МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«SaaS платформа, де користувачі реєструють компанії, додають працівників та виконують свою роботу всередині веб-додатку»

Виконав студент групи ІПЗ-19-2
ГОРДІЄНКО Микола Геннадійович

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
МАРЧУК Галина Вікторівна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
В.о. зав. кафедри інженерії програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: Гордієнко Микола Геннадійович
Керівник роботи: Кравченко Світлана Миколаївна
Тема роботи: «SaaS платформа, де користувачі реєструють компанії, додають працівників та виконують свою роботу всередині веб-додатку»,
затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с
Вихідні дані для роботи: Об’єктом дослідження є використання інформаційних технологій для полегшення управління компаніями та проектами.
Предметом дослідження є використання сучасних та веборієнтованих технологій розробки для створення вебзастосунку, що допомагає полегшення управління компаніями та проектами.
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Кравченко С.М.	16.02.2023р.	16.02.2023р.
2	Кравченко С.М.	16.02.2023р.	16.02.2023р.
3	Кравченко С.М.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання	16.02.2023-16.03.2023	Виконано
2	Проектування структури системи	16.03.2023-16.04.2023	Виконано
3	Написання програмного коду	16.04.2023-01.06.2023	Виконано
4	Тестування	01.06.2023-07.06.2023	Виконано
5	Оформлення пояснювальної записки	07.06.2023-27.06.2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу для системи управління компаніями та проектами, а також пояснювальної записки. Пояснювальна записка містить 83 сторінок, 79 ілюстрацій та 32 таблиць.

Метою роботи є розробка системи, яка дозволяє управляти компаніями та проектами. Система включає модулі для управління компаніями, проектами, співробітниками, а також відстеження прогресу робіт та розподілу матеріалів.\

У роботі визначено основні завдання розробки системи, проведений аналіз аналогічних систем. Було обрано архітектурний патерн CQRS для написання програмного комплексу як SPA. Представлені сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис бази даних (СУБД PostgreSQL), алгоритми взаємодії модулів системи, об'єктна структура системи (діаграма класів) та алгоритми роботи основних модулів. Реалізація серверної частини здійснена мовою програмування C# в середовищі .NET, а клієнтська частина реалізована з використанням Angular.

КЛЮЧОВІ СЛОВА: КОМПАНІЯ, ПРОЕКТ, РОБІТНИКИ, ФАЗИ, SINGLE PAGE APPLICATION, УПРАВЛІННЯ ПРОЕКТОМ.

ABSTRACT

The bachelor's thesis consists of a software package for a company and project management system and an explanatory note. The explanatory note contains 83 pages, 79 illustrations and 32 tables.

The aim of the work is to develop a system that allows managing companies and projects. The system includes modules for managing companies, projects, employees, as well as tracking the progress of work and distribution of materials.

The paper identifies the main tasks of system development and analyses similar systems. The CQRS architectural pattern was chosen to write the software system as a SPA. The paper presents scenarios of the software system, the general structure of the software system, a description of the database (PostgreSQL), algorithms for interaction of system modules, the object structure of the system (class diagram) and algorithms for the main modules. The server part is implemented in the C# programming language in the .NET environment, and the client part is implemented using Angular.

KEYWORDS: COMPANY, PROJECT, WORKERS, PHASES, SINGLE PAGE APPLICATION, PROJECT MANAGEMENT.

ЗМІСТ

ВСТУП **8**

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ УПРАВЛІННЯ КОМПАНІЯМИ **10**

1.1. Постановка задачі 10

1.2. Аналіз аналогів програмного продукту 11

1.3. Вибір архітектури системи управління компаніями 16

Висновки до розділу 1 22

РОЗДІЛ 2. ПРОЕКТУВАННЯ SAAS ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ КОМПАНІЯМИ **23**

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи 23

2.2. Розробка бази даних системи 33

2.3. Проектування та реалізація алгоритмів роботи системи 43

Висновки до розділу 2 49

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА **50**

3.1. Порядок встановлення та налаштування системи 50

3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з веб-орієнтованим інформаційним додатком 51

3.2.1. Функціональні можливості адміністратора. 53

3.2.2. Функціональні можливості операційного менеджера. 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **82**

ДОДАТКИ **83**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRUD – create, read, update, delete (*з англ.* створити, прочитати, оновити, видалити)

ERM – entity-relationship model (*з англ.* модель сутність — зв'язок)

SPA – single–page–application (*з англ.* односторінковий веб-додаток)

БД – база даних

СКБД – система керування базами даних

ВСТУП

Актуальність теми полягає в розробці SaaS - платформи для централізованого управління компаніями та працівниками. Додаток забезпечує зручну реєстрацію компаній, додавання працівників та виконання роботи всередині веб-додатку. Важливі аспекти цієї системи включають:

1. Централізоване управління компаніями і працівниками: Забезпечення єдиного місця для зберігання інформації про компанії та їх працівників, спрощення управління та підвищення ефективності роботи.
2. Multi-tenancy рішення: використання однієї бази даних для всіх компаній забезпечує економічну ефективність та зручність у використанні для користувачів.
3. Ефективність та зручність використання: Забезпечення зручного інтерфейсу та доступу з будь-якого пристрою, що дозволяє користувачам ефективно виконувати свою роботу всередині додатку.
4. Забезпечення безпеки та оновлень: Постачальники платформи беруть на себе відповідальність за безпеку даних, резервне копіювання та оновлення програмного забезпечення.

Метою випускної роботи є розробка SaaS-платформи для управління компаніями та працівниками з метою забезпечення зручності, ефективності та централізованого управління. Завдання включають аналіз потреб користувачів, проектування архітектури платформи, вибір відповідних технологій, розробку компонентів та реалізацію системи.

Об’єктом дослідження є платформа для управління компаніями та працівниками шляхом реєстрації компаній, додавання працівників та виконання роботи всередині веб-додатку. Дослідження спрямоване на вивчення технологій та методів, що дозволяють забезпечити зручне та ефективне управління цими компаніями та працівниками.

Предметом дослідження є архітектура та компонентна структура SaaS-платформи для управління компаніями та працівниками. Дослідження спрямоване на вивчення технологій та методів, що дозволяють забезпечити зручне та ефективне управління компаніями та працівниками в рамках цієї платформи.

Розроблена система може ефективно впроваджуватись в різні компанії для полегшення управління компаніями, працівниками та їх роботою. Вона забезпечує зручний доступ до необхідної інформації, поліпшує комунікацію та співпрацю, а також зменшує зусилля, витрачені на адміністративні завдання.

Пояснювальна записка до випускної роботи містить 83 сторінок, 79 ілюстрацій та 32 таблиці, список літературних джерел містить 14 найменування.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ УПРАВЛІННЯ КОМПАНІЯМИ

1.1. Постановка задачі

Основна мета цієї роботи полягає у розробці SaaS-платформи для управління компаніями та працівниками з метою забезпечення зручності, ефективності та централізованого управління. Ця система має на меті полегшити управління компаніями, працівниками та їх роботою шляхом надання зручного доступу до необхідної інформації, покращення комунікації та співпраці, а також зменшення зусиль, витрачених на адміністративні завдання.

1. Розробити дизайн клієнтської частини сайту. Розроблено дизайн клієнтської частини сайту з використанням Flux [10] архітектури та NgRx [11].
   1.1. Використання Flux [10] та NgRx [11] забезпечує односторонню потоку даних та керування станом додатку.
   1.2. Застосована модульна система для ефективного завантаження компонентів в браузер.
   1.3. Дизайн зосереджений на зручності використання та ефективності роботи користувачів.
2. Реалізувати ядро системи управління контентом.
   2.1. Написати програмні модулі для:
   – авторизації користувачів в системі. Розподіл прав доступу користувачів повинен визначатися за наступними ролями: адміністратор, операційний менеджер, керівник проекту;
   – можливість збереження інформації (компанії, проекти, профілі користувачів, матеріали, фази будівництва, поставники);
   – пошук та фільтрації для пошуку(проекту, компанії);
   – поставники та матеріали;

– отримання сповіщень про оновлення даних та повідомлень в реальному часі;

– часовий графік;

2.2. Реалізувати CRUD операції таблиць бази даних системи.

3. Реалізувати структуру збереження даних наступного виду:

3.1. Дані компаній (назва, адреса, email, дата приєднання, веб-сайт, опис);

3.2. Дані про проекти (назва, тип проекту, адреса, статус, бюджет, часові рамки виконання проекту);

3.3. Дані будівель які належать проекту (назва)

3.4. Дані користувачів (ім’я, фамілія, email, пароль, роль)

3.5. Дані постачальників (назва, номер телефону, email, сайт)

3.6. Дані матеріалів (ціна за одиницю, кількість, адреса компанії)

4. Створити демонстраційний та тестовий контент сайту.

5. Розробити план тестування системи управління контентом. Зокрема передбачити наступні види тестування:

5.1. Тестування функціональне.

5.2. Кросбраузерне та адаптивне тестування.

5.3. Тестування безпеки даних.

Результатом реалізації поставленого завдання є розгорнута веб-орієнтована система управління компаніями та працівниками, що містить в своєму складі серверну структуру збереження даних, багатокористувацький клієнтський додаток для реалізації функціональних можливостей, та засоби контролю та керування доступом.

1.2. Аналіз аналогів програмного продукту

Основною метою SaaS-платформи для управління компаніями та працівниками є полегшення та покращення процесу управління організацією та її працівниками. Платформа надає централізований доступ до необхідної

інформації та інструментів для управління компанією, полегшує спілкування та співпрацю між працівниками, забезпечує управління даними та аналітику, автоматизує рутинні процеси, покращує продуктивність та ефективність роботи працівників, а також забезпечує безпеку та конфіденційність даних в табл. 1.1.

Ось кілька прикладів існуючих аналогів SaaS-платформ для управління компаніями та працівниками:

1. Slack: Slack є популярною платформою для комунікації та співпраці внутрішньо в компанії. Вона надає можливості обміну повідомленнями, створення робочих груп, спільного доступу до файлів та багато іншого.

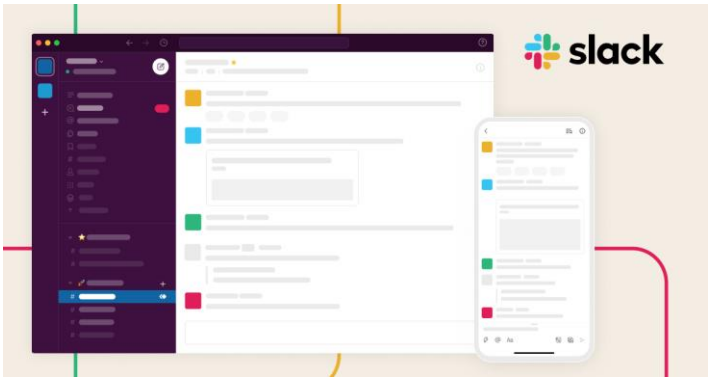


Рисунок 1.1 – Веб-додаток Slack

2. Asana: Asana є платформою для управління проектами та завданнями. Вона дозволяє створювати проекти, призначати завдання, встановлювати терміни виконання, відстежувати прогрес та спілкуватися з командою.

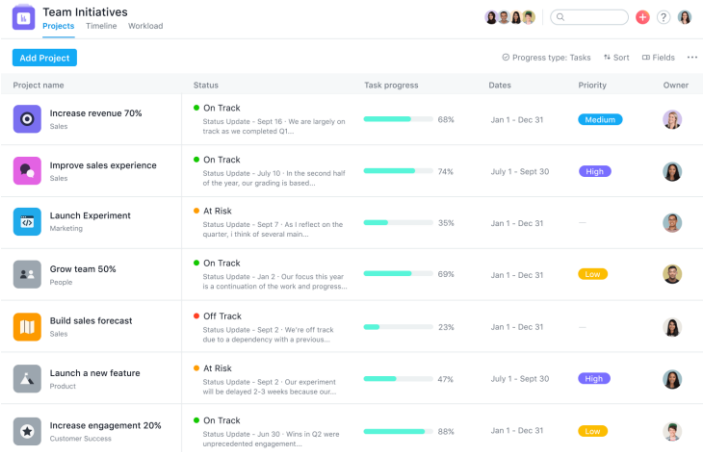


Рисунок 1.2 – Веб-додаток Asana

3. Salesforce: Salesforce є однією з провідних платформ управління відносинами з клієнтами (CRM). Вона надає засоби для керування продажами, маркетингом, обслуговуванням клієнтів та аналітикою.



Рисунок 1.2 – Веб-додаток Salesforce

4. Trello: Trello є простою інструментальною дошкою для організації завдань та проектів. Вона використовує дошки, списки та картки для відстежування прогресу та співпраці в команді.

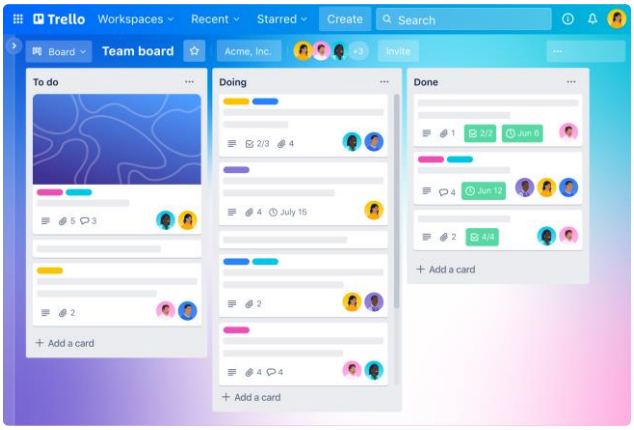


Рисунок 1.3 – Trello

5. Jira: Jira є платформою для управління розробкою програмного забезпечення та проектами. Вона надає інструменти для створення задач, відстежування помилок, управління версіями та спілкування з командою розробників.

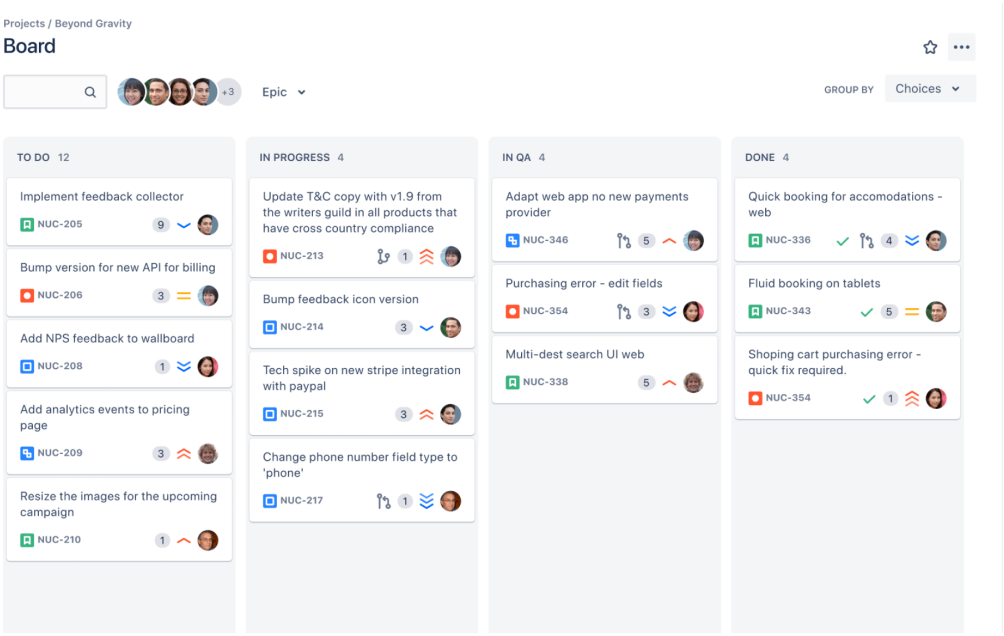


Рисунок 1.4 – Jira

6. Monday.com: Monday.com є платформою для управління проектами, завданнями та співпраці в команді. Вона надає гнучкі інструменти для створення дошок, призначення завдань, відстежування прогресу та спільної роботи.

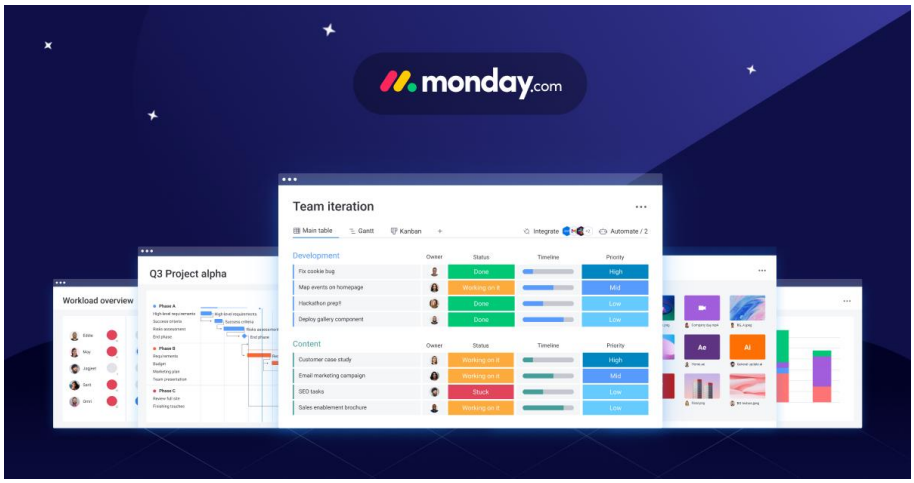


Рисунок 1.5 – Monday.com

Таблиця 1.1

Перелік аналогів SaaS-платформ для управління компаніями та працівниками

Назва	Розробник	Платформа	СУБД	Умови ліцензії
Slack	Slack Technologies	кросплатформена	Slack Database	Різні пакети, включаючи безкоштовний та платні.
Asana	Asana, Inc.	кросплатформена	Postgre SQL	Умови ліцензії доступні онлайн.
Salesforce	Salesforce.com, Inc.	кросплатформена	Salesforce Database	Різні типи ліцензій для користувачів
Trello	Atlassian	кросплатформена	Postgre SQL	Безкоштовний план та платні пакети.
Jira	Atlassian	кросплатформена	Postgre SQL	Різні опції ліцензування для користувачів.
Monday.com	monday.com Ltd	кросплатформена	Monday Database	Різні плани підписки на основі обсягу користувачів.

Порівняємо наші аналоги за не функціональними можливостями

Розглянемо наші аналоги в розрізі SaaS платформ для управлінням персоналу.

Таблиця 1.2

Функціональні можливості

	Slack	Asana	Salesforce	Trello	Jira	Monday.com
Обмін повідомленнями	Так	Так	Ні	Ні	Ні	Ні
Спільна робота над проектами	Ні	Так	Ні	Так	Так	Так
Управління задачами та завданнями	Ні	Так	Ні	Так	Так	Так
Календар та терміни виконання	Ні	Так	Ні	Ні	Так	Так
Керування проектами та процесами	Ні	Так	Так	Так	Так	Так
Відстежування прогресу та стану	Ні	Так	Так	Так	Так	Так

Продовження таблиці 1.2

Відеоконференції	Так	Ні	Ні	Ні	Ні	Ні
Інтеграція з іншими інструментами	Так	Так	Так	Так	Так	Так

Загальний висновок щодо порівняння популярних SaaS-платформ такий:

Кожна з перерахованих платформ має свої унікальні функціональні та нефункціональні можливості, які можуть задовольнити потреби різних користувачів. Slack забезпечує ефективну комунікацію в команді, Asana допомагає в управлінні проектами, Salesforce пропонує потужні інструменти для управління клієнтськими відносинами, Trello спрощує управління завданнями, Jira забезпечує потрібні інструменти для розробки програмного забезпечення. Monday.com також надає широкий спектр можливостей для управління робочими процесами.

Отже, ключовими характеристиками нової SaaS-платформи для управління компаніями та співпраці працівників повинні бути: наявність веб-інтерфейсу для зручного доступу до системи, модуль для збереження інформації про компанії та працівників, модуль для пошуку та відображення даних про проекти, завдання та робочі процеси, який дозволяє фільтрувати їх за певними параметрами, а також модуль для збору інформації про результати співпраці та подальші кроки.

1.3. Вибір архітектури системи управління компаніями

Для реалізації системи управління компаніями ми обрали клієнт-серверну архітектуру. У цій архітектурі клієнтські додатки взаємодіють з сервером, що забезпечує централізоване управління та обробку даних.

В клієнт-серверній архітектурі система розподіляється між клієнтською та серверною частинами. Клієнтська частина відповідає за інтерфейс користувача та збирає введені дані від користувачів. Серверна частина обробляє та зберігає ці дані, а також виконує бізнес-логіку системи. (Рисунок 1.1). Розглянемо їх особливості більш детально:

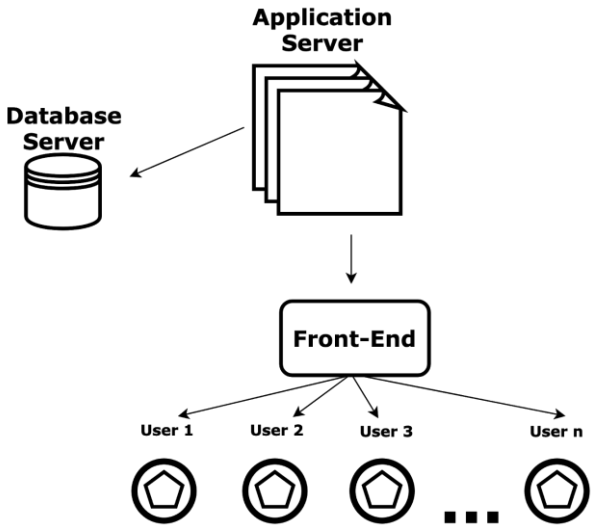


Рисунок 1.6 - Узагальнена архітектура клієнт-серверного додатку

Платформа для управління компаніями використовує клієнт-серверну архітектуру, де клієнтська частина взаємодіє з сервером для обміну даними та виконання бізнес-операцій. При розробці використовується патерн Медіатор, але виключно на серверній стороні.

1. Серверна модель: Серверна модель відповідає за реалізацію бізнес-логіки платформи на сервері. Вона обробляє запити від клієнтської частини і виконує необхідні операції, включаючи доступ до бази даних, обчислення та маніпуляції з даними. Серверна модель може використовувати підхід, який найкраще підходить для мого додатку, такий як ORM для роботи з базою даних.
2. Клієнтське представлення: Клієнтське представлення відповідає за відображення даних та інтерфейсу для користувача. Воно взаємодіє з сервером, надсилаючи запити та отримуючи відповіді. Клієнтське представлення може бути реалізоване у вигляді веб-сторінок. Використовується Flux [10] та NgRx [11] забезпечує одностороннюю потоку даних та керування станом додатку.

   Використання Flux [10] та NgRx [11] допомагає забезпечити ефективне керування станом додатку та односторонній потік даних між клієнтом та сервером в SaaS-платформі. Це сприяє поліпшенню продуктивності,

зручності використання та забезпечує зручний інтерфейс для користувачів.

3. Серверний медіатор: На серверній стороні використовується патерн Медіатор для забезпечення взаємодії між різними компонентами системи. Медіатор виступає як центральний об'єкт, який отримує запити від клієнтського представлення та керує комунікацією між різними модулями на сервері, такими як модуль автентифікації, модуль обробки даних та інші. Це дозволяє зменшити залежності між компонентами та полегшує розширення системи.

   Використання клієнт-серверної архітектури та патерна Медіатор на серверній частині моєї платформи допомагає забезпечити гнучкість та легкість розширення додатка, а також дозволяє зручно враховувати зміни вимог користувачів та обробки інформації.

4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення веб-сайту сиистема управління компаніями та працівниками, рекомендовано використовувати наступні технології:

- мова розмітки гіпертекстових документів: HTML5;
- мова препроцесора для стилів: Sass;
- JavaScript[6] фреймворк: Angular [1];
- серверна частина: Мова програмування C# та фреймворк .NET Core [2];
- база даних: PostgreSQL [8].

Ці технології дозволять вам розробляти веб-сайт сиистема управління компаніями та працівниками з використанням сучасних інструментів. HTML5 забезпечить семантичну розмітку веб-сторінок, Sass дозволить ефективно керувати стилями з підтримкою препроцесінгу, а Angular забезпечить потужний фронтенд для інтерактивної взаємодії з користувачем.

Серверна частина буде побудована на базі мови програмування C# та фреймворка .NET Core, що дозволить створити потужний та масштабований серверний додаток. База даних PostgreSQL забезпечить надійне зберігання та управління даними.

Застосування цих технологій дозволить створити сучасний та функціональний веб-сайт системи управління компаніями та працівниками, забезпечивши ефективне використання ресурсів та гнучкість у розробці.

HTML5 - використовується для створення структури та семантики веб-сторінок.

Sass – це препроцесор CSS, що дозволяє писати стилі з використанням змінних, міксинів та інших функцій, що полегшують роботу зі стилями.

Angular [1] **–** це фреймворк JavaScript для розробки складних веб-додатків з компонентною архітектурою та двостороннім зв'язуванням даних.

