НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК 004.4

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційно-реєстрова система зруйнованого майна та інфраструктури»

Виконав:
студент 2 курсу, групи ІА-31мп
Жоган Всеволод Володимирович _____

Керівник:
доцент кафедри ІСТ, к.т.н., доцент
Писаренко Андрій Володимирович _____

Рецензент:
доцент кафедри ІПІ, к.т.н., доцент
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Жоган Всеволод Володимирович

1. Тема дисертації «Інформаційно-реєстрова система зруйнованого майна та інфраструктури», науковий керівник дисертації Писаренко Андрій Володимирович, доцент кафедри ІСТ, затверджені наказом по університету від «08» 11 2024 р. № 5016-с

2. Термін подання студентом дисертації «09» 12 2024 р.

3. Об’єкт дослідження: процес відновлення та реєстрації пошкодженого майна.

4. Вихідні дані: подання заявок у реєстр, відстеження їх статусу, збереження чернеток заявок, можливість відкриття довготривалих справ, базовий адміністративний портал, зручність для людей з обмеженими можливостями, підтримка для багатьох мов.

5. Перелік завдань, які потрібно розробити: проведення огляду та аналізу існуючих рішень; проєктування структури бази даних; створення алгоритму для оцінки рівня руйнувань; розроблення модуля системи; реалізація функціоналу для адміністратора системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: ER-діаграма бази даних; діаграма послідовності: подання заявки; діаграма послідовності: отримання документа з реєстру; діаграма життєвого циклу

заявки; діаграма послідовності: авторизація адміністратора; діаграма послідовності: продовження токену авторизації адміністратором; діаграма послідовності: відкриття справи як результат заявки; діаграма життєвого циклу справи; діаграма із архітектурою системи.

7. Орієнтовний перелік публікацій: не заплановані.

8. Дата видачі завдання 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Огляд та аналіз існуючих рішень за тематикою завдання магістерської дисертації	до 09.09.24	
2	Визначення мети та задач розробки	до 11.09.24	
3	Вибір засобів та технологій	до 16.09.24	
4	Проєктування структури бази даних	до 23.09.24	
5	Розроблення алгоритму для оцінки рівня руйнувань	до 30.09.24	
6	Розроблення модуля системи	до 15.11.24	
7	Розроблення можливостей адміністратора системи	до 15.11.24	
8	Розроблення стартап-проєкту	до 15.11.24	
9	Побудова UML-діаграм	до 15.11.24	
10	Оформлення основної частини	до 25.11.24	
11	Підготовка до попереднього захисту	до 25.11.24	

Студент Всеволод ЖОГАН

Науковий керівник Андрій ПИСАРЕНКО

РЕФЕРАТ

Інформаційно-реєстрова система зруйнованого майна та інфраструктури: 131 с., 30 табл., 43 рис., 10 дод., 18 джерел.

ІНФОРМАЦІЙНО-РЕЄСТРОВІ СИСТЕМИ, ПРОЗОРІСТЬ ДАНИХ, ОПТИМІЗАЦІЯ БЮРОКРАТІЇ ТА ВИТРАТ, АВТОМАТИЗАЦІЯ ДЕРЖАВНИХ ПОСЛУГ, УПРАВЛІННЯ ВІДНОВЛЕННЯМ, ЗАЛУЧЕННЯ ГРОМАДЯН.

Актуальність даної магістерської дисертації полягає у документуванні зруйнованого майна та інфраструктури, що є важливим у контексті відновлення України після військових дій. Створення такої системи сприяє прозорості та ефективності процесів відновлення.

Об’єктом дослідження є процес відновлення та реєстрації пошкодженого майна.

У роботі виконаний аналіз подібних систем, зокрема Реєстру пошкодженого та знищеного майна та Реєстру збитків для України («Register of Damage for Ukraine»), визначені їхні переваги та недоліки, а також сформульовані вимоги до нової системи. Розроблена система з використанням сучасних технологій, таких як TypeScript, React, NestJS, PostgreSQL тощо, що дозволяє забезпечити високу продуктивність та масштабованість. Також вирішені проблеми, пов’язані з обмеженим доступом до офіційних даних, через застосування відкритих джерел та створення власної структури даних. У роботі детально описані процеси тестування системи для оцінки її стабільності та функціональності, а також представлена концепція стартап-проєкту, який дозволяє системі бути впровадженою у масштабах держави.

ABSTRACT

Information and Records System for Destroyed Property and Infrastructure: 131 p., 30 tables, 43 figures, 10 appendixes, 18 sources.

INFORMATION AND RECORDS SYSTEMS, DATA TRANSPARENCY, OPTIMISATION OF BUREAUCRACY AND COSTS, AUTOMATION OF PUBLIC SERVICES, RECOVERY MANAGEMENT, CITIZEN ENGAGEMENT.

The significance of this master's thesis is to document the destroyed property and infrastructure, which is critical in the context of Ukraine's recovery from the war. The creation of such a system contributes to the transparency and efficiency of the recovery process.

The object of the study is the process of restoration and recording of damaged property.

The paper analyses similar systems, in particular the Register of Damaged and Destroyed Property ("Реєстр пошкодженого та знищеного майна") and the Register of Damage for Ukraine, identifies their advantages and disadvantages, and frames requirements for a new system. The system was developed using modern technologies, such as TypeScript, React, NestJS, PostgreSQL, etc., which ensures high performance and scalability. The problems associated with limited access to official government data were also solved by using open sources and creating our own data structure. The paper describes in detail the processes of testing the system to assess its stability and functionality, and presents the concept of a start-up project that allows the system to be implemented on a nationwide scale.

ЗМІСТ

ВСТУП

Війна, яка досі триває в Україні, станом на грудень 2024 року, завдала значної шкоди інфраструктурі країни, майну та життю її людей.

Руйнування виходить за межі безпосереднього фізичного впливу, впливаючи на економічну стабільність, соціальне середовище та довгострокові перспективи відновлення та відбудови.

У цьому контексті потреба у комплексній інформаційній системі, яка буде реєстром зруйнованого майна та інфраструктури, є вкрай важливою. Така система не тільки служитиме інструментом для документування та підзвітності, але й відіграватиме вагому роль у плануванні, розподілі обмежених ресурсів та наданні допомоги постраждалим.

Метою роботи є підвищення ефективності обліку та управління даними про зруйноване майно та інфраструктуру шляхом створення інформаційно-реєстрової системи, яка забезпечує інтеграцію, зручність використання та безпеку даних.

Об’єкт дослідження: процеси обліку, реєстрації та управління інформацією про зруйноване майно та інфраструктуру.

Предмет дослідження: методи, алгоритми та інструменти розроблення інформаційно-реєстрових систем, які забезпечують ефективне управління даними про зруйновані об’єкти.

Для досягнення поставленої мети були поставлені і вирішені наступні задачі:

– аналіз існуючих інформаційних систем для реєстрації та управління даними про зруйноване майно;

– визначення вимог до функціональності, безпеки та інтеграції інформаційно-реєстрової системи;

– проєктування архітектури системи, включаючи базу даних, серверну частину та інтерфейс користувача;

– реалізація основних функціональних модулів системи;

– тестування системи, оцінка її ефективності.

З огляду на масові руйнування – організований та систематизований підхід до документування та управління інформацією про знищене майно є важливим для ефективних зусиль з відновлення.

Окрім нагальних гуманітарних потреб, війна залишає складну спадщину майнових суперечок, проблем власності та потреби в юридичній та бюрократичній («паперовій») ясності.

Коли люди прагнуть заново побудувати своє життя – вони мають мати змогу орієнтуватися в лабіринті проблем, пов’язаних із правами власності, іншими бюрократичними вимогами та доступом до компенсації. Ефективна інформаційна система може оптимізувати ці процеси, забезпечуючи надійні, актуальні записи, які можна використовувати для перевірки позовів, підтримки судових проваджень та забезпечення того, щоб допомога досягала тих, хто її найбільше потребує – найліпше без зайвої бюрократії, яка може бути непосильною для пересічної людини без сторонньої допомоги. Без такої системи значно зростає ризик шахрайства, корупції, неефективного розподілу ресурсів та «вічних» паперових процесів, які, зазвичай, як правило, перекладають на отримувачів допомоги – що ще більше посилює страждання населення.

У ширшому контексті війни та її наслідків, створення системи також відіграє важливу роль у просуванні прозорості та підзвітності.

У зонах збройних конфліктів або в корумпованих державах, відсутність надійних даних може призвести до дезінформації та експлуатації спільних ресурсів.

Забезпечуючи чітку та доступну інформацію, система може допомогти задокументувати реальність руйнувань точно. Це особливо важливо для судів та міжнародних учасників, таких як гуманітарні організації та іноземні уряди, які покладаються на точні дані, щоб приймати обґрунтовані рішення щодо допомоги.

Крім того, розвиток цієї системи має наслідки для справедливості та примирення. Після війн та збройних конфліктах часто виникають питання відповідальності. Точний реєстр зруйнованого майна та інфраструктури може слугувати важливим доказом у постконфліктних судових справах.

1 АНАЛІЗ ПОДІБНИХ СИСТЕМ

Проблема створення ефективної інформаційно-реєстрової системи зруйнованого майна та інфраструктури є одночасно новою та специфічною для України. Враховуючи обставини війни в Україні, існує небагато подібних систем для аналізу. Руйнування, спричинені війною, у поєднанні з потребою в точному документуванні та управлінні збитками представляють низку проблем, які не широко представлені в існуючих системах. Цей дефіцит систем для дослідження та аналізу підкреслює важливість вивчення найбільш релевантних існуючих систем, незважаючи на їх обмежений обсяг, новизну та регіональну застосовність.

У цьому контексті цей розділ досліджує дві ключові системи: «Реєстр пошкодженого та знищеного майна» (адміністратор: Міністерство цифрової трансформації України) – скорочено РПЗМ, який використовується на національному рівні разом із програмою «єВідновлення», та «Реєстр збитків для України» / «Register of Damage for Ukraine» (адміністратор: Рада Європи / Council of Europe) – скорочено RD4U. Обидві системи спрямовані на документування та керування інформацією, пов’язаною зі знищенням майна, пропонуючи основу, на якій Україна може побудувати власне комплексне, повне та цілісне рішення.

РПЗМ – це ініціатива уряду України, яка підтримує зусилля країни щодо відстеження та оцінки збитків, завданих війною. Це важливий компонент програми «єВідновлення», яка спрямована на спрощення процесу оцінки майна та компенсації для постраждалих осіб [1].

На міжнародному рівні RD4U служить ширшій меті, збираючи дані про руйнування з метою сприяння міжнародній підзвітності та є інструментом для майбутнього запровадження механізму репарацій та відшкодування завданої шкоди.

Хоча обидві системи все ще розвиваються з точки зору своїх технічних можливостей, вони представляють фреймворки з перспективними різноманітними функціями.

Однак, незважаючи на заплановані функціональні можливості, обидві системи наразі стикаються з обмеженнями, які частково перешкоджають їх ефективності. Ці

недоліки (технічний код обох систем закритий) будуть досліджені нижче, висвітлюючи проблеми, з якими стикається кожен реєстр під час виконання своїх цілей, і наслідки для позивачів, які прагнуть справедливості та компенсації.

Розуміння операційної динаміки цих систем важливе для визначення потенційних удосконалень та забезпечення того, щоб постраждалі отримували необхідну підтримку під час та після війни.

1.1 Реєстр пошкодженого та знищеного майна

Як було зазначено вище, Реєстр пошкодженого та знищеного майна – це система, призначена для обліку інформації про пошкоджене або знищене майно та інфраструктуру. Реєстр створений за ініціативи уряду України, адміністратором якого є Міністерство цифрової трансформації України. На момент написання роботи Реєстр використовується дуже щільно із державною програмою, запровадженою урядом України, «єВідновлення» – тому будемо розглядати їх разом.

Більшість функціоналу Реєстру пошкодженого та знищеного майна, який доступний для звичайного користувача (мається на увазі, що «звичайний користувач» це цільова аудиторія – постраждалі), реалізовано через сервіс державних послуг «Дія».

За допомогою «Дії» звичайні користувачі можуть подавати заявки про пошкоджене або знищене майно, отримувати сповіщення про статус своїх заявок, а також подавати заявки на відшкодування шкоди в межах програми «єВідновлення».

Для аналізу функціоналу Реєстру пошкодженого та знищеного майна використані відкриті джерела, такі як офіційні веб-сайти, документація, звіти та інші публікації, що доступні для широкої аудиторії.

Опис системи, у тому числі із частковим технічним описом, включно із функціоналом, який реалізований «на папері», представлений в двох документах, а саме:

– Закон України від 23 лютого 2023 року № 2923-IX «Про компенсацію за пошкодження та знищення окремих категорій об’єктів нерухомого майна внаслідок

бойових дій, терористичних актів, диверсій, спричинених збройною агресією Російської Федерації проти України, та Державний реєстр майна, пошкодженого та знищеного внаслідок бойових дій, терористичних актів, диверсій, спричинених збройною агресією Російської Федерації проти України»;

– Постанова Кабінету Міністрів України (Порядок) від 13 червня 2023 року № 624 «Деякі питання забезпечення функціонування Державного реєстру майна, пошкодженого та знищеного внаслідок бойових дій, терористичних актів, диверсій, спричинених збройною агресією Російської Федерації проти України».

Згідно з Законом України «Про компенсацію за пошкодження та знищення окремих категорій об’єктів нерухомого майна», Реєстр пошкодженого та знищеного майна виконує ключову роль у зборі, накопиченні, обліку, обробці, зберіганні та захисті інформації про об’єкти нерухомого майна, пошкоджені або знищені внаслідок бойових дій, терористичних актів, диверсій, спричинених збройною агресією. Ця система забезпечує облік не тільки об’єктів, але й даних про осіб, майно яких зазнало пошкоджень, розмір матеріальної шкоди, а також інформацію про компенсації, що надаються за такі пошкодження чи знищення майна. Реєстр також включає дані щодо фінансування відновлення пошкодженого або знищеного майна, що дозволяє уряду забезпечувати прозорий та систематизований процес відшкодування збитків та відновлення об’єктів житлової нерухомості та іншої інфраструктури [1].

Функціональні можливості Реєстру пошкодженого та знищеного майна дуже різноманітні та забезпечують збирання, накопичення, обробку та зберігання інформації про пошкоджене та знищене майно внаслідок бойових дій, терористичних актів або інших подібних подій. Реєстр виконує функцію єдиної державної інформаційно-комунікаційної системи, яка допомагає у впорядкуванні інформації та її аналізі для подальшого використання в процесі надання компенсацій або відновлення майна.

Реєстр складається з кількох основних компонентів із наступним функціоналом:

– база даних Реєстру служить основним сховищем інформації, включаючи відомості про об’єкти нерухомості, їхній стан, власників, а також про збитки та компенсації;

– модуль обробки та зберігання даних відповідає за обробку нових даних, їхню перевірку, зберігання та захист інформації, що міститься в Реєстрі. Він забезпечує надійне зберігання даних та їх доступність для користувачів;

– електронний кабінет користувача надає можливість реєструвати пошкоджене або знищене майно, переглядати та оновлювати інформацію про свої об’єкти, а також подавати заявки на компенсацію. Це зручний інтерфейс, що дозволяє власникам майна самостійно керувати інформацією про їхні об’єкти;

– електронний кабінет адміністратора призначений для адміністраторів системи, які будуть управляти нею, зокрема для верифікації даних та модерації інформації;

– модуль збирання, обробки та зберігання геопросторових даних дозволяє інтегрувати інформацію з геопросторовими даними, що включають картографічну інформацію та дані з дистанційного зондування Землі. Завдяки цьому стає можливим візуалізувати розташування пошкодженого або знищеного майна на карті;

– інтерактивний аналітичний дашборд забезпечує можливість аналізувати дані у режимі реального часу, у тому числі у вигляді візуалізації інформації, що міститься у Реєстрі;

– веб-сайт Реєстру пошкодженого та знищеного майна забезпечує відкритий доступ до публічної, знеособленої інформації про пошкоджене та знищене майно в режимі реального часу;

– модуль звітів відповідає за формування звітів на основі даних, внесених до Реєстру. Він дозволяє генерувати статистичні дані, звіти про стан компенсацій, а також іншу аналітичну інформацію, що є корисною для користувачів, адміністаторів системи та інших зацікавлених сторін [2].

Під час аналізу виявлено, що на поточний момент основні можливості для звичайних користувачів включають подання заявок про пошкоджене або знищене майно, отримання сповіщень про статус своїх заявок, а також можливість подання заявок на відшкодування шкоди в межах програми «єВідновлення».

Однак, залишаються відкритими питання щодо реалізації повного функціоналу, описаного в нормативних документах.

Подання заявок про пошкоджене або знищене майно наразі є єдиною формою взаємодії користувача з Реєстром, яка доступна через сервіс «Дія» (https://diia.gov.ua/services/poshkodzhene-majno). Розглянемо (на прикладі додатку «Дія»), що саме бачить користувач під час подання заявки, які кроки він проходить та яку інформацію може отримати в процесі.

Для успішного подання заявки потрібно виконати декілька послідовних кроків [3].

Спочатку потрібно оновити додаток «Дія» до останньої версії. Це необхідно для забезпечення доступу до всіх актуальних послуг та функцій, які надаються через додаток.

Потім, другим кроком, слід увійти в додаток «Дія». Після авторизації необхідно обрати розділ «Послуги», знайти пункт «Пошкоджене майно» і натиснути кнопку «Почати».

Далі потрібно ввести дані про об’єкт нерухомості. Якщо інформація про об’єкт уже міститься в реєстрі, система підтягне її автоматично. У разі відсутності даних, користувач вводить їх вручну, зокрема вказує тип майна, його площу, область, район, населений пункт та точну адресу.

Після цього користувач має описати стан свого майна. Потрібно вказати, у якому стані перебуває майно, скільки осіб проживало в об’єкті на момент пошкодження, і позначити, якщо майно належить до об’єктів культурної спадщини.

На наступному етапі необхідно надати детальний опис пошкоджень. Користувач вказує дату та час події, описує характер пошкоджень і, за можливості, додає світлини, що підтверджують стан об’єкта.

Потім необхідно вказати контактну інформацію, зокрема номер телефону та електронну пошту. Це забезпечує можливість зворотного зв’язку та отримання сповіщень про статус заявки.

Нарешті, користувач має перевірити введені дані на достовірність. Після перевірки потрібно підтвердити їх і надіслати заявку. Готову заявку можна зберегти у форматі PDF або отримати на електронну пошту, що дозволяє мати копію документа для власного архіву.

Процес подання заявок про пошкоджене або знищене майно через сервіс «Дія» було досліджено в рамках аналізу. Важливо відзначити, що більшість коментарів користувачів, залишених під відео [4], вказують на те, що технічні проблеми системи є рідкісними та зазвичай не становлять суттєвих труднощів.

Однак значна частина скарг стосується бюрократичних аспектів процесу. Відповіді на заявки часто затримуються, а комісії, які повинні оцінити завдану шкоду, можуть приходити з великим запізненням. Це створює додаткові складнощі для користувачів, які вже пережили негативний досвід через знищення майна.

Переваги процесу подання заявок про знищене майно через платформу «Дія» включають кілька аспектів:

– основною перевагою є синхронізація даних з іншими реєстрами. «Дія» інтегрована з різними реєстрами, що забезпечує доступ до вже наявних даних. Це дозволяє автоматично підтверджувати інформацію та зменшує необхідність повторного введення даних, що значно спрощує процес подання заявки та знижує ймовірність помилок;

– довільний ввід даних. У випадку, якщо необхідної інформації немає в наявних реєстрах, система дозволяє користувачам вручну ввести необхідні дані. Це забезпечує більшу гнучкість;

– зрозумілість процесу заповнення. Процес подання заявок спроєктовано таким чином, що заповнення форми є інтуїтивно зрозумілим для користувачів. Це знижує кількість помилок та допомагає уникнути непорозумінь під час заповнення заявок;

– можливість зберігати заяву як чернетку. Це є корисним, якщо з будь-яких причин (не тільки технічних), необхідно продовжити заповнення заявки пізніше.

Проте, система має і певні недоліки:

– бюрократія. Незважаючи на автоматизацію та цифрові інструменти, бюрократичні перепони залишаються досі важливою проблемою. Затримки в обробці заявок та довгі терміни очікування на відповідь від комісій продовжують створювати труднощі для користувачів;

– закритість реєстру (хоча відповідно Закону [1] він має бути відкритим). Закритість Реєстру створює значні перешкоди для власників пошкодженого чи

знищеного майна. Вони не мають змоги перевірити (публічно, не через «Дію»), чи внесено інформацію про їхнє майно до Реєстру, чи правильно зафіксовано обсяг пошкоджень, та чи прийнято рішення щодо відшкодування збитків. Це особливо проблематично для бізнесу, де процес фіксації пошкоджень часто затягується. Наприклад, підприємець із Запоріжжя Максим Дрозденко зіткнувся з значними затримками у отриманні акту обстеження свого пошкодженого майна та подальшими труднощами у процесі оцінки збитків [5].

1.2 Реєстр збитків для України

Реєстр збитків для України (Register of Damage for Ukraine, RD4U) є інформаційно-реєстровою системою, створеною для збору заяв про збитки, втрати та шкоду, завдані агресією Російської Федерації проти України. Основною функцією реєстру є прийняття та систематизація таких заяв, поданих як фізичними та юридичними особами, так і державними органами України. Важливою складовою роботи реєстру є зберігання доказів, які підтверджують ці заяви, з метою їх подальшого використання у процесі виплат компенсацій.

Основні завдання Реєстру включають приймання заяв, їх класифікацію та систематизацію відповідно до встановлених критеріїв. Після оцінки та визнання прийнятності заяви, вона вноситься до Реєстру для подальшого розгляду. Однак, важливо зазначити, що Реєстр не виконує функції суду або компенсаційного фонду. Він не оцінює суми збитків і не призначає виплати. Реєстр є першим кроком до створення міжнародного компенсаційного механізму, який надаватиме відшкодування у майбутньому [6].

Реєстр збитків для України офіційно розпочав прийом заяв 2 квітня 2024 року. На початковому етапі система приймає (фактично є реалізованим) заяви лише щодо пошкодженого та знищеного житлового нерухомого майна (категорія A3.1) – це єдина, поки що доступна, взаємодія користувача з Реєстром на даний момент. Приймає ці заяви система через сервіс державних послуг «Дія» (https://diia.gov.ua/services/RD4U) – «Дія» в даному випадку є посередником між Реєстром та користувачем.

Розглянемо (на прикладі додатку «Дія»), що саме бачить користувач під час подання такої заявки, що він надає, яка інформація вже є, та яку інформацію може отримати під час подання [7].

Спочатку користувач оновлює додаток «Дія» до останньої версії. Після цього переходить до розділу «Сервіси» та обирає плитку «Відшкодування майна».

Після цього користувач обирає майно, на яке можна подати заяву, при цьому вибір майна обмежений (тільки те, яке перевірила комісія, створена українським урядом; тільки те майно, яке є приватизоване та внесене в Реєстр). Обравши майно, користувач натискає на кнопку «Подати заяву».

Оскільки подати заяву можна тільки на те майно, яке є зареєстроване та внесене в Реєстр речових прав, то більшість даних уже є заповненими за користувача, проте в розділах «Дані про нерухомість» (інформація про нерухомість, наприклад, різні номери з кадастрів/реєстрів, адреса, площа, кількість мешканців і т.д), «Право власності» (коли набули право власностні, частка власності, і т.д), «Акти та звіти» (акти з тих комісій, про які було згадано абзацом вище), «Місце проживання» (на даний момент часу) доступна опція «Дані некоректні» / «Проживаю за іншою адресою», яка, відповідно до назви, має давати можливість виправляти неправильні дані.

Також користувач заповнює бажаний розмір компенсації, а саме: ринкова вартість нерухомості у гривні, сума витрачена на ремонт у гривні, бажаний загальний розмір компенсації у гривні. При цьому усі ці три поля не є пов’язаними між собою (не є залежними від введених значень в інших полях), тому, наприклад, вартість нерухомості плюс гроші витрачені на ремонт можуть бути меншими, а ніж бажаний розмір компенсації.

Користувач може додати додаткові матеріали (наприклад, підтвердження суми збитків, пошкоджень, ціни нерухомості, вартості ремонту, або можна добавити довільний документ) – що робить заяву більш гнучкою для того, щоб краще розібратися в ситуації в майбутньому; додаткові матеріали можна додавати навіть після надсилання заяви до Реєстру.

Також користувач надає контактні дані, щоб з ним могли зв’язатися з Реєстру.

Після цього користувач перевіряє заяву та відповідно підписує або скасовує заяву.

Протягом усього заповнення заява зберігається як чернетка.

Хоча вказано, що заявник може передавати свої «Цифрові повноваження» з подачі заявки для іншої особи (не обов'язково писати доручення на неї), проте в «Дії» такої можливості немає, повідомлення для користувача, що така функція буде реалізована пізніше, також немає.

Проте існує значно ширший перелік категорій збитків, заяви за якими можуть бути внесені до Реєстру, відповідно до затверджених категорій [8]:

A. Заяви фізичних осіб:

A1. Вимушене переміщення.

A2. Порушення особистої недоторканності.

A3. Втрата майна, доходу або засобів до існування.

A4. Втрата доступу до державних послуг.

B. Заяви з боку держави Україна:

B1. Пошкодження або знищення майна.

B2. Втрата культурної та релігійної спадщини.

B3. Шкода навколишньому середовищу.

B4. Державні витрати на гуманітарну підтримку.

B5. Розмінування та очищення територій.

C. Заяви юридичних осіб:

C1. Пошкодження або знищення майна.

C2. Втрата культурної спадщини.

C3. Економічні втрати бізнесу.

Преваги, Реєстру є наступними:

– однією з переваг Реєстру збитків для України є інтеграція з вже наявними реєстрами через «Дію». Це дозволяє значно спростити процес подання заяв, оскільки заявникам не потрібно вручну збирати та завантажувати документи – багато даних автоматично підтягуються з наявних державних баз. Така автоматизація економить час та зусилля громадян, мінімізуючи бюрократичні процедури;

– наявна можливість додавати додаткові матеріали (у тому числі після того, як заяву було надіслано в Реєстр), що забезпечує більшу гнучкість при розгляді заяви;

– наявне збереження заяви в чернетку (до 5 днів), що додає гнучкості для користувача (а не тільки при технічних проблемах) обирати час для заповнення заявки;

– категорії заяв є надзвичайно гнучкими, що дозволяє охоплювати широкий спектр збитків та втрат, пов’язаних із війною. Гнучкість також полягає в можливості коригування та доповнення заяв, що дозволяє подавати додаткові дані та докази у разі необхідності;

– можливість призначати «Цифрового представника» для подання заяви без необхідності оформлення додаткових юридичних документів у нотаріуса. Це позбавляє заявників додаткових витрат на паперову роботу, зменшуючи фінансові навантаження. Представником може бути будь-яка довірена особа, а не обов’язково юрист, що спрощує процес для звичайних громадян, особливо для тих, хто не розбирається в цифрових послугах, проте має довірену особу. На даний час поки що не доступна;

– вся інформація на сайті Реєстру написана простою та зрозумілою мовою, без складних юридичних термінів. Це робить Реєстр доступним для кожного, хто хоче подати заяву, не маючи спеціальних знань у галузі права.

Недоліки системи:

– неможливість додати представника до заяви, при заповнені через «Дія» (хоча це має бути реалізовано пізніше) – повідомлення про те, що функція буде потім реалізована – відсутнє;

– не суттєвим недоліком є те, що реалізована можливість подати лише одну категорію заяви (А3.1). Однак цей аспект не є систематичним недоліком.

Висновки до розділу 1

У розділі «Аналіз подібних систем» розглянуті дві основні інформаційно-реєстрові системи: «Реєстр пошкодженого та знищеного майна» (РПЗМ), створений

Міністерством цифрової трансформації України, та «Реєстр збитків для України» (RD4U), який адмініструє Рада Європи. Обидві системи спрямовані на документування та управління даними про руйнування майна, що спричинені війною в Україні.

Досліджені не тільки функціональні можливості цих систем, але й частково технічні та правові аспекти, відображені в нормативних документах, які регулюють їх роботу. Особливо важливо було звернути увагу на розбіжність між тим, що декларується, та фактично реалізованими можливостями систем. Наприклад, в RD4U, попри опис одного функціоналу, в реальності користувачі стикаються з обмеженими можливостями коли подають заявки через «Дію».

Що стосується правового аспекту, в нормативних документах закріплені зобов'язання щодо прозорості та доступності даних, однак на практиці це не завжди виконується, наприклад, публічний доступ до інформації в РПЗМ обмежений, що впливає на довіру до системи. Те ж саме стосується і RD4U, де передбачено широкий спектр категорій збитків для реєстрації, але на поточному етапі реалізована лише одна категорія – пошкоджене житлове майно. Це також свідчить про відставання технічної реалізації від правових зобов'язань.

Перевагами Реєстру пошкодженого та знищеного майна є інтеграція з державною системою «Дія», що спрощує подання заявок на компенсацію, а також автоматичне підтягування даних з інших реєстрів, що мінімізує помилки та заощаджує час. Система пропонує зберігати чернетки, що є додатковою перевагою. Водночас її недоліками є закритість даних, які вносяться в Реєстр.

Реєстр збитків для України також має низку переваг, зокрема інтеграцію з іншими реєстрами через «Дію» при заповнені заявки, можливість додавання матеріалів до заяв (різних матеріалів – що дуже гнучко). Однак система обмежена у прийомі заяв тільки щодо житлової нерухомості. Відсутня можливість додавати представника. А також, поки що, система не дозволяє подання заяв на нерухомість, яка не внесена в Реєстр речових прав або не оглянута комісіями.

На основі аналізу переваг та недоліків цих систем сформуємо вимоги до розроблюваної системи.

2 АНАЛІЗ ВИМОГ ДО СИСТЕМИ

2.1 Обмеження та обсяг системи

Перелік проблем, які інформаційно-реєстрова система зруйнованого майна та інфраструктури зможе вирішити:

а) надання прозорої звітності та аудиту. Система дозволить автоматично збирати, зберігати та аналізувати дані про зруйноване майно та інфраструктуру, забезпечуючи можливість контролю за процесом відновлення на кожному етапі;

б) залучення громадськості до відновлення. Завдяки публічному доступу до даних про зруйноване майно та хід робіт, населення матиме більше можливостей для участі у процесі відновлення, що сприятиме активнішій взаємодії громадян з органами влади;

в) покращення державних сервісів;

1) система дозволить більш ефективно координувати державні послуги з відновлення, оскільки кожен етап робіт, фінансування та прогрес будуть відображені в реальному часі. Це допоможе швидко виявляти затримки та усувати проблеми;

2) можливість відслідковувати статус виконання робіт та виявляти зони, де виникають проблеми або затримки. Це зменшить можливості для зловживань;

3) завдяки автоматизації збору інформації та її обробки, скоротиться час на планування та реалізацію відновлювальних заходів, що підвищить ефективність управління.

