanguage for style description, and GitHub version control system.

The result of current research is a web application to support the activities of the head of the designing organization in KZAPR. Its use significantly speeds up and automates the management process of the machine-building company.

ДОДАТОК Г

Лістинг програмного коду основних модулів web-додатку

Посилання на Git репозитарій: https://github.com/DmytroRaiko/TaskerKZAPR

BusinessPage.js

```
import React, { useState, useEffect, useContext } from 'react';
import { useQuery, useMutation } from 'react-query';
import Contracts from './business/Contracts';
import Projects from './business/Projects';
import Statistics from './business/Statistics';
import { getContracts, getProjectForContract } from '../../services/crud/contracts';
import contextUser from '../../services/context-user';
import Page404 from './404';

const BusinessPage = () => {
  const { user } = useContext(contextUser);
  const [selectedContract, setSelectedContract] = useState(0);
  const cs = useQuery(
    'contracts-business',
    () => getContracts(),
    {
      manual: true,
    },
  );
  const contracts = cs?.data?.data?.data;
  const p = useMutation(
    'contracts-business',
    (id) => getProjectForContract(id),
  );
  const projects = p?.data?.data?.data;

  useEffect(() => {
    if (contracts && contracts[0] && !cs.isLoading) {
      setSelectedContract(contracts[0]);
    }
  }, [contracts]);

  useEffect(() => {
    if (contracts && contracts[0] && !cs.isLoading) {
      p.mutate(selectedContract.contractId);
    }
  }, [selectedContract]);

  return ((user.roleName === 'owner'
      || user.roleName === 'customer')
    && (
    <div className="main-task-body">
      {contracts && (
      <div className="body-business">
        <div className="upper">
          <Contracts contracts={contracts} />
```

```
          <Projects
            contracts={contracts}
            projects={projects}
            setSelected={setSelectedContract}
            selected={selectedContract}
          />
        </div>

        {selectedContract?.contractId
          && (
          <Statistics projects={projects || []} />
          )}
      </div>
      )}
    </div>
    )
  ) || <Page404 />;
};

export default BusinessPage;
```

Contracts.js (Контейнер для блоку контрактів)

```
import React, { useContext } from 'react';
import PropTypes from 'prop-types';
import Button from '@mui/material/Button';
import { Link as DomLink } from 'react-router-dom';
import Contract from '../../../components/business/Contract';
import { LanguageContext } from '../../language/Language';

const Contracts = ({
  contracts,
}) => {
  const { dictionary } = useContext(LanguageContext);

  return (
    <div className="business-block">
      <div className="block-header">
        {dictionary.pages.business.contracts}
        <Button
          component={DomLink}
          to="/new-contract"
          variant="outlined"
          size="small"
          className="btn-new"
        >
          {dictionary.new}
        </Button>
      </div>
      <Contract contracts={contracts} />
    </div>
  );
};

Contracts.propTypes = {
  contracts: PropTypes.arrayOf(
    PropTypes.shape({}),
  ),
};
```

```
Contracts.defaultProps = {
  contracts: [],
};

export default Contracts;
```

Contract.js (Компонент блоку контрактів)

```
import React from 'react';
import PropTypes from 'prop-types';
import { Link as DomLink } from 'react-router-dom';
import Link from '@mui/material/Link';
import MobileStepper from '@mui/material/MobileStepper';
import IconButton from '@mui/material/IconButton';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-material/KeyboardArrowRight';
import SwipeableViews from 'react-swipeable-views';
import { autoPlay } from 'react-swipeable-views-utils';

const AutoPlaySwipeableViews = autoPlay(SwipeableViews);

const Contract = ({
  contracts,
}) => {
  const [activeStep, setActiveStep] = React.useState(0);
  const contractsByThree = [];
  // eslint-disable-next-line no-plusplus
  for (let i = 0; i < Math.ceil(contracts.length / 3); i++) {
    contractsByThree[i] = contracts.slice((i * 3), (i * 3) + 3);
  }

  const maxSteps = contractsByThree.length;

  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1);
  };

  const handleBack = () => {
    setActiveStep((prevActiveStep) => prevActiveStep - 1);
  };

  const handleStepChange = (step) => {
    setActiveStep(step);
  };

  const blockList = (list, index) => {
    const block = list?.map((item) => (
      <Link
        to={`/contract/${item.contractId}`}
        component={DomLink}
        className="contract-item"
        key={`bus-contract-link-${item.contractId}`}
        color="inherit"
        underline="hover"
      >
        <div
          className="name"
          style={{
            width: '343px',
            whiteSpace: 'nowrap',
          }}
        >
```

```
        {item.title}
      </div>
      <div className="status">
        {item.status}
      </div>
    </Link>
  ));

  return (
    <div className="list" key={`block-bus-contract-link-${index}`}>
      {block}
    </div>
  );
};

const contractList = contractsByThree?.map((contract, index) => (
  Math.abs(activeStep - index) <= 2
    ? (
      blockList(contract, index)
    )
    : null
));

if (contracts.length > 3) {
  return (
    <div className="list swipear">
      <AutoPlaySwipeableViews
        index={activeStep}
        onChangeIndex={handleStepChange}
        enableMouseEvents
        interval={7500}
      >
        {contractList}
      </AutoPlaySwipeableViews>
      <MobileStepper
        className="bottom"
        steps={maxSteps}
        position="static"
        activeStep={activeStep}
        sx={{
          background: 'unset',
          padding: '0',
        }}
        nextButton={(
          <IconButton
            size="small"
            onClick={handleNext}
            disabled={activeStep === maxSteps - 1}
          >
            <KeyboardArrowRight />
          </IconButton>
        )}
        backButton={(
          <IconButton size="small" onClick={handleBack} disabled={activeStep === 0}>
            <KeyboardArrowLeft />
          </IconButton>
        )}
      />
    </div>
  );
}
return (
  <div className="list">
```

```
      {contractList}
    </div>
  );
};

Contract.propTypes = {
  contracts: PropTypes.arrayOf(
    PropTypes.shape({}),
  ),
};

Contract.defaultProps = {
  contracts: [],
};

export default Contract;
```