C# – використовується для розробки серверної частини. Вона є потужною та об'єктно-орієнтованою мовою з широким набором функціональних можливостей.

.NET Core [2] **–** фреймворк для розробки серверних додатків на мовах, таких як C#. Він надає потужні інструменти для обробки запитів, маршрутизації та роботи з базами даних.

PostgreSQL [8] **–** реляційна база даних з високою надійністю та розширюваністю. Вона забезпечує надійне зберігання та управління даними.

Ці технології допоможуть створити функціональний, зручний у використанні та масштабований веб-сайт сиистема управління компаніями та працівниками.

Таблиця 1.4

Максимально можливий обсяг збережених даних для кожної СКБД

	Розмір БД	Розмір таблиці	Розмір рядка
MySQL	∞	256 ТБ	64 КВ
MS SQL	524258 ТБ	524258 ТБ	∞
PostgreSQL	∞	32 ТБ	1.6 ТБ

За критерієм тригери й збережені процедури всі альтернативи ідентичні. Усі підтримують тригери, процедури й функції.

Таблиця 1.5

Аналіз підтримуваних альтернативами операційних систем

	Windows	Linux	Unix	Android	Symbian
MySQL	+	+	+	+	+
MS SQL	+	-	-	-	-
PostgreSQL	+	+	+	-	+

Порівняємо різні методи захисту даних, які використовуються для запобігання несанкціонованому доступу до конфіденційної інформації в інформаційній системі організації. Подивимося наступну таблицю, де наведено системи забезпечення безпеки даних та їх опис.

Таблиця 1.6

Аналіз систем забезпечення безпеки даних в альтернативах

	Ідентифікація	Захист від brute-force	Шифрування	Сертифікація безпеки
MySQL	+	-	+	-
MS SQL	+	-	+	+
PostgreSQL	+	+	+	+

Після проведеного аналізу систем управління базами даних (СУБД), таких як MySQL, Microsoft SQL Server (MS SQL) та PostgreSQL, встановлено, що для виконання поставлених завдань найбільш підходить PostgreSQL. Актуальна версія PostgreSQL надає широкий функціонал та можливості, включаючи підтримку географічних даних, розширення для складних операцій та механізми захисту даних.

Таблиця 1.7

Аналіз загальних показників СКБД

	MySQL	MS SQL	PostgreSQL
1.	*2.*	*3.*	*4.*
Логічна модель даних	Реляційна	Реляційна	Постреляційна (об'єктно-реляційна)
Фізична модель даних	Сторінкова	Сторінкова	Сторінкова
Типи даних	Всі основні	Всі основні, розширені	Всі основні, довільні, множинні значення
Індекси	Всі основні, повнотекстовий	Всі основні, повнотекстовий	Звичайні, повнотекстовий

Продовження табл. 1.7

1.	*2.*	*3.*	*4.*
Мови маніпулювання	SQL	SQL, QBE, XQuery, багатомірні вирази (для OLAP)	Розширений SQL
Вбудовані мови програмування	ANSI C, ANSI C++	MS Visual Basic, C#	C, C++, Java, Perl, PHP, .Net та ін
Генератор форм, звітів	Ні	Засоби побудови звітів	Ні
Транзакції	Так (в тому числі розподілені)	Так	Так
Тригери, процедури, що зберігаються	Так	Так	Так (підтримка декількох мов програмування)
Платформи	MS Windows, Unix, Linux, MacOS X, Novell NetWare та ін.	Тільки MS Windows	MS Windows, Linux, FreeBSD, Solaris, MacOS
Область застосування	Інформаційні системи масштабу підприємства	Інформаційні системи масштабу підприємства	Інформаційні системи масштабу підприємства
Особливості	Можливість логічного об'єднання БД. Широкий вибір платформ. Реплікація	Масштабування. Кластери. Реплікації. Розширена підтримка XML	Відкриті вихідні тексти. Механізм наслідування. Масштабування. Реплікація.

Для створення програмного комплексу були обрані наступні інструментальні засоби та технології:

- база даних: PostgreSQL [8];
- мова програмування на серверній частині: C#;
- фреймворк: .Net Core [2];
- мова програмування на клієнтській частині: JavaScript [6];
- фреймворк на клієнтській частині: Angular [1];
- мова процесора для стилів: Sass.

Для розробки були використані наступні програмні продукти та інтегровані середовища розробки: графічний редактор Figma, середовище розробки Visual Studio Code та Visual Studio 2022.

Висновки до розділу 1

Проведений аналіз предметної області дозволив визначити основні напрямки реалізації Saas платформи для керування компаніями та працівниками. Необхідно реалізувати збереження даних про компанії, поректи, користувачів, матеріали, постачальників, фаз будівництва, фільтрація, а також сповіщення про зміни на серверній стороні в реальному часі.

Основними вимогами до таких систем є централізоване управління, ефективність використання, зручний інтерфейс, масштабованість та безпека даних. Розробка SaaS-платформи для управління компаніями та працівниками має на меті забезпечити ці вимоги та полегшити рутинні адміністративні завдання, покращуючи комунікацію та співпрацю в організації.

В роботі обґрунтовано вибір архітектури для реалізації системи:

– база даних: PostgreSQL;

– мова програмування на серверній частині: C#;

– фреймворк: .Net Core;

– мова програмування на клієнтській частині: JavaScript;

– фреймворк на клієнтській частині: Angular;

– мова процесора для стилів: Sass.

Буде використано такі програмні продукти та інтегровані середовища розробки: графічний редактор Figma; середа розробки Visual Studio Code та Visual Studio 2022.

РОЗДІЛ 2. ПРОЕКТУВАННЯ SAAS ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ КОМПАНІЯМИ

2.1. Визначення варіантів використання та об’єктно-орієнтованої структури системи

Платформа для управління компаніями та працівниками має наступні основні цілі: централізоване управління компаніями і працівниками, оптимізацію робочих процесів і забезпечення доступності та зручності для користувачів. Ця платформа забезпечує єдине місце для зберігання інформації про компанії та їх працівників, спрощує управління та підвищує ефективність роботи. Вона також оптимізує робочі процеси, автоматизує завдання, полегшує спільну роботу та комунікацію, і забезпечує моніторинг та аналіз продуктивності. Крім того, платформа має бути зручною та доступною, з інтуїтивним інтерфейсом, який можна використовувати з різних пристроїв.

Внутрішні користувачі

Внутрішніми користувачами, а саме адміністраторами платформи, надаються ряд функцій, що включають:

1. Можливість перегляду, додавання, пошуку та сортування компаній в системі. Адміністратори можуть переглядати існуючі компанії, додавати нові компанії, здійснювати пошук по списку компаній та сортувати їх за певними критеріями.
2. Можливість перегляду та редагування інформації про компанію. Адміністратори можуть переглядати загальну інформацію про компанію, редагувати опис компанії та додавати нових працівників до компанії.
3. Отримання сповіщень про зміни в системі. Адміністратори отримують сповіщення та повідомлення про будь-які зміни, які відбуваються в платформі, що дозволяє їм бути завжди в курсі подій.

У додаток до цього, внутрішні користувачі-адміністратори платформи мають додаткові можливості, такі як редагування бази даних та дизайну інтерфейсу, що дозволяє їм налаштовувати та модифікувати функціонал платформи згідно з власними потребами і вимогами.

Враховуючи ці можливості, адміністратори платформи мають повний контроль над компаніями, працівниками та функціоналом системи, що дозволяє їм зручно та ефективно управляти ресурсами та забезпечувати ефективну роботу компаній та їх працівників.

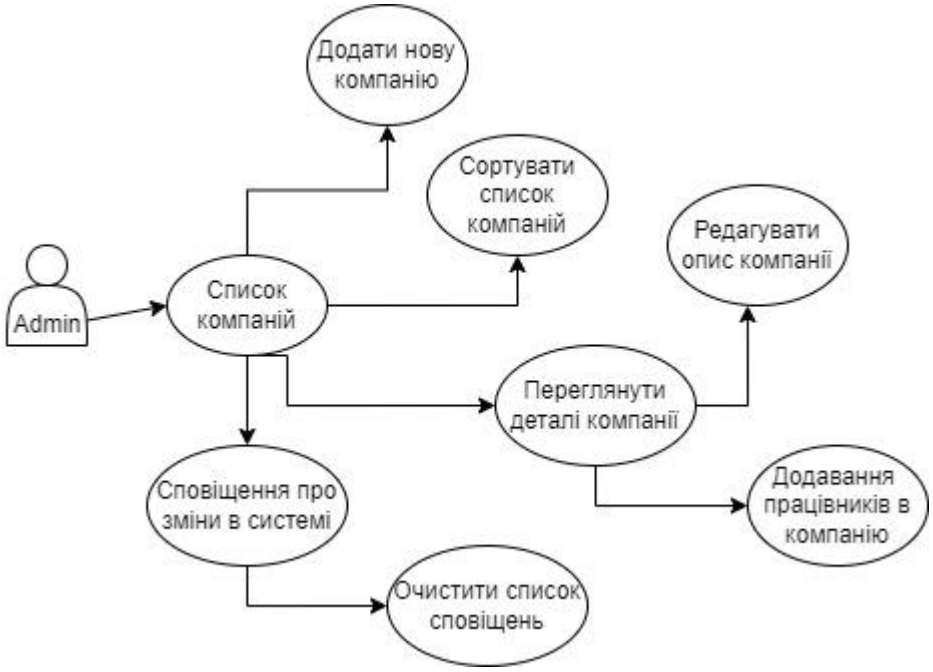


Рисунок 2.1 - Функціональні можливості адміністратора.

Внутрішніми користувачами, а саме проджект менеджерам платформи, надаються ряд функцій, що включають:

1. Можливість перегляду, додавання, пошуку та сортування компаній в системі. Проектні менеджери можуть переглядати доступні проекти в компанії, додавати нові проекти, здійснювати пошук по списку проектів, переглядати списки проектів по їх стадії та сортувати їх за певними критеріями.
2. Проектні менеджери мають розширені можливості щодо керування проектами. Вони можуть переглядати інформацію про проект, редагувати його дані, включаючи назву та опис, додавати нові будівлі

до проекту та завантажувати фотографії проекту. Також вони можуть додавати нових членів до команди проекту та встановлювати таймлайни для різних етапів проекту. В разі потреби, проектні менеджери можуть додавати ресурси та матеріали до проекту, а також відстежувати прогрес готовності будівлі через таймлайн. Крім того, вони мають можливість змінити назву будівлі або видалити її з проекту.

3. У системі також доступна вкладка "Таймлайни", яка призначена для відстеження прогресу завершення проекту. Ця функція дозволяє користувачам переглядати хронологічну послідовність подій і етапів проекту, щоб бути в курсі всіх змін і роботи, які відбуваються протягом проектного процесу.
4. У системі присутня сторінка "Ресурси", яка складається з двох вкладок: "Матеріали" і "Сервіси". На вкладці "Матеріали" можна створювати списки доступних матеріалів, включаючи опис постачальника, кінцеву ціну і доступну кількість для розподілу на проектні будівлі. У вкладці "Сервіси" доступні списки постачальників з їх контактними даними. Ця функціональність дозволяє зручно управляти ресурсами, забезпечуючи доступ до необхідних матеріалів і контактів постачальників.
5. У системі існує сторінка "Сповіщення", яка дозволяє відстежувати зміни на проектах, включаючи можливість редагування, видалення та інші дії. Ця функціональність надає зручний спосіб контролювати і відстежувати будь-які зміни, які відбуваються на проектах, дозволяючи ефективно керувати ними та приймати відповідні заходи.
6. На сторінці "Профіль" ми можемо відстежувати загальну інформацію про компанію, включаючи стадії проектів в компанії та їх статус, такі як "закінчено", "в процесі" і т.д. Крім того, ми можемо редагувати логотип компанії та контактні дані, щоб забезпечити актуальну та точну інформацію про компанію. Ця функція дозволяє нам зберігати і

керувати важливими даними компанії, що допомагає підтримувати актуальність та належний розподіл інформації.

Можливості, які доступні проектним менеджерам, значно спрощують процес управління проектами. Вони можуть з легкістю переглядати і редагувати інформацію про проекти, додавати будівлі, фотографії та членів команди проекту. Також вони мають доступ до таймлайнів, які допомагають відстежувати прогрес завершення проекту. Крім того, можливість додавання ресурсів та матеріалів до проекту дозволяє ефективно керувати його ресурсами. Всі ці функціональні можливості допомагають проектним менеджерам забезпечувати успішне виконання проектів та забезпечувати співпрацю всіх учасників команди.

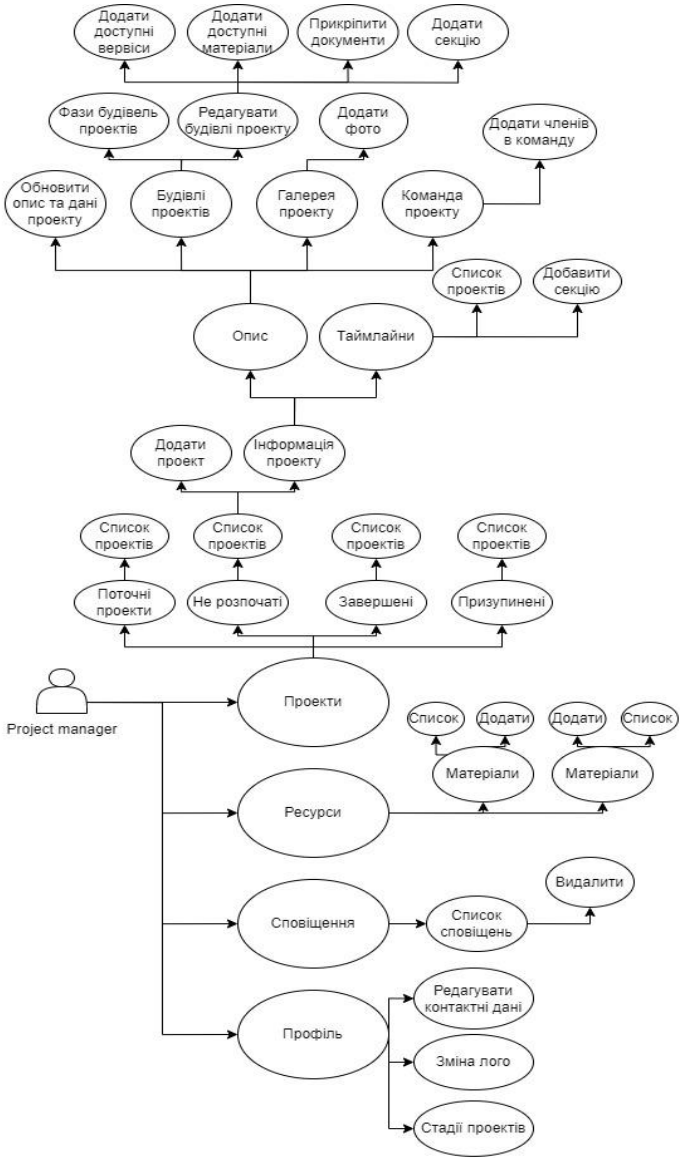


Рисунок 2.2 - Функціональні можливості проектного менеджера

Функціональні вимоги:

Система управління проектами надає проектним менеджерам такі можливості:

1. Перегляд, додавання, пошук та сортування компаній в системі.
2. Перегляд та редагування інформації про проекти, включаючи будівлі, фотографії та таймлайн.
3. Управління ресурсами, включаючи матеріали та постачальників послуг.
4. Отримання сповіщень про зміни на проектах.
5. Редагування профілю зі зміною логотипу та контактних даних.

Нефункціональні вимоги:

Навчання роботи з системою:

- Час навчання: Для звичайних користувачів - 1-2 години, для досвідчених - 1 година.
- Швидкість відповіді: Максимум 1 секунда на звичайні запити, 3 секунди - на складніші запити.
- Інтуїтивний інтерфейс: Легкий у використанні, не потребує додаткової підготовки.
- Надійність: Стабільна та надійна робота системи без збоїв.
- Доступність: Час обслуговування не перевищує 10% загального часу роботи.
- Неперервна робота: Безперебійна робота на протязі 20 робочих днів.
- Норма помилок: Максимум 1 помилка на 1000 запитів користувача.
- Продуктивність: Продуктивність системи є важливою нефункціональною вимогою, яка включає швидкість відповіді, ефективну обробку даних, масштабованість, економне використання ресурсів, оптимізацію запитів, паралельну обробку, надійність та ефективне керування ресурсами. Система повинна забезпечувати швидкість відповіді не більше 1-3 секунд, оптимальне використання ресурсів та надійну роботу. Крім того, важливо, щоб система була

масштабованою, щоб впоратися з великим обсягом даних та багатьма користувачами.

Можливість експлуатації

Окрім цього, система повинна мати інтуїтивно зрозумілий інтерфейс, що не вимагає додаткової підготовки користувача. Час навчання з інструментами роботи з системою для звичайних користувачів складає 1-2 години, а для досвідчених - 1 година.

Для реалізації проекту були обрані такі технології: Single Page Application (SPA), патерн CQRS та мови програмування .NET Core та Angular.

Аналіз функціональних вимог дозволив виділити наступні сутності, які забезпечать реалізацію програмного комплексу системи управління проектами. Діаграма класів, представлена на рисунку 2.2., відображає взаємозв'язки між цими сутностями. В розділі 2.2. буде наведена модель бази даних системи, що відповідає цим сутностям та організована на рівні моделі. В розділі 3 буде представлена структура представлень системи, яка відображається користувачам.

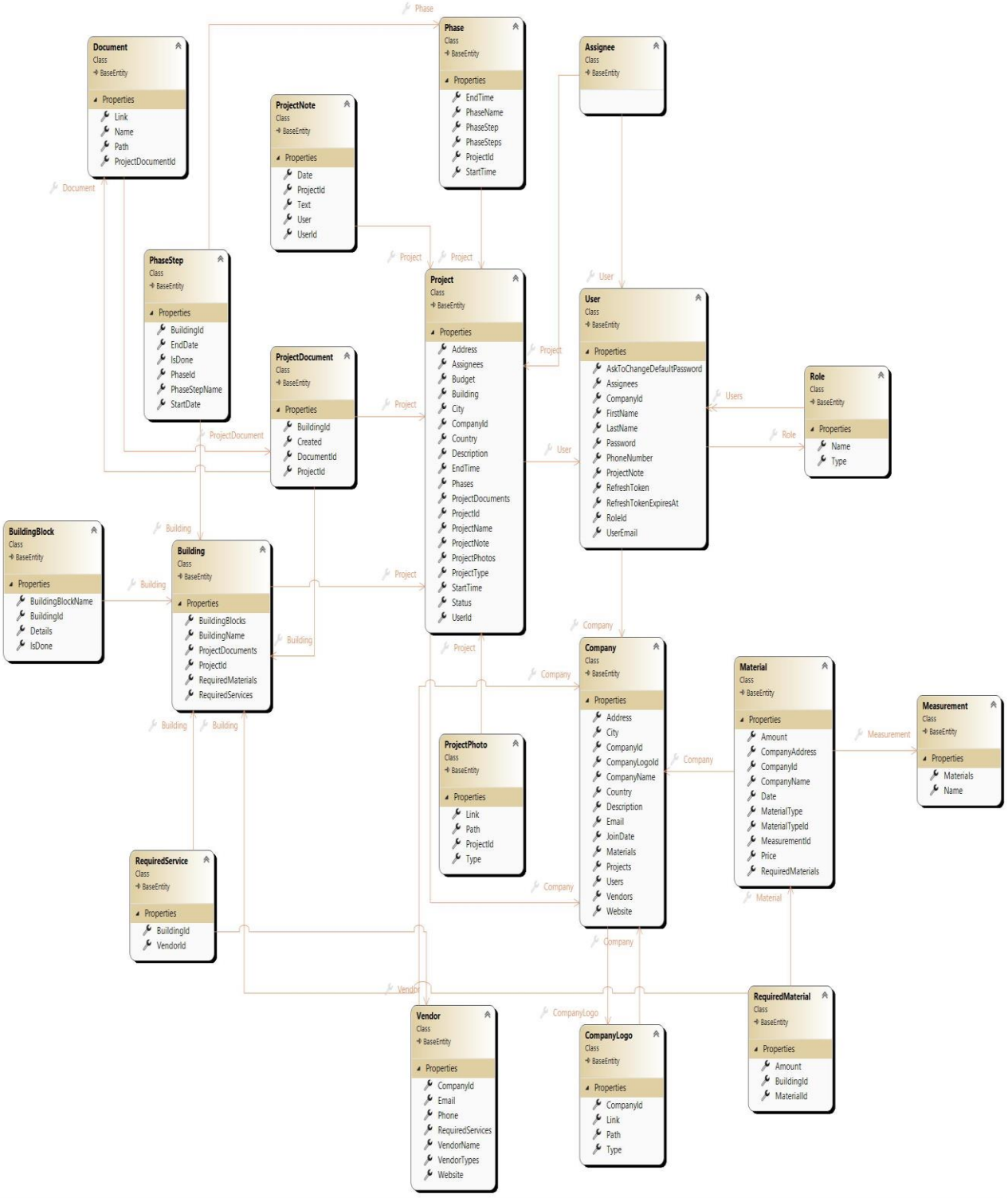


Рисунок 2.3 - Діаграма класів

На даному рисунку можна виділити наступні класи:

– Project – клас роботи з проектами які належать компанії.

Методи :

1. GetProjectWithParams([FromQuery]ProjectPaginationQueryDTO queryModel) — цей метод дозволяє отримати список проектів з параметрами для пагінації та фільтрації.
2. GetProjectDetailed(long id) — використовується для отримання детальної інформації про проект за його унікальним ідентифікатором.
3. PutProject([FromBody] ProjectDetailedRequestDto project) — даний метод дозволяє змінювати деталі проекту.
4. CreateProjects([FromBody] CreateProjectDTO project) — використовується для створення нового проекту.
5. GetProjectsByCompanyId([FromRoute] long id) — цей метод дозволяє отримати список всіх проектів, що належать компанії за її унікальним ідентифікатором.
6. GetProjectPhotos(long id) — використовується для отримання всього переліку фотографій, пов'язаних з проектом за його унікальним ідентифікатором.
7. PutProjectPhotos — метод дозволяє додавати та оновлювати фотографії проекту.
8. DeleteProjectsPhotoById([FromForm] IFormCollection data, long id) — використовується для видалення фотографії проекту за її унікальним ідентифікатором.
9. ChangeStatus([FromBody] ChangeStatusCommand command) — даний метод дозволяє змінювати поточний статус проекту.
10. GetTeam([FromQuery] GetProjectTeamQuery query) — використовується для отримання списку команди, що належить до проекту.
11. SetTeam([FromBody] SetProjectTeamCommand command) — цей метод дозволяє додавати нових членів у команду проекту.

– Company – клас роботи з компінями.

Методи:

1. GetCompanyWithParams([FromQuery(Name="filter")]string? filter, [FromQuery(Name = "sort")] string sort) — цей метод дозволяє отримати список компаній з параметрами для фільтрації та пагінації.
2. GetCompanyProjects(long id, [FromQuery] PaginationQueryModelDTO dto) — використовується для отримання списку проектів, що належать компанії, з можливістю фільтрації та пагінації.
3. CreateCompany(NewCompanyDto model) — даний метод дозволяє створити нову компанію.
4. PutCompany([FromBody]CompanyDetailedRequestDto data) — використовується для оновлення інформації про компанію.
5. GetCompanyById(long id) — метод дозволяє отримати компанію за її унікальним ідентифікатором.
6. GetCompanyByUserId(long userId) — використовується для отримання списку компаній, які пов'язані з користувачем за його унікальним ідентифікатором.
7. UpdateCompanyProfile(long id, [FromBody] UpdateCompanyProfileCommand command) — цей метод дозволяє оновити інформацію профілю компанії.
8. GetCompanyLogo(long id) — метод дозволяє отримати логотип компанії за її унікальним ідентифікатором.
9. DeleteCompanyLogo(long id) — використовується для видалення логотипу компанії за її унікальним ідентифікатором.
10. PutCompanyLogo([FromForm]IFormFile data, long id) — даний метод дозволяє оновити фотографію логотипу компанії.
11. GetNewCompanyGeneratedId() — метод генерує унікальний ідентифікатор для нової компанії.

– Building – клас роботи будівлями проекту.

Методи :

1. GetBuildingByProjectId (long projectId) – отримати список будівель, які належать до проекту з заданим ідентифікатором проекту.

2. AddBuilding([FromBody] AddBuildingCommand command) – додати нову будівлю до проекту.
3. UpdateBuilding(long id, [FromBody] UpdateBuildingCommand command) – Оновити дані будівлі з заданим ідентифікатором.
4. DeleteBuilding(long id) – видалити будівлю з заданим ідентифікатором з проекту.

– Resource – клас роботи із ресурсами компанії.