Перелік проблем, які система не зможе вирішити:

а) корупція. Хоча система допоможе зменшити корупційні ризики через прозорість, вона не зможе повністю викорінити корупцію на рівні державних органів та підприємств – це робота антикорупційних правоохоронних органів та чіткого законодавства;

б) проблеми в державному управлінні, некомпетентність або низька кваліфікація співробітників;

1) система не зможе покращити рівень кваліфікації людей, які працюють з нею. Некомпетентні дії посадовців, менеджерів або працівників можуть затримувати чи ускладнювати процеси навіть за наявності ідеальної системи;

2) система не може навчити людей кооперуватися чи взаємодіяти між собою ефективно. Відсутність комунікації та співпраці між різними органами чи громадянами залишиться проблемою (якщо така є), якщо не буде розвинута відповідна культура;

в) фінансові та політичні обмеження;

1) навіть з ідеальною системою, без належного фінансування чи політичної волі процес відновлення може зупинитися або йти повільніше, ніж очікується;

2) система не зможе компенсувати відсутність матеріальних, людських або технологічних ресурсів, необхідних для відновлення зруйнованих об’єктів.

2.2 Формування вимог до системи

Оскільки, під час написання цієї роботи, відсутній доступ до державних реєстрів, система розроблена з урахуванням цієї обмеженості.

Вимоги поділені на функціональні та нефункціональні; на MVP (обов’язкові вимоги для функціонування системи) та post-MVP (вимоги, що можуть бути додані після запуску системи) – бо час (та ресурси) на написання цієї роботи обмежений.

MVP:

– подання заявки – тільки для житлового майна (функціональна). Користувачі повинні мати можливість подати заявку на реєстрацію зруйнованого/пошкодженого майна;

– відстеження статусу заявки (функціональна). Користувачі мають відслідковувати стан своїх заявок у реальному часі через систему;

– відкриття справ (функціональна). Система повинна дозволяти створювати нові справи (довготривалі процеси) на основі поданих заявок;

– гнучкість до різних типів заявок (функціональна). Система повинна підтримувати (з плануванням на майбутнє) різні види заявок (житлова, інфраструктура, комерційна нерухомість тощо);

– збереження чернеток заявок (функціональна). Користувачі повинні мати можливість зберігати чернетки заявок та повертатися до їх заповнення пізніше;

– адміністративний портал (функціональна);

– доступність (нефункціональна, зручність використання). Система повинна бути доступною для людей з обмеженими можливостями та бути зручною для використання на різних пристроях та платформах;

– підтримка української та англійської мов (нефункціональна, зручність використання).

Post-MVP:

– реалізація інших процесів повного циклу: розгляд справи, суди, виплати, звітування тощо (функціональна);

– розширення переліку майна та інфраструктури, яке може бути внесене в реєстр (функціональна);

– додання порталу відкритих даних (функціональна);

– автоматична перевірка заявок (функціональна);

– можлива інтеграція з державними реєстрами (функціональна);

– мультифакторна аутентифікація (нефункціональна, безпека);

– вхід за допомогою ключа (Passkey) (нефункціональна, безпека);

– інтеграція з картографічними сервісами (функціональна);

– аналітичний модуль для органів влади (функціональна);

– архівування старих заявок (нефункціональна, продуктивність);

– інструменти для боротьби з шахрайством (нефункціональна, безпека);

– підтримка більшої кількості мов, наприклад, румунська чи угорська (нефункціональна, зручність використання);

– підпис документів за допомогою PGP з можливістю перевірки та публікації публічних ключів (поєднано з 11) (нефункціональна, безпека);

– перетворення електронних документів в паперові з можливістю читати документ (паперовий) машиною (функціональна).

Висновки до розділу 2

У розділі «Аналіз вимог до системи» визначено, що система спрямована на підвищення прозорості, покращення координації державних послуг та залучення громадськості до процесу відновлення. Водночас, вона має певні обмеження, зокрема щодо боротьби з корупцією, підвищення кваліфікації кадрів та подолання фінансових і політичних перешкод.

Формування вимог до системи розділене на дві категорії: MVP (мінімальний життєздатний продукт), що включає критично необхідні функції, та post-MVP, які можна реалізувати після запуску.

Основні функціональні вимоги стосуються подання та відстеження заявок, підтримки різних типів заявок та створення адміністративного порталу.

Нефункціональні вимоги включають доступність для різних користувачів та забезпечення багатомовності. Post-MVP вимоги охоплюють ширші можливості, такі як інтеграція з державними реєстрами, аналітичні модулі та інструменти безпеки.

3 РОЗРОБЛЕННЯ СИСТЕМИ

У рамках цього розділу представлені ключові технічні рішення під час розроблення інформаційно-реєстрової системи зруйнованого майна та інфраструктури. Пояснимо вибір тих, чи інших технічних рішень, враховуючи специфіку завдання та ресурсні обмеження.

Основною мовою програмування обрана мова TypeScript, оскільки вона є сильно типізованою надбудовою над JavaScript, що дає можливість уникати типових помилок на етапі компіляції. Це полегшує процес розроблення системи, робить код читабельнішим та допомагає зменшити кількість потенційних багів [9].

Використання TypeScript є доцільним як на клієнтській частині системи (фронтенд), так і на серверній частині системи (бекенд), що забезпечує уніфікацію підходу до розроблення й полегшує підтримку системи.

Для розроблення клієнтської частини обрано фреймворк React. React є однією із найбільш популярних та широко використовуваних бібліотек для розроблення інтерфейсів користувача. Основна причина вибору React – це його активна спільнота та велика кількість шаблонного коду, що може значно прискорити розроблення.

На даний момент більшість розробників використовують React як частину фреймворку Next.js. Використання Next.js виправдане швидкістю розроблення. Оскільки Next.js надає багато вбудованих рішень, це знижує потребу у написанні додаткового коду та допомагає економити час на налаштування.

Варто зазначити, що деякі можливості Next.js тісно інтегровані з платформою Vercel (хостинг), яка може значно збільшити витрати на хостинг, якщо використовувати її на комерційній основі. Тому для розроблюваної системи використані більш гнучкі рішення для хостингу.

Для серверної частини обраний фреймворк NestJS, що працює на платформі Node.js (версія LTS 20). Вибір на користь NestJS зумовлений кількома факторами [10]:

– модульна архітектура. NestJS надає чітку та структуровану архітектуру, яка базується на принципах MVC (Model-View-Controller). Це робить систему легшою в підтримці та розширенні для інших розробників;

– велика спільнота та документація. Одна з основних переваг NestJS полягає в тому, що він широко використовується та має велику кількість готових рішень. Це дозволяє розробникам швидко знаходити відповіді на запитання або використовувати вже готові бібліотеки для вирішення типових завдань;

– простота підтримки та розвитку системи. Використання широко поширеного фреймворку дозволяє уникнути необхідності документування всіх внутрішніх технічних рішень щодо системи, адже багато аспектів уже описано в офіційній документації. У випадку, якщо вся архітектура вигадується з нуля, це призводить до значної затримки в розробленні та необхідності створювати обширні документи для підтримки у майбутньому.

Для зберігання даних використовується реляційна база даних PostgreSQL. Вибір PostgreSQL базується на таких причинах [11]:

– відкритий код та фінансова незалежність. PostgreSQL є open-source продуктом, тому не вимагає ліцензійних платежів, що дозволяє уникнути фінансових зобов’язань;

– широкий набір функцій. PostgreSQL підтримує складні запити, транзакції, а також містить інструменти для роботи з великими об’ємами даних, що є важливим для системи;

– гнучкість. PostgreSQL розширюється за допомогою додаткових модулів та має підтримку JSON, що корисно для зберігання структурованих даних.

Увесь процес розроблення організований в одному монорепозиторії, що дозволило спростити управління залежностями та забезпечити консистентність коду. Це рішення також дає змогу оптимізувати процес CI/CD у майбутньому та легко підтримувати інтеграцію між різними компонентами системи.

Як бачимо, що основними критеріями вибору технологій є економія часу розробника та оптимізація ресурсів. На даному етапі, як було прописано у вимогах, головною метою є створення MVP, тому велика частина часу буде присвячена не тільки технічній реалізації, а й продуманню користувацького досвіду (UX). Вибрані технології дозволяють зосередитися на розробленні функціоналу, а не на вирішенні низькорівневих технічних проблем, що є вирішальним для успіху проєкту.

3.1 Розроблення архітектури системи

Архітектура системи складається з кількох компонентів, які забезпечують взаємодію між користувачами та системою, зберігання та обробку даних, а також управління документацією. Складовими архітектури є клієнтська частина (фронтенд), серверна частина (API сервер) та база даних, доповнені інтеграцією з хмарним сховищем AWS S3. Наочно з нею можна ознайомитися у Додатку Б.

Клієнтська частина представлена фронтенд-додатком, який відповідає за інтерфейс взаємодії з користувачами. Цей компонент дозволяє публічним користувачам та адміністраторам системи взаємодіяти із системою. Для обміну даними з серверною частиною використовуються HTTP-запити.

Серверна частина системи представлена API сервером. API сервер забезпечує отримання запитів від клієнтської частини, їх обробку та взаємодію з базою даних та зовнішнім сховищем. Він реалізує бізнес-логіку системи.

Для зберігання структурованих даних використовується база даних. Ця база даних є основою для звітності та забезпечення цілісності даних.

Система інтегрується з хмарним сервісом AWS S3 для зберігання завантажених користувачами паперових документів. Це рішення забезпечує масштабованість та надійність зберігання інформації. Публічні користувачі завантажують документи безпосередньо в AWS S3 через клієнтську частину, а API сервер використовує сховище для доступу до цих файлів.

3.2 Налаштування та структура монорепозиторію

Для оптимізації розроблення та спрощення управління залежностями в проєкті, обрана монорепозиторна архітектура з використанням інструмента pnpm.

Pnpm обраний як пакетний менеджер через його ефективність у керуванні залежностями та використання дискового простору. На відміну від npm або yarn, pnpm використовує жорсткі посилання для залежностей, що знижує їх дублювання та значно прискорює процес встановлення пакетів, особливо в умовах великого проєкту.

Функціонал pnpm workspaces дозволяєорганізувати код у кілька окремих пакетів всередині одного монорепозиторію.

Для швидкої та ефективної роботи монорепозиторію інтегрований Turborepo, який значно спрощує процес управління великою кількістю пакетів. Turborepo дозволяє паралельно збирати окремі частини проєкту, використовуючи кешування попередніх результатів збірки та виконання тестів. Це дозволяє суттєво скоротити час, необхідний для CI/CD процесів, особливо при масштабуванні системи.

Монорепозиторій системи організований таким чином, що спрощує підтримку, маштабування та розширення проєкту в майбутньому. Весь код поділений на дві основні категорії: додатки та бібліотеки.

Apps (Додатки) це основні компоненти системи, які відповідають за взаємодію з користувачем або обробку запитів. В розроблюваній системі це:

– frontend (Next.js додаток) – клієнтська частина системи, яка відповідає за інтерфейс користувача;

– backend (NestJS додаток) – серверна частина системи, яка обробляє запити та забезпечує API для фронтенду.

У Packages (Бібліотеки, пакети) зберігатимуться загальні компоненти, які можуть використовуватися в декількох додатках. Наприклад:

– UI компоненти – спільні React-компоненти, які можна використовувати на різних сторінках або в різних модулях;

– утиліти – спільні функції, такі як робота з датами або обробка запитів;

– моделі даних – спільні інтерфейси та типи, які використовуються як на фронтенді, так і на бекенді.

Вищеописаний опис в представлений на рисунку 3.1.

```yaml
packages:
  - "apps/*"
  - "packages/*"

```

Рисунок 3.1 –Yaml-файл структури workspace проєкту

Завдяки використанню pnpm workspaces та Turborepo створена гнучка та ефективна архітектура монорепозиторію, яка дозволяє швидко збирати, тестувати та розширювати систему. Така структура дає можливість легко підтримувати різні частини проєкту та забезпечує масштабованість у майбутньому.

3.3 Повнота даних для розроблення та функціонування системи

Однією з головних проблем при розробленні інформаційно-реєстрової системи зруйнованого майна та інфраструктури є обмежений або відсутній доступ до необхідних даних реєстрів (майнових, територіальних тощо).

Наприклад, відсутня можливість використовувати офіційні реєстри для отримання інформації про населені пункти, їхні вулиці та поштові індекси, що значно ускладнює побудову точного реєстру місцевостей для системи. Хоча ці дані могли б бути доступні через Реєстр територіальної громади [12], проте він є закритим для доступу через API.

Шляхи вирішення проблеми повноти даних.

Розглянуті кілька варіантів вирішення цієї проблеми, кожен з яких має свої переваги та недоліки.

Комерційні API. Наприклад, можна було би скористатися рішенням від Google, таким як Google Place Autocomplete API [13]. Цей API дозволяє отримувати точну інформацію про населені пункти, вулиці та інші географічні дані. Проте, використання такого рішення потребує додаткових фінансових витрат, що є недоцільним, зважаючи, що це соціальний проєкт, та у випадку великих обсягів даних і довгострокової експлуатації системи.

OpenStreetMap (OSM). Також розглянута можливість використання OpenStreetMap через Overpass API. Однак, дані з OSM є неповними або неактуальними, особливо щодо населених пунктів та вулиць (чим менший населений пункт, тим менш актуальні дані – або взагалі відсутні щодо нього) в Україні (наприклад, для Німеччини дані більш якісні).

Крім того, запити через Overpass API є досить складними у налаштуванні та не завжди зручними для масового використання.

Open Source рішення. Врешті-решт, обране рішення на основі відкритих даних з GitHub, що стали основою для розроблення.

Наприклад, знайдений проєкт, де один із розробників, «Valerii Nikitiuk», зібрав та систематизував інформацію про українські населені пункти в форматі JSON [14]. Цей файл містить усі необхідні дані, включаючи назви міст та сіл, їхні поштові індекси, географічні координати та ідентифікатори для інтеграції з іншими системами.

Обробка та інтеграція даних.

Після отримання JSON-файлу здійснені кілька важливих трансформацій. Серед цих трансформацій видалені зайві поля, які не використовуються у подальшій роботі, а також підготовлені структури даних у форматі, оптимальному для подальшого використання у системі. Особлива увага приділена створенню структури, яка забезпечує зручність роботи з цими даними на різних етапах обробки.

Для більшої ефективності у проєкті створено окремий пакет («@prpr/locations»), який спеціально призначений для роботи з такими даними. У цьому пакеті написано код, який дозволяє взаємодіяти з отриманою інформацією, забезпечуючи її коректне оброблення. Завдяки цьому, обробка даних виконується на серверній частині, що дозволяє уникнути надмірного навантаження на клієнтські системи через великий обсяг файлу.

Нижче наведені основні частини коду (разом із поясненнями), які безпосередньо відповідають за роботу з отриманими даними, забезпечуючи їх надійну інтеграцію в систему.

Обробка та парсинг даних. У файлі «parse.ts» використовується бібліотека «zod», яка виконує функцію валідації структури даних перед їх подальшою обробкою. Це особливо важливо, оскільки така валідація гарантує, що отримані дані відповідають заданій моделі. Це, у свою чергу, дозволяє уникнути потенційних помилок під час обробки даних у разі їх непередбачених змін або модифікацій без оновлення відповідного коду.

Більш детальний розгляд цього процесу, включаючи графічну ілюстрацію, наведено на рисунку 3.2.

```
import { z } from 'zod';
import raw from './static/locations.min.json';
import { Location } from './types';
const locationNameSchema = z.record(z.string());

const locationSchema = z.object({
  id: z.string(),
  parent: z.union([z.string(), z.null()]),
  type: z.enum(['VILLAGE', 'DISTRICT', 'STATE', 'COMMUNITY',
    'CITY', 'URBAN', 'SETTLEMENT', 'CAPITAL_CITY',
  ]),
  name: locationNameSchema,
  public_name: locationNameSchema,
  post_code: z.array(z.string()),
  loc: z.object({
    lat: z.union([z.number(), z.null()]),
    lon: z.union([z.number(), z.null()]),
  }),
  meta: z.object({
    osm_id: z.union([z.string(), z.null()]),
    google_maps_place_id: z.union([z.string(), z.null()]),
  }),
});

const locationsSchema = z.array(locationSchema);
const locationsRaw = locationsSchema.parse(raw) as Location[];
```

Рисунок 3.2 – Частина коду для валідації та парсингу даних

Запити (доступ) до даних. У файлі «query.ts» реалізовуються кілька функцій для доступу до даних, таких як отримання всіх записів («getAll»), пошук по ID («getById») та пошук по назві («getByName»).

Приклад функції (частини коду), яка здійснює пошук по назві населеного пункту та повертає обмежену кількість результатів зображено на рисунку 3.3.

```
function getByName<T extends Location>(
 {
  name,
  limit = 10,
 }: {
  name: string | string[];
  limit?: number;
 },
 data: T[] = locationsRaw as T[]
) {
 let result: T[] = [];
 if (!Array.isArray(name)) {
  result = data.filter(
   (v) =>
    v.name["uk"].toLowerCase().includes(name.toLowerCase()) ||
    v.name["en"].toLowerCase().includes(name.toLowerCase())
  );
 } else {
  result = data.filter((v) =>
   name.some(
    (n) =>
     v.name["uk"].toLowerCase().includes(n.toLowerCase()) ||
     v.name["en"].toLowerCase().includes(n.toLowerCase())
   )
  );
 }
 result = result.slice(0, limit);
 return result;
}
```

Рисунок 3.3 – Частина коду, яка здійснює пошук по назві населеного пункту та повертає обмежену кількість результатів

Трансформація даних. У файлі «transform.ts» реалізовуються функції для фільтрації даних та створення повної назви населеного пункту разом з батьківськими елементами (наприклад, область або район).

Наприклад, функція, зображена на рисунку 3.4, генерує повну назву населеного пункту, включаючи всі батьківські локації, що допомагає з точністю ідентифікувати місцезнаходження (коли є однакові назви).

```
const transformToWithFullNameOnRaw = (
  locations: Location[]
): LocationWithFullName[] => {
  return locations.map((v) => {
    return {
      ...v,
      full_name: {
        ...getFullName(v, ["uk", "en"], locations),
      },
    };
  });
};
```

Рисунок 3.4 – Частину коду, яка генерує повну назву населеного пункту

Утиліти для роботи з ієрархією локацій. У файлі «utils.ts» реалізовані ключові утиліти, які допомагають керувати складною ієрархічною структурою локацій.

Наприклад, функція «getParents», зображена на рисунку 3.5, рекурсивно знаходить усі батьківські елементи (наприклад, громада, область чи район) для певної локації. Вона використовується для побудови повної ієрархії місцевості. Це важливо для відображення повних назв або визначення всіх територій, що належать до певного регіону (громади).

```
function getParents(
  data: Location[],
  child: Location,
  filterOutThisTypes: LocationType[] = [],
  parents: Location[] = []
): Location[] {
  const parent = data.find((v) => v.id === child.parent);

  if (!parent) {
    return parents;
  }

  return getParents(
    data,
    parent,
    filterOutThisTypes,
    filterOutThisTypes.includes(parent.type) ? parents : [...parents, parent]
  );
}
```

Рисунок 3.5 – Частина коду з рекурсивною функцію пошуку батьків локації

Також дані з пакету «@prpr/locations» використовуються для наповнення бази даних інформацією про так звані «кластери» – це, по суті, ЦНАПи (Центри надання адміністративних послуг, або ратуші/ради), інформація про які в публічному доступі через API відсутня.

Ці «кластери» є важливою складовою логіки системи, тому потрібно створити та зберігати їх вручну.

Для цього інтегровані дані з пакету «@prpr/locations» у процес наповнення БД. Детальніше про структуру БД можна дізнатися з діаграми Додатку Л.

На основі даних про громади (у організаційній структурі кожна громада має кластер, який обслуговує її) з «@prpr/locations» генеруються «кластери», що представляють собою ЦНАПи (для прикладу) в цих громадах. Громади отримуються через функцію «filterByOnRaw», яка фільтрує місця на основі типу «COMMUNITY». Для кожної громади створюється новий об’єкт класу «Cluster», який представляє ЦНАП.

Також створюються зв’язки між кластерами та місцевими локаціями, які підпорядковуються цим громадам. Для цього використовується функція «findAllChildrenByParent», яка допомагає знаходити всі підлеглі локації для кожної громади.

Після цього створюються об’єкти «ClusterLocation», які зберігають інформацію про зв’язок між кластером та локаціями.

Таким чином, вирішено проблему відсутності доступу до офіційних реєстрів через використання Open Source рішень та створення власної структури для зберігання та обробки даних.

3.4 Розроблення серверної частини системи

3.4.1 Підготовка середовища для розроблення серверної частини

Одним із ключових інструментів для покращення розробницького досвіду (Dev Experience) є використання контейнеризації за допомогою Docker та docker-compose. Це дозволяє не тільки створити ізольоване середовище для роботи, але й легко

переносити його на інші машини, що забезпечує консистентність середовища розробки (між розробниками) та мінімізує потенційні проблеми на різних етапах розроблення.

Сервіси, які створюються в Docker:

– PostgreSQL – основна база даних, що буде використовуватися для зберігання даних системи. Використовується легкий образ postgres:13.11-alpine. Контейнер налаштований через змінні середовища для зручності керування базою даних, а збереження даних забезпечується за допомогою volume;

– PgAdmin – зручний веб-інтерфейс для керування базою даних (наприклад, для дебагу). Доступ до нього можливий через порт 5050, а основні налаштування користувача та пароля задаються через змінні середовища. Приклад docker-compose файлу, який використовується при розробці зображений на рисунку 3.6.

```yaml
version: '3.8'
services:
  db:
    image: postgres:13.11-alpine
    restart: always
    environment:
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    ports:
      - '5432:5432'
    volumes:
      - db:/var/lib/postgresql/data
    networks:
      - db

  pgadmin:
    image: dpage/pgadmin4
    restart: always
    environment:
      - PGADMIN_DEFAULT_EMAIL=admin@example.com
      - PGADMIN_DEFAULT_PASSWORD=admin
    ports:
      - '5050:80'
    volumes:
      - pgadmin:/var/lib/pgadmin
    networks:
      - db

volumes:
  db:
  pgadmin:

networks:
  db:
```

Рисунок 3.6 –Yaml-файл Docker сервісів

3.4.2 Налаштування бази даних на серверній частині системи

Для взаємодії з базою даних на серверній частині можна використовувати різні підходи: від прямого використання драйверів та SQL-запитів до вищого рівня абстракцій, таких як QueryBuilder чи ORM (Object-Relational Mapping). Хоча прямі SQL-запити можуть дати більше контролю, вони ускладнюють розробку та супровід, особливо в TypeScript-середовищі, оскільки не підтримують статичну типізацію та гнучкі інструменти, що покращують розробницький досвід (Dev Experience).

Критерії вибору між «QueryBuilder» та «ORM»:

– QueryBuilder надають більше свободи, оскільки дозволяють будувати SQL-запити програмно, що корисно для складних або специфічних запитів. Проте це може ускладнити підтримку й потребує більше ручної роботи;

– ORM автоматизують багато рутинних завдань, таких як збереження та отримання даних, підтримуючи при цьому статичну типізацію в TypeScript. ORM дозволяє працювати з базою даних через об’єкти, що спрощує розроблення і полегшує підтримку коду, хоча іноді може створювати додаткову «магію», що приховує деталі процесів.

Для проєкту обрані MikroORM, оскільки це ORM, яка забезпечує баланс між контролем та абстракцією.

Він добре інтегрується з TypeScript, надає гнучкі інструменти для роботи з міграціями, підтримує QueryBuilder (завдяки «Knex») та на ньому можна писати прямі запити; а також він зручний у налаштуванні [15].

MikroORM дозволяє зберігати типізовані сутності в базі даних та автоматично генерує SQL-запити на основі взаємодії з цими сутностями [15].

Для інтеграції з PostgreSQL використовується наступна конфігурація MikroORM, зображена на рисунку 3.7, у середовищі NestJS.

Основні налаштування включають імпорт змінних середовища через «ConfigService», які містять інформацію про з’єднання з базою даних.

```
import { Options } from '@mikro-orm/core';
import { Migrator } from '@mikro-orm/migrations';
import { PostgreSqlDriver } from '@mikro-orm/postgresql';
import { SeedManager } from '@mikro-orm/seeder';
import { ConfigService } from '@nestjs/config';
const configService = new ConfigService();

const MikroOrmConfig: Options = {
  dbName: configService.get('MIKRO_ORM_DB_NAME'),
  user: configService.get('MIKRO_ORM_USER'),
  password: configService.get('MIKRO_ORM_PASSWORD'),
  host: configService.get('MIKRO_ORM_HOST'),
  port: configService.get('MIKRO_ORM_PORT'),
  driver: PostgreSqlDriver,
  entities: ['dist/**/*.entity.js'],
  entitiesTs: ['src/**/*.entity.ts'],
  extensions: [Migrator, SeedManager],
  migrations: {
    tableName: 'mikro_orm_migrations',
    path: 'src/persistence/migrations',
  },
  seeder: {
    path: 'src/persistence/seeders',
    pathTs: undefined,
    defaultSeeder: 'DatabaseSeeder',
    glob: '!(*.d).{js,ts}',
    emit: 'ts',
    fileName: (className: string) => className,
  },
};
```

Рисунок 3.7 –Конфігурація MikroORM

Детальний опис конфігурації, зображеній на рисунку 3.7:

– dbName, user, password, host, port. Параметри підключення до бази даних, що отримуються з файлу конфігурації через «ConfigService». Це забезпечує гнучкість при зміні середовищ (development, production);

– PostgreSqlDriver. Драйвер для підключення до PostgreSQL;

– entities та entitiesTs. Шляхи до сутностей, які використовуються MikroORM для створення та управління таблицями в базі даних. Це файли з визначеними моделями даних, які співвідносяться з таблицями в базі;

– міграції (migrations). Налаштування міграцій для версіонування та управління змінами в структурі бази даних. Міграції зберігаються у спеціальній директорії та дозволяють автоматично виконувати зміни в базі.

3.4.3 Налаштування логування на серверній частині системи

Основна мета логування – надання розробникам, адміністраторам системи та інженерам з безпеки детальну інформацію про функціонування системи, її стан, запити та можливі помилки.

Це важливо для діагностики та усунення проблем, виявлення аномалій, моніторингу активності користувачів в реальному часі, вдосконаленню системи, маючи дані про продуктивність, аудиту безпеки тощо.

У випадку інформаційно-реєстрової системи зруйнованого майна та інфраструктури, логування відіграє важливу роль у підтримці її надійності та стабільності, забезпечуючи прозорість операцій та своєчасне виявлення будь-яких проблем або незвичної активності.

Для забезпечення ефективного та високопродуктивного логування в системі обраний Pino – один із найшвидших Node.js логерів, що забезпечує як низький вплив на продуктивність системи, так і гнучкість в налаштуванні.

Основні переваги Pino [16]:

– Pino спроєктований таким чином, щоб мінімізувати затримки під час запису логів, що особливо важливо для високонавантажених систем;

– логування з використанням Pino може бути легко масштабованим для великих додатків завдяки підтримці потокової обробки логів та гнучким опціям конфігурації.

Інтеграція Pino з NestJS. Щоб інтегрувати Pino в серверну частину системи, яка побудована на NestJS, використаний пакет nestjs-pino. Цей пакет забезпечує просту та ефективну інтеграцію Pino в структуру NestJS, дозволяючи використовувати всі можливості цього логера без необхідності у великій кількості додаткового коду.

Для додаткового контролю та прозорості, особливо під час обробки HTTP запитів, важливо мати детальну інформацію про кожен запит, що надходить до

сервера. Для цього використаний пакет pino-http, який інтегрується з pino та дозволяє автоматично логувати всі HTTP запити та відповіді.

Пакет pino-http вже включений до nestjs-pino, тому додатково нічого встановлювати не потрібно.

Перспективи розвитку логування: збір та аналіз логів. У майбутньому можна вдосконалити логування, додавши централізований збір логів та можливість аналітики. Це дозволить зручніше керувати логами з кількох серверів (чи контейнерів) та застосовувати до них інструменти для аналізу та візуалізації.

Наприклад, одним із варіантів може бути інтеграція з Grafana та Loki. Loki – це система збору логів, яка працює в парі з Grafana для зручного відображення й аналізу логів через дашборди.

Якщо серверна частина буде розгорнута в хмарі, то тоді необхідно використовувати хмарні рішення, такі як AWS CloudWatch, Azure Monitor або Google Cloud Logging, які надають аналогічні можливості збору й аналізу даних, залежно від обраного провайдера хмарних послуг.

3.4.4 Розроблення документообігу

Документообіг є важливою функцією інформаційно-реєстрової системи зруйнованого майна та інфраструктури, оскільки саме документи забезпечують офіційну фіксацію інформації про пошкодження, відновлення чи втрату майна.

Для розроблення ефективної серверної частини системи необхідно вирішити кілька важливих задач, серед яких – структуризація та можливість локалізації документів.

Основною концепцією використання однієї таблиці для зберігання всіх типів документів є введення поля «тип документа», яке визначає структуру конкретного документа, дозволяючи системі динамічно адаптуватися до різних типів вмісту.

Основними атрибутами такого документа є:

– тип документа – поле, що визначає категорію або вид документа (наприклад, заява, акт оцінки майна тощо);

– дані документа – динамічне поле, яке містить інформацію у вигляді JSON (нативно підтримується Postgres) або іншого гнучкого формату. Цей підхід дозволяє зберігати різноманітні дані для кожного типу документа, не обмежуючись жорсткою схемою;

– дата створення та модифікації – стандартні поля для контролю за змінами документа;

– статус документа – поле для відстеження стану документа (щоб був зворотній зв’язок для тих, кому це потрібно).

Таким чином, вся інформація про документи зберігається в єдиній таблиці, а структурованість досягається через різні типи документів та їх метадані.

Окрім уніфікації структури документів, забезпечена можливість локалізації окремих документів. Це потрібно для забезпечення користувачів інформацією мовою, яку вони розуміють. Локалізація документів забезпечена за рахунок гнучкості метаданих.

Вибір однієї таблиці для зберігання документів не лише спрощує структуру даних та надає можливості для локалізації, але також відкриває можливості для подальшої оптимізації (тому це немає вплинути на продуктивність в майбутньому).

У майбутньому (коли кількість документів досягне такої межі, що необхідно буде оптимізувати систему), завдяки використанню механізму partitioning в PostgreSQL, можна організувати таблицю так, щоб документи автоматично розподілялися по різних partition на основі певних критеріїв, таких як тип документа або дата створення.

Це дозволить швидше знаходити документи, оскільки запити будуть виконуватися тільки на відповідних partition, замість того, щоб сканувати всю таблицю (ефективно при достатній кількості записів).

Також це надасть можливість видалення або архівування (фізично) старих partition (наприклад, документів, створених більше ніж 5 років тому) без впливу на інші дані.

Розроблення обраного (конкретного) рішення. Для цього створюється структурована архітектура, яка дозволяє ефективно управляти документами в системі, зберігаючи при цьому гнучкість для подальшого розвитку.