Contracts.js (Контейнер сторінки контрактів)

```
import React, { useContext } from 'react';
import { useQuery } from 'react-query';
import Backdrop from '@mui/material/Backdrop';
import CircularProgress from '@mui/material/CircularProgress';
import ContractsPage from '../../components/info-page/contract/ContractsPage';
import { getAllContractsProjects } from '../../services/crud/projects';
import contextUser from '../../services/context-user';
import Page404 from './404';

const Contracts = () => {
  const { user } = useContext(contextUser);

  const { data, isLoading } = useQuery(
    ['contracts-main-pages'],
    () => getAllContractsProjects(),
    {
      manual: true,
    },
  );

  const contractsArray = data?.data?.data;

  return ((user.roleName === 'owner'
      || user.roleName === 'customer')
    && (
    <div className="main-task-body contracts">
      {!isLoading || (
        <Backdrop
          sx={{ color: '#fff', zIndex: (theme) => theme.zIndex.drawer + 1 }}
          open={isLoading}
        >
          <CircularProgress color="inherit" />
        </Backdrop>
      )}

      {contractsArray && (
      <div className="contract-body">
        <ContractsPage
          contracts={contractsArray}
        />
      </div>
      )}
    </div>
```

```
    )
  ) || <Page404 />;
};

export default Contracts;
```

ContractsPage.js(Компонент для відображення контрактів на сторінці Контрактів)

```
import React, { useContext } from 'react';
import PropTypes from 'prop-types';
import Avatar from '@mui/material/Avatar';
import Card from '@mui/material/Card';
import { CardActionArea } from '@mui/material';
import { Link as RouterLink } from 'react-router-dom';
import Link from '@mui/material/Link';
import stringAvatar from '../../../services/icons/avatarIcon';
import { dateFormat } from '../../../services/time';
import { LanguageContext } from '../../../containers/language/Language';

const ContractsPage = ({
  contracts,
}) => {
  const { dictionary } = useContext(LanguageContext);

  return (
    <>
      {contracts?.map((contract) => (
        <Card
          key={`contracts-page-contract-${contract.contractId}`}
          sx={{
            width: '100%',
          }}
        >
          <CardActionArea
            className="card-task view-task"
            component={RouterLink}
            to={`/contract/${contract.contractId}`}
          >
            <Avatar
              className="card-icon"
              {...stringAvatar(contract.title)}
            />
            <div className="card-info">
              <div className="info-title text-22">
                {' '}
                {contract.title}
                <div className="hor-line" />
              </div>

              {contract?.companyOwnerProfileId && (
              <div className="employee-task">
                <p
                  className="text-11"
                  style={{
                    marginBottom: '2px',
                  }}
                >
                  {dictionary.pages.profile.pm}
                  :
                </p>
```

```
            <Link
              component={RouterLink}
              to={`/profile/${contract?.companyOwnerProfileId}`}
              color="inherit"
              underline="hover"
              className="emp text-10"
              onClick={(e) => {
                e.stopPropagation();
              }}
            >
              {contract?.companyOwnerName}
              {' '}
              {contract?.companyOwnerSurname}
            </Link>
          </div>
          )}

          {contract?.customerProfileId && (
          <div className="employee-task">
            <p
              className="text-11"
              style={{
                marginBottom: '2px',
              }}
            >
              {dictionary.pages.customer}
              :
            </p>

            <Link
              component={RouterLink}
              to={`/profile/${contract?.customerProfileId}`}
              color="inherit"
              underline="hover"
              className="emp text-10"
              onClick={(e) => {
                e.stopPropagation();
              }}
            >
              {contract?.customerName}
              {' '}
              {contract?.customerSurname}
            </Link>
          </div>
          )}

          <div
            className="date"
            style={{ left: '0 !important' }}
          >
            <div className="pre-date text text-11" style={{ display: 'block' }}>
              {dictionary.tasksList.until}
            </div>
            <div className="date-time-end text-13" style={{ display: 'block' }}>
              {' '}
              {dateFormat(contract.end)}
            </div>
          </div>
        </div>
      </CardActionArea>
    </Card>
))}
<Card
```

```
        key="contracts-page-contract-new"
        sx={{
          width: '100%',
        }}
      >
        <CardActionArea
          className="card-task view-task"
          component={RouterLink}
          to="/new-contract"
          sx={{
            background: '#f6f6f6 !important',
          }}
        >
          <div className="card-info add-item">
            {dictionary.pages.addNewContract}
          </div>
        </CardActionArea>
      </Card>
    </>
  );
};

ContractsPage.propTypes = {
  contracts: PropTypes.arrayOf(
    PropTypes.shape({}),
  ),
};

ContractsPage.defaultProps = {
  contracts: [],
};

export default ContractsPage;
```

