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 адміністратора КЗАПР»

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

**Виконавець роботи:** студент групи ІТ-82-0 Глуховцов Дмитро Олександрович

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

Науковий керівник _____ _____к.т.н., доц., Неня В.Г.___
(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

*Глуховцов Дмитро Олександрович*

**1 Тема роботи** *Web-додаток підтримки діяльності адміністратора КЗАПР*

**керівник роботи** *Неня Віктор Григорович, 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 адміністратора 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**_____

## 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** _____ Глуховцов Д.О.
(підпис)

**Керівник роботи** _____ к.т.н., доц. Неня В.Г.
(підпис)

# РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підтримки діяльності адміністратора КЗАПР».

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел із 21 найменування, чотирьох додатків. Загальний обсяг пояснювальної записки складає 89 сторінок, у тому числі 47 сторінок основного тексту, 3 сторінки списку використаних джерел, 39 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку підтримки діяльності адміністратора КЗАПР.

У першому розділі було проведено аналіз аналогів створюваного web-додатку, сформовано їх переваги та недоліки. Розглянуто останні дослідження за тематикою роботи. Визначені засоби реалізації, мету та задачі проекту.

У другому розділі проведено структурно-функціональне моделювання, створено діаграму варіантів використання web-додатку. Спроектовано схему даних.

У третьому розділі було описано архітектуру створюваного програмного продукту та процес його розробки. Сформовано дизайн web-додатку і проведено його функціональне тестування.

Ключові слова: WEB-ДОДАТОК, АДМІНІСТРАТОР, ПРОЕКТ, БАЗА ДАНИХ, JAVASCRIPT, REACTJS, MUI, АВТОМАТИЗАЦІЯ.

# ЗМІСТ

# ВСТУП

У зв’язку зі стрімким розвитком технологій відповідно підвищується й темп перебігу всього навкруги. Додаються різноманітні можливості, що призводить до необхідності ефективного планування робочих процесів у будь-якій сфері сучасного життя. Особливо критичним це є у наш час, коли через певні зовнішні чинники безліч виробництв мають організовувати власну роботу онлайн. І підприємства машинобудівного профілю не є виключенням.

Для таких цілей можна використовувати спеціалізований web-орієнтований комплекс засобів автоматизації проектувальних робіт (КЗАПР). За його допомогою реально вибудовувати робочі взаємозв’язки.

Критично необхідно для організації роботи КЗАПР мати потужну систему адміністрування. Адже для забезпечення безперебійного функціонування має бути людина, яка буде стежити за перебігом різноманітних внутрішніх процесів.

До сфери обов’язків такої особи входить адміністрування баз даних, керування привілеями користувачів, побудова взаємозв’язків між персоналом з боку логіки додатку та субординації всередині компанії. Це основні постулати для підтримки безпеки та логіки виконуваних робіт. А, отже, даний функціонал повинен існувати у системі, яка реалізовує задачі подібні задачам КЗАПРу.

Отже, метою даного дослідження є розробка web-додатку для підтримки діяльності адміністратора у КЗАПР. Його використання дозволить автоматизувати робочу функціональність останнього. Зокрема певні дії адміністратора.

Для досягнення мети проекту потрібно виконати наступні задачі:

- визначити актуальність роботи;
- дослідити предметну область та провести аналіз аналогів web-додатків;
- виконати проектування web-додатку;
- реалізувати структуру та функціонал web-додатку;
- виконати тестування web-додатку.

Результати роботи були апробовані на конференції «ІМА-2022» [1].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд останніх досліджень і публікацій

Кожне виробництво чи компанія прагне до оптимізації та покращення робочих процесів, що позитивно впливатиме на її прибутки та підвищить комфорт працівників разом із посиленням конкурентоздатності.

У зв’язку з цим, можна актуалізувати проблему не лише з обладнанням та устаткуванням, але і відсутність сучасних програмних засобів для автоматизованого проектування технологічних процесів [2].

Використання інформаційних технологій є одним з напрямків оптимізації роботи, що характерно для успішних сучасних організацій, також вони зазвичай користуються спеціалізованими технологіями та програмами, зокрема, ERP-системами. Це об’єднує інформаційний простір підприємства, оптимізує роботу окремих підрозділів та надає можливість для коригування діяльності на кожному етапі виробничого процесу з допомогою коригування різних факторів та аналізу цих змін [3].

У загальному вигляді, ERP-система є програмним комплексом для управління компанією, що працює на єдиній технологічній платформі, опирається на єдину базу даних і в реальному часі проводять між собою синхронізацію [4].

Ще одним способом для створення та організації подібного функціоналу є САПР. Це організаційно-технічна (людино-машинна) система, що складається з комплексу засобів автоматизації проектування, взаємозв’язаних із необхідними відділами проектної організації чи колективом фахівців (користувачем системи), і виконує автоматизоване проектування [5].

У більшості випадків такі системи організовуються як web-додатки на базі web-інструментів та можливостей. Використовують саме такий підхід у зв’язку з низкою наступних переваг, які надає даний підхід:

– простота доступу до додатка: будь-яка людина, яка має комп’ютер, підключений до мережі Інтернет, може використовувати web-додаток;

– простота розгортання (установки): на відміну від локальних додатків немає необхідності в фізичній установці на комп’ютер користувача. Після виходу оновлення системи всі одразу мають доступ до найновішої версії;

– високий рівень розвитку та надійності мережевих з’єднань та web-технологій [6].

Є очевидним той факт, що для таких систем є необхідним адміністратор, який буде виконувати специфічні дії, пов’язані з внутрішньою частиною процесів web-додатку. Він може бути відповідальним за забезпечення безпеки останнього та впровадження політики безпеки оброблюваної інформації. До його обов’язків також може входити внесення змін та оновлення web-додатку [7].

Для реалізації web-додатку було обрано такі технології, як JavaScript бібліотеку React [8], яка спрощує створення інтерактивних інтерфейсів, надає можливості для покращення швидкості розроблюваних систем, робить їх простішими та масштабованішими. Каскадні таблиці стилів CSS для гнучкості та адаптивності дизайну. Скриптову метамову Sass, що спрощує файли каскадної таблиці стилів і надає більше можливостей та свободи при написанні CSS [9]. Також для графічної частини використовується бібліотека MUI, що надає широкий спектр інструментів для розробки інтерфейсів та має готові модулі для побудови. Мову програмування JavaScript для асинхронної роботи з серверною частиною. Для розробки бази даних було обрано PostgreSQL – це передова система реляційних баз даних корпоративного класу з відкритим вихідним кодом, вона підтримує декілька видів запитів, а саме SQL та JSON [10].

Загалом, проаналізувавши дані, можна дійти висновку, що дана робота дійсно є важливою та актуальною у сучасних реаліях. Також стає зрозумілим, що обрання способу реалізації – web-додаток повною мірою задовольнить потреби кінцевого користувача, разом з тим надаючи необхідний функціонал та комфорт.

## 1.2 Аналіз існуючих продуктів-аналогів

Усе частіше й частіше в мережі Інтернет стають доступними додатки для розробки проектів та контролю над ними. Вони забезпечують досить широкий функціонал для планування, розподілу та відслідковування різноманітних задач, проте, звісно, вони не надають деякі специфічні функції для вузьконаправлених сфер та виробництва.

Для формування вимог створюваного програмного продукту було розглянуто та досліджено існуючі аналоги подібних web-додатків, а саме «Trello», «Asana» та «Jira».

«Trello» [11] – це універсальний інструмент для управління проектами та процесами з ними пов’язаними. Він доволі гнучкий та дозволяє адаптуватися до кожної робочої ситуації та необхідності.

Після завантаження головної сторінки ти потрапляєш до вибору проекту з їхнім переліком (рис. 1.1).

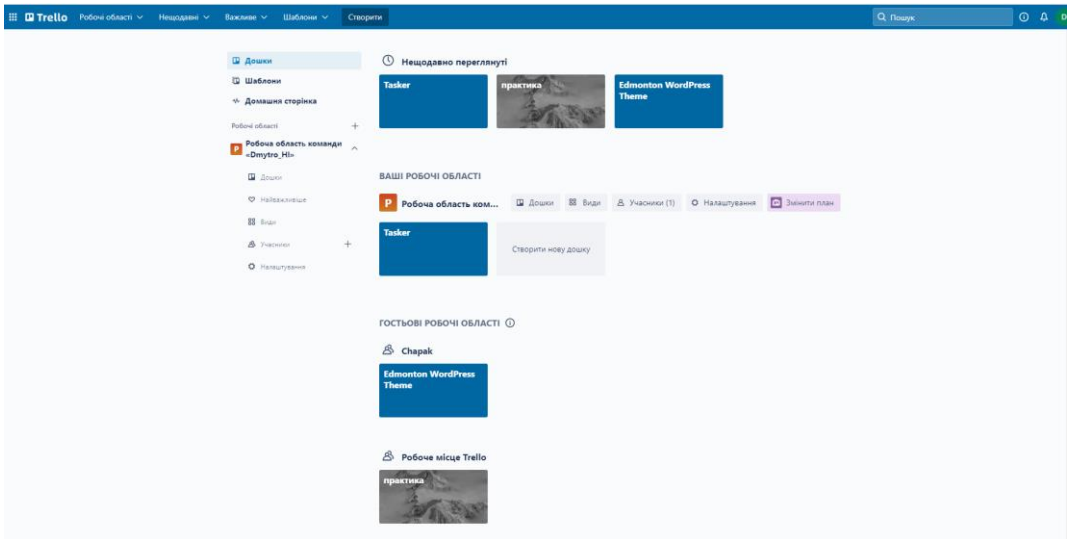



Рисунок 1.1 – Головна сторінка «Trello»

Після вибору проекту користувач знаходиться на сторінці з задачами (рис. 1.2), які він може переводити в різноманітні ним визначені статуси. Також можливим є

проведення певних операцій всередині проекту. Наприклад, створення задач, редагування їхніх параметрів, додавання різних вкладеннь тощо (рис. 1.3). Також є можливість додавати людей до власних проектів за згенерованим посиланням обмежувати їхні права доступу.

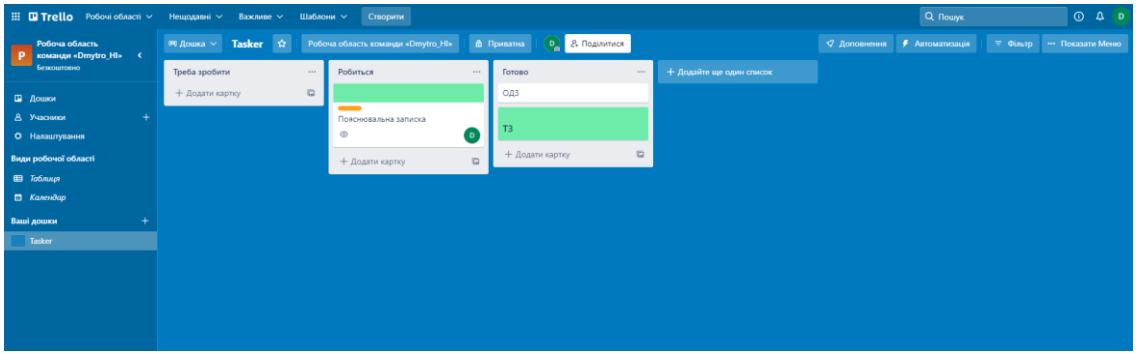



Рисунок 1.2 – Сторінка з задачами

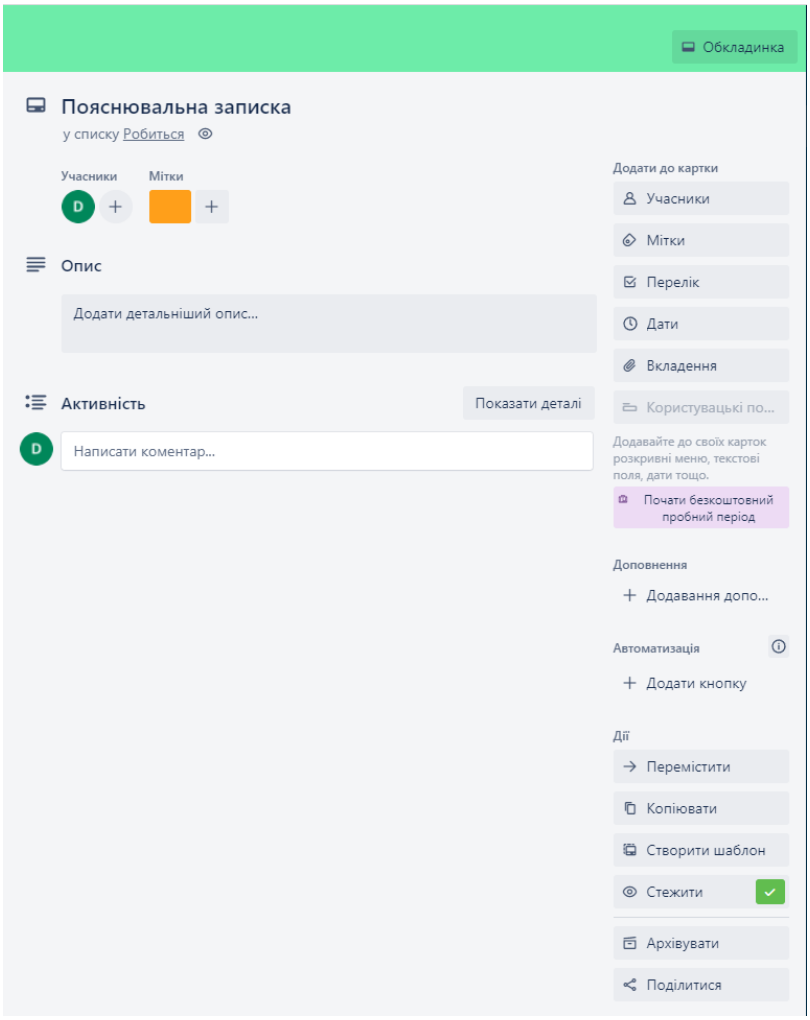



Рисунок 1.3 – Вікно для редагування задачі

Дизайн web-додатку можна охарактеризувати як сучасний, стриманий та приємний. Є можливість налаштування кольорів на власний смак. Також дизайн є доволі інтуїтивно зрозумілим і навіть людина без особливих навичок зможе швидко розібратися з специфікою використання.

«Asana»[12] – це  web-додаток керування роботою. Він розроблений для допомоги організації робочих команд. Також за її допомогою можна відстежувати й керувати задачами проекту. Після реєстрації та вибору початкових параметрів для системи, користувач опиняється на головній сторінці, де знаходяться всі його поточні завдання (рис. 1.4).

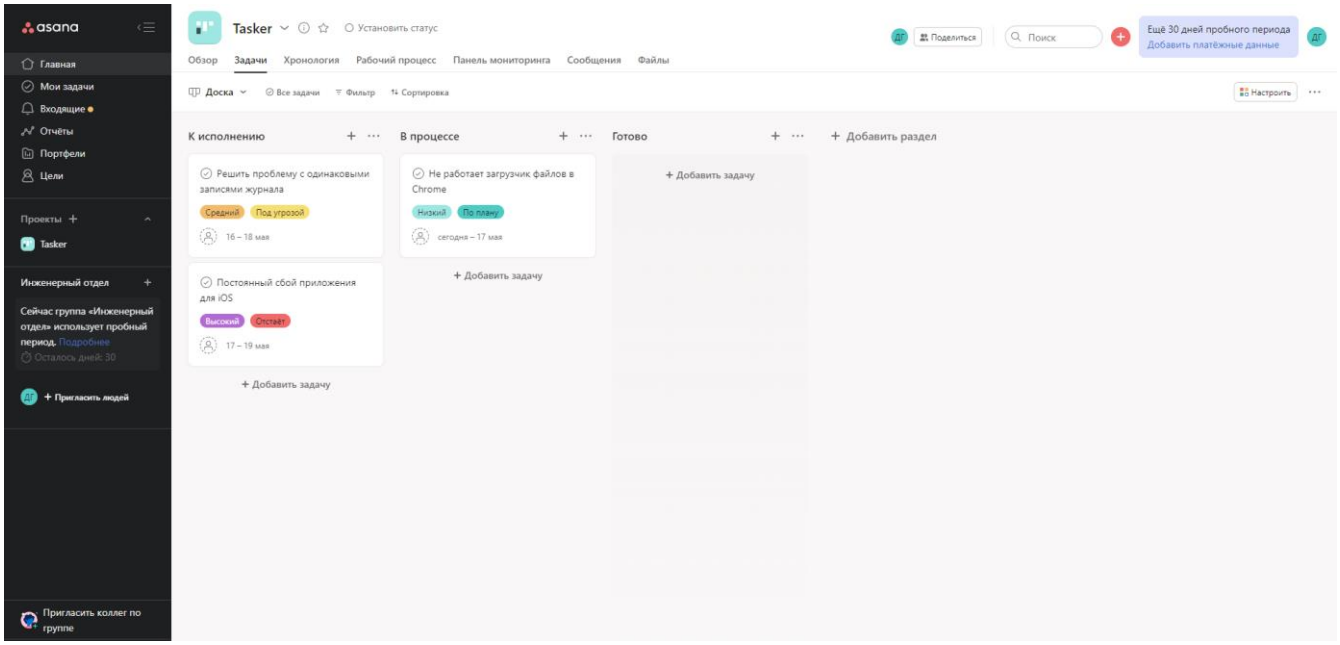



Рисунок 1.4 – Головна сторінка «Asana»

У даному сервісі також є можливість виконувати багато дій зі створеними задачами, тобто редагувати, призначати виконавців, переміщати в різноманітні стани виконання тощо (рис. 1.5).

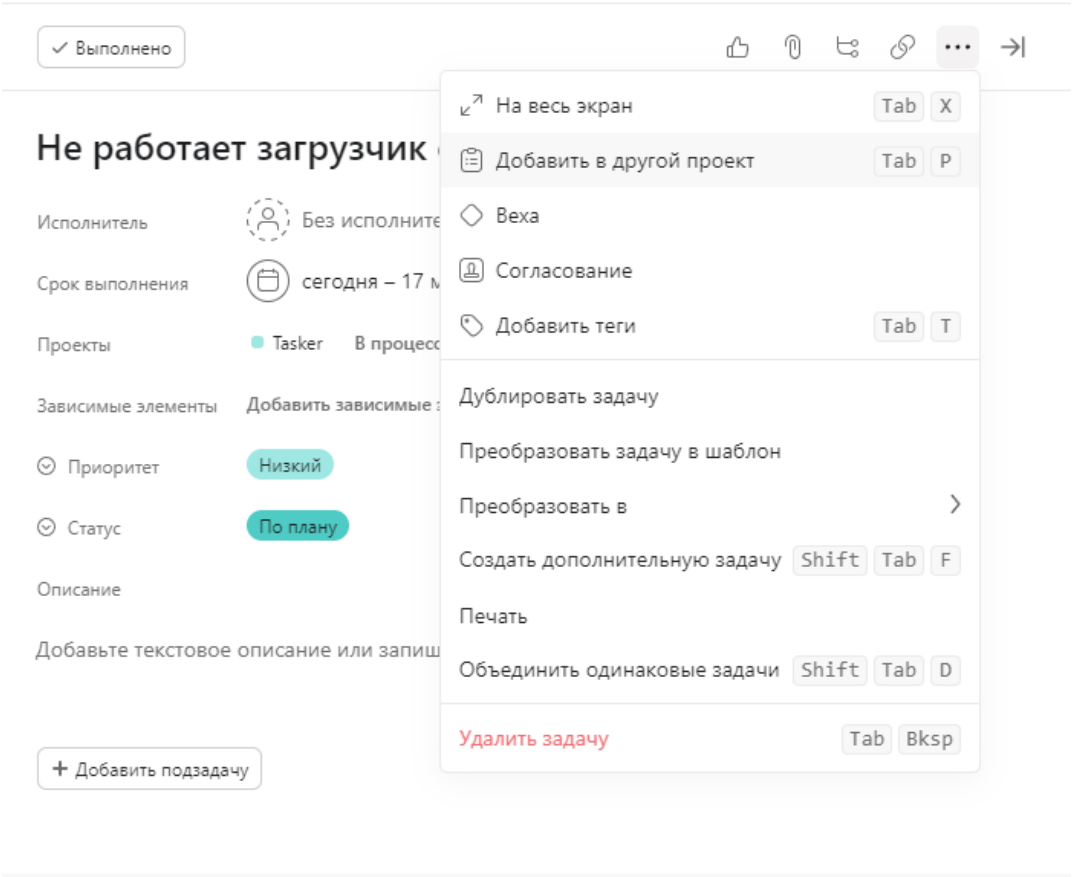



Рисунок 1.5 – Меню для редагування задачі в «Asana»

Перегляд різноманітної статистики також є доступним (рис. 1.6). Є можливість додавати користувачів до проекту та задач за допомогою посилань.

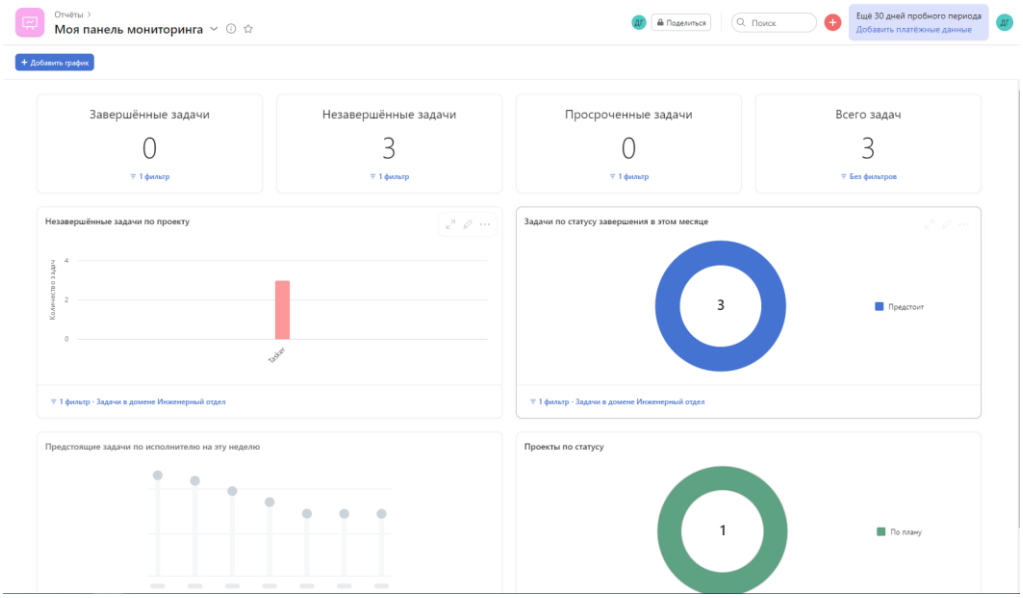



Рисунок 1.6 – Сторінка зі статистикою в «Asana»

Дизайн даної web-платформи можна описати як сучасний із приємною кольоровою гаммою, яка відповідає сучасним трендам та концепціям побудови. Також доволі зручний та інтуїтивно зрозумілий для нових користувачів інтерфейс.

«Jira Work Management»[13] – продукт, що є досить потужним інструментом для організації робочого процесу. Після налаштування профілю тієї чи іншої компанії користувач потрапляєте на головну сторінку, де будуть розміщуватися всі задачі (рис. 1.7).

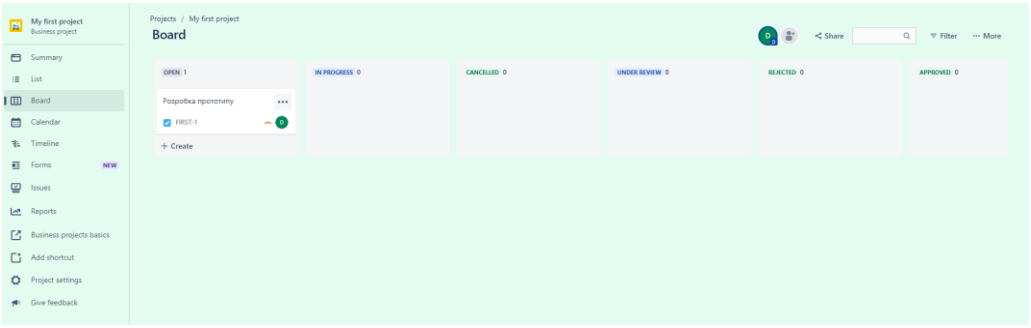



Рисунок 1.7 – Дошка з задачами в «Jira»

Звичайно є можливість редагувати задачу та виконувати різноманітні операції з нею в окремому вікні (рис. 1.8).