Уся загальна частина коду, включаючи інтерфейси та інші конструкти, буде зберігатися в пакеті @prpr/documents. Це дозволяє централізувати спільні елементи, що підвищить зручність та швидкість розробки. Для обробки документів створений окремий модуль у рамках NestJS. Цей модуль включатиме в себе всі необхідні сервіси та контролери, а також логіку для обробки різних типів документів. Кожен контролер документу доступний за HTTP маршрутом /documents/{type}, де {type} – це тип конкретного документа.

Типи документів в системі будуть представлені уніфікованими алфавітно-цифровими кодами (усе це в спільному пакеті). Це означає, що кожен тип документа матиме свій унікальний код, що складається з літер та цифр. Такий підхід дозволяє забезпечити легкість ідентифікації документів, а також стандартизувати їх обробку в системі.

Алфавітно-цифрові коди також спрощують взаємодію з іншими системами, зберігаючи структуру, що є легкою для розуміння та використання.

Для реалізації уніфікованої структури документів створюено спільну MikroORM сутність.

Ця сутність матиме наступні елементи:

– GUID – унікальний ідентифікатор документа у форматі UUID;

– ID – алфавітно-цифровий код, що генерується для зручності користувачів;

– тип документа – уніфікований алфавітно-цифровий код, що ідентифікує класифікацію документа;

– рік – рік створення документа;

– властивості (properties) – специфічні дані документа у форматі JSON, що залежать від типу документа;

– assignments – прив’язка до виконавців або відповідальних осіб;

– дата створення та оновлення – мітки часу для відстеження змін.

Як видно, з пункту «assignments», документи можуть бути підконтрольні, що передбачає наявність виконавця, дедлайну та інших параметрів. Вони призначені для відстеження відповідальних осіб за виконання певних завдань, пов’язаних з документом.

Для цього створюється сутність, яка виглядає наступним чином:

– GUID – унікальний ідентифікатор запису підконтрольного документа (UUID);

– посилання на документ – зв’язок із основним документом, який перебуває під контролем;

– виконавець (Worker) – співробітник, призначений для виконання завдання;

– дедлайн – дата, до якої має бути завершено завдання відносно документа;

– статус – поточний стан документа (обробка, виконано, відхилено);

– коментар – зауваження (публічні) виконавця щодо завдання та статусу (опційно);

– результат (ResultedIn) – посилання на інший документ, якщо результатом роботи є новий документ;

– дата створення та оновлення – мітки часу для фіксації змін та відстеження ходу роботи.

Після визначення загальної структури, для кожного документа (відповідно до класифікації) створений свій сервіс та контролер, що дозволяє обробляти специфічні бізнес-логіки та унікальні властивості.

Усі ці властивості кожного документу зберігаються у спільному пакеті, що дозволяє використовувати їх у різних частинах системи.

Усі документи, створені в системі, чітко структуровані відповідно до уніфікованих вимог та зберігаються у вигляді стандартних алфавітно-цифрових типів документів. Однак, система також може підтримувати додавання документів, створених поза межами системи, таких як скановані акти, звіти або інші юридичні документи.

Для цього створюється новий тип документа, наприклад, «сканований документ», та додається функціональність OCR (оптичного розпізнавання символів) для автоматичного перетворення зображень в текстові дані (щоб документи були

доступними). Ці документи, хоч і надходитимуть із зовнішніх джерел, також будуть інтегровані у загальну структуру системи, що дозволяє їх зручно каталогізувати, поширювати та використовувати нарівні з внутрішніми документами.

Діаграму послідовності отримання документа з реєстру можна побачити у Додатку Г.

3.4.5 Розроблення механізму реєстрації пошкодженого майна

Маючи вже готове рішення щодо документообігу в системі, переходимо до створення механізму реєстрації пошкодженого майна. Ця реєстрація організована у вигляді заявки, яку користувач подає самостійно.

Враховуючи поточні обмеження, зокрема відсутність доступу до державних реєстрів, усі необхідні дані вводитися вручну, без автоматичного підвантаження будь-якої інформації.

Заявка на реєстрацію пошкодженого майна буде інтегрована в загальну систему документообігу, використовуючи вже розроблену архітектуру.

Додається новий тип документа, який має унікальний код f0. Це дозволяє легко ідентифікувати цей тип заявки в системі та забезпечує його уніфіковану обробку.

Визначення властивостей заявки. Важливо зазначити, що властивості заявки на реєстрацію пошкодженого майна та відповідні процеси – тобто, перелік полів, які мають бути заповнені, а також їхній вміст – формуються не розробниками, а відповідними чиновниками, органами чи посадовими особами.

Саме чиновники мають встановити, які саме дані необхідні для реєстрації пошкодженого майна та як вони мають бути структуровані.

Однак, для демонстрації технічних можливостей системи, на етапі розроблення можна тимчасово створити власний набір властивостей для заявки та розписати процес.

Ознайомитися з властивостями заявки можна за допомогою таблиці 3.1.

Таблиця 3.1 – Властивості заявки для реєстрації пошкодженого майна

Назва	Опис	Тип
form	Форма заявки	Об’єкт, містить вкладені поля
status	Статус заявки	Рядок. Вибір зі списку («submitted», «approved», «rejected»)
form.personal	Особиста інформація заявника	Об’єкт, містить вкладені поля
form.personal.title	Форма звертання	Рядок. Вибір зі списку («mr», «ms», «none»)
form.personal.surname.uk	Прізвище (українською)	Рядок
form.personal.surname.en	Прізвище (англійською)	Рядок
form.personal.givenName.uk	Ім’я (українською)	Рядок
form.personal.givenName.en	Ім’я (англійською)	Рядок
form.personal.dob	Дата народження	Дата
form.personal.contact.phone	Номер телефону	Рядок
form.personal.contact.email	Адреса електронної пошти	Рядок
form.personal.contact.preferred	Бажаний спосіб контакту	Рядок. Вибір зі списку («phone», «email»)
form.personal.addressForLegalCorrespondence	Адреса для юридичної кореспонденції	Рядок
form.property	Інформація про пошкоджене майно	Об’єкт, містить вкладені поля
form.property.residenceId	Ідентифікатор населеного пункту (КАТОТТГ)	Рядок
form.property.address	Адреса житла	Рядок
form.property.buildingNumber	Номер будівлі	Число
form.property.apartmentNumber	Номер квартири (опціонально)	Число
form.property.selfAssessment	Самостійна оцінка стану майна (див. алгоритм внизу)	Об’єкт, містить вкладені поля
form.property.selfAssessment.scores.physicalDamage.externalWalls	Стан зовнішніх стін (0-5)	Число

Назва	Опис	Тип
form.property.selfAssessment.scores.physicalDamage.roof	Стан даху (0-5)	Число
form.property.selfAssessment.scores.physicalDamage.windows	Стан вікон (0-5)	Число
form.property.selfAssessment.scores.physicalDamage.internalWalls	Стан внутрішніх стін (0-5)	Число
form.property.selfAssessment.scores.physicalDamage.engineeringSystems	Стан інженерних систем (0-5)	Число
form.property.selfAssessment.scores.safety.explosives	Наявність вибухонебезпечних предметів	Булеве значення
form.property.selfAssessment.scores.safety.debris	Наявність сміття або уламків	Булеве значення
form.property.selfAssessment.scores.livingConditions.habitability	Придатність до проживання (0-5)	Число
form.property.selfAssessment.scores.livingConditions.repairability	Можливість ремонту (0-5)	Число
form.property.selfAssessment.formula.result	Результат оцінки	Число
form.property.selfAssessment.formula.howItWasCalculated	Як було обчислено результат	Рядок
form.property.selfAssessment.descriptionOfDamage	Опис пошкоджень (опціонально)	Рядок
form.property.future	Плани на майбутнє щодо майна	Рядок. Множинний вибір зі списку (див. F0PropertiesFormPropertyFutureOptions)
F0PropertiesFormPropertyFutureOptions	Опції майбутніх дій з майном	Вибір зі списку («going-to-restore-living-here», «going-to-transfer-property», інші)
form.docs	Документи, що додаються до заявки	Об’єкт, містить вкладені поля
form.docs.id	Ідентифікаційні документи заявника (посилання)	Масив рядків

Назва	Опис	Тип
form.docs.propertyId	Документи, що підтверджують право власності на майно (посилання)	Масив рядків
form.docs.evidenceOfDamagedProperty	Докази пошкодження майна (посилання) (опціонально)	Масив рядків

Алгоритм самостійного визначення (класифікації) пошкоджень. Для оцінки пошкоджень майна, використовується алгоритм самостійного визначення стану майна (без залучення спеціалістів).

Цей алгоритм базується на класифікації пошкоджень за кількома категоріями з використанням балів від 0 до 5, де 0 відповідає найкращому стану (відсутність пошкоджень), а 5 – найгіршому (максимальні пошкодження).

Для кожної категорії застосовуються власні коефіцієнти, що відображають їхню важливість для загальної оцінки стану майна.

Оцінка здійснюється за трьома основними категоріями:

– фізичні пошкодження (PhysicalDamage). Оцінюються зовнішні та внутрішні елементи будівлі;

– безпека проживання (Safety). Оцінюється наявність потенційно небезпечних факторів для проживання, таких як вибухонебезпечні об’єкти та уламки;

– умови проживання (LivingConditions). Визначається придатність житла для проживання та можливість його ремонту.

Кожна категорія має свій коефіцієнт ваги, який використовується для підсумкової оцінки:

– PhysicalDamage – 60% (коефіцієнт 0.6);

– Safety – 10% (коефіцієнт 0.1);

– LivingConditions – 30% (коефіцієнт 0.3).

Загальна оцінка розраховується шляхом підсумовування результатів оцінки кожної категорії з врахуванням коефіцієнтів.

PhysicalDamage. Оцінка фізичних пошкоджень обчислюється як середнє арифметичне всіх елементів категорії (зовнішні стіни, дах, вікна, внутрішні стіни,

інженерні системи). Це середнє множиться на коефіцієнт 0.6, що відображає важливість цієї категорії – детальніше відповідно (3.1).

$$\text{physicalDamageScore} = \left(\frac{\sum(\text{значень пошкоджень})}{\text{кількість елементів}}\right) \times 0.6. \quad (3.1)$$

Safety. Для безпеки проживання оцінюються два критерії – наявність вибухонебезпечних предметів та уламків. Кожен з цих критеріїв приймає булеві значення: «так» (присутні) – 5 балів, «ні» (відсутні) – 0 балів. Після цього обчислюється середнє арифметичне, яке множиться на коефіцієнт 0.1 – детальніше відповідно (3.2).

$$\text{safetyScore} = \left(\frac{\sum(\text{булевих значень (0 або 5)})}{\text{кількість елементів}}\right) \times 0.1. \quad (3.2)$$

LivingConditions. Оцінюється придатність житла для проживання та його можливість ремонту. Середнє арифметичне цих оцінок множиться на коефіцієнт 0.3 – детальніше відповідно (3.3).

$$\text{livingConditionsScore} = \left(\frac{\sum(\text{значень умов проживання})}{\text{кількість елементів}}\right) \times 0.3. \quad (3.3)$$

Після розрахунку окремих оцінок для кожної категорії, вони підсумовуються для отримання загального результату – детальніше відповідно (3.4).

$$\text{result} = \text{physicalDamageScore} + \text{safetyScore} + \text{livingConditionsScore}. \quad (3.4)$$

Для прозорості розрахунків кожна частина алгоритму документується через математичний вираз, що включає всі складові оцінки, що дозволяє зрозуміти, як було отримано підсумковий результат.

Після отримання остаточного результату та округлення до найближчого цілого (0 завжди нуль, від 0.01 до 0.99 завжди 1), формується наступний висновок щодо пошкоджень та майна:

– 0 балів – руйнування відсутні;

– 1 бал – незначні руйнування, які не впливають на придатність до проживання;

– 2 бали – помірні руйнування, які потребують ремонту, але не роблять будівлю непридатною до проживання;

– 3 бали – значні руйнування, які роблять будівлю частково непридатною до проживання;

– 4 бали – сильні руйнування, які роблять будівлю непридатною до проживання, але з можливістю відновлення;

– 5 балів – тотальні руйнування, коли будівля не підлягає відновленню.

Процес (алгоритм) створення заявки. Процес створення заявки: завантаження документів (файлів), що додаються до заявки.

Створення заявки на реєстрацію пошкодженого майна передбачає кілька етапів, включаючи технічні рішення щодо зберігання документів, які додаються до заявки.

Перш ніж почати створення заявки, необхідно визначити, де будуть зберігатися файли, що додаються до заявки. Можливі варіанти включають зберігання документів на сервері або використання сторонніх сервісів для зберігання даних.

Зберігання файлів безпосередньо на сервері може призвести до додаткового навантаження на сервер та потребує набагато більше ресурсів для підтримки цього всього.

У системі обрано зберігання файлів у S3 (Simple Storage Service), провайдера Amazon (AWS).

Зазвичай при завантаженні файлів клієнт передає їх спочатку для серверу (наприклад, разом із заявкою), той у свою чергу зберігає їх в оперативну пам’ять, після чого сервер вже самостійно завантажує ці файли на S3, а вже тільки тоді відповідає клієнту – проте цей метод не обраний, бо це створює додаткове навантаження на сервер, особливо при великих об’ємах даних.

Замість цього застосуваний PreSigned URLs від S3, щоб клієнт міг безпосередньо завантажувати файли на хмарне сховище без проміжного збереження на сервері – клієнт спочатку відправляє запит на сервер із метою отримати URL для попереднього завантаження файлів. Це значно оптимізує процес та розвантажує сервер.

Перед тим, як надати клієнту посилання на завантаження файлів у S3, проводиться валідація кожного файлу, щоб переконатися, що всі параметри відповідають вимогам. Це необхідно для того, щоб унеможливити завантаження непередбачених або небезпечних файлів, які не відповідають встановленим правилам і процесам.

Кожен файл, що додається до заявки, проходить перевірку за кількома параметрами. По-перше, перевіряється посилання (ref) на файл (кожен клієнт надає його, щоб потім перевірити яке кому посилання належить) та його ім’я (для метаданих).

По-друге, обов’язково перевіряється розмір файлу. Кожен тип документів, які додаються, має свої обмеження на максимальний розмір, наприклад, документи, що підтверджують особу, можуть мати до 10 МБ, а документи щодо нерухомості – до 50 МБ.

Крім того, є ще перевірка MIME-типів файлів. Для кожного типу документа дозволяються лише певні формати, такі як PDF, JPEG, PNG, і для деяких файлів ZIP. Це запобігає завантаженню неочікуваних форматів.

Відповідно, система проводить всебічну валідацію, щоб гарантувати, що клієнт може завантажити тільки ті файли, які відповідають вимогам та які передбачені процесом реєстрації пошкодженого майна.

Після успішної валідації та збору інформації про файли, підписується посилання для кожного конкретного файлу.

Це посилання повязане з конкретним ref, обмежене за розміром та типом файлу, що запобігає його використанню для завантаження інших даних.

Таблиця 3.2 – Структура відповіді для попереднього завантаження файлів

Назва	Опис	Тип
id	Масив об’єктів з підписаними URL для файлів, що підтверджують особу	Масив об’єктів FileOutputBase (див. таблиця 3.3)
propertyId	Масив об’єктів з підписаними URL для файлів щодо нерухомості	Масив об’єктів FileOutputBase (див. таблиця 3.3)
evidenceOfDamagedProperty	Масив об’єктів з підписаними URL для доказів пошкодження майна	Масив об’єктів FileOutputBase (див. таблиця 3.3) – опціонально

Таблиця 3.3 – Структура об’єкту FileOutputBase

Назва	Опис	Тип
ref	Унікальний референс для кожного файлу, який клієнт «пам’ятає», щоб зв’язати відповідь з файлом	Рядок
url	Підписане посилання на S3 для прямого завантаження файлу	Рядок
fields	Поля форми, які клієнт повинен передати разом із файлом під час завантаження до S3	Об’єкт (зокрема містить властивість «key», яка вказує на майбутнє розташування в S3 файлу)

Вище представлена таблиця 3.2 структури відповіді, яку клієнт отримує після запиту на попереднє завантаження файлів.

Сервер обробляє запит та повертає клієнту посилання, куди він може безпосередньо завантажити файли. Це дозволяє уникнути непотрібного навантаження на сервер.

Після того як клієнт отримує підписані посилання, він надсилає файли безпосередньо до S3 через POST запит на кожне з цих посилань. Кожен файл передається з відповідними полями форми, отриманими раніше разом із URL.

Ці файли завантажуються в тимчасове сховище на S3. Конфігурацію Lifecycle Rule для тимчасового сховища можна побачити на рисунках 3.8 та 3.9.

Тимчасове зберігання потрібне для того, щоб уникнути ситуацій, коли файли вже завантажені в сховище, але заявка ще не була зареєстрована в реєстрі, наприклад через помилки під час реєстрації.

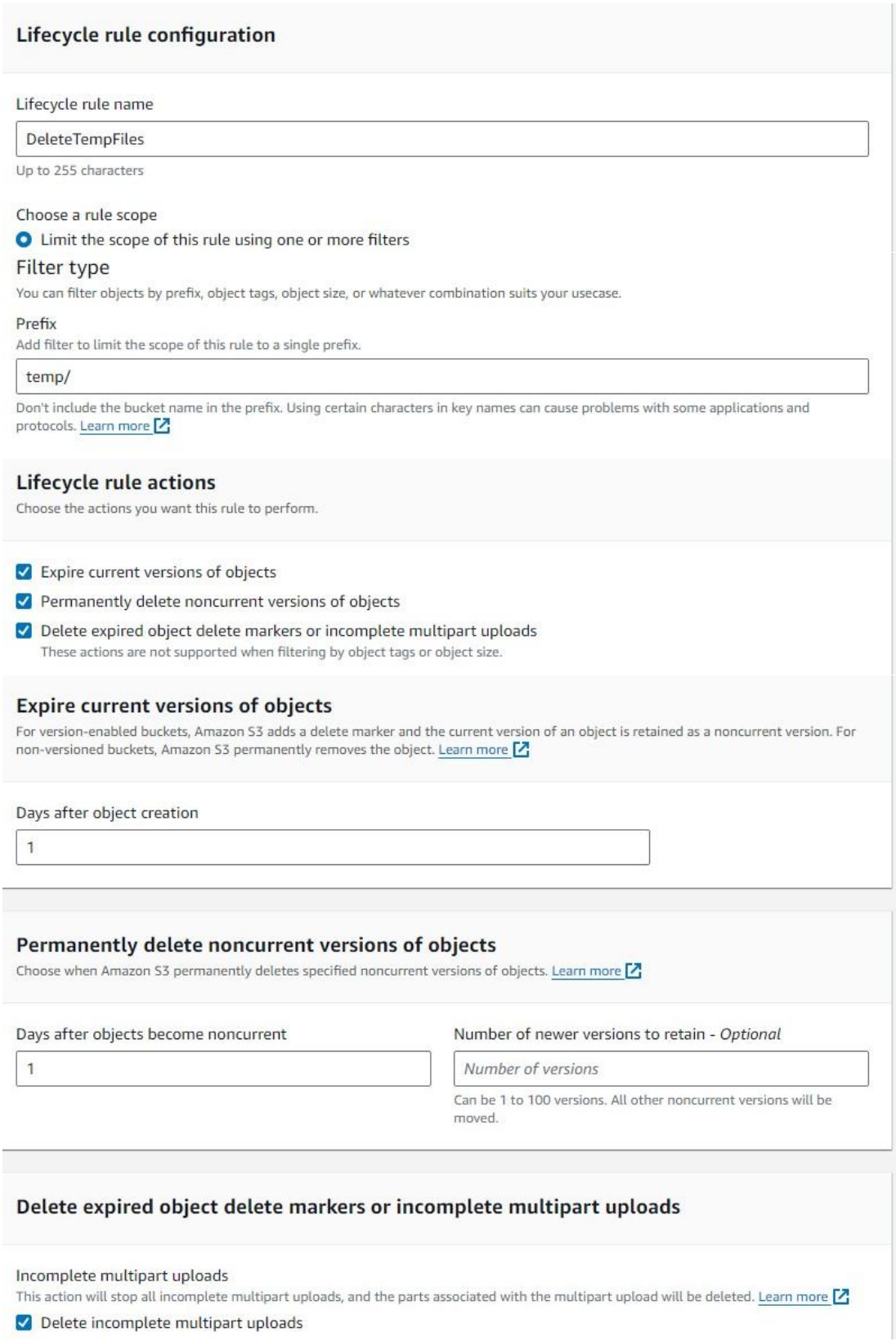


Рисунок 3.8 – Конфігурація для тимчасового сховища на AWS для S3 (1 з 2)

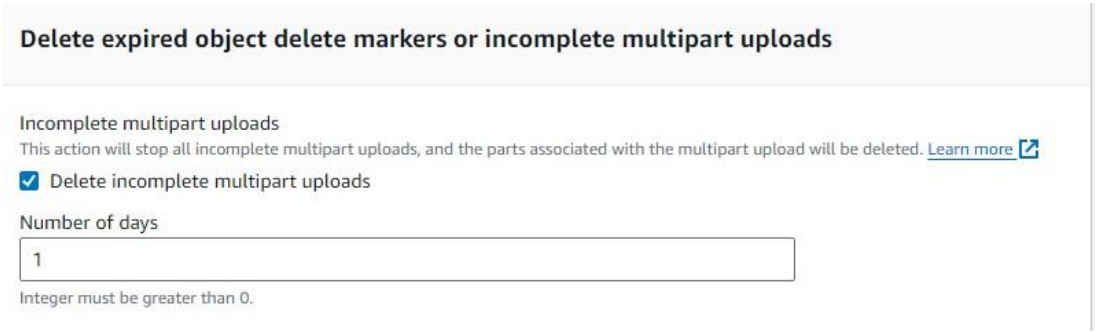


Рисунок 3.9 – Конфігурація для тимчасового сховища на AWS для S3 (2 з 2)

Після того як файли були завантажені у тимчасове сховище S3, наступним кроком є подача заявки на сервер.

Перед тим, як зберегти заявку або передати її для подальшої обробки, проводиться валідація отриманих даних.

Валідація необхідна для того, щоб переконатися, що всі обов’язкові поля заявки правильно заповнені, дотримані всі вимоги щодо форматів даних (наприклад, правильність введеної дати народження, адреси або контактної інформації), а також що завантажені файли відповідають вказаним вимогам.

Це гарантує цілісність даних, відповідність заявленій структурі, а також захищає систему від можливих помилок або спроб несанкціонованих дій.

Опис структури запиту для подачі заявки на сервер, яка включає опис полів, правила валідації та тип даних, можна побачити у таблиці 3.4.

Таблиця 3.4 – Структура запиту для подачі заявки

Назва	Опис	Правила валідації	Тип
personal	Інформація про особу, яка подає заявку	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
personal.title	Звертання до особи (наприклад, пан або пані)	Можливі значення: «mr», «ms», «none». Обов’язкове поле	Рядок

Назва	Опис	Правила валідації	Тип
personal.surname	Прізвище особи на українській та англійській мовах	Є об’єктом, повинен містити лише літери відповідної мови (українська та англійська). Не може бути порожнім	Об’єкт, містить вкладені поля
personal.givenName	Ім’я особи на українській та англійській мовах	Є об’єктом, повинен містити лише літери відповідної мови (українська та англійська). Не може бути порожнім	Об’єкт, містить вкладені поля
personal.dob	Дата народження	Обов’язкове поле, має бути дійсною датою	Дата
personal.contact	Контактна інформація	Є об’єктом, не може бути порожнім	Об’єкт, містить вкладені поля
personal.contact.phone	Номер телефону	Обов’язкове поле, є рядком	Рядок
personal.contact.email	Електронна пошта	Обов’язкове поле, повинне бути дійсною електронною адресою	Рядок
personal.contact.preferred	Бажаний спосіб контакту	Можливі значення: «phone», «email». Обов’язкове поле	Рядок
personal.addressForLegal Correspondence	Адреса для юридичного листування	Обов’язкове поле, є рядком	Рядок
property	Інформація про пошкоджене майно	Є об’єктом. Не може бути порожнім.	Об’єкт, містить вкладені поля
property.residenceId	Ідентифікатор населеного пункту (повинен починатися з «UA»)	Обов’язкове поле, є рядком. Повинне відповідати шаблону «/^UA.*/» та буди дійсним ідентифікатором населеного пункту	Рядок
property.address	Адреса майна	Обов’язкове поле, є рядком	Рядок

Назва	Опис	Правила валідації	Тип
property.buildingNumber	Номер будівлі	Обов’язкове поле, є числом	Число
property.apartmentNumber	Номер квартири	Необов’язкове поле, є числом	Число
property.selfAssessment	Самостійна оцінка стану майна	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
property.selfAssessment.physicalDamage	Оцінка фізичного стану майна	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
property.selfAssessment.physicalDamage.externalWalls	Стан зовнішніх стін	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.physicalDamage.roof	Стан даху	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.physicalDamage.windows	Стан вікон	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.physicalDamage.internalWalls	Стан внутрішніх стін	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.physicalDamage.engineeringSystems	Стан інженерних систем	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.safety	Оцінка умов безпеки	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
property.selfAssessment.safety.explosives	Наявність вибухонебезпечних предметів (так/ні)	Обов’язкове поле	Булеве значення
property.selfAssessment.safety.debris	Наявність уламків (так/ні)	Обов’язкове поле	Булеве значення
property.selfAssessment.livingConditions	Оцінка умов проживання	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
property.selfAssessment.livingConditions.habitability	Придатність до проживання	Обов’язкове поле, є числом від 0 до 5	Число

Назва	Опис	Правила валідації	Тип
property.selfAssessment.livingConditions.repairability	Можливість ремонту	Обов’язкове поле, є числом від 0 до 5	Число
property.selfAssessment.descriptionOfDamage	Опис пошкоджень	Необов’язкове поле, є рядком	Рядок
property.future	Плани щодо подальшого використання майна	Обов’язкове поле, є масивом рядків. Можливі значення: «going-to-restore-living-here», «going-to-live-in-other-place», і т.д.	Масив рядків
docs	Паперові документи, що додаються до заявки (посвідчення особи, документи на майно тощо)	Є об’єктом. Не може бути порожнім	Об’єкт, містить вкладені поля
docs.id	Посвідчення особи (ID документи)	Обов’язковий масив рядків. Максимум 3 рядки (файли). Кожен файл повинен відповідати шаблону «/^temp/documents/f0.*/»	Масив рядків
docs.propertyId	Документи, що підтверджують право на майно	Обов’язковий масив рядків. Максимум 6 рядків (файлів). Кожен файл повинен відповідати шаблону «/^temp/documents/f0.*/»	Масив рядків
docs.evidenceOfDamagedProperty	Докази пошкоджень майна	Необов’язковий масив рядків. Максимум 6 рядків (файлів). Кожен файл повинен відповідати шаблону «/^temp/documents/f0.*/»	Масив рядків

Після успішного проходження валідації, наступним кроком є пошук працівника, який буде відповідальним за виконання заявки.

Для цього використовується алгоритм round-robin, що забезпечує рівномірний розподіл заявок серед працівників. Якщо під час пошуку відповідного працівника система не може його знайти (хоча працівник повинен бути доступний), повертається помилка, що вказує на неможливість обробки заявки через відсутність виконавця.

Коли працівника було знайдено, усі файли, завантажені у тимчасове сховище на S3, переносяться до постійного сховища. Це забезпечує безпечне зберігання важливих документів та виключає ризик втрати даних.

Після цього заявку офіційно призначають знайденому працівнику, встановлюючи відповідний дедлайн для її виконання.

Нарешті, всі деталі заявки, разом з призначенням працівника та файлами, зберігаються в базі даних, та користувачу повертається відповідь із підтвердженням успішного подання заявки із даними про виконавця та дедлайн до якого вона буде виконана.

Якщо заявка на реєстрацію пошкодженого майна успішно пройшла модерацію, це означає, що всі необхідні перевірки були виконані, і дані заявки були підтверджені.

На цьому етапі відкривається офіційна справа, яка далі підлягає опрацюванню відповідними службами. Це означає, що заявка переходить у статус виконаної та розпочинається процес вирішення питання про пошкоджене майно (це зовсім інший процес).

У випадку, якщо заявка не пройшла перевірку, вона залишається без подальшого руху, тобто справа не відкривається. Користувач отримує повідомлення про те, що його заявка не була схвалена, із зазначенням причин відмови.

Діаграму послідовності подання заявки, діаграму життєвого циклу заявки можна відповідно переглянути у Додатку В та Додатку Д.

Процес перевірки заявки, та як вона переходить у справу, детально описаний у розділі 3.4.6.

3.4.6 Розроблення інструментарію адміністратора

Авторизація адміністраторів. Адміністратори володіють привілейованим доступом, через що необхідно перевіряти доступ до ресурсів або відповідних дій.

Для реалізації цього функціоналу використано JSON Web Token (JWT) для авторизації адміністратора, що надає безпечний доступ до ресурсів.

У серверній частині авторизація адміністратора реалізована на основі JWT з використанням access token (токен доступу) та refresh token (токен для оновлення доступу).

Токен доступу надає адміністраторам доступ до системи на малий час (наприклад, на 10 хвилин) – його відкликати неможливо, додаткових запитів до бази даних (наприклад, на перевірку чи акаунт заблокований) непередбачено.

Токен для оновлення доступу дозволяє подовжувати сесію без повторного введення даних для входу, дійсний такий токен тривалий час (наприклад, ця «тривалість» може бути від 1 години до 1 року). Токен для оновлення можна відкликати (проте такий функціонал в системі не реалізовано) або накласти інші обмеження під час оновлення доступу (наприклад, в системі було реалізовано перевірку чи акаунт заблокований та чи надавати в цьому випадку новий токен). Зберігатися цей токен має бути подалі від «клієнта», наприклад, в HTTP-only Cookie.

Такий підхід забезпечує безперервність роботи та зручність для користувача, підтримуючи високий рівень безпеки.

Цей підхід також є гнучким: наприклад, не заважає у майбутньому додати можливість інтеграції з механізмом єдиної авторизації (SSO) для спрощення входу та управління доступом в екосистемі різних систем.

Ознайомитися із діаграмою послідовності щодо авторизація адміністратора (отримання токену) та діаграмою послідовності щодо продовження токену авторизації адміністратором можна у Додатку Е та Додатку Ж.

Обробка завдання щодо заявки та відкриття справи. Адміністратор отримав можливість обробляти завдання, які надходять від заявок на реєстрацію пошкодженого майна (документ типу f0).

Адміністратор може переглянути завдання, прийняти або відхилити заявку, що автоматично запускає створення справи нового типу – c0 – у випадку схвалення.

Нижче розглянуто, як реалізовано цей функціонал.

Для обробки рішення адміністратора щодо завдання спочатку валідуються надані ним дані (детальніше у таблиці 3.5).

