

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інформаційно-пошукова система автоматизації
управління орендними відносинами»**

Виконав студент групи ІПЗм-22-1

РУКОСМОТРОВ Денис Володимирович

Керівник роботи:

КРАВЧЕНКО Світлана Миколаївна

Рецензент:

ЛОКТИКОВА Тамара Миколаївна

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **РУКОСМОТРОВ Денис Володимирович**

Керівник роботи: **КРАВЧЕНКО Світлана Миколаївна**

Тема роботи: **«Інформаційно-статистична система автоматизації управління орендними відносинами»**,
затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 479/с**

Вихідні дані для роботи: **предметом дослідження є моделі й методи аналізу потреб орендодавців та оренарів. Об'єктом дослідження є застосування сучасних технологій розробки та аналіз їх властивостей.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	<i>Кравченко С.М.</i>	22.09.2023р.	22.09.2023р.
2	<i>Кравченко С.М.</i>	22.09.2023р.	22.09.2023р.
3	<i>Кравченко С.М.</i>	22.09.2023р.	22.09.2023р.

Дата видачі завдання: **«22» вересня 2023 р.**

РЕФЕРАТ

Кваліфікаційна магістерська робота являє собою наукове дослідження методів та моделей аналізу даних про потреби й поведінку користувачів веб-ресурсу на прикладі веб системи управління орендними відносинами.

Робота на тему «Інформаційно-статистична система управління орендними відносинами» включає теоретичне обґрунтування та практичну реалізацію комплексу засобів для зміни ціноутворення комунальних та орендних платежів, а також пояснювальну записку.

Пояснювальна записка до магістерської роботи містить 72 сторінки, 40 ілюстрацій та 2 таблиці.

У роботі сформульовано основні задачі, проаналізовано існуючі підходи та інструменти, обґрунтовано вибір оптимальних технологій і методів дослідження на основі задач конкретного проєкту.

КЛЮЧОВІ СЛОВА: МЕТОДИ АВТОМАТИЗАЦІЇ, АНАЛІЗ ПОТРЕБ, СТАТИСТИЧНА СИСТЕМА, АВТОМАТИЗАЦІЯ УПРАВЛІННЯ, ВЕБ-ДОДАТОК, ВЕБ-ПРОЕКТ, ОРЕНДНІ ВІДНОСИНИ.

ABSTARCT

Master's thesis represents a scientific research of methods and models for analyzing data about web-resource user needs and behavior on the example of rental relationship management web-system.

The work on the topic “Information and statistical rental relationship management system” includes theoretical background and practical implementation of tools to alter utility and rental payment pricing, as well as a comprehensive account.

Thesis report contains 72 pages, 40 illustrations and 2 tables.

The work formulates main goals, analyzes existing approaches and tools, substantiates choice of best fitted technologies and research methods based on particular project tasks.

KEY WORDS: AUTOMATION METHODS, NEEDS ANALYSIS, STATISTICAL SYSTEM, MANAGEMENT AUTOMATION, WEB APPLICATION, WEB PROJECT, RENTAL RELATIONSHIPS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ ОРЕНДНИХ ВІДНОСИН ТА ПІДХОДИ ДО ЇХ ІНФОРМАЦІЙНО-ТЕХНОЛОГІЧНОГО ЗАБЕЗПЕЧЕННЯ	10
1.1. <i>Постановка завдання дослідження</i>	10
1.2. <i>Визначення орендних відносин та огляд технічної складової реалізації.....</i>	12
1.3. <i>Потреби користувачів системи</i>	13
1.4. <i>Аналіз аналогів дослідження та програмного продукту.....</i>	14
1.5. <i>Вибір методів аналізу ефективності автоматизованої системи та засобів реалізації програмного продукту.....</i>	23
<i>Висновки до першого розділу</i>	27
РОЗДІЛ 2. ПРОВЕДЕННЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ТА РИЗИКІВ ВПРОВАДЖЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ОРЕНДНИМИ ВІДНОСИНАМИ ТА ПРОЕКТУВАННЯ ДОДАТКУ	28
2.1 <i>Аналіз ефективності впровадження системи автоматизації орендних відносин</i>	28
2.2 <i>Дослідження методів автоматизації процесу орендних відносин та його необхідність</i>	31
2.3 <i>Опис структури даних за якою був розроблений проект</i>	32
2.4 <i>Проектування та реалізація алгоритмів роботи проекту з автоматизації.....</i>	35
<i>Висновки до другого розділу</i>	48
РОЗДІЛ 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОРЕНДНИМИ ВІДНОСИНАМИ.....	49
3.1 <i>Результати дослідження автоматизації процесів управління орендними відносинам.....</i>	49