Методи :

1. AddService([FromBody] AddServiceCommand command) – додати новий сервіс.
2. DeleteService([FromRoute] long id) – видалити сервіс з таблиці.
3. GetServices(long buildingId, [FromQuery(Name = "filter")] string? filter) – отримати відфільтрований список сервісів, що належать до певної будівлі.
4. GetSelectedServices(long buildingId) вивести вибрані сервіси.
5. GetListOfMaterials([FromQuery] MaterialPaginationQueryDto queryModel)) – отримати список матеріалів компанії..
6. GetListOfMaterialType() – отримати список типів матеріалів.
7. GetListOfMeasurement() – отримати список одиниць виміру.
8. CreateMatial(CreateMaterialCommandDto model) – додати новий матеріал до списку.
9. DeleteMaterialById(long id) – видалити матеріал за ідентифікатором.
10. PutMaterial([FromBody] PutMaterialCommand command) – оновити інформацію про матеріал.
11. GetAvailableMaterialsForProject([FromQuery] GetAvailableMaterialsForProjectQuery query) – отримати список матеріалів, доступних для проектуотримує список матеріалів.доступних для проекту.

–Auth – клас роботи із авторизацією.

Методи:

1. Login ([FromBody] LoginCommand command) – авторизуватися в системі.
2. Logout([FromBody] LogoutCommand command) – вийти з системи
3. ChangeDefaultPassword([FromBody] ChangeDefaultPasswordCommand command) – змінити дефолтний пароль..
4. KeepDefaultPassword([FromBody] KeepDefaultPasswordCommand command) – зберегти дефолтний пароль.

Таким чином, дана система реалізує функціонал підбору вакансій, пошуку резюме, збереження даних про процес працевлаштування, що забезпечується методами контролерів Auth, Resource, Project, Building, Company.

2.2. Розробка бази даних системи

Реалізована база даних PostgreSQL складається з 32 таблиць, які містять у собі всі дані для роботи програми. База даних має назву CtorDB.

Таблиця 2.1

Інформація про таблиці бази даних CtorDB.

Назва таблиці	Призначення
Assignee	Проміжна таблиця користувачем та проектами
Building	Таблиця будівель
BuildingBlock	Таблиця блоків прогресу будівництва будівель
BuildingVendor	Проміжна таблиця зв’язку між будівлями та постачальниками
Company	Таблиця компаній
CompanyLogo	Таблиця логотипів компаній
Document	Таблиця документів
Material	Таблиця матеріалів
MaterialType	Таблиця типів матеріалів
Measurement	Таблиця вимірювальних одиниць
Notifications	Таблиця сповіщень
Phase	Таблиця фаз будівництва
PhaseStep	Таблиця етапів фаз будівництва
Project	Таблиця проектів
ProjectDocument	Проміжна таблиця зв’язку між проектами та документами

Продовження таблиці 2.1

ProjectNote	Проміжна таблиця зв’язку між проектами та нотатками
ProjectPhoto	Проміжна таблиця зв’язку між проектами та фотографіями
RequiredMaterials	Таблиця необхідних матеріалів
RequiredService	Таблиця необхідних сервісів
Role	Таблиця ролей
User	Таблиця користувачів
Vendor	Таблиця постачальників
VendorType	Таблиця типів постачальників
VendorVendorType	Проміжна таблиця зв’язку між постачальниками та типами постачальників

Структура бази даних наведена на Рисунок 2.4.

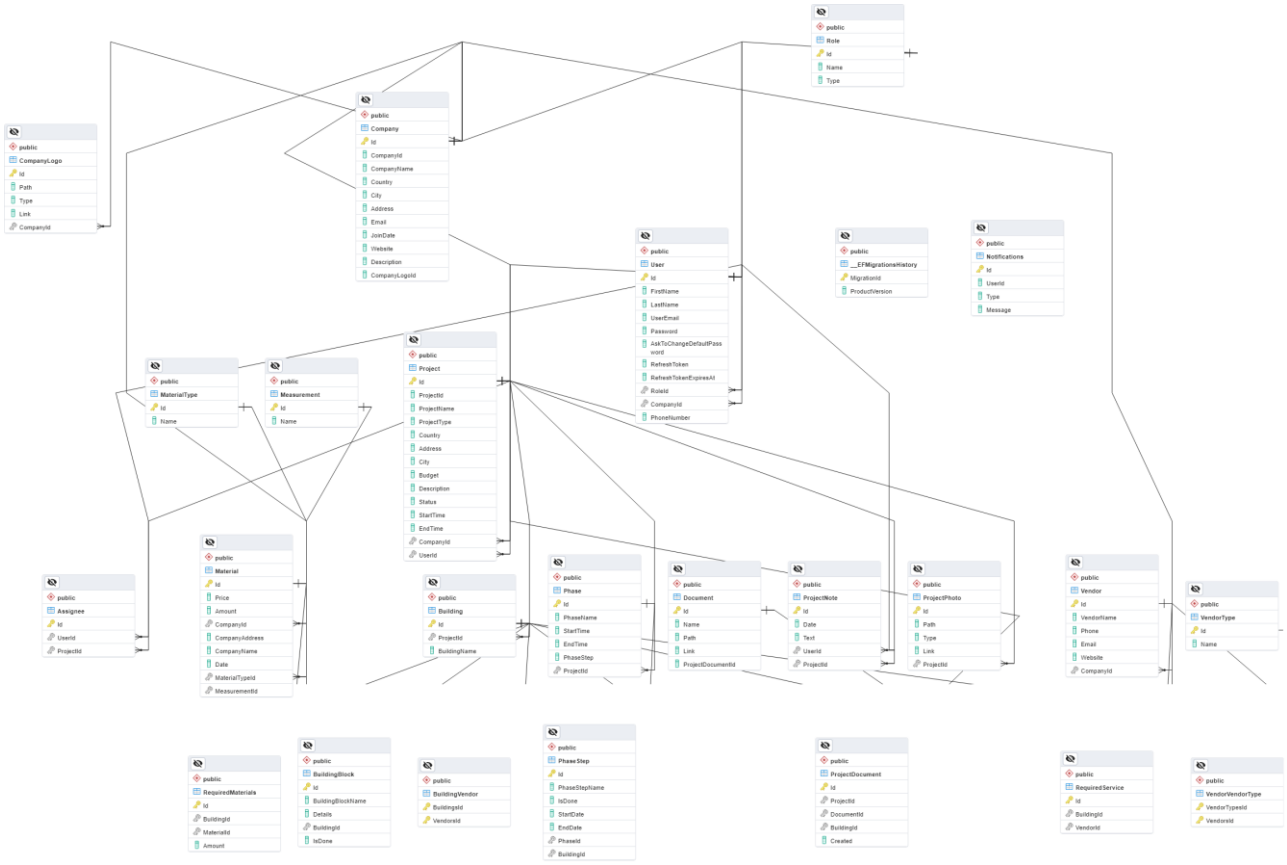


Рисунок 2.4 - Структурна схема БД

Далі наведемо опис основних таблиць бази даних.

Таблиця "Assignee" призначена для встановлення відповідності між проектами та проектними менеджерами. Ця таблиця містить зв'язок між користувачами та проектами, до яких вони мають доступ, з використанням відношення "багато до багатьох". Нижче наведена структура таблиці.

Таблиця 2.2

Опис полів таблиці «Assignee»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
UserId	bigint	-	User.Id	Номер користувача
ProjectId	bigint	-	Project.Id	Номер проекту

Таблиця «Building» призначена для збереження даних про будівлі проекту. В дану таблицю вводяться назва будівлі. Структура таблиці наведена нижче:

Таблиця 2.3

Опис полів таблиці «Building»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
ProjectId	bigint	-	Project.Id	Номер проекту
BuildingName	text	-	-	Назва будівлі

Таблиця «BuildingBlock» зберігає блоки прогресу будівництва будівель. Структура таблиці наведена нижче:

Таблиця 2.4

Опис полів таблиці «BuildingBlock»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
BuildingBlockName	text	-	-	Назва блоку будівлі
Details	text	-	-	Деталі
BuildingId	bigint	-	-	Номер будівлі
IsDone	boolean	-	-	Позначка виконання

Таблиця "BuildingVendor" використовується для встановлення зв'язку між таблицями "Будівлі" та "Постачальники" з використанням відношення "багато до багатьох". Структура таблиці наведена нижче:

Таблиця 2.5

Опис полів таблиці «BuildingVendor»

Назва	Тип даних	PK	FK	Опис поля
BuildingsId	bigint	+	Building.Id	Номер будівлі
VendorsId	bigint	+	Vendor.Id	Номер вендора

Таблиця «Company» призначена для зберігання даних компанії. В дану таблицю вводяться дані про компанії. Структура таблиці наведена нижче:

Таблиця 2.6

Опис полів таблиці «Company »

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
CompanyId	bigint	-	-	Номер компанії
CompanyName	text	-	-	Назва компанії
Country	text	-	-	Країна
City	text	-	-	Місто
Address	text	-	-	Адреса
Email	text	-	-	Email
JoinDate	timestamp	-	-	Дата приєднання
Website	text	-	-	Веб-сайт
Description	text	-	-	Опис
CompanyLogoId	bigint	-	CompanyLogo.Id	Номер логотипу компанії

Таблиця «CompanyLogo» призначена для збереження логотипу компанії. В дану таблицю вводяться посилання на логотип та деталі. Структура таблиці наведена нижче:

Таблиця 2.7

Опис полів таблиці «CompanyLogo»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Path	text	-	-	Шлях до логотипу
Type	integer	-	-	Тип логотипу
Link	text	-	-	Посилання
CompanyId	bigint	-	Company.Id	Номер компанії

Таблиця «Document» призначена для збереження документів. В дану таблицю вводяться дані про документ та шлях. Структура таблиці наведена нижче:

Таблиця 2.8

Опис полів таблиці «Document»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Name	text	-	-	Назва
Path	text	-	-	Шлях до документа
Link	text	-	-	Посилання
ProjectDocumentId	bigint	-	ProjectDocument.Id	Номер документа проекту

Таблиця «Material» призначена для збереження інформації про матеріали. Структура таблиці наведена нижче:

Таблиця 2.9

Опис полів таблиці «Material»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Price	numeric	-	-	Ціна матеріалу
Amount	integer	-	-	Кількість матеріалу
CompanyId	bigint	-	Company.Id	Номер компанії
CompanyAddress	text	-	-	Адреса компанії
CompanyName	text	-	-	Назва компанії
Date	timestamp	-	-	Дата
MaterialTypeId	bigint	-	MaterialType.Id	Номер типу матеріалу
MeasurementId	bigint	-	Measurement.Id	Номер одиниці виміру

Таблиця «MaterialType» призначена для збереження типів матеріалу. Структура таблиці наведена нижче:

Таблиця 2.10

Опис полів таблиці «MaterialType»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Name	text	-	-	Назва типу

Таблиця «Measurement» призначена для збереження одиниць виміру. Структура таблиці наведена нижче:

Таблиця 2.11

Опис полів таблиці «Measurement»

Назва	Тип даних	PK	FK	Опис поля
id	bigint	+	-	Номер
Name	text	-	-	Назва одиниці виміру

Таблиця «Notifications» призначена для збереження оповіщень. Структура таблиці наведена нижче:

Таблиця 2.12

Опис полів таблиці «Notifications»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
UserId	bigint	-	User.Id	Номер користувача
Type	text	-	-	Тип повідомлення
Message	text	-	-	Текст повідомлення

Таблиця «Phase» призначена для збереження фази будівництва проектів. Структура таблиці наведена нижче:

Таблиця 2.13

Опис полів таблиці «Phase»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
PhaseName	text	-	-	Назва фази
StartTime	timestamp	-	-	Початковий час фази
EndTime	timestamp	-	-	Кінцевий час фази
PhaseStep	integer	-	-	Крок фази
ProjectId	bigint	-	Project.Id	Номер проекту

Таблиця «PhaseStep» призначена для збереження кроків фази з більш детальним описом. Структура таблиці наведена нижче:

Таблиця 2.14

Опис полів таблиці «PhaseStep»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
PhaseStepName	text	-	-	Назва кроку
IsDone	boolean	-	-	Прапорець виконання кроку
StartDate	timestamp	-	-	Початкова дата кроку
EndDate	timestamp	-	-	Кінцева дата кроку
PhaseId	bigint	-	Phase.Id	Номер фази
BuildingId	bigint	-	-	Номер будівлі

Таблиця «Project» призначена для збереження даних про проекти. Структура таблиці наведена нижче:

Таблиця 2.15

Опис полів таблиці «Project»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
ProjectId	text	-	-	Номер проекту
ProjectName	text	-	-	Назва проекту
ProjectType	text	-	-	Тип проекту
Country	text	-	-	Країна
Address	text	-	-	Адреса
City	text	-	-	Місто
Budget	numeric	-	-	Бюджет
Description	text	-	-	Опис проекту
Status	integer	-	-	Статус проекту
StartTime	timestamp	-	-	Початковий час проекту
EndTime	timestamp	-	-	Кінцевий час проекту
CompanyId	bigint	-	Company.Id	Номер компанії
UserId	bigint	-	User.Id	Номер компанії

Таблиця «ProjectDocument» призначена для збереження проектних документів. Структура таблиці наведена нижче:

Таблиця 2.16

Опис полів таблиці «ProjectDocument»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
ProjectId	bigint	-	Project.Id	Номер проекту
DocumentId	bigint	-	Document.Id	Номер документу
BuildingId	bigint	-	-	Номер будівлі
Created	timestamp	-	-	Дата створення

Таблиця «ProjectNote» призначена для збереження нотатків проекта. Структура таблиці наведена нижче:

Таблиця 2.17

Опис полів таблиці «ProjectNote»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Date	timestamp	-	-	Дата
Text	text	-	-	Текст нотатки
UserId	bigint	-	User.Id	Номер користувача
ProjectId	bigint	-	Project.Id	Номер проекту

Таблиця «ProjectPhoto» призначена для збереження фото проекту. Структура таблиці наведена нижче:

Таблиця 2.18

Опис полів таблиці «ProjectPhoto»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Path	timestamp	-	-	Шлях до фотографії
Type	text	-	-	Тип фотографії
Link	text	-	-	Посилання на фотографію
ProjectId	bigint	-	Project.Id	Номер проекту

Таблиця «RequiredMaterials» призначена для збереження списку необхідних матеріалів. Структура таблиці наведена нижче:

Таблиця 2.19

Опис полів таблиці «RequiredMaterials»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
MaterialId	bigint	-	Material.Id	Номер матеріалу
ProjectId	bigint	-	Project.Id	Номер проекту
Quantity	numeric	-	-	Кількість матеріалу
BuildingId	bigint	-	-	Номер будівлі

Таблиця «RequiredService» призначена для збереження списку необхідних сервісів. Структура таблиці наведена нижче:

Таблиця 2.20

Опис полів таблиці «RequiredService»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
BuildingId	bigint	-	-	Номер будівлі
VendorId	bigint	-	-	Номер виробника

Таблиця «User» призначена для збереження даних користувача. Структура таблиці наведена нижче:

Таблиця 2.21

Опис полів таблиці «User»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
FirstName	text	-	-	Ім'я
LastName	text	-	-	Прізвище
Email	text	-	-	Електронна пошта
Password	text	-	-	Пароль
Role	text	-	-	Роль

Продовження таблиці 2.21

CompanyId	bigint	-	Company.Id	Номер компанії

Таблиця «Role» призначена для збереження ролей користувачів. Структура таблиці наведена нижче:

Таблиця 2.22

Опис полів таблиці «Role»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Name	text	-	-	Назва
Type	integer	-	-	Тип

Таблиця «Vendor» призначена для збереження даних про виробників. Структура таблиці наведена нижче:

Таблиця 2.23

Опис полів таблиці «Vendor»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
VendorName	text	-	-	Назва виробника
Phone	text	-	-	Телефон
Email	text	-	-	Електронна пошта
Website	text	-	-	Веб-сайт
CompanyId	bigint	-	-	Номер компанії

Таблиця «Role» призначена для збереження ролей користувачів. Структура таблиці наведена нижче:

Таблиця 2.24

Опис полів таблиці «Role»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Name	text	-	-	Назва
Type	integer	-	-	Тип

Таблиця «VendorType» призначена для збереження типу виробника. Структура таблиці наведена нижче:

Таблиця 2.25

Опис полів таблиці «VendorType»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Name	text	-	-	Назва
Phone	text	-	-	Телефон

Таблиця «VendorVendorType» призначена для створення зв'язку між таблицею виробників та типом виробника. Структура таблиці наведена нижче:

Таблиця 2.26

Опис полів таблиці «VendorVendorType»

Назва	Тип даних	PK	FK	Опис поля
VendorTypesId	bigint	-	-	Номер типу виробника
VendorsId	bigint	-	-	Номер виробника

Отже реалізована база даних у відповідності до вимог реляційної моделі надає зручний та ефективний спосіб збереження та колективного доступу до інформації кадрової агенції. З використанням таблиць, таких як User, Company, Project, Material, Service та Notification, можна організувати та управляти даними, забезпечуючи зручний інструментарій для роботи з інформацією про користувачів, компанії, проекти, матеріали, послуги та повідомлення. База даних складається з 24 таблиць.

2.3. Проектування та реалізація алгоритмів роботи системи

Клієнтський додаток системи буде реалізований як односторінкова програма (Single Page Application, SPA). При переході на сторінку додатку або відкритті мобільного додатку, сервер надсилає актуальну версію клієнтської частини користувачеві. Після цього, залежно від дій користувача, клієнт надсилає запити на сервер. Після отримання відповіді від серверу, клієнт змінює вигляд сторінки відповідно.

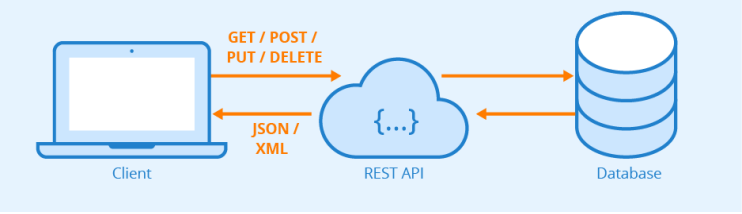


Рисунок 2.5 - Загальний алгоритм роботи додатку

2.4 Реалізація додатку системи управління компаніями та працівниками.

```
[AllowAnonymous]
[HttpPost("login")]
0 references
public async Task<ActionResult<LoginDto>> Login([FromBody] LoginCommand command)
{
    return await Mediator.Send(command);
}
```

Рисунок 2.6 - Метод Login в контролері AuthController

```
4 references
public record LoginCommand(string Email, string Password) : IRequest<LoginDto>;

1 reference
public class LoginCommandHandler : IRequestHandler<LoginCommand, LoginDto>
{
    private readonly IUserManager _userManager;

    0 references
    public LoginCommandHandler(IUserManager userManager)
    {
        _userManager = userManager;
    }

    0 references
    public async Task<LoginDto> Handle(LoginCommand request, CancellationToken ct)
    {
        var result = await _userManager.LoginAsync(request.Email, request.Password, ct);

        return new LoginDto(
            new LoginUserDto(result.UserId, result.Email),
            askToChangePassword: result.AskToChangePassword,
            new LoginTokenDto(result.AccessToken.Value, result.AccessToken.ExpiresAt),
            new LoginTokenDto(result.RefreshToken.Value, result.RefreshToken.ExpiresAt));
    }
}
```

Рисунок 2.7 - Клас LoginCommand

```
2 references
public async Task<LoginResult> LoginAsync(
    string email,
    string password,
    CancellationToken ct)
{
    var user = await _userRepository.GetByEmailAsync(email, ct);

    EnsurePasswordIsCorrect(user, password);

    var accessToken = _tokenProvider.GenerateAccessToken(user.Id, user.Role.Type, user.CompanyId);
    var refreshToken = _tokenProvider.GenerateRefreshToken(user.Id);

    user.RefreshToken = refreshToken.Value;
    user.RefreshTokenExpiresAt = refreshToken.ExpiresAt;

    await _applicationDbContext.SaveChangesAsync(ct);

    return new LoginResult(user.Id, user.UserEmail, user.AskToChangeDefaultPassword, accessToken, refreshToken);
}
```

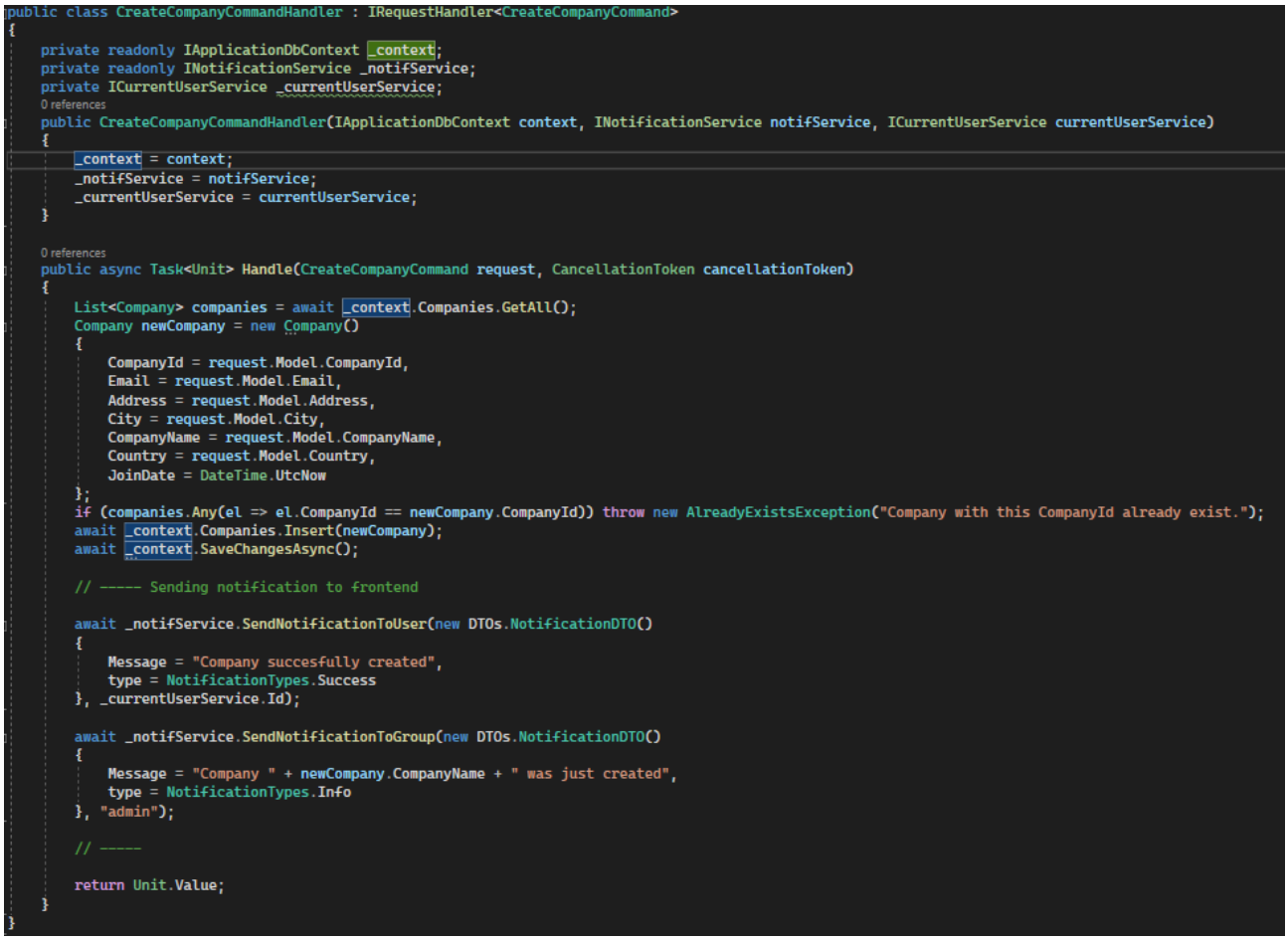
Рисунок 2.8 - Метод LoginAsync в UserManager

Авторизація реалізована з використанням JWT (JSON Web Token), що забезпечує безпечний спосіб передачі даних між клієнтом і сервером. У процесі авторизації, користувач надає свої облікові дані (електронна пошта та пароль), які перевіряються на коректність. Після успішної перевірки, генеруються токени доступу (Access Token) та оновлення (Refresh Token), які включають необхідну інформацію про користувача та час їхньої дії. Токен доступу використовується для авторизації запитів користувача до захищених ресурсів, тоді як токен оновлення використовується для оновлення та отримання нового токену доступу без повторної аутентифікації. JWT забезпечує безпеку шляхом підпису та шифрування токенів, що дозволяє перевіряти їхню цілісність та автентичність. Такий підхід до авторизації забезпечує захист конфіденційності даних та зменшує ризик несанкціонованого доступу до ресурсів системи.



```
[HttpPost]
[Route("create")]
0 references
public async Task<IActionResult> CreateCompany(NewCompanyDto model)
{
    if (!ModelState.IsValid) return BadRequest("Model is not valid");
    await Mediator.Send(new CreateCompanyCommand(model));
    return StatusCode(201);
}
```

Рисунок 2.9 - Метод для створення копанії в котроллері CompanyController



```
public class CreateCompanyCommandHandler : IRequestHandler<CreateCompanyCommand>
{
    private readonly IApplicationDbContext _context;
    private readonly INotificationService _notifService;
    private ICurrentUserService _currentUserService;
    0 references
    public CreateCompanyCommandHandler(IApplicationDbContext context, INotificationService notifService, ICurrentUserService currentUserService)
    {
        _context = context;
        _notifService = notifService;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<Unit> Handle(CreateCompanyCommand request, CancellationToken cancellationToken)
    {
        List<Company> companies = await _context.Companies.GetAll();
        Company newCompany = new Company()
        {
            CompanyId = request.Model.CompanyId,
            Email = request.Model.Email,
            Address = request.Model.Address,
            City = request.Model.City,
            CompanyName = request.Model.CompanyName,
            Country = request.Model.Country,
            JoinDate = DateTime.UtcNow
        };
        if (companies.Any(el => el.CompanyId == newCompany.CompanyId)) throw new AlreadyExistsException("Company with this CompanyId already exist.");
        await _context.Companies.Insert(newCompany);
        await _context.SaveChangesAsync();

        // ----- Sending notification to frontend

        await _notifService.SendNotificationToUser(new DTOs.NotificationDTO()
        {
            Message = "Company succesfully created",
            type = NotificationTypes.Success
        }, _currentUserService.Id);

        await _notifService.SendNotificationToGroup(new DTOs.NotificationDTO()
        {
            Message = "Company " + newCompany.CompanyName + " was just created",
            type = NotificationTypes.Info
        }, "admin");

        // -----

        return Unit.Value;
    }
}
```

Рисунок 2.10 - Метод для створення копанії в CreateCompanyCommandHandler

Метод Handle класу CreateCompanyCommandHandler відповідає за обробку команди створення нової компанії. При отриманні команди, спочатку виконується запит до бази даних для отримання списку всіх компаній. Далі створюється новий об'єкт компанії на основі даних з команди. Перевіряється, чи існує компанія з таким самим ідентифікатором (CompanyId). Якщо така компанія вже існує, генерується виняток AlreadyExistsException.

