

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:

«Система трекінгу процесу будівництва»

Виконав студент групи ІПЗ-20-4
КОРМИШ Роман Іванович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: Кормиш Роман Іванович

Керівник роботи: Савіцький Роман Святославович

Тема роботи: «Система трекінгу процесу будівництва»,
затверджена Наказом закладу вищої освіти від 23.02.2024р., №74/с

Вихідні дані для роботи: Об'єктом дослідження є використання інформаційних технологій для автоматизації процесів будівельних компаній.

Предметом дослідження є використання технологій розробки вебзастосунків для автоматизації процесів будівельних компаній.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р. С	19.02.2024	19.04.2024
2	Савіцький Р. С	19.02.2024	15.05.2024
3	Савіцький Р. С	19.02.2024	31.05.2024

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «бакалавр» складається з системи трекінгу процесів будівництва для будівельних компаній та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 78 сторінок, 74 ілюстрацій та 18 таблиць. Загальний обсяг 85 сторінок.

Метою роботи є розробка системи трекінгу процесу будівництва для будівельних компаній. Система включає модулі для управління проєктами, рекламацією, працівниками, компаніями, документообігом, відстеження прогресу забудови, а також управління матеріалами і підрядниками.

У роботі поставлені основні завдання на розробку системи. Також було проведено порівняльний аналіз аналогічних систем. Було обрано клієнт-серверну архітектуру, яка працює в зв'язці з патерном CQRS. Приведені основні сценарії використання системи, логіка взаємодії між модулями системи, діаграма класів та алгоритми роботи ключових модулів, спроектована база даних з використанням PostgreSQL. Клієнтська частина розроблена за допомогою фреймворку Angular, а серверна частина мовою програмування C# в середовищі .NET.

КЛЮЧОВІ СЛОВА: БУДІВНИЦТВО, АВТОМАТИЗАЦІЯ, ПРОЄКТ, РЕКЛАМАЦІЇ, КОМПАНІЯ, ДОКУМЕНТООБІГ, УПРАВЛІННЯ ПРОЦЕСАМИ.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система трекінгу процесу будівництва Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Кормиш Р.І.					3	85
Керівник		Савіцький Р.С.				Житомирська політехніка, група ІПЗ-20-4		
Рецензент		Чижмотря О.В.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a construction process tracking system for construction companies and an explanatory note. The explanatory note to the qualification work contains 78 pages, 74 illustrations and 18 tables. The total volume is 85 pages.

The goal of the work is to implement a construction process tracking system for construction companies. The system contains modules for managing projects, claims, employees, companies, document management, tracking construction progress, as well as managing materials and contractors.

The paper sets out the main tasks for the development of the system. A comparative analysis of similar systems was also conducted. A client-server architecture was chosen that works in conjunction with the CQRS pattern. The main scenarios of using the system, the logic of interaction between the system modules, the diagram and algorithms of the key modules are presented, and the database is designed using PostgreSQL. The client side was developed using the Angular framework, and the server side was developed in the C# programming language in the .NET environment.

KEY WORDS: CONSTRUCTION, AUTOMATION, PROJECT, CLAIM, DOCUMENT MANAGEMENT, PROCESS MANAGEMENT, COMPANY.

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Вибір архітектури системи управління компаніями	14
1.4. Обґрунтування вибору інструментальних засобів.....	15
Висновки до першого розділу	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ТРЕКІНГУ ПРОЦЕСУ БУДІВНИЦТВА.....	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	19
2.2. Розробка бази даних системи	24
2.3. Проєктування та реалізація алгоритмів роботи системи	29
2.4 Реалізація системи трекінгу процесів будівництва.....	34
Висновки до другого розділу	43
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	45
3.1 Порядок встановлення та налаштування системи	45
3.2 Структура інтерфейсу та порядок взаємодії з системою трекінгу процесів будівництва	46
3.3. Тестування роботи програмного продукту	68
Висновки до третього розділу.....	74
ВИСНОВКИ	75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	76
ДОДАТКИ.....	78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

API – Application Programming Interface

CQRS – command and query responsibility segregation

JSON – JavaScript Object Notation

JWT – JSON Web Token

RESTFull API – representational state transfer API

SPA – single page application

ВСТУП

Актуальність теми. Сьогодні будівельна галузь стикається з низкою викликів, таких як складність управління проектами, контроль якості, ефективність використання ресурсів та суворе дотримання термінів. У зв'язку з цим, автоматизовані системи стають все більш актуальними. Впровадження сучасних інформаційних технологій дозволяє значно підвищити ефективність будівельних проектів, знизити витрати та мінімізувати ризики.

Одним із ключових аспектів, що сприяють досягненню цих цілей, є розробка системи трекінгу процесів будівництва. Така система дозволяє в режимі реального часу відслідковувати прогрес виконання робіт, отримувати актуальну інформацію, вчасно виявляти та виправляти проблеми, що виникають на різних етапах будівництва. Подібні системи також сприяють покращенню комунікації між учасниками проекту та забезпечують більшу прозорість процесів.

Більше того, системи трекінгу процесів будівництва допомагають економити час як співробітників так і клієнтів. Реальний час доступу до даних проекту, централізовані канали комунікації та інструменти спільної роботи сприяють безшовному обміну інформацією та прийняттю рішень. Незалежно від того, чи є це менеджери, інженери чи бригадири, всі учасники проекту можуть залишатися синхронізованими, що допомагає будівельному проекту бути успішним.

Метою кваліфікаційної роботи є розробка системи для трекінгу та управління проектами будівельних компаній задля підвищення ефективності керування процесами будівельних компаній, зменшення ризиків та пришвидшення виконання робіт на об'єктах.

Основними завданнями кваліфікаційної роботи є:

1. Розробити серверну частину сайту з використання фреймворків .NET [1] та MediatR [2].

2. Розробити клієнтську частину сайту з використанням Flux [3] архітектури та NgRx [4], використовуючи фреймворк Angular [5].

3. Реалізувати наступні ключові модулі:

- компаній;
- користувачів;
- рекламації;
- проєктів;
- документообігу;
- матеріалів;
- часового графіку.

4. Протестувати основні потоки виконання на основі тестових даних.

Об’єктом дослідження є система для трекінгу процесів будівельних компаній, яка надає можливості управління проєктами, відслідковування процесів будівництва, документообігу та рекламації.

Предметом дослідження є автоматизація процесу будівництва. Дослідження направлене на вивчення технологій та методів, задля забезпечення автоматизованого управління процесами будівельних компаній.

В майбутньому розроблена система може бути ефективно застосована в найрізноманітніших будівельних компаніях для того, щоб спростити управління проєктами, процесами документообігу і рекламації. Також забезпечує легкий доступ до потрібної інформації, покращує комунікацію, зменшує фінансові витрати та зусилля, які необхідні для адміністрування процесів будівельних компаній.

Тези на тему «Необхідність автоматизації менеджменту проєктів будівельних компаній» приведені в CSNT 2024 [6].

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Основна мета роботи є розробка системи трекінгу процесів будівництва, задля підвищення ефективності керування процесами будівельних компаній, зменшення ризиків та пришвидшення виконання робіт на об'єктах.

Система трекінгу процесу будівництва має складатися з деяких ключових програмних модулів, а саме:

- Модуль компаній. Цей модуль дозволяє адміну створювати компанії в системі, а членам компаній редагувати та додавати інформацію про компанію.
- Модуль проєктів. Цей модуль дозволяє компаніям створювати сторінки своїх будівельних проєктів, вказуючи деталі, такі як опис, бюджет, статус тощо. Він надає можливість відстежувати та оновлювати прогрес кожного проєкту.
- Модуль матеріалів. Цей модуль дозволяє компаніям додавати інформацію про матеріали, які використовуються у проєктах, такі як назва, ціна, міра тощо. Він надає можливість відстежувати наявність, поставки та використання матеріалів.
- Модуль постачальників. Цей модуль дозволяє компаніям додавати інформацію про своїх постачальників та підрядників з якими воно заключили договори і до кожного проєкту необхідних постачальників.
- Модуль документообігу. Цей модуль дозволяє генерувати звіти про проєкти, використані матеріали та сервіси, а також переглядати документи та звіти відповідних проєктів.
- Модуль рекламації. Дозволяє керувати скаргами про недоліки проєктів з розподіленням на статуси з відповідним наданням можливостей по управлінню відносно наявної ролі користувача.

- Модуль таймлайну. Цей модуль дозволяє компаніям створювати та відстежувати календарні плани для своїх проєктів, відображаючи ключові віхи, завдання та терміни. Він надає можливість проводити аналіз, планувати та прогнозувати хід виконання робіт.

Основними етапами розробки платформи є:

1. Розробка дизайну вебсайту:
 - створення загальної концепції дизайну додатку;
 - створення UI компонентів;
 - розробка дизайну вебсторінок.
2. Побудова архітектури серверної та клієнтської частин системи.
3. Розробка основного функціоналу:
 - створення логіки авторизації користувачів з відповідними ролями;
 - створення віддаленої бази даних для зберігання інформації;
 - імплементація модулів бізнес логіки;
 - інтеграція нотифікацій.
4. Тестування додатку:
 - функціональне тестування додатку;
 - тестування графічного інтерфейсу.

Результатом реалізації поставленого завдання є веборієнтована система для трекінгу процесів будівництва, яка включає багатокористувацький клієнтський додаток, сервер для обробки та зберігання даних у базі та ключові модулі для автоматизації процесів будівельних компаній, такі як управління проєктами, документообігом та реклаमाцією.

1.2. Аналіз аналогів програмного продукту

Основна мета платформи для трекінгу процесів будівництва полягає в покращенні управління проєктами шляхом забезпечення централізованого моніторингу, наявності єдиного «джерала правди», яким слугує інформація

занесена в систему, та низки інших важливих функцій, таких як відслідковування процесів будівництва, розподілення матеріали, управління скаргами та дефектами та іншими. Платформа підвищує прозорість проєкту, дозволяє контролювати прогрес, інтегрується з іншими системами управління за допомогою поширених форматів звітності, що покращує якість будівництва, забезпечуючи своєчасне прийняття обґрунтованих рішень. Відносно цих критерій розглянемо аналоги системи для дослідження ринку (табл. 1.1).

Таблиця 1.1

Перелік аналогів системи

Назва	Розробник	Платформа	СУБД	Умови ліцензії
BAS Будівництво. ERP	Kamala Soft	кросплатформена	MS SQL	Різні пакети, включаючи безкоштовний та платні.
Dedal Construction Project Management Software	Dedalsoft	кросплатформена	MySQL	Умови ліцензії доступні онлайн.
UGLA	UGLA	кросплатформена	PostgreSQL	Різні типи ліцензій для користувачів

Розглянемо аналоги детальніше:

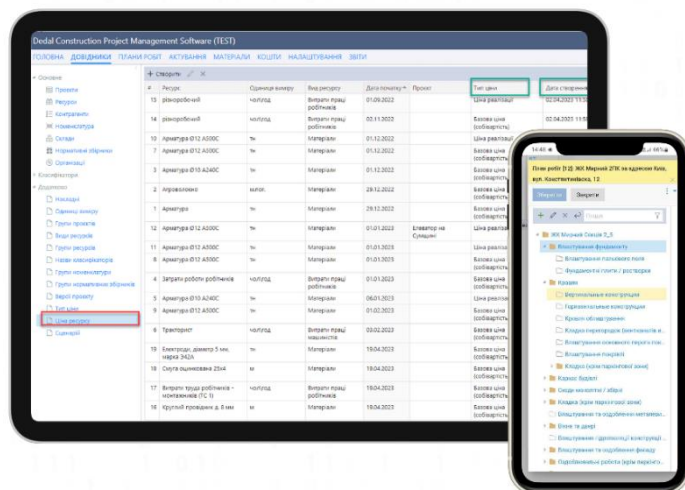


Рис 1.1 Dedal Construction Project Management Software

Демонстраційний режим: Монітор цільових показників

Позиція:

Виконання планів

- Виконання плану продажів / % / 31.03.2023: **95 %**
- Виконання плану купівель / % / 31.03.2023: **87 %**
- Виконання плану виробництва / % / 31.03.2023: **42 %**

Грошові кошти

Залишок ГК в упр. валюті / тис. / USD / 25.03.20...

План-факт надходжень ГК / 25.03.2023 - 31.03.2...

План-факт списання ГК / тис. / USD / 25.03.2023 - 31.03...

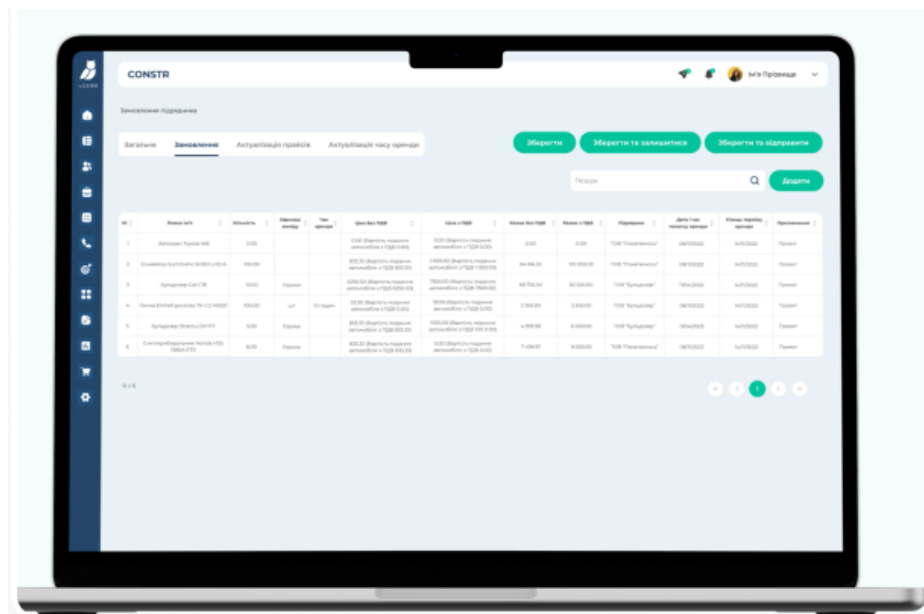
Фінанси

Простроч. борги клієнтів / тис. / USD / 25.03.20...

Простроч. борги постачальникам / тис. / USD / ...

Запаси / тис. / USD / 25.03.2023 - 31.03.2023

BAS Будівництво. ERP [8] (рис 1.2) – рішення, яке забезпечує автоматизацію всіх основних бізнес-процесів в апараті управління підприємств, а також автоматизацію всіх видів будівельної діяльності, зокрема, для вертикально-інтегрованих холдингів, які реалізують повний цикл будівництва.



UGLA [9] (рис 1.3) – система, яка модернізує будівельну галузь не тільки за допомогою впровадженні хмарних програмних ERP продуктів, але й

імплементатії інноваційних технологій, які можуть ефективно замінити людську працю за допомогою інтелектуального обладнання.

Порівняння функціональних можливостей продуктів приведено нижче (табл. 1.2).

Таблиця 1.2

Функціональні можливості

	BAS	Dedal	UGLA	Architector
Наявність зручного вебінтерфейсу	Ні	Ні	Так	Так
Доступ з будь-якої точки світу	Ні	Так	Так	Так
Керування матеріалами та постачальниками	Ні	Ні	Так	Так
Управління реклаमाцією	Ні	Ні	Ні	Так
Звітність	Ні	Так	Так	Так
Доступність для малого бізнесу	Ні	Ні	Ні	Так

У кожної з представлених платформ є свої особливості та унікальні функції, які можуть задовольнити потреби різних будівельних компаній. UGLA пропонує не тільки автоматизацію процесів виробництва, а й нових перевірених постачальників матеріалів; Dedal допомагає налаштувати пряму комунікацію між інвесторами, підрядниками та замовниками; Kamala BAS надає великий спектр пакетів для широкої аудиторії підприємств.

Отже, основними характеристиками нової системи трекінгу процесів будівництва мають бути:

- наявність вебінтерфейсу задля забезпечення зручності системного доступу;

- модуль для автоматизації роботи з компаніями та проєктами;
- можливість користуватися з будь-якої точки світу;
- автоматизація роботи з документацією та рекламацією;
- модуль для пошуку та перегляду даних про проєкти, компанії, доступні матеріали та послуги та їх потреби в конкретній будівлі;
- можливість планувати та відстежувати процес будівництва, використовуючи шкали часу та графіки.

1.3. Вибір архітектури системи трекінгу процесів будівництва

Для розробки системи трекінгу процесів будівництва було обрано спроектувати додаток на основі клієнт-серверної архітектури, клієнтські програми взаємодіють із сервером для отримання та обробки необхідних даних.

Клієнт-серверна архітектура базується на концепції розподілу ролей між двома основними компонентами клієнтом і сервером. Клієнт відповідає за вебінтерфейс, який бачить та використовує користувач, та збирає користувацькі дані для надсилання та обробки на сервері. В свою чергу сервер аналізує, валідує, забезпечує виконання певної бізнес-логіки та зберігає дані в базі даних (рис 1.4).

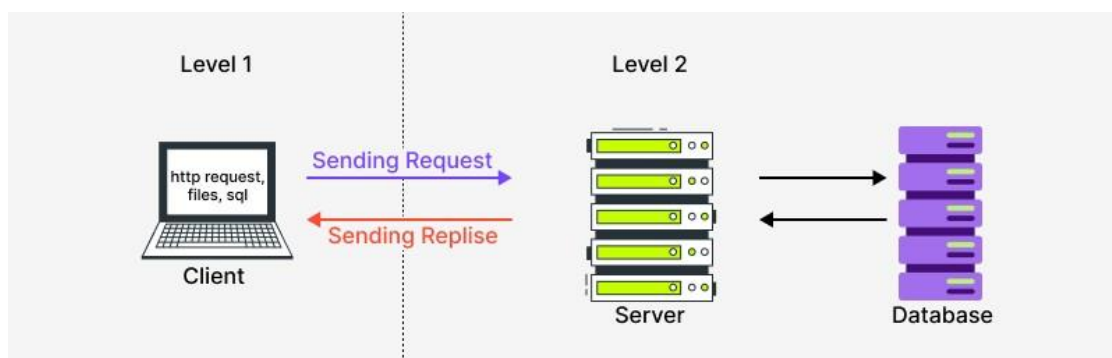


Рис 1.4 Клієнт-серверна архітектура

Платформа для трекінгу процесу будівництва побудована на концепції клієнт-серверної архітектури використовуючи додатково патерн CQRS [10].

Серверна модель слугує для реалізації бізнес-логіки системи трекінгу процесу будівництва. Сервер приймає запити від клієнтів, такі як додавання нового проєкту, оновлення інформації про проєкт або отримання даних про стан

будівництва, додання скарги чи зміна її статусу, тощо. Також сервер взаємодіє з базою даних для збереження, оновлення та видалення даних про відповідні сутності системи.

Клієнтське представлення слугує для представлення даних користувачу та забезпечує інтерфейс для взаємодії з системою трекінгу процесу будівництва. Вебсторінки відображають користувачу дані про проекти, їх стан, прогрес, проблеми на будівлях, документи тощо. Також клієнтська частина надсилає запити до сервера для отримання або обробки даних, таких як оновлення статусу проєкту або додавання нових вимог чи скарг.

На серверній стороні також використовується патерн Медіатор, який забезпечує та ізолює комунікацію між модулями бізнес-логіки та представлення. Медіатор слугує посередником, що отримує запити від контролера та передає необхідні дані модулю з бізнес-логікою. Застосування даної архітектури зменшує зв'язність між компонентами та спрощує розширення та підтримку системи.

1.4. Обґрунтування вибору інструментальних засобів

Для розробки системи було обрано такі технології:

- мова розмітки гіпертекстових документів: HTML5 [11];
- мова препроцесора для стилів: Sass [12];
- мова програмування: JavaScript [13];
- фронтент фреймворк: Angular.

Для серверної частини було обрано наступні технології:

- мова програмування C# [14];
- бекенд фреймворк ASP.NET Core [15];
- база даних: PostgreSQL [16].

Для визначення бази, яка відповідає потребам системи, проведемо порівняння декількох СКБД. Почнемо з порівняння максимального обсягу даних, які можливо зберегти (табл. 1.3).

Таблиця 1.3

Максимально доступний обсяг збережених даних для кожної СКБД

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

Проведемо аналіз за допомогою порівняння методів захисту даних, що можуть бути використані, щоб запобігти несанкціонованому доступу до важливих даних в інформаційній системі компанії. Порівняння алгоритмів забезпечення безпеки даних в аналогах приведено нижче (табл. 1.4).

Таблиця 1.4

Аналіз алгоритмів забезпечення безпеки даних в аналогах

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

Після проведення аналізу систем управління базами даних таких як MS SQL, PostgreSQL та MySQL, було визначено, що найкраще підходить для виконання поставлених завдань саме PostgreSQL, оскільки він є безкоштовною базою даних, що дозволить використовувати дані систему малому бізнесу. А також поточна версія PostgreSQL пропонує широкий спектр функцій, що включає підтримку легкого розширення бази даних та надає низку механізмів та алгоритмів захисту даних. Детальне порівняння показників приведено нижче (табл. 1.5).

Аналіз показників

	MySQL	MS SQL	PostgreSQL
Логічна модель даних	Реляційна	Реляційна	Постреляційна (об'єктно-реляційна)
Мови маніпулювання	SQL	QBE, XQuery, SQL,	Розширений SQL
Транзакції	Так	Так	Так
Тригери, процедури, що зберігаються	Так	Так	Так
Платформи	Linux, MacOS X, Windows, Unix	Windows	Windows, Solaris, MacOS, Linux, FreeBSD,
Область застосування	Малий та середній бізнес	Великий бізнес	Великий, середній та малий бізнес
Особливості	Можливість об'єднання баз даних.	Розширена підтримка XML	Низька ціна підтримки

При розробці використовувалися інтегровані середовища розробки та програмні продукти: середовища розробки Visual Studio Code та Visual Studio 2022, а також графічний редактор Figma.