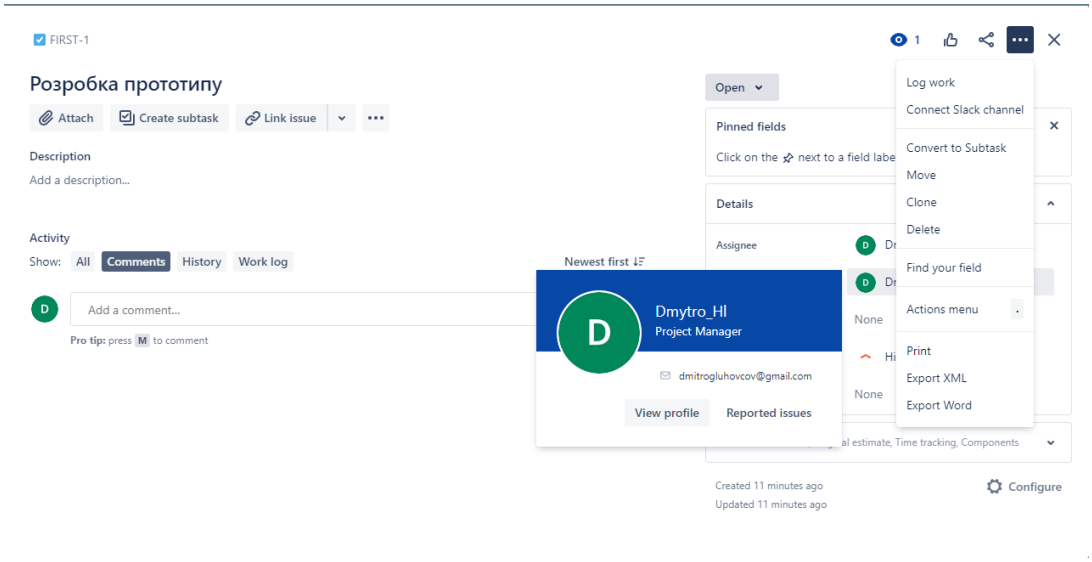



Рисунок 1.8 – Меню редагування задачі в «Jira»

Також є доступним створення певних статистичних звітів із діаграмами. Є можливість запрошувати людей через різноманітні соціальні мережі та облікові записи Google.

Ще одним досить цікавим для розгляду інструментом є функція редагування доступів і прав користувачів. Це є основним функціоналом для адміністрування внутрішніх процесів. Меню з налаштуваннями зображено на рисунку 1.9.

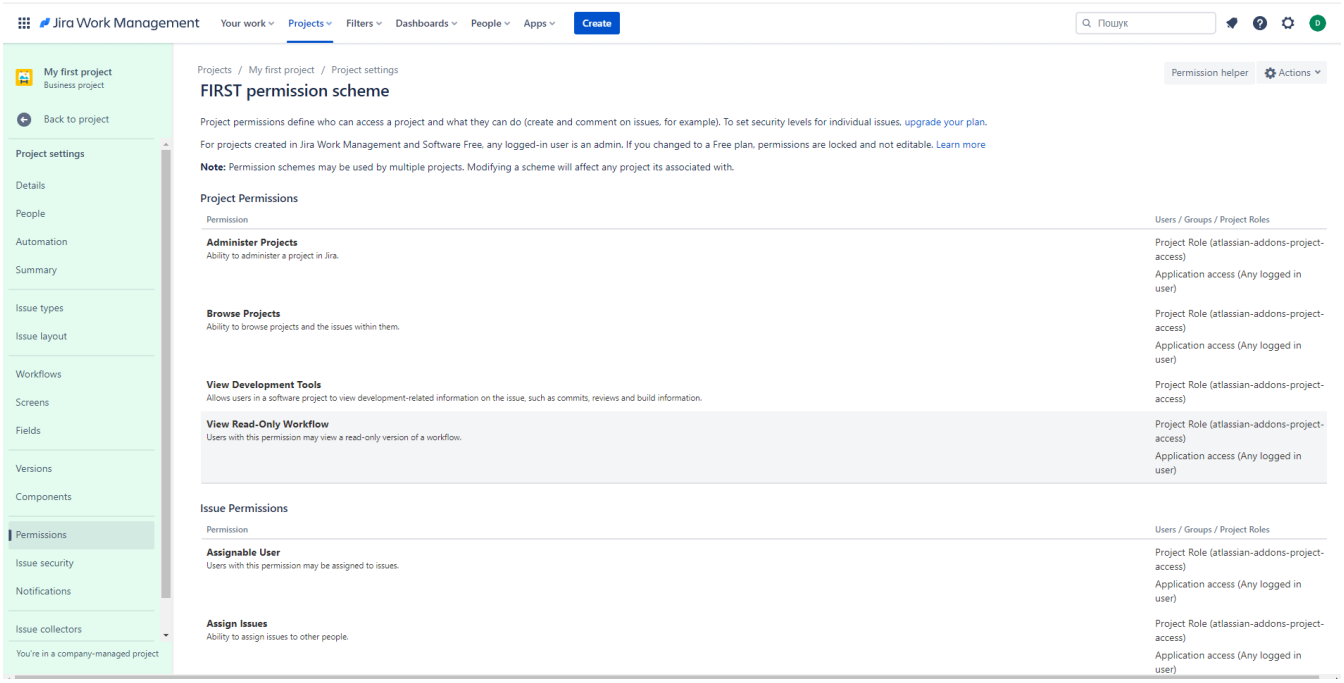



Рисунок 1.9 – Меню з налаштуваннями доступу

Дизайн web-системи є сучасним, кольорове оформлення є досить приємним та має можливість налаштовуватися індивідуально. Інтерфейс є дещо заплутаним для нового користувача, але якісні навчальні матеріали нівелюють цю проблему.

Після детального аналізу аналогів зі схожими функціями та призначенням, було сформовано порівняльну таблицю з перевагами та недоліками переглянутих web-додатків. Результати представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця продуктів-аналогів

| **Характеристика/ Web-додаток** | **«Trello»** | **«Asana»** | **«Jira»** |
|---|---|---|---|
| Сучасний та приємний дизайн | + | + | + |
| Інтуїтивно зрозумілий інтерфейс | + | + | - |
| Наявність основного функціоналу | + | + | + |
| Навігація | + | + | + |
| Авторизація користувачів | + | + | + |
| Перегляд статистики | - | + | + |
| Перегляд історії змін у задачі | + | + | + |
| Можливість запрошувати співробітників до проекту | + | + | + |
| Розгорнуте налаштування розмежування доступу | - | - | + |
| Можливість налаштування взаємодії між персоналом типу «Начальник-Підлеглий» | - | - | - |
| Сповіщення | + | + | + |

Дані з таблиці 1.1 допомагають розібратися з основними моментами, на які необхідно звернути увагу під час проектування та розробки web-додатку. У першу чергу необхідно опиратися на спільні позитивні сторони розглянутих аналогів та намагатися доопрацювати специфічні можливості для окремих випадків використання. До важливих функціональних компонентів, які мають бути присутніми у розроблюваному web-додатку необхідно віднести таке, як зручний інтерфейс, достатній функціонал, який покриває потреби користувача, зокрема створення та маніпулювання зада-

чами, можливості керування персоналом та спеціальний інструментарій для контролю за обмеженнями доступу, можливості реєстрації нових користувачів та надання доступу до проекту за посиланнями.

Також необхідно врахувати всі недоліки, які були знайдені та можуть бути ще поміченими під час розробки. А саме надання можливості налаштовувати робочі взаємозв'язки між співробітниками, додавання функціоналу для розширених налаштувань із розмежуванням доступу до ресурсів та інформації окремим користувачам. Тому є потреба у створенні такого web-додатку, який би подолав виявлені недоліки.

## 1.3 Постановка задачі

Метою цього дослідження є розробка web-додатку для підтримки діяльності адміністратора КЗАПР. Цей програмний продукт повинен забезпечити належну організацію функціоналу для адміністрування даного комплексу та оптимізувати робочі процеси за рахунок автоматизації відповідних процесів.

Основними вимогами до створюваного web-додатку є наступні:

– створення зручного та інтуїтивно зрозумілого інтерфейсу;

– забезпечення функціоналу контролю над обліковими записами користувачів;

– організація можливості розподілу прав та обов'язків;

– можливість редагувати певну інформацію в базі даних.

Для досягнення мети проекту потрібно виконати наступні задачі:

– визначити актуальність та дослідити предметну область;

– розглянути існуючі розробки з подібної тематики у джерелах;

– провести аналіз аналогів та виконати їхню порівняльну характеристику;

– дослідити та обрати технології для проектування та розробки;

– провести прототипування та змоделювати структуру web-додатку;

– реалізувати спроектований web-додаток;

– імплементувати функціонал для роботи адміністратора КЗАПР;

– провести тестування web-додатку.

Загальні вимоги до проекту, структури web-додатку, засобів для забезпечення його працездатності описано у технічному завданні (Додаток А).

# 2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

## 2.1 Структурно-функціональне моделювання

IDEF0 – це методологія графічного опису систем і процесів діяльності організації як безлічі взаємозалежних функцій. Використовуючи її, є можливість дослідити функції об’єкта, що розглядається без прив’язки до конкретної реалізації [14].

За цим стандартом входи ідентифікують потоки інформації та отриманих матеріалів, які надходять до бізнес-процесу. Усі інші стрілки відповідають за демонстрації відповідних ресурсних потоків, призначених для перетворення у досягнення. На виході процес отримує конкретний результат, що відповідає кінцевій точці маніпуляцій усередині бізнес-процесу. Функціональне моделювання web-додатку для підтримки діяльності адміністратора КЗАПР в IDEF0 продемонстроване на рисунку 2.1.

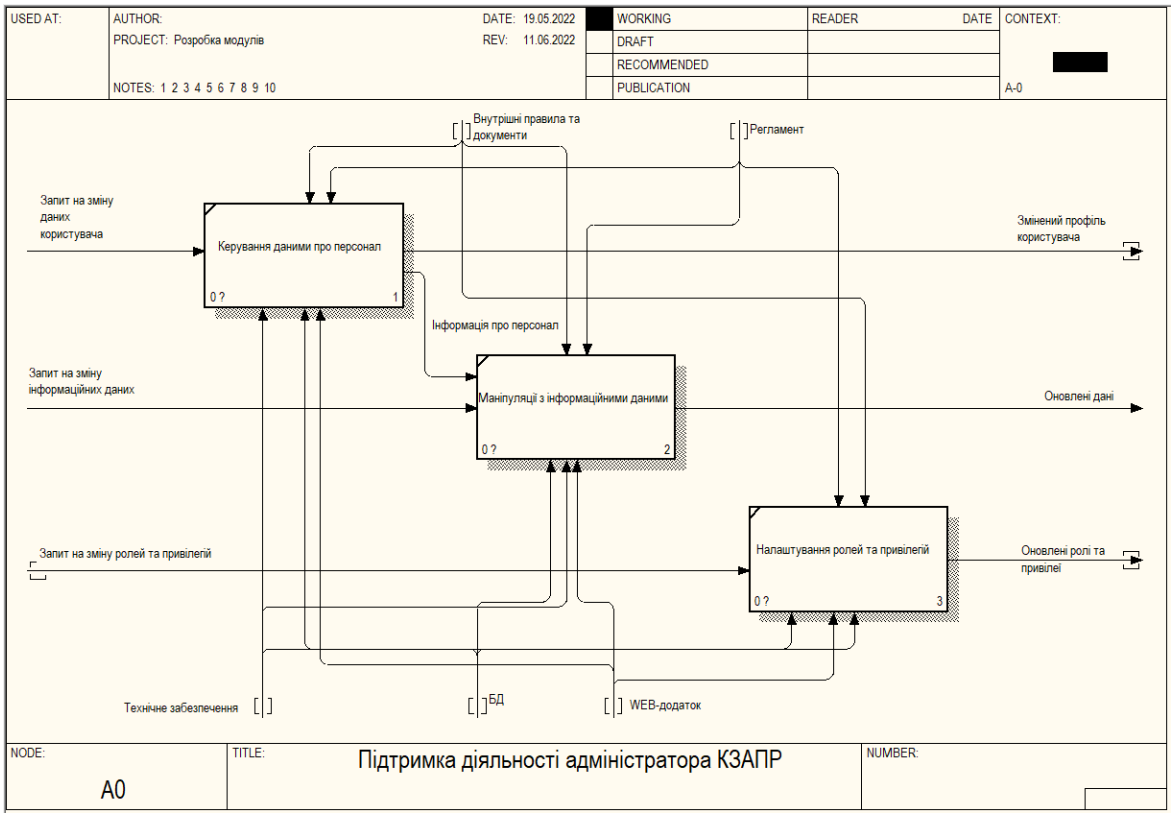



Рисунок 2.1 – Діаграма IDEF0 web-додатку для підтримки діяльності адміністратора КЗАПР

Для конкретизації внутрішніх процесів діаграми IDEF0 її було декомпозовано. Здійснення розбиття на менші функціональні компоненти для більшої наочності внутрішніх процесів та потоків даних, що вони потребують. Діаграма зображена на рисунку 2.2.

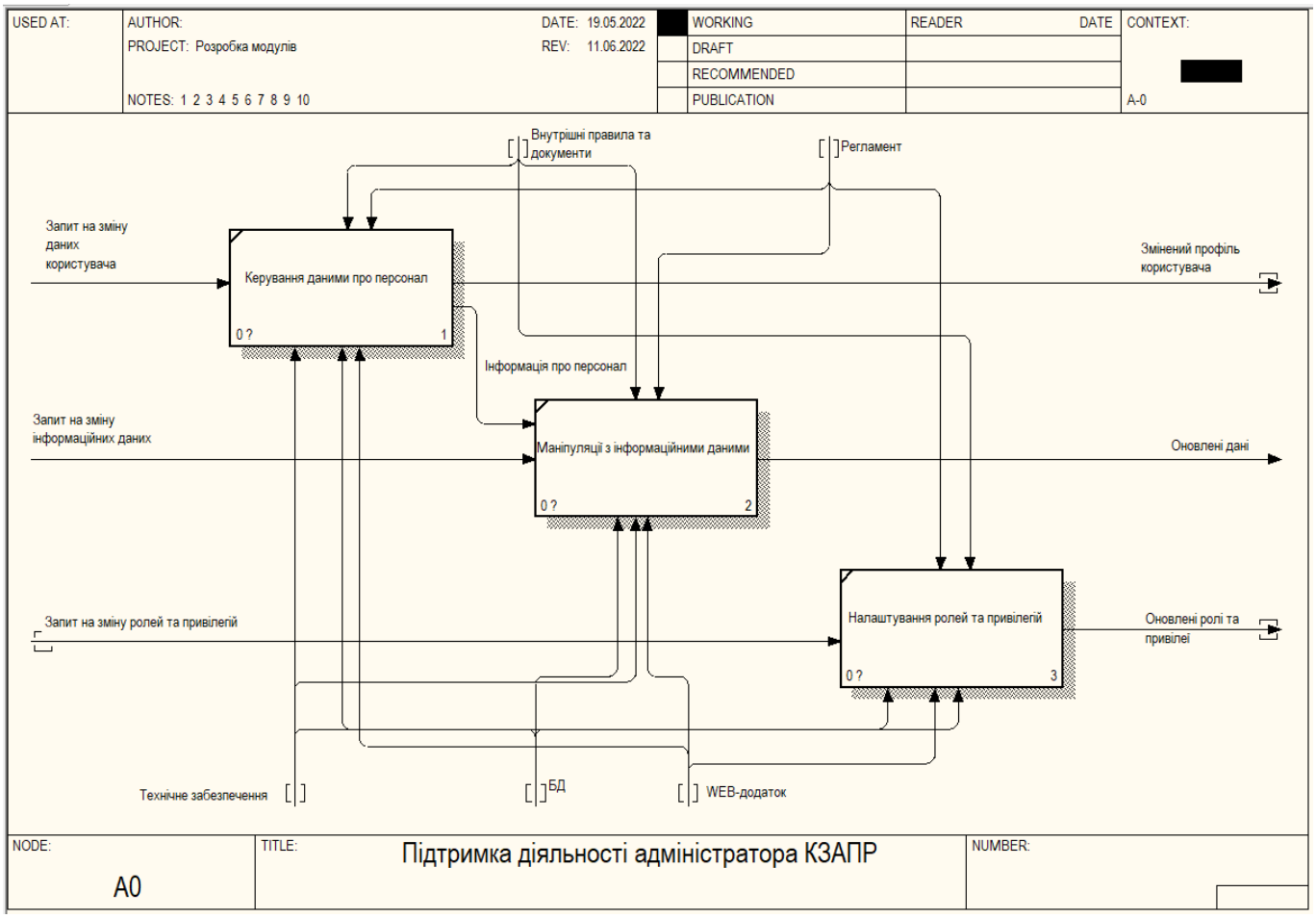



Рисунок 2.2 – Діаграма декомпозиції процесу забезпечення основного функціоналу для адміністратора КЗАПР

Для структуризації та узагальнення даних, зображених на діаграмі декомпозиції сформовано таблицю з функціональними блоками та відповідними їм стрілками – потоками даних (табл. 2.1). Було виділено наступні підпроцеси:

– керування персоналом;
– маніпуляції з інформаційними даними;
– налаштування ролей та привілеїв.

Таблиця 2.1 – Дані для діаграми декомпозиції процесу забезпечення основного функціоналу для адміністратора КЗАПР

| **Стрілка/ Підпроцес** | **Вхідні дані** | **Управління** | **Механізми** | **Вихідні дані** |
|---|---|---|---|---|
| Керування даними про персонал | Запит на зміну даних користувача | Внутрішні правила та документи. Регламент | БД. WEB-додаток. Технічне забезпечення | Змінений профіль користувача. Інформація про персонал |
| Маніпуляції з інформаційними даними | Запит на зміну інформаційних даних | | | Оновлені дані |
| Налаштування ролей та привілеїв | Запит на зміну ролей та привілеїв | | | Оновлені привілеї |

## 2.2 Моделювання варіантів використання

UML (англ. Unified Modeling Language) – це мова графічного опису об’єктного моделювання, яка зазвичай застосовується під час розробки програмного забезпечення для дизайну різноманітних бізнес-процесів, проектування та відображення певних систем та структур [15].

Розробка такої моделі відбувається за допомогою стандартизованих графічних позначень. Зазвичай вона використовується для наступних процесів, пов’язаних з системами: визначення, візуалізація, проектування та документування.

Візуальне моделювання складається з пов’язаних етапів. Розпочинається цей процес із побудови абстрактної концептуальної схеми. Після треба перейти до фізичної та логічної моделей.

Для їх реалізації необхідно побудувати модель у формі так званої діаграми варіантів використання (use-case diagram) [16]. Вона призначена для опису функціональних можливостей системи. На ній зазвичай зображені актори, тобто користувачі, які взаємодіють із системою. Також продемонстровано функціональні вимоги та цілі, яких планується досягти.

Для web-додатку підтримки діяльності адміністратора до варіантів використання відносяться наступні:

- керування користувачами;
- контроль основних сутностей;
- відповідь на запити;
- керування ролями;
- зміна особистих даних користувачів.

Діаграма варіантів використання web-додатку підтримки діяльності адміністратора у КЗАПР представлена на рисунку 2.3.

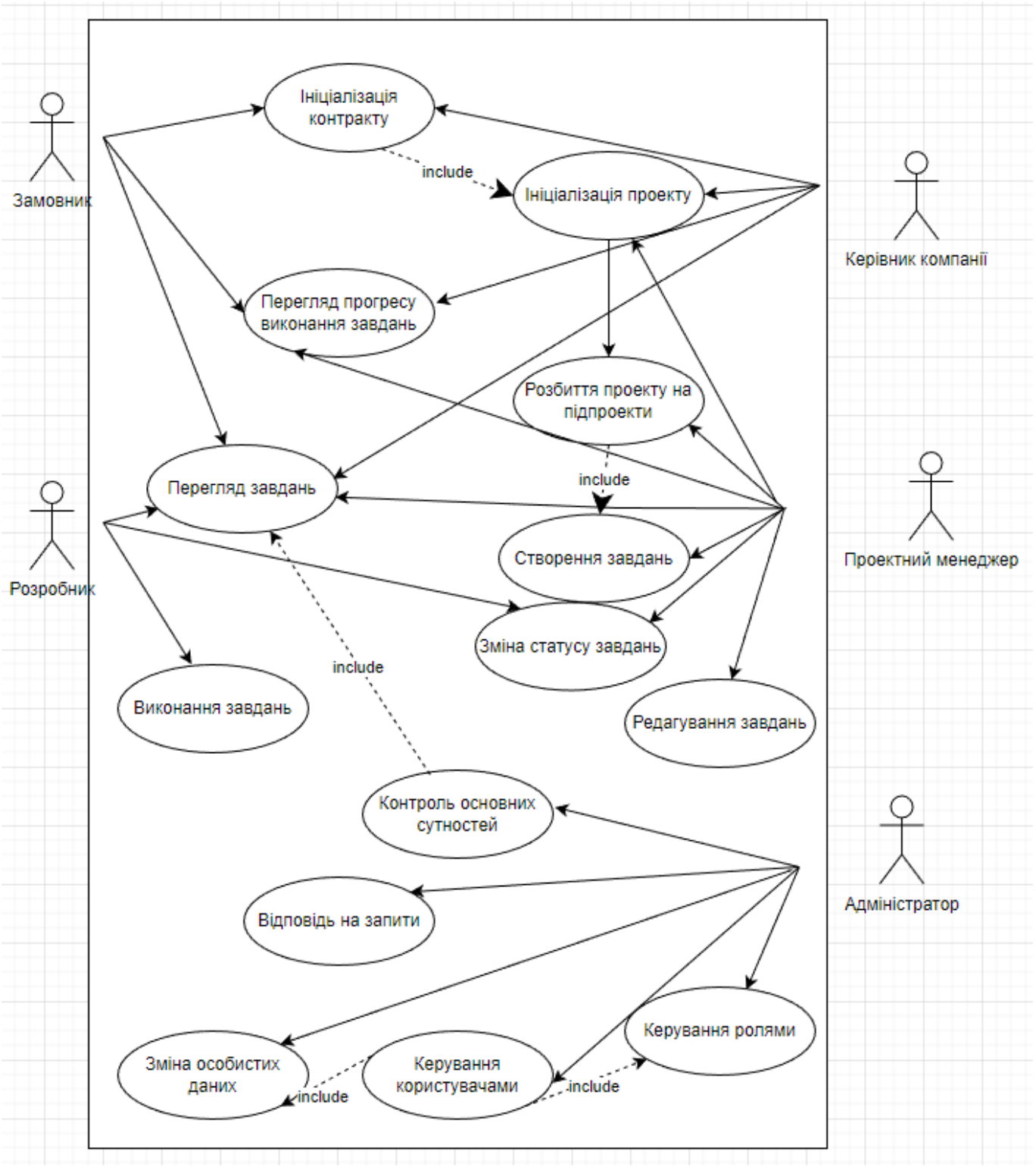



Рисунок 2.3 – Діаграма варіантів використання

## 2.3 Проектування бази даних

Після створення основного каркасу проекту необхідно розробити сховище для даних, яке буде зберігати всю інформацію, що надходить до системи та таку, що необхідна для її функціонування.

База даних (БД) є досить важливою для адміністрування будь-якого web-додатку. Це так, оскільки структурована та зібрана в одному місці інформація значно полегшує всі процеси, пов’язані з обробкою даних та різноманітними маніпуляціями з системними сутностями.

У результаті проектування БД визначається кількість таблиць, їхній склад та побудова взаємозв’язків між ними. Кожна з них має власну структуру, тобто стовпці, що визначають тип, розмір та особливості даних, які будуть зберігатися.

Під час проектування та розробки бази даних було виділено наступні сутності:

– Профіль (Profile) – містить особисту інформацію про користувача;

– Користувач (User) – містить інформацію для ідентифікації користувачів у системі;

– Проект (Project) – містить повну інформацію про проекти;

– Персонал проекту (ProjectStaff) – допоміжна таблиця для поєднання проектів з працівниками, зберігає інформацію про ієрархію взаємозв’язків на проекті;

– Контракт (Contract) – зберігає інформацію про проект;

– Ролі (Roles) – зберігає інформацію про існуючі ролі і допомагає в розмежуванні доступу та відокремленні функціоналу;

– Правила(Rules) – тут знаходиться інформація про наявні правила, за якими відбувається розмежування доступу на серверній частині;

– Роль профілю (ProfileRoles) – допоміжна таблиця, що зберігає інформацію про ролі користувача;

– Логи (Logs) – таблиця для логування;

– Доступ (Permission) – таблиця з описом можливих доступів;

– Ресурс (Resource) – зберігає інформацію про наявні в системі ресурси для застосування в правилах.