У разі, якщо компанія не існує, нова компанія додається до бази даних, і зміни зберігаються. Після успішного збереження, надсилаються повідомлення до фронтенду та адміністратору за допомогою сервісу нотифікацій. Повідомлення містять відповідну інформацію про створену компанію. Завершення методу позначається поверненням значення Unit.Value. Це підтверджує успішне виконання команди створення компанії.

```
[HttpGet]
0 references
public async Task<ActionResult> GetProjectsWithParams([FromQuery] ProjectPaginationQueryDTO queryModel)
{
    return Ok(await Mediator.Send(new GetProjectsOverviewQuery(queryModel)));
}
```

Рисунок 2.11 - Метод GetProjectsWithParams в котроллері ProjectController

```
public record GetProjectsOverviewQuery(ProjectPaginationQueryDTO QueryModel) : IRequest<PaginationModel<ProjectOverviewDto>>;

1 reference
public class GetProjectsOverviewQueryHandler : IRequestHandler<GetProjectsOverviewQuery, PaginationModel<ProjectOverviewDto>>
{
    private readonly IApplicationDbContext _context;
    private readonly ICurrentUserService _currentUserService;

    0 references
    public GetProjectsOverviewQueryHandler(IApplicationDbContext context, ICurrentUserService currentUserService)
    {
        _context = context;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<PaginationModel<ProjectOverviewDto>> Handle(GetProjectsOverviewQuery request, CancellationToken cancellationToken)
    {
        var query = request.QueryModel.Query;

        Expression<Func<Domain.Entities.Project, bool>> filterPredicate = project
            => (string.IsNullOrWhiteSpace(query) || project.ProjectName.ToLower().Contains(query.ToLower()))
               && (project.UserId == _currentUserService.Id!.Value || project.Assignees.Any(a => a.User.Id == _currentUserService.Id!.Value))
               && project.Status == request.QueryModel.Status;

        var sort = request.QueryModel.Sort;
        var order = request.QueryModel.Order;
        var page = request.QueryModel.Page;
        var count = request.QueryModel.Count;

        var (projects, total) = await _context.Projects
            .GetFilteredWithTotalSum<ProjectOverviewDto>(filterPredicate, page, count, sort, order);

        return new() { List = projects, Total = total };
    }
}
```

Рисунок 2.12 - Метод отримання списку проектів

Метод Handle класу GetProjectsOverviewQueryHandler обробляє запит для отримання огляду проектів. Використовуючи передані параметри, він будує предикат фільтрації для відбору проектів, які відповідають критеріям. За допомогою методу бази даних, він отримує відфільтровані проекти та їх загальну кількість, і повертає результат у вигляді об'єкта PaginationModel<ProjectOverviewDto>, який містить список проектів та загальну кількість.

```
    [HttpPost]
    [Route("material/create")]
    0 references
    public async Task<IActionResult> CreateMatial(CreateMaterialCommandDto model)
    {
        if (!ModelState.IsValid) return BadRequest("Model is not valid");
        await Mediator.Send(new CreateMaterialCommand(model));
        return StatusCode(201);
    }
```

Рисунок 2.13 - Метод CreateMatial в контроллері ResourceController

```
4 references
public class CreateMaterialCommand : IRequest
{
    1 reference
    public CreateMaterialCommand(CreateMaterialCommandDto model)
    {
        Model = model;
    }
    8 references
    public CreateMaterialCommandDto Model { get; set; }
}

1 reference
public class CreateMaterialCommandHandler : IRequestHandler<CreateMaterialCommand>
{
    private readonly IApplicationDbContext _context;
    private readonly ICurrentUserService _currentUser;

    0 references
    public CreateMaterialCommandHandler(IApplicationDbContext context, ICurrentUserService currentUser)
    {
        _context = context;
        _currentUser = currentUser;

    }

    0 references
    public async Task<Unit> Handle(CreateMaterialCommand request, CancellationToken cancellationToken)
    {
        var materialType = await _context.MaterialType.SingleOrDefault(x => x.Name == request.Model.MaterialType);

        if (materialType == null)
            materialType = new MaterialType() { Name = request.Model.MaterialType };

        var measurement = await _context.Measurements.FirstOrDefault(x => x.Name == request.Model.Measurement);

        var materials = new Material()
        {
            MaterialType = materialType,
            CompanyName = request.Model.CompanyName,
            CompanyAddress = request.Model.CompanyAddress,
            Amount = request.Model.Amount,
            Measurement = measurement,
            Price = request.Model.Price,
            Date = DateTime.UtcNow,
            CompanyId = (long)_currentUser.CompanyId,
        };

        await _context.Materials.Insert(materials);
        await _context.SaveChangesAsync(cancellationToken);
        return Unit.Value;
    }
}
```

Рисунок 2.14 - Клас CreateMaterialCommand для добавлення матеріалу в список

Метод Handle класу CreateMaterialCommandHandler обробляє запит на створення матеріалу. Спочатку він перевіряє наявність типу матеріалу в базі даних і, якщо такий тип не існує, створює новий запис. Далі, він шукає в базі

даних одиницю виміру, яка відповідає переданому значенню, і створює новий об'єкт Material з відповідними властивостями. Після цього, новий матеріал додається до бази даних за допомогою методу Insert, і зміни зберігаються. Після успішного виконання, метод повертає значення Unit.Value.

Висновки до розділу 2

Saas – платформа для управління компаніями та працівниками надає широкий функціонал для ефективного управління компаніями та їх персоналом. Ця платформа дозволяє створювати компанії, розподіляти команди, керувати ресурсами та сервісами, а також відстежувати прогрес у будівництві. Вона надає реалізовані методи, що дозволяють забезпечувати всі необхідні функції для управління компаніями та їх працівниками.

У розділі проекту була реалізована база даних відповідно до вимог реляційної моделі, що забезпечує збереження та доступ до даних у потрібний момент для користувачів системи. База даних складається з 24 таблиць та має назву CtorDB. Ця база даних дозволяє зберігати всю необхідну інформацію та забезпечує зручний доступ до неї для використання у функціях платформи.

В останньому пункті детально розглядається реалізований код додатку. Основна увага зосереджена на функціях збереження даних та їх виведенні на клієнтську частину, а також на функціях пошуку, фільтрації та основних операцій роботи з базою даних. Це дозволяє забезпечити ефективну та зручну роботу зі збереженими даними на платформі.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання є візуальним зображенням конфігурації вузлів обробки та компонентів системи під час їх розгортання. Це один з видів структурних діаграм, що використовуються для моделювання фізичних аспектів об'єктно-орієнтованої системи, зокрема топології апаратного забезпечення. Діаграма розгортання дозволяє візуально представити, як компоненти системи розміщені на обчислювальних вузлах та взаємодіють між собою.

На рисунку 3.1 наведена діаграма розгортання, яка ілюструє конфігурацію вузлів та компонентів системи під час розгортання.

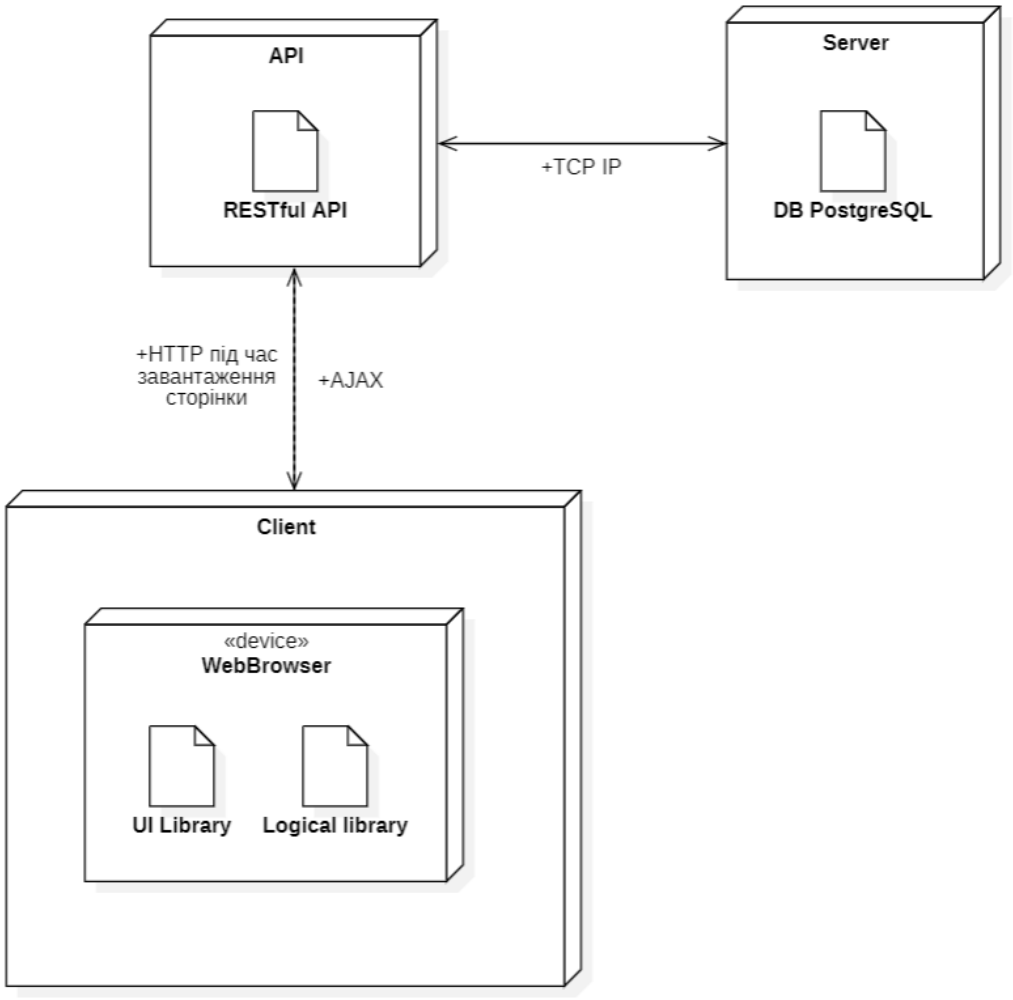


Рисунок 3.1 - Діаграма розгортання додатку

Після побудови даної діаграми розгортання можна зробити наступні висновки щодо порядку встановлення та налаштування параметрів системи:

1. Бекенд частина API веб-застосунку розгортається на порті 7271, в той час як клієнтська частина розгортається на порті 4200.
2. Бекенд частина взаємодіє з сервером для отримання необхідних ресурсів.
3. Клієнтська частина представляє собою додаток, який використовує бібліотеки для побудови користувацького інтерфейсу та налаштування логіки оновлення даних і сторінок.
4. Клієнтська частина отримує початкову HTML-розмітку при запуску додатку і далі оновлює її за допомогою асинхронних звернень до API.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-орієнтованим інформаційним додатком

Після відкриття веб-застосунку користувача зустрічає сторінка авторизації.

На рисунках 3.2.1, 3.2.2, 3.3, 3.4 зображено авторизацію в систему.

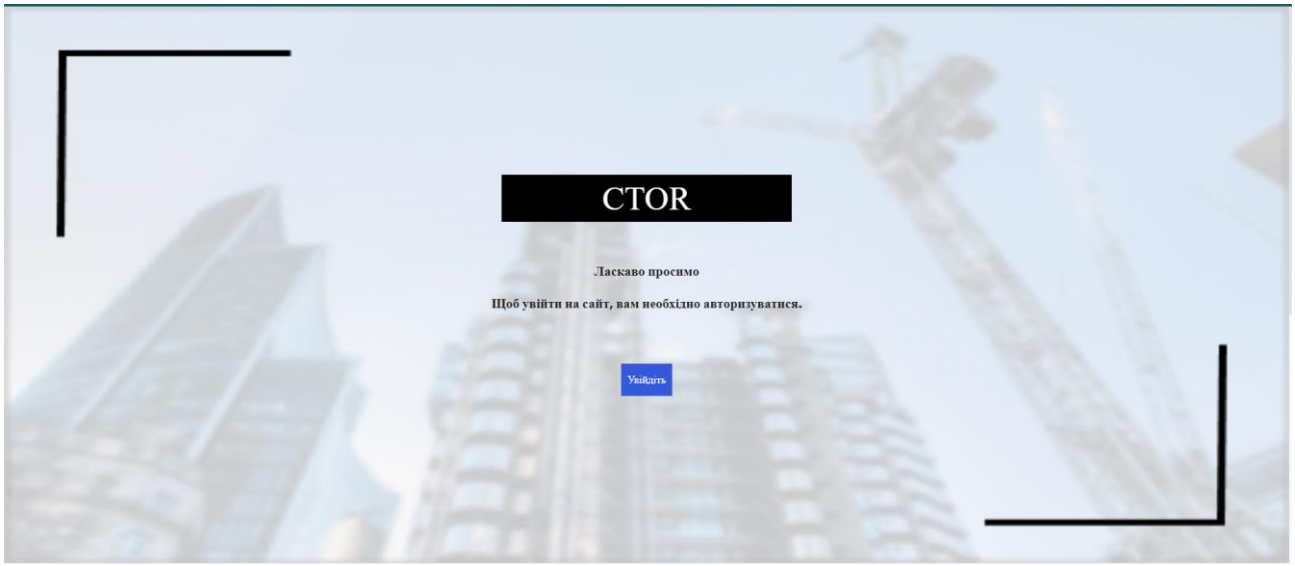


Рисунок 3.2.1 - Сторінка авторизації



Рисунок 3.2.2 - Сторінка авторизації(опис)

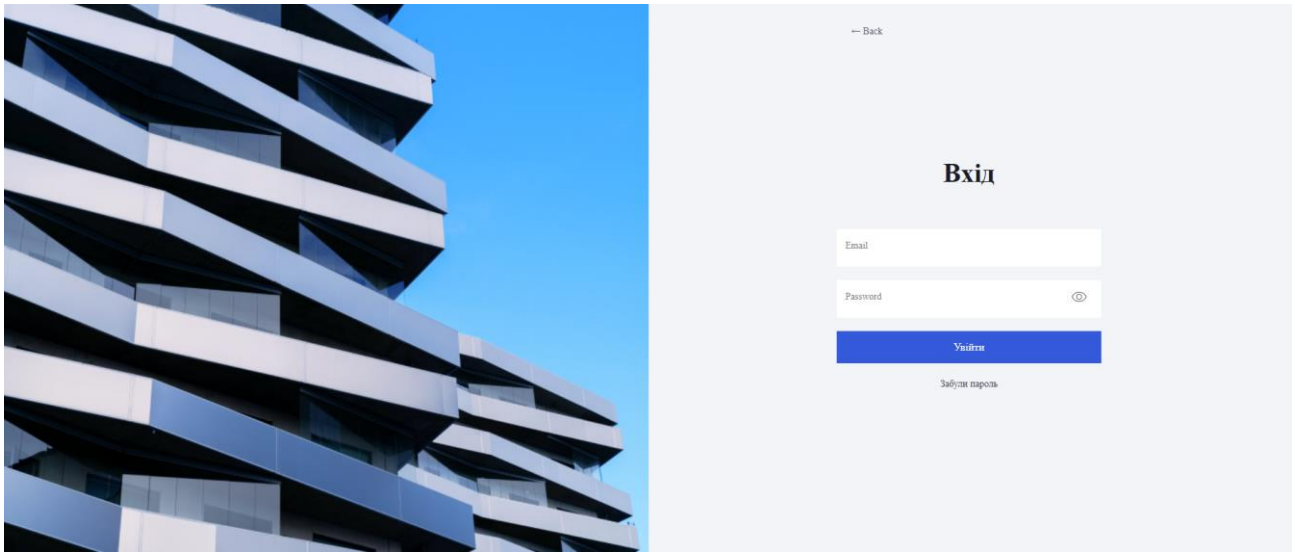


Рисунок 3.3 - Форма авторизації

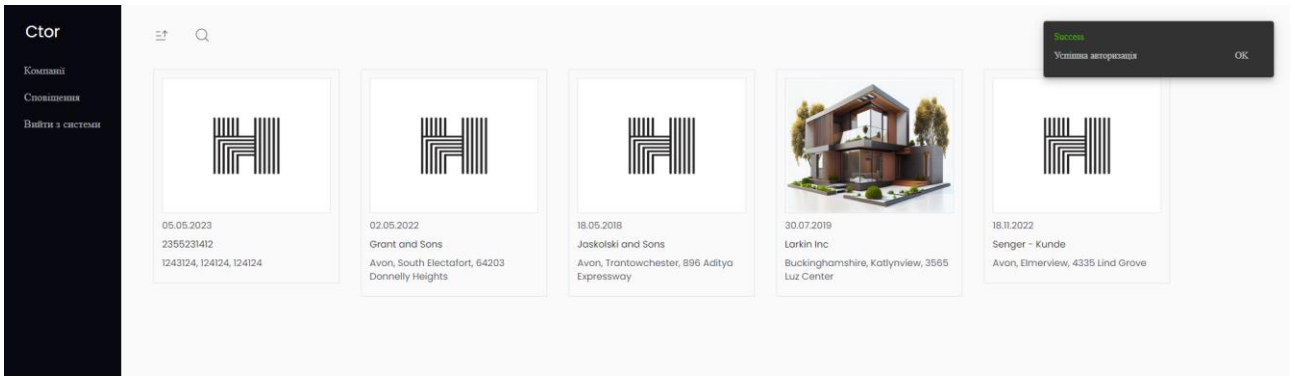


Рисунок 3.4 - Авторизація користувача адміна

Було показано процес авторизації адміністратора системи користувачів, після чого відбулося перенаправлення на головну сторінку компаній. Також успішно спрацювало сповіщення про успішну авторизацію, яке було надіслано користувачеві.

3.2.1 Функціональні можливості адміністратора.

1. Для створення компанії необхідно натиснути кнопку"Додати компанію", яка перенаправить нас на форму створення компанії. На рисунках 3.5, 3.6, 3.7, 3.8 та 3.9 показано послідовність процесу.

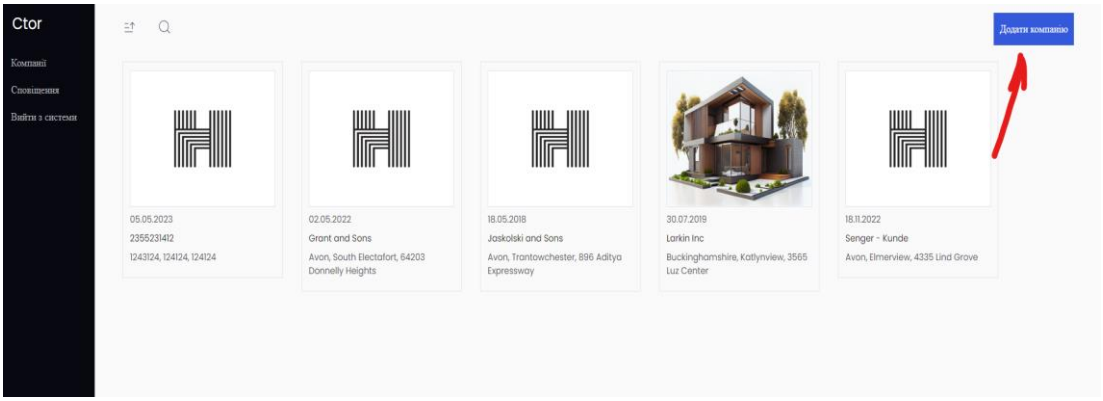


Рисунок 3.5 – Кнопка створення нової компанії

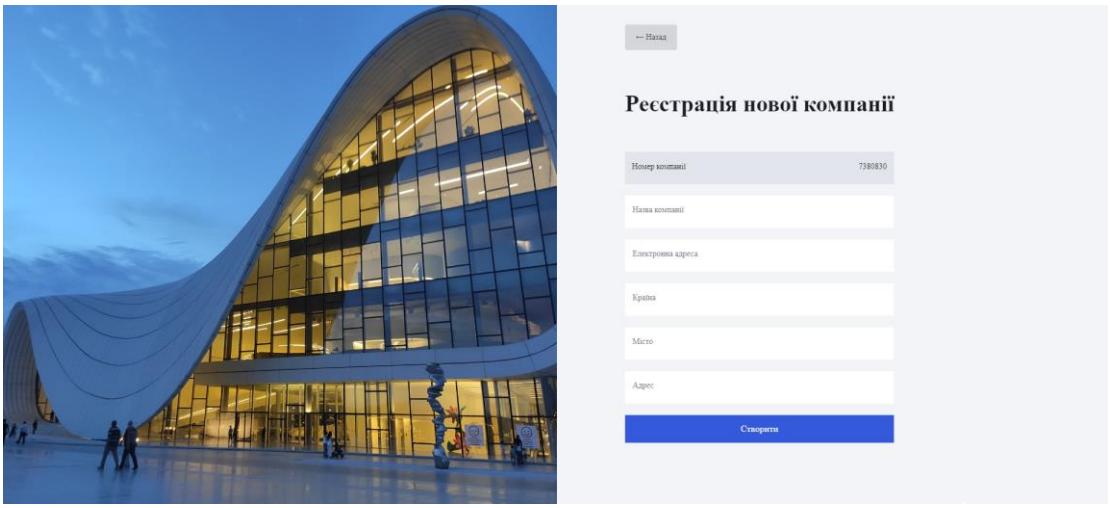


Рисунок 3.6 - Форма створення компанії

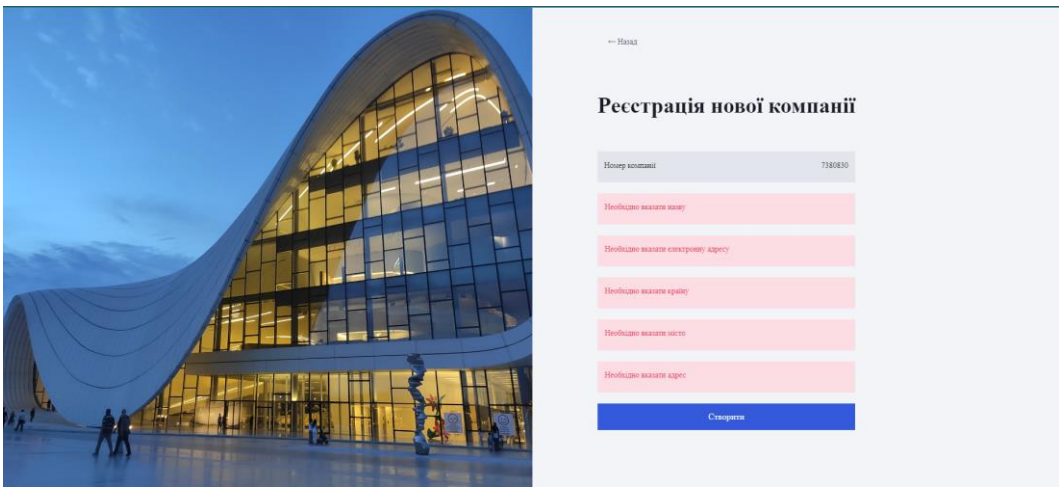


Рисунок 3.7 - Валідація на пусті поля в формі

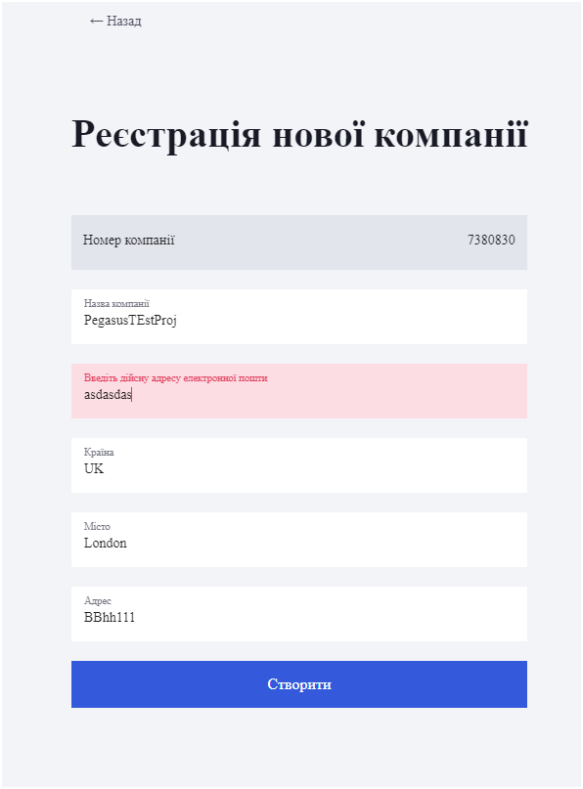


Рисунок 3.8 – Валідація електроної пошти

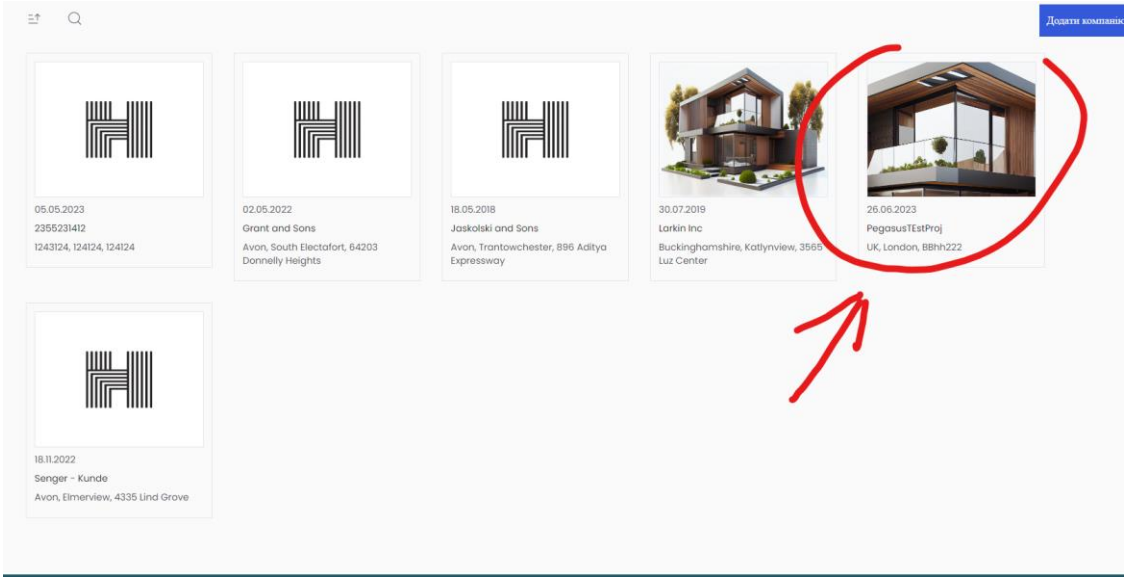


Рисунок 3.9 – Результат нової створеної компанії "Pegasustestproj"