	6
3.2 Структура інтерфейсу системи	54
<i>Висновки до третього розділу.....</i>	70
ВИСНОВКИ	71
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ	72
ДОДАТКИ	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	– Application Programming Interface
ROI	– Net present value
NPV	– Cascading Style Sheets
БД	– база даних
MERN	– MongoDB, Express.js, ReactJS, Node.js

ВСТУП

Актуальність теми: У сьогоденні більшість процесів до яких ми звикли продовжують розвиватись, ставати більш зручними та швидкими за допомогою автоматизації роботи з ними, що дозволяє покращувати ефективність та надійність користування звичними послугами.

Складно уявити, які буденні процеси ще не перейшли від формату ручного вводу та обов'язкової особистої присутності користувачів послуг до автоматизованого формату роботи та веб застосунків, від оплати комунальних послуг до навчання в університеті, від замовлення доставки їжі до керування підприємницькою діяльністю. Тому я пропоную провести дослідження необхідності та можливостей автоматизації керування орендними відносинами між орендарем та орендодавцем. Звичайна система орендних відносин застаріла, має низку недоліків та може привести до збитків як орендаря так і орендодавця. Для того щоб задовольнити потреби сучасного користувача послуг оренди пропонуємо дослідити методи автоматизації системи орендних відносин, яка допоможе відслідкувати оплати орендарів та зробити виставлення рахунків простим, швидким та надійним.

Мета даної роботи: Є проведення дослідження методів автоматизації процесу орендних відносин розробка на базі веб-додатку для керування орендними відносинами.

Завдання. Для досягнення цієї мети робота включає наступні завдання:

1. Проаналізувати існуючі автоматизовані системи , їх методи та літературні джерела з тематики роботи.
2. Зібрати необхідні дані про потреби користувачів у подібних системах. Зробити аналіз подібних рішень.
3. На основі отриманих результатів дослідження створити ефективне рішення для реалізації нашої системи.
5. Оформити результати дослідження у вигляді магістерської роботи відповідно до встановлених вимог.

Об’єктом дослідження є моделі та методи аналізу потреб орендодавців та оренарів у покращенні існуючої системи відносин.

Предметом дослідження є методи автоматизації управління орендними відносинами та аналізі потреб учасників процесу оренди.

Практичне значення полягає у збільшенні ефективності керування орендними відносинами за допомогою веб додатку на базі Next.js, що сприяє зменшенню навантаження на учасників процесу оренди.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи доповідалися на Тези VI Всеукраїнської науково-технічної конференції «Комп’ютерні технології: інновації, проблеми, рішення» [25, 26].

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ ОРЕНДНИХ ВІДНОСИН ТА ПІДХОДИ ДО ЇХ ІНФОРМАЦІЙНО-ТЕХНОЛОГІЧНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Постановка завдання дослідження

Головною метою даного наукового дослідження є вивчення методів автоматизації та підвищення ефективності процесів управління орендними відносинами на базі сучасної інформаційно-статистичної системи. Це передбачає виконання таких дослідницьких завдань:

1. Детально дослідити економічну сутність та правовий зміст орендних відносин в Україні, проаналізувати характерні риси, принципи, об'єкти, суб'єктів та механізми здійснення орендних операцій.