– Статус(Status) – таблиця з списком всіх можливих статусів для різних типів ресурсів;

– Завдання(Task) – містить вичерпну інформацію про завдання;

– Список завдань (TaskList) – допоміжна таблиця для поєднання профілю та завдань, зберігає інформацію про людей причетних до завдання;

– Коментар (Comment) – містить інформацію про коментар, завдання, до якого він був закріпленим та ким;

– Запити в підтримку (Support_Requests) – містить сам запит в підтримку та ким він був залишений;

– Сповіщення (Notification) – містить інформацію про сповіщення та його пріоритет;

– Список сповіщень (NotificationList) – допоміжна таблиця для поєднання сповіщень та їхніх адресатів, містить інформацію про отримувача повідомлення та статус;

– Вкладення (Attachment) – містить посилання на вкладення та інформацію про тип вкладення;

– Список вкладень (AttachmentList) – поєднує вкладення з ресурсами, до яких його можна залишити, містить інформацію про ресурс до якого було зроблено вкладення.

На рисунку 2.4 зображено логічну модель бази даних web-додатку підтримки діяльності адміністратора у КЗАПР.

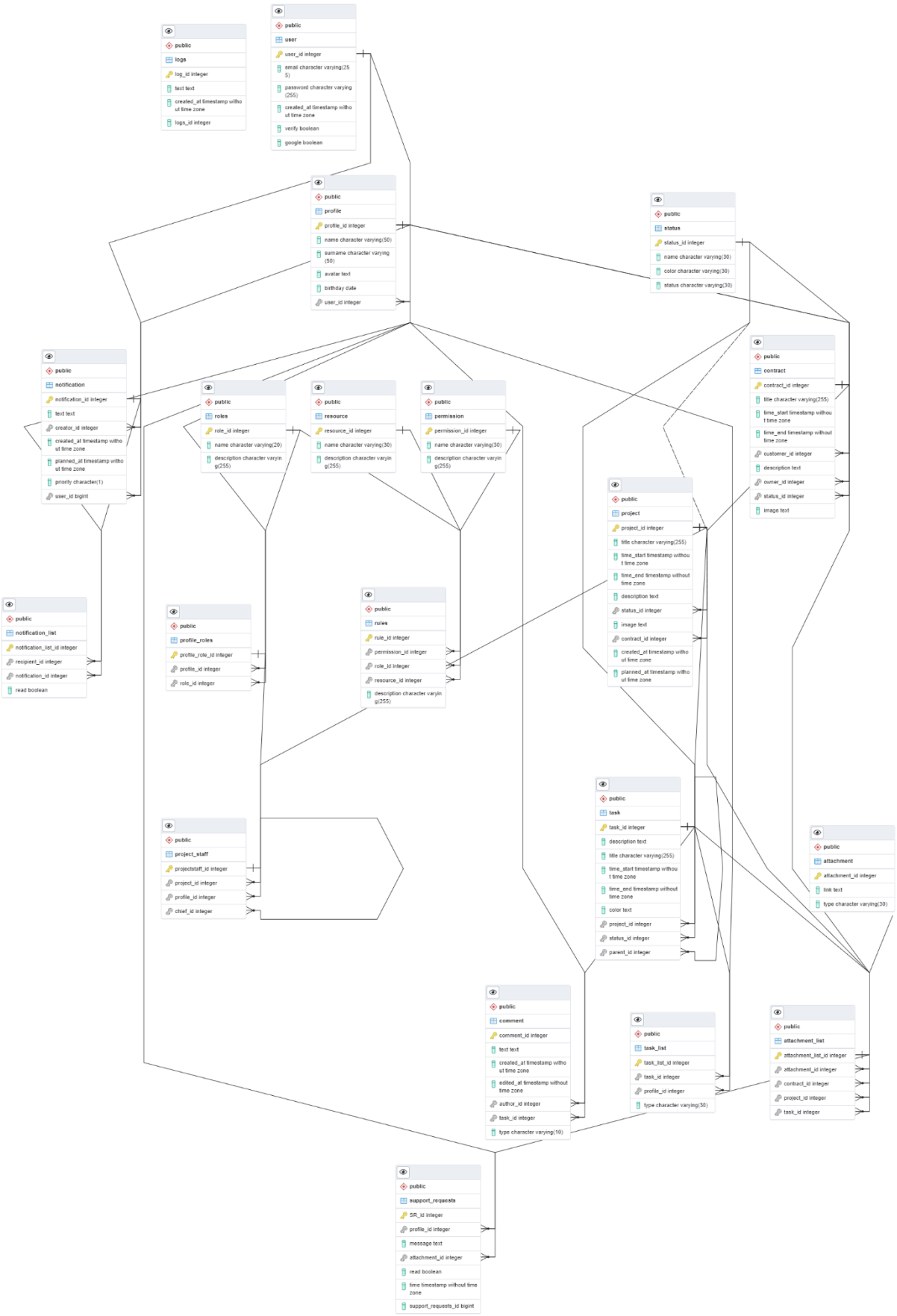



Рисунок 2.4 – Логічна модель бази даних

# 3 РОЗРОБКА WEB-ДОДАТКУ ПІДТРИМКИ ДІЯЛЬНОСТІ АДМІНІСТРАТОРА КЗАПР

## 3.1 Архітектура web-додатку

Перед початком програмної реалізації необхідно розробити архітектуру web-додатку. Вона описує його будову. Архітектура web-додатку допомагає зрозуміти з яких модулів розробка буде складатися. Також надається інформація щодо їх взаємодії між собою. Дана архітектура містить два модулі. Це клієнт і сервер, який в свою чергу поділяється на безпосередньо серверні процеси та базу даних. Схема архітектури web-додатку зображена на рисунку 3.1.

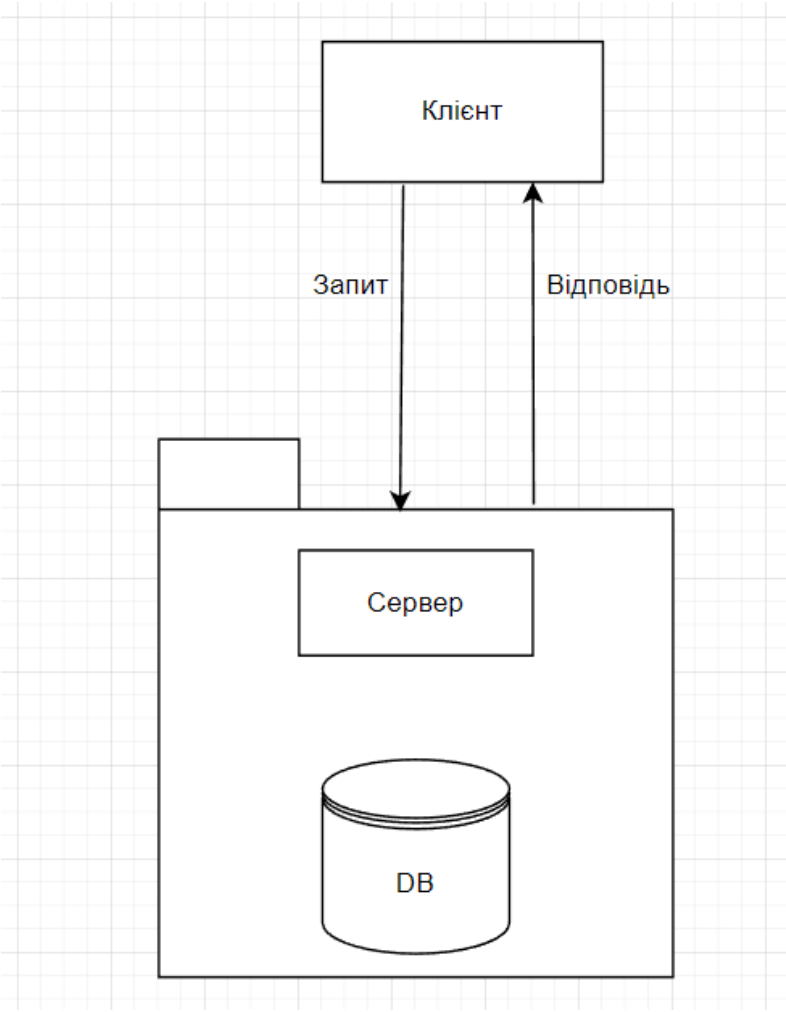



Рисунок 3.1 – Архітектура web-додатку

## 3.2 Розробка дизайну web-додатку

Наступним кроком є проектування макету та продумування дизайну сторінок розроблюваного web-додатку. Для цього необхідно визначити їх структуру, взаємне розміщення компонентів та кольорову гамму. Відповідно до вимог, описаних у Додатку А було створено макет типової сторінки даної розробки (рис. 3.3).

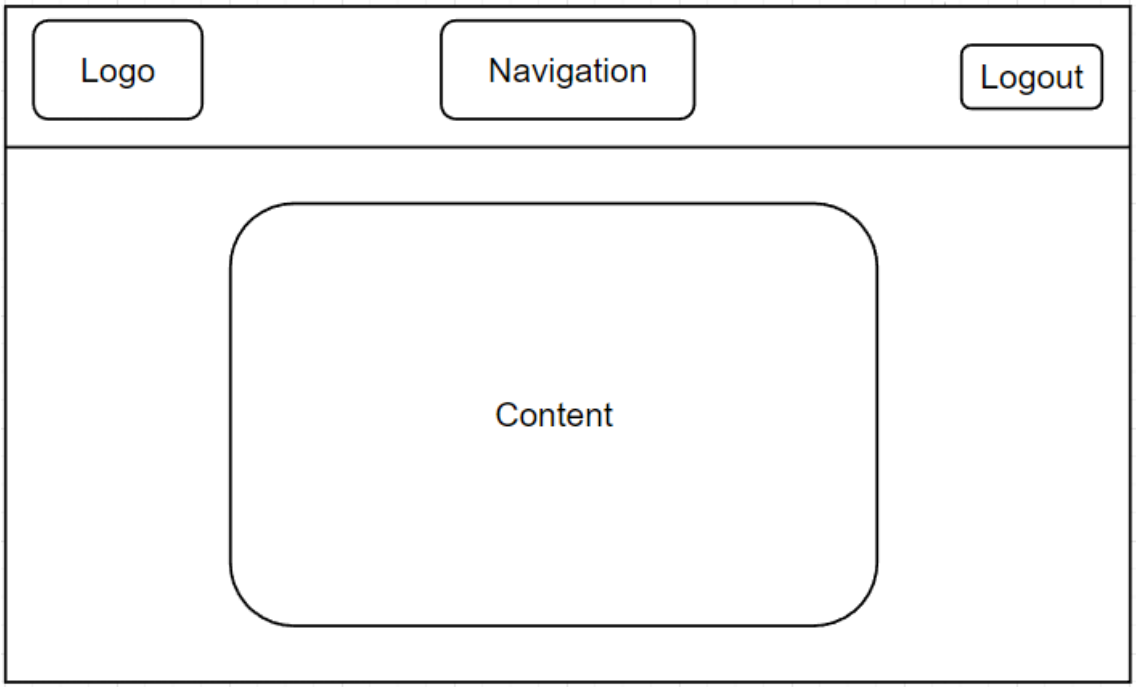



Рисунок 3.3 – Макет типової сторінки web-додатку

Після перегляду та аналізу типових web-додатків було реалізовано дизайн даного програмного продукту, опираючись на вигляд його основної частини. Графічний макет сторінок реєстрації та авторизації зображено на рисунках 3.4-3.5.

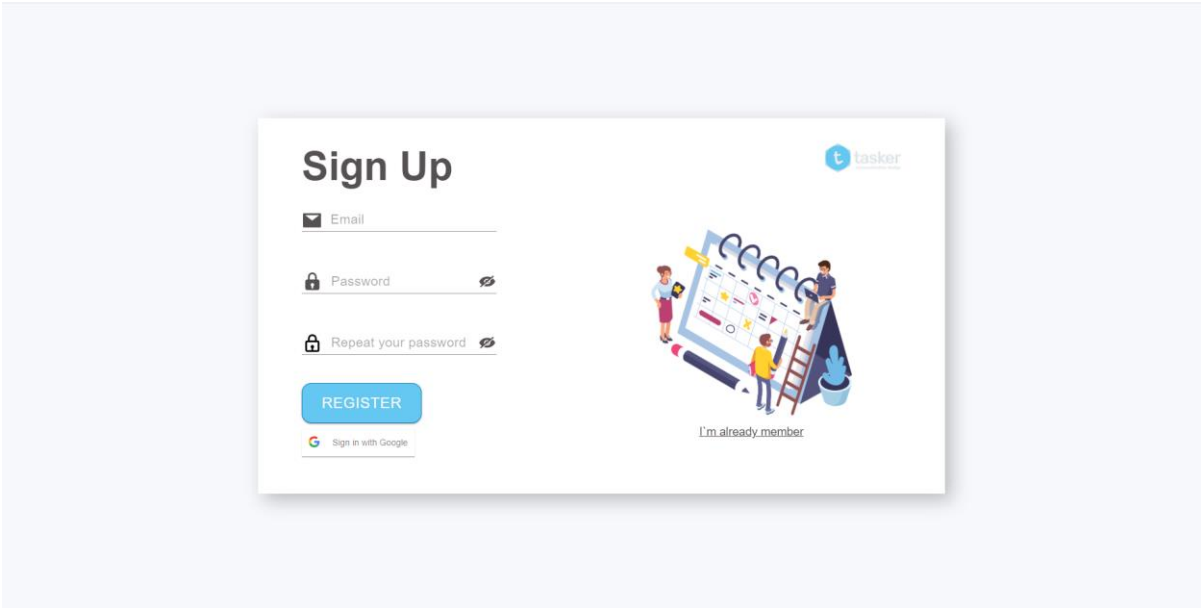



Рисунок 3.4 – Зовнішній вигляд сторінки реєстрації

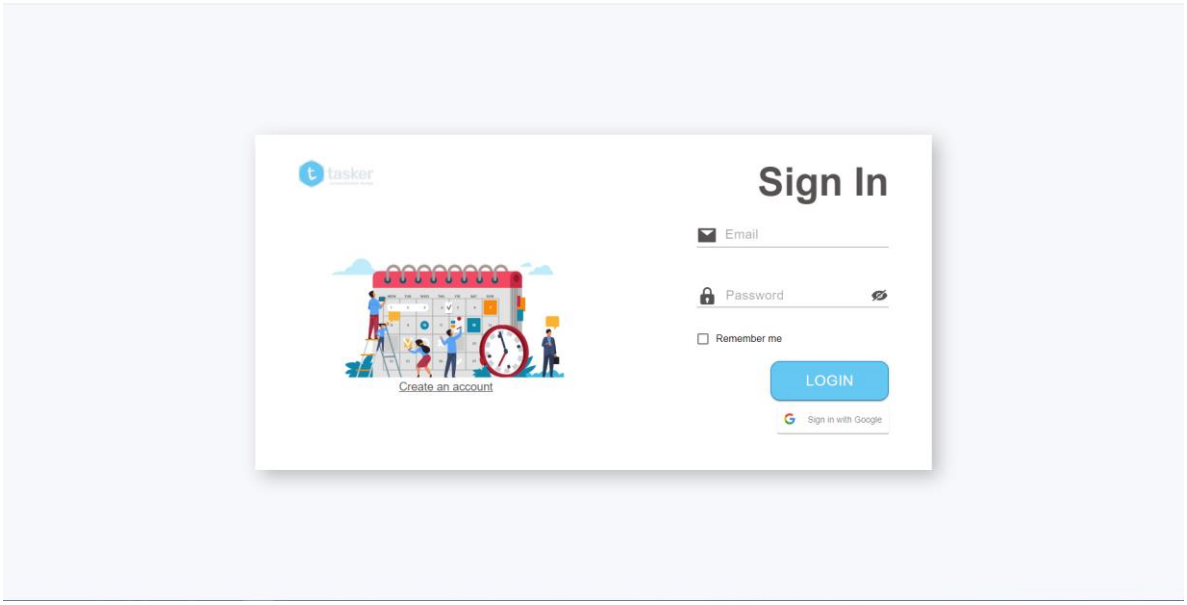



Рисунок 3.5 – Зовнішній вигляд сторінки авторизації

Дизайн сторінки для перегляду наявних зареєстрованих користувачів у системі та інформації про окремого з них зображено на рисунках 3.6 та 3.7 відповідно.

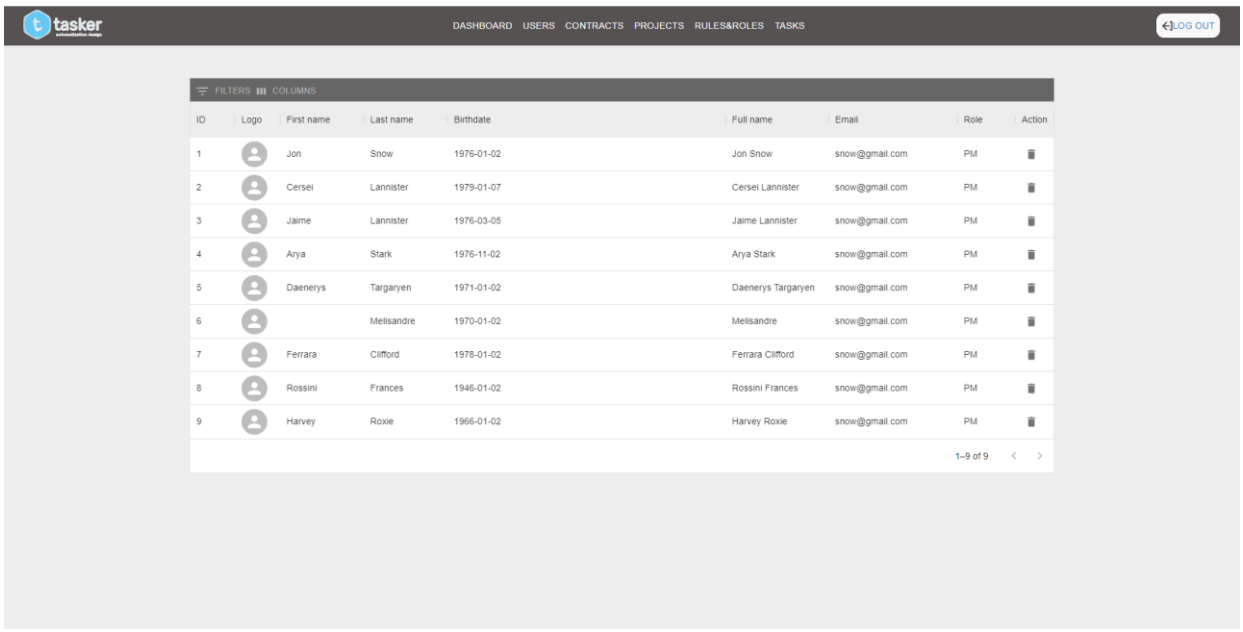



Рисунок 3.6 – Зовнішній вигляд сторінки перегляду зареєстрованих користувачів

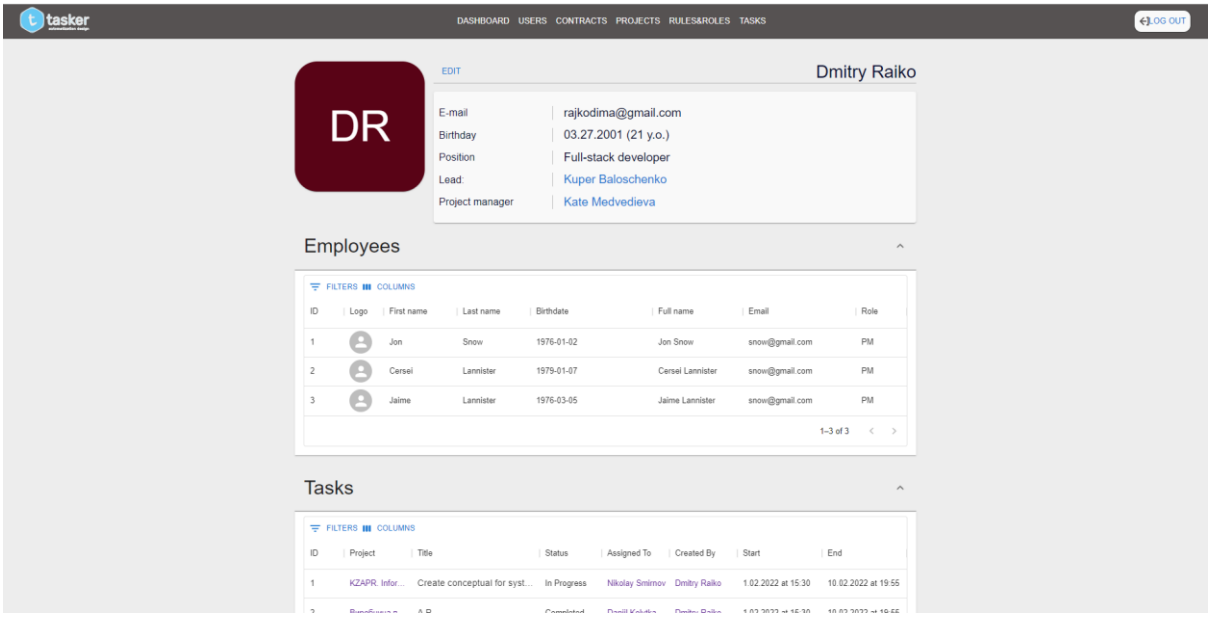



Рисунок 3.7 – Відображення сторінки перегляду інформації про конкретного користувача

За типовим шаблоном було сформовано інші web-сторінки. Для прикладу на рисунках 3.8-3.11 представлено відображення для деяких сторінок та функціональних компонентів.