2. У цьому розділі буде показано функціонал, пов'язаний зі зміною опису проекту та додаванням учасників до компанії. На рисунках 3.10, 3.11, 3.12, 3.13:



Рисунок 3.10 – Сторінка новоствореної компанії



Рисунок 3.11 – Кнопка редагування опису компанії

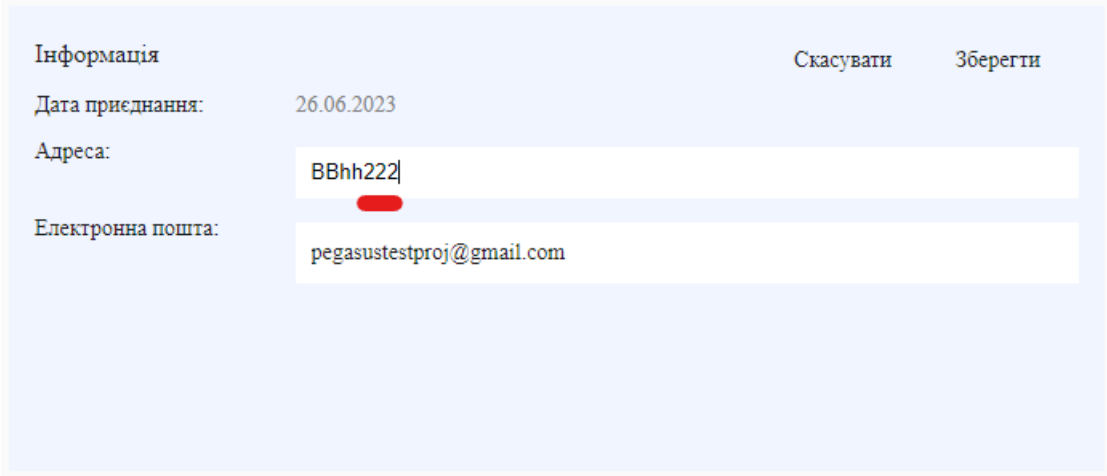


Рисунок 3.12 - Процес зміни опису компанії

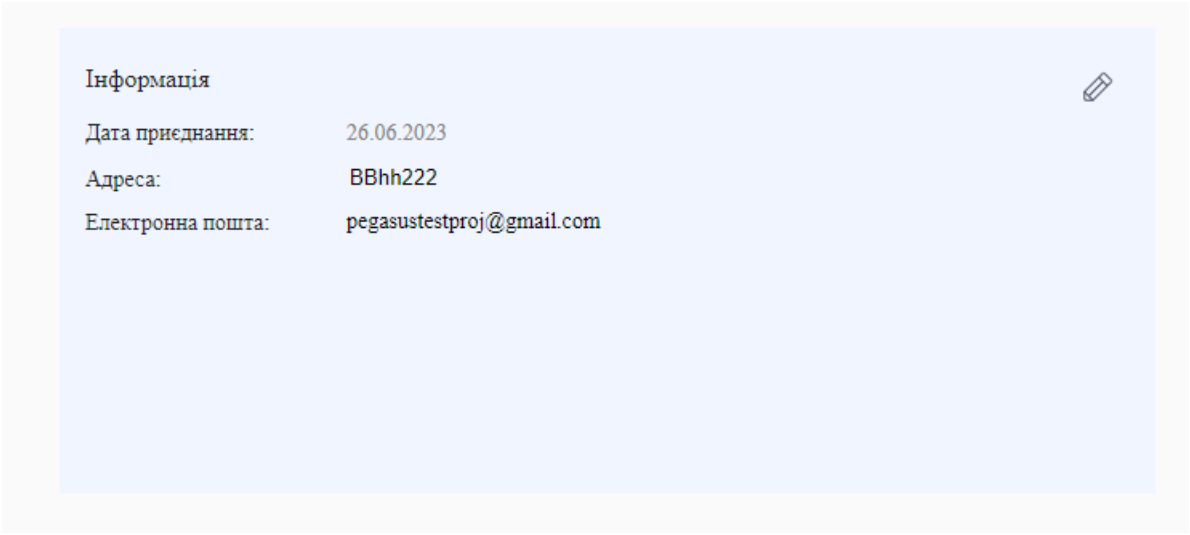


Рисунок 3.13 - Успішний результат зміни опису

Ці рисунки ілюструють послідовність дій користувача під час зміни опису проекту та додавання учасників до компанії.

3. У цьому пункті буде розглянуто процес додавання працівників до компанії. Будуть описані дії та кроки, необхідні для успішного включення нових працівників до команди компанії. На рисунках 3.14, 3.15, 3.16, 3.17, 3.18.

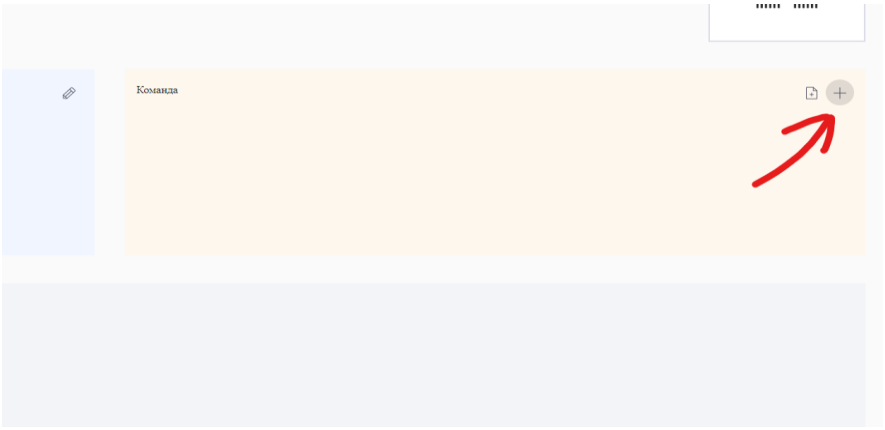


Рисунок 3.14 – Копка добавлення нового парацівника

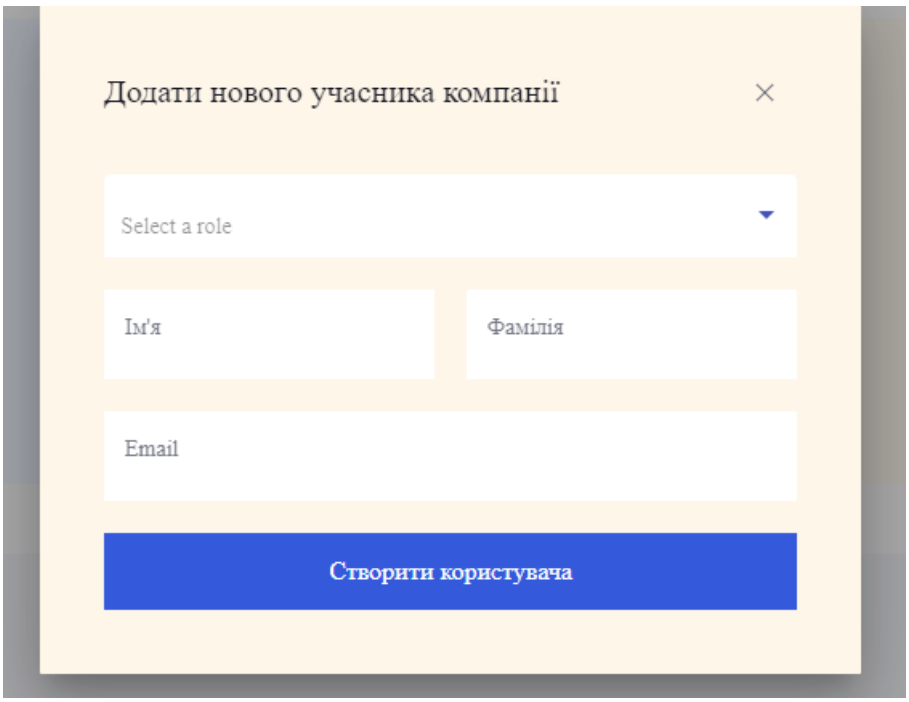


Рисунок 3.15 форма для введення даних нового члена команди

На скріншоті показана форма для введення даних нового члена команди, де можна вказати необхідну інформацію.

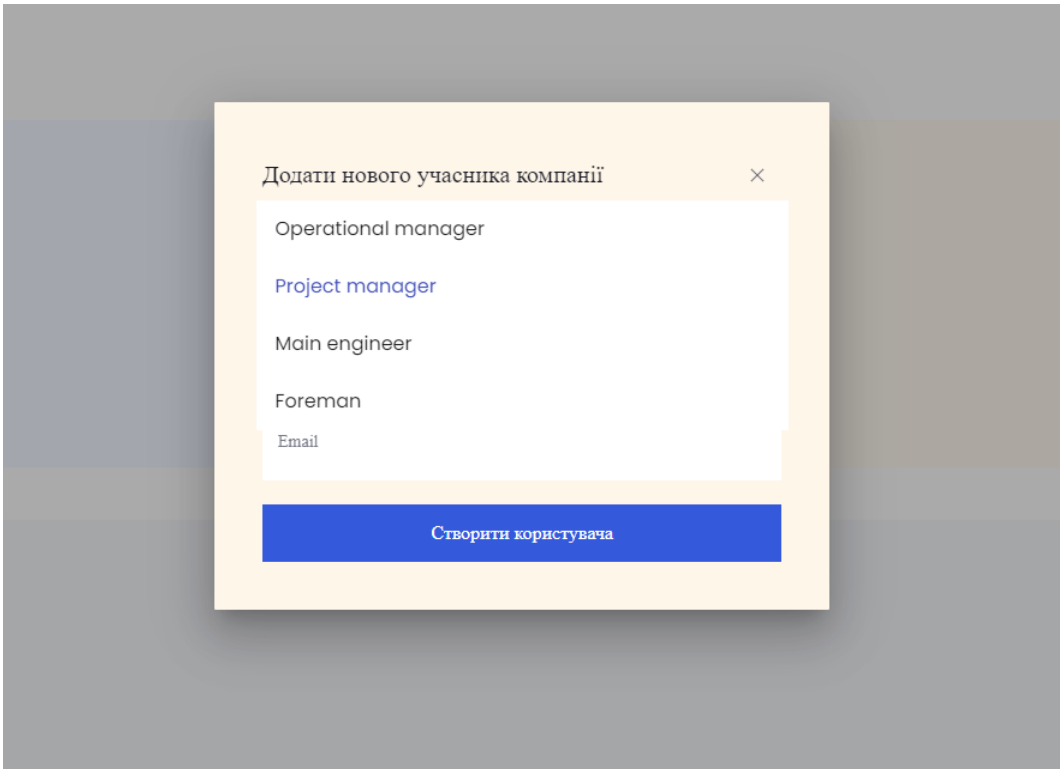


Рисунок 3.16 – Випадаючий список

Даний рисунок демонструє випадаючий список з доступними ролями, з якими можна пов'язати нового працівника.

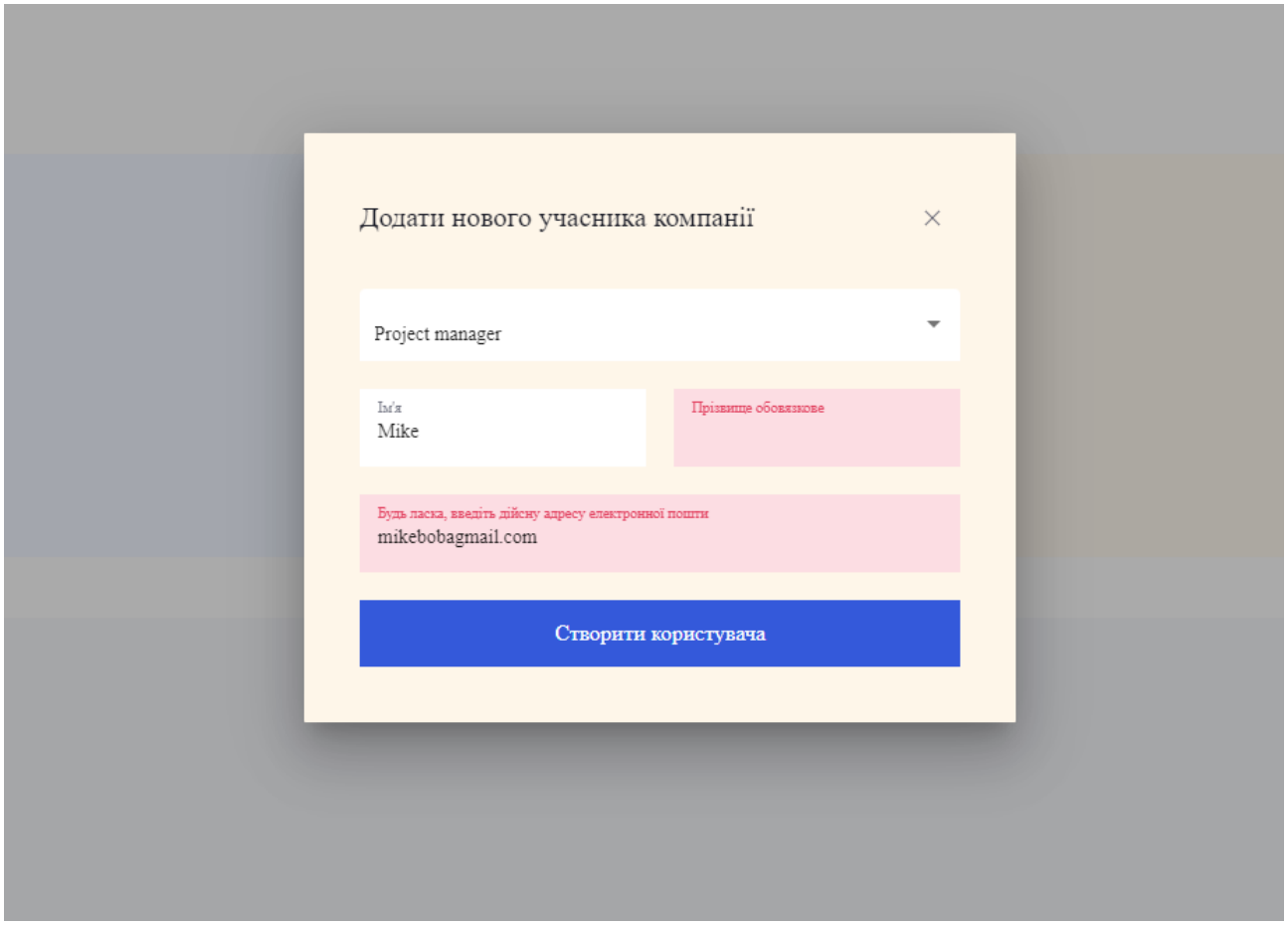


Рисунок 3.17 – Валідація

Демонструється процес валідації пустих полів для email та прізвища нового працівника.

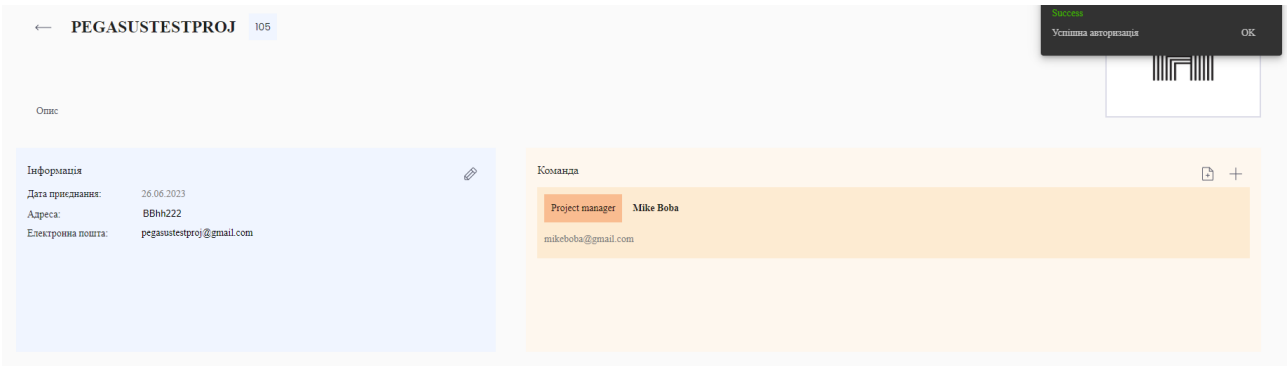


Рисунок 3.18 - Успішне додавання працівника

3.2.2 Функціональні можливості операційного менеджера.

4. В цьому розділі буде описано функціонал роботи з проектами для користувача з правами Операційного менеджера. На рисунках 4.9 і 4.10 будуть показані наступні етапи та результати:

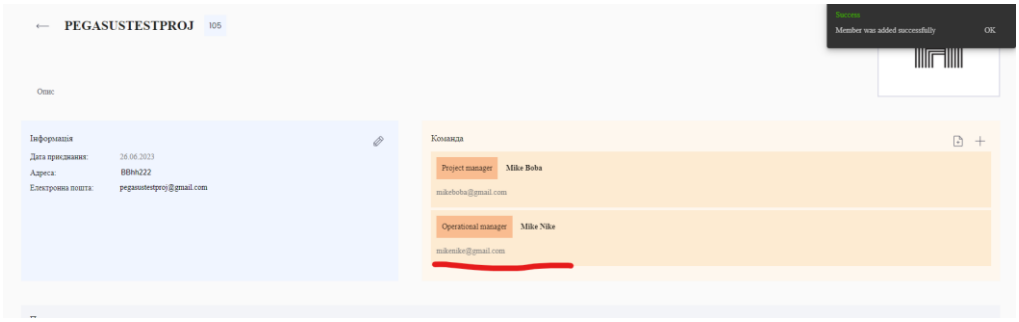


Рисунок 3.19 - Успішне додавання працівника

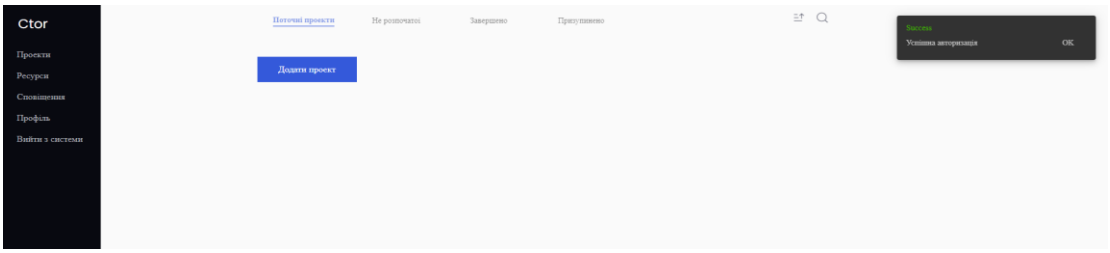


Рисунок 3.20 – Успішна авторизація

Показує успішну авторизацію під новоствореним користувачем.

Цей рисунок підтверджує, що користувач зміг успішно увійти в систему під своїм новим обліковим записом.

Ці рисунки демонструють функціонал, пов'язаний з додаванням працівників до компанії та авторизацією користувачів з правами Операційного менеджера.

5. У цьому етапі буде демонстрована форма для створення нового проекту для компанії.