2. Провести дослідження методів автоматизації процесу орендних відносин, ризиків при впровадженні подібного рішення та оцінити його необхідність.

3. Дослідити вітчизняний та зарубіжний досвід створення автоматизованих систем управління орендними відносинами, проаналізувати функціональні можливості існуючих рішень.

4. На основі аналізу предметної області сформулювати вимоги до функціоналу розроблюваної інформаційно-статистичної системи управління орендними відносинами.

5. Обґрунтувати вибір оптимальних технологій та засобів реалізації запланованого функціоналу системи, визначити архітектуру програмного забезпечення.

6. Змодельовати інформаційне забезпечення та розробити структуру бази даних системи, спроектувати алгоритми реалізації ключових бізнес-процесів.

7. Безпосередньо реалізувати програмне забезпечення розробленої інформаційно-статистичної системи управління орендними відносинами з використанням обраних технологій та засобів.

8. Ґрунтовно протестувати створене програмне забезпечення, оцінити ефективність та практичну значущість розробленої системи.

Успішне виконання цих дослідницько-прикладних завдань дозволить визначити метод автоматизації та створити і впровадити у практику сучасну автоматизовану систему управління орендними відносинами.

Вважаю за потрібне додати, що веб-додатки відіграють важливу роль у перетворенні буденних процесів на більш зручні, автоматизовані та інтуїтивно зрозумілі.

Можна навести безліч прикладів використання подібного підходу для вдосконалення досвіду користування певними послугами, ось декілька з них:

1. **Онлайн-банкінг:** Веб-додатки для онлайн-банкінгу дозволяють користувачам переводити гроші, перевіряти баланс, оплачувати рахунки та виконувати інші банківські операції без необхідності відвідувати банк.

2. **Електронна комерція:** Веб-додатки для електронної комерції, такі як Amazon та eBay, дозволяють користувачам купувати та продавати товари та послуги в Інтернеті.

3. **Освітні платформи:** Веб-додатки, такі як Coursera та Khan Academy, дозволяють студентам вивчати різні предмети та отримувати сертифікати або дипломи від відомих університетів.

4. **Системи управління проектами:** Веб-додатки, такі як Trello та Jira, дозволяють командам ефективно управляти проектами, відслідковувати прогрес та співпрацювати.

5. **Соціальні мережі:** Веб-додатки, такі як Facebook та Twitter, дозволяють користувачам спілкуватися, ділитися контентом та взаємодіяти з іншими користувачами.

У випадку орендних відносин, веб-додаток може автоматизувати процеси, такі як створення договорів оренди, відслідковування платежів, управління витратами на обслуговування та ремонт, а також надання звітів та аналітики для орендодавців. Це не тільки полегшує управління орендними відносинами, але й забезпечує більшу прозорість та ефективність [9].

1.2 Визначення орендних відносин та огляд технічної складової реалізації

Оренда є зручним механізмом для тимчасового отримання у користування необхідних активів без придбання їх у власність. Це дозволяє оптимізувати витрати та підвищити ефективність використання ресурсів. Однак процеси управління орендними відносинами є досить складними та трудомісткими.

З науково-технічної точки зору, орендні відносини являють собою комплекс взаємопов'язаних бізнес-процесів, що включають пошук та підбір об'єктів оренди, укладання і супровід договорів, облік орендарів і орендодавців, нарахування та контроль платежів, формування звітності тощо. Автоматизація таких процесів потребує застосування сучасних інформаційних технологій, зокрема проектування баз даних, розробки алгоритмів обробки інформації, створення зручних користувацьких інтерфейсів.