Таблиця 3.5 – Структура даних, які подає адміністратор

Назва	Опис	Правила валідації	Тип
status	Вказує дію: approve (схвалити) або reject (відхилити)	Обов’язкове поле, є рядком	Рядок. Вибір зі списку («approve», «reject»)
comment	Коментар для відхиленої заявки (заповнюється, якщо status = reject)	Обов’язкове поле, якщо status = reject	Рядок
keysWithDescription	Містить список файлів (документів) з описами від адміністратора	Обов’язкове поле, містить об’єкти	Масив об’єктів
keysWithDescription[0].key	Посилання на файл, який описується	Обов’язкове поле, є рядком який розпочинається на /documents	Рядок
keysWithDescription[0].description	Опис файлу від адміністратора	Обов’язкове поле, є рядком	Рядок

Цей підхід дозволяє адміністратору описати причини відхилення або додати коментарі до документів, забезпечуючи прозорість та структурованість процесу обробки заявки.

Усі процеси щодо завдання, описані у сервісі F0AssignmentsService, який реалізує функціонал для адміністрування завдань документів типу f0.

Сервіс надає можливість отримати завдання на обробку, а також затвердити або відхилити його. Ключові методи сервісу наведено в таблиці 3.6.

Метод approveOrRejectAssignment перевіряє, чи адміністратор має права на дане завдання і чи перебуває завдання у статусі «в процесі». Якщо так, статус оновлюється на «затверджено» або «відхилено», після чого залежно від статусу заявка або створює новий документ, або оновлюється з коментарем про відхилення.

Таблиця 3.6 – Методи сервісу

Назва	Опис
getWorkersNextAssignmentToProcessByTheirGuid	Отримує наступне завдання адміністратора за ідентифікатором користувача (адміністратора), сортує завдання за датою створення
approveOrRejectAssignment	Дозволяє адміністратору затвердити або відхилити завдання, оновлює статус заявки та додає відповідні коментарі або описи

Якщо адміністратор схвалює заявку, автоматично створюється новий документ типу c0, що представляє собою основну справу по об’єкту.

Функціонал для створення документа реалізований через метод createFromF0Assignment у сервісі C0Service.

Цей метод копіює дані з заявки типу f0 у новий документ типу c0 (детальніше про властивості у таблиці 3.7), присвоює йому статус відкриття та призначає людину, яка буде допомагати людині у справі.

Ознайомитися із діаграмою послідовності щодо відкриття справи c0 як результат заявки f0 та діаграмою життєвого циклу справи c0 можна відповідно у Додатку И та Додатку К.

Таблиця 3.7 – Властивості справи щодо майна

Назва	Опис	Тип
f0Form	Дані заявки f0, які передаються в справу	Об’єкт, містить вкладені поля (див. таблицю 3.1)
mainLongTermAssignmentGuid	Ідентифікатор основного завдання для відстеження зв’язку між завданням адміністратора та справи	Рядок
statusChange	Масив, який фіксує, як змінювався статус справи	Масив обєктів
statusChange[0].status	Статус справи	Рялок. Вибір зі списку (наприклад, «opened»)
statusChange[0].changedAt	Дата зміни	Дата

3.5 Розроблення клієнтської частини системи

3.5.1 Розроблення локалізації клієнтської частини системи

При створенні інформаційно-реєстрової системи зруйнованого майна та інфраструктури, особлива увага приділяється локалізації клієнтської частини.

Система є доступною для користувачів, які говорять різними мовами, тому вибір інструменту для локалізації відіграє важливу роль у забезпеченні багатомовної підтримки та користувацького досвіду.

Перш ніж перейти до вибору конкретного інструменту, оцінені кілька популярних рішень для локалізації клієнтських додатків.

i18next є одним із найпоширеніших інструментів для локалізації веб-додатків. Він пропонує широкий набір функцій, таких як підтримка інтерполяції, форматування дат та чисел, динамічне завантаження перекладів та управління контекстом перекладу.

Однак, i18next більше орієнтований на загальні JavaScript або React-додатки (проте його все одно можна використовувати в Next.js) та не інтегрується настільки органічно (особливо з Next.js 13+) з платформою Next.js, яка використовується для розроблення системи.

Тому, попри його гнучкість, вирішено зосередитися на інструментах, спеціально створених для Next.js.

Inlang це новий інструмент для локалізації, що активно розвивається та пропонує інноваційний підхід до управління перекладами. Inlang має привабливий інтерфейс для роботи з перекладами, підтримує інтеграцію з різними CI/CD системами та базується на модульній архітектурі, що дозволяє легко розширювати функціональність.

Однак на момент аналізу цей інструмент ще перебував у процесі активного розвитку, а його стабільність та підтримка не були достатньо перевіреними на великих проєктах.

Це стало одним із факторів, чому вирішено не використовувати Inlang, попри його перспективність.

Next-intl – це бібліотека, спеціально створена для інтеграції з фреймворком Next.js. Вона забезпечує просту та зрозумілу архітектуру для локалізації, підтримує render на стороні клієнта та сервера (включно з інтеграцією з Next.js 13+), а також дозволяє динамічно завантажувати мовні файли залежно від мови користувача.

Однією з головних переваг Next-intl є її детальна та зрозуміла документація, що дозволяє швидко та ефективно розробляти локалізаційні рішення.

Вибір був двома основними інструментами – Next-intl та Inlang, але остаточно обраний Next-intl. Основною причиною стало те, що Next-intl на момент розроблення системи був більш зрілим та стабільним рішенням.

Його документація виявилася найбільш доступною для розуміння та впровадження, а функціональні можливості повністю відповідали вимогам до

локалізації. Окрім цього, бібліотека має підтримку з боку спільноти, що дозволяє знаходити рішення на форумах та отримувати оперативну допомогу у разі виникнення проблем.

Впровадження Next-intl. Усі налаштування, пов’язані з локалізацією, організовані у спеціальній директорії «src/localization», що дозволяє структуровано зберігати всі файли та логіку локалізації.

Усі переклади (словники) будуть зберігатися в папці «dictionaries».

На початковому етапі додана підтримка двох мов: української (основної) та англійської. Кожна мова має свій окремий JSON-файл у папці «dictionaries», який містить ключі та значення для текстів інтерфейсу.

Далі описані частини коду щодо локалізації та інтеграції з Next.js.

«config.tsx». У цьому файлі визначаються основні конфігурації для локалізації:

– «defaultLocale». Основною мовою системи встановлена українська («uk»). Це значення використовується для ініціалізації мови за замовчуванням, якщо користувач не вказав іншу;

– «locales». Список підтримуваних мов: українська та англійська («["uk", "en"]»);

– «localePrefix». Параметр «localePrefix» визначає, чи потрібно додавати префікс мови до URL-адреси. Встановлено значення «as-needed», що означає, що префікс буде додаватися лише тоді, коли це необхідно (наприклад, для іншої мови, ніж основна);

– «defaultTranslationValues». У цьому об’єкті зберігаються значення для спеціальних форматів перекладів, таких як форматування важливих фрагментів тексту («{chunks}») та додавання розривів рядків («
»);

– «dateFnsLocales». Інтегроване значення мов для бібліотеки «date-fns», яка використовується для форматування дат. Це забезпечує правильне відображення дат відповідно до обраної мови (наприклад, для української – «uk», для англійської – «enGB»).

«i18n.ts». Цей файл відповідає за завантаження перекладів для кожної сторінки на основі мови користувача:

– функція «getRequestConfig» використовується для перевірки, чи існує запитувана мова в списку підтримуваних мов. Якщо мова не підтримується, викликається функція «notFound()», яка спрямовує користувача на сторінку з помилкою 404. Функція динамічно завантажує відповідний словник на основі обраної мови. Якщо мова користувача збігається з основною (українською), завантажується файл «uk.json». Якщо інша – завантажується відповідний словник для цієї мови («en.json» для англійської);

– «defaultTranslationValues». Окрім завантаження словника, також передається значення за замовчуванням для форматування текстів (визначені в «config.tsx»).

«middleware.ts». Файл відповідає за управління маршрутами з урахуванням локалізації. Використовується для правильного перенаправлення користувачів на сторінки з обраною мовою:

– «createMiddleware». Ця функція використовується для автоматичного додавання мовного префікса до URL-адрес залежно від мови користувача та обраних нами налаштувань (наприклад, якщо користувач обирає англійську мову, у URL буде додано «/en/»).

«navigation.ts». Цей файл відповідає за навігацію та роботу з мовними префіксами в URL-адресах:

– «createSharedPathnamesNavigation». Ця функція використовується для створення компонентів навігації, які враховують мовні маршрути. Функція дозволяє генерувати правильні посилання для кожної мови, щоб користувач міг легко переходити між різними мовними версіями сторінок;

– «Link», «redirect», «permanentRedirect». Ці інструменти забезпечують правильне перенаправлення та генерацію посилань з урахуванням локалізації.

Для того щоб інтегрувати локалізацію безпосередньо у структуру маршрутизації Next.js, створюється параметр «[locale]». Це дозволяє автоматично передавати мову для кожної сторінки.

Усі сторінки додатку тепер будуть знаходитись під префіксом «[locale]», що дає змогу Next.js динамічно завантажувати потрібний переклад та відображати вміст сторінки відповідно до обраної мови.

Наприклад, якщо користувач обере англійську мову, URL сторінки виглядатиме як «/en/назва-сторінки», а якщо українську – «/uk/назва-сторінки», але так як це основна, то її буде змінено на «/назва-сторінки».

Цей підхід дозволяє зберігати структуру URL-адрес інтуїтивно зрозумілою для користувачів, а також автоматично адаптувати систему під різні мови, що полегшує як підтримку багатомовності, так та навігацію між сторінками.

3.5.2 Розроблення візуальної складової для клієнтської частини системи (UI)

Однією з складових будь-якої інформаційно-реєстрової системи є її візуальна частина, або користувацький інтерфейс (UI). Саме вона забезпечує взаємодію користувача із системою та визначає загальний користувацький досвід.

У розробленні системи реєстру зруйнованого майна та інфраструктури є особливо важливим створити інтерфейс, який буде не лише зручним та інтуїтивно зрозумілим, але й відповідатиме основним вимогам до сучасних веб-систем.

Основні вимоги до візуальної частини. При розробленні користувацького інтерфейсу системи було визначено три основні вимоги, які описані нижче.

Простота. Ключова ідея полягає в тому, щоб уникнути зайвого «візуального шуму» та забезпечити чітке, зрозуміле відображення інформації. Це означає, що на екрані повинна бути мінімальна кількість елементів, які не є суттєвими для виконання завдань користувача. Система не повинна перевантажувати користувача надмірною кількістю інформації або складними елементами керування. Простота також знижує когнітивне навантаження та допомагає користувачам швидше орієнтуватися в інтерфейсі, що важливо в умовах обмеженого часу та стресових ситуацій, чи якщо є проблеми з увагою.

Доступність. Незважаючи на те, що законодавство України щодо вимог до доступності веб-систем не є жорстким та залишається на досить низькому рівні, є важливим дотримання міжнародних стандартів. Візуальна частина системи повинна бути доступною для користувачів із різними фізичними та сенсорними особливостями. Це стосується не тільки людей із порушеннями зору, включно з

кольоровою сліпотою, але й тих, хто має обмеження у використанні миші або клавіатури.

Наприклад, система повинна бути сумісною з екранними дикторами, а також підтримувати навігацію за допомогою клавіатури. Дотримання цих принципів є вагомим, щоб зробити систему якомога більш інклюзивною та дружньою до всіх користувачів.

Темний режим. Зважаючи на сучасні тенденції та запити користувачів, підтримка темного режиму є однією з важливих характеристик. Темний режим знижує навантаження на зір, особливо при роботі в темних приміщеннях, та допомагає економити заряд акумулятора на пристроях із OLED-дисплеями. Це є важливим елементом дизайну, щоб забезпечити максимальний комфорт користувачів під час роботи із системою у різних умовах.

Використання сторонніх бібліотек. Оскільки розроблення повністю власного рішення для забезпечення візуальної складової могла б вимагати значних ресурсів та часу, вирішено звернутися до використання сторонніх бібліотек та компонентів. Це дозволяє суттєво скоротити час розроблення та уникнути помилок, характерних для створення інтерфейсів з нуля, забезпечивши при цьому високий рівень якості та відповідність вимогам.

Вибір бібліотек для UI є дуже широким, тому детально зупинятися на аналізі кожного варіанту є недоцільним. Проте кінцевий вибір впав на бібліотеку shadcn, яка стала основним рішенням для реалізації візуальної частини клієнтської частини системи.

Причини вибору shadcn. Бібліотека shadcn обрана з кількох важливих причин, описаних нижче.

Гнучкість та можливість кастомізації. Однією з найбільш привабливих характеристик shadcn є її підхід до компонування інтерфейсу.

Фактично, бібліотека пропонує компоненти як шаблони, які користувач може налаштовувати під свої потреби. Це означає, що можна легко змінювати або адаптувати надані компоненти, щоб вони відповідали вимогам [17].

Так званий принцип «збери сам» дозволяє копіювати код компонентів та інтегрувати їх у власний проєкт, що робить процес розроблення значно швидшим та ефективнішим [17].

Доступність компонентів завдяки використанню Radix Primitives. Shadcn базується на компонентах Radix Primitives, які вже забезпечують високий рівень доступності.

Ці компоненти спроектовані з урахуванням стандартів доступності, включаючи підтримку клавіатурної навігації, екранних дикторів та взаємодію користувачів з обмеженими можливостями.

Використовуючи shadcn, можна бути впевненими, що інтерфейс відповідає основним вимогам щодо доступності.

Інтуїтивність дизайну та мінімалізм. Компоненти, які пропонує shadcn, відзначаються мініміалістичним дизайном, що добре співвідноситься з вимогою щодо простоти інтерфейсу. Вони не перевантажують користувача зайвими елементами та легко вписуються в загальний дизайн системи.

Легка інтеграція темного режиму. Ще однією перевагою shadcn є те, що бібліотека забезпечує інтеграцію темного режиму. Завдяки готовим рішенням можна швидко додати підтримку темного режиму, що допомагає зробити систему ще більш дружньою до користувачів.

Використання Tailwind CSS. Ще однією значною перевагою shadcn є те, що бібліотека використовує Tailwind CSS для стилізації компонентів.

Це є важливою характеристикою, оскільки Tailwind забезпечує гнучкий та зручний спосіб створення стилів за допомогою утилітарних класів, що значно спрощує розроблення.

Також однією з переваг Tailwind є те, що він дозволяє уникати класичного підходу до CSS, який вимагає написання великої кількості специфічних класів. Tailwind дає можливість швидко стилізувати компоненти без створення окремих стилів для кожного елементу, що прискорює процес розроблення.

Розроблення візуальних компонентів. Усі компоненти, які можуть бути використані повторно в різних проєктах, будуть організовані в окремий пакет «@prpr/ui».

Така структура дозволяє уникнути дублювання коду та спрощує підтримку й розвиток системи.

Завдяки виділенню спільних компонентів у окремий пакет, можна легко масштабувати рішення, використовуючи одні й ті ж візуальні елементи в різних частинах або навіть в інших проєктах. Це також сприяє підтримці єдиного стилю та інтерфейсних рішень у межах всієї екосистеми розробки, полегшуючи тестування й внесення змін без ризику порушення сумісності.

Таблиця 3.8 – Перелік розроблених компонентів

Назва файлу	Опис компонента
icons/loading.tsx	Компонент для відображення індикатора завантаження у вигляді іконки
alert.tsx	Компонент для відображення сповіщень чи попереджень
badge.tsx	Компонент для відображення міток або позначок статусів
button.tsx	Кнопка для різних інтерактивних дій
calendar.tsx*	Компонент для відображення та вибору дат (а саме, календар, на якому можна обрати дату – використовується з date-picker.tsx)
card.tsx	Компонент для відображення карток з контентом
checkbox.tsx	Компонент для відображення чекбоксів
combobox.tsx*	Компонент випадаючого списку з можливістю пошуку та вибору елемента (може буде використаний для функції автозаповнення варіантів)
command.tsx	Компонент для введення команд або тексту
date-picker.tsx	Компонент вибору дати, на який натискаєш – та обираєш дату (використовує calendar.tsx)

Назва файлу	Опис компонента
dialog.tsx	Компонент діалогового вікна для спливаючих повідомлень чи опцій
dropdown-menu.tsx	Компонент випадаючого меню для вибору опцій
file-uploader.tsx*	Компонент для завантаження файлів (який приймає та валідує файли на клієнті)
form.tsx	Компонент форми для введення даних (інтегрується з react-hook-form)
input.tsx	Компонент текстового поля для введення інформації
label.tsx	Компонент мітки для полів введення
popover.tsx	Компонент спливаючого вікна з додатковою інформацією
progress.tsx	Компонент для індикатора прогресу
radio-group.tsx	Компонент групи радіокнопок для вибору одного варіанту з кількох
scroll-area.tsx	Компонент для області зі смугами прокручування
select.tsx	Компонент випадаючого списку для вибору елемента
separator.tsx	Компонент для візуального розділення елементів
skeleton.tsx	Компонент для показу контенту, що завантажується
sonner.tsx	Інструмент (та компонент) для управління (та відображення) спливаючими сповіщеннями
textarea.tsx	Компонент поля для введення великого обсягу тексту
typography.tsx	Компонент для управління стилізацією тексту

* – компонент було модифіковано чи повністю створено з нуля; опис змін/нововведень до компоненту відсутні у офіційній документації «shadcn» (детальніше – нижче)

У таблиці 3.8 надане систематизоване представлення компонентів, які використовуються в системі, та які можуть бути застосовані повторно в інших проєктах.

У таблиці описані наявні компоненти з їхніми назвами (відповідно до назви файлу в «packages/ui» / «@prpr/ui») та описом (з функціональним призначенням).

При розробленні візуальної частини клієнтської системи використовується, як уже було згадано, бібліотека компонентів shadcn. Однак деякі компоненти модифіковані або створені з нуля, оскільки їх функціонал не повністю відповідав потребам.

Варто зазначити, що зміни або нововведення до цих компонентів не зазначені в офіційній документації shadcn, тому нижче наводиться опис цих змін.

«calendar.tsx». Додана можливість вибору року та місяця через випадаючий список. Для цього використані параметри (properties): «fromDate», «toDate» та «captionLayout="dropdown"».

Це рішення було прийняте для значного покращення UX, оскільки у стандартній реалізації «shadcn» навігація між місяцями та роками можлива виключно за допомогою стрілок вліво-вправо. Такий підхід не завжди виявляється зручним, особливо коли користувачу необхідно обрати дату, що розташована на значній часовій відстані від поточної. Це може спричиняти зайві кроки та втрату часу, що ілюструється на рисунку 3.10.

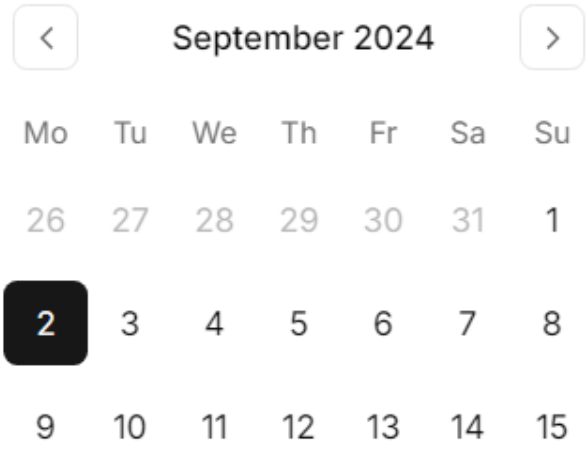


Рисунок 3.10 – Календар з обмеженим UX

Нова реалізація, показана на рисунку 3.11, дозволяє користувачу швидше обирати потрібні дати через випадаючий список.

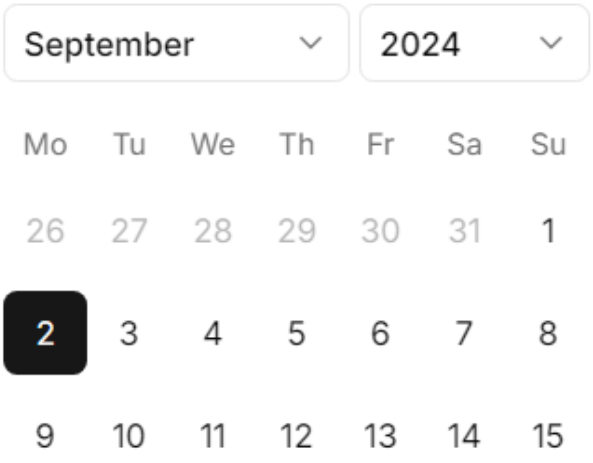


Рисунок 3.11 – Календар із покращеним UX

«combobox.tsx». Основне нововведення – компонент адаптований для роботи з асинхронними даними (наприклад, через HTTP-запити).

Крім того, залишилася можливість локальної фільтрації через параметр (property) «shouldFilter».

Зміни також торкнулися UX: замість того, щоб вводити запит у «popover.tsx» (спливаюче вікно), користувач вводить дані безпосередньо в «input.tsx» (текстове поле), після чого з’являється список результатів. Такий підхід є більш зручним для користувачів та покращує інтуїтивність використання компонента. Детальніше це можна побачити на рисунку 3.12.

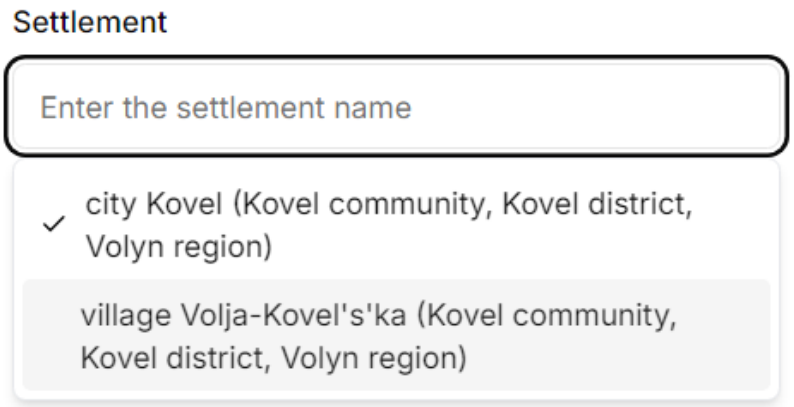


Рисунок 3.12 – Покращений компонент автозаповнення

«file-uploader.tsx». Цей компонент повністю створений з нуля, використовуючи бібліотеку «react-dropzone». Він підтримує валідацію файлів за кількома критеріями, такими як максимальний розмір файлу, кількість файлів та допустимі MIME-типи (наприклад, application/pdf). Компонент дозволяє додавати файли як через «drag-and-drop» (DnD), так і через стандартний системний діалог.

Файли можна також видаляти. Окрім цього, компонент підтримує перегляд завантажених файлів (наприклад, PDF або зображень) безпосередньо в браузері, якщо він підтримує відповідні формати.

Інтеграція компонентів з клієнтом (а саме з локалізацією). Деякі компоненти проєкту потребують інтеграції з системою локалізації. Для цього використовується бібліотека «next-intl» (описано вище) та вона інтегрується з існуючими компонентами.

Локалізовані версії (для конкретного клієнта) компонентів розміщені у папці «components/with-i18n».

Нижче наводяться інтеграції та модифікації компонентів для підтримки локалізації.

«ComboboxWithI18n (combobox.tsx)». Основні параметри, такі як «emptyMessage», «errorMessage», «placeholder», та «loadingMessage», беруться з файлів перекладів. Це дозволяє компоненту показувати відповідні повідомлення різними мовами.

У результаті компонент використовує локалізовані повідомлення для зручнішого UX.

«DatePickerWithI18n (date-picker.tsx)». Для компонента «DatePicker» додано підтримку форматування дат відповідно до локалі (мови) користувача.

Мова користувача отримується через «useParams», щоб динамічно змінювати мовні налаштування. А потім використовується об’єкт «dateFnsLocales» для адаптації компоненту до різних мовних форматів.

Форматування дат та діапазонів дат здійснюється через функції «format» та «formatRange».

«FileUploaderWithI18n (file-uploader.tsx)». Компонент «FileUploader» був створений з нуля, і для його локалізації також використовується «next-intl».

Перекладені повідомлення включають текст для взаємодії з користувачем при завантаженні файлів (наприклад, повідомлення про помилки, підказки та кнопки).

Компонент підтримує локалізовані повідомлення для різних MIME-типів файлів, кількості файлів та їх розміру.

Кожен з цих компонентів отримав додаткові можливості завдяки інтеграції з системою локалізації, що дозволяє забезпечити користувачів комфортною та інтуїтивною взаємодією незалежно від їх мови.

3.5.3 Розроблення механізму реєстрації пошкодженого майна

Усі тонкості та деталі, щодо того, як подається заявка (та пов’язані процеси) описані в розділі 3.3.5.

Основна увага в цьому розділі буде приділена користувацькому досвіду (UX).

Однією з важливих вимог є можливість збереження чернеток заявок.

Розглянуті кілька варіантів, зокрема збереження чернеток на сервері. Це мало б перевагу в тому, що користувач міг би редагувати заявку з будь-якого пристрою. Однак цей підхід вимагав би створення облікових записів для зберігання інформації, чого не передбачено на цьому етапі розвитку.

Крім того, збереження даних на сервері пов’язане з додатковими вимогами до дотримання регуляцій, наприклад, таких як GDPR.

З огляду на ці чинники, вирішено зберігати чернетки на пристрої користувача. Це гарантує, що особиста інформація залишається локальною, зменшуючи ризики, пов’язані з приватністю.

Основний недолік цього підходу – можливість редагування чернетки лише на тому пристрої, де вона була створена. Для реалізації локального зберігання використано бібліотеку zustand, яка дозволяє ефективно управляти станом додатка на клієнтській стороні.

Опис розроблення та UX. Створено зручний інтерфейс для подачі заявки, поділений на етапи, щоб зменшити ментальне навантаження користувачів під час заповнення форми.

На початковому екрані (див. рисунок 3.13) користувач бачить опис процесу заповнення заявки та інформацію про те, що його заявка автоматично зберігається як чернетка. Це допомагає користувачеві знати, що він може повернутися до заявки пізніше та продовжити її заповнення впродовж 7 днів.

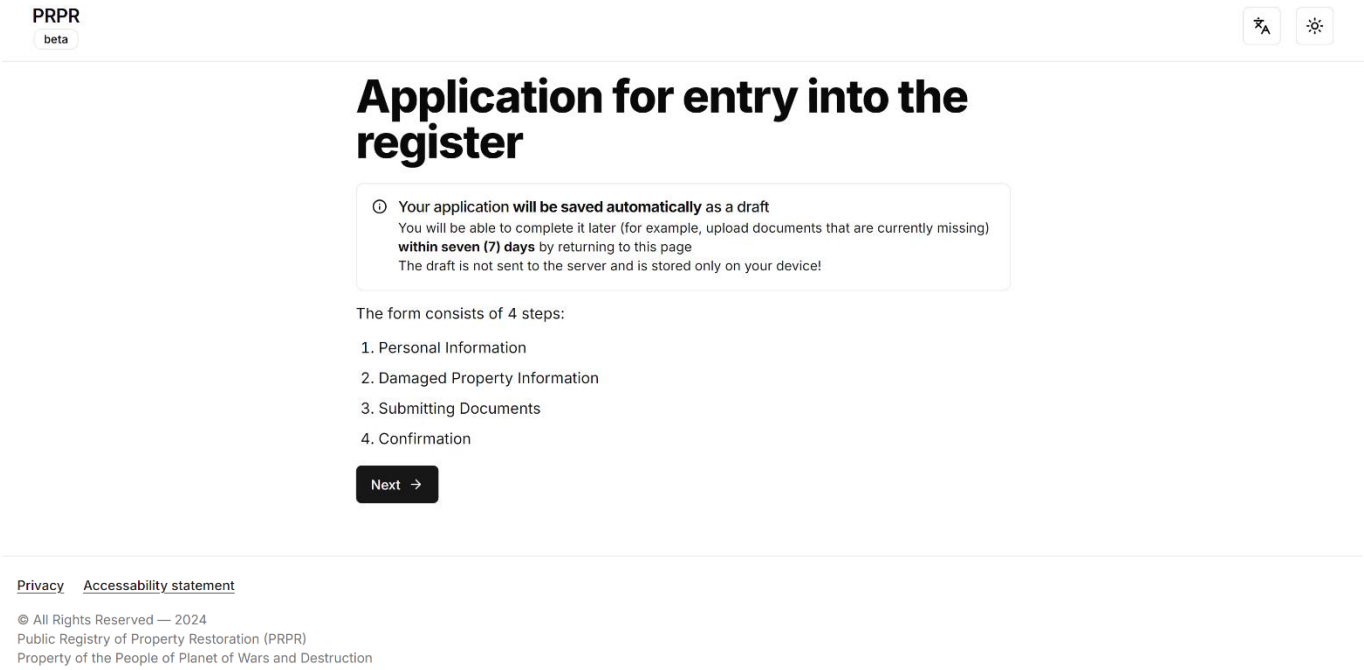


Рисунок 3.13 – Початковий екран

Форма поділена на 4 частини:

– особиста інформація;

– інформація про пошкоджене майно;

– надання документів;

– підтвердження.

Такий підхід зменшує ментальне навантаження та допомагає поступово проходити процес, не перевантажуючи користувача всією інформацією одночасно.

Наступний етап заповнення особистої інформації (див. рисунок 3.14).

На цьому етапі користувач заповнює особисту інформацію, де він може ввести свої прізвище та ім’я українською та латинськими літерами, дату народження, контактні дані (телефон, електронну пошту) та вибрати спосіб зв’язку.

У десктопній версії справа відображається прогрес заповнення заявки та є можливість скинути форму та почати заповнення з початку.

У мобільній версії доступна тільки кнопка скидання.