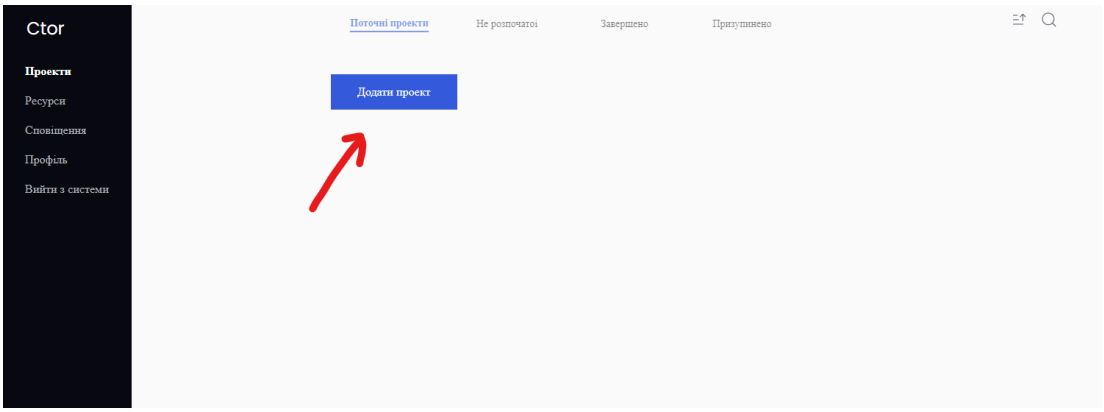


Рисунок 3.21 - Кнопка для створення проекту

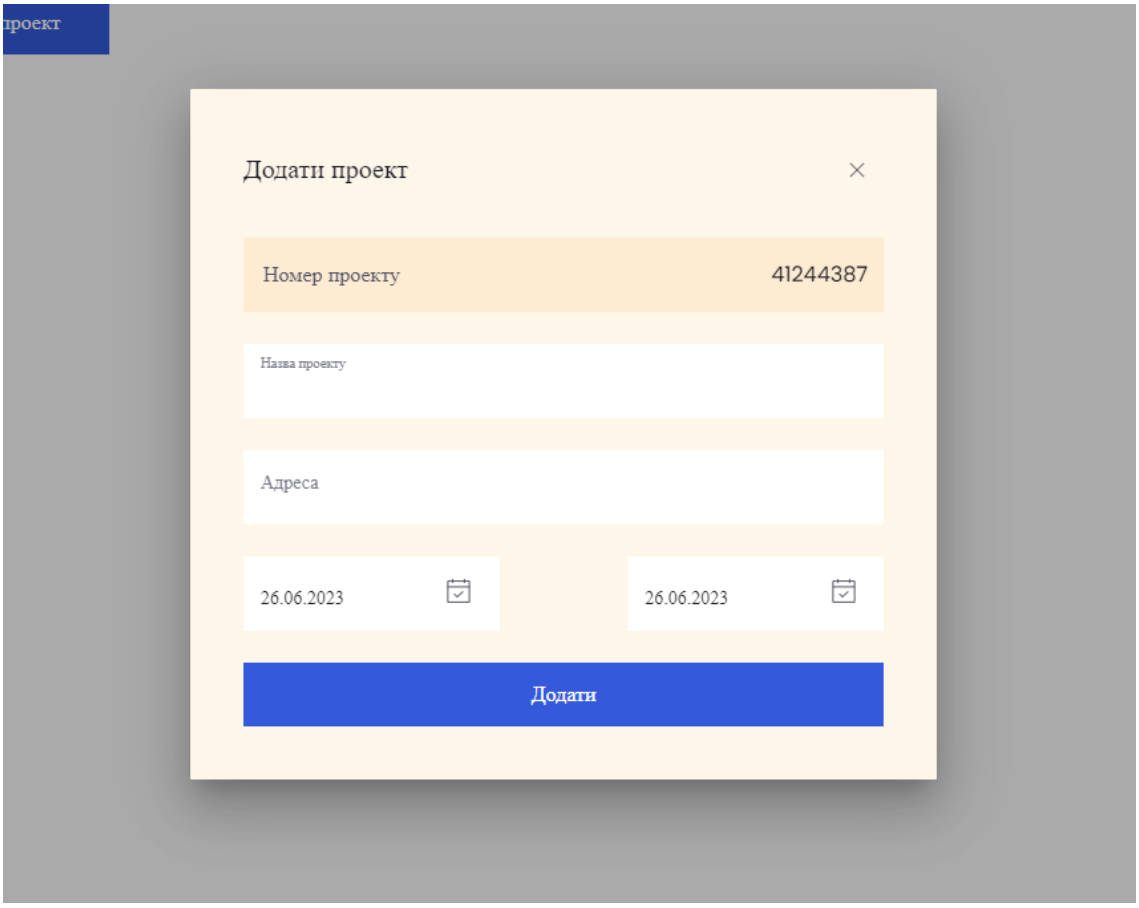


Рисунок 3.22 - Форма для створення нового проекту

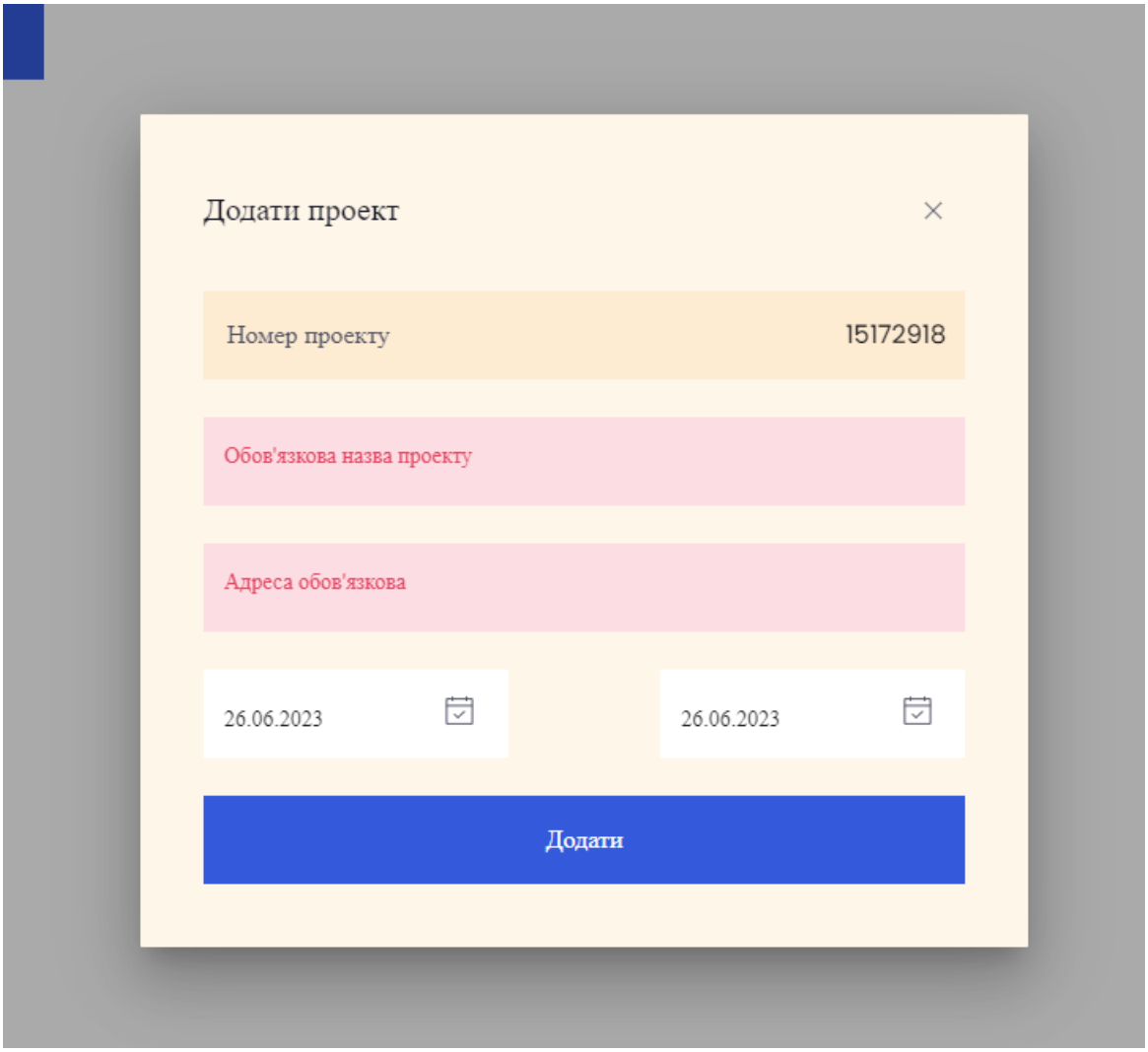


Рисунок 3.23 - Валідація

На формі вказані обов'язкові поля та проведена валідація, включаючи номер проекту, назву проекту та адресу.

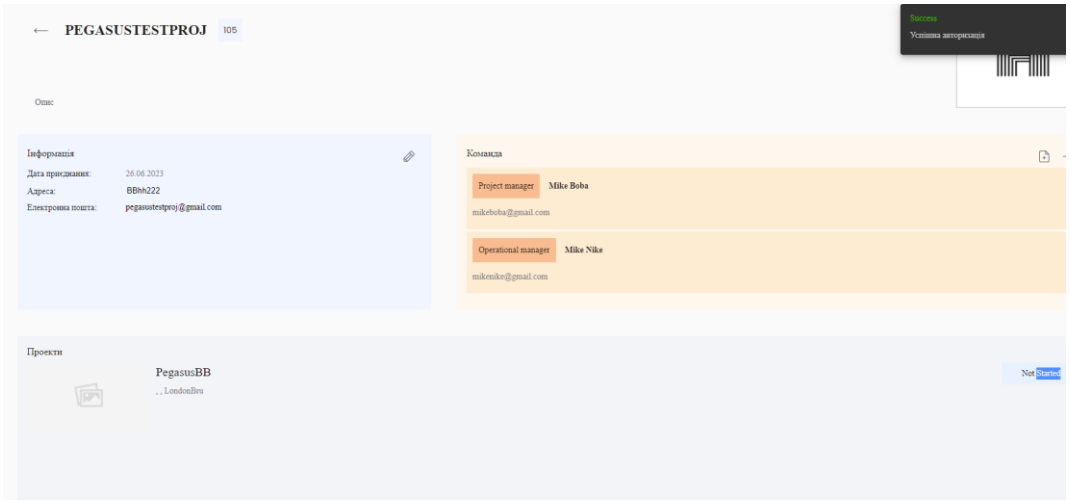


Рисунок 3.24 - Проект успішно створено

Відображається в кабінеті компанії, до якої належить (для користувача-адміністратора).

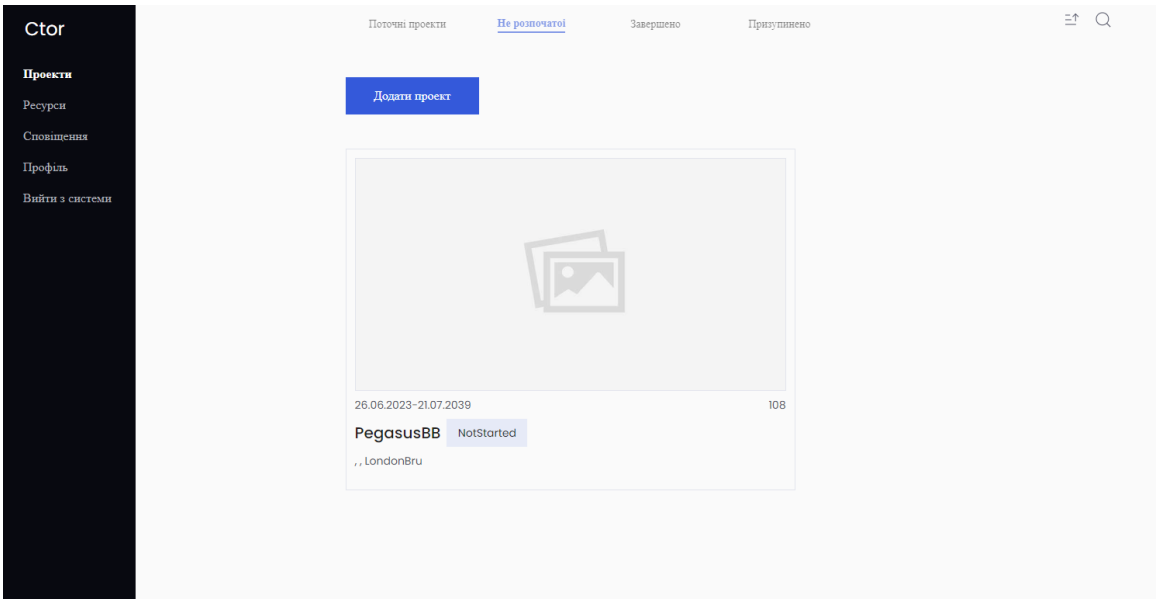


Рисунок 3.25 – Результат

Проект також відображається на головній сторінці оперативного менеджера, оскільки він належить до компанії, в якій він працює. На даний момент проект знаходиться у вкладці "Нерозпочаті проекти" зі статусом "NotStarted".

6. В даному пункті буде показана функція зміни опису компанії.

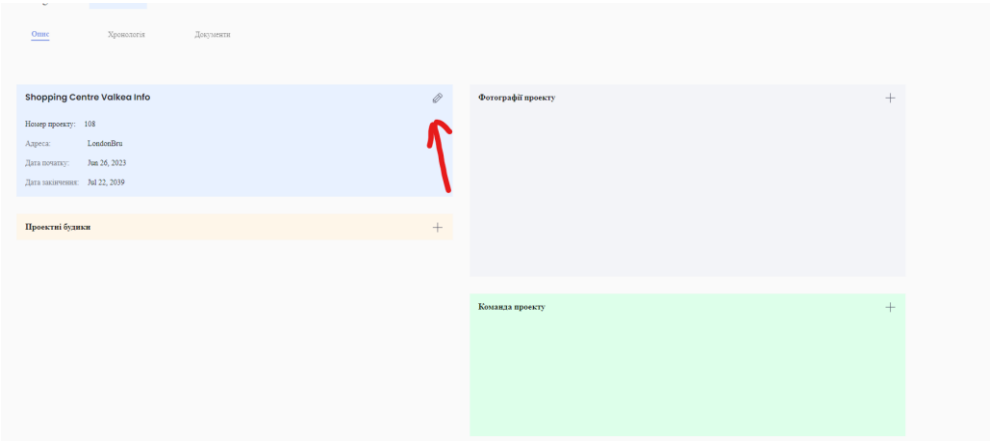


Рисунок 3.26 - Кнопка для зміни опису проекту

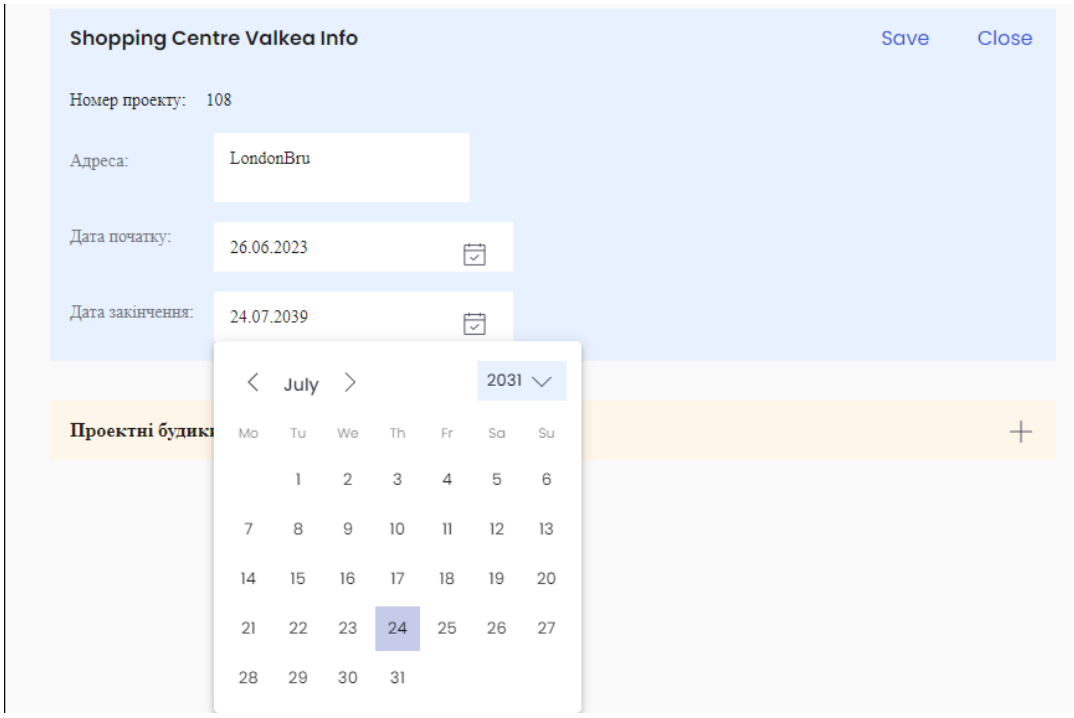


Рисунок 3.27 - Зміна дати

Показана зміна дати закінчення проекту з 2039 року на 2031 рік



Рисунок 3.28 - Успішний результат зміни даних

7. На даному прикладі буде продемонстровано добавлення фотографій на проект за допомогою модального вікна та функції перетягування і розміщення (drag and drop).

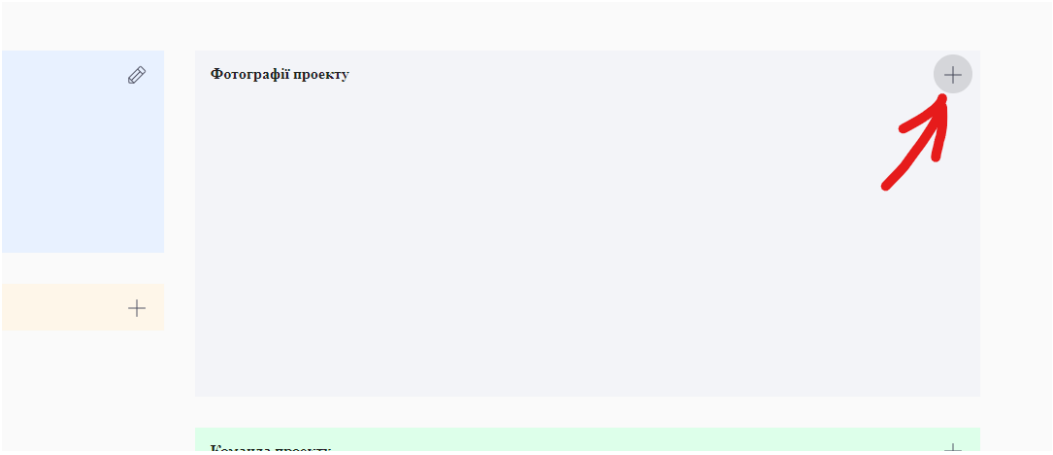


Рисунок 2.29 - Кнопка "Додати фотографії"

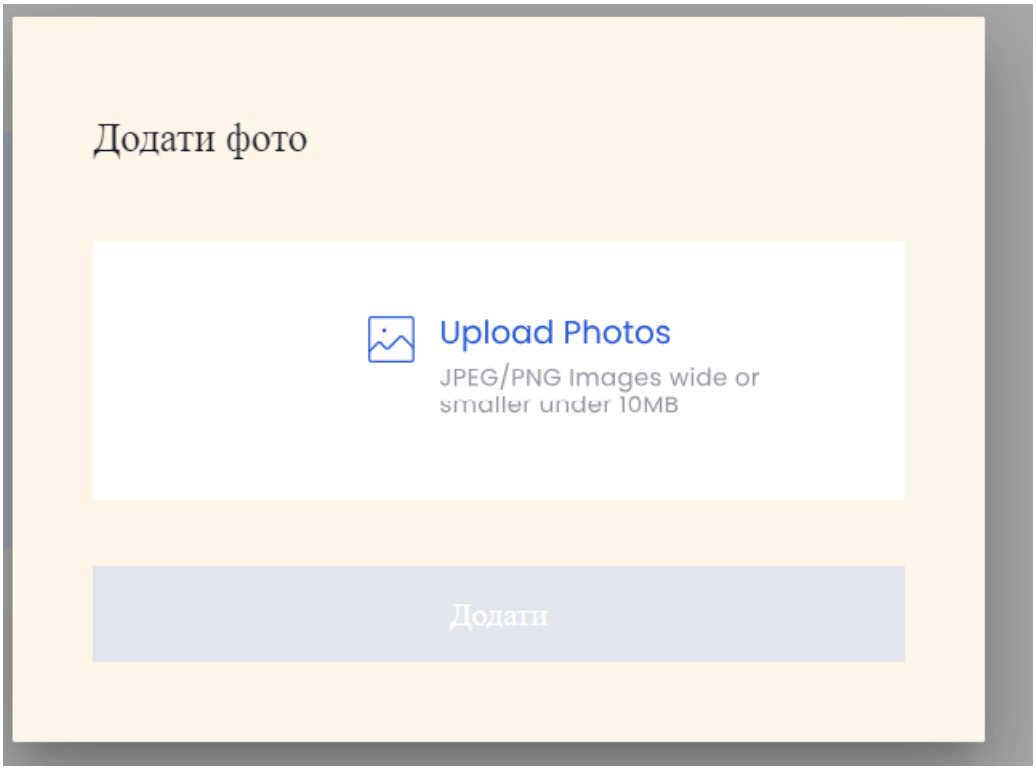


Рисунок 3.30 - Модальне вікно для вибору фотографій

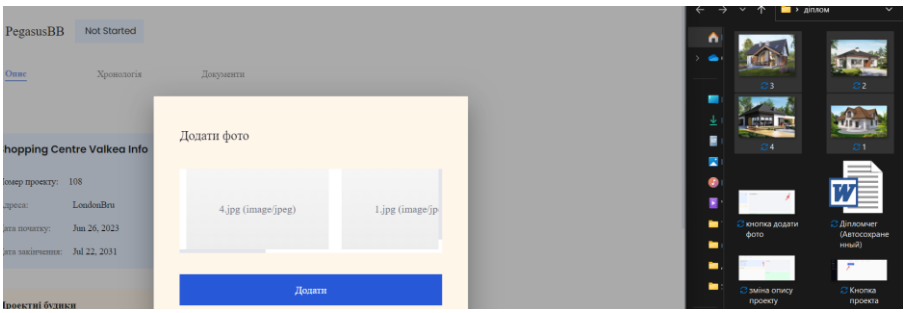


Рисунок 3.31 – Прикріплені файли

Вибрані фотографії можна перетягувати та розміщувати в потрібному порядку на проекті.

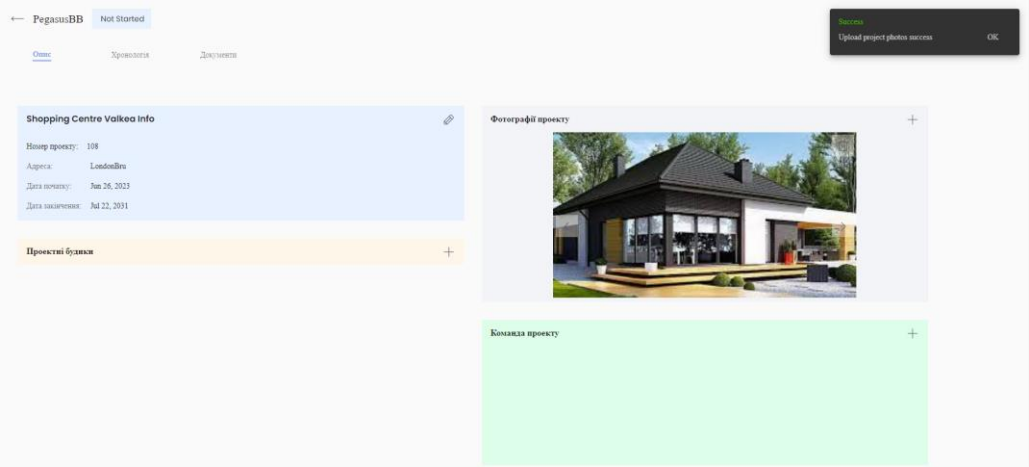


Рисунок 3.32 - Фотографії успішно додані



Рисунок 3.33 - Показ роботи слайдера



Рисунок 3.33 - Видалення фотографії з галереї

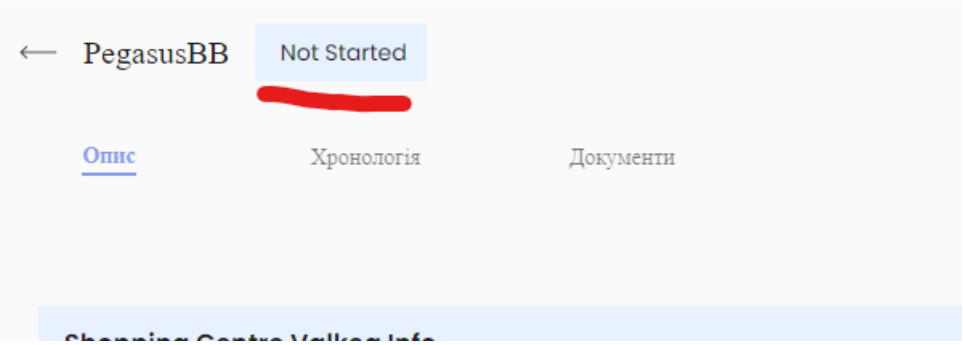


Рисунок 3.34 - Зміна статусу проекту на "In Progress"



Рисунок 3.35 - Успішна зміна статусу проекту

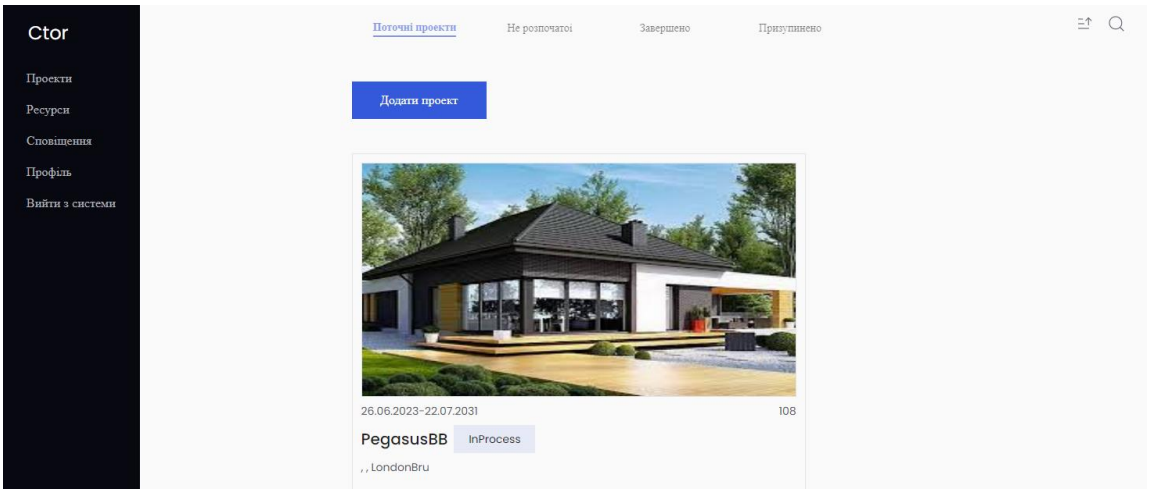


Рисунок 3.36 - розділ "Поточні проекти" зі статусом "In Progress"

Ці зображення показують дії, пов'язані з роботою з фотографіями, зміною статусу проекту та їх відображенням в системі.