ProjectPage (Контейнер для відображення інформації про проект на сторінці проекту)

```
import React, { useContext } from 'react';
import { Link as DomLink, useParams } from 'react-router-dom';
import PropTypes from 'prop-types';
import { useQuery } from 'react-query';
import Button from '@mui/material/Button';
import AvatarPage from '../../../components/info-page/AvatarPage';
import TitlePage from '../../../components/info-page/TitlePage';
import ContentPage from '../../../components/info-page/ContentPage';
import TasksPage from '../../../components/tasks/TasksPage';
import { LanguageContext } from '../../language/Language';
import ProjectTables from '../../../components/admin/projectTables';
import { getProjectById, getTasksForProject } from '../../../services/crud/projects';
import contextUser from '../../../services/context-user';

//   attached: [],

const ProjectPage = ({ handleModalTaskOpen }) => {
  const { dictionary } = useContext(LanguageContext);
  const { projectId } = useParams();
  const role = sessionStorage.getItem('role');
  const { user } = useContext(contextUser);

  const p = useQuery(
    `project-${projectId}`,
    () => getProjectById(projectId),
    {
```

```
    manual: true,
  },
);
const project = p?.data?.data?.data;

const t = useQuery(
  `projects-tasks-${projectId}`,
  () => getTasksForProject(projectId),
  {
    manual: true,
  },
);
const tasksArray = t?.data?.data?.data;

return (
  <div className="main-task-body page-info">
    {project
      && (
        <>
          <div className="left-bar">
            <AvatarPage
              avatarLink={project.avatarLink}
              title={project.title}
            />
          </div>

          <div className="content">
            <TitlePage
              type={dictionary.pages.project}
              title={`${project.title}`}
              endAction={
                (((Number(project.projectManagerProfileId) === Number(user.id)
                    || Number(project.contractOwner) === Number(user.id))
                  && (
                    <Button
                      component={DomLink}
                      to="edit"
                      variant="contained"
                    >
                      {dictionary.edit}
                    </Button>
                  ))) || null
                }
              />

              <div className="block-content">
                <ContentPage
                  dates={{
                    start: project?.dateStart,
                    end: project?.dateEnd,
                  }}
                  pm={{
                    profileId: project?.projectManagerProfileId,
                    name: project?.projectManagerName,
                    surname: project?.projectManagerSurname,
                    email: project?.projectManagerEmail,
                  }}
                  contract={{
                    id: project?.contractContractId,
                    title: project?.contractTitle,
                  }}
                  description={project?.description}
                  status={project?.statusName}
```

```
              />
            </div>

            {/*    {(attached && project.attached.length !== 0) */}
            {/* && <DocumentsPage documents={project.attached} />} */}

            {(tasksArray && tasksArray.length !== 0) && ((role !== 'admin')
              ? (
                <div className="tasks-block">
                  <TasksPage
                    tasks={tasksArray}
                    onOpen={handleModalTaskOpen}
                  />
                </div>
              )
              : (
                <ProjectTables
                  employees={usersEmp}
                  tasks={tasksArray}
                  handleModalTaskOpen={handleModalTaskOpen}
                />
              )
            )}
          </div>

        </>
      )}
    </div>
  );
};

ProjectPage.propTypes = {
  handleModalTaskOpen: PropTypes.func.isRequired,
};

export default ProjectPage;
```