Висновки до першого розділу

У даному розділі було виконано постановку задачі на кваліфікаційну роботу та визначено функціональні вимоги до системи. Проведено дослідження ринку та аналогів, завдяки чому було визначено основні функціональні можливості системи: автоматизований процес управління будівельними проєктами та компаніями, зручний користувацький інтерфейс, звітність, керування матеріалами, постачальниками і рекламацією та доступність для малого бізнесу.

Також було продемонстровано, обгрунтовано та доведено вибір архітектури для розробки системи. Для розробки клієнтської частини було використано мову програмування JavaScript, мову гіперрозмітки HTML, процесор для стилів Sass та фреймворк Angular. Для серверної частини – мову програмування C#, фреймворк ASP.NET Core, а в якості бази даних PostgreSQL.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ТРЕКІНГУ ПРОЦЕСУ БУДІВНИЦТВА

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

Для забезпечення функціоналу трекінгу процесів будівництва, система включає модулі для автоматизації управління проєктами, рекламацією та документообігом, з можливістю генерувати звіти. Також надає можливості для зберігання даних про проєкти, матеріали, компанію, рекламації, звіти, матеріали та будівлі, що підвищує ефективність керування будівельною компанією та автоматизує процеси.

Внутрішні користувачі

Внутрішні користувачі мають різні обов'язки відповідно до їх ролей, що приведено нижче (табл. 2.1).

Таблиця 2.1

Внутрішні користувачі

Актор	Короткий опис
Адмін	Управляє всією системою та може додавати користувачів.
Оперативний менеджер	Може створювати проєкти, управляти інформацією про компанію, запрошувати інших працівників та управляти проєктами.
Менеджер проєкту	Має доступ тільки до тих проєктів, на які його призначили, керує фазами та елементами забудови, може додавати постачальників та ресурси.

Головний інженер	Має доступ до проєктів, які він додав, має можливість позначати елементи забудови.
Бригадир	Оперативно отримує інформації про зміни на проєкті і керує бригадою працівників.

На (рис. 2.1) зображено побудовану діаграму базових варіантів використання відповідно до кожної ролі.

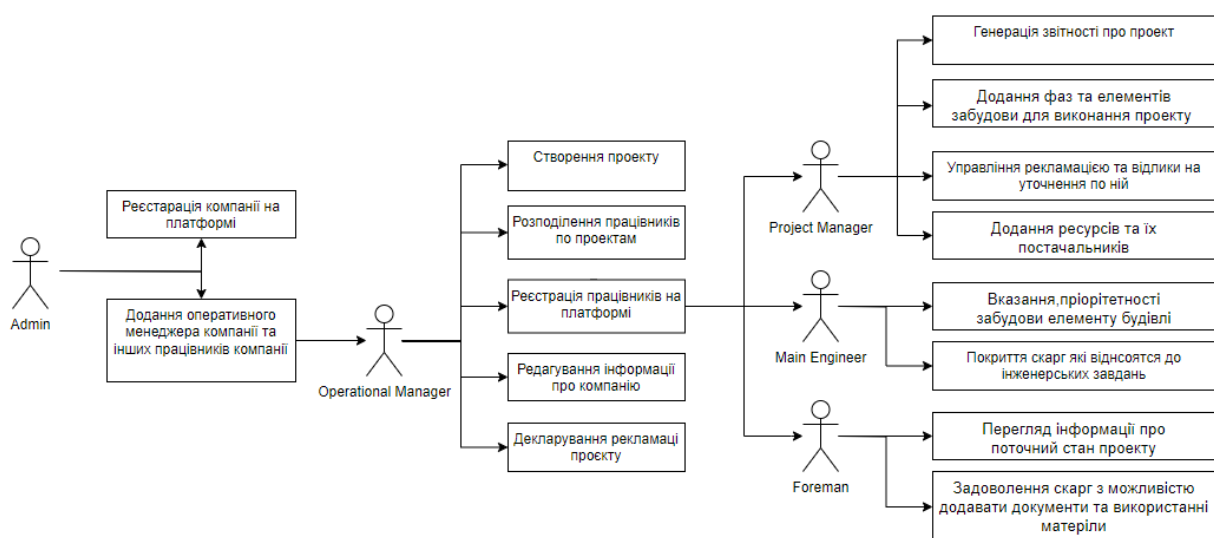


Рис 2.1 Діаграма базових варіантів використання

Специфікація основних прецендентів для кожної ролі приведена в додатку А.

Основними функціями внутрішніх користувачів, які мають ролі менеджерів, є такі:

1. Можливість додавання, редагування всієї інформації про проєкт, розподілення матеріалів та постачальників по відповідним будівлям проєктів.
2. Можливість розробки плану забудови з вказання фаз і строків, до яких певна частина проєкту має бути виконана.

3. Можливість генерувати звітність з використаних матеріалів, а також звітність про сам проєкт, для інтеграції з іншими системами компанії або інших цілей.
4. Можливість декларувати скарги з подальшим процесом перевірки виконання на різних етапах.
5. Можливість додавати інших виконавців на проєкти і давати доступи на платформу, за допомогою листів на електронну пошту.

Основними функціями внутрішніх користувачів, які мають ролі бригадира або інженера, є такі:

1. Можливість перегляду інформації про проєкт та план забудови.
2. Можливість генерувати звіти з необхідними даними про проєкт.
3. Можливість відкликатися на скарги та задовольняти їхні потреби, з вказання інформації про виконання і комунікацією з менеджментом через відповідний функціонал повідомлень і статусів.

Функціональні вимоги:

Система трекінгу процесу будівництва надає користувачам наступні можливості:

1. Додавання, перегляд, фільтрування та редагування проєктів компаній в системі.
2. Створення та перегляд планів забудови для відслідковування необхідних дедлайнів певних секцій відповідного об'єкту.
3. Управління реклаमाцією з можливістю створення, просування, редагування, додавання документів та використаних матеріалів для покриття вимог відповідної скарги, з повним контролем комунікації між менеджментом і виконавцями.
4. Додавання та перегляд звітності та документів відповідних проєктів з можливістю подальшого використання звітності в інших системах.
5. Редагування інформації про компанію.
6. Керування ресурсами і постачальниками з можливістю розподілення їх по проєктам.

Нефункціональні вимоги:

- для звичайних користувачів час навчання складає 1-2 години, а для досвідчених - 1 година;
- відповідь на звичайні запити має складати до 1 секунди, а на складніші до 3 секунд;
- стійкість до збоїв та стабільна робота при високому навантаженні;
- від загального часу роботи час обслуговування не більше 10%;
- максимальна кількість помилок має бути 1 на 1000 запитів користувача;
- продуктивність системи повинна забезпечувати швидкість відгуку, ефективну обробку даних, масштабованість, економічне використання ресурсів, оптимізацію запитів, паралельну обробку, надійність, ефективну систему управління ресурсами, швидкість відгуку 1-3 секунди або менше, оптимальне використання ресурсів і надійну роботу. Крім того, не менш важливо, щоб система була масштабованою і дозволяла обробляти великі обсяги даних і працювати з декількома користувачами;
- система повинна мати інтуїтивно зрозумілий інтерфейс.

Для розробки системи були використані наступні технології: фреймворки .NET Core та Angular і шаблон CQRS (MediatR). Можемо виділити наступні сутності, які зображені на діаграмі нижче, виходячи з аналізу вимог функціональності, що забезпечують реалізацію системи трекінгу процесів будівництва. Об'єктно-орієнтована структура системи у вигляді діаграми класів приведена нижче (рис. 2.2), яка відображає зв'язки між сутностями.

залишати скарги та надавати можливість працівникам компанії виправляти їх з відслідковуванням статусів.

Клас Report є сутністю для збереження звітності в системі, яка надає можливості генерувати звітність про проєкт, використані матеріали та сервіси.

2.2. Розробка бази даних системи

Дані в системі зберігаються в базі даних PostgreSQL, яка складається з 27 таблиць. Основними з них є 9, які забезпечують виконання основного функціоналу системи (табл. 2.2).

Таблиця 2.2

Інформація про таблиці бази даних.

Назва таблиці	Призначення
User	Таблиця користувачів
Company	Таблиця компаній
Project	Таблиця проєктів
Document	Таблиця документів
Report	Таблиця звітності
Claim	Таблиця рекламації
ClaimResponse	Таблиця відклику на регламацію
Building	Таблиця будівель
Material	Таблиця матеріалів
Vendor	Таблиця постачальників

Структура бази даних висвітлена на (рис 2.3).

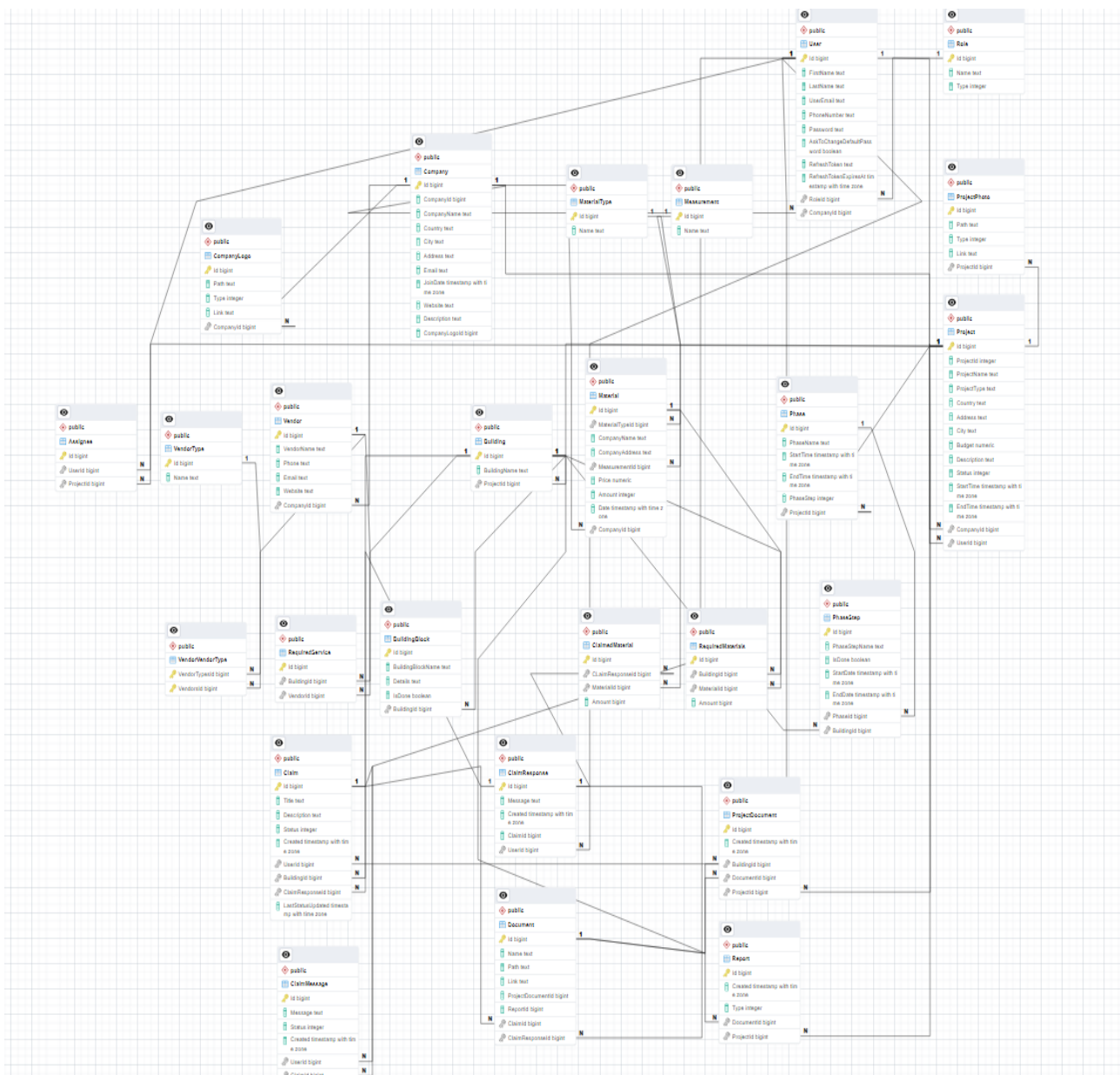


Рис 2.3 Структура бази даних

Таблиця «User» слугує для збереження даних користувача. Опис полів таблиці наведений нижче (табл. 2.3).

Таблиця 2.3

Опис 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
CompanyId	bigint	-	Company.Id	Номер компанії

Таблиця «Company» слугує для зберігання даних компанії. Опис полів таблиці наведений нижче (табл. 2.4).

Таблиця 2.4

Опис 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	-	-	Вебсайт
Description	text	-	-	Опис
CompanyLogoId	bigint	-	CompanyLogo.Id	Номер логотипу компанії

Таблиця «Project» слугує для збереження даних про проекти. Опис полів таблиці наведений нижче (табл. 2.5).

Таблиця 2.5

Опис 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 «Document» слугує для збереження документів. В дану таблицю вводяться дані про ім'я та шлях, по якому лежить документ. Опис полів таблиці наведений нижче (табл. 2.6).

Таблиця 2.6

Опис 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 проєкту

Таблиця «Report» слугує для збереження звітності. Опис полів таблиці наведений нижче (табл. 2.7).

Таблиця 2.7

Опис таблиці «Report»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Type	int	-	-	Тип звіту
ProjectId	bigint	-	ProjectId	Номер проєкту
DocumentId	bigint	-	DocumentId	Номер документа

Таблиця «Claim» слугує для збереження скарг. Опис полів таблиці наведений нижче (табл. 2.8).

Таблиця 2.8

Опис таблиці «Claim»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Title	text	-	-	Назва
Description	text	-	-	Опис
Status	int	-	-	Статус
Created	timestamp	-	-	Дата створення
LastStatusUpdated	timestamp	-	-	Дата останньої зміни статусу
UserId	bigint	-	UserId	Номер юзера
BuildingId	bigint	-	BuildingId	Номер будівлі
ClaimResponseId	bigint	-	ClaimResponseId	Номер відповіді на вимогу

Таблиця «ClaimResponse» слугує для збереження відгуків на скарги зі всією необхідною інформацією. Опис полів таблиці наведений нижче (табл. 2.9).

Таблиця 2.9

Опис таблиці «ClaimResponse»

Назва	Тип даних	PK	FK	Опис поля
Id	bigint	+	-	Номер
Message	text	-	-	Повідомлення про виконання
Created	timestamp	-	-	Дата створення
UserId	bigint	-	UserId	Номер юзера
ClaimId	bigint	-	ClaimId	Номер вимоги

Таблиця «Building» слугує для зберігання інформації про споруди проєкту. Назва будівлі вводиться в цю таблицю. Опис полів таблиці наведений нижче (табл. 2.10).

Таблиця 2.10

Опис таблиці «Building»

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

Таблиця «Material» слугує для збереження інформації про матеріали. Опис полів таблиці наведений нижче (табл. 2.11).

Таблиця 2.11

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 «Vendor» слугує для збереження даних про виробників. Опис полів таблиці наведений нижче (табл. 2.12).

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	-	-	Вебсайт
CompanyId	bigint	-	-	Номер компанії

Використовуючи такі таблиці як User, Project і Company, було побудовано логіку для зберігання основної інформації про будівельні компанії та їх проекти, з розподіленням доступів відповідно до ролей користувачів. Використовуючи такі таблиці як Building, Material і Vendor, було реалізовано функціонал для збереження інформації про будівлі. Використовуючи таблиці Document і Report, реалізовано функціонал для генерації звітності та зберігання в системі файлів. А також реалізовано модуль реклаमाції за допомогою таблиць Claim та ClaimResponse.

2.3. Проєктування та реалізація алгоритмів роботи системи

Так як обраної архітектурою програмного додатку є клієнт-серверна, то вебзастосунок має явне розподілення на серверну та клієнтську частину і передбачає використання підходу RESTful API [17].

RESTful API є набором правил та протоколів, які дозволяють різним програмним додаткам взаємодіяти між собою через мережу.

Можна виділити наступні основні характеристики та принципи RESTful API:

- RESTful API базується на концепції ресурсів, кожен з яких має унікальну URL-адресу. Клієнти взаємодіють з цими ресурсами за допомогою стандартних HTTP-запитів.
- RESTful API використовує стандартні HTTP-методи для виконання різних операцій над ресурсами:

- GET: Отримання даних;
- POST: Створення нового об'єкту;
- PUT: Повне оновлення сутності;
- PATCH: Часткове оновлення сутності;
- DELETE: Видалення ресурсу.

Таким чином, RESTful API дозволяє клієнтам взаємодіяти з серверними ресурсами через уніфікований інтерфейс HTTP-запитів та отримувати потрібні дані у зручному форматі.

У випадку даної системи, коли клієнтська частина буде робити звернення до закритого API для отримання необхідних даних за допомогою HTTP-запитів. Для цього за допомогою фреймворку ASP.NET сервер буде повертати відповідь на запит у вигляді JSON-об'єкту. На стороні серверу відбувається отримання даних з бази даних, їх обробка та модифікація, відповідно до бізнес логіки методу і вхідних параметрів. Якщо ж користувач надсилає інший запит із користувацького інтерфейсу клієнтської частини, то повторно відбувається запит до серверу вебзастосунку і процес відбувається схожим чином.

На (рис. 2.4) зображено загальний алгоритм комунікації в системі між клієнтом, сервером через HTTP-запити та базою даних.

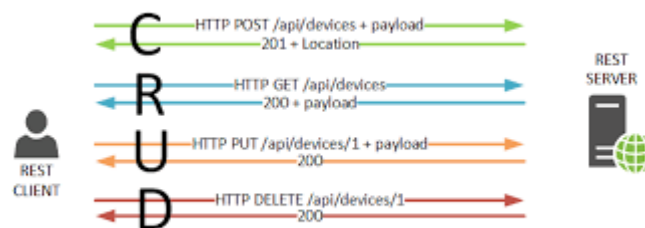


Рис 2.4 Загальний алгоритм комунікації між сервером та клієнтом через RESTfull API

Також, для зображення алгоритму роботи додатку структурну діаграму (рис 2.5), де зображені основні потоки виконання логіки для користувачів з певними ролями.

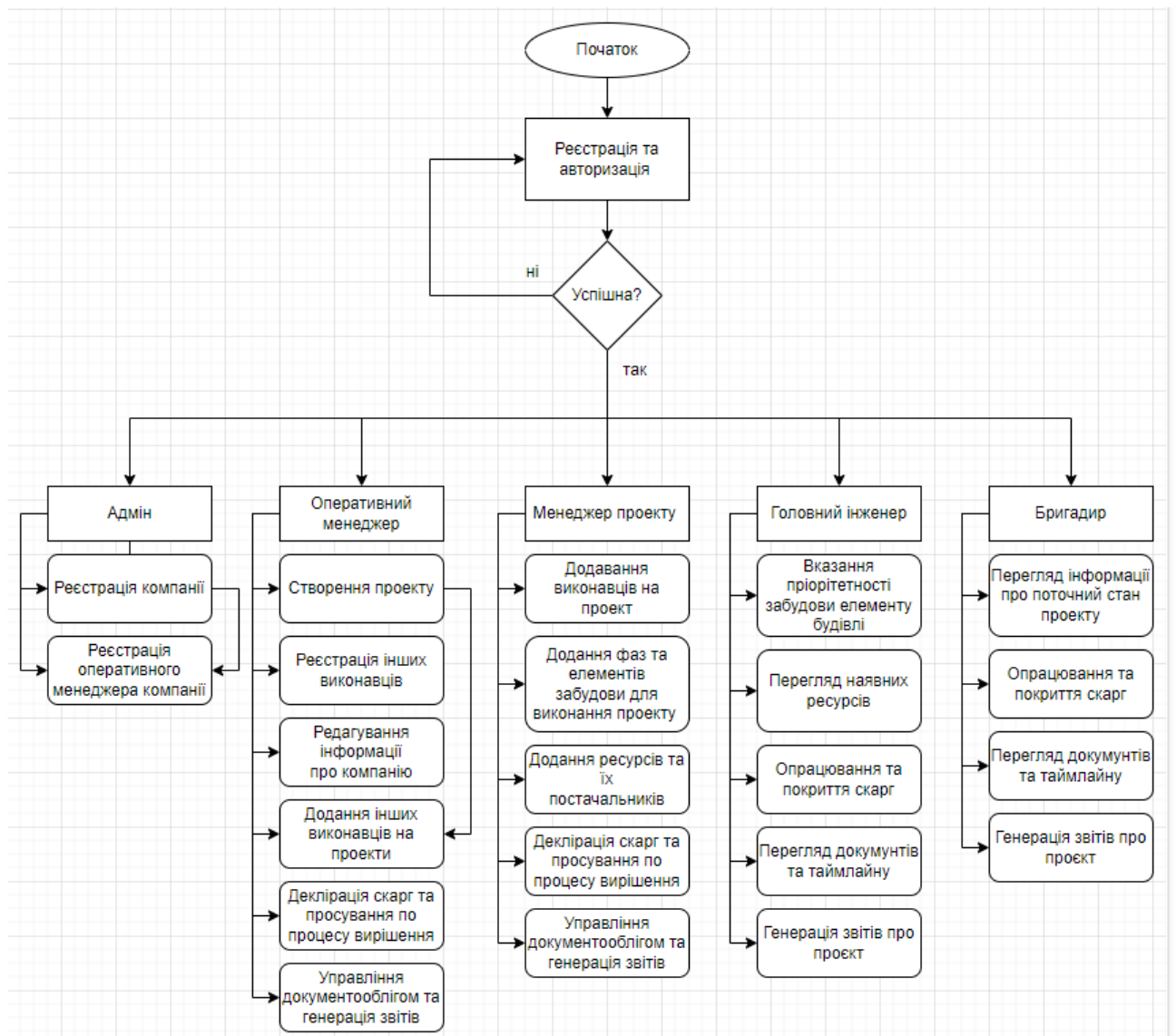


Рис 2.5 Структурна діаграма системи

Опис отриманої діаграми:

- авторизація (так – перехід до використання системи, ні – перехід на початок з повідомленням про необхідність наявності аккаунту).
- отримання прав відповідно ролі;
- отримання доступу до певного функціоналу відносно отриманої ролі. Для адміна це буде реєстрація компанії і оперативного менеджера для неї і тощо;
- кінець роботи вебзастосунку.