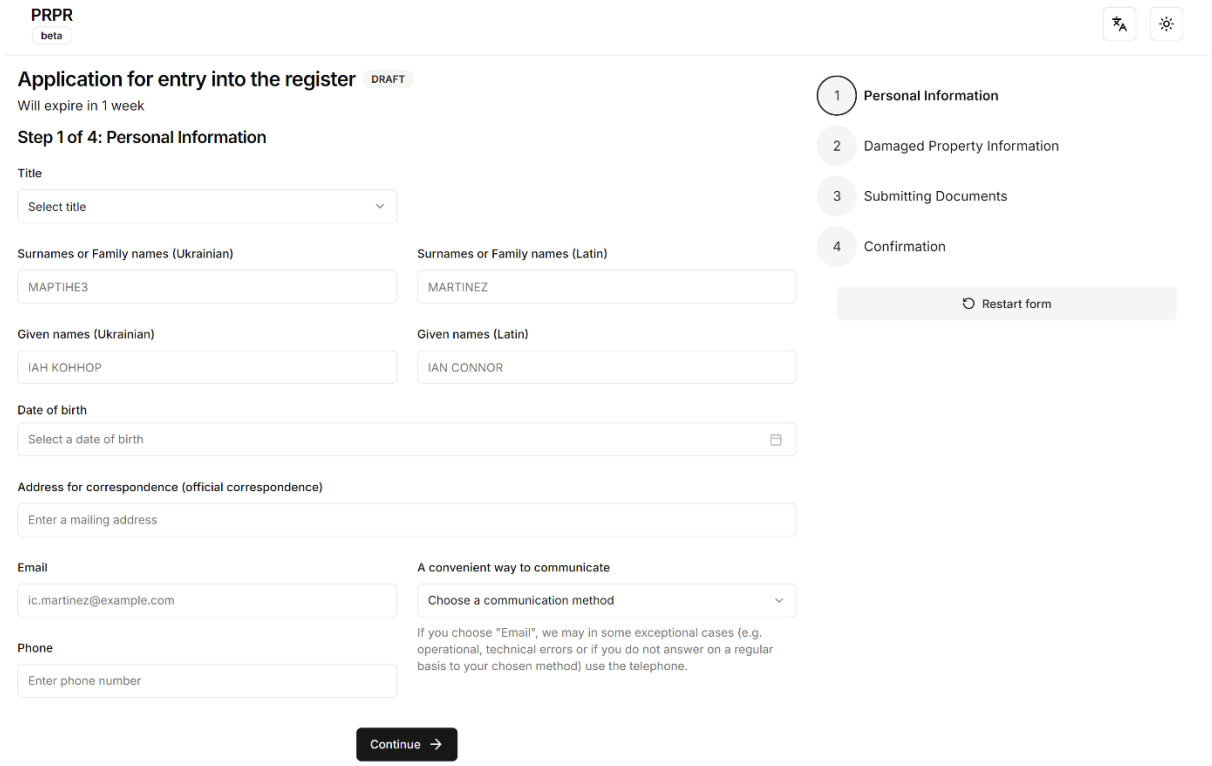


Рисунок 3.14 – Екран заповнення персональної інформації

Після введення особистої інформації користувач переходить до введення даних про своє майно та його пошкодження (див. рисунок 3.15). Серед цих даних вказуються адреса, номер будинку та квартири (якщо вона є). Користувач також може самостійно оцінити рівень пошкоджень за шкалою від 0 до 5, що робить процес зручним і наочним.

Інтерфейс містить прості та зрозумілі вказівки, які допомагають користувачу оцінити стан різних частин свого майна, таких як стіни, дах, вікна, інженерні системи. Також є можливість визначити рівень придатності об’єкта для проживання та оцінити перспективи його відновлення.

Додатково користувач може детально описати характер пошкоджень у текстовому полі. Інтерфейс передбачає також відповіді на запитання щодо майбутніх планів користувача стосовно свого майна, таких як його відновлення, продаж чи переїзд до іншого місця.

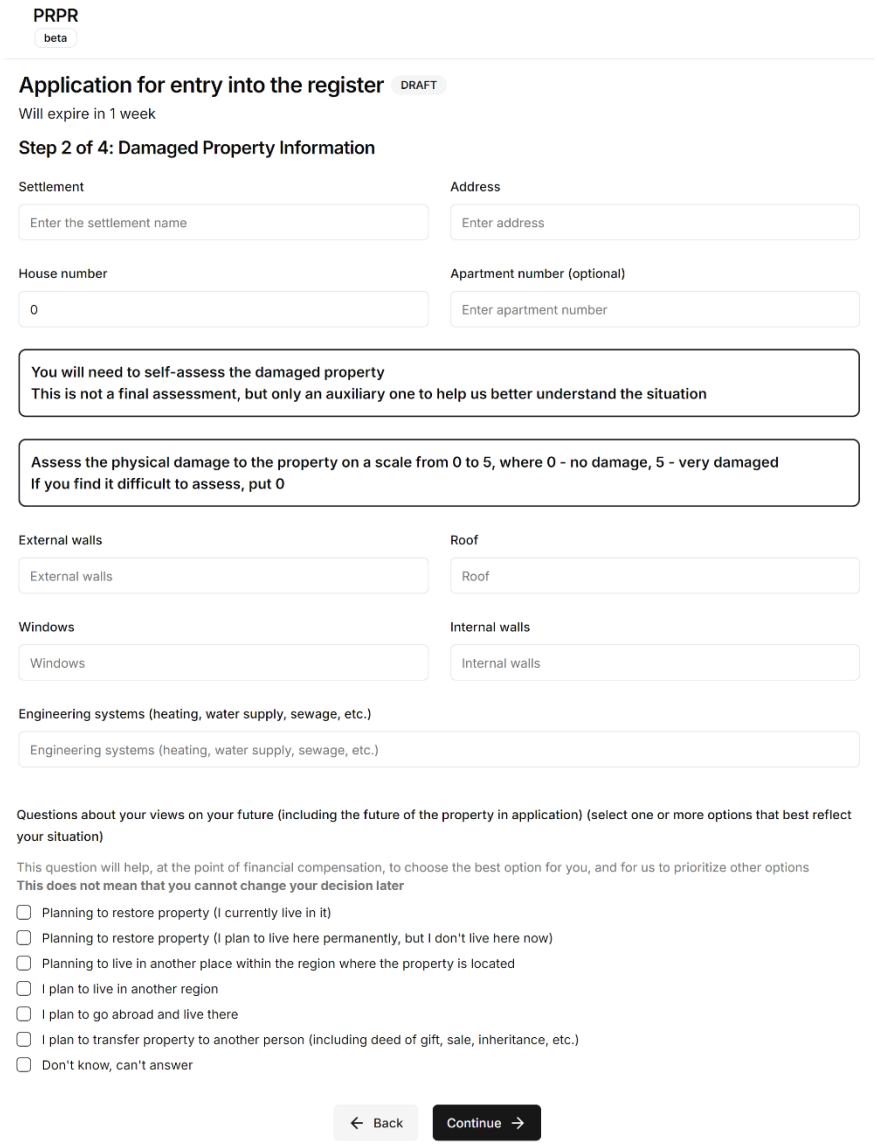


Рисунок 3.15 – Частина екрану заповнення інформації про майно та пошкодження

Після заповнення інформації про майно користувач переходить до етапу додавання всіх необхідних паперових документів, таких як документи про право власності на майно (див. рисунок 3.16). Інтерфейс забезпечує зручний механізм завантаження файлів, дозволяючи користувачу підготувати всі потрібні матеріали для подальшої обробки.

На цьому етапі завантаження документів на сервер не відбувається, що одразу зазначається в інтерфейсі. Крім того, документи не зберігаються в чернетці, про що користувач отримує відповідне повідомлення. Відповідний підхід забезпечує прозорість і чітке розуміння користувачем статусу завантажених файлів на етапі чернетки.

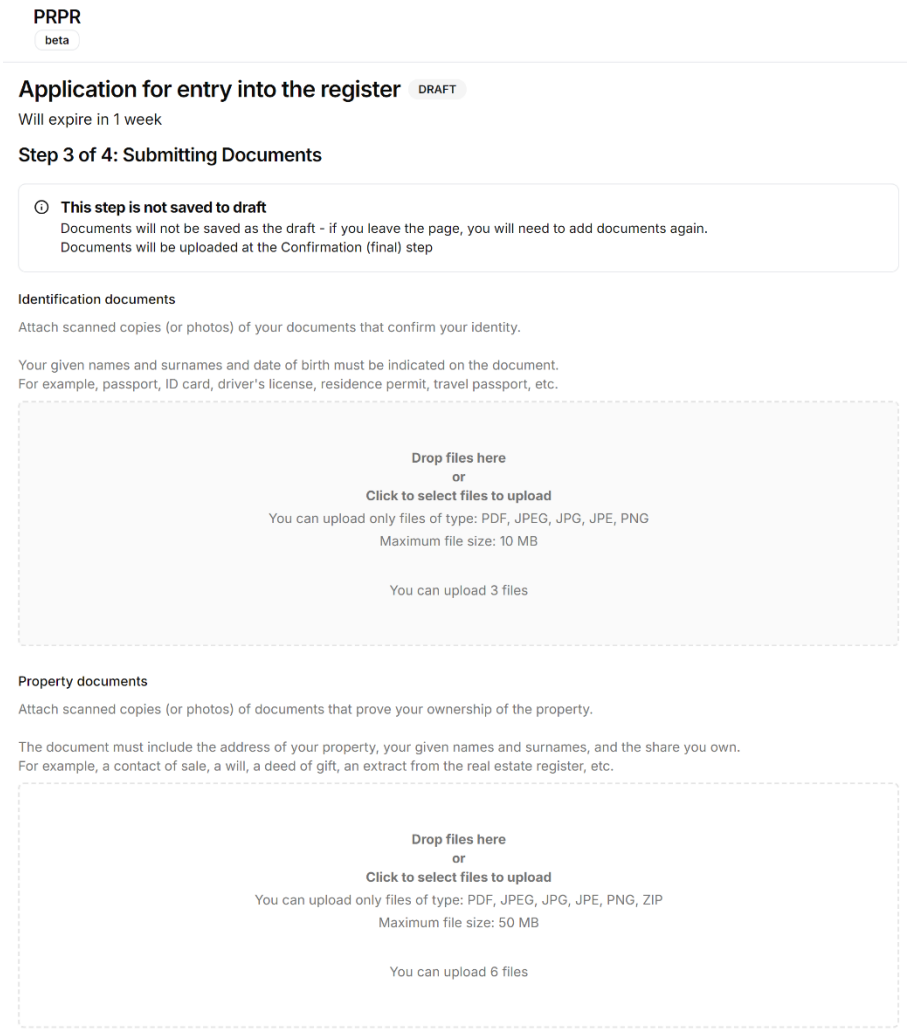


Рисунок 3.16 – Частина екрану додання паперових документів до заявки

Виконавши усі три етапи, користувач переходить до заключного етапу – підтвердження наданої інформації (див. рисунок 3.17). Цей етап є останньою можливістю виправити інформацію чи скасувати заявку.

Після перевірки інформації користувач підтверджує правдивість інформації, ставить підпис та відправляє.

Оскільки процес завантаження заявки в реєстр є асинхронним (наприклад, спочатку завантажуються паперові документи в сховище), то користувач під час виконання завантаження бачить повідомлення про те, що виконується на даному етапі (у тому числі помилки).

Після успішного завершення процесу реєстрації заявки, користувач отримує усю необхідну інформацію про заявку: номер, виконавця та дати (рисунок 3.17).

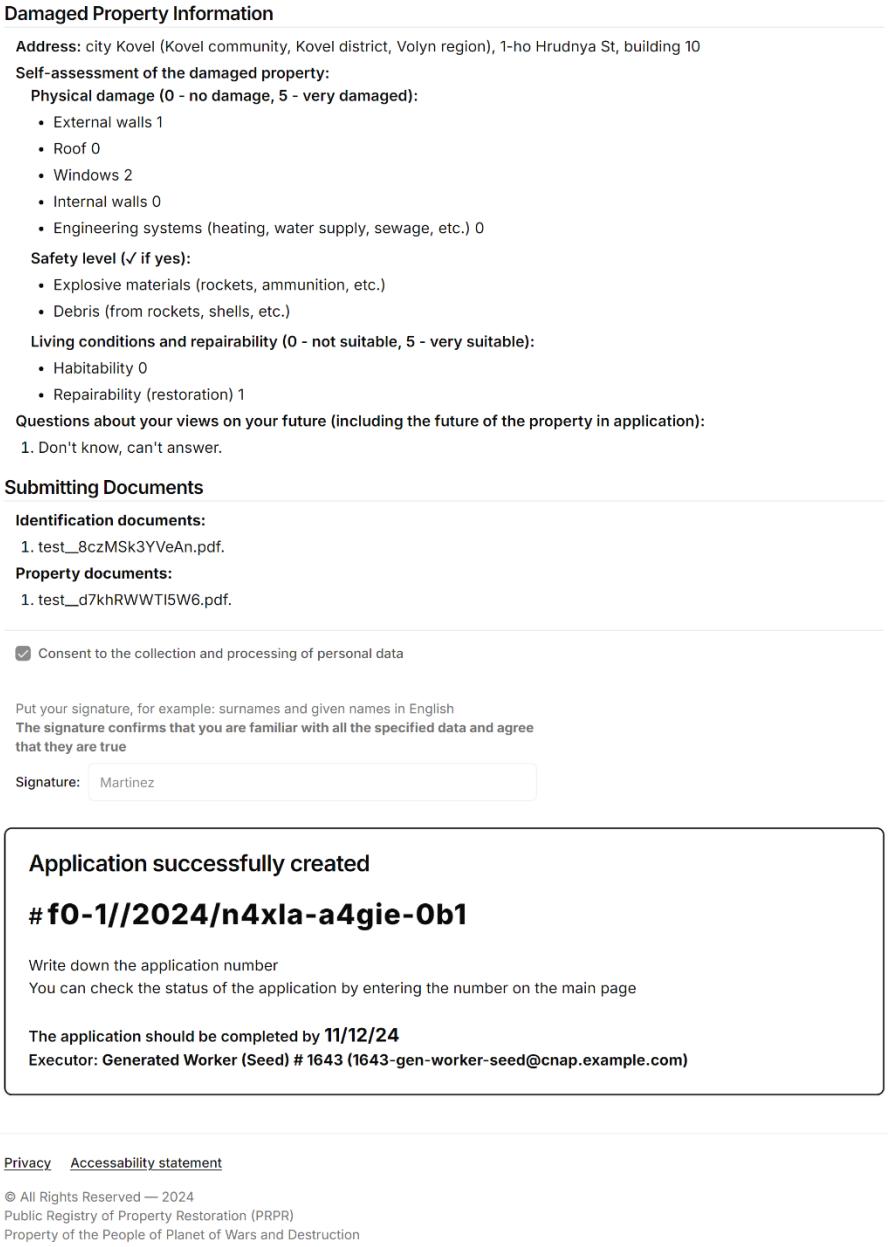


Рисунок 3.17 – Екран підтвердження інформації

3.5.4 Розроблення інструментарію адміністратора

Авторизація адміністраторів. Для адміністраторів у системі розроблена клієнтська частина, яка забезпечує можливість авторизації та роботи із захищеними ресурсами (за допомогою токенів JWT – детальніше в розділі 3.3.6).

Клієнтське зберігання токенів та їх обробка організовані таким чином, щоб забезпечити безпеку та зручність у використанні, використовуючи бібліотеки Zustand для керування станом та Axios для обробки запитів.

Для збереження токену, отриманого під час авторизації, використовується спеціальне сховище стану (store), побудоване на Zustand.

Це дозволяє централізовано керувати токеном та надавати доступ до нього різним компонентам. Токен зберігається у store, а також надається функціонал для встановлення нового токену або очищення токену при виході.

Для інтеграції токену в запити до сервера використовується Axios Interceptor. Він дозволяє автоматично додавати токен до кожного запиту та оновлювати його в разі завершення терміну дії.

Інтеграція токену в запити до сервера відбувається наступним чином:

– запити з токеном. Перевіряється наявність токену в сховищі Zustand. Якщо токен є, він автоматично додається в заголовок Authorization у форматі Bearer {token} до кожного запиту;

– оновлення токену. Якщо сервер повертає помилку 401 (Unauthorized), відбувається спроба оновити токен, звертаючись до функції refresh. Якщо оновлення токену успішне, токен зберігається в стані Zustand, і запит повторно надсилається.

Якщо оновлення не вдається, відбувається вихід із системи (очищення токену зі сховища).

На клієнтській частині для захисту доступу до сторінок адміністратора використаний компонент вищого порядку (HOC) AuthGuard.

Цей компонент виконує перевірку наявності токену та автоматично оновлює його при першому відвідуванні захищеної сторінки (далі токен оновлює Axios Interceptor). Якщо токену немає (помилка 401 Unauthorized), HOC перенаправляє користувача на сторінку входу.

Якщо токен присутній, HOC відображає захищений контент для адміністратора. Якщо ні – показується компонент із завантаженням (поки йде оновлення токену).

Таким чином, на клієнтській частині забезпечено надійний контроль доступу адміністратора до захищених сторінок, а також гнучке управління токеном авторизації

з автоматичним оновленням та виходом із системи в разі завершення терміну дії токену.

Обробка завдання щодо заявки. Інтерфейс для перевірки заявок створено таким чином, щоб адміністратори могли бачити лише ті завдання, які призначені саме їм. Це означає, що кожен адміністратор має доступ до персональної черги завдань на перевірку, а всі інші заявки, призначені іншим адміністраторам, залишаються прихованими від нього.

Після входу в систему адміністратор перенаправляється на сторінку, де відображається заявка, яку необхідно перевірити (приклад наведено на рисунку 3.18). Заявки обробляються послідовно, одна за одною. Якщо у черзі є наступна заявка, вона автоматично відображається з усією необхідною інформацією, потрібною для прийняття обґрунтованого рішення.

Також на сторінці відображається загальна кількість заявок у черзі, що дозволяє адміністратору оцінити поточний обсяг роботи та планувати свій час. Це забезпечує зручність у роботі з великою кількістю запитів (приклад наведено на відповідному рисунку 3.18).

У разі відсутності заявок у черзі адміністратору показується спеціальне повідомлення, яке інформує про те, що на даний момент нових завдань немає. У такій ситуації пропонується можливість оновити чергу для перевірки наявності нових заявок (приклад на рисунку 3.19).

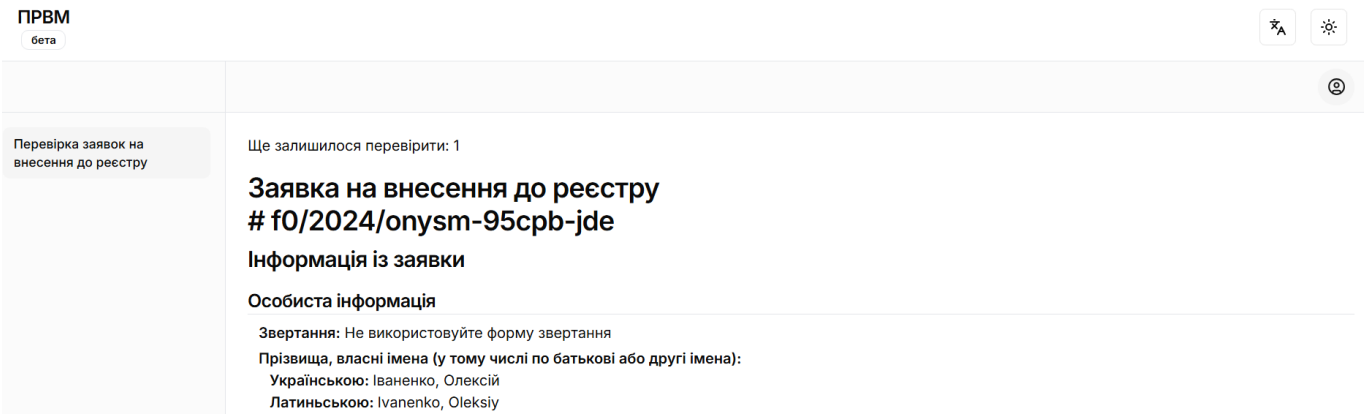


Рисунок 3.18 – Скріншот із заявкою, яку має перевірити адміністратор

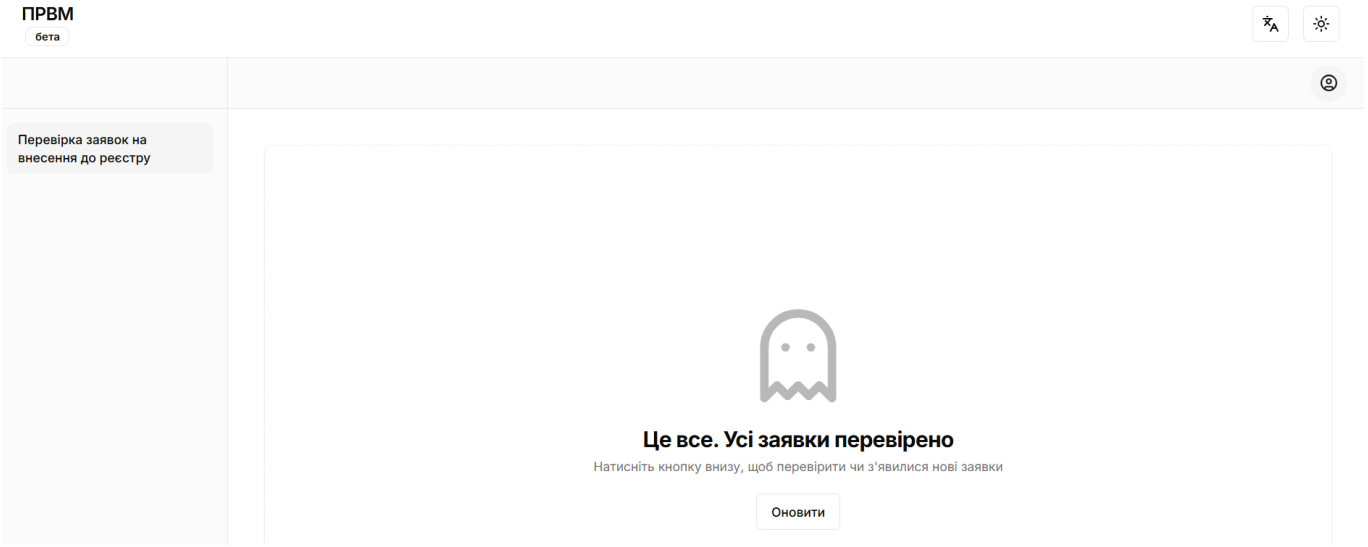


Рисунок 3.19 – Скріншот із повідомленням про відсутність заявок та можливістю оновити чергу

Висновки до розділу 3

У розділі 3, присвяченому розробленню інформаційно-реєстрової системи зруйнованого майна та інфраструктури, прийняті технічні рішення, які визначають архітектуру (з нею можна ознайомитися на діаграмі у Додатку Б) та функціонал системи.

Обраний TypeScript як основна мова програмування. Для фронтенду використовується React у поєднанні з Next.js. Серверна частина реалізується за допомогою NestJS, а для зберігання даних застосована реляційна база даних PostgreSQL.

Архітектура проєкту побудована у вигляді монорепозиторію з використанням pnpm та Turborepo, що оптимізує управління залежностями.

Для відсутніх даних інтегровані рішення з відкритих джерел, щоб обійти проблему обмеженого доступу до офіційних реєстрів, використовуючи альтернативні Open Source ресурси.

Серед основних реалізованих функцій: документообіг, можливість подання заявок на реєстрацію пошкодженого майна, відстеження статусу цих заявок у реальному часі, збереження чернеток заявок тощо.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Тестування є важливим етапом перевірки коректності роботи інформаційно-реєстрової системи зруйнованого майна та інфраструктури, адже саме на цьому етапі виявляються потенційні помилки та недоліки, що можуть вплинути на точність та стабільність функціонування системи в реальних умовах.

Мета тестування – перевірити, чи всі процеси, закладені в архітектуру системи, виконуються коректно, та оцінити її здатність відповідати на потенційні помилки.

4.1 Ручне тестування

З огляду на обмеженість ресурсів, тестування системи проведене вручну із застосуванням інструменту Postman, що дозволяє проаналізувати кожен етап взаємодії клієнт-сервер, зокрема, відправлення запитів, обробку відповідей та помилок.

Тестові сценарії включають в собі функціональні тести та тести на обробку помилок.

Функціональні тести включають перевірку коректності виконання операцій, наприклад, на відповідність бізнес-логіці. Сценарії з обробки помилок спрямовуються на виявлення некоректних даних чи відсутніх параметрів у запитах, на що система повинна реагувати відповідно. Тестування є успішним за умови, що система правильно обробила запит та повернула результати (у тому числі неуспішні).

Спочатку протестований ендпоінт «/documents/f0/files/pre-upload» (POST) – попереднє завантаження файлів до заявки, що включало перевірку реакції системи на різні варіанти некоректних вхідних параметрів: відсутність обов’язкових параметрів, неправильний тип файлу та порушення допустимого розміру у байтах. У кожному з цих випадків система належним чином виявляє помилку. На рисунках 4.1 та 4.2 представлено результати для таких помилок.

При правильних вхідних параметрах система успішно генерує pre-signed URL для завантаження файлів, забезпечуючи їх безпечну передачу до сховища. Результат цього тесту показано на рисунку 4.3.

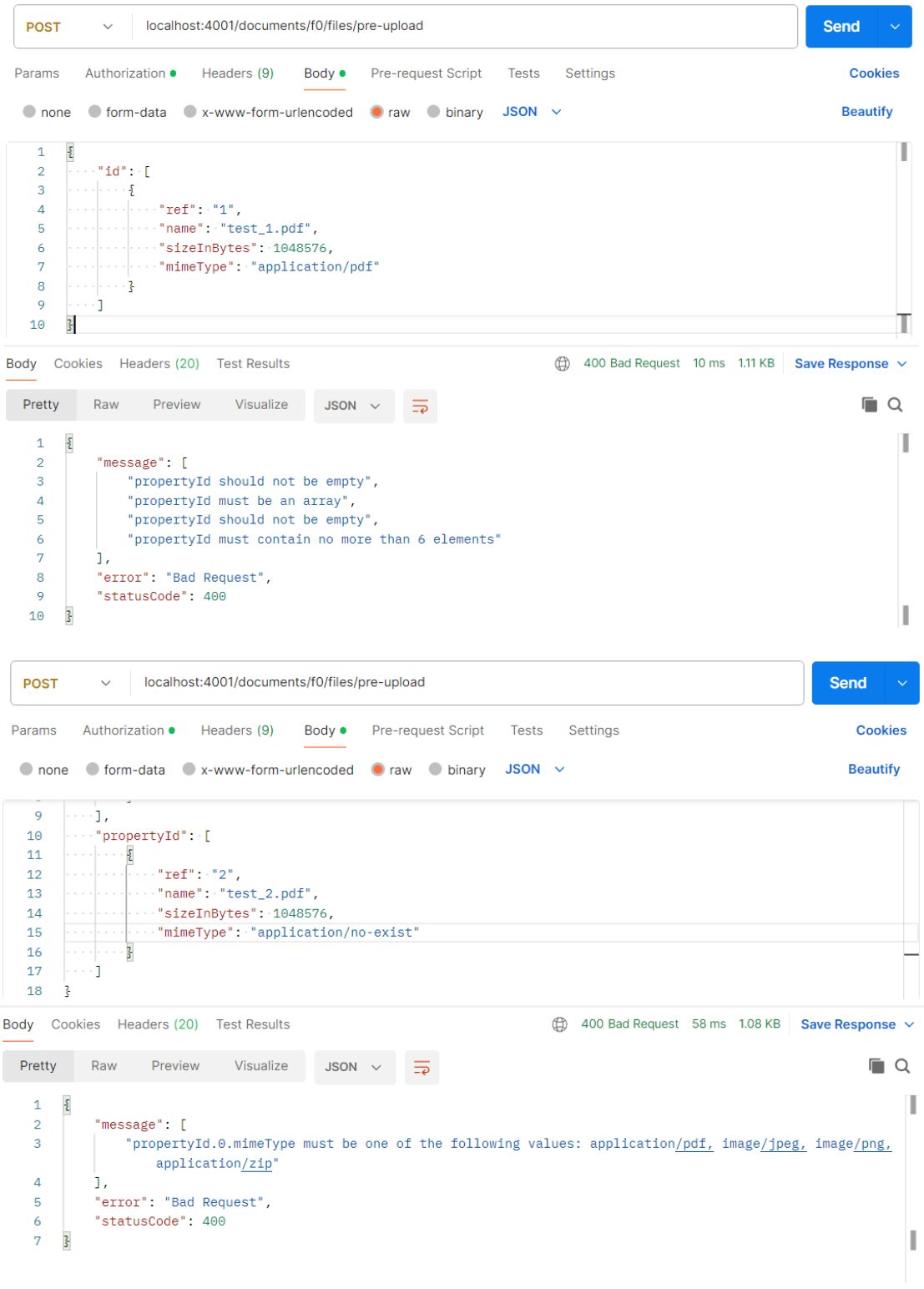


Рисунок 4.1 – Результати із неправильними вхідними параметрами (відсутність параметру, неправильний тип)

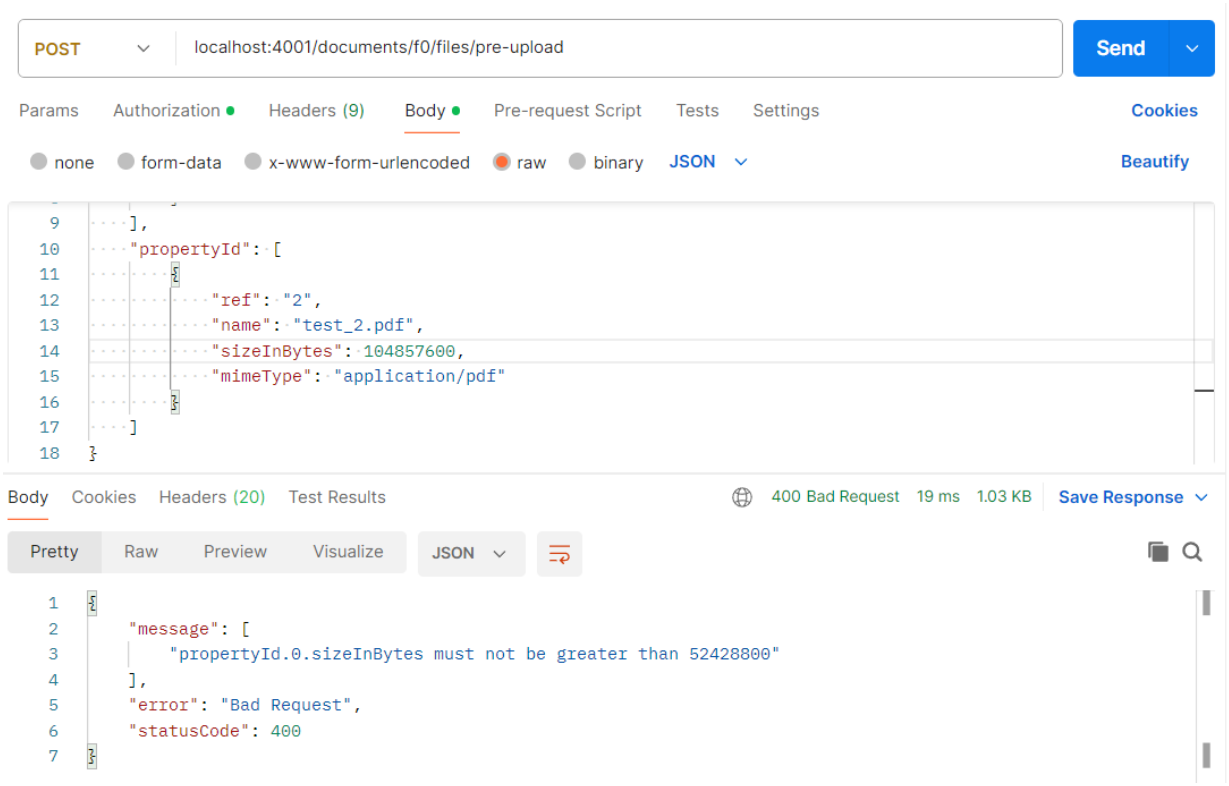


Рисунок 4.2 – Результати із неправильними вхідними параметрами (розмір у байтах)

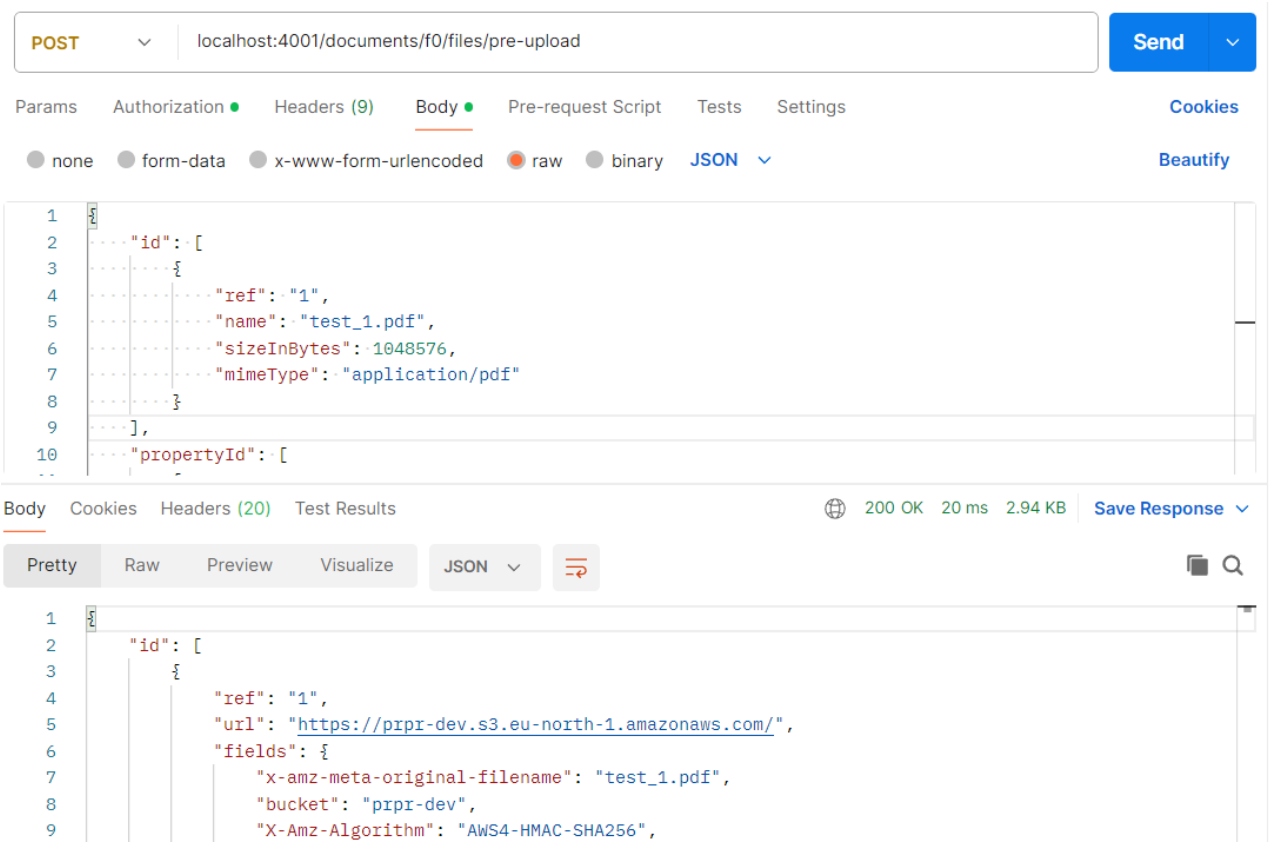


Рисунок 4.3 – Успішний результат попереднього завантаження

Під час тестування ендпоінту «/documents/f0» (POST) – реєстрація заявки – перевірена обробка системою некоректних вхідних параметрів. Зокрема, система правильно реагувала на відсутність обов'язкових параметрів, помилки в символах (наприклад, відсутність українських або англійських букв) (див. рисунки 4.4 та 4.5).