ContentPage (Компонент для відображення інформації про проект)

```
import React, { useContext } from 'react';
import PropTypes from 'prop-types';
import Link from '@mui/material/Link';
import { Link as DomLink } from 'react-router-dom';
import { Button } from '@mui/material';
import { dateFormat } from '../../services/time';
import { LanguageContext } from '../../containers/language/Language';

const ContentPage = ({
  description, dates, customer, pm, contract, status,
}) => {
  const { dictionary } = useContext(LanguageContext);
  const role = sessionStorage.getItem('role');

  return (
    <div className="">
      <div className="info-line">
        <div className="title">
          {dictionary.pages.description}
          :
        </div>

        <div className="value description">
```

```
    {description || (
      <i>
        {dictionary.pages.description}
      </i>
    )}
  </div>
</div>

<div className="info-line dates">
  <div className="title">
    {dictionary.pages.dates}
    :
  </div>

  <div className="time-table">
    <div className="info-line">
      <div className="title subtitle">
        {dictionary.pages.start}
      </div>
      <div className="value">
        {dateFormat(dates.start)}
      </div>
    </div>

    <div className="info-line">
      <div className="title subtitle">
        {dictionary.pages.end}
      </div>
      <div className="value">
        {dateFormat(dates.end)}
      </div>
    </div>
  </div>
</div>

{customer?.profileId && (
  <div className="info-line emp">
    <div className="title">
      {dictionary.pages.customer}
      :
    </div>
    <div className="value center">
      <Link
        component={DomLink}
        to={`/profile/${customer.profileId}`}
        underline="hover"
      >
        {`${customer.name} ${customer.surname}`}
      </Link>
    </div>
  </div>
)}

<div className="info-line emp">
  <div className="title">
    {dictionary.pages.profile.pm}
    :
  </div>
  <div className="value center">
    <Link
      component={DomLink}
      to={`/profile/${pm.profileId}`}
      underline="hover"
```

```
          >
            {`${pm.name} ${pm.surname}`}
          </Link>
        </div>
      </div>

      { contract.id
        && (
          <div className="info-line emp p-btn-action">
            <div className="left-side">
              <div className="title">
                {dictionary.pages.contract}
                :
              </div>
              <div className="value center">
                <Link
                  component={DomLink}
                  to={`/contract/${contract.id}`}
                  underline="hover"
                >
                  {contract.title}
                </Link>
              </div>
            </div>
            {role !== 'admin' && (
            <Button
              variant="outlined"
              component={DomLink}
              to="employees"
              size="small"
            >
              {dictionary.pages.employees.employees}
            </Button>
            )}
          </div>
        )}

      <div className="info-line emp">
        <div className="title">
          {dictionary.status.status}
          :
        </div>
        <div className="value center">
          {status}
        </div>
      </div>
    </div>
  );
};

ContentPage.propTypes = {
  description: PropTypes.string,
  dates: PropTypes.shape({}).isRequired,
  customer: PropTypes.shape({}),
  pm: PropTypes.shape({}).isRequired,
  contract: PropTypes.shape({}),
  status: PropTypes.string,
};

ContentPage.defaultProps = {
  description: null,
  customer: {},
  contract: {},
```

```
  status: 'Waiting',
};

export default ContentPage;
```

EditProjectPage.js

```
import React, { useContext } from 'react';
import { useQuery } from 'react-query';
import { Link, useParams } from 'react-router-dom';
import { IconButton } from '@mui/material';
import ArrowBackIcon from '@mui/icons-material/ArrowBack';
import TitlePage from '../../../components/info-page/TitlePage';
import EditProjectContentPage from '../../../components/info-page/project/EditProjectContentPage';
import EditAvatar from '../../../components/info-page/EditAvatar';
import { LanguageContext } from '../../language/Language';
import { getEmployeesByPositions, getStatus } from '../../../services/crud/pages';
import { getProjectById } from '../../../services/crud/projects';
import contextUser from '../../../services/context-user';
import Page404 from '../404';

const EditProjectPage = () => {
  const { dictionary } = useContext(LanguageContext);
  const { projectId } = useParams();
  const { user } = useContext(contextUser);

  const s = useQuery(
    'status-project',
    () => getStatus('project'),
  );
  const statusArray = s?.data?.data?.data || [];

  const p = useQuery(
    `project-${projectId}`,
    () => getProjectById(projectId),
    {
      manual: true,
    },
  );
  const project = p?.data?.data?.data;

  const pm = useQuery(
    `pm-${projectId}`,
    () => getEmployeesByPositions('pm'),
  );
  const projectManagers = pm?.data?.data?.data;

  return ((Number(project?.projectManagerProfileId) === Number(user.id)
      || Number(project?.contractOwner) === Number(user.id))
    && (
    <div className="main-task-body page-info">
      <div className="left-bar">
        {project
          && (
          <EditAvatar
            profileId={project.projectId}
            name={project.title}
            avatarLink={project.avatarLink}
          />
          )}
      </div>
```

```
      <div className="content">
        <TitlePage
          type=""
          title={dictionary.pages.editProject}
          startAction={(
            <IconButton
              component={Link}
              size="small"
              to={`/project/${projectId}`}
              sx={{
                marginRight: '5px',
              }}
            >
              <ArrowBackIcon />
            </IconButton>
          )}
        />

        {(projectManagers && project && statusArray) && (
          <div className="block-content">
            <EditProjectContentPage
              project={project}
              projectManagers={projectManagers}
              statusArray={statusArray}
            />
          </div>
        )}
      </div>
    </div>
    )
  ) || <Page404 />;
};

export default EditProjectPage;
```