Безпосередньо застосунок можна умовно поділити на певні модулі. Для презентативності побудуємо діаграму компонентів (рис 2.6) додатку для декількох модулів.

- Claim – компонент відповідальний за управління скаргами на певні будівлі компаній, з можливістю прикріплення документів та використаних матеріалів, а також комунікації для уточнень.
- Reports – компонент відповідальний за управління документообліком та звітами, надає можливість генерації звітів про проєкт, використані матеріали та сервіси.

Детальніше розглянемо алгоритм роботи модуля для управління скаргами. Для цього було побудовано діаграму послідовності (рис. 2.7).

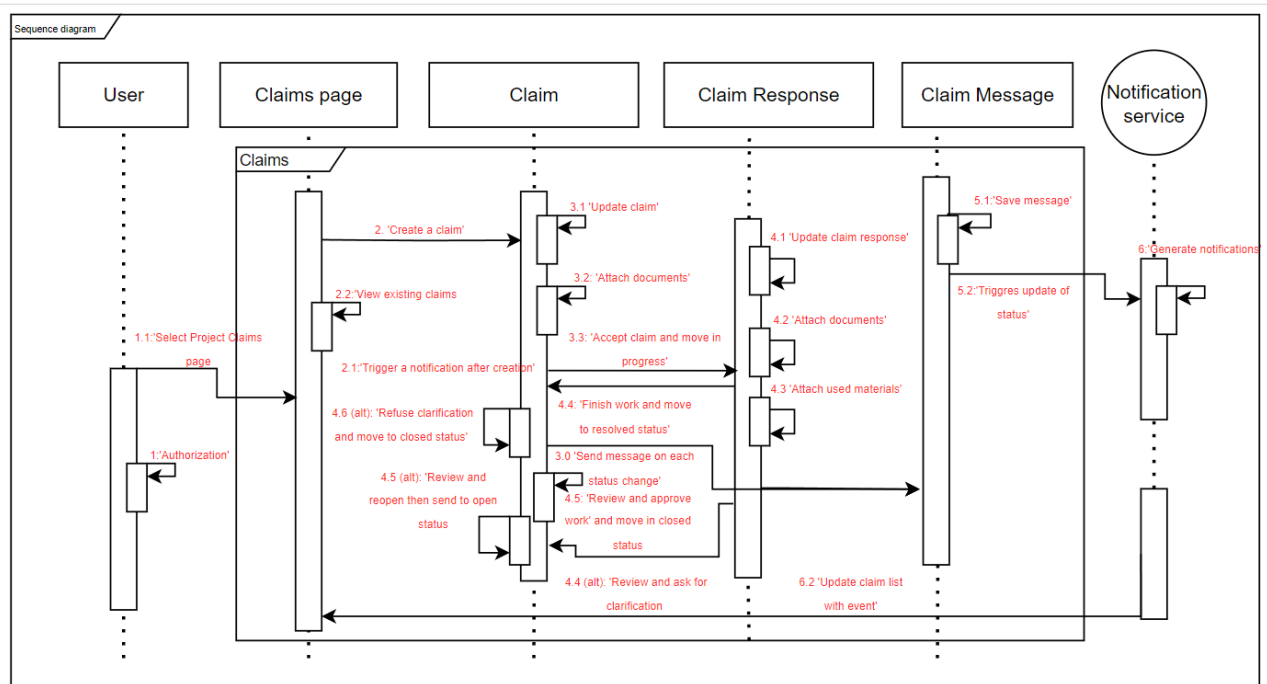


Рис 2.7 Діаграма послідовності для модуля рекламації

На даній діаграмі приведено основний потік виконання при роботі з модулем рекламації. Спочатку користувач авторизується на платформі і потім має можливість перейти на сторінку певного проєкту, де він може обрати вкладку зі скаргами. Далі якщо користувач має роль менеджера, то він може задекларувати скаргу, додатково вказавши документи з поясненнями чи доказами. Також після того як скаргу покрито він перевіряє чи все було виконано, якщо так то переводить в стан завершеної скарги, а якщо ні, то переводить в відкритий стан знову з повідомлення про необхідні правки. Якщо користувач має роль робочого (інженера або бригадира), то він може опрацювати скаргу, перенісши в стан в прогресі, якщо все зрозуміло, або в стан уточнення,

для отримання додаткової інформації. При отриманні оновлень скарга переноситься в стан відкритої знову, а при відмові в стан завершеної і признається невалідною. Якщо є нова інформація зрозуміла, то скарга переводиться в стан в прогресі, де виконавець додає документи, повідомлення та використанні матеріали, а потім переводить в стан вирішеної, яка потребує перевірки від менеджера.

2.4 Реалізація системи трекінгу процесів будівництва.

Розглянемо бізнес логіку одного з модулів системи, а саме модуля рекламаций. Робота з модулем рекламаций починається з створення першої скарги. Тож розглянемо цей функціонал (рис. 2.8 – 2.10)

```
[HttpPost]
0 references
public async Task<ActionResult<ClaimDto>> CreateClaim(
[FromBody] CreateClaimDto claim)
{
    return await Mediator.Send(new CreateClaimCommand(claim));
}
```

Рис. 2.8 Метод контролера CreateClaim

```
2 references
public class CreateClaimDto
{
    2 references
    public string Title { get; set; }
    1 reference
    public string Description { get; set; }
    2 references
    public int BuildingId { get; set; }
}
```

Рис. 2.9 Модель параметрів для створення скарги

```

4 references
public record CreateClaimCommand(CreateClaimDto Claim) : IRequest<ClaimDto>;

1 reference
public class CreateClaimCommandHandler : IRequestHandler<CreateClaimCommand, ClaimDto>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;
    private readonly IDateTime _dateTime;
    private readonly ICurrentUserService _currentUserService;
    private readonly INotificationService _notificationService;

    0 references
    public CreateClaimCommandHandler(
        IMapper mapper,
        IApplicationDbContext context,
        IDateTime dateTime,
        INotificationService notificationService,
        ICurrentUserService currentUserService)
    {
        _mapper = mapper;
        _context = context;
        _dateTime = dateTime;
        _notificationService = notificationService;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<ClaimDto> Handle(CreateClaimCommand request, CancellationToken cancellationToken)
    {
        var date = _dateTime.UtcNow;

        var claim = new Claim
        {
            Title = request.Claim.Title,
            Description = request.Claim.Description,
            Status = ClaimStatus.Open,
            Created = date,
            LastStatusUpdated = date,
            BuildingId = request.Claim.BuildingId,
            UserId = (long)_currentUserService.Id!,
        };

        await _context.Claims.Insert(claim);
        await _context.SaveChangesAsync(cancellationToken);

        var users = await _context.Users.Get(x => x.Assignees.Any(a => a.Project.Building.Any(b => b.Claims.Any(c => c.Id == claim.Id))));
        var userIds = users.Select(x => x.Id).ToList();

        foreach (var userId in userIds.Distinct())
        {
            await _notificationService.UpdateClaims(userId);
        }

        return await _context.Claims.GetById<ClaimDto>(claim.Id, cancellationToken);
    }
}

```

Рис. 2.10 Команда для створення скарги

Поскільки в системі використовується бібліотека MediatR, то коли серверний додаток отримує запит на створення відповідної скарги, він обробляє отримані дані в об'єкт з параметрами і передає в класі команди медіатору, який викликає метод Handle і виконує бізнес логіку створення скарги. В даній команді отримані дані доповнюються необхідною інформацією про поточного користувача і зберігає нову скаргу в базі даних. Також потім надсилається подія через SignalR [18] бібліотеку в сесії інших користувачів, які слухають цю подію і додає цю скаргу на їхні сторінки також.

Далі ми можемо змінити інформацію про новостворену скаргу за допомогою команди на оновлення (рис. 2.11 – 2.13).

```

[HttpPut("update")]
0 references
public async Task<ActionResult<Unit>> UpdateClaim([FromBody] UpdateClaimDto claim)
{
    return await Mediator.Send(new UpdateClaimCommand(claim));
}

```

Рис. 2.11 Метод контролера UpdateClaim

```

- references
public class UpdateClaimDto
{
    - references
    public long Id { get; set; }
    2 references
    public string Title { get; set; }
    2 references
    public string Description { get; set; }
    2 references
    public long BuildingId { get; set; }
}

```

Рис. 2.12 Модель параметрів для оновлення скарги

```

4 references
public record UpdateClaimCommand(UpdateClaimDto Model) : IRequest;

1 reference
public class UpdateClaimCommandHandler : IRequestHandler<UpdateClaimCommand, Unit>
{
    private readonly IApplicationDbContext _context;

    0 references
    public UpdateClaimCommandHandler(IApplicationDbContext context)
    {
        _context = context;
    }

    0 references
    public async Task<Unit> Handle(UpdateClaimCommand request, CancellationToken cancellationToken)
    {
        var claim = await _context.Claims.GetById(request.Model.Id);

        claim.Title = request.Model.Title;
        claim.Description = request.Model.Description;
        claim.BuildingId = request.Model.BuildingId;

        _context.Claims.Update(claim);
        await _context.SaveChangesAsync(cancellationToken);

        return Unit.Value;
    }
}

```

Рис. 2.13 Команда для оновлення скарги

Після отримання даних сервер так само формує команду з наявними даними і оновлює назву, опис і будівлю, до якої прив'язана скарга і зберігає оновлення в базі.

Також користувач може додати документи з доказами або місцем проблеми (рис. 2.14 – 2.15).

```

[HttpPost("{id:long}/documents")]
0 references
public async Task<ActionResult<List<AddClaimDocumentsResponseDto>>> AddClaimDocuments(
    [FromForm] IFormCollection data, long id, CancellationToken ct)
{
    var urls = data["url"].ToArray() ?? Array.Empty<string>();
    var files = new ConcurrentBag<(byte[] Data, string FileName)>();

    await Parallel.ForEachAsync(data.Files, ct, async (file, ct2) =>
    {
        ct2.ThrowIfCancellationRequested();
        files.Add((await file.GetBytes(), file.FileName));
    });

    return await Mediator.Send(new AddClaimDocumentCommand(files.ToArray(), id, urls), ct);
}

```

Рис. 2.14 Метод контролера AddClaimDocuments

```

0 references
public async Task<List<AddClaimDocumentsResponseDto>> Handle(AddClaimDocumentCommand request,
    CancellationToken cancellationToken)
{
    var claim = await _context.Claims.GetById(request.ClaimId, cancellationToken);
    var files = new ConcurrentBag<(byte[] Data, string FileName)>(request.Files);

    await Parallel.ForEachAsync(request.Urls.Where(u => !string.IsNullOrEmpty(u)), cancellationToken,
        async (url, ct) =>
        {
            try
            {
                using var client = new HttpClient();
                using var response = await client.GetAsync(url, ct).ConfigureAwait(false);
                var bytes = await response.Content.ReadAsByteArrayAsync(ct).ConfigureAwait(false);
                files.Add((bytes, url));
            }
            catch (Exception e)
            {
                _logger.LogWarning(e, "Ignoring error an error while downloading file from url '{@Url}'", url);
            }
        });

    var claimDocuments = new List<Document>();

    foreach (var file in files)
    {
        var fileType = Path.GetExtension(file.FileName);
        var path = Path.Combine("claimDocuments", $"{Guid.NewGuid()}{fileType}");

        var link = await _fileManipulatorService.Save(file.Data, path);

        if (link == null)
            throw new IOException();

        var claimDocument = new Document
        {
            ClaimId = request.ClaimId,
            Link = link,
            Path = path,
            Name = file.FileName
        };

        claimDocuments.Add(claimDocument);
    }

    await _context.Documents.AddRangeAsync(claimDocuments);
    await _context.SaveChangesAsync(cancellationToken);

    var response = _mapper.Map<List<AddClaimDocumentsResponseDto>>(claimDocuments);
    return response;
}

```

Рис. 2.15 Команда для додання документів до певної скарги

Після отримання файлів та посилань на файли відбувається перетворення даних в масиви байтів для уніфікації обробки файлів. Після обробки вони зберігаються в системі і інформація про них зберігається в базі. Коли іде звернення до цих файлі, то використовується реляційний шлях всередині самого додатку.

Для відображення користувачу наявних скарг використовується запит (Query), які не модифікують дані, але повертають уже у відповідному форматі, який потрібний клієнтському додатку (рис 2.16 – 2.17).

```
[HttpPut]
0 references
public async Task<ActionResult<List<ClaimDto>>> GetClaims([FromBody] GetClaimsDto model)
{
    return Ok(await Mediator.Send(new GetClaimsQuery(model)));
}
```

Рис. 2.16 Метод контролера GetClaims

```
4 references
public record GetClaimsQuery(GetClaimsDto Model) : IRequest<List<ClaimDto>>;

1 reference
public class GetClaimsQueryHandler : IRequestHandler<GetClaimsQuery, List<ClaimDto>>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;

    0 references
    public GetClaimsQueryHandler(IMapper mapper, IApplicationDbContext context)
    {
        _mapper = mapper;
        _context = context;
    }

    0 references
    public async Task<List<ClaimDto>> Handle(GetClaimsQuery request, CancellationToken cancellationToken)
    {
        var response = await _context.Claims.Get<ClaimDto>(x => x.Building!.ProjectId == request.Model.ProjectId
            && (!request.Model.StartDate.HasValue || x.LastStatusUpdated >= request.Model.StartDate)
            && (!request.Model.EndDate.HasValue || x.LastStatusUpdated <= request.Model.EndDate));

        return response.OrderByDescending(x => x.LastStatusUpdated).ToList();
    }
}
```

Рис. 2.17 Обробник запиту на отримання списку скарг

Даний обробник запиту дістає всі скарги з бази даних для відповідного проєкту, які клієнтський додаток потім використовує для відображення в інтерфейсі користувача.

Після того як користувач як менеджер проєкту створив скаргу її може взяти в опрацювання користувач, який має роль або бригадира або інженера. Для цього після перегляду детальної інформації про скаргу від може перевести її в стан «In

Progress» і вказати повідомлення для менеджера, також під час цього скарга прив'язується до виконавця (рис. 2.18 – 2.23).

```
,
[HttpPost("{id:long}/response")]
0 references
public async Task<ActionResult<CreateClaimResponseDto>> CreateClaimResponse(long id)
{
    return await Mediator.Send(new CreateClaimResponseCommand(id));
}
```

Рис. 2.18 Метод контролера CreateClaimResponse

```
4 references
public record CreateClaimResponseCommand(long ClaimId) : IRequest<CreateClaimResponseDto>;

1 reference
public class CreateClaimResponseCommandHandler : IRequestHandler<CreateClaimResponseCommand, CreateClaimResponseDto>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;
    private readonly IDateTime _dateTime;
    private readonly ICurrentUserService _currentUserService;

    0 references
    public CreateClaimResponseCommandHandler(
        IMapper mapper,
        IApplicationDbContext context,
        IDateTime dateTime,
        ICurrentUserService currentUserService)
    {
        _mapper = mapper;
        _context = context;
        _dateTime = dateTime;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<CreateClaimResponseDto> Handle(CreateClaimResponseCommand request, CancellationToken cancellationToken)
    {
        var date = _dateTime.UtcNow;
        var claim = await _context.Claims.GetById(request.ClaimId, cancellationToken);

        var claimResponse = new ClaimResponse
        {
            Created = date,
            UserId = (long)_currentUserService.Id!,
            ClaimId = claim.Id,
            Message = ""
        };

        claim.ClaimResponse = claimResponse;
        claim.Status = ClaimStatus.InProgress;
        claim.LastStatusUpdated = date;

        _context.Claims.Update(claim);
        await _context.SaveChangesAsync(cancellationToken);

        return _mapper.Map<CreateClaimResponseDto>(claim);
    }
}
```

Рис. 2.19 Команда для створення відповіді на скаргу

```
-
[HttpPost("{id:long}/change-status")]
0 references
public async Task<ActionResult<Unit>> ChangeClaimStatus(long id, [FromBody] ClaimStatus status)
{
    return await Mediator.Send(new ChangeClaimStatusCommand(id, status));
}
```

Рис. 2.20 Метод контролера ChangeClaimStatus

```

4 references
public record CreateClaimResponseCommand(long ClaimId) : IRequest<CreateClaimResponseDto>;

1 reference
public class CreateClaimResponseCommandHandler : IRequestHandler<CreateClaimResponseCommand, CreateClaimResponseDto>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;
    private readonly IDateTime _dateTime;
    private readonly ICurrentUserService _currentUserService;

    0 references
    public CreateClaimResponseCommandHandler(
        IMapper mapper,
        IApplicationDbContext context,
        IDateTime dateTime,
        ICurrentUserService currentUserService)
    {
        _mapper = mapper;
        _context = context;
        _dateTime = dateTime;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<CreateClaimResponseDto> Handle(CreateClaimResponseCommand request, CancellationToken cancellationToken)
    {
        var date = _dateTime.UtcNow;
        var claim = await _context.Claims.GetById(request.ClaimId, cancellationToken);

        var claimResponse = new ClaimResponse
        {
            Created = date,
            UserId = (long)_currentUserService.Id!,
            ClaimId = claim.Id,
            Message = ""
        };

        claim.ClaimResponse = claimResponse;
        claim.Status = ClaimStatus.InProgress;
        claim.LastStatusUpdated = date;

        _context.Claims.Update(claim);
        await _context.SaveChangesAsync(cancellationToken);

        return _mapper.Map<CreateClaimResponseDto>(claim);
    }
}

```

Рис. 2.21 Команда для зміни статусу певної скарги

```

[HttpPost("message")]
0 references
public async Task<ActionResult<ClaimMessageDto>> CreateMessage(
    [FromBody] CreateClaimMessageDto message)
{
    return await Mediator.Send(new CreateClaimMessageCommand(message));
}

```

Рис. 2.22 Метод контролера CreateMessage

```

4 references
public record CreateClaimMessageCommand(CreateClaimMessageDto Model) : IRequest<ClaimMessageDto>;

1 reference
public class CreateClaimMessageCommandHandler : IRequestHandler<CreateClaimMessageCommand, ClaimMessageDto>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;
    private readonly IDateTime _dateTime;
    private readonly ICurrentUserService _currentUserService;

    0 references
    public CreateClaimMessageCommandHandler(
        IMapper mapper,
        IApplicationDbContext context,
        IDateTime dateTime,
        ICurrentUserService currentUserService)
    {
        _mapper = mapper;
        _context = context;
        _dateTime = dateTime;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<ClaimMessageDto> Handle(CreateClaimMessageCommand request, CancellationToken cancellationToken)
    {
        var date = _dateTime.UtcNow;

        var claim = await _context.Claims.GetById(request.Model.ClaimId, cancellationToken);

        var message = new ClaimMessage
        {
            Message = request.Model.Message,
            UserId = (long)_currentUserService.Id,
            ClaimId = claim.Id,
            Status = claim.Status,
            Created = date
        };

        await _context.ClaimMessages.Insert(message);
        await _context.SaveChangesAsync(cancellationToken);

        return await _context.ClaimMessages.GetById<ClaimMessageDto>(message.Id, cancellationToken);
    }
}

```

Рис. 2.23 Команда для додання повідомлення до скарги

Коли інженер або бригадир вирішують почати роботу над скаргою, створюється відповідь на скаргу в системі і підв'язую користувача до цієї скарги. Після чого користувач може редагувати і доповнювати свою відповідь про виконані роботи, додаючи опис робіт, використані матеріали та документи або фото з доказами (рис. 2.24 – 2.27). Логіка для додавання документів до відповіді до скарги реалізована схожим чином як і для самої скарги.

```

[HttpPut("response")]
0 references
public async Task<ActionResult<Unit>> UpdateClaimResponse([FromBody] UpdateClaimResponseDto model)
{
    return await Mediator.Send(new UpdateClaimResponseCommand(model));
}

```

Рис. 2.24 Метод контролера UpdateClaimResponse

```

4 references
public record CreateClaimResponseCommand(long ClaimId) : IRequest<CreateClaimResponseDto>;

1 reference
public class CreateClaimResponseCommandHandler : IRequestHandler<CreateClaimResponseCommand, CreateClaimResponseDto>
{
    private readonly IMapper _mapper;
    private readonly IApplicationDbContext _context;
    private readonly IDateTime _dateTime;
    private readonly ICurrentUserService _currentUserService;

    0 references
    public CreateClaimResponseCommandHandler(
        IMapper mapper,
        IApplicationDbContext context,
        IDateTime dateTime,
        ICurrentUserService currentUserService)
    {
        _mapper = mapper;
        _context = context;
        _dateTime = dateTime;
        _currentUserService = currentUserService;
    }

    0 references
    public async Task<CreateClaimResponseDto> Handle(CreateClaimResponseCommand request, CancellationToken cancellationToken)
    {
        var date = _dateTime.UtcNow;
        var claim = await _context.Claims.GetById(request.ClaimId, cancellationToken);

        var claimResponse = new ClaimResponse
        {
            Created = date,
            UserId = (long)_currentUserService.Id!,
            ClaimId = claim.Id,
            Message = ""
        };

        claim.ClaimResponse = claimResponse;
        claim.Status = ClaimStatus.InProgress;
        claim.LastStatusUpdated = date;

        _context.Claims.Update(claim);
        await _context.SaveChangesAsync(cancellationToken);

        return _mapper.Map<CreateClaimResponseDto>(claim);
    }
}