Під час тестування перевірялися різні помилки валідації, включаючи введення неправильного ідентифікатора КАТОТТГ, некоректний вибір значень зі списку, а також використання помилкових ключів до файлів (наприклад, таких, що не починаються з «/temp/documents/f0»). Крім того, тестувалося перевищення допустимого розміру масиву. Усі ці випадки система обробляла коректно, повертаючи повідомлення про виявлені помилки. Результати тестування та приклади таких повідомлень наведені на рисунках 4.6, 4.7 та 4.8.

Для успішного сценарію, коли всі параметри вводилися правильно, система без проблем створювала заявку, після чого повертала інформацію про її успішну реєстрацію. Цей процес проілюстровано на рисунку 4.9.

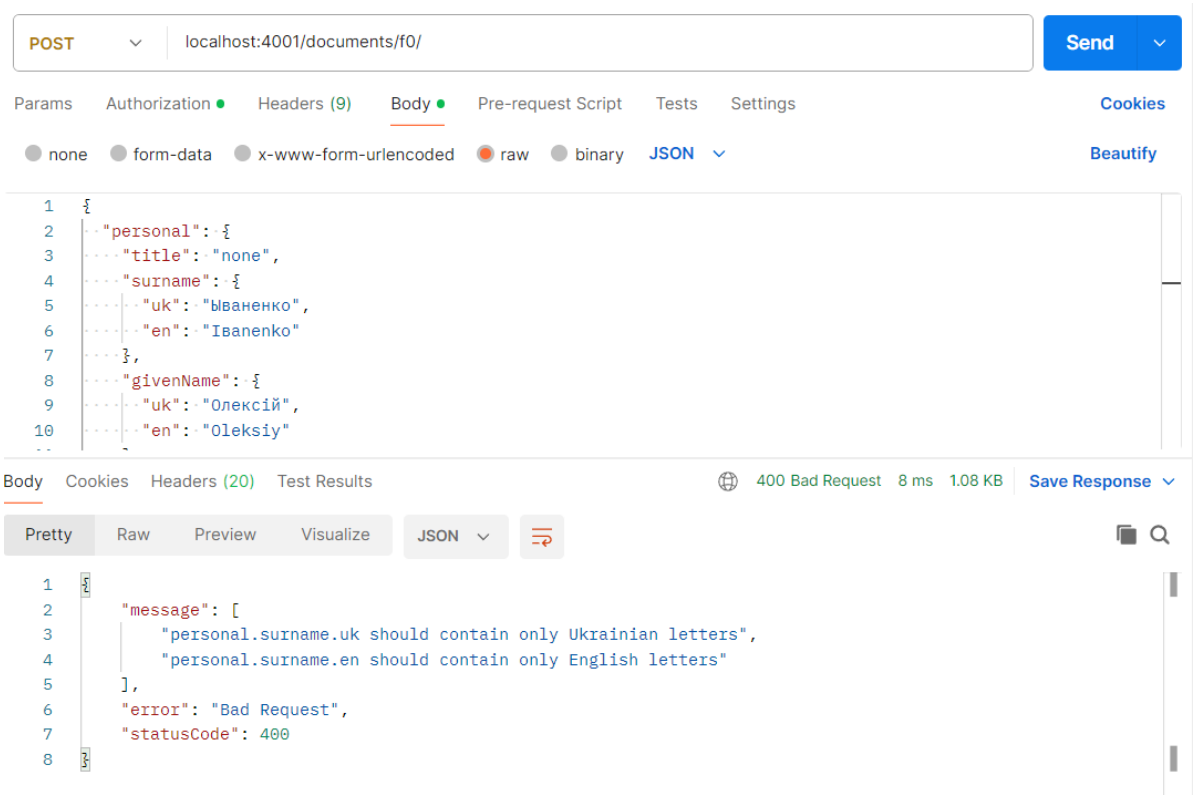


Рисунок 4.4 – Результат із неправильними параметрами (відсутність українських, англійських символів)

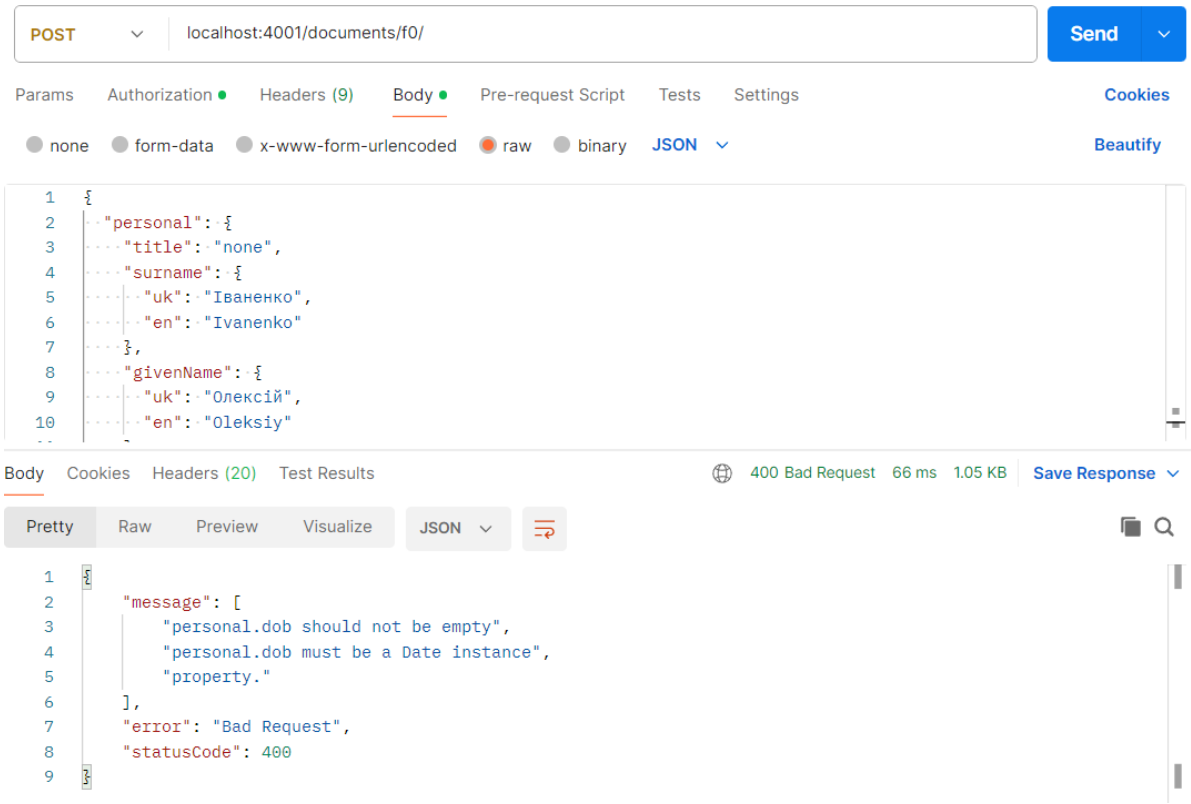


Рисунок 4.5 – Результат із неправильними параметрами (відсутність параметру)

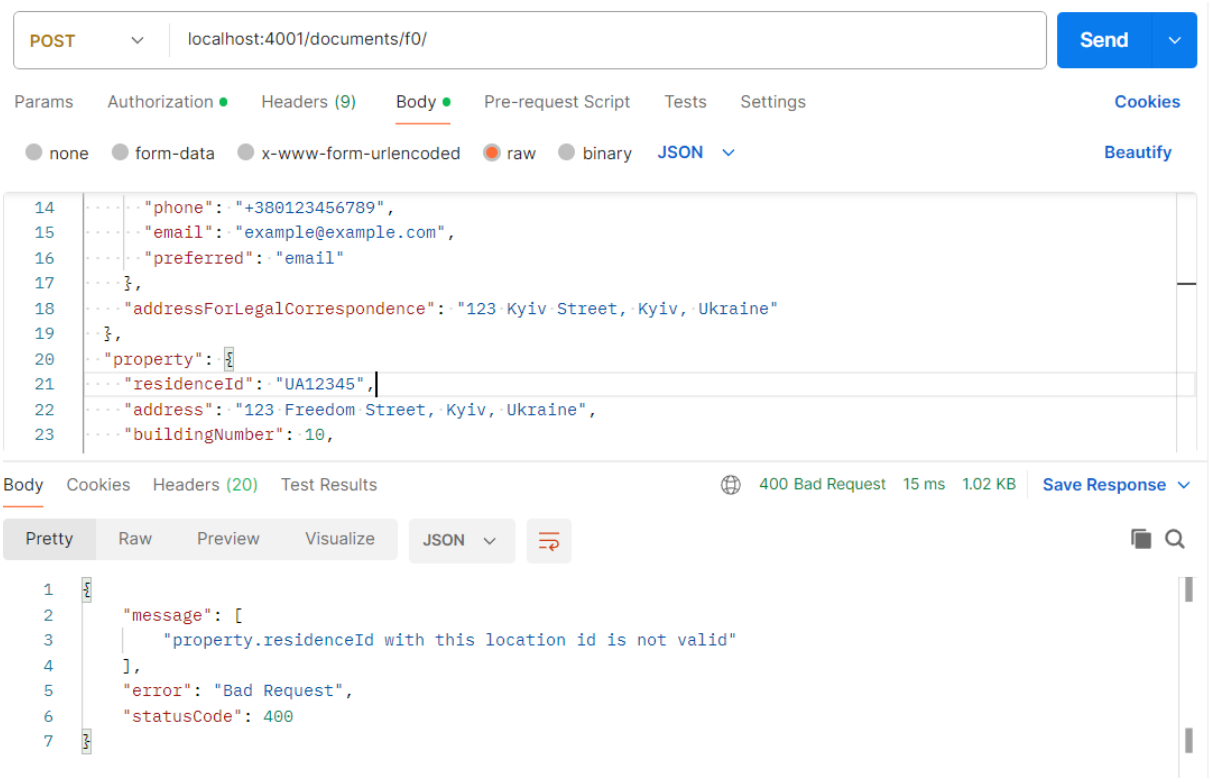


Рисунок 4.6 – Результат із неправильними параметрами (неправильний ідентифікатор КАТОТТГ)

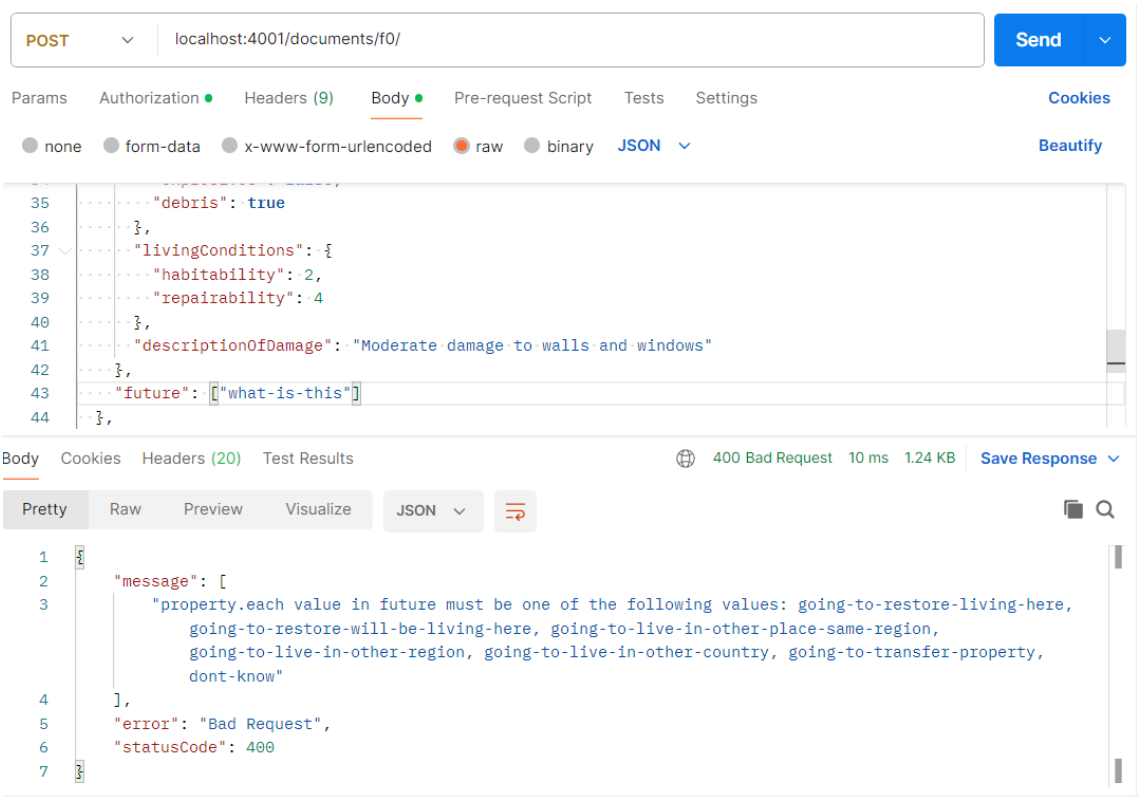


Рисунок 4.7 – Результати із неправильними параметрами (неправильний вибір із списку)

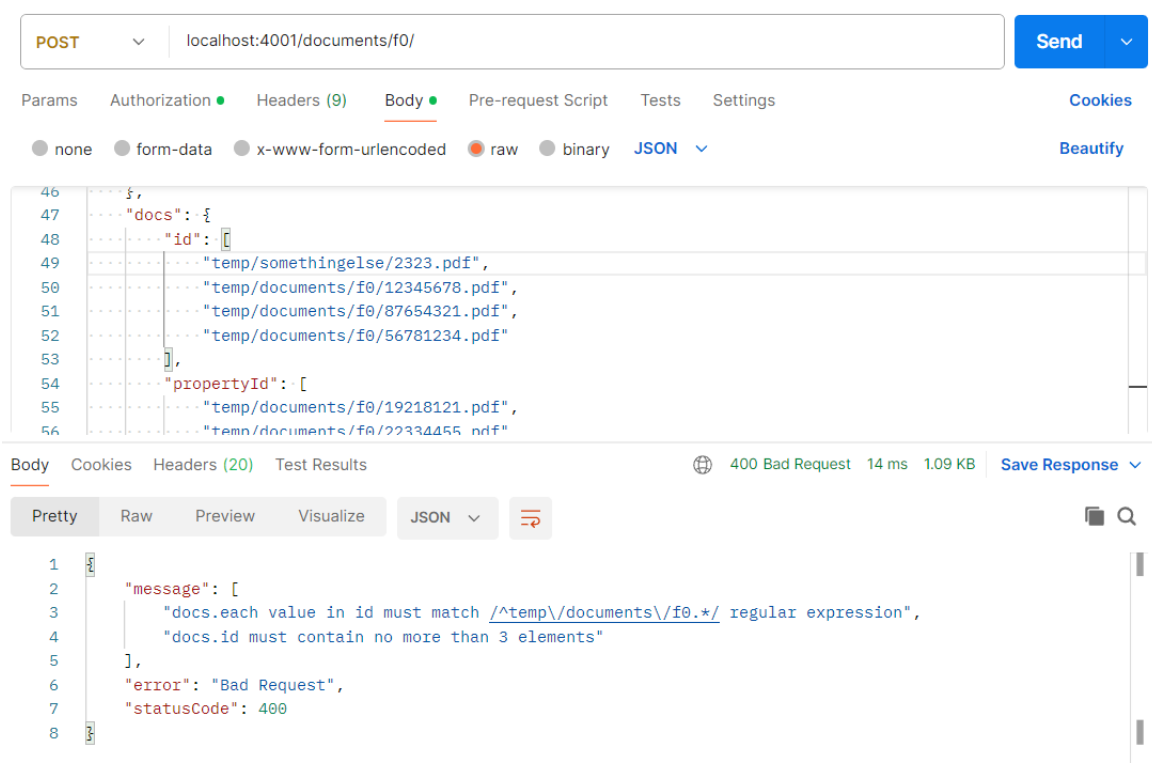


Рисунок 4.8 – Результат із неправильним параметром (неправильний ключ, перевищення розмірів масиву)

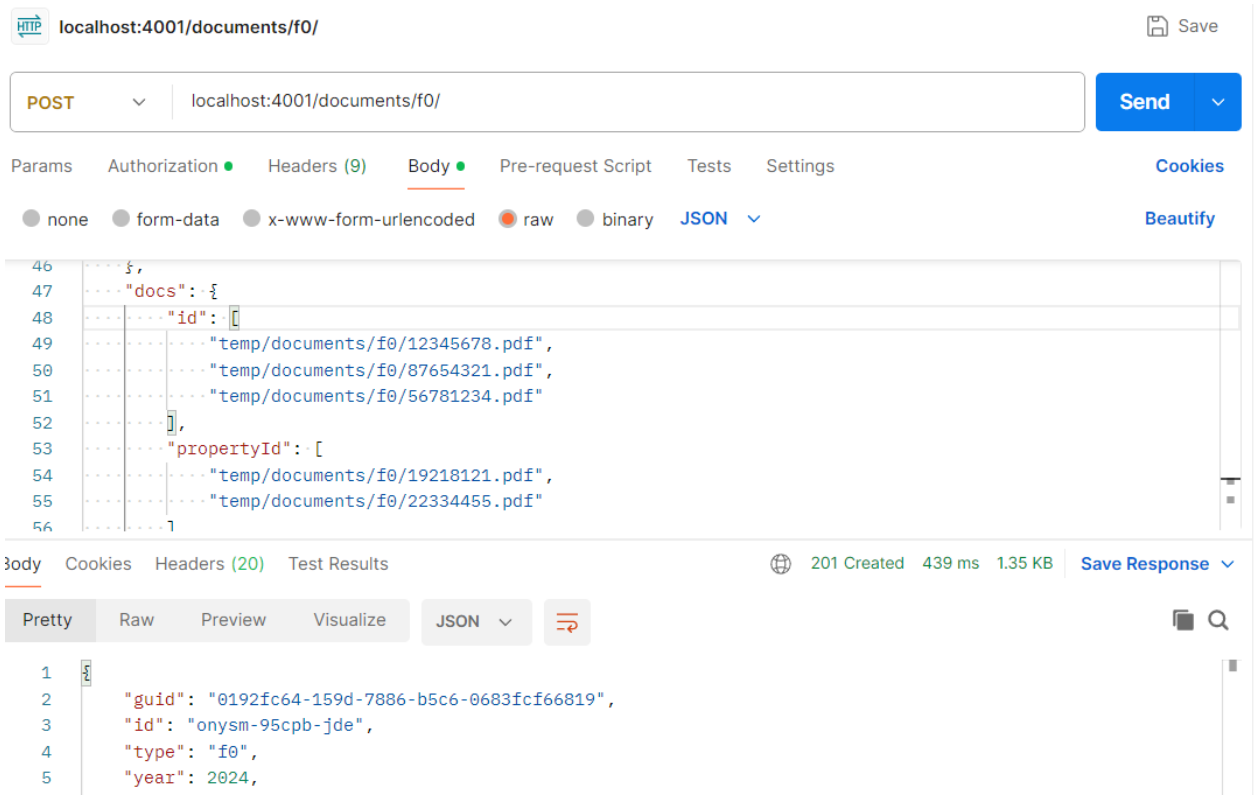


Рисунок 4.9 – Результат із успішним створенням заявки

Тестування процесу авторизації (набір ендопоінтів «/admin/auth») адміністратора включало перевірку реакції системи на правильні та некоректні облікові дані. У випадках, коли користувач вводив неправильний юзернейм або пароль, система повертала помилку, не надаючи доступу (див. рисунок 4.10). При введенні коректних даних авторизація проходила успішно, та система повертала токен у відповіді та зберігала токен оновлення у файлі Cookie (див. рисунок 4.11).

Додатково протестований процес оновлення токену через ендпоінт «admin/auth/refresh». У випадках, коли обліковий запис адміністратора був заблокований (значення поля «is_suspended» у базі даних дорівнювало істині, див. рисунок 4.12), система коректно реагувала, відмовляючи в оновленні токену та видаючи відповідне повідомлення про помилку (приклад наведено на рисунку 4.13).

У ситуаціях, коли обліковий запис адміністратора не був заблокований, оновлення токену проходило успішно. Система належним чином видавала новий токен без будь-яких помилок, що показано на рисунку 4.14.

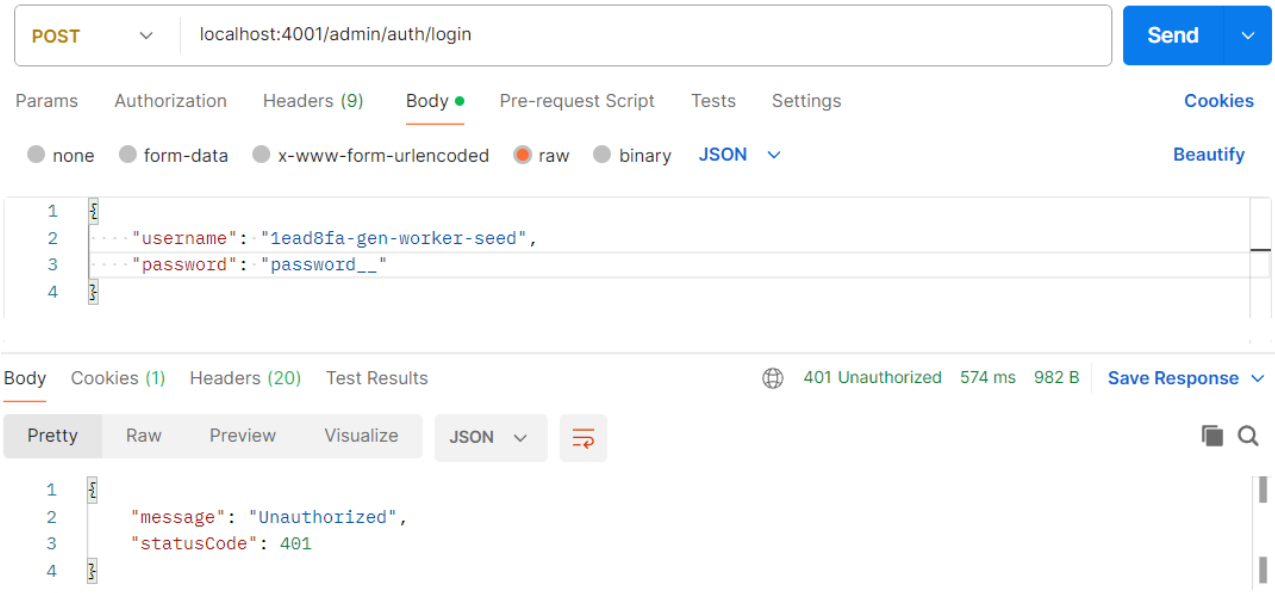


Рисунок 4.10 – Неуспішний результат входу (неправильний юзернейм або пароль)

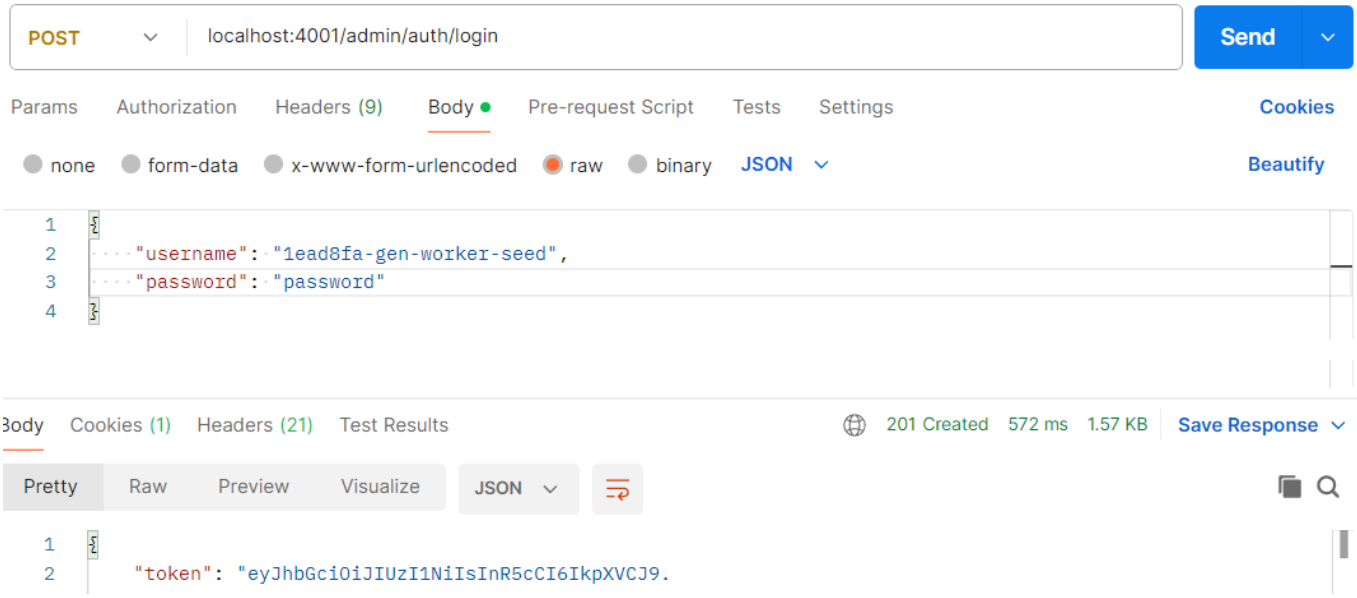
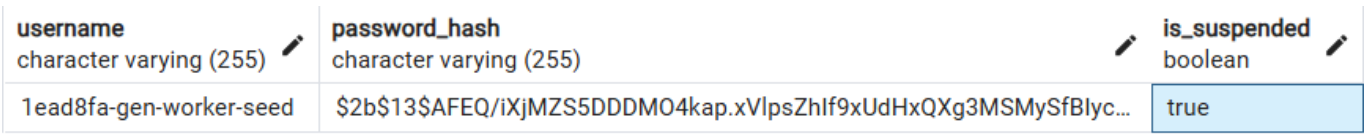


Рисунок 4.11 – Успішний вхід із токеном



username character varying (255)	password_hash character varying (255)	is_suspended boolean
1ead8fa-gen-worker-seed	$2b$13$AFEQ/iXjMZS5DDDMO4kap.xVlpsZhlf9xUdHxQXg3MSMySfBlyc...	true

Рисунок 4.12 – Адміністратор заблокований

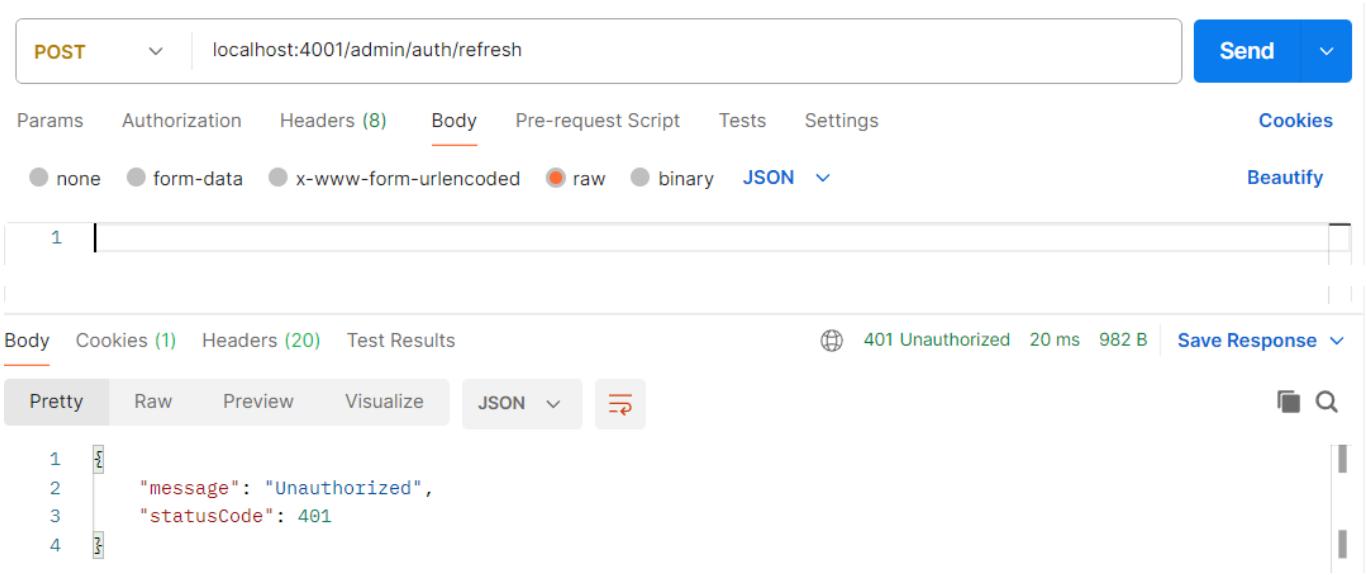


Рисунок 4.13 – Неуспішне оновлення токену (користувач заблокований)

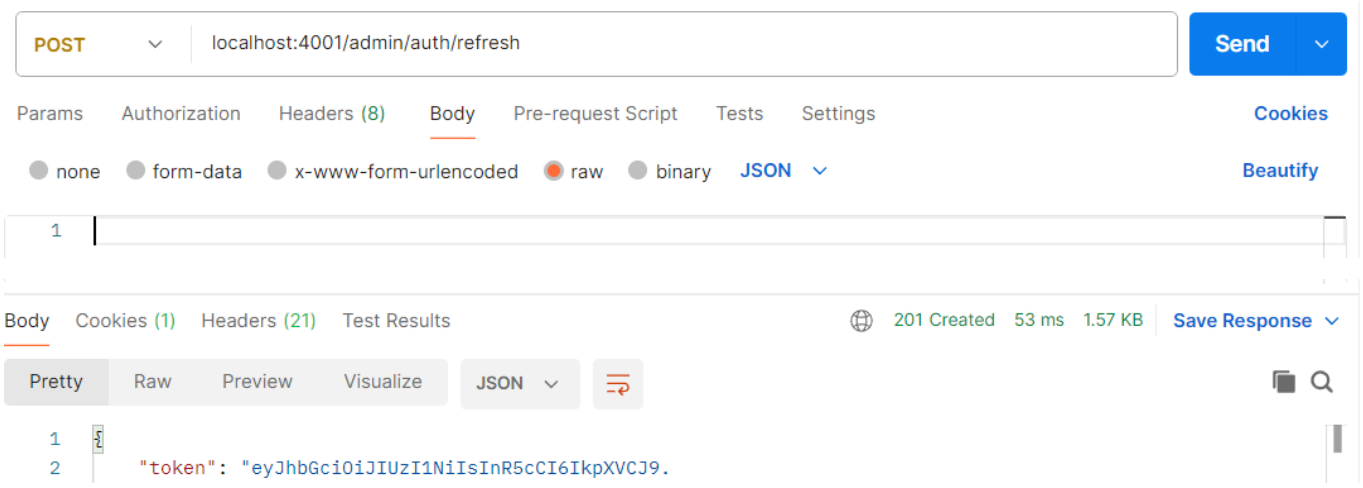


Рисунок 4.14 – Успішне оновлення токену

Для ендпоінту «/admin/documents/f0assignments/next» протестований процес отримання наступних завдань для адміністратора, пов’язаних із перевіркою заявок.