8. В цьому етапі буде показано, як додавати нових працівників на проект з боку операційного менеджера.

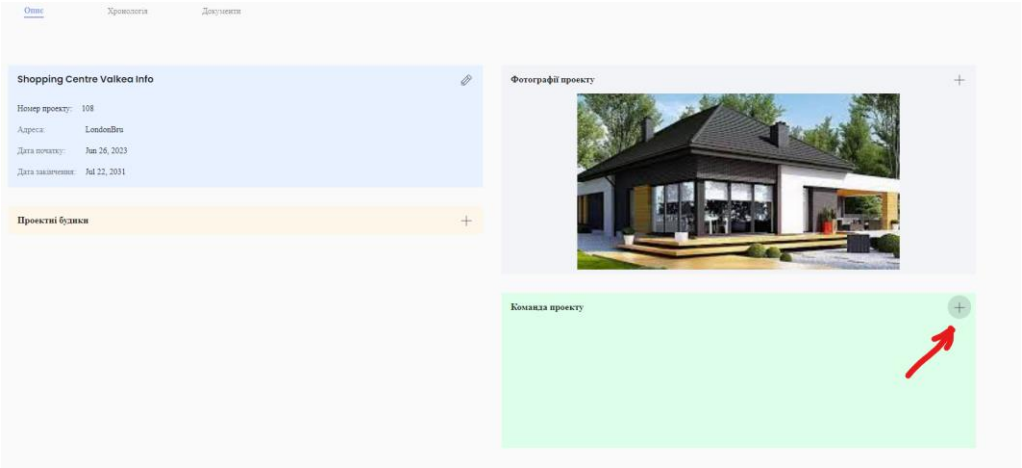


Рисунок 3.37 - Кнопка для додавання нового працівника

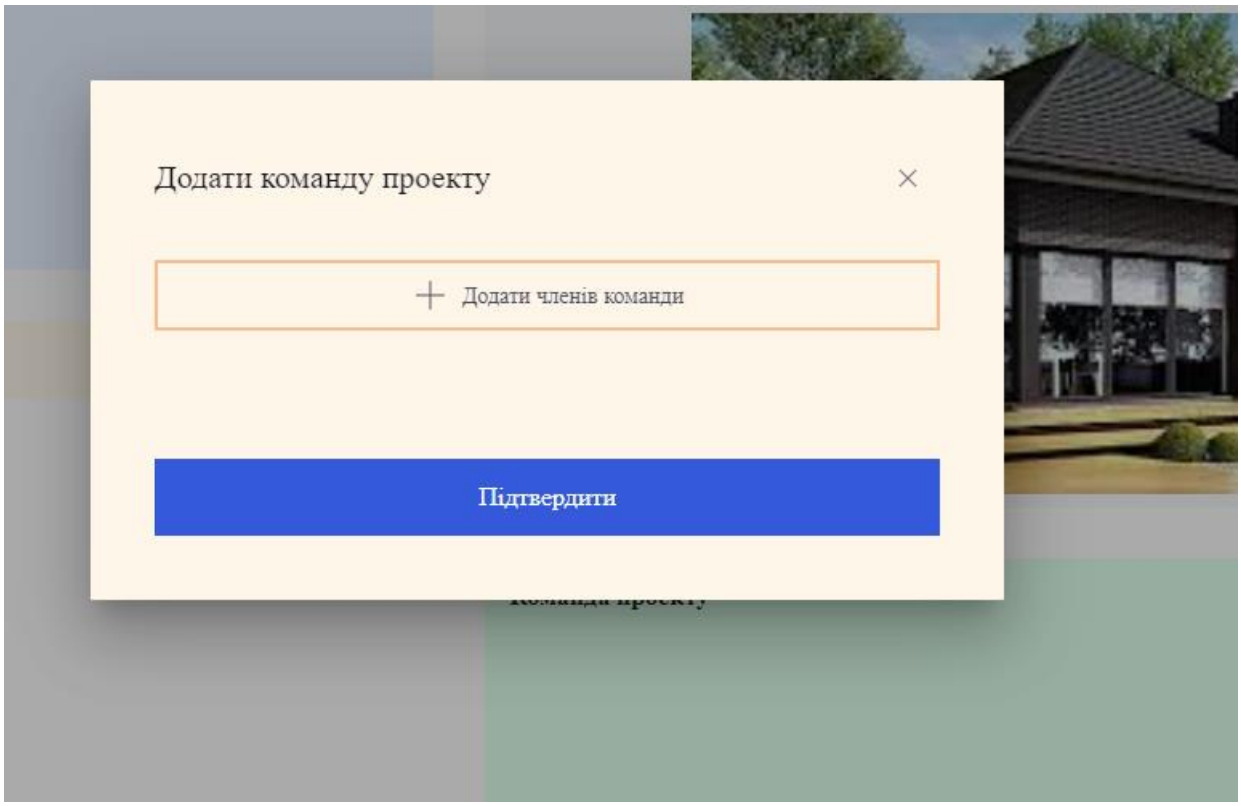


Рисунок 3.38 - Форма для додавання користувачів

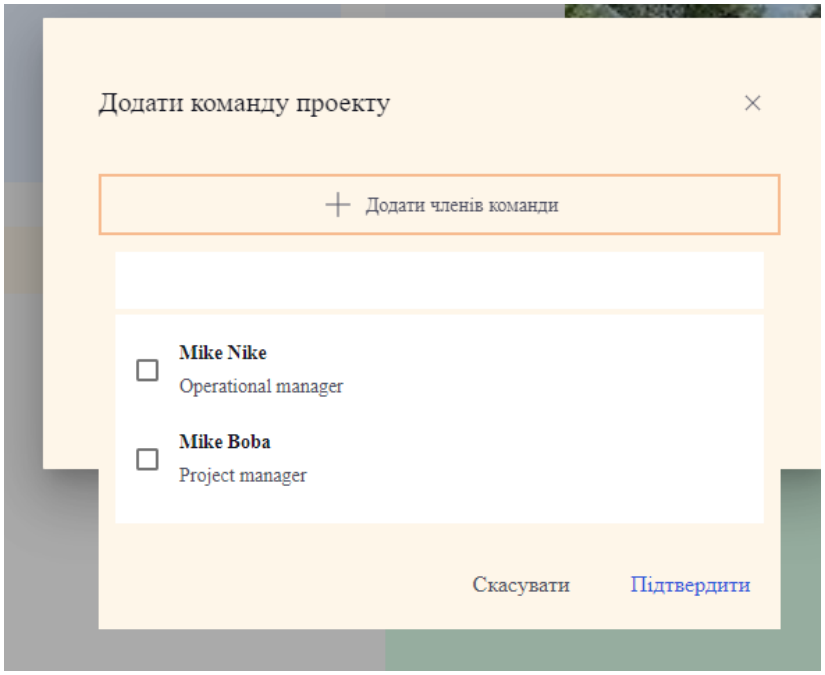


Рисунок 3.39 - Випадаючий список зі списком доступних користувачів для проекту

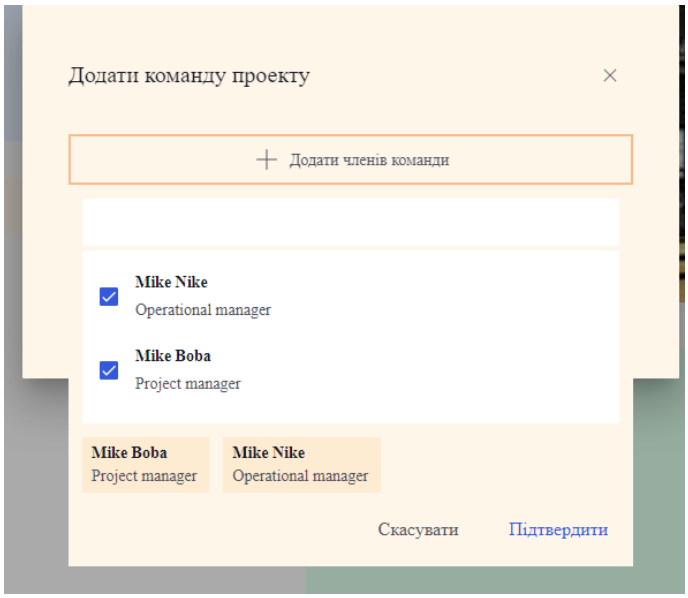


Рисунок 3.40 - Вибрані користувачі зі списку

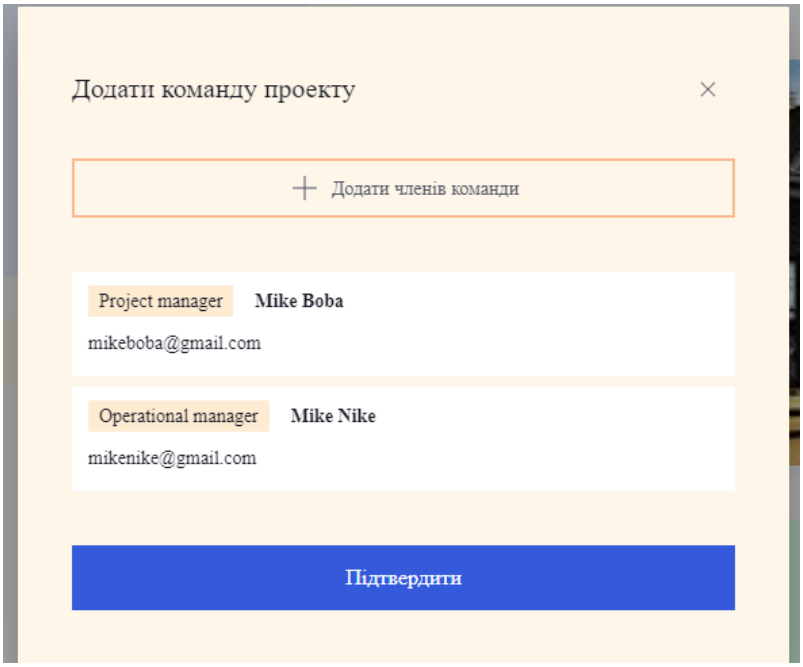


Рисунок 3.41 - Вікно перед відправленням форми

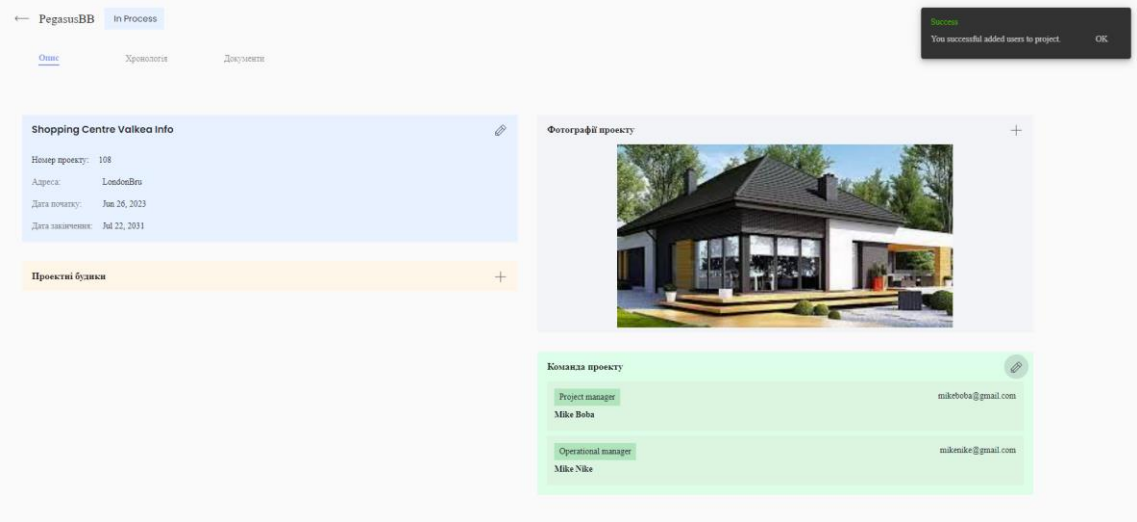


Рисунок 3.42 - Успішне додавання користувачів та остаточний результат

9. В даному пункті буде показано як відбувається добавлення нових споруд які належать проекту, розподіл ресурсів та створення фаз.

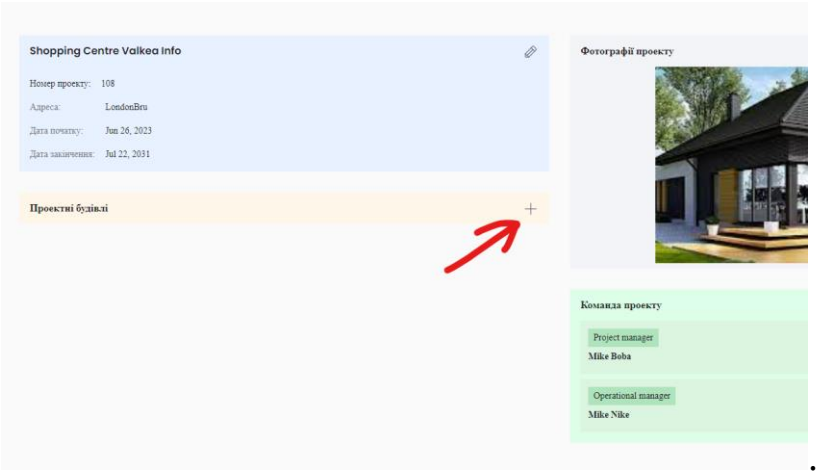


Рисунок 3.43 - Кнопка для додавання нової споруди



Рисунок 3.44 - Модифікована форма для додавання будівлі

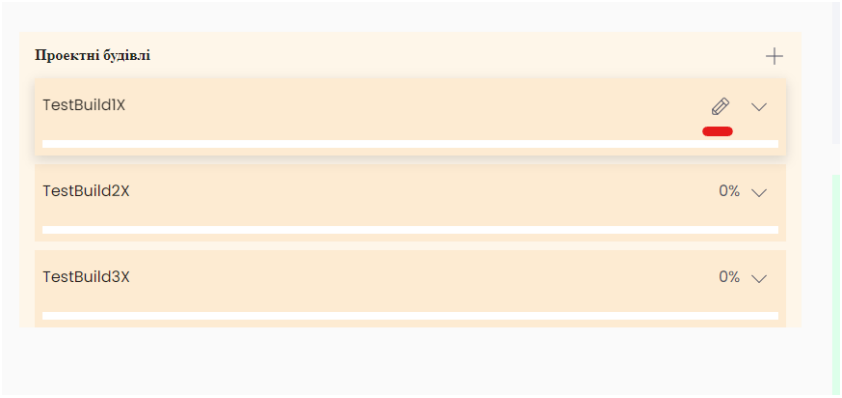


Рисунок 3.45 - Кнопка для редагування потрібного блоку

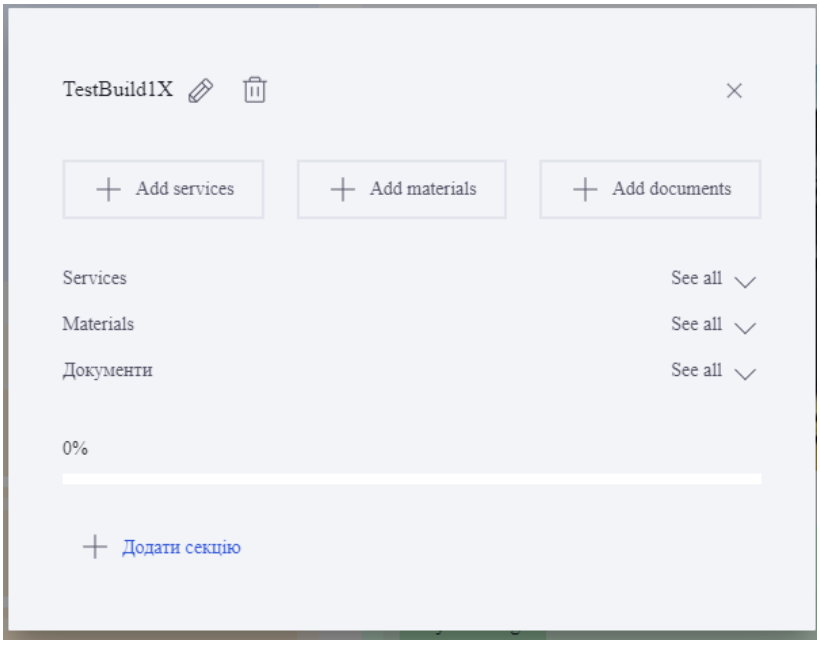


Рисунок 3.46 - Форма редагування блоку

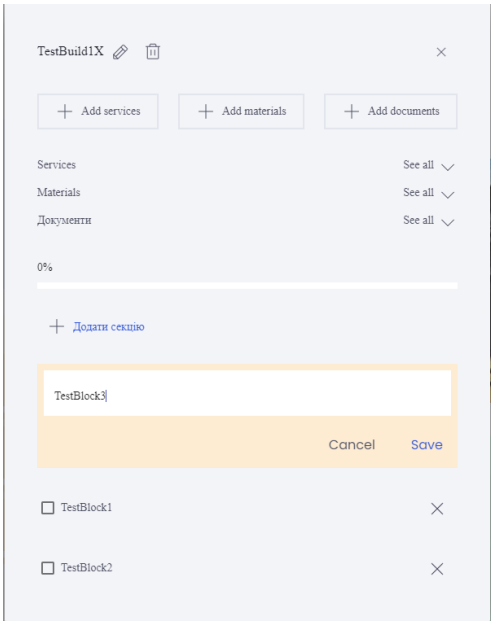


Рисунок 3.47 - Додавання нових секцій будівництва

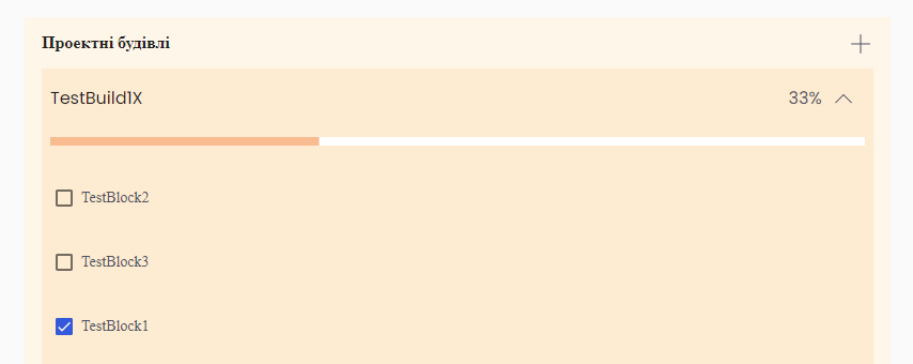


Рисунок 3.48 - Результат показу доданих секцій

Також є можливість прикріплювати нові сервіси до будівлі, але перед цим необхідно перейти до головного меню, вибрати вкладку "Ресурси" і додати нові послуги до списків.

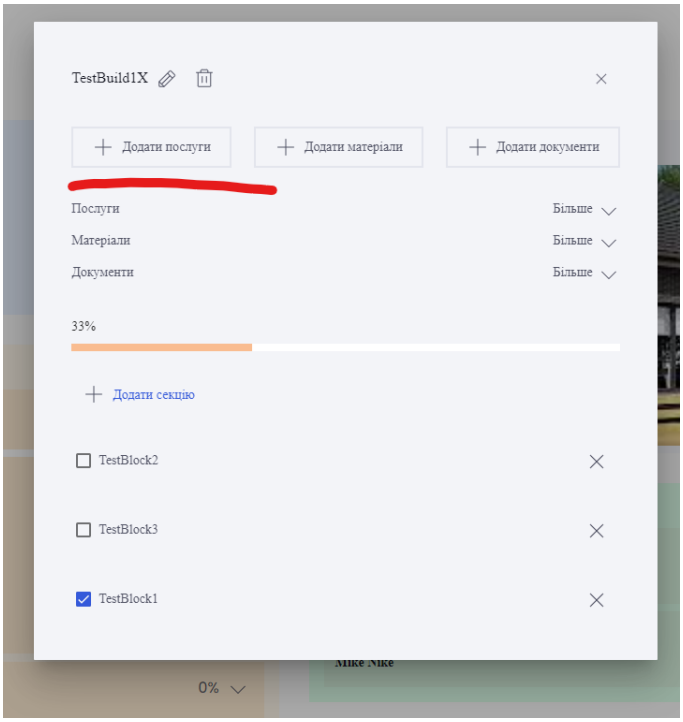


Рисунок 3.49 - Кнопка для додавання послуг

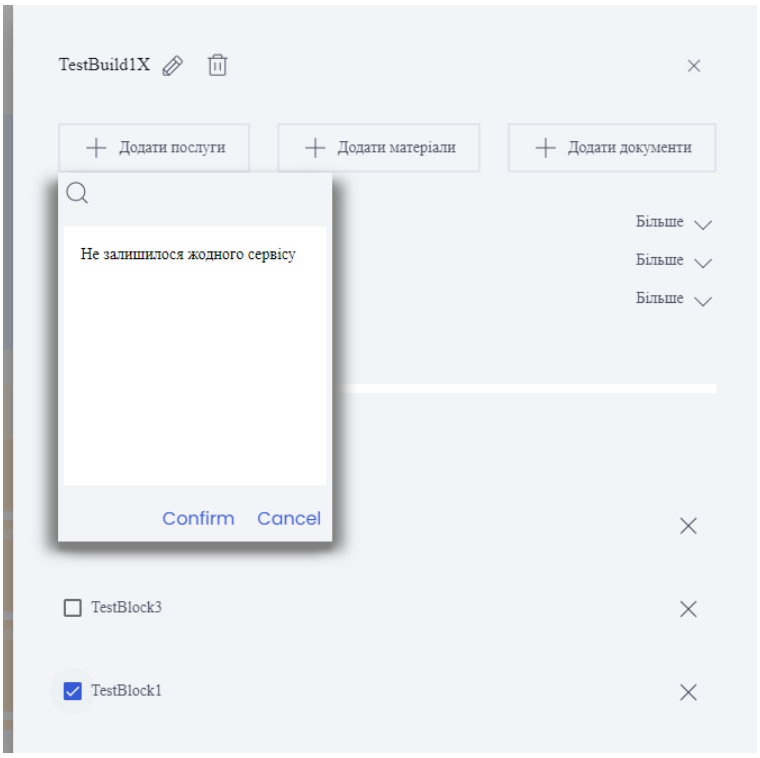


Рисунок 3.50 - Модальне вікно, коли в компанії немає доступних ресурсів



Рисунок 3.51 - Додавання нових сервісів в розділі ресурсів



Рисунок 3.52 - Успішне додавання нових послуг та результат валідації незаповнених полів

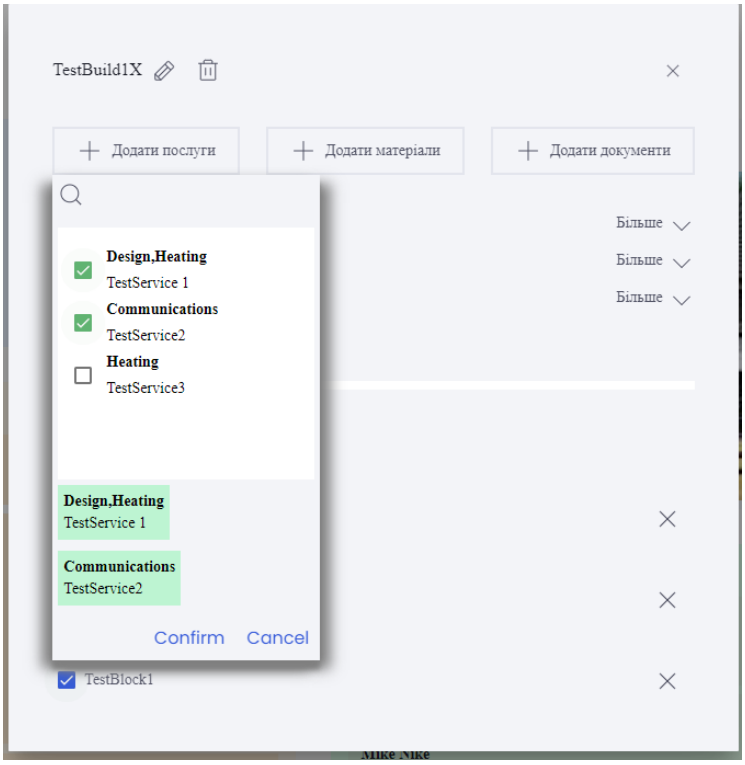


Рисунок 3.53 - Додавання нових послуг до стадії проекту

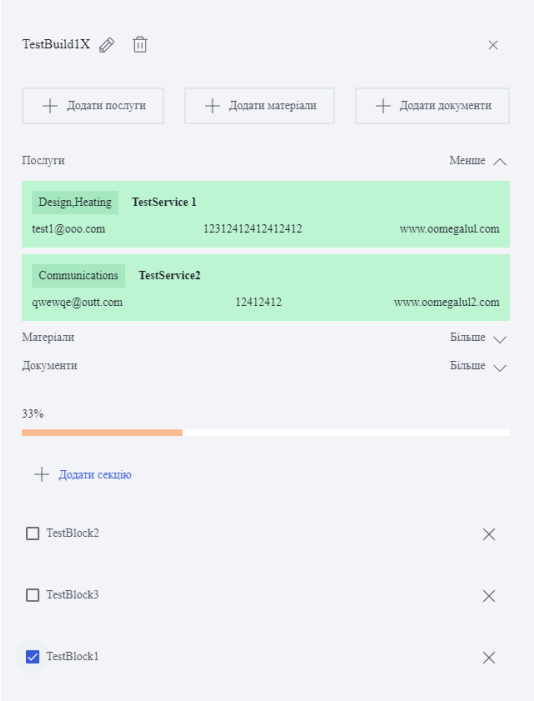


Рисунок 3.54 - Результат доданих послуг до секції будівлі

Також ми можемо прикріпити документи різних форматів до секції проекту.