EditProjectContentPage.js (Компонент форми для редагування інформації проекту)

```
import React, { useContext } from 'react';
import PropTypes from 'prop-types';
import { TextField } from 'formik-mui';
import { Field, Form, Formik } from 'formik';
import Button from '@mui/material/Button';
import { Link as RouterLink } from 'react-router-dom';
import { Link } from '@mui/material';
import { useMutation } from 'react-query';
import DateTime from '../../tasks/elements/DateTime';
import Autocomplete from '../../form-elements/Autocomplete';
import ProjectPageSchema from '../../../services/Schems/ProjectPageSchema';
// import TaskAttachment from '../../tasks/elements/TaskAttachment';
import { LanguageContext } from '../../../containers/language/Language';
import { changeProjectById } from '../../../services/crud/projects';

const EditProjectContentPage = ({
  projectManagers, project, statusArray,
}) => {
  const { dictionary } = useContext(LanguageContext);


  const mutation = useMutation(
    `project-edit-${project.projectId}`,
    (formData) => changeProjectById(project.projectId, formData),
  );
```

```
const onEdit = (dataForm, { setSubmitting }) => {
  mutation.mutate({
    title: dataForm.title,
    description: dataForm.description,
    dateStart: dataForm.dateTimeStart,
    dateEnd: dataForm.dateTimeEnd,
    status_id: dataForm.status.statusId,
  });
  setSubmitting(false);
};

const formValues = {
  title: project.title,
  description: project.description || '',
  dateTimeStart: new Date(project.dateStart) || new Date(),
  dateTimeEnd: new Date(project.dateEnd) || new Date(),
  projectManager: project?.projectManagerProfileId ? {
    profileId: project.projectManagerProfileId,
    name: project.projectManagerName,
    surname: project.projectManagerSurname,
    email: project.projectManagerEmail,
  } : null,
  status: project?.statusId ? {
    statusId: project.statusId,
    name: project.statusName,
    color: project.statusColor,
  } : null,
  contract: {
    contractId: project?.contractContractId,
    title: project?.contractTitle,
  },
};

const schema = ProjectPageSchema;

return (
  <div>
    <Formik
      onSubmit={onEdit}
      initialValues={formValues}
      validationSchema={schema}
    >
      {({
        setFieldValue, values, dirty, isValid,
      }) => (
        <Form>
          <Field
            component={Autocomplete}
            name="status"
            placeholder={dictionary.status.status}
            options={statusArray}
            sx={{
              width: '200px',
              marginLeft: 'auto',
            }}
            optionLabel={
              (option) => option.name
            }
            optionEqual={
              (option, value) => option.statusId === value.statusId
            }
          />
```

```
<div className="info-line new">
  <div className="title">
    {dictionary.pages.title}
    :
  </div>

  <div className="value">
    <Field
      component={TextField}
      type="text"
      name="title"
      multiline
      label={dictionary.pages.title}
      placeholder={dictionary.pages.title}
      variant="standard"
      fullWidth
      margin="dense"
    />
  </div>
</div>

<div className="info-line new">
  <div className="title">
    {dictionary.pages.description}
    :
  </div>

  <div className="value">
    <Field
      component={TextField}
      type="text"
      name="description"
      multiline
      label={dictionary.pages.description}
      placeholder={dictionary.pages.description}
      variant="standard"
      fullWidth
      margin="dense"
    />
  </div>
</div>

<div className="info-line new">
  <div className="title">
    {dictionary.pages.deadlines}
    :
  </div>
  <div className="time-table">
    <div className="info-line">
      <div className="title subtitle">
        {dictionary.task.from}
      </div>

      <div className="value">
        <DateTime
          name="dateTimeStart"
          label={dictionary.task.from}
          setFieldValue={setFieldValue}
          time={values.dateTimeStart}
        />
      </div>
    </div>
```

```
                <div className="info-line">
                  <div className="title subtitle">
                    {dictionary.task.till}
                  </div>
                  <div className="value">
                    <DateTime
                      name="dateTimeEnd"
                      label={dictionary.task.till}
                      setFieldValue={setFieldValue}
                      time={values.dateTimeEnd}
                    />
                  </div>
                </div>
              </div>

              <div className="info-line new">
                <div className="title">
                  {dictionary.pages.profile.pm}
                  :
                </div>
                <div className="value center">
                  <Field
                    component={Autocomplete}
                    name="projectManagers"
                    placeholder={dictionary.task.unassigned}
                    options={projectManagers}
                    optionLabel={(option)    =>    `${option.name}    ${option.surname}
[${option?.email}]`}
                    optionEqual={(option,     value)     =>     option.profileId     ===
value.profileId}
                  />
                </div>
              </div>

              <div className="info-line new">
                <div className="title">
                  {dictionary.pages.contract}
                  :
                </div>
                <div className="value">
                  <Link
                    component={RouterLink}
                    to={`/contract/${project.contractContractId}`}
                    className="text-19"
                    underline="hover"
                    color="inherit"
                  >
                    {project.contractTitle}
                  </Link>
                </div>
              </div>

              <div className="button-block">
                <Button
                  variant="contained"
                  type="submit"
                  disabled={!(isValid && dirty)}
                >
                  {dictionary.save}
                </Button>
              </div>
            </Form>
```

```
        )}
      </Formik>
    </div>
  );
};

EditProjectContentPage.propTypes = {
  project: PropTypes.shape({}),
  projectManagers: PropTypes.arrayOf(
    PropTypes.shape({}),
  ),
};

EditProjectContentPage.defaultProps = {
  project: {},
  projectManagers: [],
};

export default EditProjectContentPage;
```