```

Рис. 2.25 Команда для оновлення відповіді на скаргу

```

[HttpPost("response/{id:long}/materials")]
0 references
public async Task<ActionResult<List<ClaimedMaterialDto>>> SaveUsedMaterials([FromBody] AddClaimResponseMaterialsDto model)
{
    return await Mediator.Send(new AddClaimResponseMaterialsCommand(model));
}

```

Рис. 2.26 Метод контролера SaveUsedMaterials

```

4 references
public record AddClaimResponseMaterialsCommand(AddClaimResponseMaterialsDto Model) : IRequest<List<ClaimedMaterialDto>>;

1 reference
public class AddClaimResponseMaterialsCommandHandler: IRequestHandler<AddClaimResponseMaterialsCommand, List<ClaimedMaterialDto>>
{
    private readonly IApplicationDbContext _context;
    private readonly ICurrentUserService _currentUserService;

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

    0 references
    public async Task<List<ClaimedMaterialDto>> Handle(AddClaimResponseMaterialsCommand request,
        CancellationToken cancellationToken)
    {
        var materialIds = request.Model.Materials.Select(x => x.MaterialId).Distinct();
        var claimedMaterials = await _context.ClaimedMaterials.Get(el => el.ClaimResponse.Claim!.Building!.Project.CompanyId == _currentUserService.CompanyId && materialIds.Contains(el.MaterialId));

        foreach (var material in request.Model.Materials)
        {
            var originMaterial = await _context.Materials.FirstOrDefault(x => x.Id == material.MaterialId && x.CompanyId == _currentUserService.CompanyId, cancellationToken);
            var existingMaterialsSum = claimedMaterials.Where(x => x.MaterialId == material.MaterialId).Sum(x => x.Amount);

            if (originMaterial == null || material.Amount + existingMaterialsSum > originMaterial.Amount)
                continue;

            var existingMaterial = claimedMaterials.FirstOrDefault(x => x.MaterialId == material.MaterialId && x.ClaimResponseId == request.Model.ClaimResponseId);

            if (existingMaterial != null)
            {
                existingMaterial.Amount += material.Amount;
                _context.ClaimedMaterials.Update(existingMaterial);
            }
            else
            {
                var newMaterial = new ClaimedMaterial()
                {
                    Amount = material.Amount,
                    ClaimResponseId = request.Model.ClaimResponseId,
                    MaterialId = material.MaterialId,
                };
                await _context.ClaimedMaterials.AddRangeAsync(newMaterial);
            }
        }
        await _context.SaveChangesAsync();

        var materials = await _context.ClaimedMaterials.Get<ClaimedMaterialDto>(x => x.ClaimResponseId == request.Model.ClaimResponseId);
        return materials;
    }
}

```

Рис. 2.27 Команда для додавання використаних матеріалів

Після того як користувач оновив дані про виконання робіт і додав використані матеріали та документи з звітністю про виконання. Він може переглянути як це буде виглядати для менеджера і перевести в стан «Resolved». Після чого менеджер може перевірити чи скарга повністю виконана і покрита, і якщо та, то перевести в статус «Closed», тобто у фінальний статус виконання. А якщо він вважає, що робота не до кінця виконання або потребує певних правок, то може повернути в статус відкритої скарги «Open». Після чого робітник бере її у виконання по новому колу, або може запросити детальніші пояснення з переведенням в статус «Clarify». Де менеджер може додати додаткові пояснення або додати документи, якщо в тому є потреба. Якщо ж менеджер вважає, що скаргу можна признати невалідною, то переводить напрямку в фінальний статус «Closed».

Висновки до другого розділу

Під час проектування системи трекінгу процесів будівництва були приведені основні варіанти використання системи відносно наявних прав

користувача. За допомогою чого було спроектовано архітектуру системи. Також під час проєктування було продумано способи комунікації між модулями системи для забезпечення розділення логіки презентації та бізнес логіки, а саме завдяки використанню бібліотеки MediatR.

Також в даному розділі була розроблена структура бази даних відповідно до бізнес вимог до системи, яка зберігання та доступ до необхідних даних. База даних має 9 основних таблиць, а загальний обсяг 27 таблиць, в яких зберігається відповідна інформація про сутності системи, яка забезпечує потрібна відповідного виконання бізнес логіки.

В останньому пункті детально розглядається код, який є частиною модулю рекламаций додатку. Основний акцент приділяється операціям, пов'язаним із зберіганням і відображенням даних для користувача, створення та редагування скарг і змін статусів з відповідним надсиланням та зберіганням повідомлень.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання виступає як візуальне уявлення конфігурації вузлів обробки та компонентів системи під час їх встановлення. Це один із типів структурних діаграм, які використовуються для моделювання різних фізичних аспектів об'єктно-орієнтованої системи.

На (рис. 3.1) зображена діаграма розгортання, на якій показано конфігурацію вузлів та компонентів системи під час розгортання.

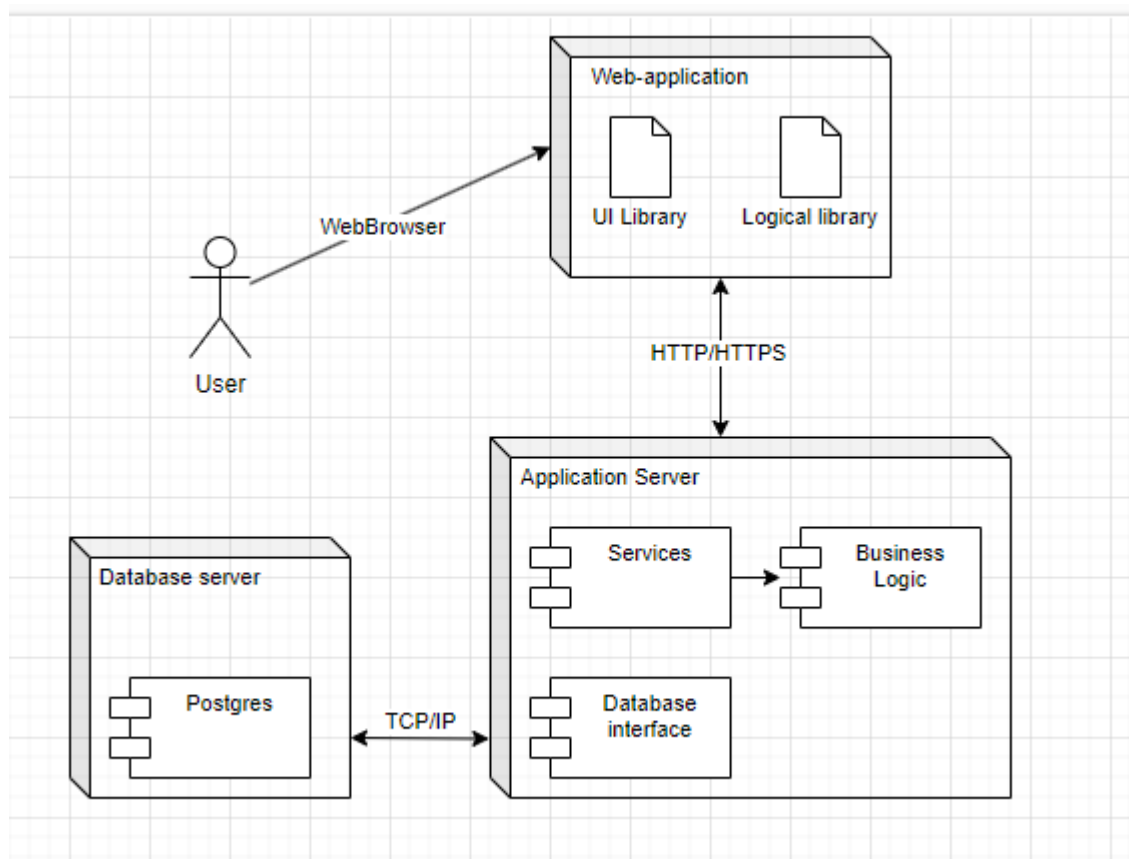


Рис 3.1 Діаграма розгортання додатку

Опишемо діаграму процесу розгортання:

- Web-application – розгортання інтерфейсу користувача (клієнтської частини) з відповідними модулями, бібліотеками та сервісами, які комунікують з серверною частиною та надають дані з сервера на обробку і візуалізацію.

- Application Server – розгортання серверної частини додатку з метою забезпечення можливості отримання та обробки даних з клієнтської частини з відповідними виконанням бізнес логіки.
- Database Server – розгортання бази даних, з якою взаємодіє сервер і зберігає оброблені дані.

3.2 Структура інтерфейсу та порядок взаємодії з системою трекінгу процесів будівництва

Продемонструємо інтерфейс системи на прикладі основного потоку виконання додатку. Коли користувач авторизується в системі, його зустрічає сторінка проєктів, (рис 3.2), де він може обрати існуючий або створити новий, якщо є оперативним менеджером.

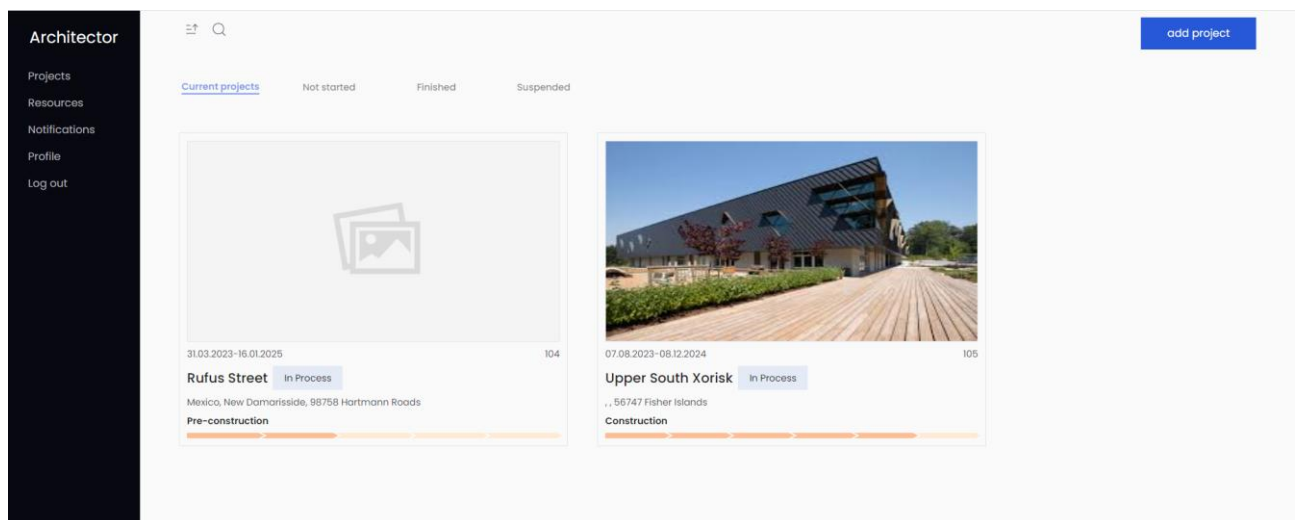


Рис. 3.2 Сторінка проєктів

Коли користувач відкриває сторінку певного проєкту, йому показується вся інформація про проєкт, а саме деталі проєкту, будівлі, команда, фото та багато іншого (рис 3.3).

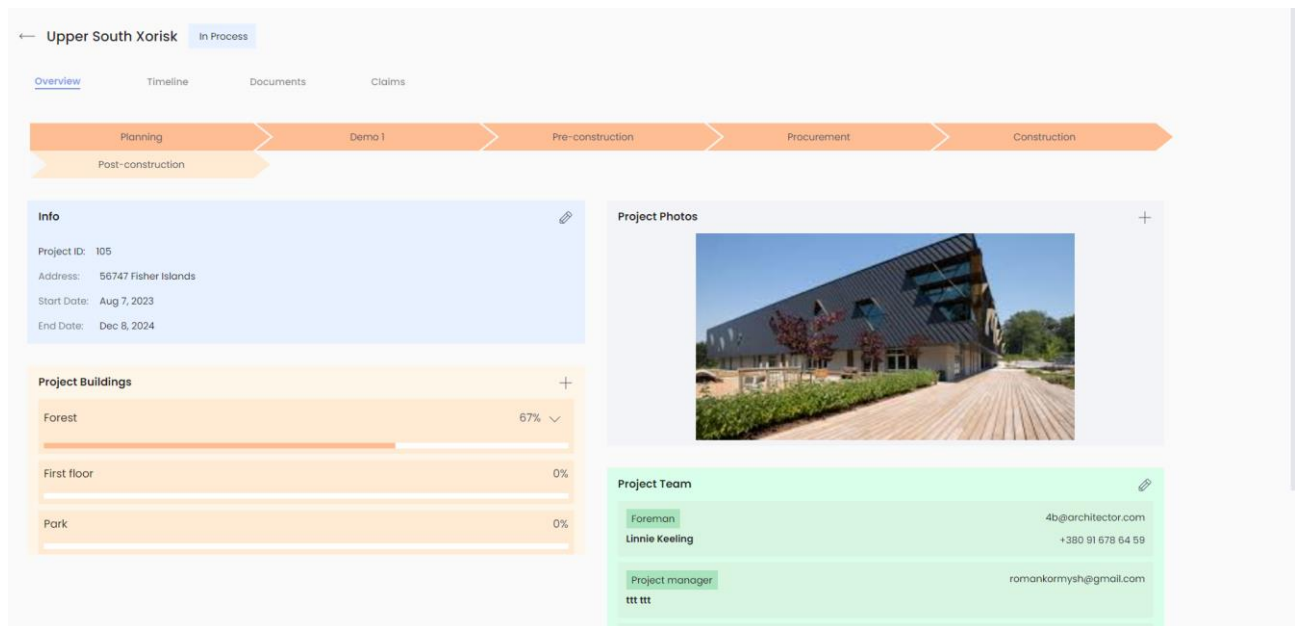


Рис. 3.3 Сторінка обраного проекту

Також користувач має можливість планувати процес забудови за допомогою таймлайну, до якого він може перейти натиснувши на другу вкладку «Timeline» (рис 3.4), де він може створювати фази і під фази, а також редагувати їх (рис. 3.5 – 3.6).

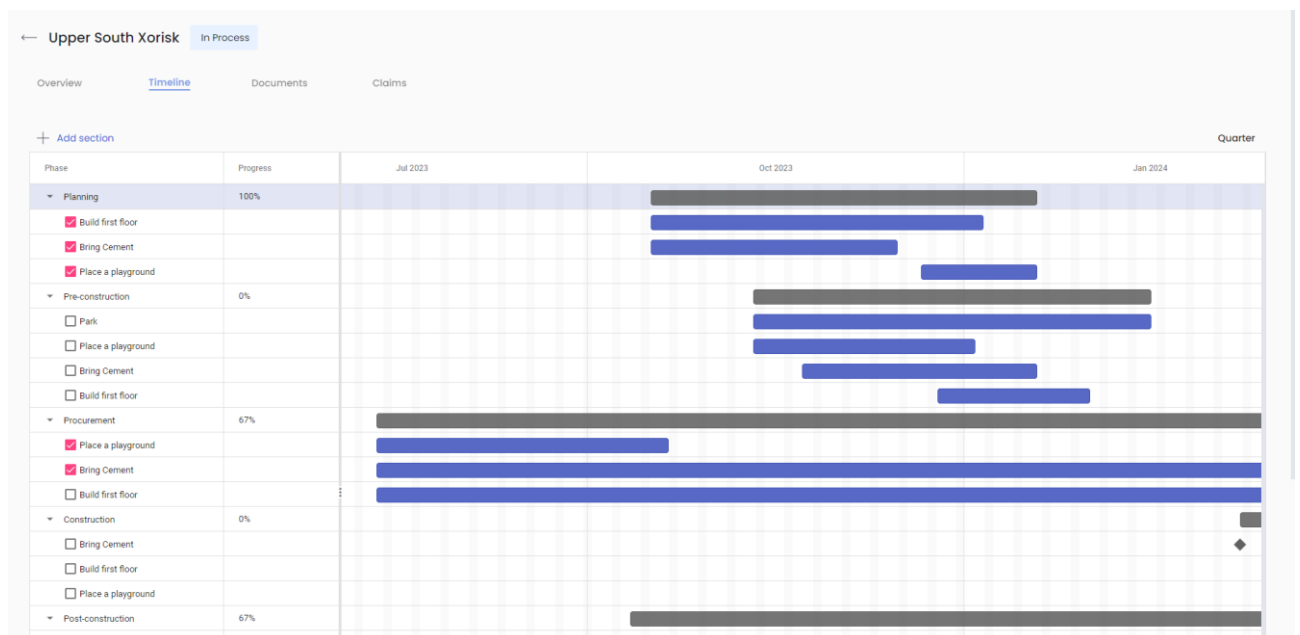


Рис. 3.4 Таймлайн для проекту

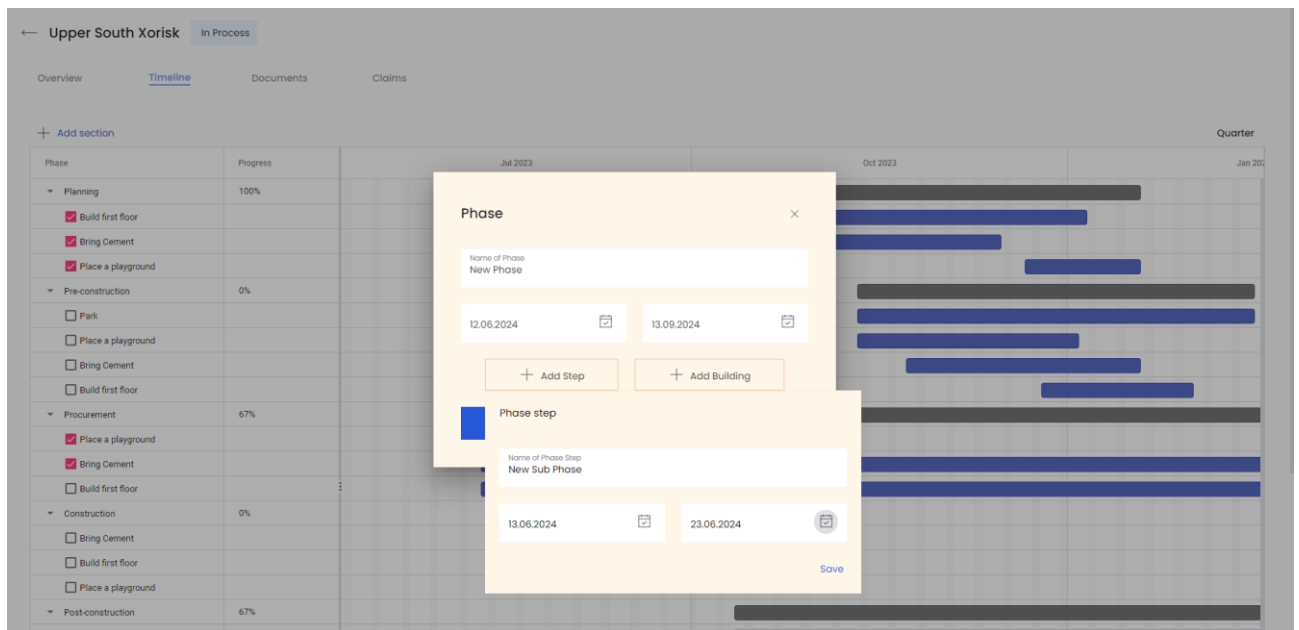


Рис. 3.5 Додання фази та під-фази

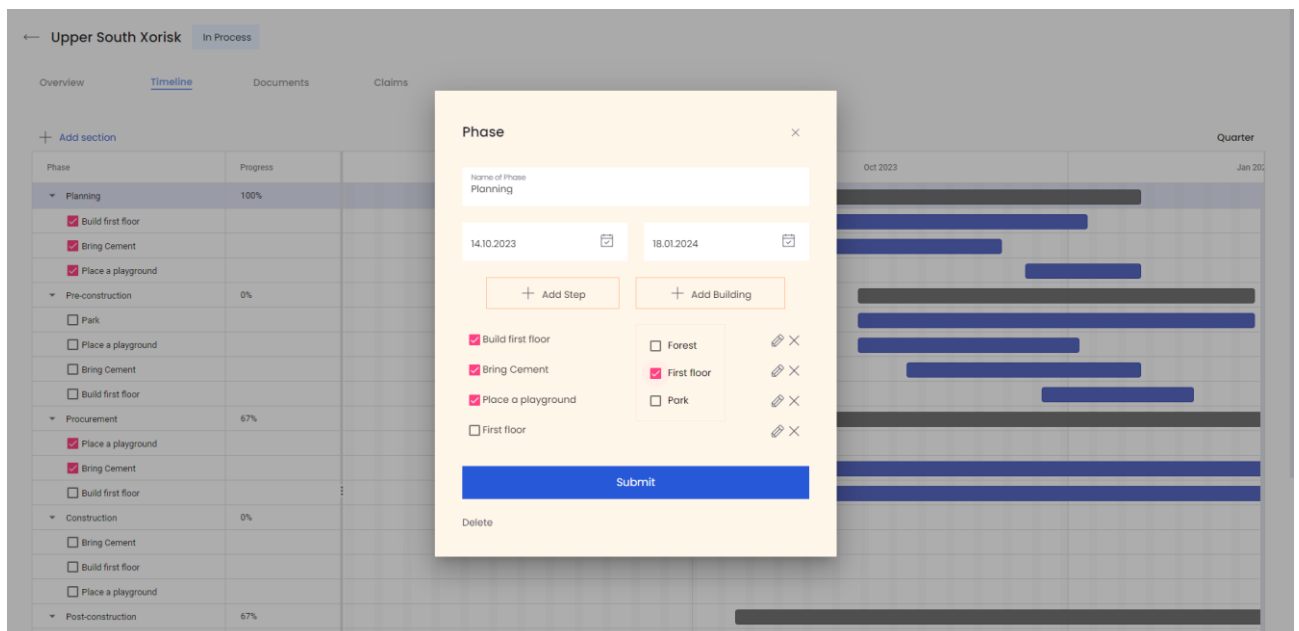


Рис. 3.6 Редагування під фази з додання нової будівлі

Також користувач може створити звіти по різним даним проєкту на вкладці «Documents», а саме основна інформація про проєкт в PDF, інформація про використані матеріали та сервіси в CSV (рис. 3.7 – 3.8).