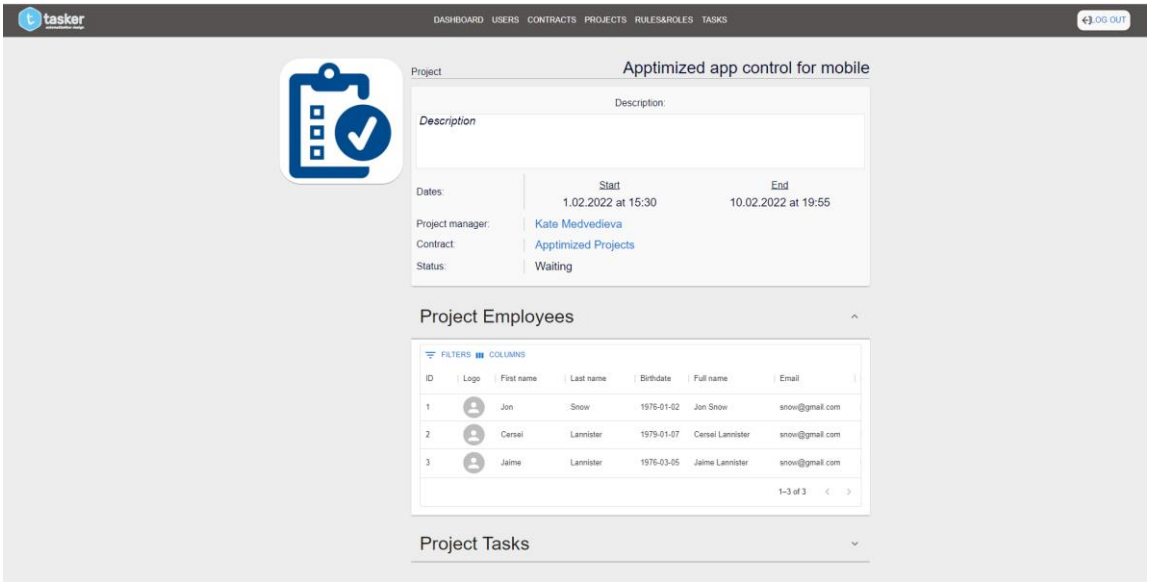



Рисунок 3.8 – Шаблон сторінки проекту

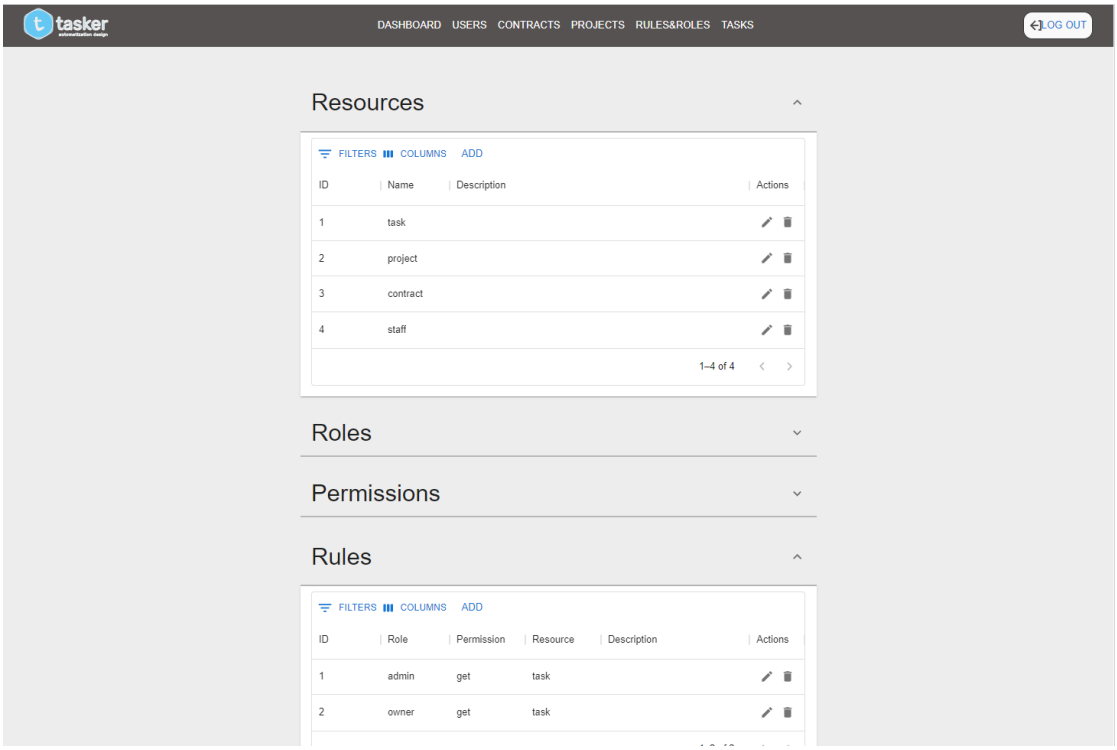



Рисунок 3.9 – Шаблон сторінки перегляду ролей та правил

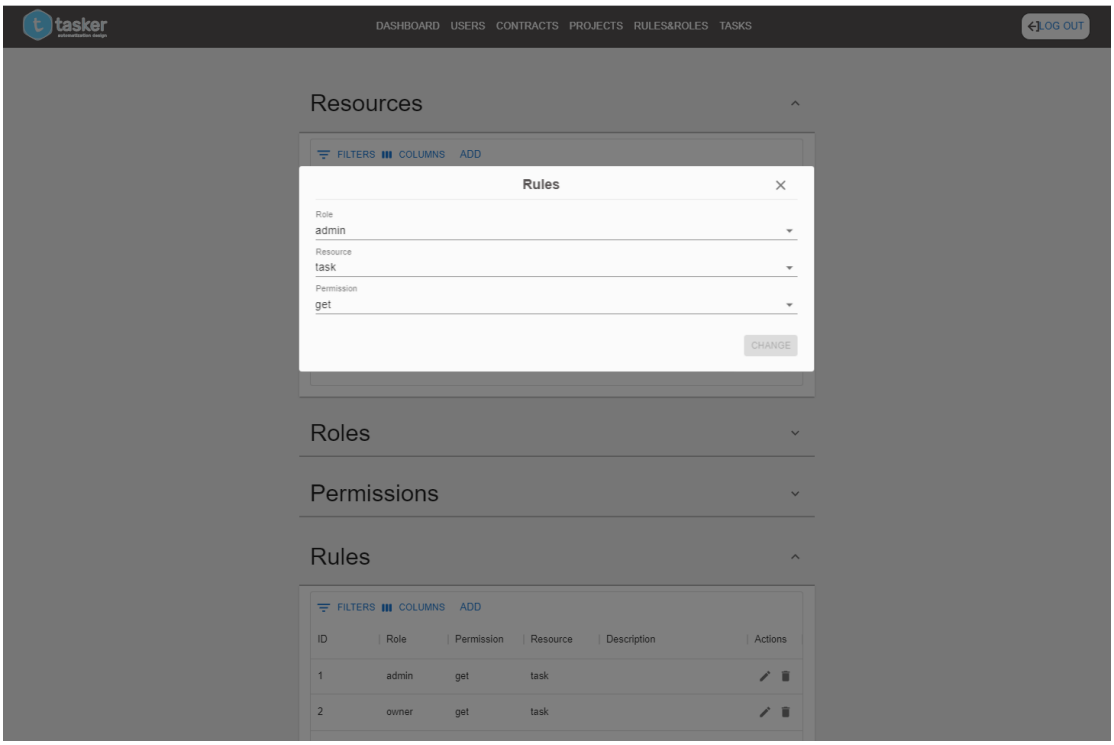



Рисунок 3.10 – Шаблон модального вікна для зміни правила

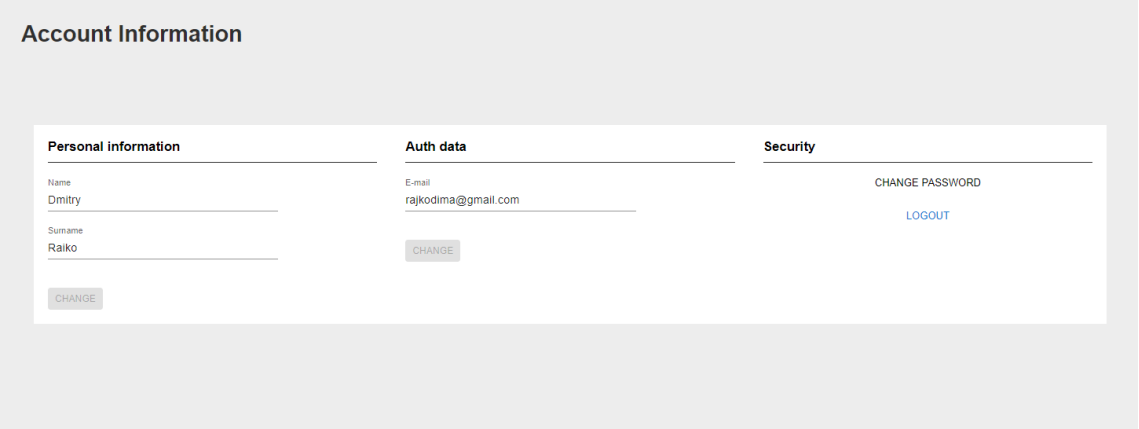



Рисунок 3.11 – Сторінка налаштувань для користувача

Також для надання більшої унікальності та індивідуальності було розроблено логотип даного web-додатку (рис. 3.12). Він виконаний у блакитних кольорах із світлими вставками й зображає шестерню, яка символізує рух та автоматизацію.


Рисунок 3.12 – Логотип web-додатку

Усі шрифти, які використовувалися під час розробки, виконано з застосуванням Google Fonts [17]. Верстку було релізовано за допомогою бібліотеки MUI, яка відповідає за зовнішній вигляд компонентів та побудову взаємного розташування блоків, а також інтерпретатора SASS. Його використано для додаткової стилізації елементів.

## 3.3 Програмна реалізація web-додатку

Програмна реалізація розпочинається з підготовки середовища для подальшої розробки програмного продукту. Для цього необхідно створити папку на персональному комп’ютері для майбутнього проекту. Потім треба відкрити консоль. Далі за допомогою команди «npx create-react-app «Tasker»» ініціювати створення проекту. Наступним кроком клонуємо проект до репозиторію на GitHub [18], використовуючи команду «git init». Установлюємо всі залежності та необхідні бібліотеки. Для написання програмного коду обрано IDE WebStorm [19].

Так як шапка web-додатку буде доступною на всіх його сторінках, її було винесено на найвищий рівень відображення. За допомогою умовних операторів налаштовано виведення різних її видів для окремих користувачів (рис. 3.13).

```
{(role === 'admin' && (
<AdminHeader />
)) || (
<>
  <MobileHeader handleShowNotificationModal={handleNotificationModal} />
  <Header handleShowNotificationModal={handleNotificationModal} />
</>
)}
```

Рисунок 3.13 – Лістинг для відображення шапки web-додатку

Далі починаємо програмну реалізацію сторінок відповідно зазначеним шаблонам. Проводимо декомпозицію більших блоків та створюємо на їх основі менші компоненти, які можливо повторно використовувати. Приклад створення компоненту сторінки відображення всіх користувачів показано на рисунку 3.14.

```
const UsersPage = () => {
  const navigate = useNavigate();
  const handleRowClick = (data) => {
    navigate(`/profile/${data.id}`);
  };
  return (
    <div className="main-admin-wrapper">
      <DataTable
        rows={users}
        columns={usersTableColumns}
        isDelete
        handleRowClick={handleRowClick}
      />
    </div>
  );
};

export default UsersPage;
```

Рисунок 3.14 – Компонент для відображення всіх користувачів

Для коректного рендерингу відповідних компонентів необхідно створити механізм роутингу, тобто відображення необхідної сторінки при переході за певною адресою (рис. 3.15).

```
<>
  <Route path="/">
    <Route index path="" />
    <Route path="rules" element={<RulesPage />} />
    <Route path="support" />
    <Route path="profile">
      <Route path="" element={<UsersPage />} />
      <Route path=":userId" element={<UserPage handleModalTaskOpen={handleModalTaskOpen} />} />
    </Route>
    <Route
      path="tasks"
      element={<TasksPage handleModalTaskOpen={handleModalTaskOpen} />}
    />
    <Route path="contracts">
      <Route path="" element={<ContractsPage />} />
      <Route path=":contractId" element={<ContractPage />} />
    </Route>
    <Route path="projects">
      <Route path="" element={<ProjectsPage />} />
      <Route path=":projectId" element={<ProjectPage handleModalTaskOpen={handleModalTaskOpen} />} />
    </Route>

  </Route>
```

Рисунок 3.15 – Програмний код для відображення основних маршрутів та компонентів, які для них відображаються для адміністратора

Лістинг основних модулів web-додатку для підтримки діяльності адміністратора КЗАПР описано у Додатку В.

## 3.4 Демонстрація роботи web-додатку

Під час першого сеансу роботи з розробленим web-додатом відвідувач має пройти процедуру реєстрації для нового користувача. В іншому випадку здійснити авторизацію для вже існуючого. Він може це зробити звичайним шляхом, тобто через електронну пошту та пароль. Але також доступним є спосіб OAuth авторизації за допомогою Google-сервісів. Для його налаштування необхідно зареєструвати новий додаток на Google Cloud Platform [20], заповнити всі необхідні дані й отримати спеціальний ключ для користування функціями реєстрації. Форми реєстрації та авторизації за допомогою Google зображено на рисунках 3.16-3.17 відповідно.

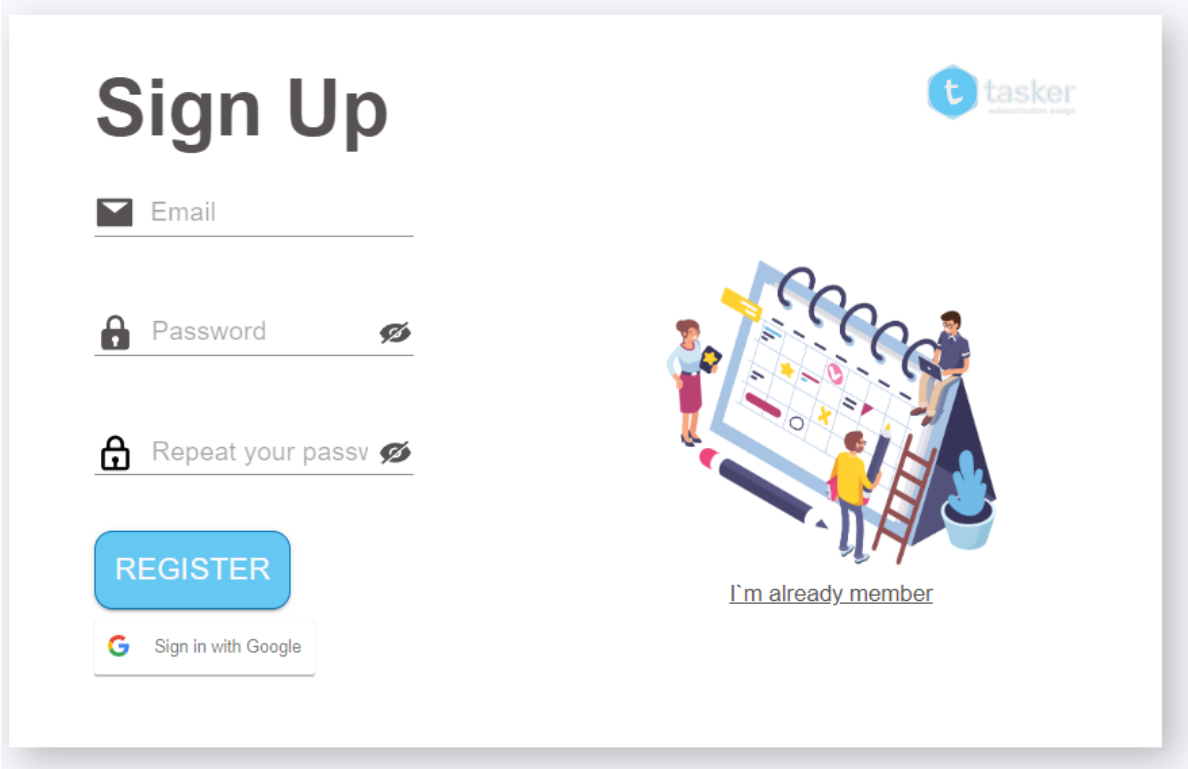



Рисунок 3.16 – Форма реєстрації

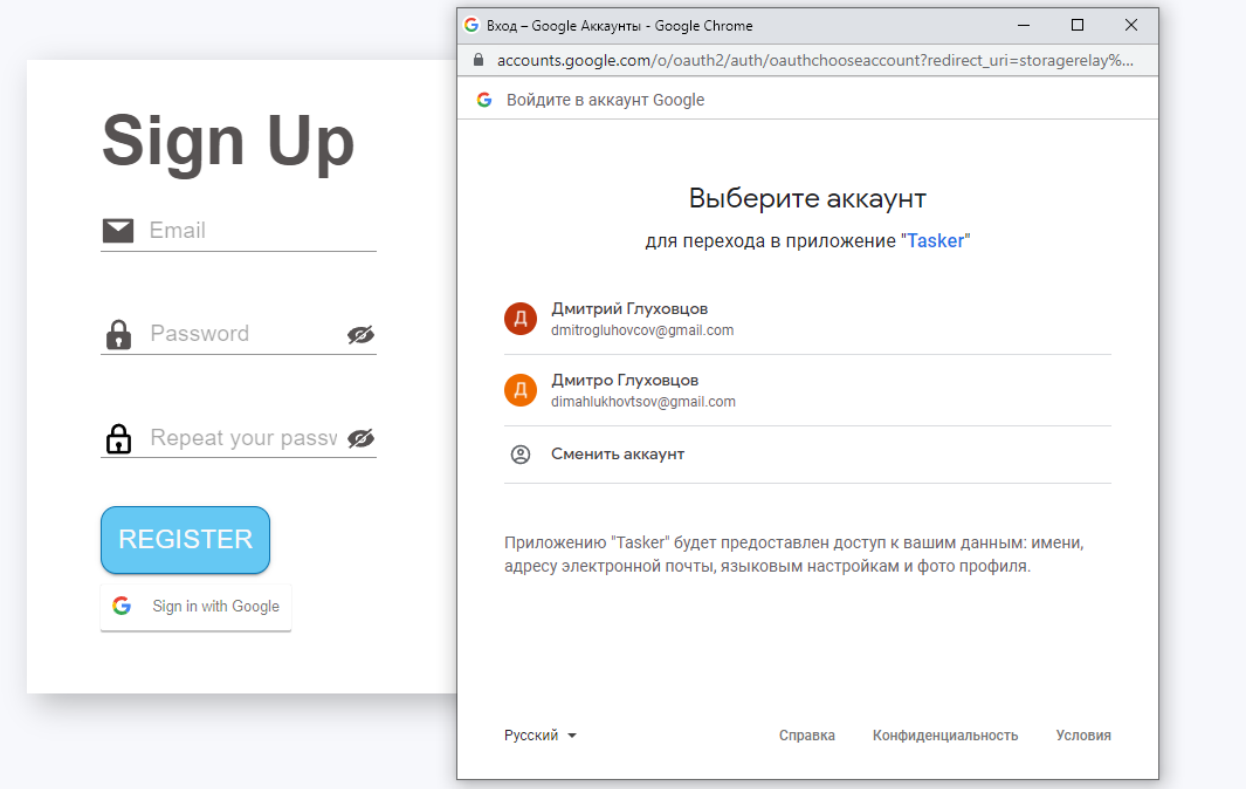



Рисунок 3.17 – Вигляд Google авторизації

Після успішного проходження авторизації чи реєстрації відбувається запис необхідної інформації до сховища сесій [20], звідки потім беруться такі дані як роль, інформація про користувача та інше. Далі відвідувач потрапляє на сторінку зі списком усіх користувачів, де є можливість у табличній формі переглянути всі профілі системи (рис. 3.18).

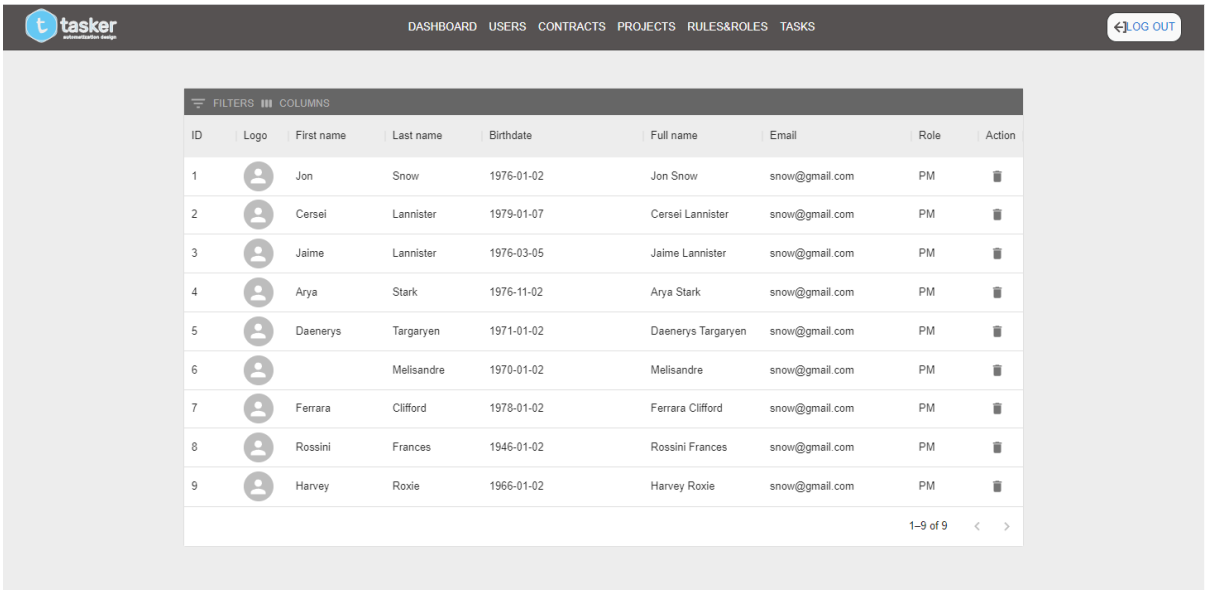



Рисунок 3.18 – Вигляд сторінки з усіма користувачами

Так як, для кожної сторінки є необхідним виведення великої кількості інформації в табличному вигляді, тому було розроблено головний компонент – таблицю. Вона може підключатися в коді в будь-якому місці і, отримавши необхідні параметри, відображати дані у зручному вигляді. Також для кожної таблиці є доступними фільтрація за полем (рис. 3.19), сортування та можливість вимкнення відображення для певної колонки.

При натисканні на необхідний рядок відбувається переадресація до профіля користувача. Сторінка останнього представлена на рисунку 3.20. На ній розміщена основна інформація про профіль та випадаючі списки з задачами для цього користувача та його підлеглими, якщо це керівна посада. Деякі значення в таблицях є посиланнями. Як-то, наприклад, проект та люди, які призначені до задачі.

Також є можливість редагувати керівника людини та її посаду (рис. 3.21).

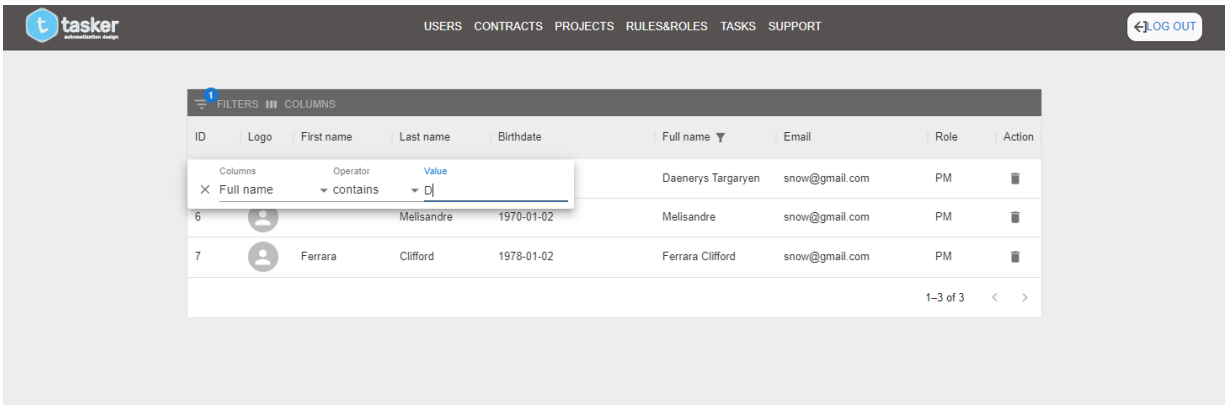



Рисунок 3.19 – Робота фільтра

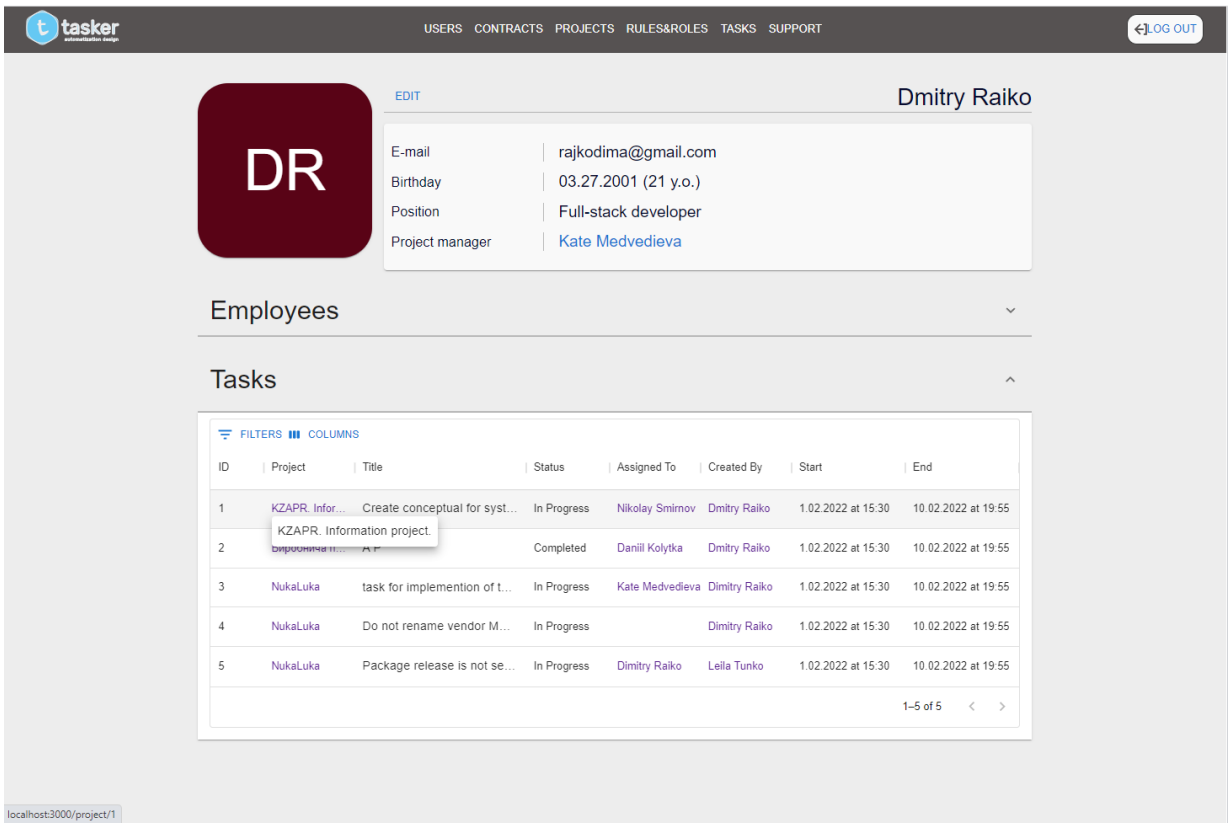



Рисунок 3.20 – Сторінка користувача

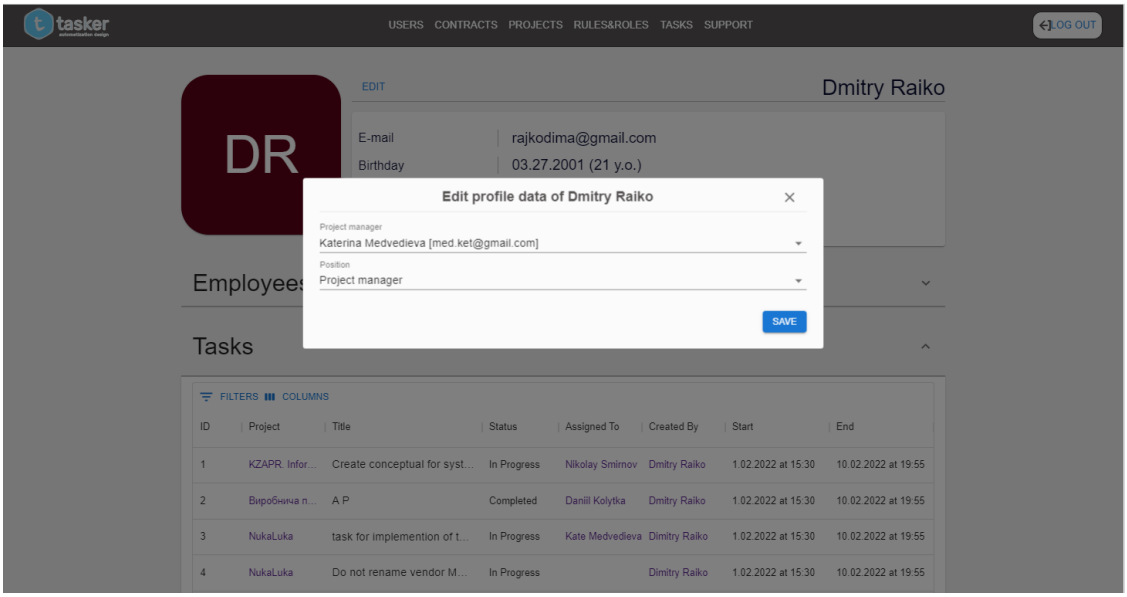



Рисунок 3.21 – Форма для редагування профілю

При переході за посиланням до проекту, користувач потрапляє на сторінку з його основним описом. Адміністратор може лише переглядати наявні проекти та закріплений за ним персонал і його список задач. Також там знаходиться список тих, хто працює на ньому й прив’язані задачі. Сторінка проекту зображена на рисунку 3.22.