EmployeesProjectPage.js

```
import React, { useContext, useState } from 'react';
import { Link, useParams } from 'react-router-dom';
import ArrowBackIcon from '@mui/icons-material/ArrowBack';
import { IconButton } from '@mui/material';
import Button from '@mui/material/Button';
import AddIcon from '@mui/icons-material/Add';
import { useQuery } from 'react-query';
import TitlePage from '../../../components/info-page/TitlePage';
import          EmployeesContentPage          from          '../../../components/info-
page/project/EmployeesContentPage';
import { LanguageContext } from '../../language/Language';
import NewEmployeeForProjectModal from '../../modals/NewEmployeeForProjectModal';
import {
  getProjectById,
  getProjectStaffById,
  getProjectStaffProjectManagers,
  getUnattachedUsers,
} from '../../../services/crud/projects';
import { getPositions } from '../../../services/crud/pages';
import contextUser from '../../../services/context-user';

const EmployeesProjectPage = () => {
  const { dictionary } = useContext(LanguageContext);
  const { projectId } = useParams();
  const { user } = useContext(contextUser);

  const p = useQuery(
    `project-empl-${projectId}`,
    () => getProjectById(projectId),
    {
      manual: true,
    },
  );
  const project = p?.data?.data?.data;

  const pe = useQuery(
    `project-staff-list-${projectId}`,
    () => getProjectStaffById(projectId),
    {
      manual: true,
```

```
  },
);
const projectEmployees = !pe?.isLoading ? pe?.data?.data?.data || [] : null;

const m = useQuery(
  `project-staff-list-managers-${projectId}`,
  () => getProjectStaffProjectManagers(projectId),
  {
    manual: true,
  },
);
const projectManagers = !m?.isLoading ? m?.data?.data?.data || [] : null;

const uu = useQuery(
  `project-staff-list-unattached-${projectId}`,
  () => getUnattachedUsers(projectId),
  {
    manual: true,
  },
);
const unattachedUsers = uu?.data?.data?.data || [];

const ps = useQuery(
  'project-staff-list-positions',
  () => getPositions(),
);
const positions = ps?.data?.data?.data;

const [newEmployee, setNewEmployee] = useState({
  show: false,
  project: 0,
});

const handleOpenNewEmployee = () => {
  setNewEmployee({ show: true, project: projectId });
};

const handleCloseNewEmployee = () => {
  setNewEmployee({ show: false, project: 0 });
};

return (
  <div className={`main-task-body page-info emp ${projectId}`}>
    <div className="content">
      {project
        && (
        <TitlePage
          type={dictionary.pages.employees.employees}
          title={project.title}
          startAction={(
            <IconButton
              component={Link}
              size="small"
              to={`/project/${projectId}`}
              sx={{
                marginRight: '5px',
              }}
            >
              <ArrowBackIcon />
            </IconButton>
        )}
          endAction={
            (Number(project?.projectManagerProfileId) === Number(user.id)
```

```
                || Number(project?.contractOwner) === Number(user.id))
              && (
              <Button
                startIcon={<AddIcon />}
                variant="contained"
                onClick={() => handleOpenNewEmployee()}
              >
                {dictionary.add}
              </Button>
              )
            }
          />
          )}

        {(projectEmployees && projectManagers && positions)
        && (
        <div className="block-content" style={{ padding: 0 }}>
          <EmployeesContentPage
            projectId={projectId}
            employees={projectEmployees}
            projectManagers={projectManagers}
            positions={positions}
            edit={(Number(project?.projectManagerProfileId) === Number(user.id)
              || Number(project?.contractOwner) === Number(user.id))}
          />
        </div>
        )}

        {(unattachedUsers && projectManagers && positions)
          && (
          <NewEmployeeForProjectModal
            handleCloseModal={handleCloseNewEmployee}
            data={newEmployee}
            projectManagers={projectManagers}
            positions={positions}
            newEmployeesWhichDontAttached={unattachedUsers}
          />
          )}
      </div>
    </div>
  );
};

export default EmployeesProjectPage;
```