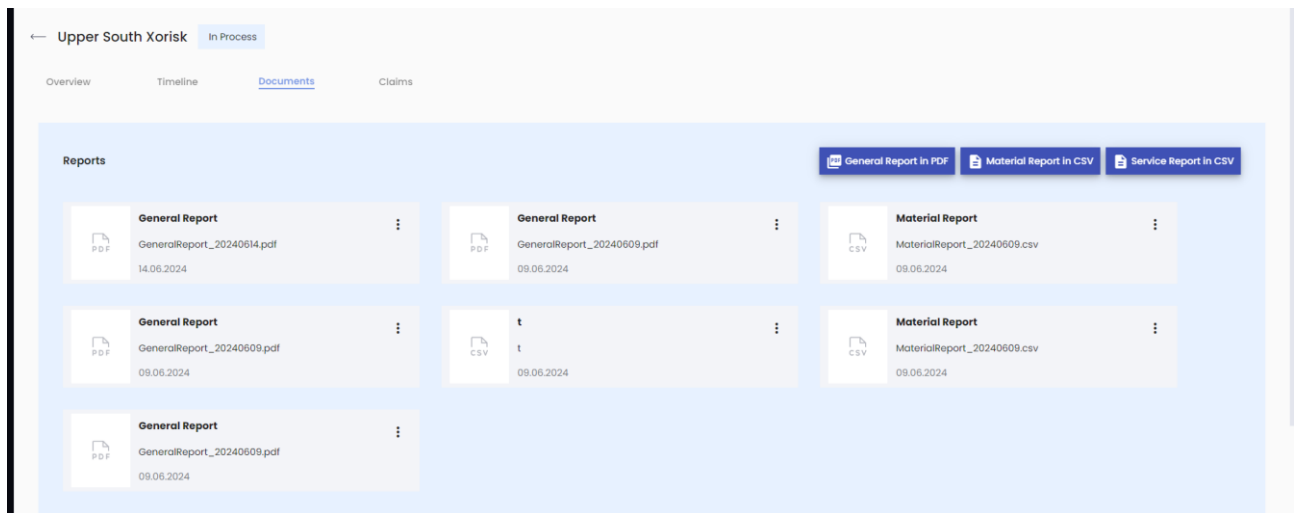


Рис. 3.7 Створення звіту

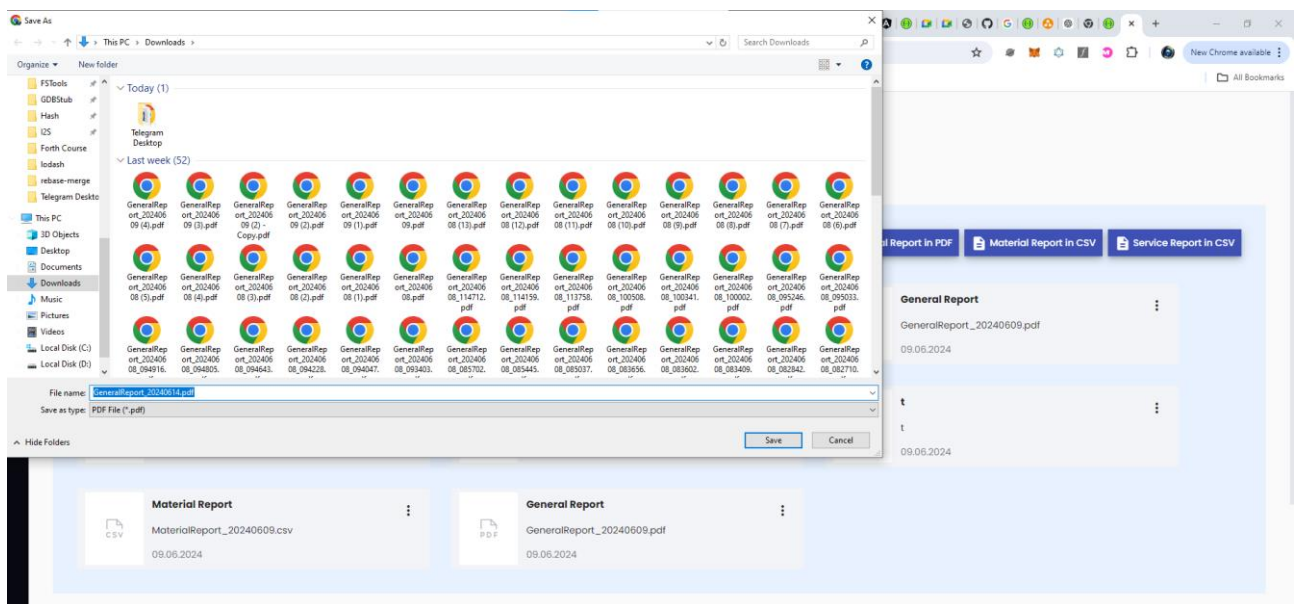


Рис. 3.8 Збереження звіту після створення

Зразу після створення звітів користувач може переглянути їх, а також документи по поточному проєкті. В залежності від типу файлу чи звіту надаються різні види перегляду (рис. 3.9 – 3.12).

[illegible]

Рис. 3.11 Звіт по використаним сервісам

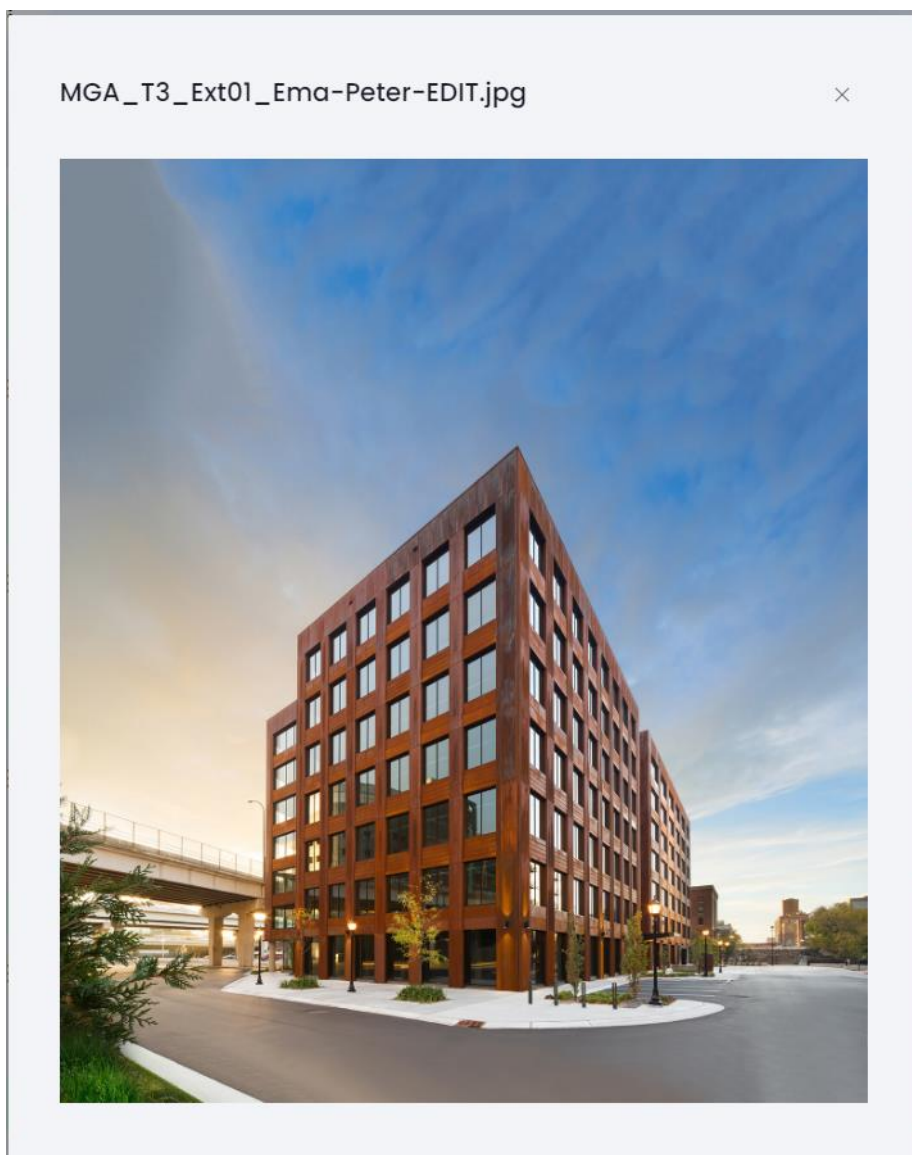


Рис. 3.12 Перегляд файлу типу фото

Ключовим функціоналом даної платформи є те, що вона надає можливості управління рекламацією на відміну від аналогів. Для цього користувач може перейти на вкладку «Claims» поточного проєкту (рис 3.13).

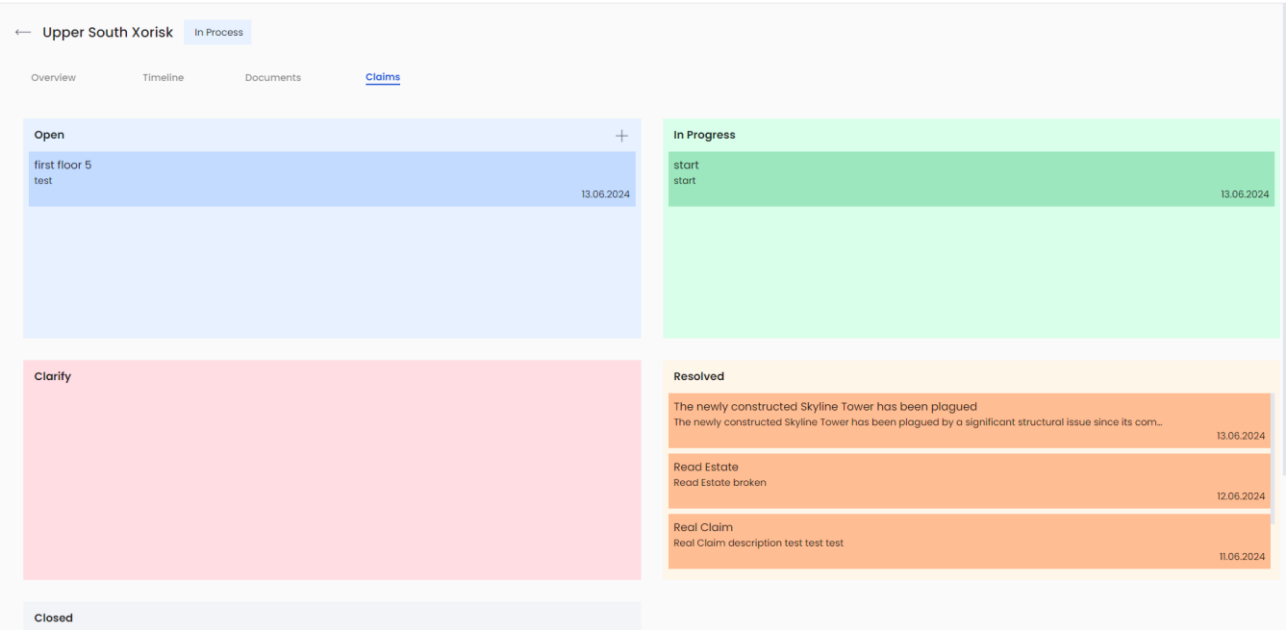


Рис. 3.13 Сторінка скарг

Якщо користувач є менеджером проєкту або оперативним менеджером, то він може декларувати скарги та редагувати їх, а також на кожному з цих етапів додати необхідну інформацію та файли з поясненням або показом проблеми (рис 3.14 – 3.19).

Create claim

Title *

Real claim

Description *

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building *

First floor

+ Add documents

Submit

Рис. 3.14 Форма створення скарги

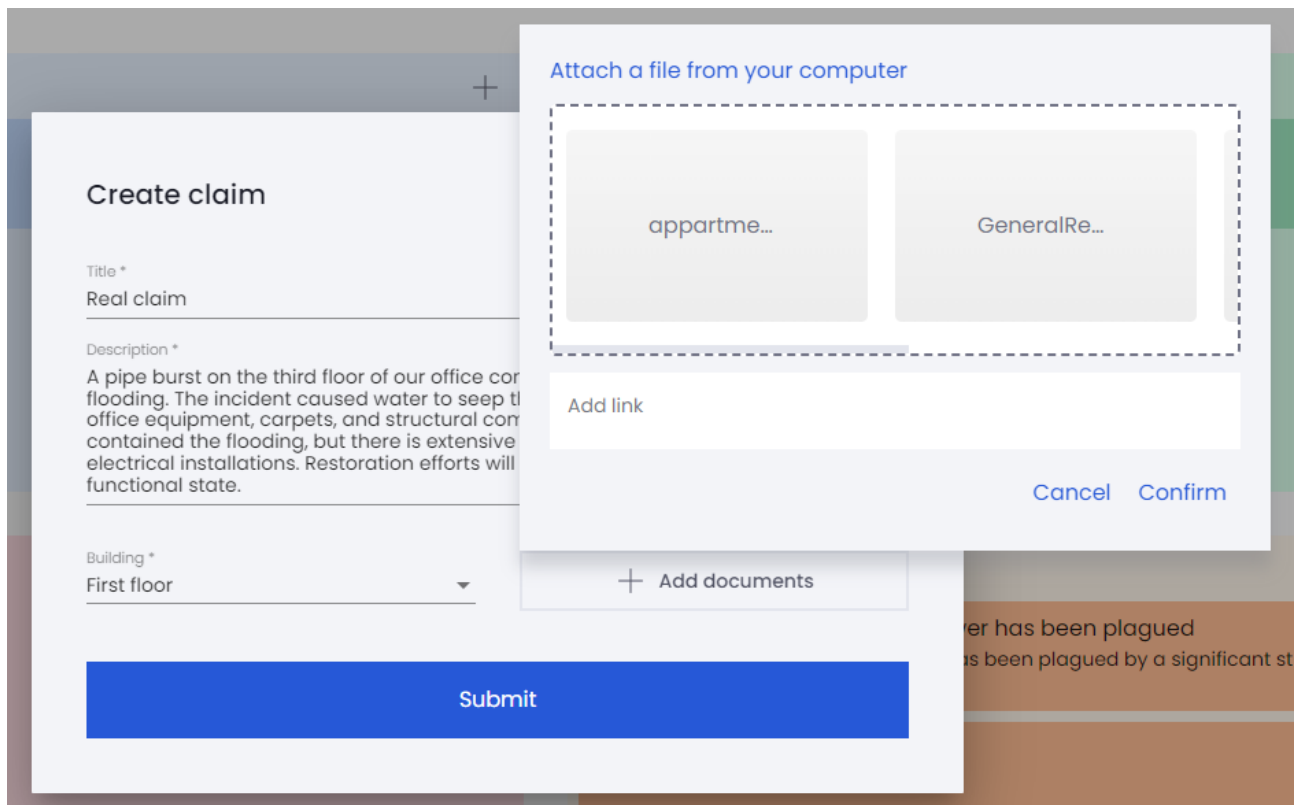


Рис. 3.15 Додання документів до скарги

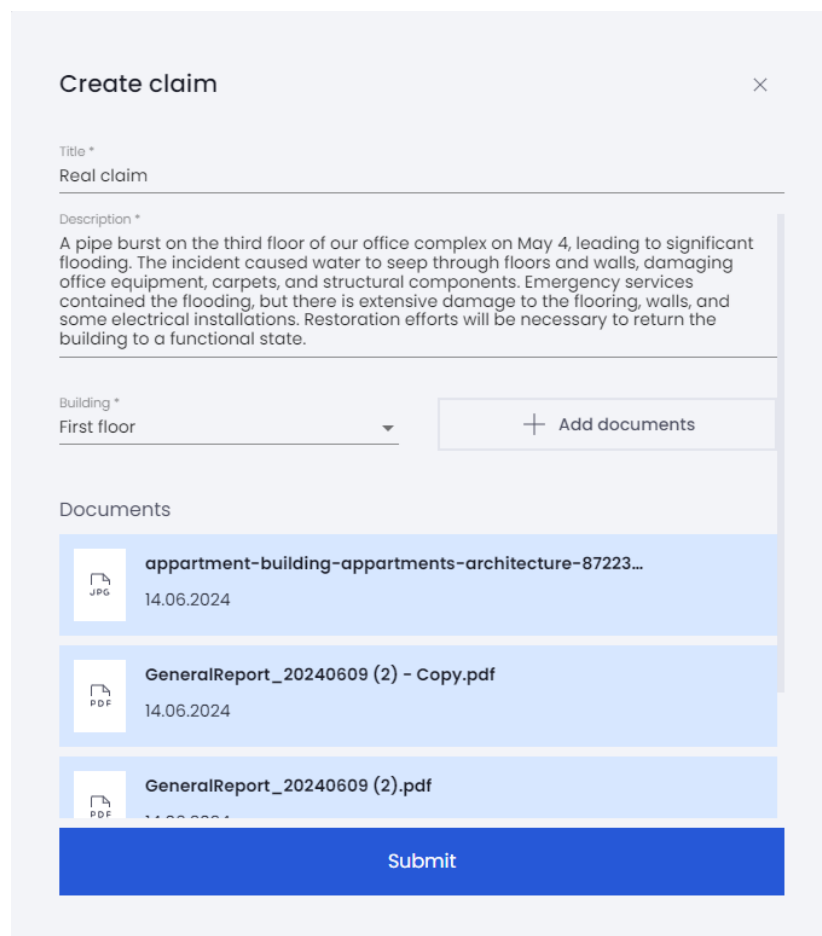


Рис. 3.16 Результат додання документів до скарги

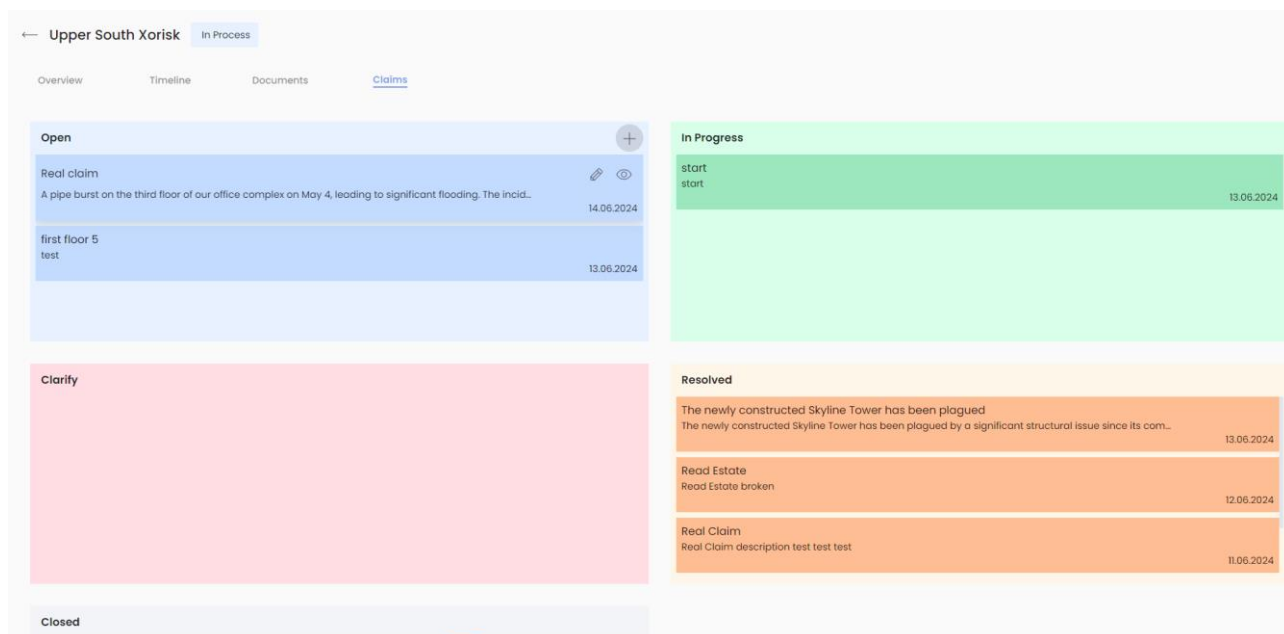


Рис. 3.17 Успішне створення скарги

Edit claim

Info

History

Title *

Real claim

Description *

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building *

Forest

+ Add documents

Documents

See all

PDF

GeneralReport_20240609 (1).pdf

14.06.2024

Save

Рис. 3.18 Редагування скарги

Real claim

×

Info

History

Description

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building

First floor

Creator

Operational manager

lb@architector.com

Kody Wiegand

+380 99 962 40 30

Documents

See all

GeneralReport_20240609 (1).pdf

Рис. 3.19 Перегляд новоствореної скарги

Коли скарга задекларована і знаходиться в статусі «Open», користувач, який має роль чи бригадира чи інженера, може взяти дану скаргу в опрацювання, зі зміною статусу скарги на “In Progress”, або уточнити додатково уточнити деталі скарги, з зміною статус на “Clarify”. Також при зміні статус користувач може вказати додаткові деталі в вигляді повідомлення (рис. 3.20 – 3.21).