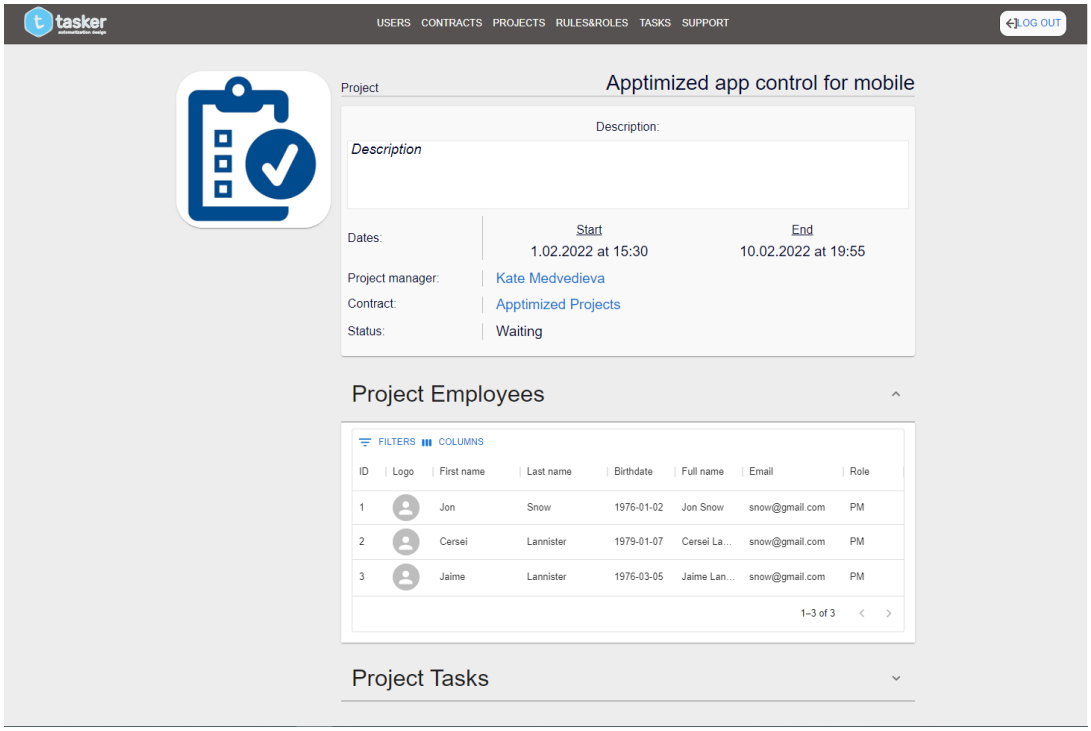



Рисунок 3.22 – Сторінка проекту

За допомогою посилання в полі контракту можна перейти до сторінки з демонстрацією даних про контракт та проектами, що за ним закріплені для кращого розуміння структури контракту та проектів, що створені для нього. Адміністратор може переглядати та переходити за посиланнями до профілів Сторінка контракту зображена на рисунку 3.23.

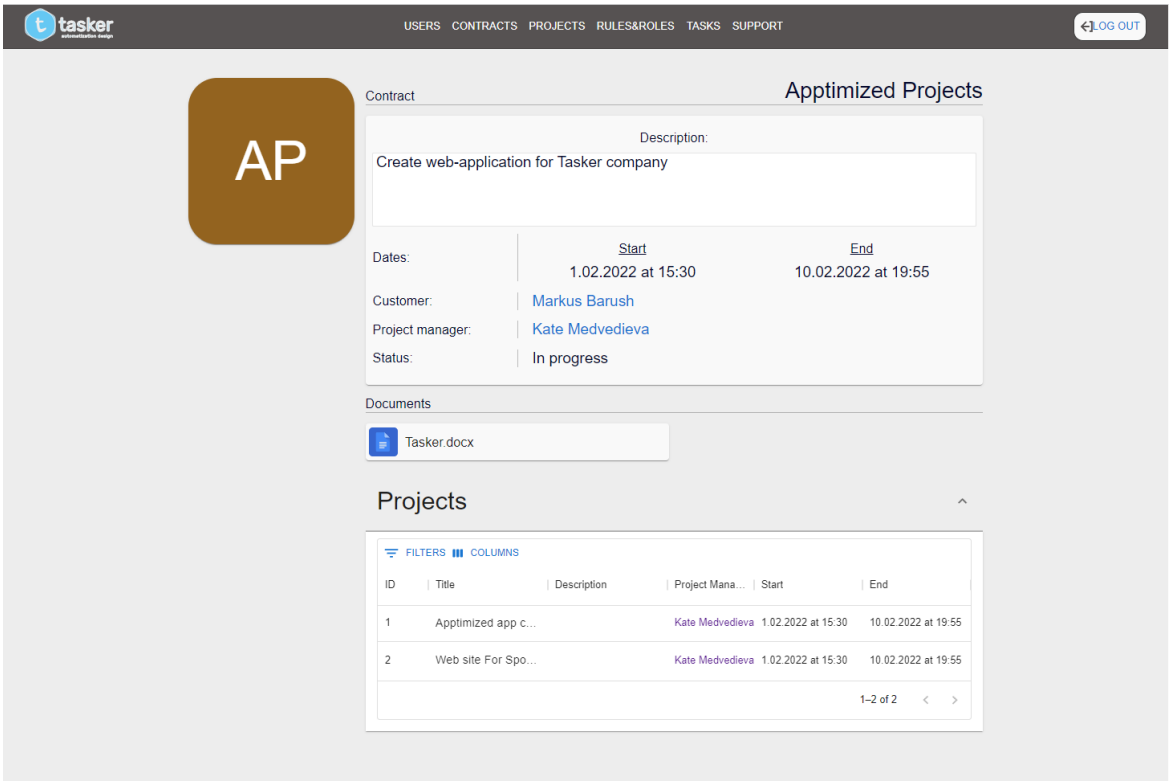



Рисунок 3.23 – Сторінка контракту

Перейшовши за посиланням «Tasks» у шапці web-додатку можна перейти до списку всіх задач наявних в системі, для можливостей перегляду та ознайомлення з ними, а також за необхідністю видаленням, які також демонструються у вигляді таблиці (рис. 3.24).

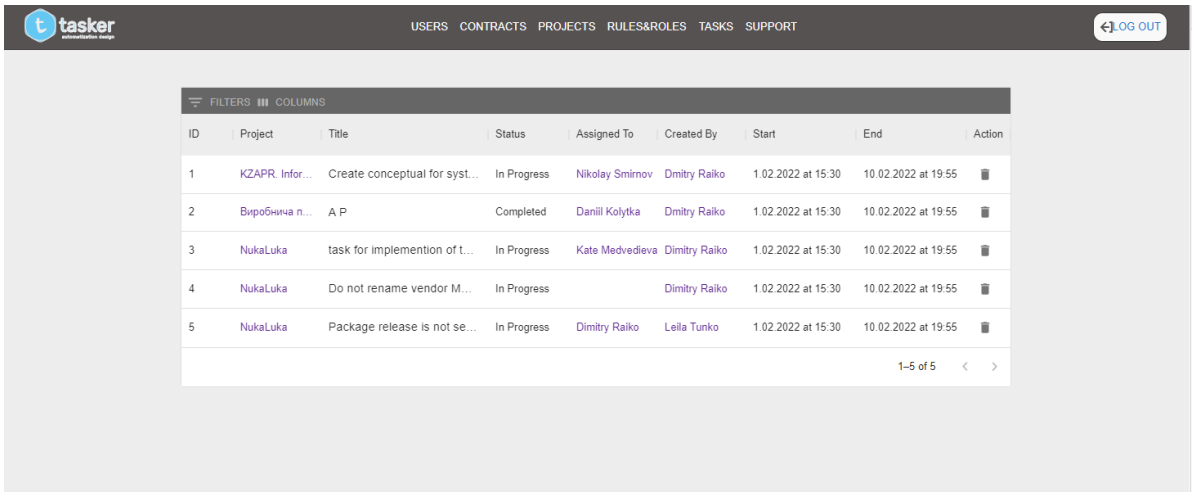



Рисунок 3.24 – Таблиця з усіма завданнями

Після натискання на окрему задачу відкривається вікно для перегляду подробиць щодо неї (рис. 3.25). Адміністратор може лише переглядати таку інформацію, але не змінювати.

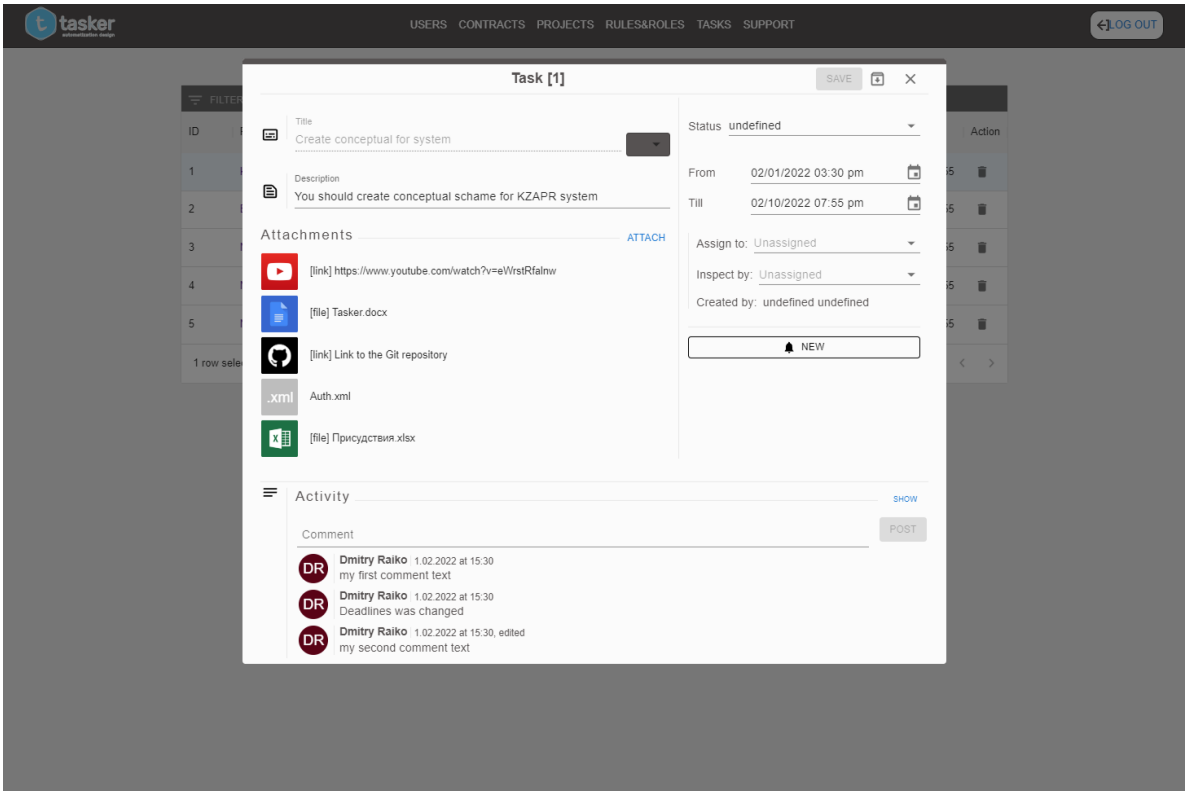



Рисунок 3.25 – Перегляд задачі

Перейшовши на сторінку «Rules&Roles» можна побачити всі наявні сервісні таблиці, які використовуються для розмежування доступу та надання прав на відповідні дії на серверній частині та на клієнтській. Наприклад, за допомогою ролей можна керувати відображенням різного функціоналу та блоків даних, як-то шапка web-додатку чи панель адміністратора. Вигляд даної сторінки зображено на рисунку 3.26.

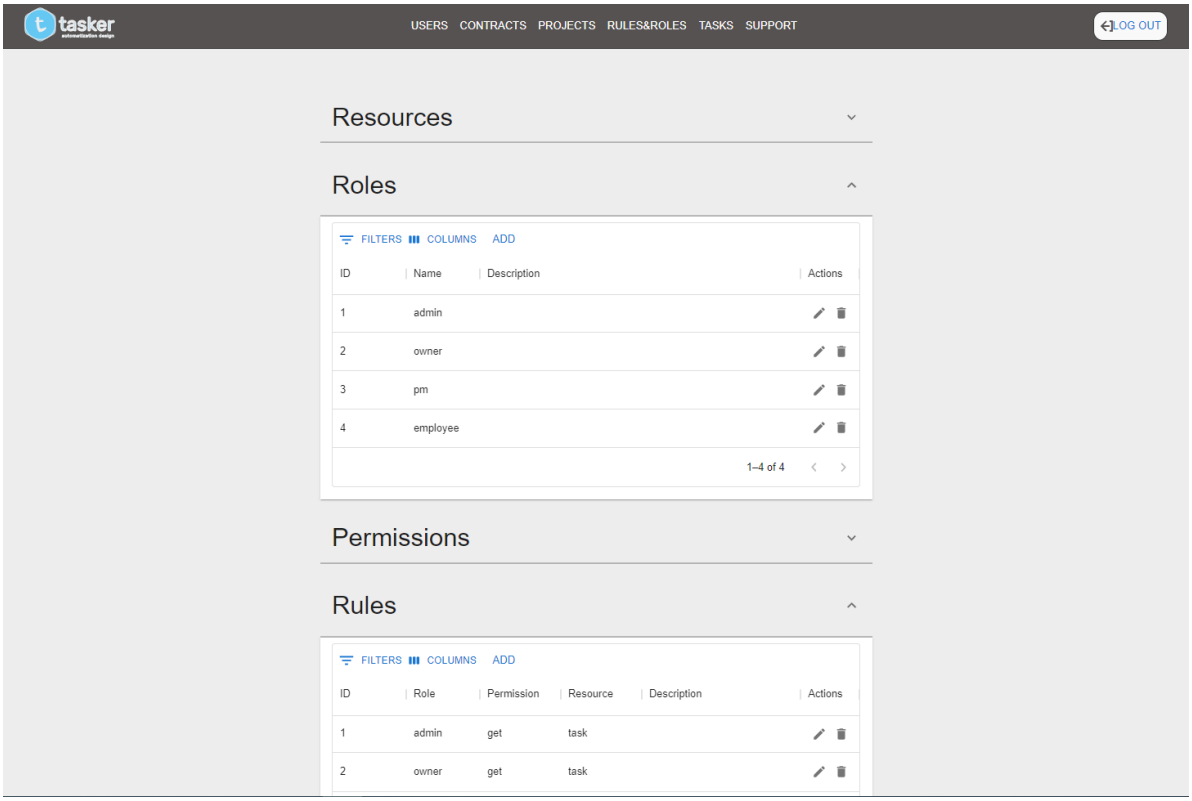



Рисунок 3.26 – Сторінка з ролями та правилами

Для кожної таблиці є можливість редагувати та додавати дані за допомогою відповідних модальних вікон та форм. На рисунках 3.27-3.28 продемонстровано як це відбувається.

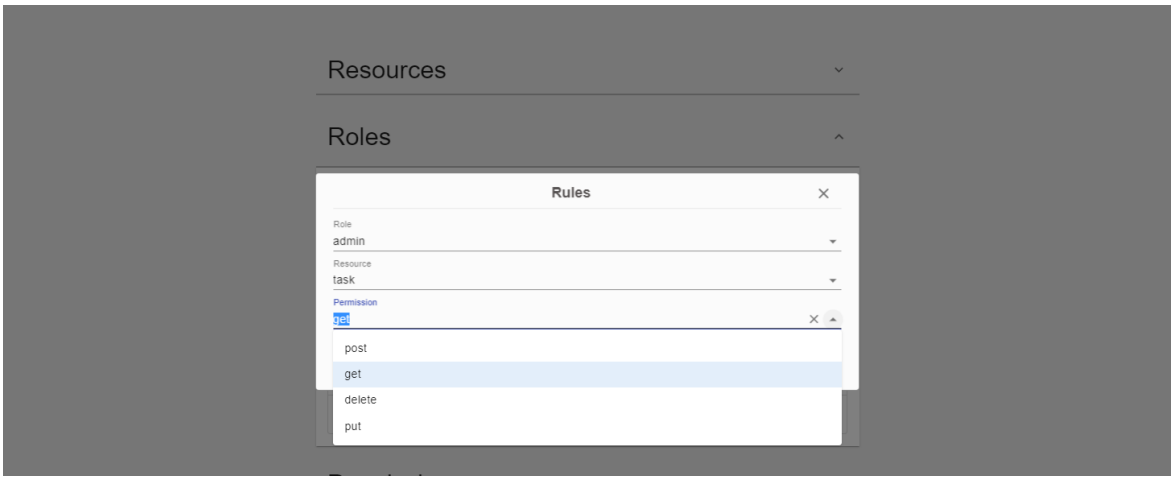



Рисунок 3.27 – Форма для оновлення даних про правила

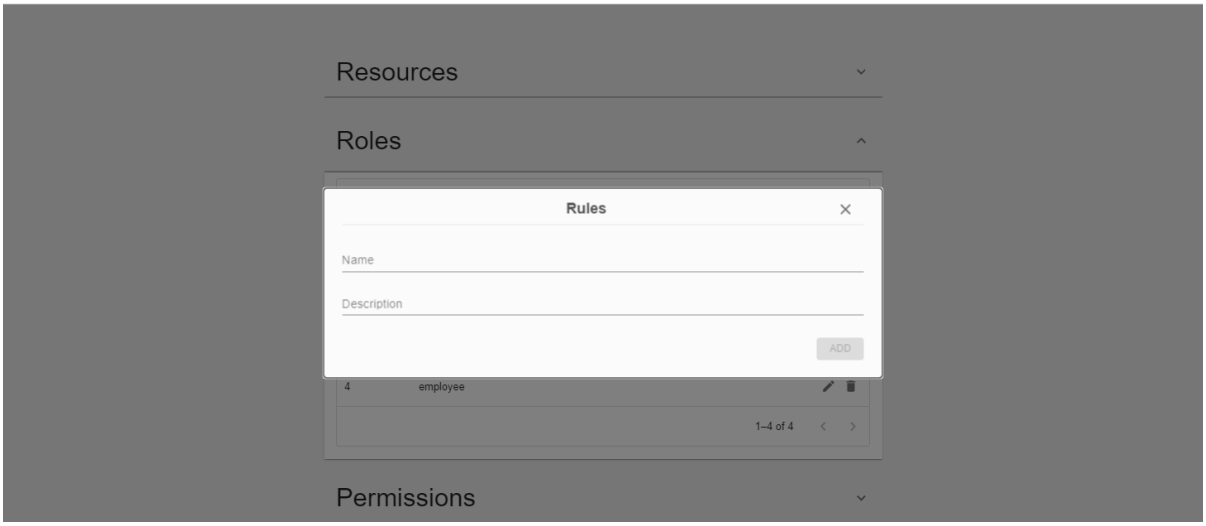



Рисунок 3.28 – Форма для додавання нової ролі

На вкладці «Support» знаходяться повідомлення, які надсилають користувачі (рис. 3.29)

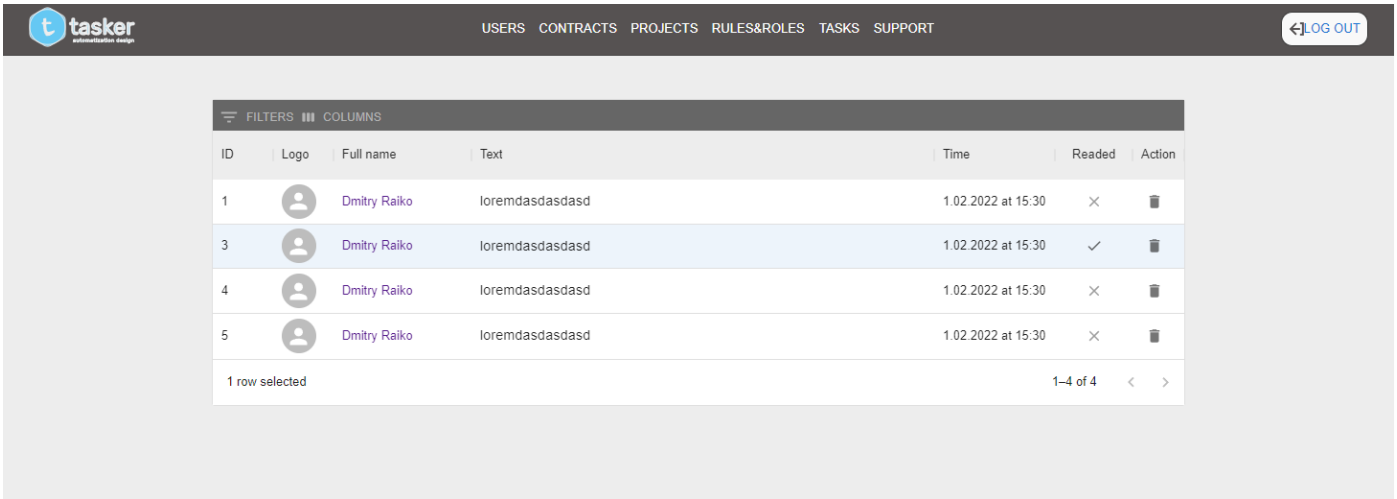



Рисунок 3.29 – Повідомлення на вкладці Support

Після натискання на будь-яке повідомлення відкривається модальне вікно для додавання свого тексту та надсилання його на електронну пошту відправнику (рис. 3.30).