EmplyeesContentPage.js

```
import React, { useContext, useState } from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import { styled } from '@mui/material/styles';
import TableCell, { tableCellClasses } from '@mui/material/TableCell';
import EditIcon from '@mui/icons-material/Edit';
import DeleteIcon from '@mui/icons-material/Delete';
import IconButton from '@mui/material/IconButton';
import Link from '@mui/material/Link';
import { Link as DomLink } from 'react-router-dom';
import PropTypes from 'prop-types';
import { useMutation, useQueryClient } from 'react-query';
```

```
import                          EditPositionPMProfileModal                          from
'../../../containers/modals/EditPositionPMProfileModal';
import { LanguageContext } from '../../../containers/language/Language';
import { deleteEmployee } from '../../../services/crud/projects';



const EmployeesContentPage = ({
  employees, projectId, edit,
  positions, projectManagers,
}) => {
  const { dictionary } = useContext(LanguageContext);
  const [changeModal, setChangeModal] = useState({ show: false });
  const queryClient = useQueryClient();

  const handleOpenChangeModal = (pm, position, profileData, id) => {
    setChangeModal({
      show: true, pm, position, profileData, id,
    });
  };

  const handleCloseChangeModal = () => {
    setChangeModal({ show: false });
  };

  const mutationDelete = useMutation(
    ['contract-edit-employee', new Date()],
    (employee) => deleteEmployee(employee),
    {
      onSuccess: () => {
        queryClient.invalidateQueries(`project-staff-list-${projectId}`);
      },
    },
  );

  const onDelete = (id) => {
    mutationDelete.mutate(id);
  };

  return (
    <>
      <TableContainer>
        <Table
          sx={{
            width: '100%',
          }}
          aria-label="simple table"
        >
          <TableHead>
            <TableRow>
              <StyledTableCell>
                {dictionary.pages.employees.displayName}
              </StyledTableCell>
              <StyledTableCell>
                {dictionary.pages.profile.email}
              </StyledTableCell>
              <StyledTableCell align="right">
                {dictionary.pages.employees.position}
              </StyledTableCell>
              <StyledTableCell>
                {dictionary.pages.employees.manager}
              </StyledTableCell>
              {edit
```

```
      && (
        <StyledTableCell align="right">
          {dictionary.pages.employees.actions}
        </StyledTableCell>
      )}
  </TableRow>
</TableHead>
<TableBody>
  {employees?.map((employee) => (
    <StyledTableRow
      className="table-background-link active"
      key={`row-employee-index-${employee.profileId}`}
      id={`manager-employee-${employee.profileId}`}
      sx={{ '&:last-child td, &:last-child th': { border: 0 } }}
    >
      <StyledTableCell className="row" component="th" scope="row">
        <Link
          component={DomLink}
          to={`/profile/${employee.profileId}`}
          underline="hover"
          color="inherit"
        >
          {`${employee.name} ${employee.surname}`}
        </Link>
      </StyledTableCell>
      <StyledTableCell className="row" component="th" scope="row">
        {employee.email}
      </StyledTableCell>
      <StyledTableCell className="row" align="right">
        {employee.positionName}
      </StyledTableCell>
      <StyledTableCell className="row" align="left">
        { (employee.managerProfileId && (
          <Link
            href={`#manager-employee-${employee.managerProfileId}`}
            underline="hover"
            color="inherit"
          >
            {employee.managerName}
            {' '}
            {employee.managerSurname}
          </Link>
        )) || null}
      </StyledTableCell>
      {edit
        && (
        <StyledTableCell
          sx={{
            padding: '2px',
            marginRight: '5px',
          }}
          align="right"
        >
          <IconButton
            onClick={() => handleOpenChangeModal(
              employee.managerProfileId ? {
                profileId: employee.managerProfileId,
                name: employee.managerName,
                surname: employee.managerSurname,
                email: employee.managerEmail,
              } : null,
              {
                positionId: employee.positionId,
```

```
                          name: employee.positionName,
                        },
                        {
                          profileId: employee.profileId,
                          name: employee.name,
                          surname: employee.surname,
                        },
                        employee.projectStaffId,
                      )}
                      size="small"
                    >
                      <EditIcon />
                    </IconButton>
                    <IconButton          size="small"          onClick={()          =>
onDelete(employee.projectStaffId)}>
                      <DeleteIcon />
                    </IconButton>
                  </StyledTableCell>
                  )}
              </StyledTableRow>
            ))}
          </TableBody>
        </Table>
      </TableContainer>
      {(projectManagers && positions)
        && (
        <EditPositionPMProfileModal
          handleCloseModal={handleCloseChangeModal}
          data={changeModal}
          positions={positions}
          projectManagers={projectManagers}
          projectId={projectId}
        />
        )}
    </>   ); };

EmployeesContentPage.propTypes = {
  employees: PropTypes.arrayOf(
    PropTypes.shape({}),
  ), };

EmployeesContentPage.defaultProps = {
  employees: [],
};

export default EmployeesContentPage;
```

ContractPage.js

```
import React, { useContext } from 'react';
import { Link as DomLink, useParams } from 'react-router-dom';
import { useQuery } from 'react-query';
import Button from '@mui/material/Button';
import AvatarPage from '../../../components/info-page/AvatarPage';
import TitlePage from '../../../components/info-page/TitlePage';
import ContentPage from '../../../components/info-page/ContentPage';
// import DocumentsPage from '../../../components/info-page/DocumentsPage';
import ProjectsPage from '../../../components/info-page/ProjectsPage';
import { LanguageContext } from '../../language/Language';
import ContractTables from '../../../components/admin/contractTables';
import       {       getContractById,       getProjectForContract       }       from
'../../../services/crud/contracts';
import contextUser from '../../../services/context-user';
```

```
const ContractPage = () => {
  const { dictionary } = useContext(LanguageContext);
  const { contractId } = useParams();
  const { user } = useContext(contextUser);
  const role = sessionStorage.getItem('role');

  const c = useQuery(
    `contract-${contractId}`,
    () => getContractById(contractId),
    {
      manual: true,
    },
  );
  const contract = c?.data?.data?.data[0];

  const p = useQuery(
    `contract-projects-${contractId}`,
    () => getProjectForContract(contractId),
    {
      manual: true,
    },
  );
  const projects = p?.data?.data?.data;

  return (
    <div className="main-task-body page-info">
      {contract
        && (
          <>
            <div className="left-bar">
              <AvatarPage
                avatarLink={contract?.avatarLink}
                title={contract?.title}
              />
            </div>

            <div className="content">
              <TitlePage
                type={dictionary.pages.contract}
                title={`${contract?.title}`}
                endAction={
                  ((Number(contract.customerProfileId) === Number(user.id)
                  || Number(contract.companyOwnerProfileId) === Number(user.id))
                  && (
                    <Button
                      component={DomLink}
                      to="edit"
                      variant="contained"
                    >
                      {dictionary.edit}
                    </Button>
                  )) || null
                }
              />

              <div className="block-content">
                <ContentPage
                  dates={{
                    start: contract?.dateStart,
                    end: contract?.dateEnd,
                  }}
```

```
                    pm={{
                      profileId: contract?.companyOwnerProfileId,
                      name: contract?.companyOwnerName,
                      surname: contract?.companyOwnerSurname,
                      email: contract?.companyOwnerEmail,
                    }}
                    customer={{
                      profileId: contract?.customerProfileId,
                      name: contract?.customerName,
                      surname: contract?.customerSurname,
                      email: contract?.customerEmail,
                    }}
                    description={contract?.description}
                    status={contract?.statusName}
                  />
                </div>
                {/* {(contract.documents && contract.documents.length !== 0) */}
                {/*  && <DocumentsPage documents={contract.documents} />} */}

                {(projects && projects.length && role !== 'admin')
                      && <ProjectsPage projects={projects} />}

                {(projects && projects.length && role === 'admin')
                    && <ContractTables projects={projects} />}
              </div>
            </>
          )}
      </div>
    );
};

export default ContractPage;
```