Real claim

×

Info

History

flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building

First floor

Creator

Operational manager

1b@architector.com

Kody Wiegand

+380 99 962 40 30

Documents

See all



GeneralReport_20240609 (1).pdf

14.06.2024

Message *

Gimme more details

Accept

Clarify

Рис. 3.20 Перегляд скарги від лиця інженера або бригадира із запитом на уточнення даних

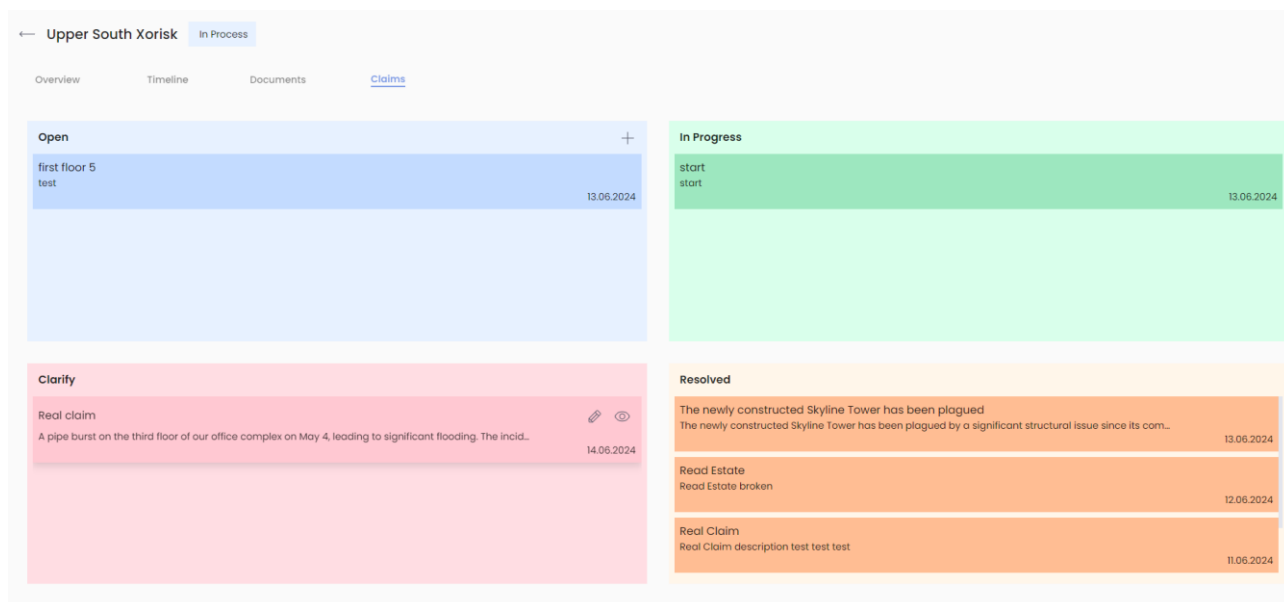


Рис. 3.21 Зміна статусу на «Clarify»

На скаргу, яка знаходиться в статус «Clarify», може відкликнутися менеджер і додати необхідні деталі і потім повернути в статус «Open», або ж визначити скаргу невалідною і перевести її в статус «Closed». Також на кожному етапі менеджер може вказати додаткове повідомлення для користувача з робочою роллю (рис. 3.22 – 3.24)

Edit claim

Info

History

Title *

Real claim

Description *

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building *

Forest

+ Add documents

Documents

See all

GeneralReport_20240609 (1).pdf

14.06.2024

Save

Рис. 3.22 Редагування скарги в статусі «Clarify»

Real claim

×

Info

History

flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building

First floor

Creator

Operational manager

1b@architector.com

Kody Wiegand

+380 99 962 40 30

Documents

See all



GeneralReport_20240609 (1).pdf

14.06.2024

Message

Added clarification notes

Reject

Clarify

Рис. 3.23 Додання уточнювальної інформації зі зміною статусу на «Open»

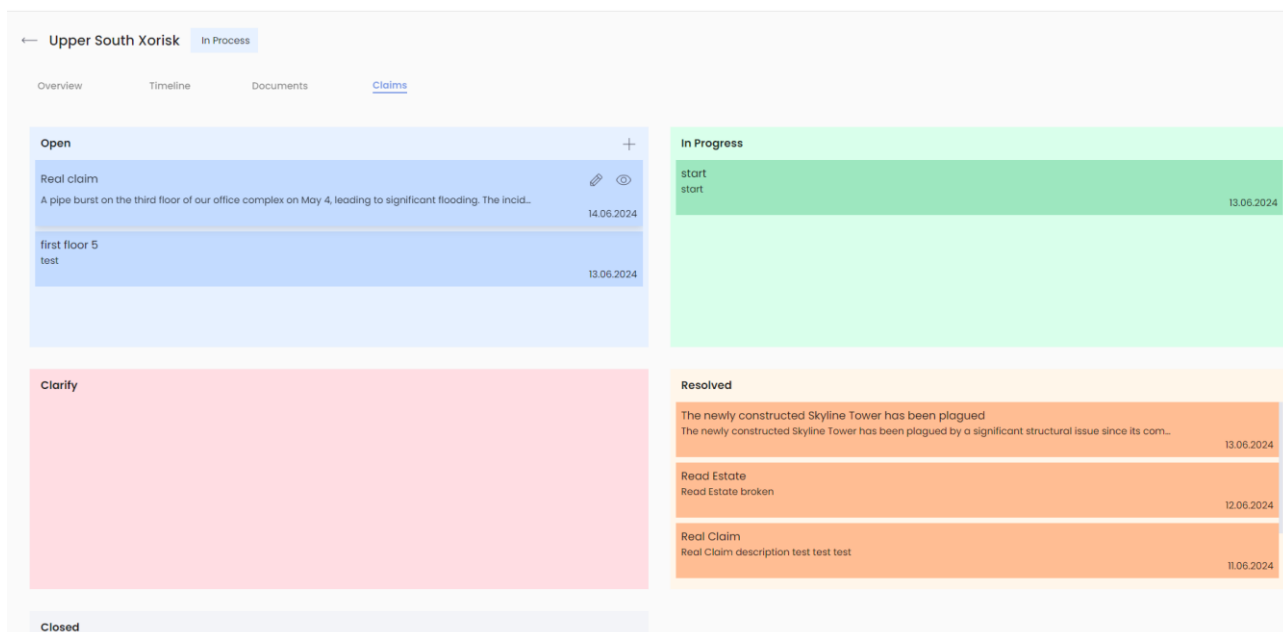


Рис. 3.24 Успішне повернення назад в статус «Open»

Далі, коли задача повернулася з уточненнями, користувач з робочою роллю може взяти її в опрацювання або звернутися за подальшими уточненнями (рис 3.25 – 3.26).

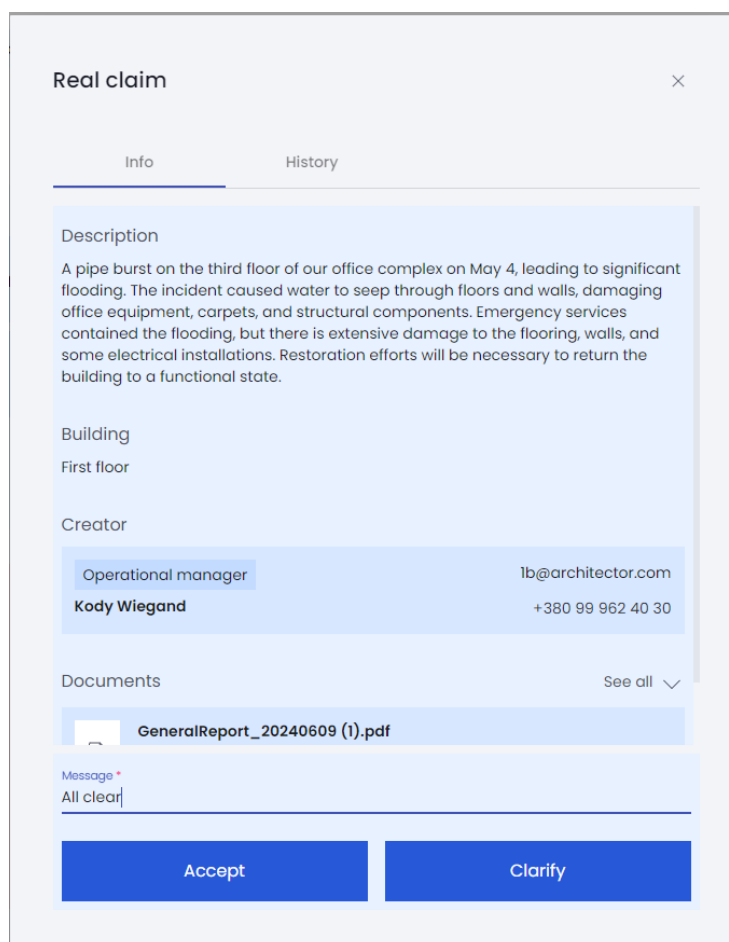


Рис. 3.25 Зміна статусу на «In Progress»

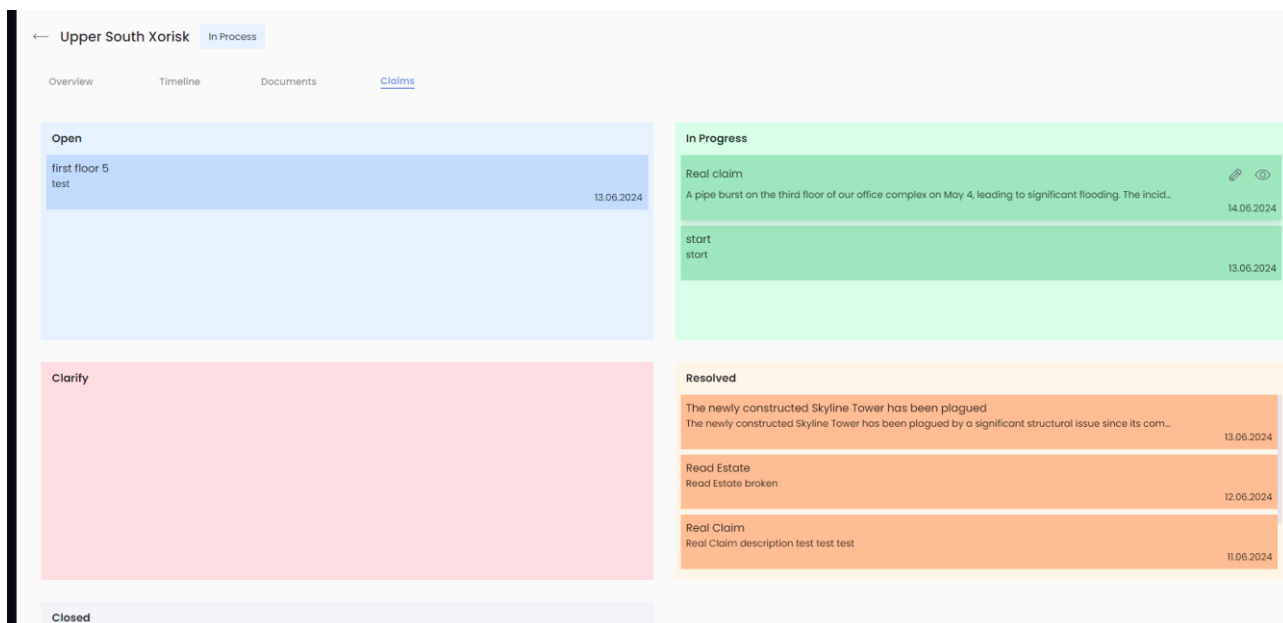


Рис. 3.26 Відображення успішної зміни статусу

Далі користувач з робочою роллю виконує роботу і доповнює звіт про виконання робіт, а також може додати інформацію про використанні матеріали, прикріпити документи зі змінами або доказами (рис. 3.27 – 3.29).

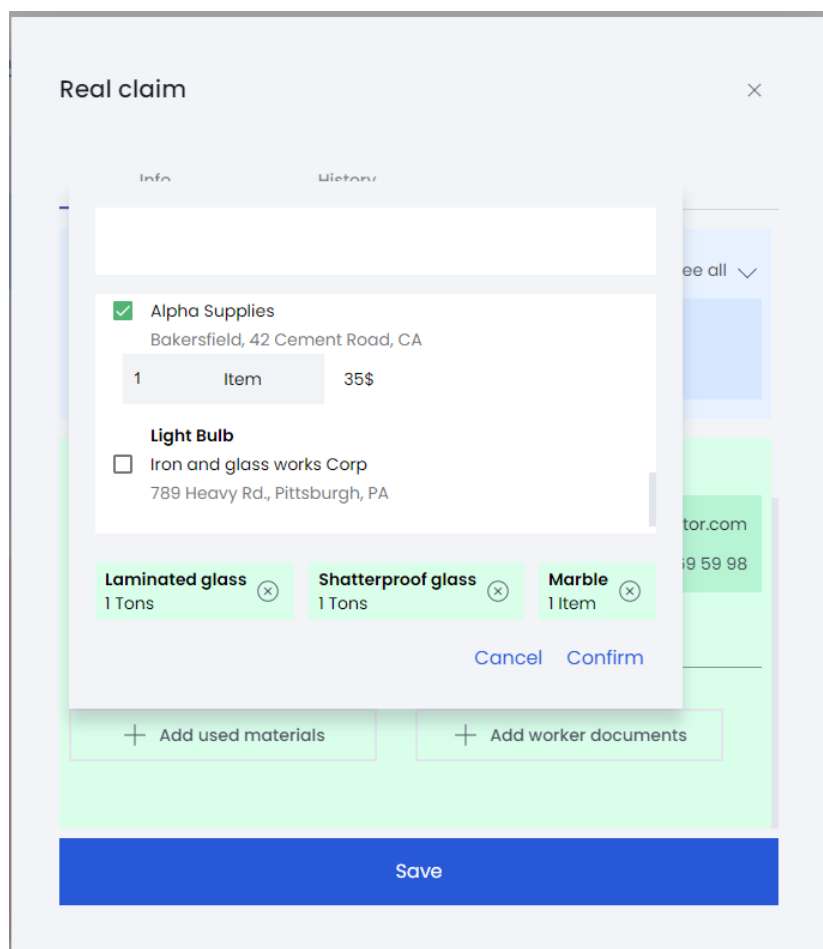


Рис. 3.27 Додання інформації про використанні матеріали

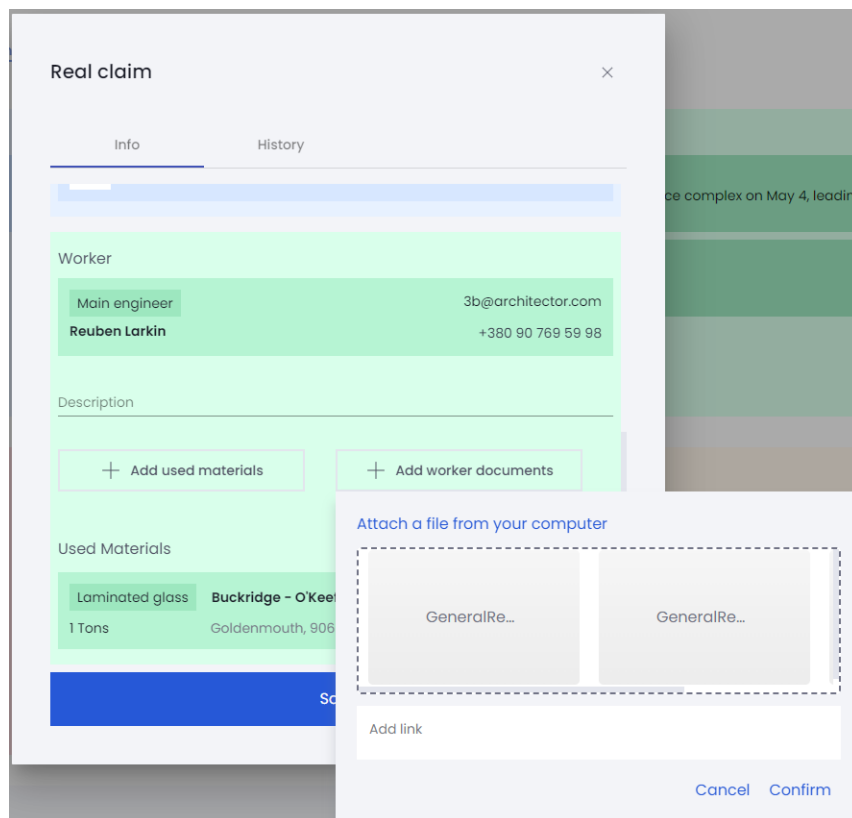


Рис. 3.28 Додання документів з доказами або змінами

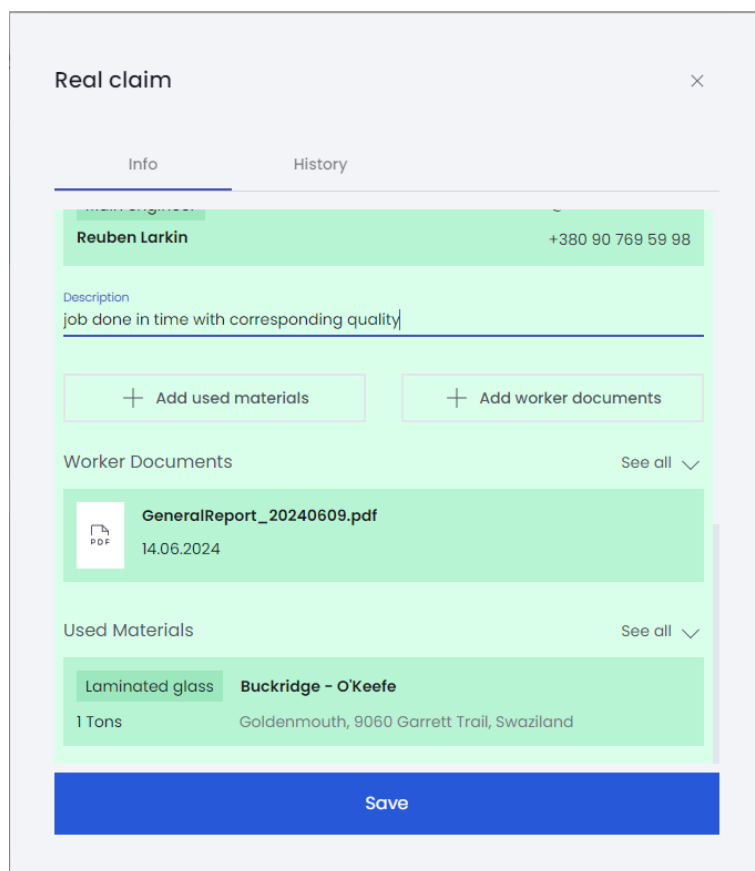


Рис. 3.29 Успішна зміна даних

Після виконаних робіт робочий змінює статус скарги на «Resolved», додаючи повідомлення про виконання (рис. 3.30 – 3.31).