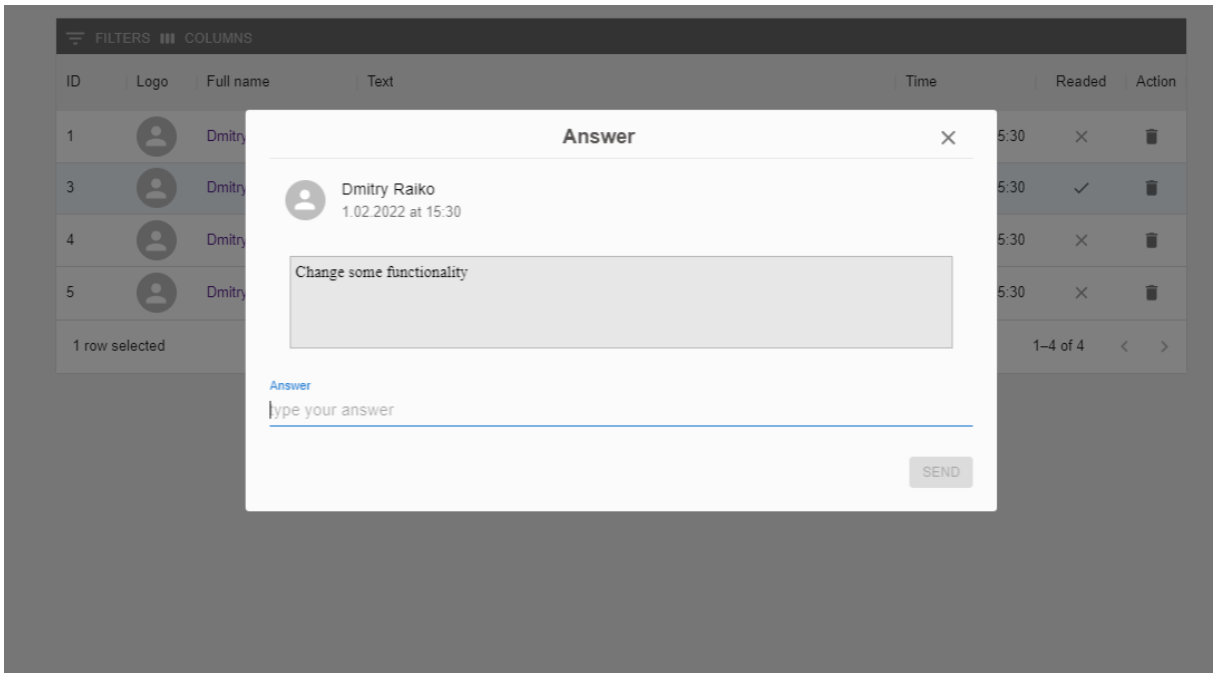



Рисунок 3.30 – Форма для написання та відправки відповіді

Також було розроблено сторінку налаштувань для користувача, де він може змінити свої контактні дані та пароль за допомогою відповідних форм. Сторінка налаштувань зображена на рисунку 3.31.

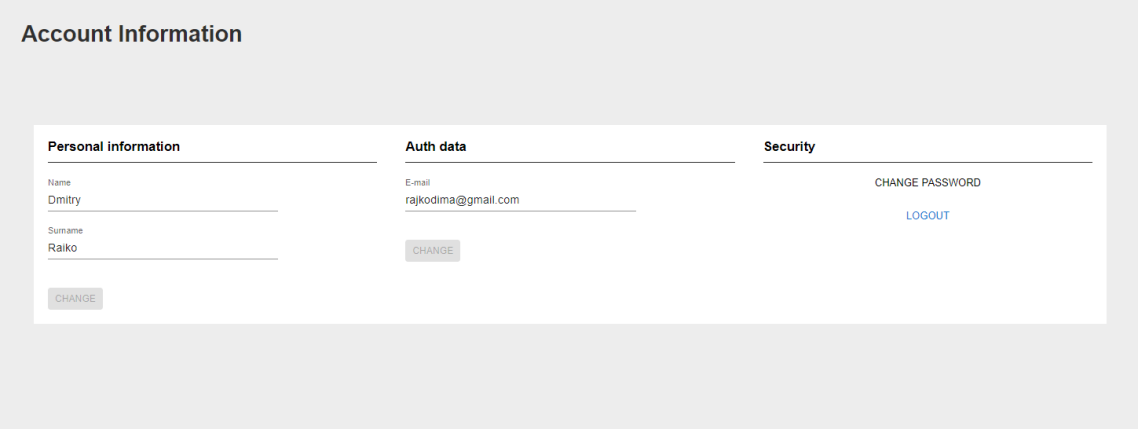



Рисунок 3.31 – Сторінка Налаштування

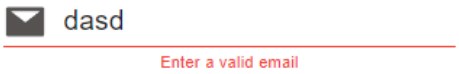
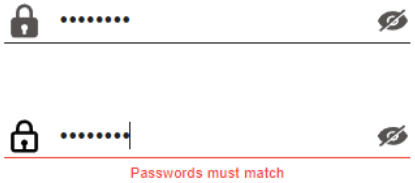
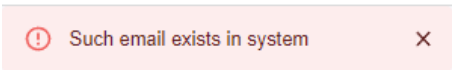
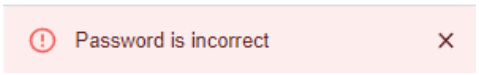
## 3.5 Тестування web-додатку

Для перевірки працездатності функціоналу web-додатку необхідно провести функціональне тестування. Тобто виконати введення вхідних даних у форми та дослідити їхні реакцій на внесення різного роду даних. До них входять реєстраційна та авторизаційна сторінки, редагування користувача. Також важливим є перевірка створення та редагування записів у сервісних таблицях.

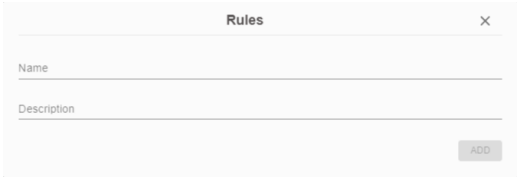
У першу чергу було протестовано можливість реєстрації та авторизації та коректного відпрацювання ролей після цих дій і перебіг перенаправлення на ті сторінки, які цими умовами передбачені. Також перевірено коректність відображення даних скрізь по розробленому web-додатку та чи є правильними переходи за різними посиланнями.

Під час тестування було використано метод «чорного ящика». Тобто перевірявся функціонал без знання внутрішньої структури та коду [21]. Основні результати проведення тестування відображені у таблиці 3.1.

Таблиця 3.1 – Результати тестування web-додатку

| № | Назва | Очікуваний результат | Фактичний результат | 0/1 |
|---|---|---|---|---|
| 1 | Перевірка форми реєстрації не некоректно введений email | Поле підсвічено червоним та виведено відповідне повідомлення про неправильно введені дані |  | 1 |
| 2 | Перевірка форми реєстрації на невідповідність паролів | Поля підсвічено червоним і виведено відповідне повідомлення про неправильне введення даних |  | 1 |
| 3 | Перевірка форми реєстрації на введення email існуючого в системі | Виведення відповідного повідомлення про неправильно введені дані |  | 1 |
| 4 | Перевірка форми авторизації на невірний пароль | Виведення повідомлення про неправильно введені дані |  | 1 |
| 5 | Перевірка форми на введення валідних даних | Переадресація до головної сторінки | Відбувається переадресація на сторінку усіма зареєстрованими користувачами | 1 |

Продовження таблиці 3.1

| | | | | |
|---|---|---|---|---|
| 6 | Перевірка форми редагування правила використовуючи значення не зі списку | Не активна кнопка «Change» |  | 1 |
| 7 | Перевірка роботи фільтру на профілях користувачів шляхом фільтрації за полем повного імені | Фільтрація відбувається успішно | Дані в таблиці відфільтровано та отримано вибірку з необхідними нам даними | 1 |
| 8 | Перевірка реакції форми для додавання ресурсу на пусті поля | Не активна кнопка «Add |  | 1 |

У результаті проведеного тестуванні не було виявлено критичних багів. Усі перевірки пройшли успішно. Функціонал працює відповідно вимогам.

# ВИСНОВОК

У результаті кваліфікаційної роботи бакалавра розроблено web-додаток підтримки діяльності адміністратора КЗАПР.

Було проведено дослідження та проаналізовано предметну область використання web-додатку, розглянуто аналоги подібного продукту та останні публікації з подібною проблематикою, поставлено мету та задачі на проектування та створення програмного продукту «Web-додаток для підтримки діяльності адміністратора КЗАПР». Розроблено технічне завдання на розробку web-додатку (Додаток А). Було описано планування робіт і разом з цим досліджено ризики, які можуть виникнути при розробці web-додатку (Додаток Б).

Під час виконання проектної частини дипломної роботи було проведено функціонально-структурне моделювання й розроблено діаграму використання. У результаті проектування бази даних було виділено основні сутності та їхні атрибути, розроблено її структуру та застосовано для створення БД.

У практичній частині реалізації визначено архітектуру web-додатку, описано дизайн сторінок та імплементовано їх програмним шляхом, розроблено основний функціонал та продемонстровано його роботу, проведено успішне тестування.

Розроблений web-додаток для підтримки діяльності адміністратора КЗАПР дозволить автоматизувати та пришвидшити дії, пов’язані з адмініструванням систем. Розроблені модулі для реєстрації та авторизації надають вагомий функціонал для розмежування та контролю доступу користувачів.

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. Hlukhovtsov D. Web application to support the activities of the administrator in KZAPR. *Інформатика, математика, автоматика* : Матеріали та програма міжнародної наукової конференції молодих учених, Суми, 18 April – 22 May 2022. Суми, 2022. P. 112. (Дата звернення 05.05.2022)

2. Севостьянов І. В. Автоматизація проектування технологічних процесів механічної обробки та складання. *Вісник машинобудування та транспорту*. 2018. Т. 1, № 7. С. 118. URL: https://vmt.vntu.edu.ua/index.php/vmt/article/view/127/115 (дата звернення: 06.05.2022).

3. Нечепуренко Д. С. Модернізація систем автоматизації управління підприємствами машинобудування : Кваліфікаційна наукова праця на правах рукопису. Запоріжжя, 2019. 274 с. URL: http://phd.znu.edu.ua/page/dis/08_2019/dis_Nechepurenko.pdf (дата звернення: 07.05.2022).

4. Що таке ERP-система та як вона допоможе вашому бізнесу?. *Дія.Бізнес - Головна сторінка*. URL: https://business.diia.gov.ua/cases/sistematizacia-biznes-procesiv/so-take-erp-sistema-ta-ak-vona-dopomoze-vasomu-biznesu (дата звернення: 18.05.2022).

5. Тема 1. Поняття про системи автоматизованого проектування. *Навчально-інформаційний портал НУБіП України*. URL: https://elearn.nubip.edu.ua/mod/book/tool/print/index.php?id=332249 (дата звернення: 10.05.2022).

6. Переваги та недоліки web-додатків. *Stud*. URL: https://stud.com.ua/97611/informatika/perevagi_nedoliki_dodatkiv (дата звернення: 10.05.2022).

7. Освіта.ua. Адміністратор web-сайту. *Освіта.UA*. URL: https://osvita.ua/proforientation/profession/71532/ (дата звернення: 12.05.2022).

8. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. *React*. URL: https://uk.reactjs.org/ (дата звернення: 13.05.2022).

9. SASS (syntactically awesome stylesheets) в drupal. URL: https://internetdevels.ua/blog/sass (дата звернення: 13.05.2022).

10. What is PostgreSQL. *PostgreSQL Tutorial - Learn PostgreSQL from Scratch*. URL: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/ (date of access: 13.05.2022).

11. Trello. *Trello*. URL: https://trello.com/uk (дата звернення: 16.05.2022).

12. Управляйте работой, проектами и задачами своего коллектива в сети Asana. *Asana*. URL: https://asana.com/ru (дата звернення: 16.05.2022).

13. Jira. *Atlassian*. URL: https://www.atlassian.com/ru/software/jira (дата звернення: 16.05.2022).

14. Учасники проектів Вікімедіа. IDEF0 – вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/IDEF0 (дата звернення: 09.06.2022).

15. Уніфікована мова моделювання UML. *Портал знань – Знання повинні бути доступними*. URL: http://www.znannya.org/?view=uml (дата звернення: 09.06.2022).

16. UML use case diagram tutorial. *Lucidchart*. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (date of access: 09.06.2022).

17. Google fonts. *Google Fonts*. URL: https://fonts.google.com/ (date of access: 09.06.2022).

18. GitHub: where the world builds software. *GitHub*. URL: https://github.com/ (date of access: 09.06.2022).

19. WebStorm. *JetBrains*. URL: https://www.jetbrains.com/ru-ru/webstorm/ (дата звернення: 09.06.2022).

20. Authentication overview | google cloud. *Google Cloud*. URL: https://cloud.google.com/docs/authentication (date of access: 09.06.2022).

21. Люта М. В., Розломій І. О., Новикова К. В. Аналіз методів тестування програмного забезпечення : Thesis. 2017. URL: https://er.knutd.edu.ua/handle/123456789/8421 (дата звернення: 09.06.2022).

# ДОДАТОК А

## ТЕХНІЧНЕ ЗАВДАННЯ

**на розробку**

**«Web-додаток підтримки діяльності адміністратора КЗАПР»**

**ПОГОДЖЕНО:**

Доцент кафедри комп’ютерних наук

_____ Антипенко В.П.

Студент групи ІТ-82-0

_____ Глуховцов Д.О.

**Суми 2022**

# ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ WEB-ДОДАТКУ

## 1.1 Призначення web-додатку

Web-додаток має надавати вичерпний функціонал для адміністрування та підвищення автоматизації різноманітних робочих процесів, що відбуваються на виробництві чи в компанії, наприклад керування правами та доступом, управління персоналом.

## 1.2 Мета створення web-додатку

Метою даного дослідження є розробка web-додатку для підтримки діяльності адміністратора у КЗАПР. Його використання дозволить автоматизувати робочу функціональність останнього.

## 1.3 Цільова аудиторія

До цільової аудиторії відносяться організації машинобудівного профілю, що хочуть перенести свій бізнес в онлайн та оптимізувати його.

# 2 Вимоги до web-додатку

## 2.1 Вимоги до web-додатку в цілому

### 2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток для підтримки діяльності адміністратора КЗАПР необхідно реалізувати за допомогою різноманітних спеціалізованих інструментів та забезпечити вичерпний набір функціональних можливостей, таких як контроль над правами та привілеями користувачів, нагляд за списком співробітників, функціонал для створення та модифікування специфічних даних та інформації.

Кінцевий продукт має бути представлена web-додатком, а точніше одним із модулів, який містить якісне функціональне та інформаційне наповнення та необхідні для роботи матеріали.

### 2.1.2 Вимоги до персоналу

Повинен мати навички у адмініструванні різноманітних систем, розбиратися у тонкощах налаштування та побудови робочих взаємозв'язків, на достатньому рівні розуміти предметну область.

### 2.1.3 Вимоги до збереження інформації

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

### 2.1.4 Вимоги до розмежування доступу

Розроблюваний продукт має бути доступним лише для спеціальної групи користувачів – Адміністратора, який має необмежені права на маніпуляції з даними. Доступ до функціоналу має здійснюватися шляхом переходу на певну сторінку та авторизації за допомогою спеціальних логіна та пароля.

Реєстраційна та авторизаційна сторінки для звичайних користувачів мають бути загальнодоступними у мережі Інтернет. До переліку можливостей та прав доступу відвідувачів цих сторінок відноситься лише реєстрація та авторизація.

## 2.2 Структура web-додатку

### 2.2.1 Загальна інформація про структуру web-додатку

До структури розроблюваних компонентів відносяться сама панель для адміністратора та web-сторінки з формами реєстрації та авторизації.

Перелік сторінок наступний:

- «Реєстрація» – містить форму для реєстрації з різними варіантами проведення цієї операції;
- на сторінці «Авторизація» знаходиться форма для входження до системи різними способами;
- сторінка «Ролі та привілеї» призначена для редагування та створення різноманітних ролей та призначення привілегій певним групам користувачів та визначення можливостей доступу до ресурсів;
- для управління персоналом є сторінка «Персонал», що містить перелік користувачів в системі та їхню взаємодію між собою;
- сторінка «Дані» містить інформацію про проекти, контракти та пов’язані з ними документи.

### 2.2.2 Навігаційне меню

Для зручної навігації необхідно створити меню, що забезпечить швидке та зрозуміле переміщення користувача всіма доступними web-сторінками додатку. Меню має бути закріплене та розташовуватися зверху (у шапці) на кожній сторінці.

### 2.2.3 Управління контентом

Розроблюваний web-додаток сам по собі є адмінпанеллю, тому управління контентом відбувається безпосередньо за участю створеного функціоналу. Цим займається уповноважена особа – адміністратор.

### 2.2.4 Дизайн web-додатку

Дизайн web-додатку має бути у сучасному та мінімалістичному виконанні. Кольорова гама має бути у світлих і м’яких кольорах приємних для сприйняття.

Шрифти, їх вид та розмір має бути зручним для читання та сприймання. Різноманітні блоки та елементи мають мати зручне та інтуїтивно зрозуміле розміщення. Шаблон майбутнього програмного продукту зображено на рисунку А.1

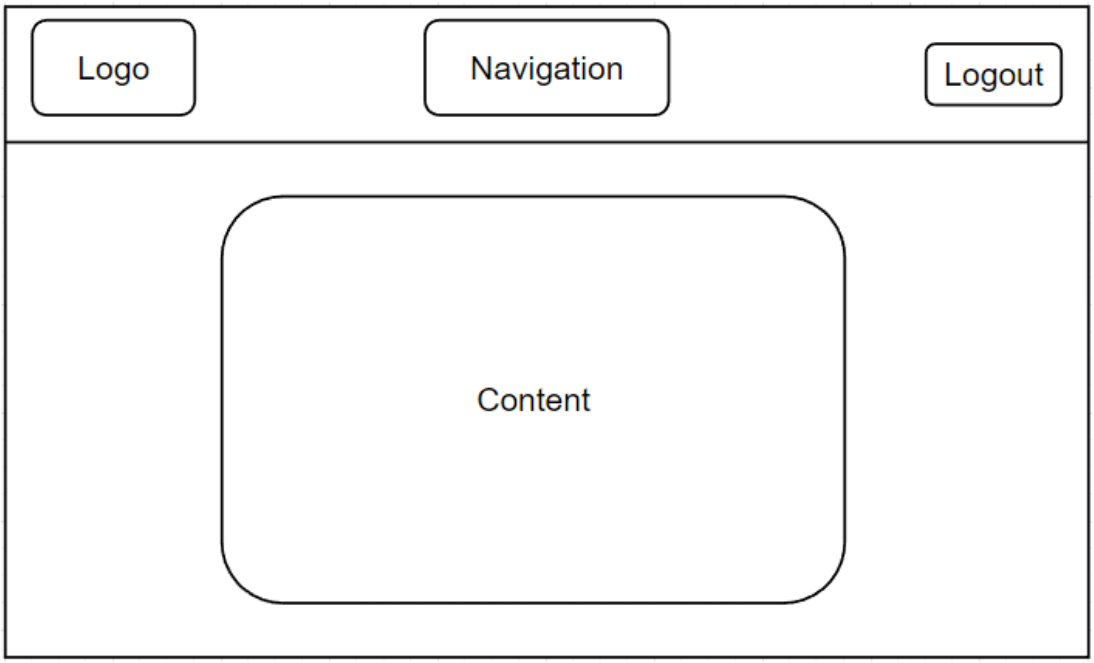



Рисунок А.1 – Схема сторінки

### 2.2.5 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.2.

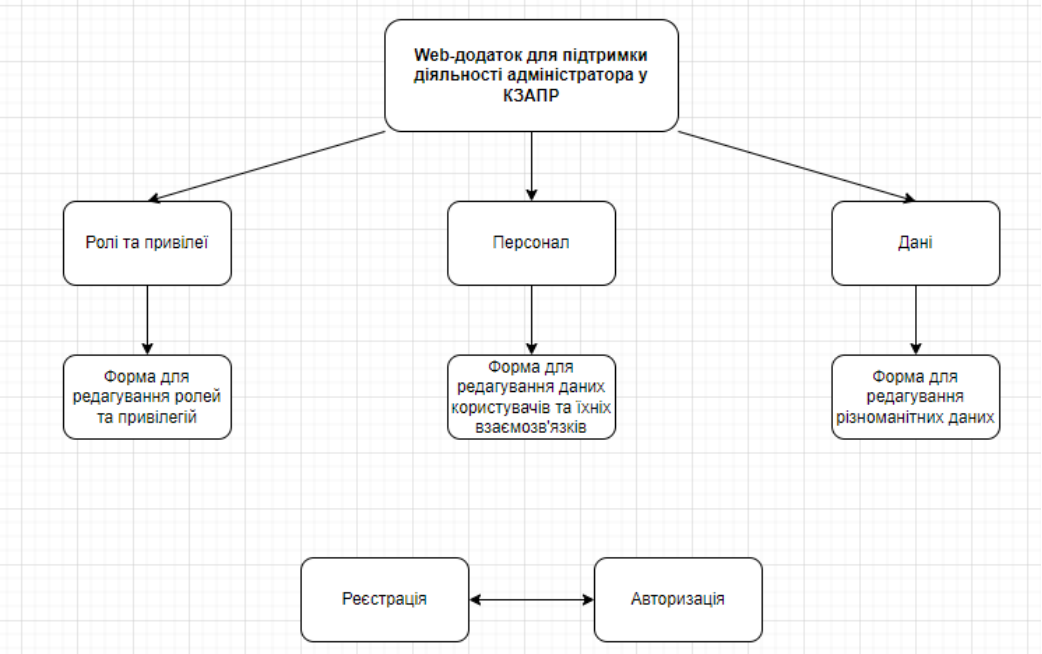



Рисунок А.2 – Система навігації

## 2.3 Вимоги до видів забезпечення

### 2.3.1 Вимоги до лінгвістичного забезпечення

Весь текст у web-додатку має бути виконаний українською або англійською мовами.

## 2.4 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи web-додатку web-браузер має бути Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

## 2.5 Вимоги до функціонування системи

### 2.5.1 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 | Налаштування привілеїв та ролей для користувачів | Адміністратор |
| UN-02 | Внесення змін до певних даних системи | Адміністратор |
| UN-03 | Перегляд всіх даних системи | Адміністратор |
| UN-04 | Перегляд даних про користувачів та їх редагування | Адміністратор |
| UN-05 | Можливість керувати користувачами та налаштовувати їхню робочу ієрархію | Адміністратор, Голова компанії, PM |
| UN-06 | Реєстрація та авторизація | Користувач, Відвідувач |

### 2.5.2 Системні вимоги

Проаналізувавши потреби користувачів, було визначено наступні вимоги:

- наявність реєстрації та авторизації користувачів;
- можливість перегляду даних з різних джерел в базі даних;
- наявність функціоналу для контролю над користувачами та персоналом;
- можливість налаштовувати привілеї та ролі відповідно до потреб;
- зрозумілий інтерфейс для здійснення різноманітних операцій.

## 3 Склад і зміст робіт зі створення web-додатку

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 | Постановка задачі і її затвердження | 4 дні |
| 2 | Проектування та моделювання | 8 днів |
| 3 | Розробка макету | 3 дні |
| 4 | Розробка бази даних | 10 днів |
| 5 | Верстка web-сторінок | 9 днів |
| 6 | Розробка модулю реєстрація/авторизація | 8 днів |
| 7 | Розробка функціоналу для керування персоналом | 8 днів |
| 8 | Розробка функціоналу для управління привілеями та ролями | 10 днів |
| 9 | Розробка функціоналу для управління даними | 10 днів |
| 10 | Модульне тестування | 2 дні |
| 11 | Поєднання компонентів у єдину систему | 2 дні |
| 12 | Інтеграційне тестування | 2 дні |
| 13 | Написання супровідної документації | 1 день |
| 14 | Реліз web-додатку | 1день |
| | Загальна тривалість робіт | 78 днів |

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

Web-додаток має бути затверджено та розміщено на хостингу.

# ДОДАТОК Б

## Планування робіт

Останнім часом вектор організації робочих процесів компаній та виробництв почав змінюватися на можливості віддаленої роботи. Це можна пов’язати і з пандемією і з бажанням працювати з міжнародною командою у співпраці з іноземними партнерами. У свою чергу такі задачі потребують необхідних ресурсів, адже, використання неспеціалізованого забезпечення може значно знизити якість та ефективність виконуваних робіт.

Тому розробка та вдосконалення процесів автоматизації робочих процесів унаслідок проектування та імпелементації web-додатку для підтримки діяльності адміністратора КЗАПР є актуальною та може принести багато дивідендів і значно облегшити життя та підвищити продуктивність, як компаній так і їхніх працівників за рахунок автоматизації таких процесів, як контроль над користувачами, їхніми правами та привілеями.