NewContractPage.js

```
import React, { useContext } from 'react';
import { useQuery } from 'react-query';
import TitlePage from '../../../components/info-page/TitlePage';
import         NewContractContentPage         from         '../../../components/info-
page/contract/NewContractContentPage';
import { LanguageContext } from '../../language/Language';
import { getEmployeesByPositions, getStatus } from '../../../services/crud/pages';
import contextUser from '../../../services/context-user';
import Page404 from '../404';

const NewContractPage = () => {
  const { dictionary } = useContext(LanguageContext);
  const { user } = useContext(contextUser);

  const s = useQuery(
    'status-contract',
    () => getStatus('contract'),
  );
  const statusArray = s?.data?.data?.data || [];

  const cs = useQuery(
    'customers-newContract',
    () => getEmployeesByPositions('customer'),
  );
  const customers = cs?.data?.data?.data;

  return ((user.roleName === 'owner'
```

```
    || user.roleName === 'customer')
  && (
    <div className="main-task-body page-info new">
      <div className="content">
        <TitlePage type="" title={dictionary.pages.newContract} />

        {(customers && statusArray && user)
          && (
          <div className="block-content">
            <NewContractContentPage
              customers={customers}
              statusArray={statusArray}
              user={user}
            />
          </div>
          )}
      </div>
    </div>
  )) || <Page404 />;
};

export default NewContractPage;
```

EditContractPage.js

```
import React, { useContext } from 'react';
import { useQuery } from 'react-query';
import { useParams } from 'react-router-dom';
import TitlePage from '../../../components/info-page/TitlePage';
import         EditContractContentPage         from         '../../../components/info-
page/contract/EditContractContentPage';
import EditAvatar from '../../../components/info-page/EditAvatar';
import { LanguageContext } from '../../language/Language';
import { getContractById } from '../../../services/crud/contracts';
import { getEmployeesByPositions, getStatus } from '../../../services/crud/pages';
import contextUser from '../../../services/context-user';
import Page404 from "../404";

//   attached: [
//     {
//       type: 'file',
//       title: 'Terms of reference (T3)',
//       link: '/files/Tasker.docx',
//     },
//   ],

const EditContractPage = () => {
  const { dictionary } = useContext(LanguageContext);
  const { contractId } = useParams();
  const { user } = useContext(contextUser);

  const s = useQuery(
    'status-contract',
    () => getStatus('contract'),
  );
  const statusArray = s?.data?.data?.data || [];

  const c = useQuery(
    `contract-${contractId}-edit`,
    () => getContractById(contractId),
    {
      manual: true,
```

```
    },
  );
  const contract = c?.data?.data?.data[0];

  const cs = useQuery(
    `customers-${contractId}`,
    () => getEmployeesByPositions('customer'),
  );
  const customers = cs?.data?.data?.data;

  return (
    (Number(contract?.customerProfileId) === Number(user.id)
      || Number(contract?.companyOwnerProfileId) === Number(user.id)
      || user.roleName === 'customer')
    && (
    <div className="main-task-body page-info">
      <div className="left-bar">
        {contract
          && (
          <EditAvatar
            profileId={contract.contractId}
            name={contract.title}
            avatarLink={contract.avatarLink}
          />
          )}
      </div>

      <div className="content">
        <TitlePage type="" title={dictionary.pages.editContract} />

        {(contract && customers && statusArray) && (
        <div className="block-content">
          <EditContractContentPage
            contract={contract}
            customers={customers}
            statusArray={statusArray}
            attached=""
          />
        </div>
        )}
      </div>
    </div>
    )
  ) || <Page404 />;
};

export default EditContractPage;
```