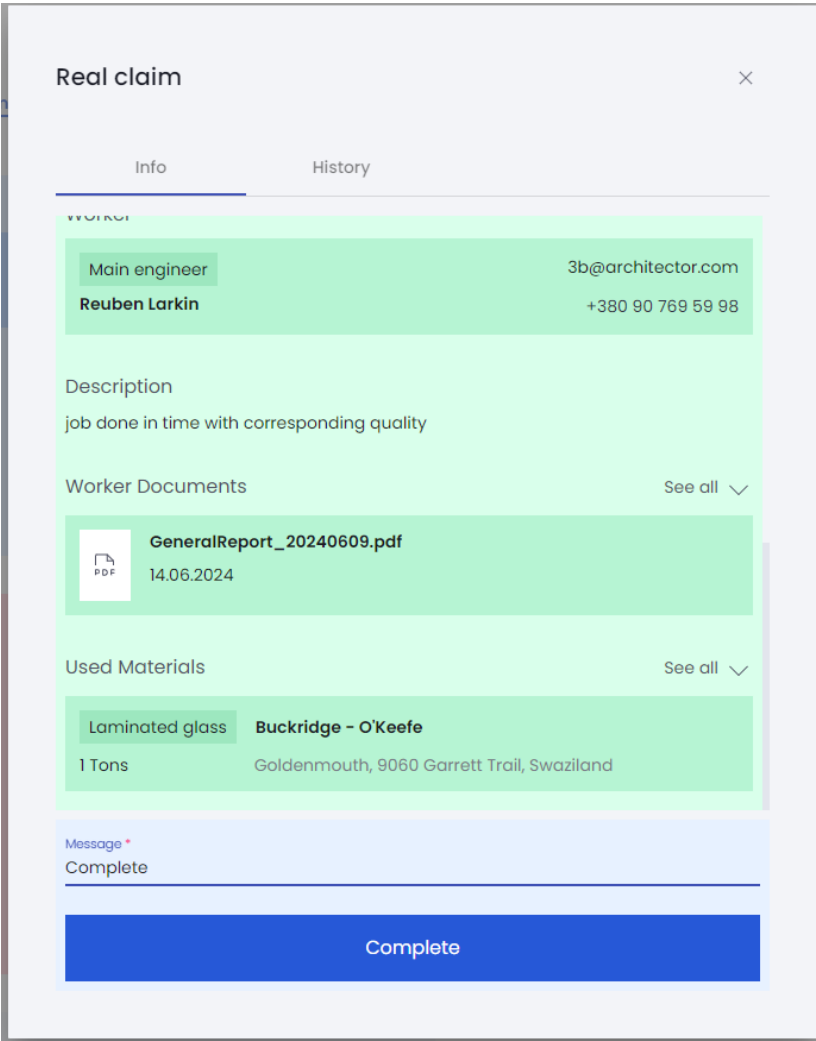


Рис. 3.30 Підтвердження виконання та успішна зміна статусу на «Resolved»

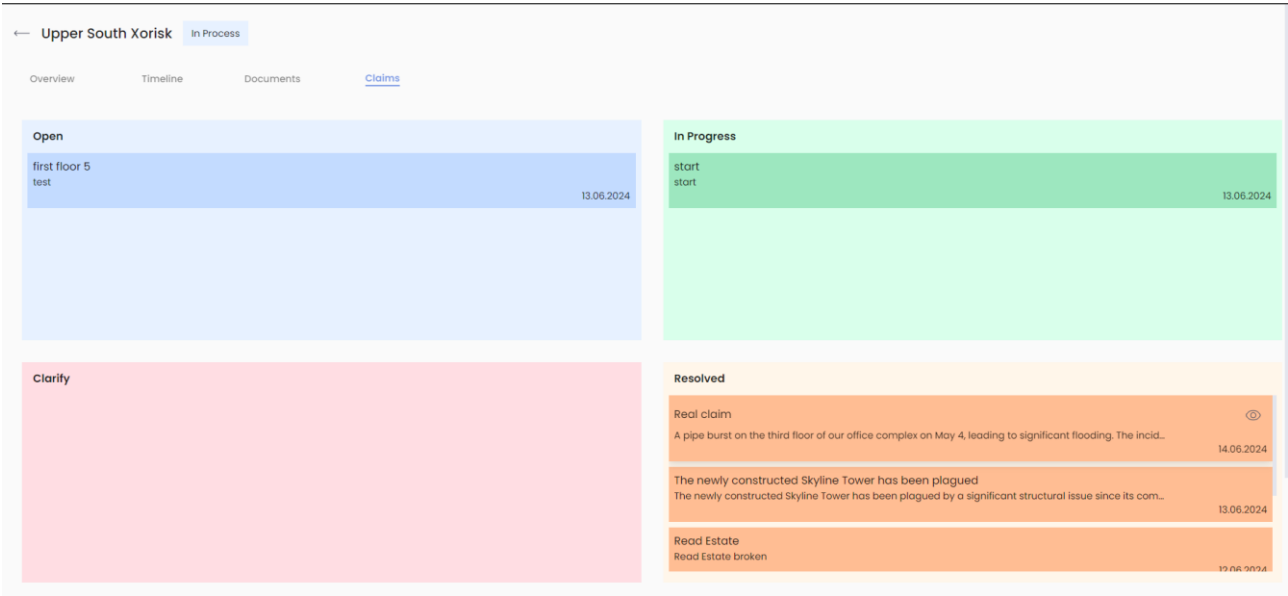


Рис. 3.31 Відображення успішної зміни статусу

Далі користувач з роллю менеджера може переглянути всю виконану роботу і порівняти з відповідною скаргою. Якщо все відповідає потребам виконання робіт, він змінює статус скарги на фінальний «Closed», а якщо ж ні, то може перевідкрити скаргу, змінивши статус на «Open». Потім скарга може пройти всі вище описані кроки з відповідною кількістю необхідних ітерацій (рис 3.32 – 3.36).

Real claim

Info

History

Description

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building

First floor

Creator

Operational manager

Kody Wiegand

lb@architector.com

+380 99 962 40 30

Documents

See all

GeneralReport_20240609 (1).pdf

Message

not done

Approve

Reopen

Рис. 3.32 Відхилення повноти виконання необхідної роботи

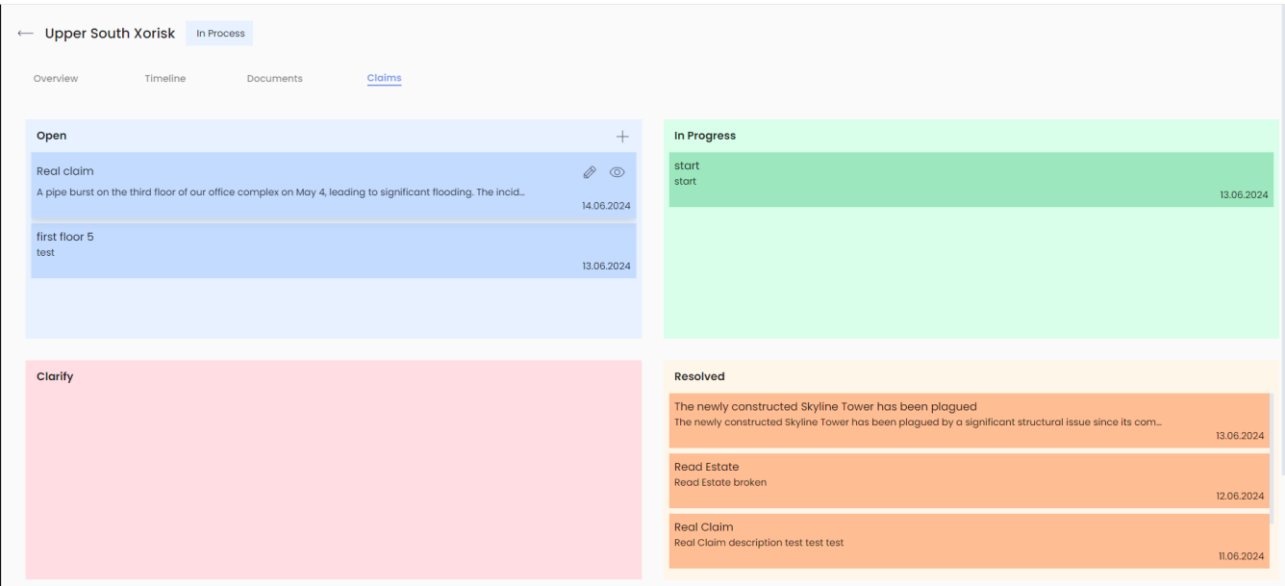


Рис. 3.33 Перевідкриття скарги з поверненням в статус «Open»

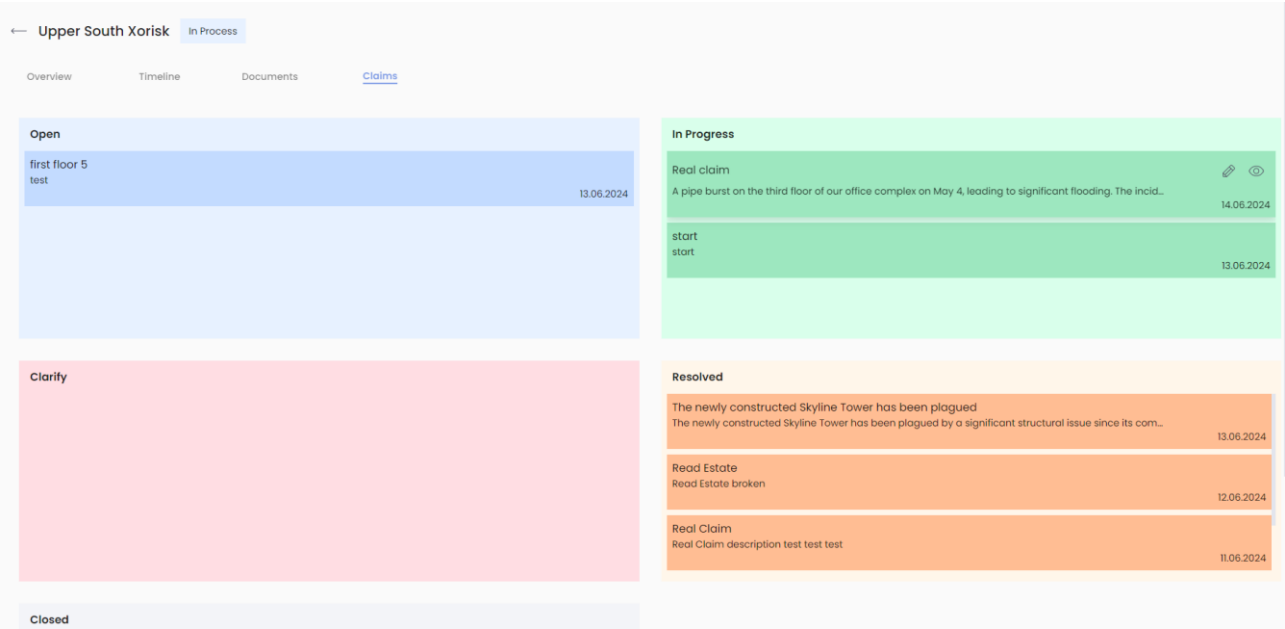


Рис. 3.34 Повторне виконання з переведенням в статус «In Progress»

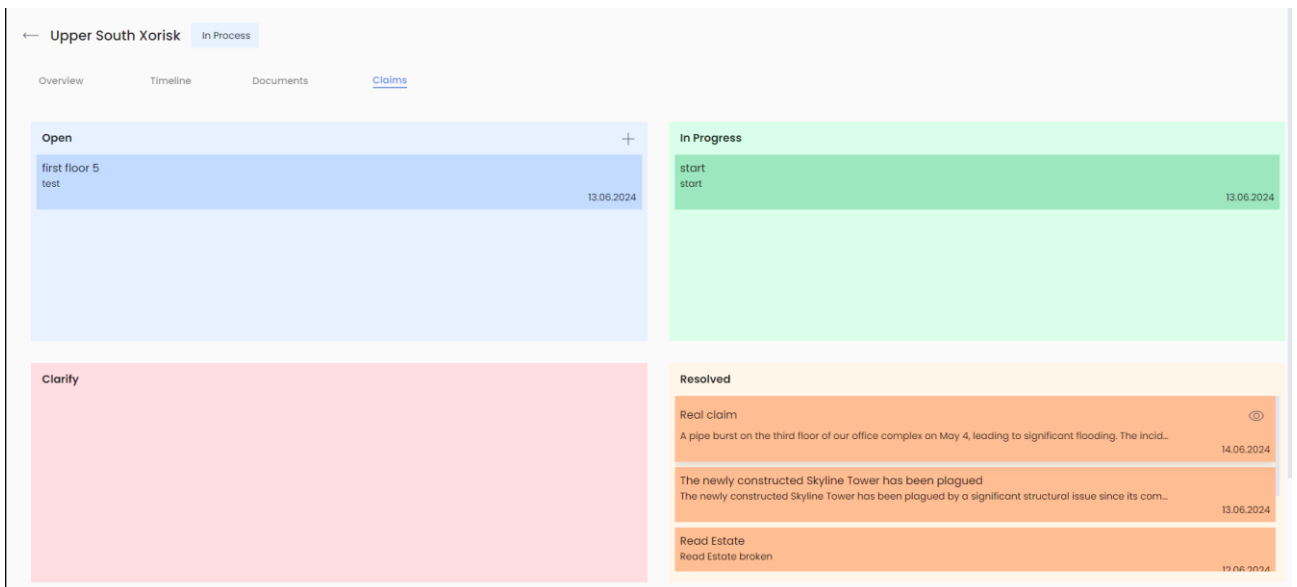


Рис. 3.35 Повернення в статус «Resolved»

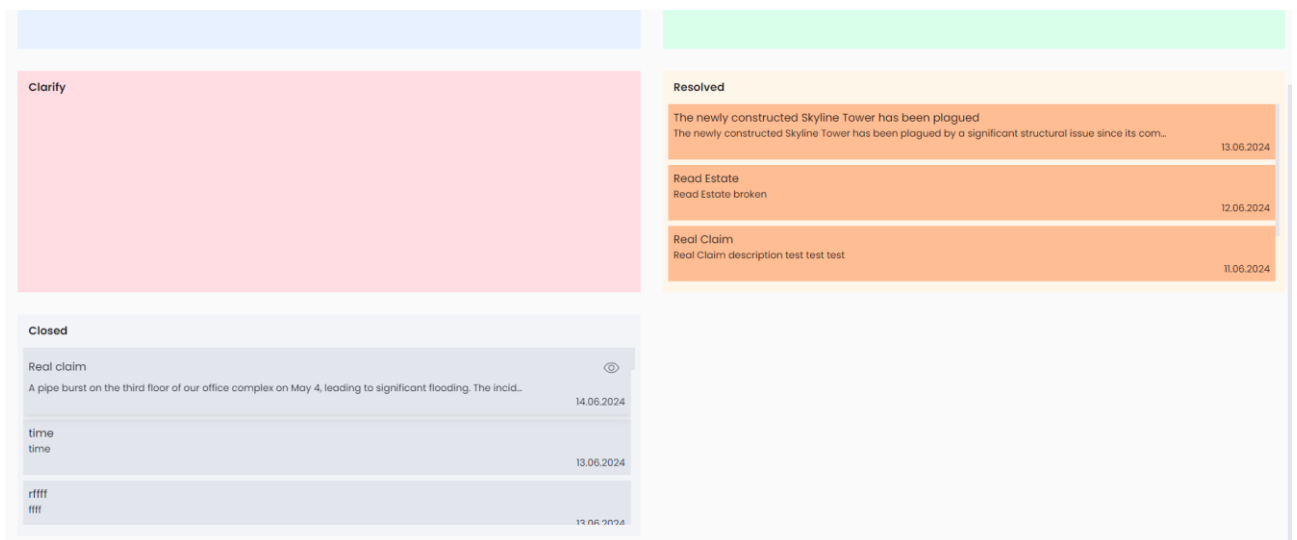


Рис. 3.36 Підтвердження виконання робіт з зміною статус на «Closed»

Коли скарга в фінальному статусі «Closed», то її ніхто не може редагувати, тільки переглядати. Також при перегляді скарги можна подивитися історію змін статусів з відповідними повідомленнями (рис. 3.37 – 3.40)

Real claim

Info

History

Description

A pipe burst on the third floor of our office complex on May 4, leading to significant flooding. The incident caused water to seep through floors and walls, damaging office equipment, carpets, and structural components. Emergency services contained the flooding, but there is extensive damage to the flooring, walls, and some electrical installations. Restoration efforts will be necessary to return the building to a functional state.

Building

First floor

Creator

Operational manager

1b@architector.com

Kody Wiegand

+380 99 962 40 30

Documents

See all

GeneralReport_20240609 (1).pdf

Рис. 3.37 Перегляд скарги

Real claim

Info

History

Worker

Main engineer

3b@architector.com

Reuben Larkin

+380 90 769 59 98

Description

job done in time with corresponding quality

Worker Documents

See all

GeneralReport_20240609.pdf

14.06.2024

Used Materials

See all

Laminated glass

Buckridge – O’Keefe

1 Tons

Goldenmouth, 9060 Garrett Trail, Swaziland

Рис. 3.38 Перегляд відповіді на скаргу

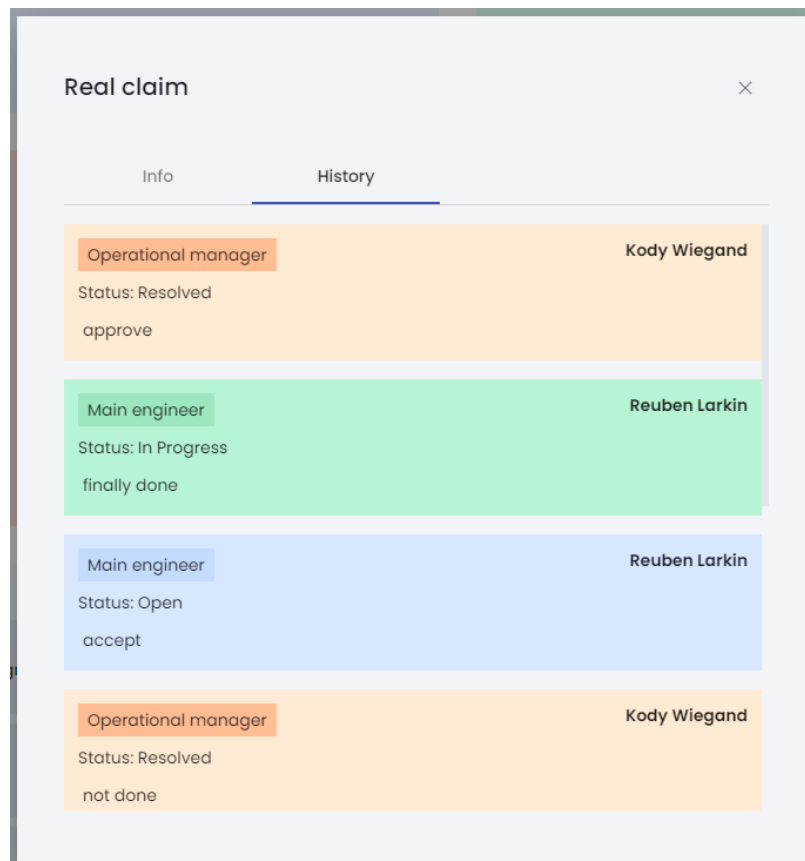


Рис. 3.39 Перегляд кінця історії змін статусів

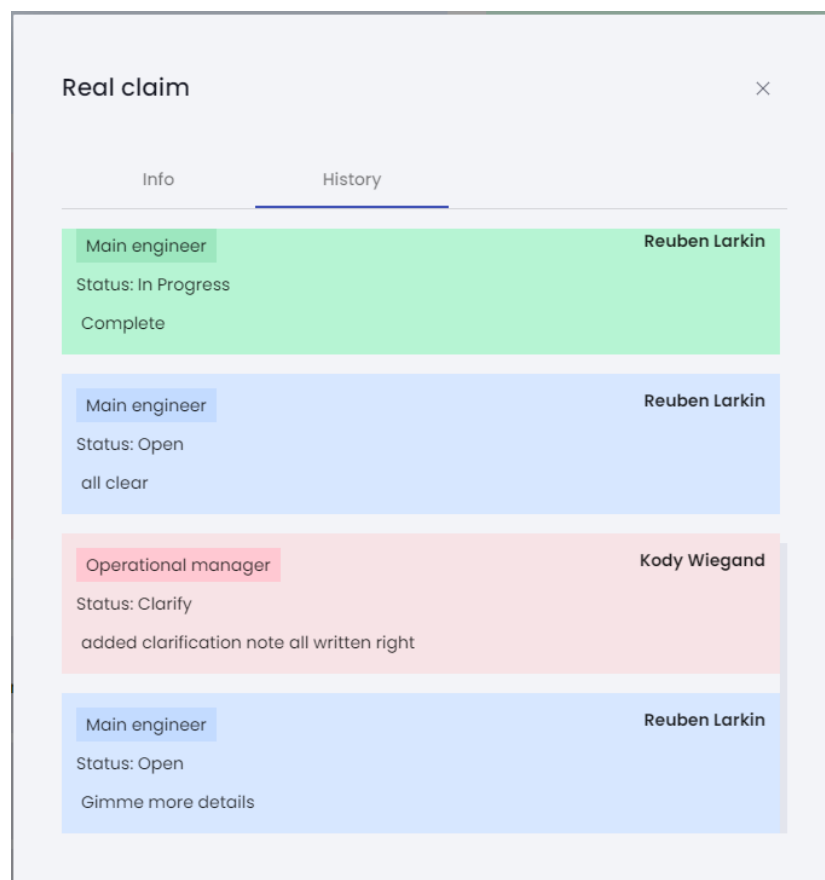


Рис. 3.40 Перегляд початка історії змін статусів

На вкладці з історією відображена зміна статусів скарги з відповідними повідомленнями, де користувач, не залучений до виконання скарги, може дізнатися як проходило виконання.

3.3. Тестування роботи програмного продукту

Створення будь-якого програмного продукту передбачає припущення деяких помилок у ході розробки, тому що не можливо передбачити всі можливі випадки зразу. Тому без тестування програмного забезпечення неможливо отримати додаток доволі високої якості. Під час тестування перевіряються різні аспекти роботи програми, такі як точність вхідних і вихідних даних, релевантність функцій і алгоритмів, обробка помилок, відновлення після збоїв, тощо. Метою тестування є знаходження та виправлення помилок і недоліків в програмних продуктах, які можуть бути виправлені до того, як вони будуть випущені або введені в експлуатацію на ринку.

Для початку створимо так звані Test Cases [19]. Test Case – це тестовий артефакт, суть якого полягає у виконанні деякої кількості дій та/або умов, необхідних для перевірки певної функціональності програмної системи, що розробляється. Представимо створені тестові випадки у вигляді таблиці (табл. 3.1).

Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка авторизації з дійсними даними	1. Перейти на сторінку https://archite.ctor/login 2. Введіть email або логін 3. Введіть пароль 4. Натисніть "Увійти"	Email = admin@gmail.com Пароль = 12345	Користувач повинен увійти в систему	Як і очікувалося	Pass

Продовження Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка авторизації з недійсними даними	1. Перейти на сторінку https://archite.ctor/login 2. Введіть email або логін 3. Введіть пароль 4. Натисніть "Увійти"	Email = admin@gmail.com Пароль = 54321	Користувач не повинен увійти в систему і отримати відповідне повідомлення	Як і очікувалося	Pass
Перевірка скидання пароля	1. Перейти на сторінку https://archite.ctor/login 2. Натиснути "Забули пароль" 3. Введіть email 4. Натисніть "Підтвердити" 5. Перейти до пошти та перейти за вказаним посиланням 6. Введіть email 5. Введіть пароль отриманий з листа щоб увійти до системи	Email = testemail@gmail.com Пароль = отриманий на пошту	Пароль користувача повинен змінитися	Як і очікувалося	Pass

Продовження Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка скидання пароля	1. Перейти на сторінку https://archite.ctor/login 2. Натиснути "Забули пароль" 3. Введіть email 4. Натисніть "Підтвердити" 5. Перейти до пошти та перейти за вказаним посиланням 6. Введіть email 6. Введіть пароль отриманий з листа щоб увійти до системи	Email = testemail@gmail.com Пароль = отриманий на пошту	Пароль користувача повинен змінитися	Як і очікувалося	Pass
Перевірка додання нового робітника на проєкт	1. Увійти в систему у ролі операційного або проєктного менеджера 2. Обрати відповідний проєкт 3. Натиснути на іконку олівця біля складу команди проєкту 4. Обрати відповідного працівника компанії 7. Підтвердити наявний стан команди, натиснувши на кнопку підтвердити	Наявний список працівників компанії	Повинно додати працівника компанії в склад команди проєкту і надати йому відповідний доступ до нього	Як і очікувалося	Pass

Продовження Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка сортування та фільтрування проєктів	1. Увійти в систему у ролі користувача 2. Перейти на сторінку з проєктами 3. Змінити тип сортування або вкладку, або ввести щось в пошук	Критерій = за новизною Фільтрація = "Town"	Повинно відобразитися список проєктів в порядку відповідно до обраного сортування та фільтрування	Як і очікувалося	Pass
Перевірка додання фото до проєкту	1. Увійти в систему у ролі операційного або проєктного менеджера 2. Перейти на сторінку з проєктами 3. Натиснути на іконку прюса в секціях проєктих фотографій 4. Натиснути на його 5. Завантажити необхідні фото 6. Підтвердити додання фото до проєкту	Фото проєкту, які потрібно додати	Повинні відобразитися в слайдері проєкту його фото	Як і очікувалося	Pass

Продовження Таблиця 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка додання скарги до проєкту	1. Увійти в систему у ролі операційного або проєктного менеджера 2. Перейти на сторінку з проєктами 3. Обрати вкладку зі скаргами 4. Натиснути на кнопку з плюсом для створення скарги 5. Додати поля з назвою та описом 6. Обрати будівлю 7. Опціонально додати документи 4. Підтвердити створення скарги	Інформація про скаргу і документи, які слугують доказом	Повинна додатися скарги в списку відкритих скарг	Як і очікувалося	Pass
Перевірка генерації звіту про використанні матеріали на проєкті	1. Увійти в систему 2. Перейти на сторінку з проєктами 3. Натиснути на відповідний проєкт 4. Перейти на вкладку з документами 5. Натиснути на кнопку звіт з матеріалами в CSV	Дані про використанні матеріали для будівель на проєкті	Повинне відкритися вікно на збереження файлу	Як і очікувалося	Pass

Проаналізувавши таблицю вище робимо висновок, що наведені тестові випадки відповідають очікуваним результатам.

Також, необхідно перевірити обмеження певного функціоналу для користувачів, з чим може допомогти Swagger [20].

На (рис. 3.41) зображено спробу отримання проєкту не своєї кампанії.

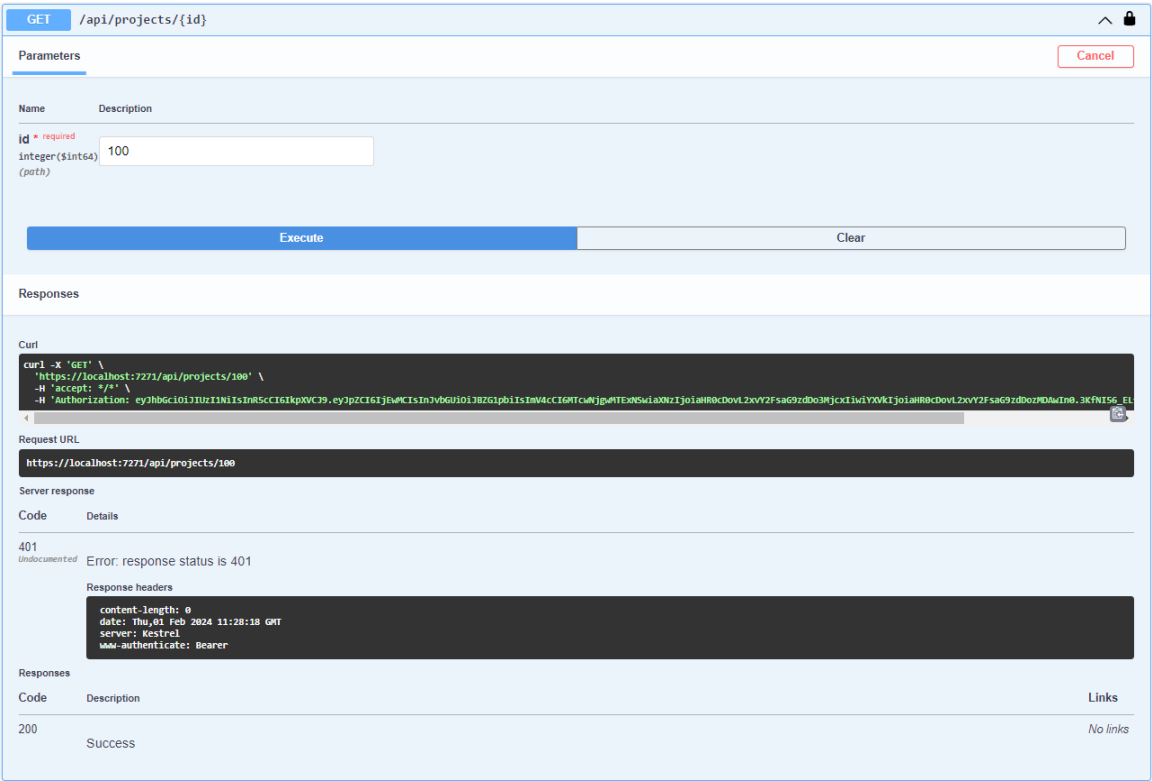
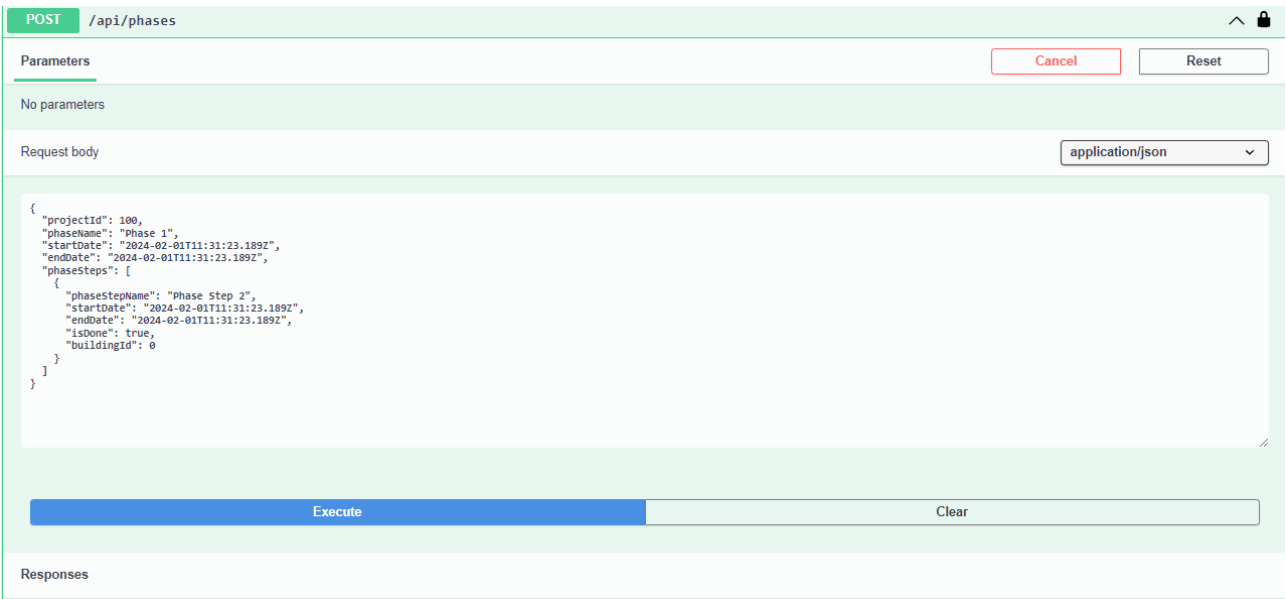


Рис 3.41 Спроба отримання проєкту не своєї кампанії

Цю дію виконувати як і очікувалося не вдалося. Отримано відповідне повідомлення про відсутність доступу до певного проєкту.

Перевіримо спробу створення фази проєкту з невалідними датами (рис. 3.42 – 3.43)



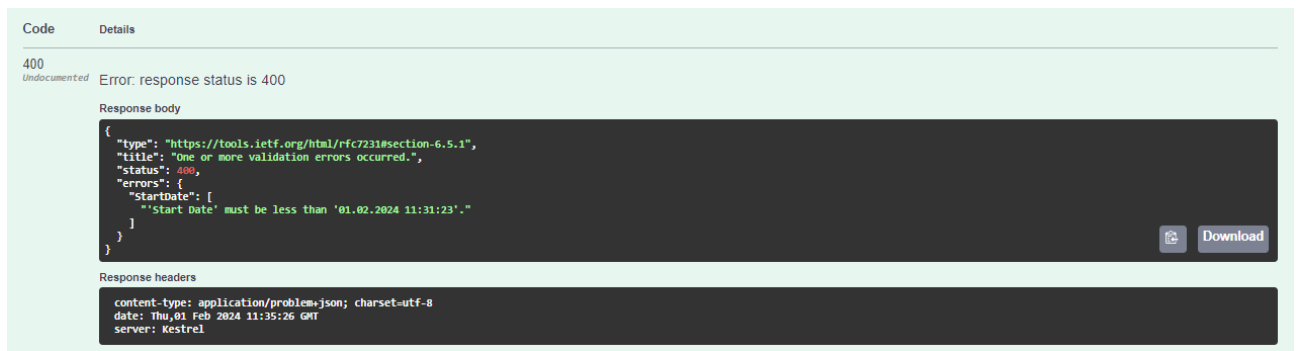


Рис 3.42, 3.43 Спроба створення фази проєкту з невалідними датами

Цю дію виконувати як і очікувалося не вдалося. Отримано відповідне повідомлення про помилки в валідації даних.

Тож, в ході проведення тестування та перевірок було виявлено та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

Висновки до третього розділу

В даному розділі було побудовано та показано діаграму розгортання системи. Також було продемонстровано інтерфейс та порядок взаємодії з системою трекінгу процесів будівництва. Було докладно описано функціональні можливості додатку в залежності від ролі користувача. А також проведено тестування як серверної так і клієнтської частини розробленої системи за допомогою Test Cases та Swagger. В ході проведення тестування та перевірок було виявлено та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було спроектовано та розроблено систему трекінгу процесів будівництва, проведено дослідження предметної області і ринку, з порівняльним аналізом аналогів системи, що автоматизують роботу та управління будівельними компаніями. На основі результатів досліджень та аналізу були визначені вимоги для системи, яка розроблялася.

Проектування системи мало одними з елементів аналіз основних варіантів використання платформи та проектування структури системи. Також було продумано та спроектовано базу даних для зберігання необхідної інформації про відповідні сутності системи, такі як проекти, рекламації, звіти, матеріали, тощо.

Продемонстровано та описано основний потік роботи з системою. Розглянуто основний функціонал, який включає перегляд інформації про проекти, створення звітів по ним, створення рекламацій і процес їх вирішення.

А також показані і описані діаграми про роботу додатку. Наведено приклади з візуалізацією використання інтерфейсу розробленої системи. Система має доволі легкий для сприйняття та освоєння інтерфейс на відміну від аналогів та є такою, яку можуть дозволити собі невеличкі будівельні компанії.

Результатом кваліфікаційної роботи є готова до експлуатації система трекінгу процесів будівництва, яка надає можливості автоматизації процесів будівельних компаній, звітності та управління рекламацією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація .NET [Електронний ресурс] – Режим доступу до ресурсу: <https://dotnet.microsoft.com/en-us/>
2. Документація по MediatR [Електронний ресурс] – Режим доступу до ресурсу: https://docs.discordnet.dev/guides/other_libs/mediatr.html
3. Flux (документація) [Електронний ресурс] – Режим доступу до ресурсу: <https://facebookarchive.github.io/flux/>
4. Офіційна документація NGRX [Електронний ресурс] – Режим доступу до ресурсу: <https://ngrx.io/docs>
5. Офіційна документація Angular [Електронний ресурс] – Режим доступу до ресурсу: <https://angular.io/docs>
6. Кормиш Р. І., Савіцький Р.С. НЕОБХІДНІСТЬ АВТОМАТИЗАЦІЇ МЕНЕДЖЕМЕНТУ ПРОЄКТІВ БУДІВЕЛЬНИХ КОМПАНІЙ. Тези XV міжнародної науково-практичної конференції "Комп'ютерні системи та мережні технології", 25 – 26 квітня 2024. Київ: "НАУ", 2024. С. 89
7. DedalSoft [Електронний ресурс] – Режим доступу до ресурсу: <https://dedalsoft.com.ua/en>
8. BAS Soft [Електронний ресурс] – Режим доступу до ресурсу: <https://www.bas-soft.eu/soft/bas-sectoral/bas-construction-erp/>
9. UGLA [Електронний ресурс] – Режим доступу до ресурсу: <https://ugla.ua/en/cost/>
10. Офіційна документація CQRS (Command Query Responsibility Segregation) [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>
11. Документація HTML5 [Електронний ресурс] – Режим доступу до ресурсу: <https://webplatform.github.io/docs/html/html5/>
12. Офіційна документація Sass [Електронний ресурс] – Режим доступу до ресурсу: <https://sass-lang.com/>
13. Офіційна документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://devdocs.io/javascript/>

- 14.Офіційна документація C# [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/>
- 15.Офіційна документація ASP.NET [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/>
- 16.Офіційна документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/>
- 17.Документація RESTful API [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.digitalocean.com/glossary/restful-api/>
- 18.Офіційна документація SignalR [Електронний ресурс] – Режим доступу до ресурсу: <https://dotnet.microsoft.com/en-us/apps/aspnet/signalr>
- 19.Test Cases [Електронний ресурс] – Режим доступу до ресурсу <https://luxequality.com/blog/how-to-write-test-cases-for-software-testing/>
- 20.Офіційна документація Swagger [Електронний ресурс] – Режим доступу до ресурсу: <https://swagger.io/docs/>
- 21.Документація SPA [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/entra/identity-platform/index-spa>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 24 – ПЗ	
						78

Таблиця А.1

Ім'я варіанту використання: Реєстрація компанії		Рівень важливості: критичний
Основний актор: Адмін		
Інші учасники: Відсутні		
Короткий опис:	Реєстрація певної компанії в системі після заключення договору з компанією	
Передумови:	Ініціалізація в системі	
Основний потік:	Адмін має окрему сторінку, де він може бачити всі компанії, які підписали договір з платформуою та були зареєстровані адміном. Для реєстрації нової компанії адмін повинен заповнити та надіслати форму з певною інформацією про компанію. На сторінці передбачений інтерфейс для реєстрації нової компанії у системі.	
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.	

Таблица А.2

Ім'я варіанту використання: Реєстрація оперативного менеджера компанії		Рівень важливості: дуже високий
Основний актор: Адмін		
Інші учасники: Оперативний менеджер		
Короткий опис:	Цей варіант використання дозволяє адміну додати оперативного менеджера компанії до створеної компанії	

Передумови:	Ініціалізація в системі, наявність створеної компанії в системі
Основний потік:	Після додавання компанії на платформу адмін додає оперативного менеджера команди, створюючи першого користувача компанії з правами Оперативного менеджера. Йому надсилається логін та згенерований пароль на особистий поштовий ящик. Для проведення даних операцій на сторінці компанії є інтерфейс додавання користувачів
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.

Таблиця А.3

Ім'я варіанту використання: Реєстрація інших виконавців		Рівень важливості: високий
Основний актор: Оперативний менеджер		
Інші учасники: Менеджер проєкту, бригадир, головний інженер		
Короткий опис:	Надає менеджеру можливість додати бригадирів, менеджерів та інженерів	
Передумови:	Ініціалізація в системі, наявність доступу до компанії в системі	
Основний потік:	Даний варіант використання дозволяє Менеджеру реєструвати інших учасників компанії. Менеджер має функціонал для запрошення колег в профілі компанії, де він заповнює форму та після її відправлення користувачам надсилається запрошення на платформу з логіном та згенерованим паролем.	

Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.
---------------------------------------	---

Таблиця А.4

Ім'я варіанту використання: Створення проєкту		Рівень важливості: високий
Основний актор: Оперативний менеджер		
Інші учасники: відсутні		
Короткий опис:	Створення нових проєктів	
Передумови:	Ініціалізація в системі, наявність доступу до компанії в системі	
Основний потік:	Даний варіант використання дозволяє Менеджеру створювати проєкти. Для цього на сторінці проєктів компанії передбачений інтерфейс додання проєкту, де менеджер повинен заповнити відповідну форму. Після додання проєкту менеджер має змогу переглянути створений проєкт на даній сторінці.	
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.	

Таблиця А.5

Ім'я варіанту використання: Редагування інформації про компанію		Рівень важливості: середній
Основний актор: Оперативний менеджер		
Інші учасники: відсутні		
Короткий опис:	Дає змогу змінювати інформацію про компанію	

Передумови:	Ініціалізація в системі, наявність доступу до компанії в системі
Основний потік:	Даний варіант використання дозволяє Менеджеру редагувати інформацію про компанію. Для цього на сторінці профілю компанії передбачений відповідний інтерфейс, де оперативний менеджер має змогу редагувати дані про компанію.
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.

Таблиця А.6

Ім'я варіанту використання: Додання фаз та елементів забудови для виконання проєкту		Рівень важливості: середній
Основний актор: Оперативний менеджер		
Інші учасники: відсутні		
Короткий опис:	Менеджер додає фази виконання проєкту та елементи, які будуть побудовані	
Передумови:	Ініціалізація в системі, наявність доступу до проєкту	
Основний потік:	Даний варіант використання дозволяє Менеджеру додавати фази виконання проєкту та елементи забудови. Для цього на сторінці проєкту передбачений відповідний інтерфейс, де менеджер має змогу додати фази на сторінці timeline та елементи забудови на сторінці проєкту. Для цього потрібно заповнити відповідні форми за допомогою відповідного інтерфейсу.	

Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.
---------------------------------------	---

Таблиця А.6

Ім'я варіанту використання: Додання фаз та елементів забудови для виконання проєкту		Рівень важливості: середній
Основний актор: Менеджер проєкту		
Інші учасники: відсутні		
Короткий опис:	Менеджер додає фази виконання проєкту та елементи, які будуть побудовані	
Передумови:	Ініціалізація в системі, наявність доступу до проєкту	
Основний потік:	Даний варіант використання дозволяє Менеджеру додавати фази виконання проєкту та елементи забудови. Для цього на сторінці проєкту передбачений відповідний інтерфейс, де менеджер має змогу додати фази на сторінці timeline та елементи забудови на сторінці проєкту. Для цього потрібно заповнити відповідні форми за допомогою відповідного інтерфейсу.	
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.	

Таблиця А.7

Ім'я варіанту використання: Додання ресурсів та їх постачальників		Рівень важливості: середній
Основний актор: Менеджер проєкту		
Інші учасники: відсутні		

Короткий опис:	Цей варіант використання дозволяє додавати компанії, які постачають матеріали та вказувати, які матеріали будуть використовуватися на будівництві
Передумови:	Ініціалізація в системі, наявність доступу до проєкту
Основний потік:	Даний варіант використання дозволяє Менеджеру додавати ресурси та їх постачальників. Для цього на сторінці проєкту передбачений відповідний інтерфейс, де менеджер має змогу додати фази на сторінці ресурсів. Для цього потрібно заповнити відповідну форму за допомогою відповідного інтерфейсу.
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.

Таблиця А.8

Ім'я варіанту використання: Вказання пріоритетності забудови елементу будівлі		Рівень важливості: середній
Основний актор: Головний інженер		
Інші учасники: відсутні		
Короткий опис:	Цей варіант використання дозволяє головному інженеру вказувати, що будувати першим	
Передумови:	Ініціалізація в системі, наявність доступу до проєкту	
Основний потік:	Головний інженер може перейти на сторінку проєкту і внести зміни в пріоритетність забудови певного елементу проєкту. Для цього передбачений інтерфейс з можливістю	

	переміщення та редагування елементів забудови.
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.

Таблиця А.9

Ім'я варіанту використання: Перегляд інформації про поточний стан проєкту		Рівень важливості: середній
Основний актор: Бригадир		
Інші учасники: відсутні		
Короткий опис:	Цей варіант використання дозволяє бригадиру отримувати поточну інформацію про забудову	
Передумови:	Ініціалізація в системі, наявність доступу до проєкту	
Основний потік:	Бригадир може перейти на певний проєкт, до якого йому дали доступ, та переглянути поточний стан проєкту, зміни і пріоритетність забудови певних елементів. Але при цьому не має змоги нічого самотушки змінювати.	
Альтернативні/виключні потоки:	Якщо сталася помилка під час процесу, то необхідно проінформувати користувача за допомогою повідомлення з помилкою.	