**Деталізація мети проекту методом SMART.** Для збільшення конкурентноздатності та успішності проекту необхідно коректно визначити його мету, що можна зробити за допомогою SMART-методу. Деталізований результат методу зображено в таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

| | |
|---|---|
| Specific (конкретна) | Створити компоненти web-додатку для підтримки діяльності адміністратора, розробити базу даних та організувати розподіл прав для користувачів для комплексу засобів автоматизованого проектування робіт. |
| Measurable (вимірювана) | Розробити визначені компоненти до кінця 4 курсу, використовуючи мінімальний об’єм ресурсів необхідний для виконання завдання. |

Продовження таблиці Б.1

| | |
|---|---|
| Achievable (досяжна, узгоджена) | Для розробки проекту потрібні знання HTML, CSS, мови програмування JavaScript(React), PostgreSQL та навички написання документації. |
| Relevant (реалістична) | Створений web-додаток допоможе автоматизувати та пришвидшити такі процеси як: контроль над правами та привілеями користувачів, нагляд за списком співробітників, створення та модифікація специфічних даних та інформації . |
| Time-framed (обмежена в часі) | Термін досягнення мети проекту визначено за навчальним планом до кінця червня 2022 року. |

**Планування змісту робіт.** Ключовим результатом проекту, що розбиває командну роботу на відслідковувані та керовані частини – це WBS (Work Breakdown Structure – Ієрархічна структура робіт). Іншими словами, це візуальний план компонентів проекту, які згруповано ієрархією. Також WBS надає необхідний вигляд для оцінки термінів та контролю за графіком виконання.

Структурно, на найвищому рівні розміщено кінцевий результат проекту. На другому рівні знаходяться різноманітні дії та заходи для досягнення мети проекту. Таким чином декомпозиція відбувається до моменту, коли дії стають атомарними і мають чіткий та однозначний результат і мають одного виконавця для якого можна розрахувати витрати на працю і часові затрати. На рисунку Б.1 зображено WBS з розробки бази даних та компонентів для особистого кабінету та функціоналу адміністратора.

**Планування структури виконавців.** Наступним кроком після декомпозиції процесів є розробка організаційної структури виконавців або ще OBS, яка зображується як графічна структура, що відображає учасників проекту відповідальних за певний етап реалізації.

До відповідальних осіб відносяться співробітники, що виконують елементарну роботу зазначену в OBS.

На рисунку Б.2 зображено організаційну структуру планування проекту

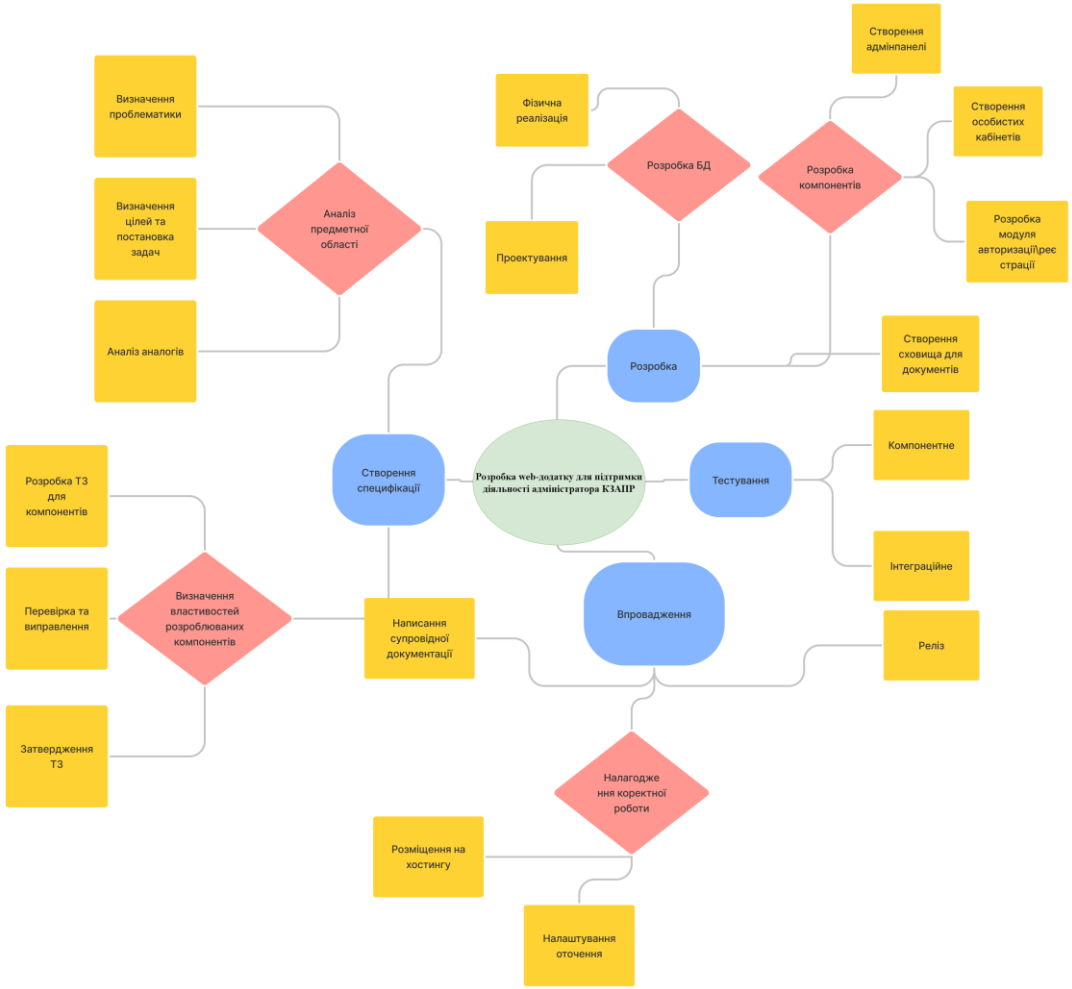



Рисунок Б.1 – WBS-структура робіт проекту

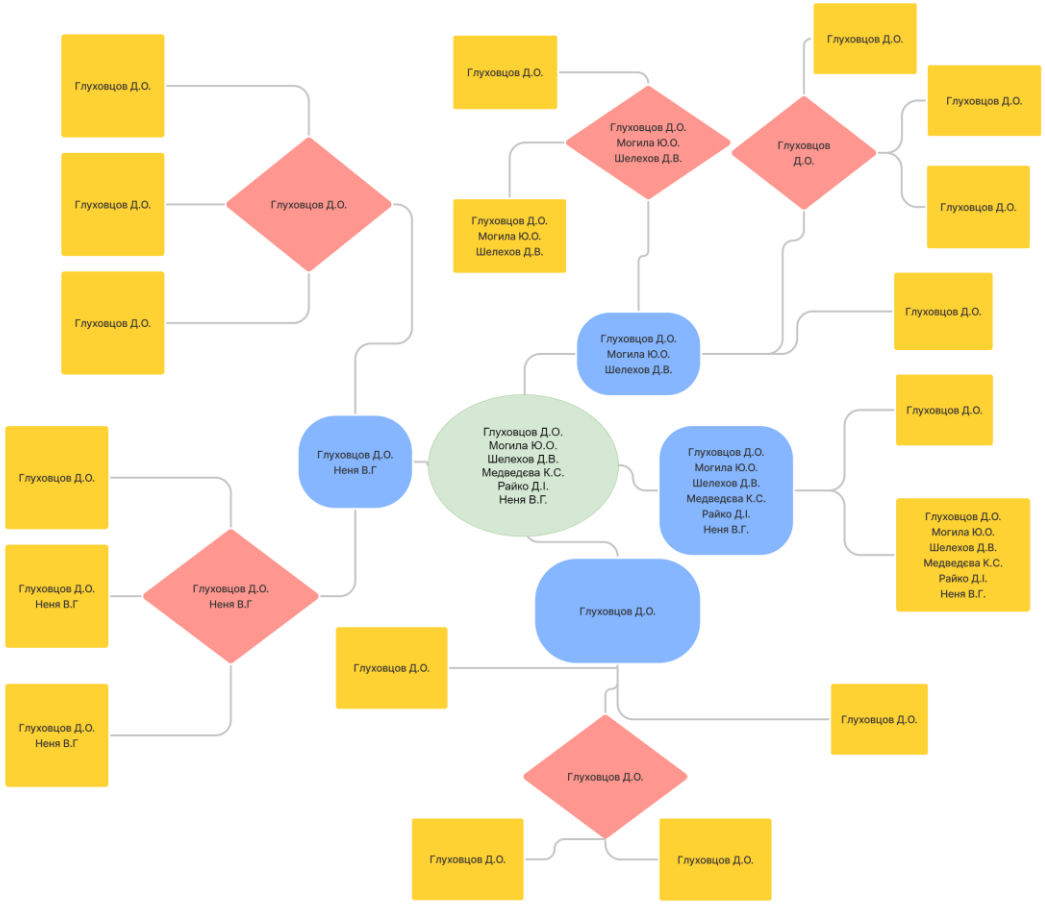



Рисунок Б.2 – OBS-структура робіт проекту

Список виконавців даного проекту описано в таблиці Б.2.

Таблиця Б.2 – Виконавці проекту

| **Роль** | **Ім’я** | **Проектна роль** |
|---|---|---|
| Розробник | Глуховцов Д.О. | Виконує розробку модулів. |
| Проектувальник | Глуховцов Д.О | Відповідає за проектування бази даних та структуру програмного забезпечення. |
| Тестувальник | Глуховцов Д.О.<br>Могила Ю.О.<br>Шелехов Д.В.<br>Медведєва К.С.<br>Райко Д.І.<br>Неня В.Г. | Здійснює тестування дизайну та функціональності створених компонентів. |
| Керівник проекту | Неня В.Г | Видає завдання на розробку проекту. |
| Менеджер проекту | Глуховцов Д.О. | Відповідальний за розподіл ресурсів, завдань та своєчасне виконання |

**Діаграма Ганта.** Побудова календарного цієї діаграми - це один з найважливіших етапів під час планування проекту, - під час цього відбувається розбиття виконання робіт з реальним розподілом дат. Вона дозволяє отримати коректне уявлення про процеси та їхню тривалість маючи обмеження у ресурсах, при врахуванні вихідних днів та свят. Календарний графік проекту представлено на рисунку Б.3.

| | | | | | |
|---|---|---|---|---|---|
| | **Web-додаток для підтримки діяльності адміністратора КЗАПР** | **117 днів** | **Чт 13.01.22** | **Сб 25.06.22** | |
| | **Створення специфікації** | **37 днів** | **Чт 13.01.22** | **Пт 04.03.22** | |
| | **Аналіз предметної області** | **17 днів** | **Чт 13.01.22** | **Пт 04.02.22** | |
| | Визначення проблематики | 5 днів | Чт 13.01.22 | Ср 19.01.22 | |
| | Визначення цілей та постановка задач | 7 днів | Чт 20.01.22 | Пт 28.01.22 | 4 |
| | Аналіз аналогів | 5 днів | Пн 31.01.22 | Пт 04.02.22 | 5 |
| | **Визначення властивостей розроблюваних компонентів** | **20 днів** | **Пн 07.02.22** | **Пт 04.03.22** | **3** |
| | Розробка ТЗ для компонентів | 11 днів | Пн 07.02.22 | Пн 21.02.22 | 3 |
| | Перевірка та виправлення | 4 днів | Вт 22.02.22 | Пт 25.02.22 | 8 |
| | Затвердження ТЗ | 5 днів | Пн 28.02.22 | Пт 04.03.22 | 9 |
| | **Розробка** | **40 днів** | **Пн 07.03.22** | **Пт 29.04.22** | **2** |
| | **Розробка БД** | **14 днів** | **Пн 07.03.22** | **Чт 24.03.22** | |
| | Проектування | 9 днів | Пн 07.03.22 | Чт 17.03.22 | 2 |
| | Фізична реалізація | 5 днів | Пт 18.03.22 | Чт 24.03.22 | 13 |
| | Створення сховища для документів | 4 днів | Пт 25.03.22 | Ср 30.03.22 | 12 |
| | **Розробка компонентів** | **22 днів** | **Чт 31.03.22** | **Пт 29.04.22** | **15** |
| | Створення адмінпанелі | 4 днів | Чт 31.03.22 | Вт 05.04.22 | 15 |
| | Створення особистих кабінетів | 8 днів | Ср 06.04.22 | Пт 15.04.22 | 17 |
| | Розробка модуля авторизації\реєстрації | 10 днів | Пн 18.04.22 | Пт 29.04.22 | 18 |
| | **Тестування** | **13 днів** | **Пн 02.05.22** | **Ср 18.05.22** | **11** |
| | Компонентне | 7 днів | Пн 02.05.22 | Вт 10.05.22 | 11 |
| | Інтеграційне | 6 днів | Ср 11.05.22 | Ср 18.05.22 | 21 |
| | **Впровадження** | **27 днів** | **Чт 19.05.22** | **Сб 25.06.22** | **20** |
| | Написання супровідної документації | 3 днів | Чт 19.05.22 | Пн 23.05.22 | 20 |
| | **Налагодження коректної роботи** | **24 днів** | **Вт 24.05.22** | **Сб 25.06.22** | **24** |
| | Налаштування оточення | 4 днів | Вт 24.05.22 | Пт 27.05.22 | 24 |
| | Розміщення на хостингу | 5 днів | Пн 30.05.22 | Пт 03.06.22 | 26 |
| | Реліз | 1 день | Сб 25.06.22 | Сб 25.06.22 | 27 |

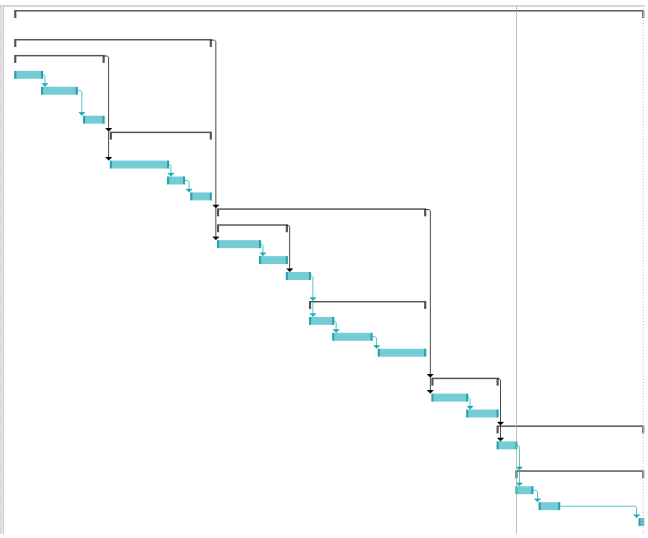


Рисунок Б.3 – Календарний графік розробки web-додатку для підтримки діяльності адміністратора КЗАПР

**Управління ризиками проекту.** Визначення ризиків для проекту також є невід'ємною частиною при його створенні. Під час якісної оцінки ризиків треба визначити ті, що мають бути усунені в першу чергу. Реагування має бути відповідним степені ризику. Далі виконується кількісна оцінка ризиків. Кількісне та якісне оцінювання виконуються одночасно або окремо, що залежить від ступеня забезпечення проекту. У таблиці Б.3 представлено шкалу для класифікації ризиків за величиною впливу на проект та ймовірністю їх виникнення.

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

| Оцінка | Ймовірність виникнення | Вплив ризику | Тип ризику |
|---|---|---|---|
| 1 | Низька | Низький | Прийнятні |
| 2 | Середня | Середній | Виправдані |
| 3 | Висока | Високий | Неприпустимі |

Для зниження негативного впливу ризиків на проект є необхідним виконання планування реакцій на виникнення подібного. Під час цього визначається

ефективність розробки та оцінка наслідків впливу. Оцінюють зазвичай за показниками, описаними в таблиці Б.3. Результатом планування реагування є отримана матриця ймовірності виникнення ризиків та впливу ризику, що зображена на рисунку Б.4. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

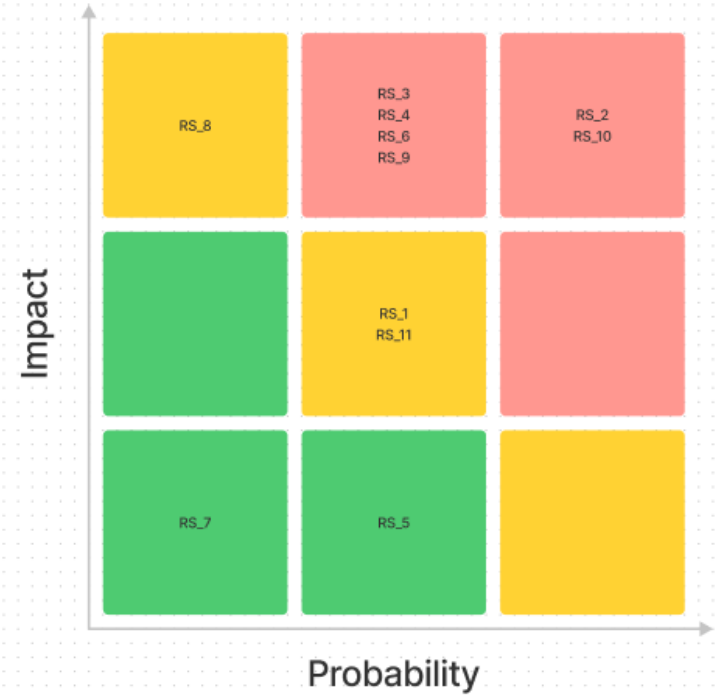



Рисунок Б.4 – Матриця ймовірності

Отриманий індекс для кожного ризику відповідає класифікації, які представлені у таблиці Б.4. Наступним етапом потрібно розробити стратегії реагування на кожну загрозу, вони описані у таблиці Б.5.

Таблиця Б.4 – Шкала оцінювання за рівнем ризику.

| № | Назва | Межі | Ризики, які входять (номера) |
|---|---|---|---|
| 1 | Прийнятні | 1≤R≤2 | RS_7, RS_5 |
| 2 | Виправдані | 3≤R≤4 | RS_8, RS_1, RS_11 |
| 3 | Недопустимі | 6≤R≤9 | RS_3, RS_4, RS_6, RS_9, RS_2, RS_10 |

Таблиця Б.5 – Ризики та стратегії реагування

| № | Статус ризику | Опис ризику | Ймовірність виникнення | Вплив ризику | Ранг ризику | План А | Тип стратегії реагування | План Б |
|---|---|---|---|---|---|---|---|---|
| RS_1 | Відкритий | Розбіжності у поглядах розробника та замовника | Середня | Середній | 4 | 1. Визначити можливу першопричину виникнення розбіжностей.<br>2. Спробувати усунути причину<br>3. Налагодити стосунки | Попередження | Створити здорову атмосферу |
| RS_2 | Відкритий | Допущення помилок під час проектування проекту | Високий | Високий | 9 | 1. Своєчасна перевірка різних даних.<br>2. Залучення контролюючої особи. | Пом’якшення | Проведення постійного контролю та перевірки етапів проектування. |

Продовження таблиці Б.5

| № | Статус ризику | Опис ризику | Ймовірність виникнення | Вплив ризику | Ранг ризику | План А | Тип стратегії реагування | План Б |
|---|---|---|---|---|---|---|---|---|
| RS_3 | Відкритий | Втрата результатів розробки через необачність | Середня | Високий | 9 | 1. Використовувати ПЗ, що включає функцію автоматичного збереження.<br>2. Використовувати резервне копіювання. | Попередження | Розробити резервні сховища та шляхи відновлення даних |
| RS_4 | Відкритий | Зміна ТЗ під час розробки проекту | Середня | Високий | 8 | 1. Своєчасно виділити вимоги до проекту.<br>2. Обговорити з замовником та командою всі технічні питання і умови реалізації. | Пом’якшення | Переоцінка та перепланування проекту після зміни. |

Продовження таблиці Б.5

| № | Статус ризику | Опис ризику | Ймовірність виникнення | Вплив ризику | Ранг ризику | План А | Тип стратегії реагування | План Б |
|---|---|---|---|---|---|---|---|---|
| RS_5 | Відкритий | Вимкнення світла під час розробки | Середня | Низький | 2 | 1. Мати резервний генератор або техніку з акумуляторними батареями. | Попередження | Моніторити повідомлення про планові відключення та погіршення погоди |
| RS_6 | Відкритий | Зміна ТЗ під час розробки проекту | Середня | Високий | 8 | 1. Своєчасно виділити та зафіксувати вимоги до проекту.<br>2. Обговорити з замовником та командою всі технічні питання і умови реалізації. | Пом’якшення | Переоцінка та перепланування проекту після зміни. |

Продовження таблиці Б.5

| **№** | **Статус ризику** | **Опис ризику** | **Ймовірність виникнення** | **Вплив ризику** | **Ранг ризику** | **План А** | **Тип стратегії реагування** | **План Б** |
|---|---|---|---|---|---|---|---|---|
| RS_7 | Відкритий | Реалізація зайвого функціоналу | Низька | Низький | 2 | 1. Своєчасно обговорити та виділити необхідний функціонал | Попередження | Попередити замовника про можливі зміни в функціоналі |
| RS_8 | Відкритий | Виникнення невідповідностей з ПЗ користувачів | Низька | Високий | 3 | 1. Враховувати вимоги до ПЗ.<br>2. Модифікувати та уніфікувати версії ПЗ. | Попередження | Розробити шляхи швидкого вирішення |
| RS_9 | Відкритий | Вибір неефективних технологій розробки | Середня | Високий | 7 | 1. Проаналізувати можливі технології<br>2. Проаналізувати потреби при розробці<br>3. Обрати оптимальну | Попередження | Звернутися за порадою до компетентних осіб |

Продовження таблиці Б.5

| **№** | **Статус ризику** | **Опис ризику** | **Ймовірність виникнення** | **Вплив ризику** | **Ранг ризику** | **План А** | **Тип стратегії реагування** | **План Б** |
|---|---|---|---|---|---|---|---|---|
| RS_10 | Відкритий | Виникнення проблем з розподіленням часу | Висока | Високий | 7 | 1. Провести реорганізацію робочого процесу<br>2. Створити план виконання | Пом’якшення | Скоригувати пріоритетність робіт та спробувати оптимізувати існуючу систему |
| RS_11 | Відкритий | Низька кваліфікація виконавців | Середня | Середній | 5 | 1. Підвищити кваліфікацію<br>2. Надати необхідні навчальні матеріали. | Пом’якшення | Переоцінка процесів з врахуванням навичок |

# ДОДАТОК В

## Web application to support the activities of the administrator in KZAPR

Dmytro Hlukhovtsov, *Student of IT-82*; Viktor Nenia, *Associate Professor*

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

All organizations faces the implementation and planning of large amounts of tasks for staff every day. This is quite a time consuming process. Without planning the tasks and delegating authorities, it is impossible to plan any process at every level of management of any organization.

With the development of information and digital technologies, the creating of systems that can organize the construction of such interactions is one of the important research areas. First of all, they are aimed at designing and implementing individual solutions that will be able to provide the necessary functionality for the management and control of processes and staff, as the part of companies that "go online" is growing each year. That is what project organizations have become also.

One of the most important components of such system is the administrator module. It provides basic functionality for control over the integrity of mechanisms, user actions as well as virtual file storage for a variety of digitized paper documents. Thus, in particular, it can manage access levels and the internal structure of various data warehouses, which maintains data integrity and to some extent the security of workflows.

To specify the use scope of the administrator module, it has been decided to develop it as a web-oriented subsystem of the tools complex of automation the designing works (KZAPR). The considering issues are related to the support of the administrator's field of activity. The solution of this problem includes all the previously listed functions and mechanisms of interaction and provides maximum comfort and productivity from use of this module by the client.

It has been decided to implement this web application using HTML, Express web-applications framework for Node.js for the server side, React JavaScript library for the client side, SASS metalanguage for styles and GitHub version control system for cooperative work within the project.

The result of presented research is a web application to support the activities of the administrator in KZAPR. Its use providers clients with comfort work through understandable interface as an administrator of the designing organization.