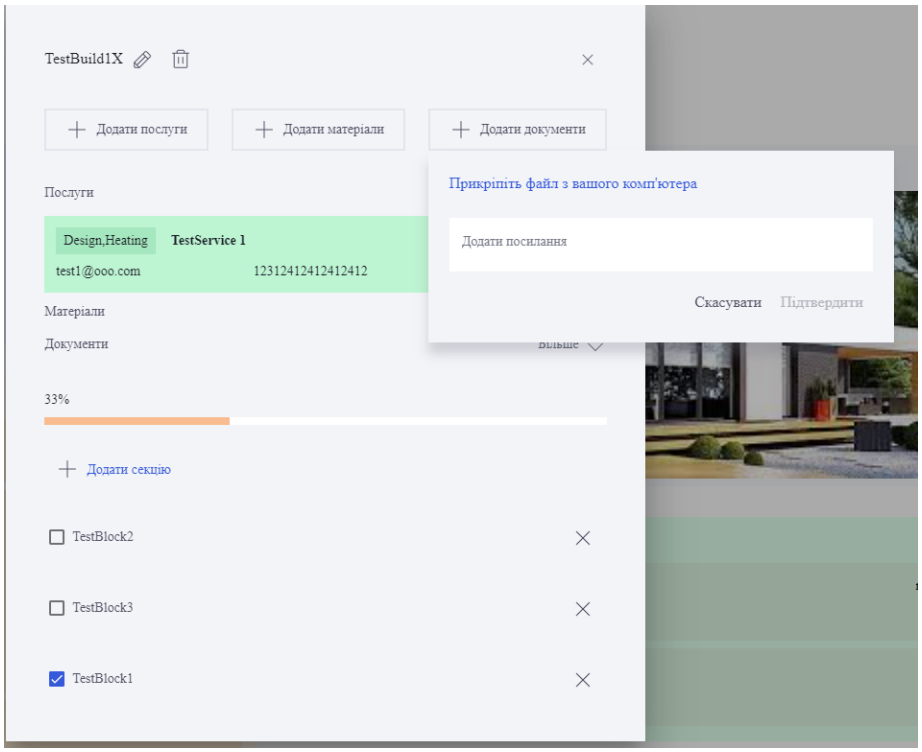


Рисунок 3.55 - Кнопка для додавання документу

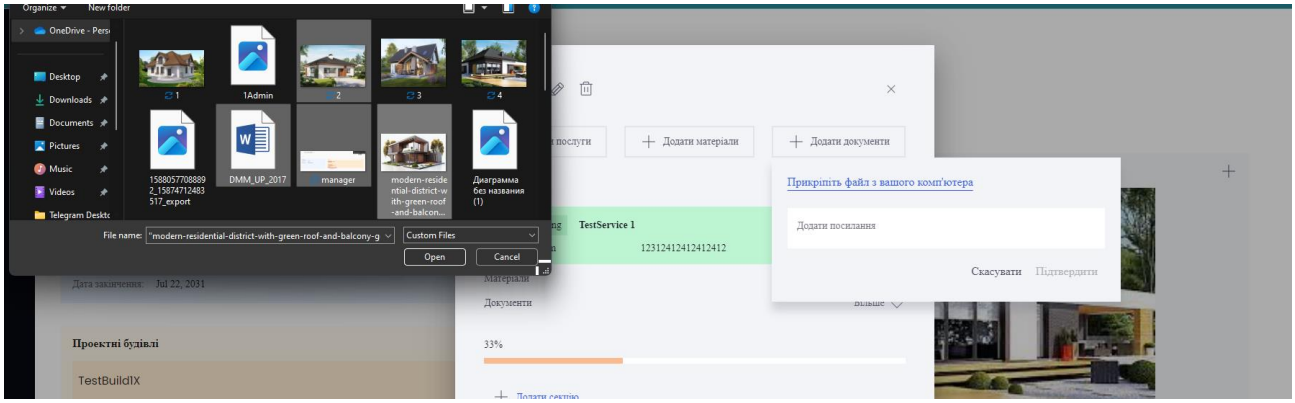


Рисунок 3.56 - Додавання документу

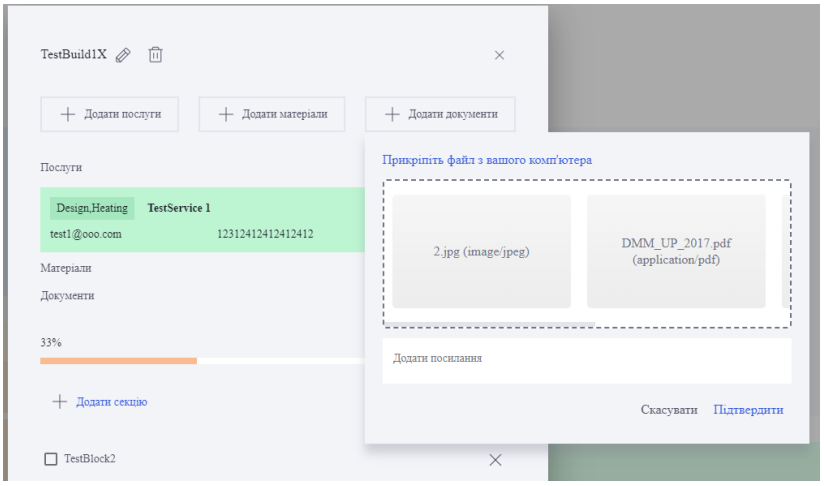
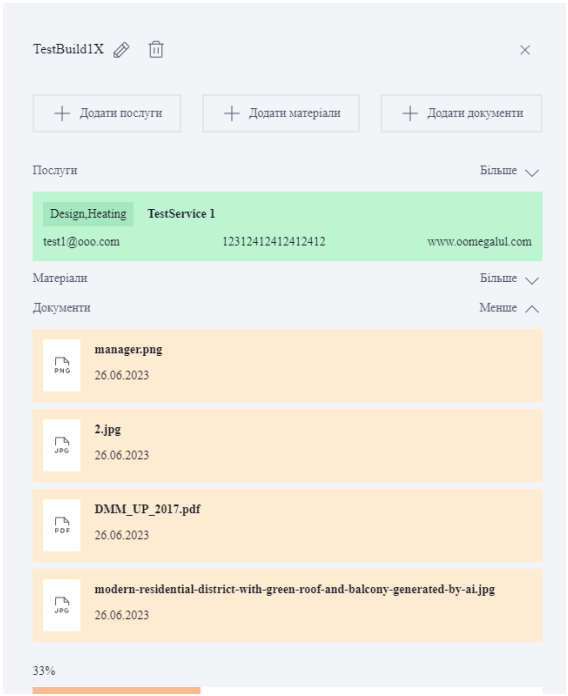


Рисунок 3.57 - Вигляд модального вікна з завантаженими документами



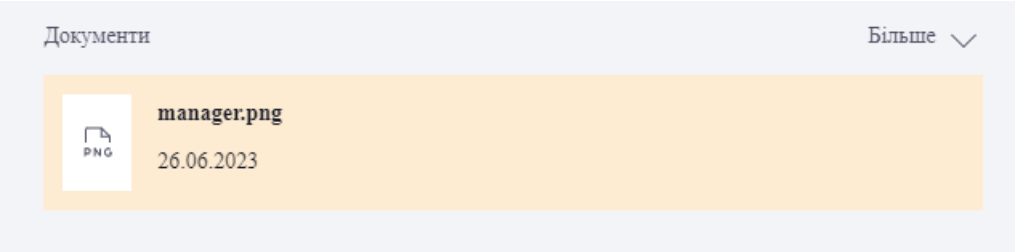


Рисунок 3.58, 3.59 - Результат доданих документів до секції у відкритому та закритому вигляді

Також у вкладці ресурсів у розділі Хронологія ми можемо відстежувати етапи розробки будівлі.

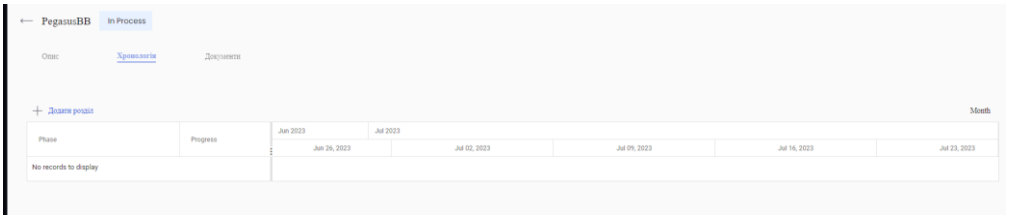


Рисунок 3.60 - Вигляд хронолігії

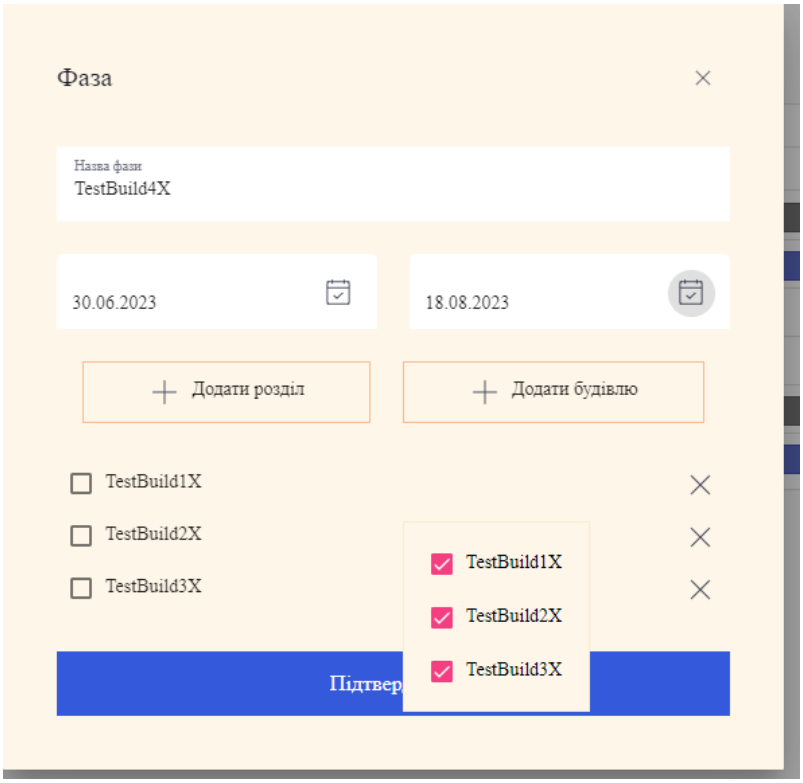
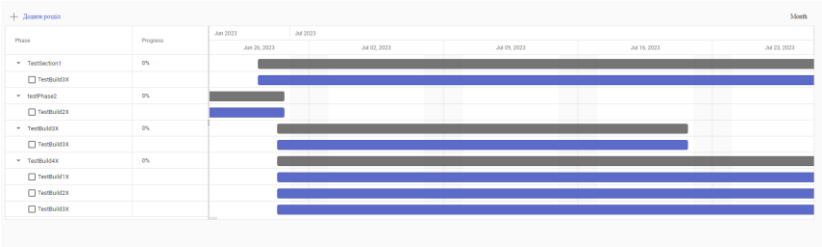


Рисунок 3.61 - Модальне вікно при створенні нової секції з будівлею



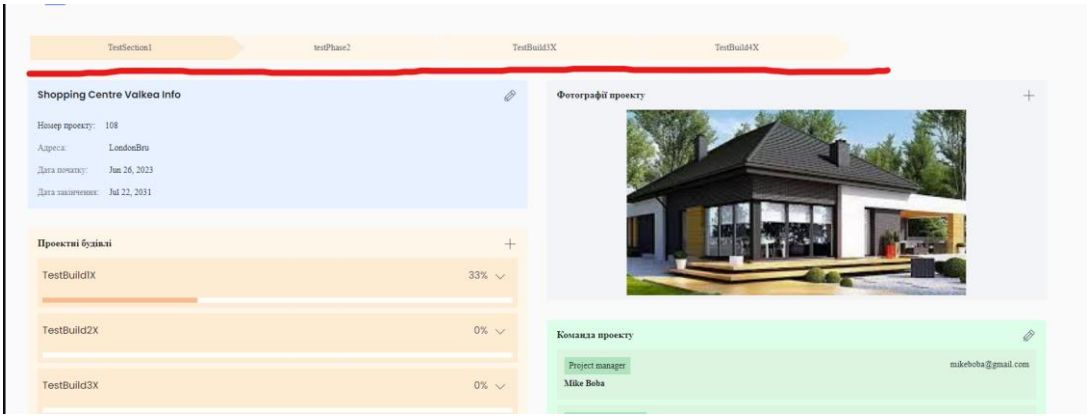


Рисунок 3.62, 3.63 – Секції та їх погрес

10. Вкладка сповіщення містить в собі список змін на проекті.

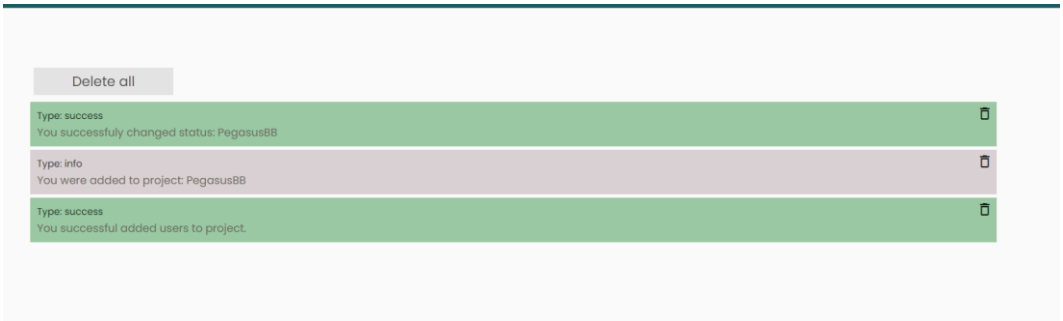


Рисунок 3.64 - Модальне вікно при створенні нової секції з будівлею.

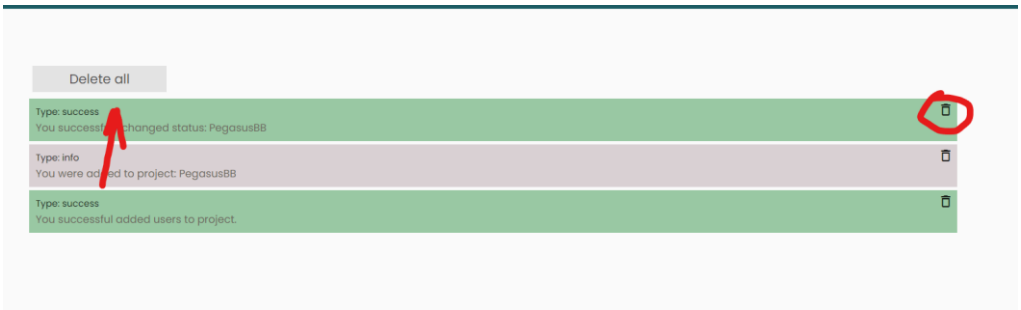


Рисунок 3.65 - Також ми можем очистити весь список нажавши на кнопкувидалити все і очистим список сповіщень.

11. У розділі "Профіль" знаходиться загальна інформація про проект з логотипом компанії, який можна змінити. Тут також є можливість редагувати опис компанії та переглянути всі наявні проекти та їх статуси, які належать компанії.

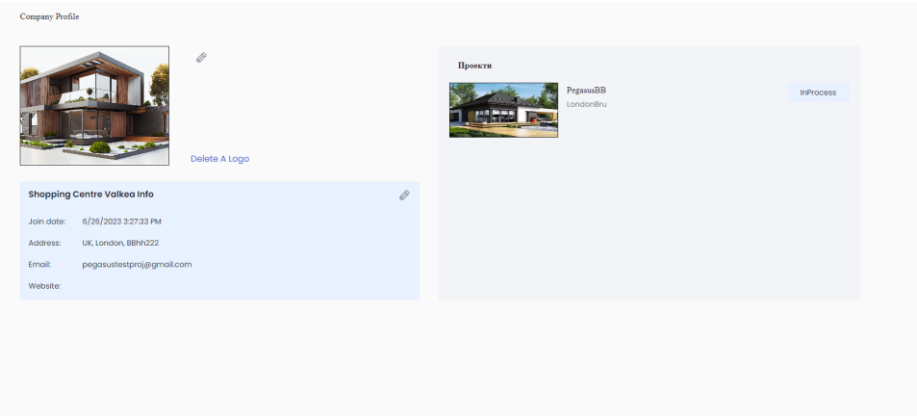


Рисунок 3.66 – Вкладка профіль

На вкладці показано вікно профілю, де доступна Кнопка для очищення списку сповіщень, що призведе до видалення всіх сповіщень зі списку.

ВИСНОВКИ

Під час виконання бакалаврської роботи була розроблена система управління компаніями та працівниками з наступними завданнями:

1. Виконано порівняльний аналіз популярних SaaS-платформ. Кожна платформа має свої унікальні можливості, але для нової системи ключовими характеристиками повинні бути веб-інтерфейс для зручного доступу, модуль для збереження інформації про компанії та працівників, модуль для пошуку та відображення даних про проекти та завдання, а також модуль для збору результатів співпраці.
2. Вибрано архітектурні компоненти для реалізації системи: база даних PostgreSQL, мова програмування C# для серверної частини, фреймворк .Net Core, мова програмування JavaScript для клієнтської частини, фреймворк Angular і мова процесора Sass для стилів. Використано графічний редактор Figma та середовища розробки Visual Studio Code та Visual Studio 2022.
3. Проведено аналіз функціональних та нефункціональних вимог до системи, побудовано діаграму варіантів використання та діаграму класів для аналізу структури системи.
4. Реалізовано базу даних з 24 таблицями, що забезпечує зручне збереження та колективний доступ до інформації про компанії та працівників.
5. Розроблено алгоритми функціонування системи, визначено порядок взаємодії класів та реалізовано додаток.
6. Реалізовано інтерфейс системи для адміністратора і користувача. Адміністратор має можливість створювати та налаштовувати компанії і додавати нових працівників. Користувач з ролью операційного менеджера може створювати проекти, додавати учасників, переглядати таймлайни прогресу будівництва, вести списки послуг та матеріалів та моніторити стани проектів.

Результати роботи представляють готовий до використання програмний комплекс, який сприяє полегшенню управління компаніями та працівниками. У подальшому розвитку системи можливо впровадження засобів інтелектуального аналізу. Ці вдосконалення дозволять покращити функціональність та ефективність системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. " Офіційна документація Angular:https://angular.io/docs
2. Офіційна документація .NET: https://docs.microsoft.com/en-us/dotnet/
3. Офіційна документація SQL: https://docs.microsoft.com/en-us/sql/
4. Офіційна документація CQRS (Command Query Responsibility Segregation): https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs
5. Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin (книга)
6. "Introduction to JavaScript" на MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
7. "The Complete Guide to Angular Dependency Injection" стаття на Angular University: https://blog.angular-university.io/angular-dependency-injection/
8. "SQL Server Tutorial" на Mode: https://mode.com/sql-tutorial/
9. Офіційна документація NGRX: https://ngrx.io/docs
10. Flux: Application Architecture for Building User Interfaces (Facebook) (документація): https://facebook.github.io/flux/
11. "NGRX Patterns and Techniques" на Ultimate Courses: https://ultimatecourses.com/blog/ngrx-patterns-and-techniques
12. "Introduction to the Theory of Computation" by Michael Sipser: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-045j-automata-computability-and-complexity-fall-2005/
13. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: https://mitpress.mit.edu/sites/default/files/sicp/full-text/book/book-Z-H-4.html

ДОДАТКИ

```
using Ctor.Application.Auth.Queries.ChangeDefaultPassword;
using Ctor.Application.Auth.Queries.ChangePassword;
using Ctor.Application.Auth.Queries.ForgotPassword;
using Ctor.Application.Auth.Queries.KeepDefaultPassword;
using Ctor.Application.Auth.Queries.Login;
using Ctor.Application.Auth.Queries.Logout;
using Ctor.Application.Auth.Queries.RefreshToken;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Ctor.Controllers;

[Authorize(AuthenticationSchemes = "Bearer")]
public class AuthController : ApiControllerBase
{
    [AllowAnonymous]
    [HttpPost("login")]
    public async Task<ActionResult<LoginDto>> Login([FromBody] LoginCommand command)
    {
        return await Mediator.Send(command);
    }

    [HttpPost("logout")]
    public async Task<ActionResult<LogoutDto>> Logout([FromBody] LogoutCommand
command)
    {
        return await Mediator.Send(command);
    }

    [AllowAnonymous]
    [HttpPost("refresh-token")]
    public async Task<ActionResult<RefreshTokenDto>> RefreshToken([FromBody]
RefreshTokenCommand command)
    {
        return await Mediator.Send(command);
    }

    [HttpPost("change-password")]
    public async Task<ActionResult<ChangePasswordDto>> ChangePassword([FromBody]
ChangePasswordCommand command)
    {
        return await Mediator.Send(command);
    }

    [HttpPost("change-default-password")]
    public async Task<ActionResult<ChangeDefaultPasswordDto>> ChangeDefaultPassword(
        [FromBody] ChangeDefaultPasswordCommand command)
    {
        return await Mediator.Send(command);
    }

    [HttpPost("keep-default-password")]
    public async Task<ActionResult<KeepDefaultPasswordDto>> KeepDefaultPassword(
        [FromBody] KeepDefaultPasswordCommand command)
    {
        return await Mediator.Send(command);
    }

    [HttpPost("forgot-password")]
    public async Task<IActionResult> ForgotPassword([FromBody] ForgotPasswordCommand
command)
    {
```

```
        return Ok(await Mediator.Send(command));
    }
```

```
using Ctor.Application.Phases.Commands.AddPhase;
using Ctor.Application.Phases.Commands.DeletePhase;
using Ctor.Application.Phases.Commands.UpdatePhase;
using Ctor.Application.Phases.Commands.UpdatePhaseStep;
using Ctor.Application.Phases.Commands.UpdatePhaseSteps;
using Ctor.Application.Phases.Queries.GetPhasesByProjectId;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace Ctor.Controllers;

[Route("api/phases")]
public class PhaseController : ApiControllerBase
{
    [HttpGet("project/{projectId:long}")]
    public async Task<List<PhaseDto>> GetPhasesByProjectId(long projectId)
    {
        return await Mediator.Send(new GetPhasesByProjectIdQuery(projectId));
    }

    [HttpPost]
    public async Task<IActionResult> PostPhase([FromBody] AddPhaseCommand command)
    {
        await Mediator.Send(command);
        return Ok();
    }

    [HttpPut("{id:long}")]
    public async Task<IActionResult> PutPhase(long id, [FromBody] UpdatePhaseCommand
command)
    {
        if (id != command.Id)
        {
            return BadRequest();
        }

        await Mediator.Send(command);
        return Ok();
    }

    [HttpDelete("{id:long}")]
    public async Task<IActionResult> DeletePhase(long id)
    {
        await Mediator.Send(new DeletePhaseCommand(id));
        return Ok();
    }

    [HttpPut("steps")]
    public async Task<IActionResult> EditPhaseSteps([FromBody]
UpdatePhaseStepsCommand command)
    {
        await Mediator.Send(command);
        return Ok();
    }

    [HttpPut("steps/{id:long}")]
    public async Task<IActionResult> EditPhaseStep(long id, [FromBody]
UpdatePhaseStepCommand command)
    {
        await Mediator.Send(command);
        return Ok();
    }
}
```