Система працювала коректно, повертаючи інформацію про загальну кількість завдань у черзі. У разі наявності завдань на виконання система також надавала деталі конкретного завдання, яке було призначене адміністратору. Це забезпечує зручність та ефективність роботи адміністратора із заявками (приклад наведено на рисунку 4.15).

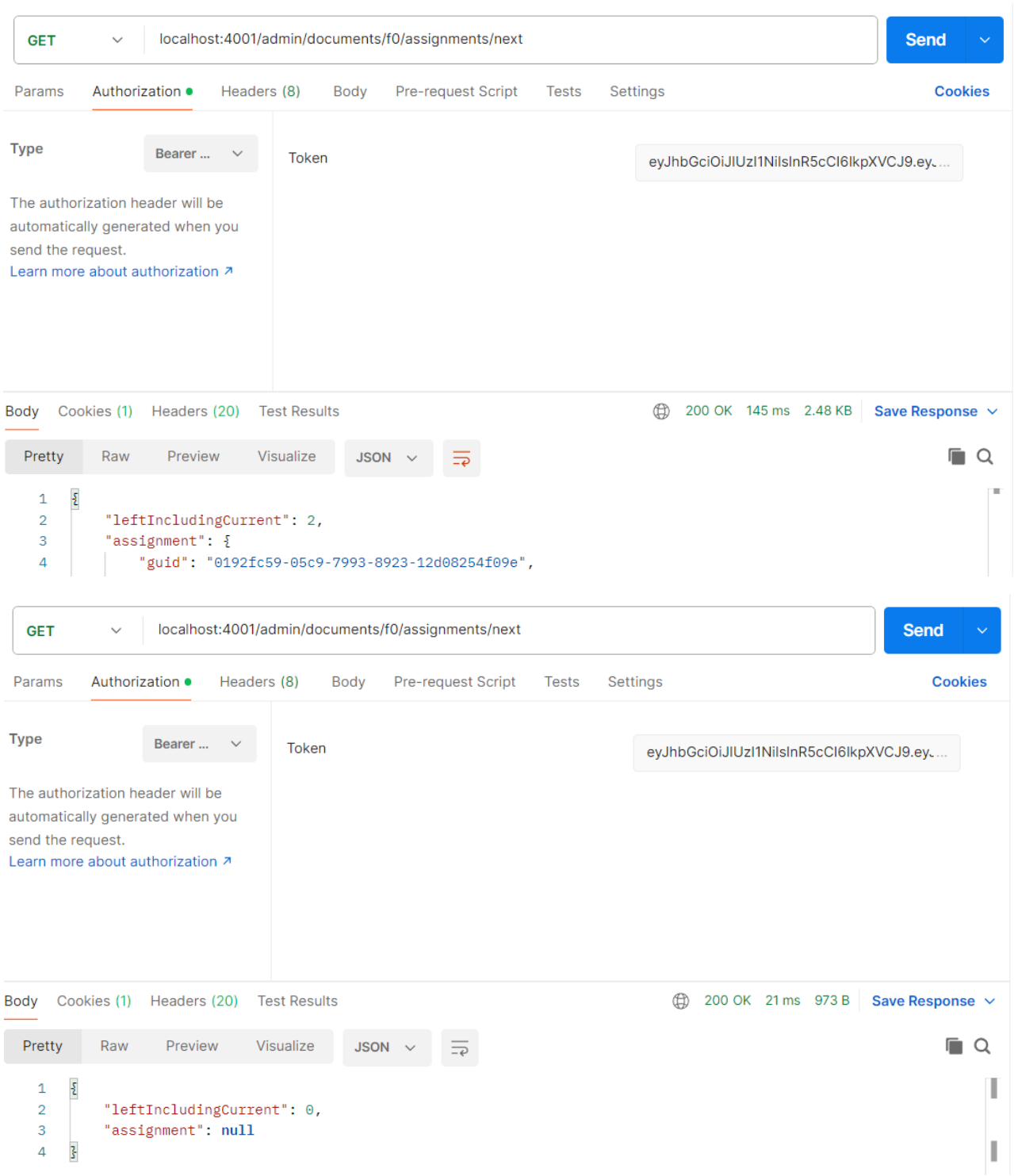


Рисунок 4.15 – Успішне отримання наступних завдань (присутня загальна кількість та, за наявності, завдання)

Під час тестування функції перевірки заявки адміністратором система коректно реагувала на різні сценарії. Перевірено, що у разі відсутності необхідних параметрів у запиті система повертає помилку (див. рисунок 4.16). Крім того, система блокує зміну заявки, якщо вона призначена іншому адміністратору (див. рисунок 4.17). У

випадку коректного запиту, коли адміністратор має доступ до власного завдання, система успішно виконує зміну статусу заявки та повертає оновлені дані (див. рисунок 4.18).

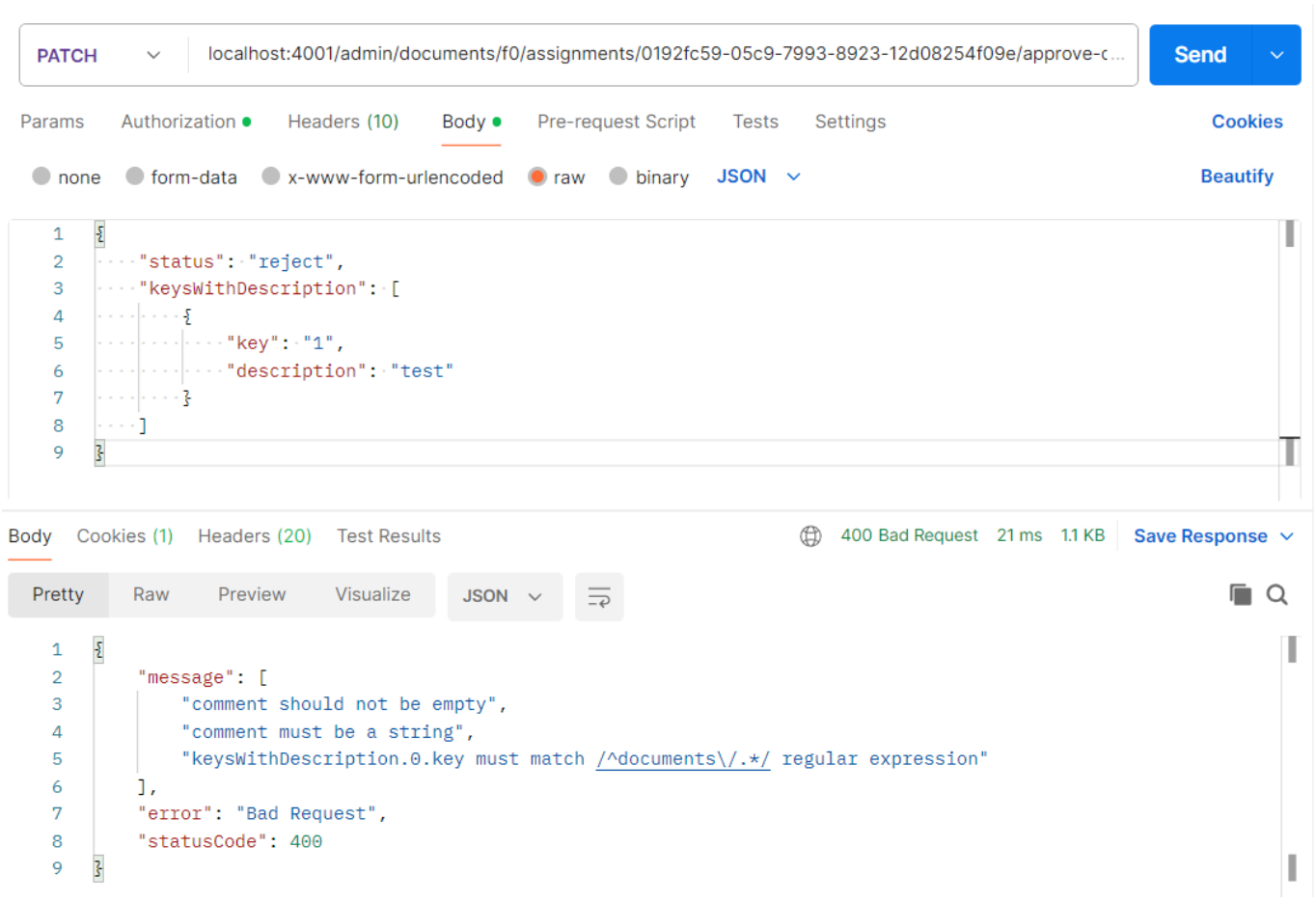


Рисунок 4.16 – Неуспішна перевірка заявки (відсутність параметрів)

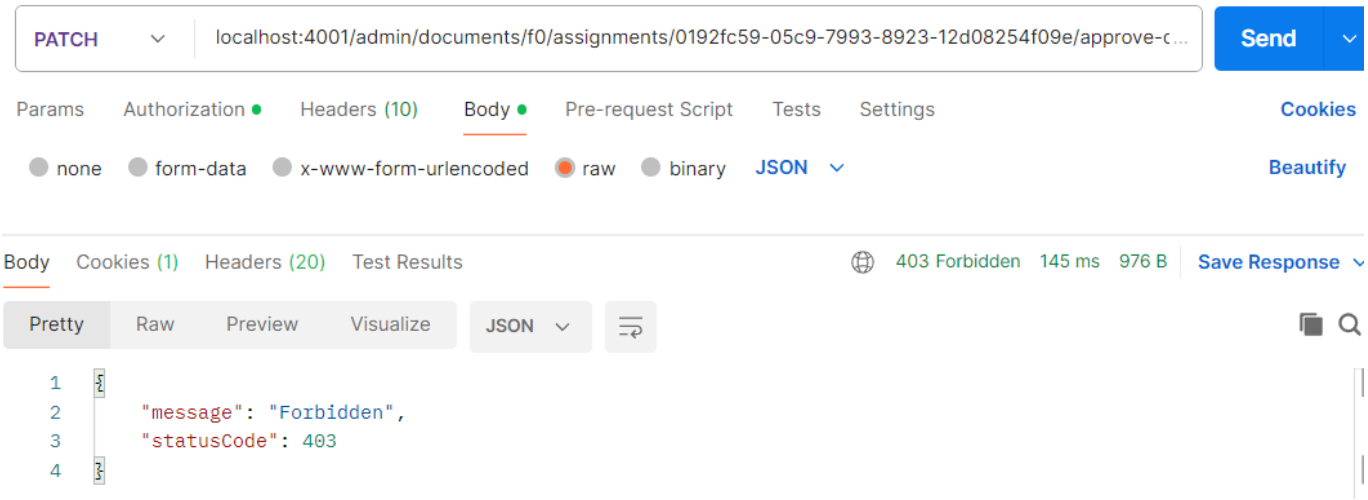


Рисунок 4.17 – Неуспішна перевірка заявки (запит відхилено, бо завдання не належить цьому адміністратору)

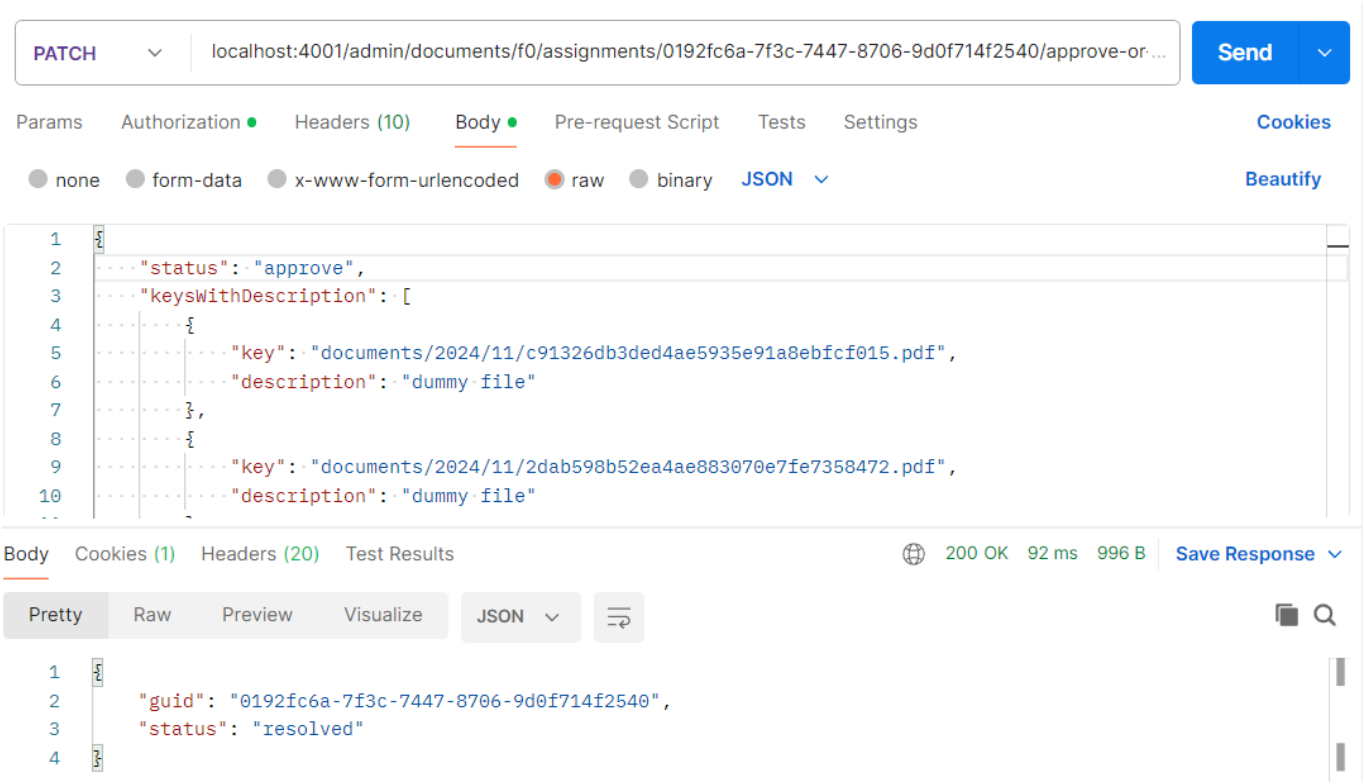


Рисунок 4.18 – Успішна перевірка заявки

Система успішно повертає інформацію про зареєстровану заявку, зокрема коректно приховує особисті дані для публічного доступу (див. рисунок 4.20). Також перевірена обробка помилок: у випадку запиту на заявку, якої не існує, система повертає відповідне повідомлення про відсутність даних (див. рисунок 4.19).

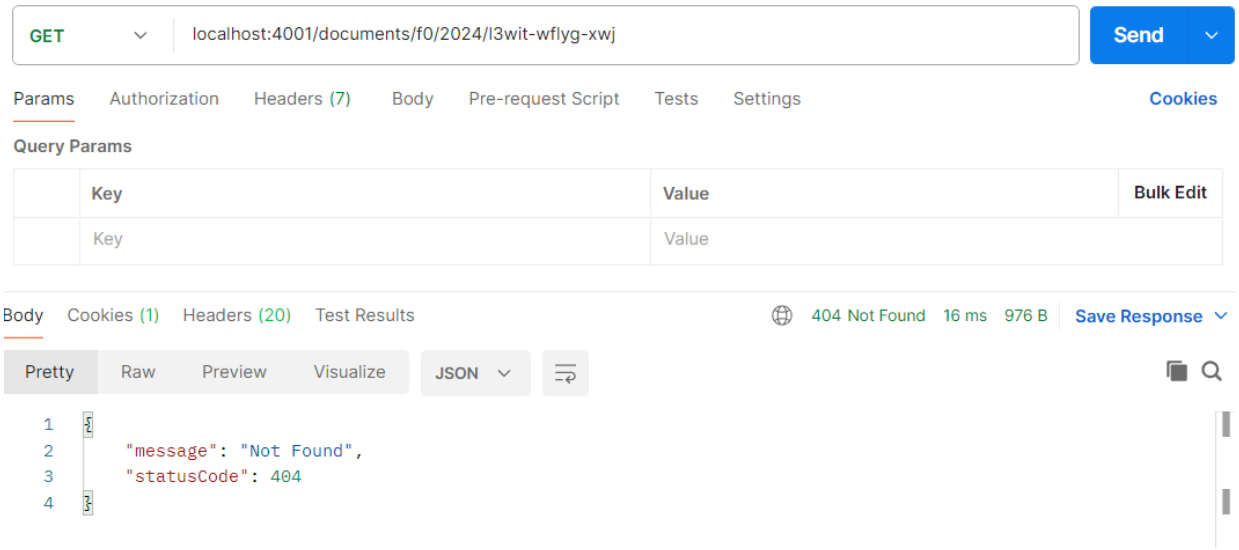


Рисунок 4.19 – Неуспішне отримання заявки (не знайдено)

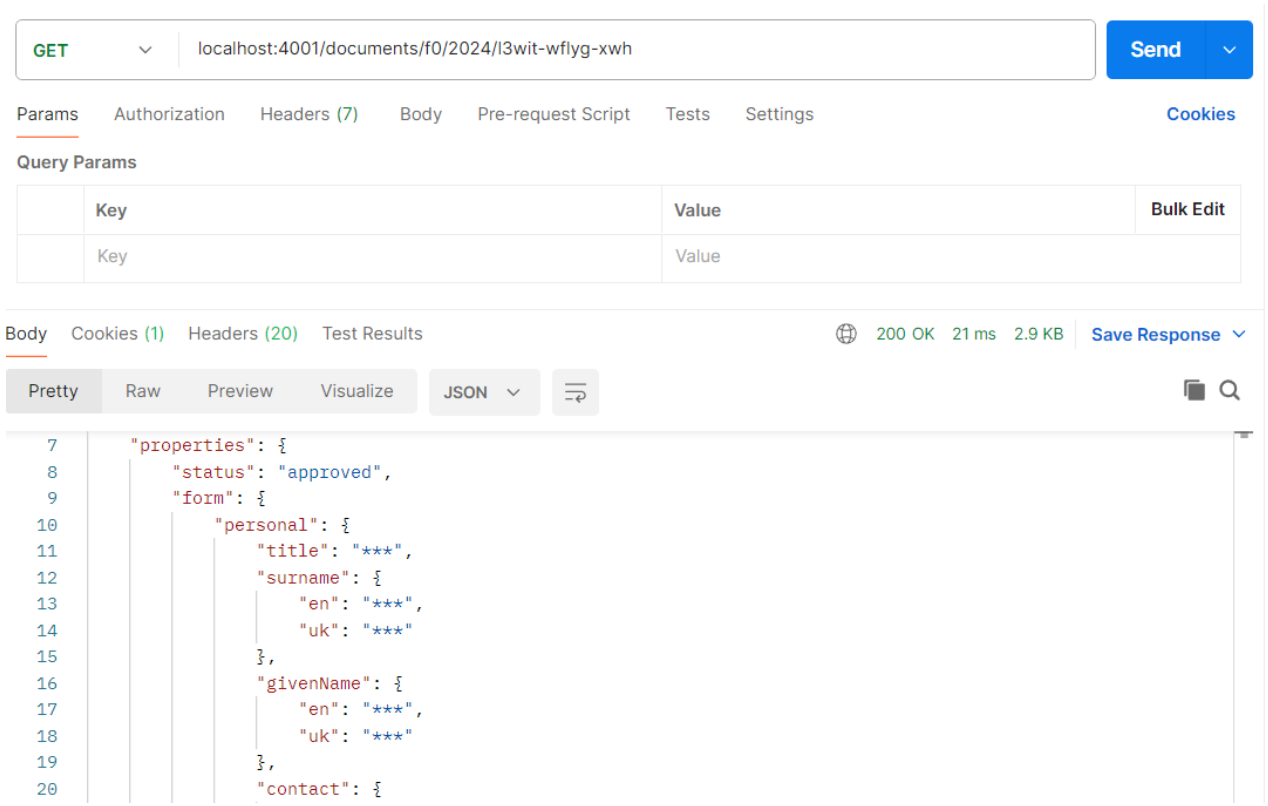


Рисунок 4.20 – Успішне отримання заявки

Система також успішно обробляє запити на отримання інформації про справу, забезпечуючи коректне відображення даних та приховуючи конфіденційну інформацію у публічних запитах (див. рисунок 4.22). У разі відсутності запитуваної справи система повертає повідомлення про помилку, що свідчить про правильне опрацювання такого випадку (див. рисунок 4.21).

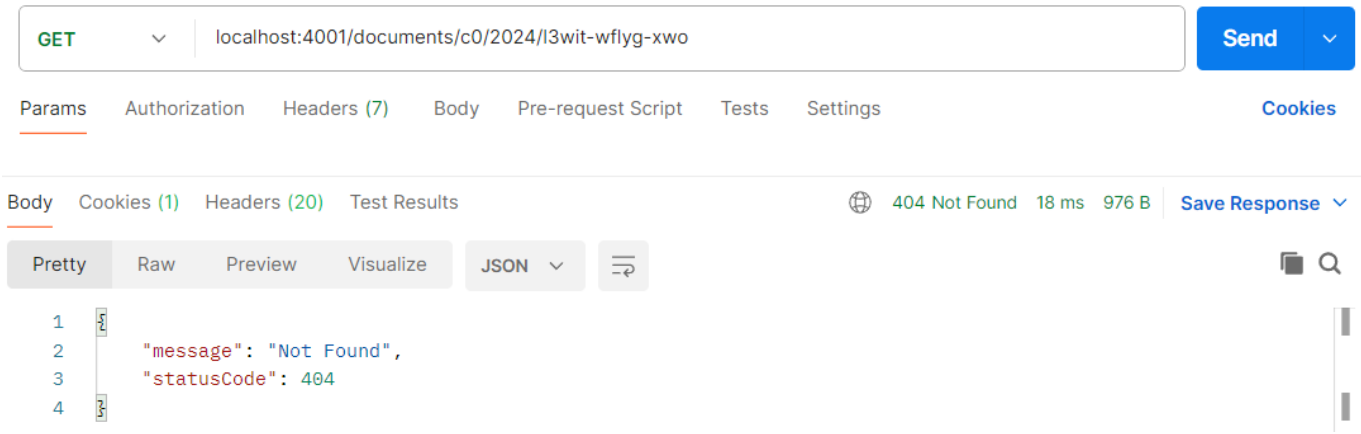


Рисунок 4.21 – Неуспішне отримання справи (не знайдено)

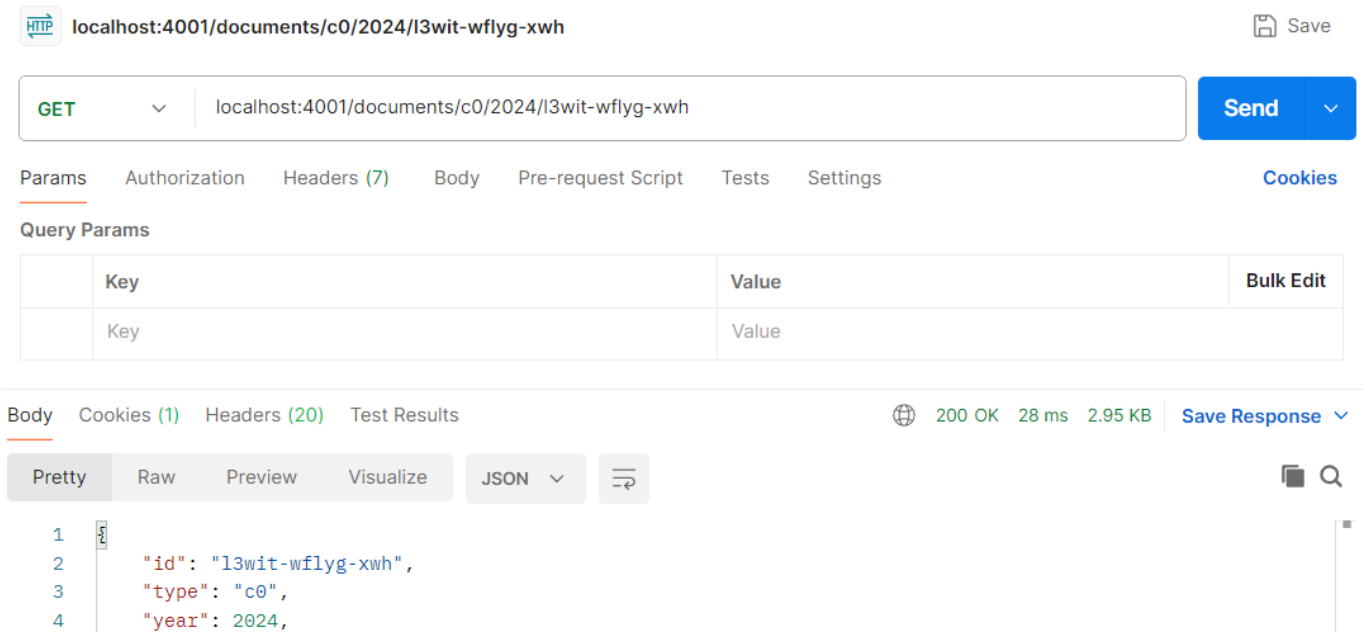


Рисунок 4.22 – Успішне отримання справи

4.2 Перевірка доступності

Доступність клієнтської частини системи перевірена за допомогою автоматизованого інструменту Lighthouse на різних сторінках.

Результати тестування продемонстрували високий рівень доступності, з показниками, що наближаються до максимального значення (100): 98, 98, 94, 100, 98, 100, 96. Детальні результати наведені на рисунку 4.23.

Окрім цього, проведені додаткові ручні перевірки підтвердили високий рівень доступності, який оцінено у 9 із 10 можливих балів (приклад наведено на рисунку 4.24).

Загалом система відповідає основним стандартам доступності. Водночас тестування виявило певні недоліки, які потребують доопрацювання. Для їх усунення та більш глибокого аналізу необхідно провести розширене ручне тестування із залученням фахівців з доступності, щоб забезпечити повну відповідність вимогам.