# ДОДАТОК Г

## Лістинг програмного коду основних модулів web-додатку

### Модулі реєстрації

LoginPage.js

```
import React from 'react';
import { Link } from 'react-router-dom';
import LoginForm from '../../../../components/auth/LoginForm';
import signinImg from '../../../../styles/inc/images/signin_img.png';
import siteIconSvg from '../../../../icons/citeIcon.svg';

const LoginPage = () => (
  <div className="log-wrapper">
    <div id="signin-form" className="sign-form">
      <div id="form-header">
        <div className="site-logo">
          <img src={siteIconSvg} alt="logo" />
        </div>
        <h1 className="form-header">Sign In</h1>
      </div>
      <div className="log-form-wrapper">
        <div className="log-image-side-block">
          <img src={signinImg} alt="signin-img" />
          <div className="next-sign-form">
            <Link to="/auth/registry">
              Create an account
            </Link>
          </div>
        </div>

        <div className="log-form-side-block">
          <LoginForm />
          <div id="alt-login" />

        </div>
      </div>

    </div>
  </div>
);

export default LoginPage;
```

LoginForm.js

```
import * as Yup from 'yup';
import React, { useState } from 'react';
import { useFormik } from 'formik';
import {
  Checkbox, FormControlLabel, InputAdornment, TextField,
} from '@material-ui/core';
```

```
import Button from '@mui/material/Button';
import IconButton from '@mui/material/IconButton';
import CloseIcon from '@mui/icons-material/Close';
import { Alert } from '@mui/material';
import emailIconSvg from '../../icons/emailIcon.svg';
import passwordIconSvg from '../../icons/passwordIcon.svg';
import eyeIconSvg from '../../icons/eye-icon.svg';
import closeEyeIconSvg from '../../icons/close-eye-icon.svg';
import GoogleAuth from './googleAuth';
import handleLogin from './handleLogin';

const validationSchema = Yup.object({
  signinEmail: Yup
    .string('Enter your email')
    .email('Enter a valid email')
    .required('Email is required'),
  signinPassword: Yup
    .string('Enter your password')
    .min(8, 'Password should be of minimum 8 characters length')
    .required('Password is required'),
});

const LoginForm = () => {
  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState(null);
  const [open, setOpen] = useState(true);
  if (errors && !open) {
    setOpen(true);
  }
  const handleClickShowPassword = () => {
    setShowPassword(!showPassword);
  };

  const handleMouseDownPassword = (event) => {
    event.preventDefault();
  };

  const formik = useFormik({
    initialValues: {
      signinEmail: '',
      signinPassword: '',
      rememberCheck: false,
    },
    validationSchema,
    onSubmit: async (values) => {
      const res = handleLogin(values);
      if (res.text === 'success') {
        await sessionStorage.setItem('token', JSON.stringify(values));
        await sessionStorage.setItem('role', 'admin');
        document.location.reload();
      } else {
        setErrors({ text: res.text });
      }
    },
  });

  return (
    <>
      <div className="log-form">
        <form onSubmit={formik.handleSubmit} id="login-form">
          <TextField
```

```
            fullWidth
            className="input-block"
            id="signinEmail"
            name="signinEmail"
            placeholder="Email"
            value={formik.values.signinEmail}
            onChange={formik.handleChange}
            error={formik.touched.signinEmail && Boolean(formik.errors.signinE-
mail)}
            helperText={formik.touched.signinEmail && formik.errors.signinEmail}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <img src={emailIconSvg} alt="email" />
                </InputAdornment>
              ),
            }}
          />
          <TextField
            className="input-block"
            fullWidth
            id="signinPassword"
            name="signinPassword"
            placeholder="Password"
            type={showPassword ? 'text' : 'password'}
            value={formik.values.signinPassword}
            onChange={formik.handleChange}
            error={formik.touched.signinPassword && Boolean(formik.er-
rors.signinPassword)}
            helperText={formik.touched.signinPassword && formik.errors.signinPass-
word}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <img src={passwordIconSvg} alt="email" />
                </InputAdornment>
              ),
              endAdornment: (
                <InputAdornment position="end">
                  <img
                    aria-label="toggle password visibility"
                    onClick={handleClickShowPassword}
                    onMouseDown={handleMouseDownPassword}
                    src={showPassword ? eyeIconSvg : closeEyeIconSvg}
                    alt="password"
                  />
                </InputAdornment>
              ),
            }}
            variant="standard"
          />
          <div id="remember-block">
            <FormControlLabel
              name="rememberCheck"
              control={(
                <Checkbox
                  color="default"
                />
            )}
              label="Remember me"
            />
```

```
          </div>
          <Button id="signup-btn" color="primary" variant="contained" type="sub-
mit">
            Login
          </Button>
        </form>
        <div className="auth-footer-block">
          <GoogleAuth />
        </div>

      </div>
      {errors && (
      <Alert
        className={`alert-message ${open === true ? 'open' : 'close'}`}
        severity="error"
        action={(
          <IconButton
            aria-label="close"
            color="inherit"
            size="small"
            onClick={() => {
              setErrors(null);
              setOpen(false);
            }}
          >
            <CloseIcon fontSize="inherit" />
          </IconButton>
                )}
        sx={{ mb: 2 }}
      >
        {errors?.text}
      </Alert>
      )}
    </>
  );
};

export default LoginForm;
```

## Форма реєстрації

### RegistryForm.js

```
import React, { useState } from 'react';
import { useFormik } from 'formik';
import * as Yup from 'yup';
import { InputAdornment, TextField } from '@material-ui/core';
import Button from '@mui/material/Button';
import { Alert } from '@mui/material';
import IconButton from '@mui/material/IconButton';
import CloseIcon from '@mui/icons-material/Close';
import emailIconSvg from '../../icons/emailIcon.svg';
import passwordIconSvg from '../../icons/passwordIcon.svg';
import repeatPasswordIconSvg from '../../icons/repeatPassword.svg';
import eyeIconSvg from '../../icons/eye-icon.svg';
import closeEyeIconSvg from '../../icons/close-eye-icon.svg';
import GoogleAuth from './googleAuth';
import handleRegistry from './handleRegistry';
```

```
const validationSchema = Yup.object({
  signupEmail: Yup
    .string('Enter your email')
    .email('Enter a valid email')
    .required('Email is required'),
  signupPassword: Yup
    .string('Enter your password')
    .min(8, 'Password should be of minimum 8 characters length')
    .required('Password is required'),
  repeatPassword: Yup
    .string('Enter your password')
    .min(8, 'Password should be of minimum 8 characters length')
    .required('Password is required')
    .oneOf([Yup.ref('signupPassword'), null], 'Passwords must match'),
});

const RegisterForm = () => {
  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState(null);
  const [open, setOpen] = useState(true);
  if (errors && !open) {
    setOpen(true);
  }
  const handleClickShowPassword = () => {
    setShowPassword(!showPassword);
  };

  const handleMouseDownPassword = (event) => {
    event.preventDefault();
  };

  const formik = useFormik({
    initialValues: {
      signupEmail: '',
      signupPassword: '',
      repeatPassword: '',
    },
    validationSchema,
    onSubmit: async (values) => {
      const res = handleRegistry(values);
      if (res.text === 'success') {
        await sessionStorage.setItem('token', JSON.stringify(values));
        await sessionStorage.setItem('role', 'admin');
        document.location.reload();
      } else {
        setErrors({ text: res.text });
      }
    },
  });

  return (
    <>
      <div className="log-form">
        <form onSubmit={formik.handleSubmit} id="register-form">
          <TextField
            fullWidth
            className="input-block"
            id="signupEmail"
            name="signupEmail"
            placeholder="Email"
            value={formik.values.signupEmail}
```

```
            onChange={formik.handleChange}
            error={formik.touched.signupEmail &&
(Boolean(formik.errors.signupEmail))}
            helperText={formik.touched.signupEmail &&
(formik.errors.signupEmailt)}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <img src={emailIconSvg} alt="email" />
                </InputAdornment>
              ),
            }}
          />
          <TextField
            className="input-block"
            fullWidth
            id="signupPassword"
            name="signupPassword"
            placeholder="Password"
            type={showPassword ? 'text' : 'password'}
            value={formik.values.signupPassword}
            onChange={formik.handleChange}
            error={formik.touched.signupPassword &&
Boolean(formik.errors.signupPassword)}
            helperText={formik.touched.signupPassword &&
formik.errors.signupPassword}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <img src={passwordIconSvg} alt="email" />
                </InputAdornment>
              ),
              endAdornment: (
                <InputAdornment position="end">
                  <img
                    aria-label="toggle password visibility"
                    onClick={handleClickShowPassword}
                    onMouseDown={handleMouseDownPassword}
                    src={showPassword ? eyeIconSvg : closeEyeIconSvg}
                    alt="password"
                  />
                </InputAdornment>
              ),
            }}
            variant="standard"
          />
          <TextField
            className="input-block"
            fullWidth
            id="repeatPassword"
            name="repeatPassword"
            type={showPassword ? 'text' : 'password'}
            value={formik.values.repeatPassword}
            onChange={formik.handleChange}
            error={formik.touched.repeatPassword &&
Boolean(formik.errors.repeatPassword)}
            helperText={formik.touched.repeatPassword &&
formik.errors.repeatPassword}
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
```

```
                <img src={repeatPasswordIconSvg} alt="email" />
              </InputAdornment>
            ),
            endAdornment: (
              <InputAdornment position="end">
                <img
                  aria-label="toggle password visibility"
                  onClick={handleClickShowPassword}
                  onMouseDown={handleMouseDownPassword}
                  src={showPassword ? eyeIconSvg : closeEyeIconSvg}
                  alt="password"
                />
              </InputAdornment>
            ),
          }}
          placeholder="Repeat your password"
        />
        <Button id="signup-btn" color="primary" variant="contained"
type="submit">
          Register
        </Button>
      </form>
      <div className="auth-footer-block">
        <GoogleAuth />
      </div>
    </div>
    {errors && (
      <Alert
        className={`alert-message ${open === true ? 'open' : 'close'}`}
        severity="error"
        action={(
          <IconButton
            aria-label="close"
            color="inherit"
            size="small"
            onClick={() => {
              setErrors(null);
              setOpen(false);
            }}
          >
            <CloseIcon fontSize="inherit" />
          </IconButton>
          )}
        sx={{ mb: 2 }}
      >
        {errors?.text}
      </Alert>
    )}
  </>
 );
};

export default RegisterForm;
```

## Модуль для виведення інформації про зареєстрованого користувача

UserPage.js

```
const UserPage = ({ handleModalTaskOpen }) => {
  const { userId } = useParams();
```

```
  return (
    <div className="admin-page-wrapper">
      <div className={`main-task-body page-info ${userId}`}>
        <div className="left-bar">
          <AvatarPage
            avatarLink={profile.avatarLink}
            title={`${profile.name} ${profile.surname}`}
          />
        </div>

        <div className="content">
          <TitlePage
            type={profile.position !== 'customer'
              ? (<ProfileEditModal profile={projectProfile} />)
              : ('Customer')}
            title={`${profile.name} ${profile.surname}`}
          />
          <div className="block-content">
            <ContentProfilePage
              aboutMe={profile.aboutMe}
              pm={profile.pm}
              position={profile.position}
              email={profile.email}
              birthday={profile.birthday}
            />
          </div>
        </div>

      </div>
      {profile.position !== 'customer'
        ? (<UserTables users={usersEmp} tasks={tasksList}
handleModalTaskOpen={handleModalTaskOpen} />)
        : (<CustomerTables contracts={contract} />)}

    </div>
  );
};

export default UserPage;
```

## Модуль для виведення даних в табличному форматі

DataTable.js

```
import React, { useState } from 'react';
import {
  DataGrid,
  GridActionsCellItem,
} from '@mui/x-data-grid';
import DeleteIcon from '@mui/icons-material/Delete';
import ToolBar from './toolBar';

const DataTable = ({
  rows, columns, isDelete, rowOnPage = 30, handleRowClick, typeToolbar = false,
...props
}) => {
  const deleteUser = (event) => {
    event.stopPropagation();
  };
```

```
  const deleteColumn = [
    {
      field: 'action',
      headerName: 'Action',
      width: 60,
      renderCell: (params) => [
        <GridActionsCellItem
          key={`del-${params.id}`}
          icon={<DeleteIcon />}
          label="Delete"
          onClick={(e) => deleteUser(e, params.id)}
          showInMenu={false}
        />],
    },
  ];


  const [column, setColumns] = useState(columns);
  const isInArray = (element) => element.field === 'action';

  if (isDelete && column.some(isInArray) === false) {
    setColumns([...columns, ...deleteColumn]);
  }

  return (
    <div className="table-wrapper" style={{ width: '70%' }}>
      <DataGrid
        className="data-table"
        rows={rows}
        columns={column}
        pageSize={rowOnPage}
        rowsPerPageOptions={[rowOnPage]}
        autoHeight
        onRowClick={(rowData) => handleRowClick(rowData.row)}
        components={
          { Toolbar: () => ToolBar(typeToolbar, true) }
        }
        disableColumnMenu
        {...props}
      />
    </div>

  );
};

export default DataTable;
```

**Лістинг коду програмного модулю, який описує логіку побудови структури колонок таблиці з проектами**

projectsTableColumns.js

```
import React from 'react';
import PopoverCell from '../../components/admin/additional/popoverCell';
import { dateFormat } from '../time';

const ProjectTableColumns = [
  { field: 'id', headerName: 'ID', width: 70 },
```

```
  {
    field: 'title', headerName: 'Title', width: 150, flex: 1, renderCell: (params)
=> (<PopoverCell text={params.row.title} />),
  },
  {
    field: 'description', headerName: 'Description', width: 150, flex: 1, render-
Cell: (params) => (<PopoverCell text={params.row.description} />),
  },
  {
    field: 'pm',
    headerName: 'Project Manager',
    width: 120,
    renderCell: (params) => (
      <a href={`/profile/${params.row.pm.profileId}`} className="link-button" on-
Click={(e) => e.stopPropagation()}>
        {params.row.pm.name || ''}
        {' '}
        {params.row.pm.surname || ''}
      </a>
    ),
    valueGetter: (params) => `${params.row.pm.name || ''} ${params.row.pm.surname
|| ''}`,
  },
  {
    field: 'start', valueGetter: (params) => `${dateFor-
mat(params.row.dates.start)}`, headerName: 'Start', width: 150,
  },
  {
    field: 'end', valueGetter: (params) => `${dateFormat(params.row.dates.end)}`,
headerName: 'End', width: 150,
  },

  {
    field: 'contract',
    headerName: 'Contract',
    width: 120,
    renderCell: (params) => (<PopoverCell text={params.row.contract.title}
path="contracts" data={params.row.contract.id} isLink />),
    valueGetter: (params) => params.row.contract.title,
  },

];

export default ProjectTableColumns;
```

## Лістинг коду модулю для контролю правил та ролей

rulesPage.js

```
const RulesPage = () => {
  const resources = resource;
  const permissions = permission;
  const roles = role;
  const rules = rule;
  const handleRowClick = () => {

  };
  return (
    <div className="main-task-body rules">
```

```
<div className="admin-accordion-wrapper">
  <Accordion
    className="admin-table-accordion"
  >
    <AccordionSummary
      expandIcon={<ExpandMoreIcon />}
      aria-controls="panel1a-content"
      id="panel1a-header"
      className="accordion-header"
    >
      <Typography variant="h4">Resources</Typography>
    </AccordionSummary>
    <AccordionDetails className="accordion-body">
      <DataTable
        isDelete={false}
        rows={resources}
        columns={rulePartsTableColumns}
        handleRowClick={handleRowClick}
        typeToolbar="ruleParts"
      />
    </AccordionDetails>
  </Accordion>

  <Accordion
    className="admin-table-accordion"
  >
    <AccordionSummary
      expandIcon={<ExpandMoreIcon />}
      aria-controls="panel1a-content"
      id="panel1a-header"
      className="accordion-header"
    >
      <Typography variant="h4">Roles</Typography>
    </AccordionSummary>
    <AccordionDetails className="accordion-body">
      <DataTable
        isDelete={false}
        rows={roles}
        columns={rulePartsTableColumns}
        handleRowClick={handleRowClick}
        typeToolbar="ruleParts"
      />

    </AccordionDetails>
  </Accordion>
  <Accordion
    className="admin-table-accordion"
  >
    <AccordionSummary
      expandIcon={<ExpandMoreIcon />}
      aria-controls="panel3a-content"
      id="panel3a-header"
      className="accordion-header"
    >
      <Typography variant="h4">Permissions</Typography>
    </AccordionSummary>
    <AccordionDetails className="accordion-body">
      <DataTable
        isDelete={false}
        rows={permissions}
        columns={rulePartsTableColumns}
```

```
              handleRowClick={handleRowClick}
              typeToolbar="ruleParts"
            />
          </AccordionDetails>
        </Accordion>

        <Accordion
          className="admin-table-accordion"
        >
          <AccordionSummary
            expandIcon={<ExpandMoreIcon />}
            aria-controls="panel4a-content"
            id="panel4a-header"
            className="accordion-header"
          >
            <Typography variant="h4">Rules</Typography>
          </AccordionSummary>
          <AccordionDetails className="accordion-body">
            <DataTable
              isDelete={false}
              rows={rules}
              columns={ruleTableColumns}
              handleRowClick={handleRowClick}
              typeToolbar="rules"

            />
          </AccordionDetails>
        </Accordion>
      </div>
    </div>
  );
};

export default RulesPage;
```

## Лістинг коду модулю модального вікна для зміни правил

ruleEditModal.js

```
const RuleEditModal = ({ show, handleModal }) => {
  const isAdd = Object.keys(show.value).length === 0;
  const formValues = {
    role: show.value?.role?.name ? {
      name: show.value?.role?.name,
      id: show.value?.role?.id,
    } : null,
    permission: show.value?.permission?.name ? {
      name: show.value?.permission?.name,
      id: show.value?.permission?.id,
    } : null,
    resource: show.value?.resource?.name ? {
      name: show.value?.resource?.name,
      id: show.value?.resource?.id,
    } : null,
  };

  const formSchema = Yup.object().shape({
    role: Yup.object().required('Enter role').nullable(),
    resource: Yup.object().required('Enter role').nullable(),
```

```
  permission: Yup.object().required('Enter role').nullable(),
});
const handleSubmit = (formData, { setSubmitting }) => {
  setTimeout(() => {
    // eslint-disable-next-line no-console
    console.log(formData, show.value?.id);
    setSubmitting(false);
  }, 300);
};
return (
  <Modal
    open={show.open}
    onClose={handleModal}
    sx={{
      height: '100vh',
      overflow: 'auto',
    }}
  >
    <div className="modal-of modal-of-center notification-modal modal-of-show">
      <div className="modal notification-modal-view">
        <Formik
          initialValues={formValues}
          onSubmit={handleSubmit}
          validationSchema={formSchema}
        >
          {({
            isValid, dirty,
          }) => (
            <Form className="form form-notification">
              <div className="modal-header">
                <div className="title-header text text-20">
                  Rules
                </div>

                <IconButton
                  aria-label="close"
                  size="medium"
                  onClick={handleModal}
                  sx={{
                    marginRight: '5px',
                  }}
                >
                  <CloseRoundedIcon />
                </IconButton>
              </div>
              <div className="modal-body">
                <Field
                  component={Autocomplete}
                  name="role"
                  label="Role"
                  placeholder="Role"
                  variant="outlined"
                  options={role}
                  optionLabel={(option) => `${option.name}`}
                  optionEqual={(option, value) => option.id === value.id}
                  className="form-field"
                />
                <Field
                  component={Autocomplete}
                  name="resource"
                  label="Resource"
```

```
                    placeholder="Resource"
                    variant="outlined"
                    options={resource}
                    optionLabel={(option) => `${option.name}`}
                    optionEqual={(option, value) => option.id === value.id}
                    className="form-field"
                  />
                  <Field
                    component={Autocomplete}
                    name="permission"
                    label="Permission"
                    placeholder="Permission"
                    variant="outlined"
                    options={permission}
                    optionLabel={(option) => `${option.name}`}
                    optionEqual={(option, value) => option.id === value.id}
                    className="form-field"
                  />

                  <div className="btn-row">
                    <Button
                      type="submit"
                      variant="contained"
                      size="small"
                      disabled={!(isValid && dirty)}
                    >
                      {isAdd ? 'Add' : 'Change'}
                    </Button>
                  </div>
                </div>
              </Form>
            )}
          </Formik>
        </div>
      </div>
    </Modal>
  );
};

RuleEditModal.propTypes = {
  show: PropTypes.shape({
    open: PropTypes.bool.isRequired,
    value: PropTypes.shape({
      role: PropTypes.shape({}),
      permission: PropTypes.shape({}),
      resource: PropTypes.shape({}),
    }),

  }).isRequired,
  handleModal: PropTypes.func.isRequired,
};

export default RuleEditModal;
```

## Лістинг коду модулів налаштувань користувача

SettingsPage.js

```
import React from 'react';
```

```
import PersonalInformationForm from '../../components/settings/PersonalInfor-
mationForm';
import ContactInformationForm from '../../components/settings/ContactInformation-
Form';
import SecurityInformationForm from '../../components/settings/SecurityInfor-
mationForm';

const SettingsPage = () => {
  const profileInfo = {
    profileId: 1,
    email: 'rajkodima@gmail.com',
    name: 'Dmitry',
    surname: 'Raiko',
  };

  return (
    <div id="settings-page-main-block">
      <h1>Account Information</h1>

      <div id="info-blocks-wrapper">
        <PersonalInformationForm profile={profileInfo} />

        <ContactInformationForm profile={profileInfo} />

        <SecurityInformationForm profile={profileInfo} />
      </div>
    </div>
  );
};

export default SettingsPage;
```

## SecurityInformationForm.js

```
import React, { useContext, useState } from 'react';
import { Button } from '@mui/material';
import { LanguageContext } from '../../containers/language/Language';
import ChangePassword from '../../containers/modals/ChangePassword';

const SecurityInformationForm = () => {
  const { dictionary } = useContext(LanguageContext);
  const [resetPassword, setResetPassword] = useState(false);

  const handleResetPassword = () => {
    setResetPassword(!resetPassword);
  };

  return (
    <>
      <div id="security-info" className="info-block">
        <div className="info-blocks-heading">
          <h3>
            {dictionary.pages.settings.security}
          </h3>
        </div>
        <div className="info-blocks-body">
          <Button
            fullWidth
            className="security-btn"
            data-modal="change-password"
```

```
            color="inherit"
            onClick={handleResetPassword}
          >
            {dictionary.pages.settings.changePassword}
          </Button>

          <Button
            className="security-btn"
            fullWidth
          >
            {dictionary.header.menu.logout}
          </Button>
        </div>
      </div>

      {resetPassword && (
      <ChangePassword
        show={resetPassword}
        handleModal={handleResetPassword}
      />
      )}
    </>
  );
};

export default SecurityInformationForm;
```

## ChangePassword.js

```
/* eslint-disable react/jsx-props-no-spreading */
import React, { useContext } from 'react';
import { Field, Form, Formik } from 'formik';
import * as Yup from 'yup';
// materials [MUI]
import {
  IconButton, Modal, Button,
} from '@mui/material';
// icon materials [MUI]
import CloseRoundedIcon from '@mui/icons-material/CloseRounded';
// own containers
import { TextField } from 'formik-mui';
// own components
import { LanguageContext } from '../language/Language';

const ChangePassword = ({ show, handleModal }) => {
  const { dictionary } = useContext(LanguageContext);
  const formValues = {
    password: '',
    passwordR: '',
  };

  const formSchema = Yup.object().shape({
    password: Yup.string().required('Enter password').min(8, 'Too short'),
    passwordR: Yup.string().oneOf([Yup.ref('password'), null], 'Passwords do not
match').required('Repeat your password').min(8, 'Too short'),
  });

  const sendHelp = (formData, { setSubmitting }) => {
    setTimeout(() => {
      // eslint-disable-next-line no-console
```

```
    console.log(formData);
    setSubmitting(false);
  }, 300);
};

return (
  <Modal
    open={show}
    onClose={handleModal}
    sx={{
      height: '100vh',
      overflow: 'auto',
    }}
  >
    <div className="modal-of modal-of-center notification-modal modal-of-show">
      <div className="modal notification-modal-view">
        <Formik
          initialValues={formValues}
          onSubmit={sendHelp}
          validationSchema={formSchema}
        >
          {({
            isValid, dirty,
          }) => (
            <Form className="form form-notification">
              <div className="modal-header">
                <div className="title-header text text-20">
                  {dictionary.pages.settings.changePassword}
                </div>

                <IconButton
                  aria-label="close"
                  size="medium"
                  onClick={handleModal}
                  sx={{
                    marginRight: '5px',
                  }}
                >
                  <CloseRoundedIcon />
                </IconButton>
              </div>
              <div className="modal-body">
                <Field
                  component={TextField}
                  className="form-field text text-19"
                  type="password"
                  name="password"
                  label={dictionary.password}
                  variant="standard"
                  placeholder={dictionary.password}
                  fullWidth
                />

                <Field
                  component={TextField}
                  className="form-field text text-19"
                  type="password"
                  name="passwordR"
                  label={dictionary.repeatPassword}
                  variant="standard"
                  placeholder={dictionary.repeatPassword}
```

```
                  fullWidth
                />

                <div className="btn-row">
                  <Button
                    type="submit"
                    variant="contained"
                    size="small"
                    disabled={!(isValid && dirty)}
                  >
                    {dictionary.change}
                  </Button>
                </div>
              </div>
            </Form>
          )}
        </Formik>
      </div>
    </div>
  </Modal>
 );
};

export default ChangePassword;
```