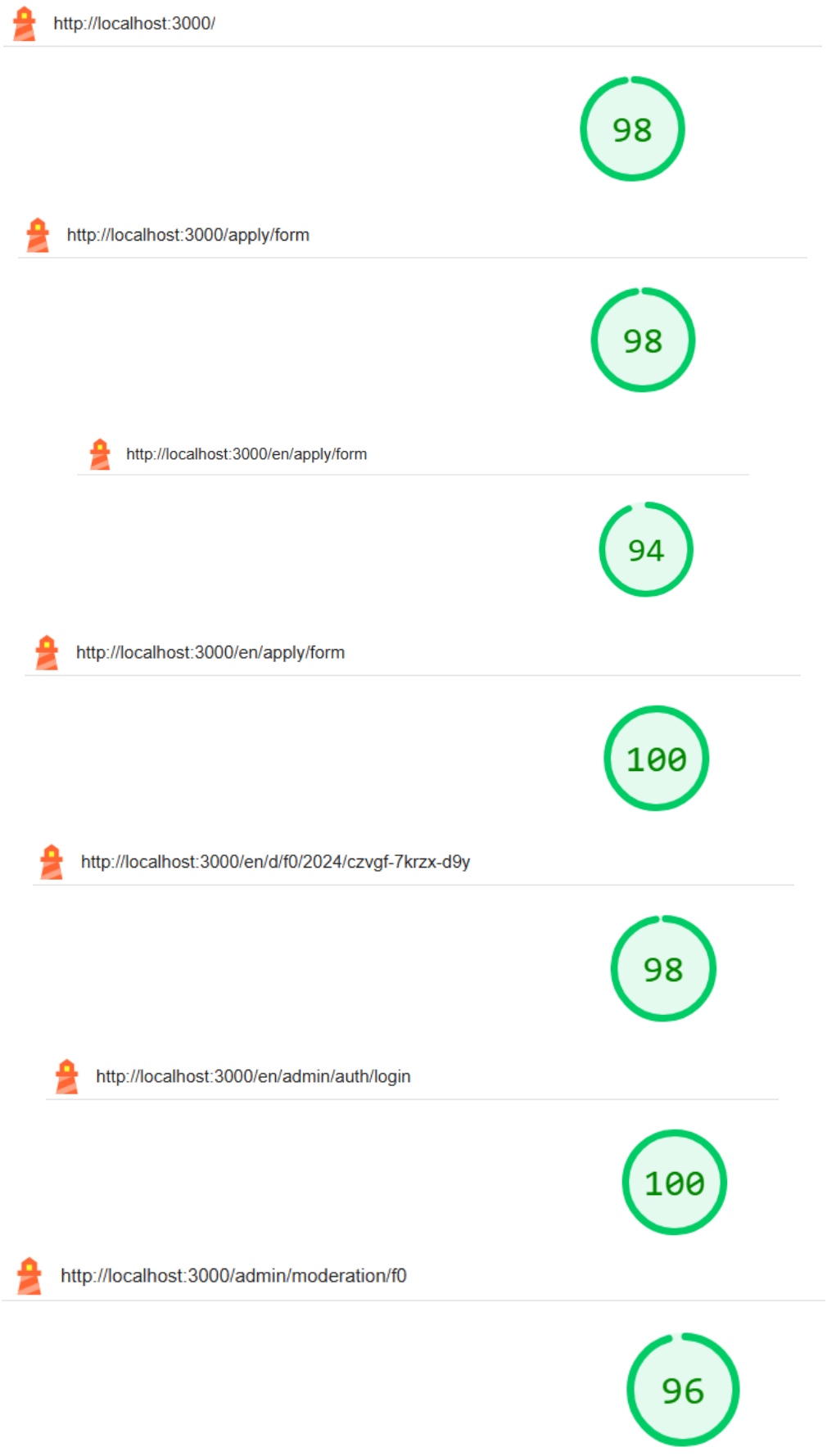


Рисунок 4.23 – Результати тестування доступності на різних сторінках клієнтської частини системи

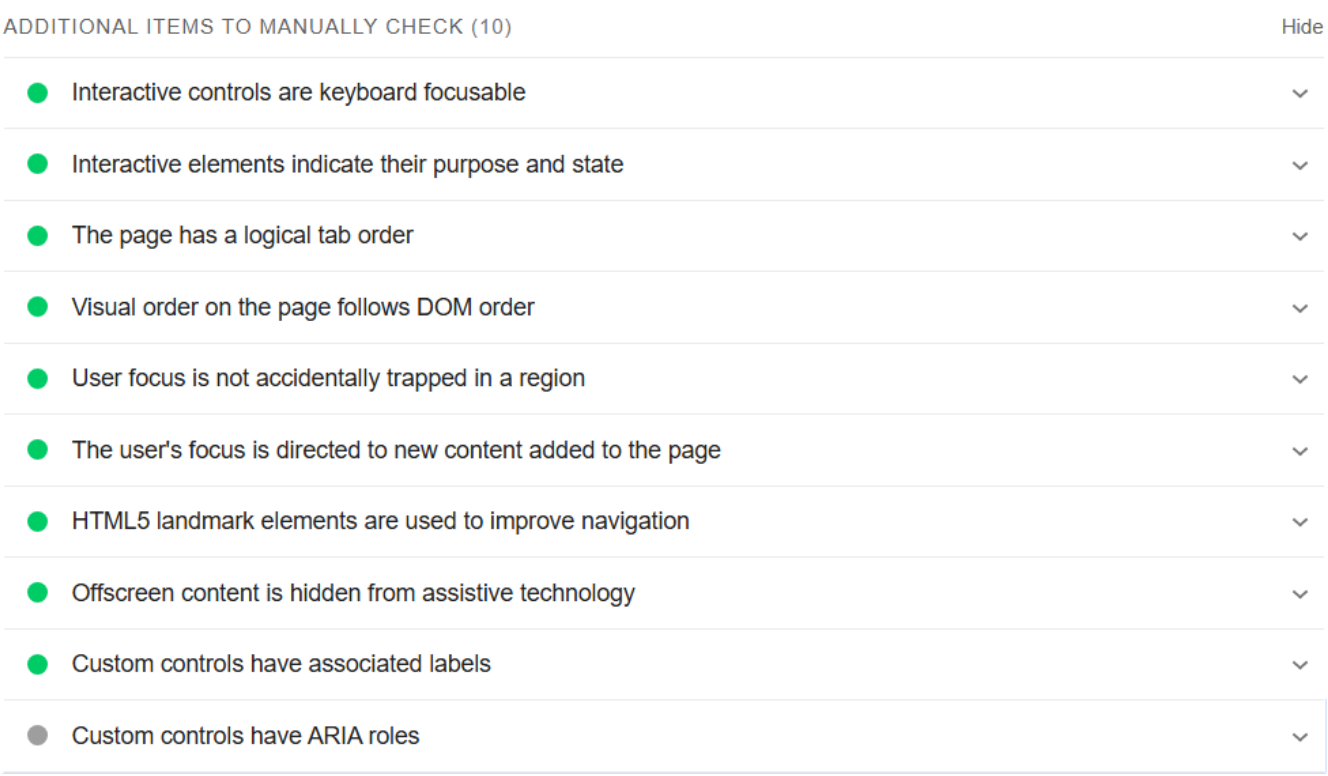


Рисунок 4.24 – Результати ручного тестування (зелені успішні, сірі – потребує додаткового вивчення)

Висновки до розділу 4

У розділі 4 перевірені функціональність та стабільність всіх ключових компонентів системи.

Тестування проводилося вручну із застосуванням інструментів, таких як Postman, для імітації запитів та перевірки відповідей сервера. Перевірені різні сценарії взаємодії з системою, включаючи обробку правильних та некоректних параметрів, створення заявок, завантаження файлів, а також процеси авторизації й управління завданнями адміністраторів.

У результаті система демонструє стабільну роботу, надаючи чіткі повідомлення про помилки та успішно виконуючи запити, що свідчить про готовність до реального використання.

Також перевірена доступність системи, переконуючись, що всі її компоненти працюють належним чином відповідно до вимог щодо забезпечення доступності для людей з обмеженими можливостями.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Інформаційно-реєстрова система зруйнованого майна та інфраструктури створена для збору, обліку та управління інформацією про пошкоджене або знищене майно внаслідок військових дій в Україні.

Основна мета проєкту – забезпечення надійного механізму обліку руйнувань та прозорої оцінки збитків, що є важливим для ефективного відновлення та розподілу ресурсів.

У таблиці 5.1 описано ідею стартап-проєкту, напрямки застосування та відповідно вигоди для користувача.

Таблиця 5.1 – Опис ідеї стартап-проєкту [18]

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення єдиної інформаційної системи для реєстрації зруйнованого майна та інфраструктури	1. Державні установи та місцеві органи влади	Полегшення доступу до даних про пошкодження та планування ресурсів для відновлення
	2. Постраждалі власники майна	Спрощення процесів (типу подачі заявок на компенсацію) та можливість контролювати ресурси, які відповідно надаються
	3. Гуманітарні та міжнародні організації	Прозора інформація про масштаб руйнувань для координації допомоги

Основними техніко-економічними характеристиками ідеї є:

– інтеграція з державними реєстрами (типу автоматичне заповнення заявок);

– автоматизація процесів (типу подання заявок);

– прозорість у відшкодуванні збитків (типу користувач має доступ до усієї відповідної інформації);

– підтримка користувачів через зручний інтерфейс;

– доступність для людей з обмеженими можливостями.

Аналогами проєкту (системи, PRPR) можна виділити системи Реєстр пошкодженого та знищеного майна (РПЗМ) та Реєстр збитків для України (RD4U) – детальніше розділ 1.

У таблиці 5.2 наведені порівняння, визначення сильних, слабких, та нейтральних характеристик ідеї проєкту.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту [18]

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		PRPR	РПЗМ	RD4U			
1.	Інтеграція з державними реєстрами	Потенційно має	Має	Має	-	+	-
2.	Автоматизація процесів	Має	Має	Частково	-	-	+
3.	Прозорість у відшкодуванні збитків	Має	Немає	Має	-	-	+

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		PRPR	РПЗМ	RD4U			
4.	Підтримка користувачів через зручний інтерфейс	Має	Має	Має	-	+	-
5.	Доступність для людей з обмеженими можливостями	Має	Невідомо	Невідомо	-	-	+

Зважаючи на характеристики, порівнюючи з аналогами, вбачаючи на те, що проєкт (та аналоги) є неприбутковим та спрямований на вирішення актуальної проблеми, він має значний потенціал для успішної реалізації та може суттєво сприяти відновленню країни.

5.2 Технологічний аудит ідеї проєкту

У таблиці 5.3 описано аналіз технологічної здійсненності ідеї проєкту.

Аналіз технологічної здійсненності проєкту, представлений у таблиці 5.3, показав, що 5 з 6 необхідних технологій для реалізації інформаційно-реєстрової системи є доступними та широко застосовуваними, 1 з 6 вимагає співпраці.

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту [18]

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Єдина база даних для зберігання інформації	SQL/NoSQL бази даних (PostgreSQL, MongoDB тощо)	Наявна	Доступна (вільне ПЗ, широко поширене)
2	Інтерактивний аналітичний дашборд	BI-інструменти (Tableau, Power BI тощо) або власний дашборд	Наявна	Доступна
3	Інтеграція з державними реєстрами та «Дія»	API інтеграція	Наявна	Доступна за умови прямої співпраці (API, документація закриті)
4	Геопросторова візуалізація зруйнованого майна	GIS-технології (Google Maps API, Leaflet)	Наявна	Доступна (відкрите ПЗ або ліцензії)
5	Захищене зберігання даних	Методи шифрування (AES, RSA), захищені сервери	Наявна	Доступна (потребує конфігурації)
6	Модуль користувацької реєстрації та авторизації	OAuth, JWT для автентифікації та авторизації	Наявна	Доступна (відкриті протоколи та бібліотеки)

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

У таблиці 5.4 описується попередня характеристика ринку (потенційного), де розгортатиметься стртап-проєкт.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проєкту [18]

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	Не застосовується
3	Динаміка ринку (якісна оцінка)	Зростає, враховуючи актуальність реєстрації збитків та руйнувань
4	Наявність обмежень для входу	Доступ до державних даних
5	Специфічні вимоги до стандартизації та сертифікації	Обмеження, що стосуються безпеки та захисту інформації
6	Середня норма рентабельності в галузі (або по ринку), %	Не застосовується

Проєкт є соціально значущим, але не комерційно прибутковим, оскільки спрямований на підтримку державних органів, міжнародних гуманітарних організацій та постраждалих громадян. Загальний обсяг продажу не застосовується, а середня норма рентабельності у галузі відсутня, що робить проєкт невигідним для інвесторів, орієнтованих на фінансовий прибуток.

У таблиці 5.5 визначені потенційні групи клієнтів та формуємо перелік вимог до товару для кожної групи.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проєкту [18]

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба у системі для реєстрації пошкодженого та знищеного майна	Постраждалі особи	Потребують швидкого та зручного способу подання заявок на відшкодування збитків	Простота використання. Доступність для людей з обмеженими можливостями. Прозорість процесу
2	Потреба у системі для документування пошкоджень та планування відновлення	Державні органи України	Потребують точних даних для планування відновлення та розподілу ресурсів	Інтеграція з державними реєстрами. Надійність та безпека даних. Можливість аналітики та звітності
3	Потреба у системі для надання допомоги постраждалим	Гуманітарні організації	Потребують дані для надання допомоги та координації зусиль	Доступ до актуальних даних. Можливість співпраці з іншими організаціями. Прозорість та підзвітність

Визначивши потенційні групи клієнтів, проведений аналіз ринкового середовища, щоб краще спланувати напрями розвитку.

У таблиці 5.6 визначена таблиця факторів, які несуть загрозу.

У таблиці 5.7 визначені фактори, які сприяють розвитку проєкту.

Таблиця 5.6 – Фактори загроз [18]

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Нестабільність політичної ситуації	Політична нестабільність може вплинути на реалізацію проєкту та доступ до державних реєстрів	Постійний моніторинг політичної ситуації та адаптація стратегії відповідно до змін
2	Відсутність фінансування	Недостатнє фінансування може затримати або зупинити реалізацію проєкту	Залучення додаткових джерел фінансування (як-от донорські кошти)
3	Конкуренція з боку інших систем	Інші системи можуть запропонувати подібні рішення	Розробка унікальних функцій та переваг, які відрізняють проєкт від конкурентів
4	Технічні проблеми	Можливі технічні проблеми під час розробки та впровадження системи	Залучення висококваліфікованих спеціалістів та проведення регулярного тестування системи

Таблиця 5.7 – Фактори можливостей [18]

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Підтримка з боку держави	Державна підтримка може сприяти швидшій реалізації проєкту	Активна співпраця з державними органами
2	Зростання попиту на прозорі системи	Зростаючий попит на прозорі та ефективні системи управління	Активне просування системи на ринку та демонстрація її переваг
3	Можливість міжнародного співробітництва	Співпраця з міжнародними організаціями може розширити можливості проєкту	Встановлення партнерських відносин з міжнародними гуманітарними організаціями
4	Інноваційні технології	Використання новітніх технологій може підвищити ефективність системи	Постійне оновлення та вдосконалення системи з використанням сучасних технологій

Далі розглянуті пропозиції ринку.

У таблиці 5.8 описані загальні риси конкуренції у середовищі.

У таблиці 5.9 проведений більш детальний аналіз умов конкуренції за моделлю Портера.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку [18]

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія	Основними конкурентами є державні/урядові реєстри, які пропонують подібні послуги	Співпраця з державними органами, впровадження нових функцій та технологій для підвищення ефективності
2. За рівнем конкурентної боротьби: національний	Конкуренція з іншими національними системами, що пропонують схожі послуги	Забезпечення високого рівня функціоналу та зручності використання, активне просування системи
3. За галузевою ознакою: внутрішньогалузева	Конкуренти знаходяться в одній галузі, пропонуючи схожі рішення для реєстрації пошкодженого майна	Розробка продукту, який охоплює застосування популярних технологій та вирішує ключові проблеми
4. Конкуренція за видами товарів: товарно-видова	Товаром є програмне забезпечення для реєстрації пошкодженого майна	Створення конкурентоспроможного продукту, який не поступається аналогам за функціоналом та зручністю

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5. За характером конкурентних переваг: нецінова	Конкуренція базується на якості, функціональності та зручності використання системи	Забезпечення високої якості продукту, активна підтримка користувачів, прозорість процесів
6. За інтенсивністю: не марочна	Товар з'являється на ринку без відомого бренду	Забезпечення безперешкодного виходу проєкту на ринок, активне просування та маркетинг

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером [18]

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Реєстр пошкодженого та знищеного майна (РПЗМ), Реєстр збитків для України (RD4U)	Високі бар'єри входження через необхідність інтеграції з державними реєстрами та забезпечення високого рівня безпеки	Постачальники можуть диктувати умови через обмежену кількість провайдерів високоякісного обладнання	Споживачі можуть диктувати умови через високі вимоги до якості та безпеки системи	Інші системи реєстрації пошкодженого майна, які можуть з'явитися на ринку

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Висновки:	Інтенсивність конкурентної боротьби з боку прямих конкурентів є високою, оскільки існують вже реалізовані системи зі схожим функціоналом	Бар'єри входження в ринок є значними через необхідність інтеграції з державними реєстрами та забезпечення високого рівня безпеки	Постачальники можуть диктувати умови через обмежену кількість провайдерів високоякісного обладнання	Споживачі можуть диктувати умови через високі вимоги до якості та безпеки системи	Обмеження для роботи на ринку можуть виникати через появу нових систем реєстрації пошкодженого майна

За результатами аналізу таблиці 5.9 можна зробити висновок, що на даний момент на ринку існують прямі конкуренти, які вже закріпили свої позиції. Проте, враховуючи високі бар'єри входження, зокрема значні інвестиції в інфраструктуру, складність технологічних рішень та необхідність інтеграції з державними реєстрами, можливість появи нових конкурентів є обмеженою.

Для того, щоб проєкт був конкурентоспроможним на ринку, він повинен мати не лише високий рівень інтеграції з державними реєстрами, але й забезпечувати максимальний рівень автоматизації, зручності для користувачів, прозорості процесів та відповідності законодавчим вимогам. Крім того, важливо орієнтуватися на інноваційні рішення та забезпечувати безпеку даних, що є ключовим фактором довіри споживачів у даній сфері.

Зважаючи на таблиці 5.9, 5.2, 5.5, 5.6 та 5.7 – визначаємо перелік факторів конкурентоспроможності, що відображено у таблиці 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності [18]

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Інтеграція з державними реєстрами	Високий рівень інтеграції з державними реєстрами забезпечує точність даних та спрощує процеси
2	Автоматизація процесів	Автоматизація процесів знижує ризик помилок та економить час користувачів
3	Прозорість	Прозорість підвищує довіру користувачів та забезпечує юридичну ясність

Визначивши перелік факторів конкурентоспроможності проведемо аналіз сильних та слабких сторін стартап-проєкту в таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін стартап-проєкту [18]

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим проєктом						
			–3	–2	–1	0	+1	+2	+3
1	Інтеграція з державними реєстрами	1							RD4U, РПЗМ
2	Автоматизація процесів	20				RD4U, РПЗМ			
3	Прозорість	20	РПЗМ			RD4U			

Зважаючи на таблиці 5.6, 5.7 та 5.11 складемо SWOT матрицю у таблиці 5.12.

Таблиця 5.12 – SWOT- аналіз стартап-проєкту [18]

Сильні сторони: автоматизація процесів, прозорість	Слабкі сторони: відсутність інтеграції з державними реєстрами
Можливості: співпраця державою, зростання попиту на прозорі системи, можливість міжнародного співробітництва, використання новітніх технологій, розширення функціоналу системи	Загрози: політична нестабільність, відсутність фінансування, технічні проблеми, залежність від постачальників технологій

Провівши SWOT-аналіз розробимо перелік заходів для виведення нашого стартап-проєкту на ринок, що описано в таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проєкту [18]

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Співпраця з державними органами для інтеграції системи	Середня	1-2 роки
2	Пошук міжнародної донорської підтримки	Середня	1-3 роки
3	Розроблення та впровадження маркетингової кампанії для залучення користувачів	Висока	6 місяців - 1 рік

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
4	Партнерство з гуманітарними організаціями для розширення функціоналу системи	Середня	1-2 роки

Для реалізації даного проекту обрані альтернативні поведінки як-от співпраця з державними органами для інтеграції системи та розроблення та впровадження маркетингової кампанії для залучення користувачів.

З означених альтернатив обрані дві, для яких отримання ресурсів є більш простим та ймовірним, а строки реалізації – більш стислі.

5.4 Розроблення ринкової стратегії проєкту

Для розроблення ринкової стратегії опишемо цільові групи потенційних споживачів у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів [18]

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Постраждалі особи	Висока	Високий	Низькій	Висока
2	Державні органи України	Середня	Середній	Середній	Середня

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
3	Гуманітарні організації	Середня	Середній	Низькій	Середня

За результатами аналізу потенційних груп споживачів (сегментів) обрані наступні цільові групи: постраждалі особи, державні органи України, гуманітарні організації.

Для реалізації проєкту обрана стратегія диференційованого маркетингу, оскільки компанія працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу. Це дозволить максимально ефективно задовольнити потреби кожної цільової групи та забезпечити успішне впровадження проєкту на ринку.

У таблиця 5.15 описана стратегія розвитку в обраних сегментах.

Таблиця 5.15 – Визначення базової стратегії розвитку [18]

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Співпраця з державними органами для інтеграції системи	Диференційова-ний маркетинг	Високий рівень інтеграції з державними реєстрами, прозорість у відшкодуванні збитків	Стратегія диференціації

Для роботи в обраних сегментах ринку обрана альтернатива розвитку проєкту: співпраця з державними органами для інтеграції системи. Альтернатива передбачає використання стратегії диференційованого маркетингу та базової стратегії диференціації, що дозволить проєкту виділитися на ринку завдяки унікальним властивостям, забезпечити підтримку користувачів та підвищити рентабельність за рахунок прихильності клієнтів.

Після цього обрана стратегія конкурентної поведінки у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки [18]

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Шукати нових споживачів	Копіювання можливе лише фундаментальних (основних) принципів, процесів, атрибутів чи технологій	Стратегія заняття конкурентної ніші

На основі вимог споживачів у таблиці 5.5, базової стратегії розвитку у таблиці 5.15 та стратегії конкурентної поведінки у таблиці 5.16 розроблена стратегія позиціонування у таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування [18]

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота використання, доступність для людей з обмеженими можливостями, прозорість процесу	Стратегія диференціації	Підтримка користувачів, доступність для людей з обмеженими можливостями, прозорість у відшкодуванні збитків	Зручність, доступність, прозорість
2	Високий рівень інтеграції з державними реєстрами, точність даних, безпека	Стратегія диференціації	Високий рівень інтеграції з державними реєстрами, прозорість у відшкодуванні збитків	Інтеграція, точність, безпека

Обрані асоціації для позиціонування проекту включають зручність, доступність, прозорість, інтеграцію, точність та безпеку, що дозволить ефективно

задовольнити потреби цільових груп споживачів та забезпечити успішне впровадження проєкту на ринку.

5.5. Розроблення маркетингової програми стартап-проєкту

Спочатку формується маркетингова концепція товару у таблиці 5.18.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару [18]

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Простота використання	Легкість використання та доступність для всіх користувачів	Зручний інтерфейс, доступність для людей з обмеженими можливостями
2	Прозорість процесу	Прозорість у відшкодуванні збитків та юридична ясність	Прозорість у відшкодуванні збитків, підтримка користувачів
3	Інтеграція з державними реєстрами	Точність даних та спрощення процесів	Високий рівень інтеграції з державними реєстрами, безпека даних

Далі описується маркетингова модель товару у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару [18]

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Задоволення потреби у реєстрації пошкодженого та знищеного майна, забезпечення прозорості у відшкодуванні збитків, планування відновлення та надання допомоги постраждалим</td></tr>
<tr><td rowspan="4">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Економічні: низька вартість експлуатації.
Призначення: можливість реєстрації пошкодженого майна, інтеграція з державними реєстрами (потенційно).
Ергономічні: зручний інтерфейс, доступність для людей з обмеженими можливостями.
Безпека: високий рівень захисту даних.</td><td>-/+</td><td>+/+/+/+/+</td></tr>
<tr><td colspan="2">Якість: постійне та інтегроване в розробку тестування, підтримка зворотного зв’язку з клієнтами</td></tr>
<tr><td colspan="2">Марка: PRPR</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: маркетингова кампанія для залучення користувачів</td></tr>
<tr><td colspan="3">Після продажу: підтримка користувачів, оновлення та вдосконалення системи</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: розробка та реєстрація патентів на кардинальні розробки проєкту</td></tr>
</table>

Наступним кроком визначаються цінові межі потенційного товару у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни [18]

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Не застосовується	Не застосовується	Проєкт неприбутковий	Не застосовується

Далі визначаються оптимальна система збуту в таблиці 5.21.

Таблиця 5.21 – Формування системи збуту [18]

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Державні органи України: централізовані закупівлі, високі вимоги до безпеки та інтеграції	Забезпечення інтеграції з державними реєстрами, підтримка користувачів, забезпечення безпеки даних	Неглибока	Власна система збуту
2	Гуманітарні організації: потреба в актуальних даних для надання допомоги	Надання доступу до актуальних даних, підтримка користувачів, забезпечення прозорості	Неглибока	Власна система збуту
3	Постраждалі особи: індивідуальні заявки, потреба в простоті використання	Забезпечення зручного інтерфейсу, підтримка користувачів, забезпечення доступності	Неглибока	Власна система збуту

Після цього формується концепція маркетингової комунікації у таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій [18]

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціону-вання	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Державні органи України	Офіційні державні канали, спеціалізовані конференції, урядові портали	Інтеграція, точність, безпека	Інформування про можливості інтеграції та безпеки системи	Підкреслити можливий рівень інтеграції з державними реєстрами та безпеку даних
2	Гуманітарні організації	Спеціалізовані форуми, конференції, соціальні мережі	Зручність, доступність, прозорість	Інформування про доступність та прозорість системи	Підкреслити зручність використання та прозорість процесів
3	Постраждалі особи	Соціальні мережі, місцеві ЗМІ, реклама	Зручність, доступність, прозорість	Інформування про простоту використання та доступність системи	Підкреслити зручність інтерфейсу, процесів та доступність для всіх користувачів

Висновки до розділу 5

Розроблення стартап-проєкту інформаційно-реєстрової системи для обліку зруйнованого майна та інфраструктури є стратегічно важливим для відновлення України при умовах постконфліктного середовища. Виконаний аналіз ринкових можливостей свідчить про потенційну наявність високого попиту на запропоноване рішення, зокрема, на актуальність прозорих систем обліку та управління. Хоча проєкт є соціально орієнтованим та не передбачає прибутку, його значимість може забезпечити ймовірна широка підтримка з боку громад, держави та гуманітарних організацій.

Перспективи впровадження підкріплюються можливістю співпраці з державними структурами, такими як «Дія», а також міжнародними донорами, що може сприяти прискоренню розроблення. Однак бар'єрами є політична нестабільність та складнощі з інтеграцією в існуючі державні реєстри. Водночас стан конкуренції на ринку порівняно низький, і основними суперниками є наявні державні системи. Потенційні конкурентні переваги проєкту – інтеграція, прозорість, підтримка користувачів – дозволяють йому бути перспективним.

Для ринкової реалізації найдоцільнішими є дві альтернативи: активна співпраця з державними установами для інтеграції системи та розроблення маркетингової кампанії з метою залучення користувачів.

ВИСНОВКИ

У ході дослідження створена комплексна інформаційно-реєстрова система для реєстрації та управління інформацією про зруйноване майно та інфраструктуру в Україні.

Розглянуті існуючі рішення, зокрема Реєстр пошкодженого та знищеного майна (РПЗМ) та Реєстр збитків для України (RD4U). Проведений аналіз їхніх функціональних можливостей та обмежень. Виявлено, що ці системи, хоча і є важливими інструментами в умовах війни, мають низку технічних та правових недоліків.

Визначені обмеження та обсяг системи, зокрема описані ключові функціональні та нефункціональні вимоги.

Розроблена система з використанням сучасних технологій, таких як TypeScript, React (Next.js) для клієнтської частини та NestJS для серверної, із використанням PostgreSQL для бази даних та організації коду у форматі монорепозиторію. Застосовані відкриті джерела для створення структурованих даних в умовах обмеженого доступу до офіційних реєстрів.

Проведене всебічне тестування системи, зокрема ручне тестування та перевірка доступності, щоб забезпечити її відповідність вимогам та стабільність роботи.

Останній розділ роботи присвячений аналізу ринкових можливостей запуску стартап-проєкту на основі створеної системи. Описана бізнес-ідея, проведено технологічний аудит, а також розроблена маркетингова стратегія, враховуючи особливості ринку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України від 23.02.2023 № 2923-IX «Про компенсацію за пошкодження та знищення окремих категорій об'єктів нерухомого майна внаслідок бойових дій, терористичних актів, диверсій, спричинених збройною [...]». URL: https://zakon.rada.gov.ua/laws/show/2923-20 (дата звернення: 19.09.2024).

2. Постанова Кабінету Міністрів України від 13.06.2023 № 624 «Деякі питання забезпечення функціонування Державного реєстру майна, пошкодженого та знищеного внаслідок бойових дій, терористичних актів, диверсій, спричинених [...]». URL: https://zakon.rada.gov.ua/laws/show/624-2023-%D0%BF (дата звернення: 19.09.2024).

3. Як подати заяву про пошкоджене майно в застосунку Дії? URL: https://web.archive.org/web/20240620050131/https://paperless.diia.gov.ua/instruction/yak-podati-zayavu-pro-poskodzene-maino-v-zastosunku-diyi (дата звернення: 19.09.2024).

4. єВідновлення в Дії: отримуйте кошти на ремонт пошкодженого житла. YouTube. URL: https://www.youtube.com/watch?v=fpzyN53rugw (дата звернення: 19.09.2024).

5. Дані про знищене та пошкоджене майно. Чому реєстр досі закритий? Економічна правда URL: https://web.archive.org/web/20240520170242/https://www.epravda.com.ua/columns/2024/05/9/713466/ (дата звернення: 19.09.2024).

6. Поширені запитання. Реєстр збитків для України. URL: https://rd4u.coe.int/uk/faq (дата звернення: 19.09.2024).

7. Як подати заяву про пошкоджене майно в Дії. YouTube. URL: https://www.youtube.com/watch?v=LJ2NRuLGwhM (дата звернення: 19.09.2024).

8. Реєстр збитків для України. КАТЕГОРІЇ ЗАЯВ, ЯКІ МОЖУТЬ БУТИ ВНЕСЕНІ ДО РЕЄСТРУ. URL: https://rd4u.coe.int/documents/358068/386726/Board-RD4U%282027%2907-UA+-+%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D1%96%D1%9

7+%D0%B7%D0%B1%D0%B8%D1%82%D0%BA%D1%96%D0%B2.pdf (дата звернення: 19.09.2024).

9. TypeScript: Documentation. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (дата звернення: 15.11.2024).

10. Documentation: NestJS. URL: https://docs.nestjs.com/ (дата звернення: 15.11.2024).

11. PostgreSQL: About. URL: https://www.postgresql.org/about/ (дата звернення: 15.11.2024).

12. Реєстр територіальної громади. URL: https://rtg.dmsu.gov.ua/home (дата звернення: 15.11.2024).

13. Place Autocomplete. Maps JavaScript API. Google for Developers. URL: https://developers.google.com/maps/documentation/javascript/place-autocomplete (дата звернення: 15.11.2024).

14. GitHub: Medniy2000/ua_locations. URL: https://github.com/Medniy2000/ua_locations (дата звернення: 15.11.2024).

15. MikroORM: TypeScript ORM for Node.js. URL: https://mikro-orm.io/ (дата звернення: 15.11.2024).

16. pino: Documentation. URL: https://getpino.io/ (дата звернення: 15.11.2024).

17. Introduction. shadcn/ui. URL: https://ui.shadcn.com/docs (дата звернення: 15.11.2024).

18. Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с. URL: https://ela.kpi.ua/handle/123456789/35763 (дата звернення: 15.11.2024).