В орендних відносинах задіяні такі сторони:

- Орендодавець - фізична або юридична особа, яка передає право користування та володіння майном орендареві на обумовлений термін за орендну плату.
- Орендар - фізична або юридична особа, яка отримує право тимчасового користування та володіння майном за плату на підставі договору оренди.

Орендодавцем може бути власник майна або особа, яка має право надання майна в оренду. Орендарем може бути будь-яка особа, що має здатність і потребу у тимчасовому користуванні майном[9].

З позиції інформаційних технологій, учасники орендних відносин виступають як ключові бізнес-ролі в інформаційній системі, для яких необхідно передбачити відповідні функції та права доступу.

Таким чином, комплексне застосування сучасних інформаційних технологій дає змогу автоматизувати та оптимізувати процеси управління в сфері орендних послуг, підвищити їх ефективність.[8]

1.3 Потреби користувачів системи

Управління орендними відносинами вимагає врахування потреб різних груп користувачів. Давайте розглянемо деякі з них:

1. **Орендодавці:** Орендодавці шукають способи максимізувати дохід від своєї власності, зберігаючи при цьому її стан та забезпечуючи її довгострокову цінність. Вони потребують інструментів для ефективного управління орендою, включаючи відслідковування платежів, управління ремонтами та обслуговуванням, а також ведення обліку орендарів. Вони також цінують прозорість у відносинах з орендарями та хочуть мати можливість легко комунікувати з ними.

2. **Орендарі:** Орендарі шукають зручне, безпечне та доступне житло. Вони хочуть, щоб процес оренди був простим та прозорим, з чіткими умовами оренди та без прихованих витрат. Вони також цінують можливість легко комунікувати з орендодавцями та швидко отримувати відповіді на свої запити.

3. **Менеджери власності:** Менеджери власності відповідають за управління власністю від імені орендодавців. Вони потребують інструментів для ефективного управління власністю, включаючи відслідковування платежів, управління ремонтами та обслуговуванням, а також ведення обліку орендарів. Вони також потребують засобів для ефективної комунікації з орендарями та орендодавцями [8].

Наш додаток може задовольнити ці потреби шляхом автоматизації ключових процесів управління орендою, забезпечення прозорості та полегшення комунікації між усіма учасниками. Наприклад, він може автоматизувати створення договорів оренди, відслідковування платежів та управління витратами на обслуговування. Він також може надавати зручний графік тенденцій цін, що допоможе орендодавцям та орендарям робити обґрунтовані рішення [15].

Все це може допомогти зробити процес оренди більш прозорим, ефективним та приємним для всіх учасників. Таким чином, ваш додаток може відіграти важливу роль у вдосконаленні управління орендними відносинами.

1.4 Аналіз аналогів дослідження та програмного продукту

В сучасному світі існує багато систем, які вже використовуються для управління орендними відносинами. Це можуть бути як веб-додатки, так і традиційні методи. Давайте розглянемо декілька з них:

1. **Buildium:** Це хмарна система управління нерухомістю, яка дозволяє менеджерам резидентських та асоціативних об'єктів контролювати всі аспекти свого бізнесу. Вона надає можливість відслідковувати платежі, управляти ремонтами та обслуговуванням, а також вести облік орендарів. Однак, ця система може бути дорогою для деяких користувачів, а також вимагає певного рівня технічних знань для ефективного використання. За словами постачальника, Buildium спрощує управління нерухомістю, автоматизуючи бухгалтерський облік, збір та оплату орендної плати, комунікацію з орендарями та власниками, звітність та інше.

Переваги:

- Масштабованість (може використовуватися для 1 або 5000 одиниць)
- Основний пакет включає всі функції обліку нерухомості
- Автоматизований бухгалтерський облік, збір та оплата орендної плати

Недоліки:

- Електронні договори оренди безкоштовні тільки з версією Plus
- Сплачується плата за налаштування банківських рахунків
- Важко налаштувати плату за прострочення для кожної одиниці
- Звіти можуть бути незручними для бухгалтерів та власників

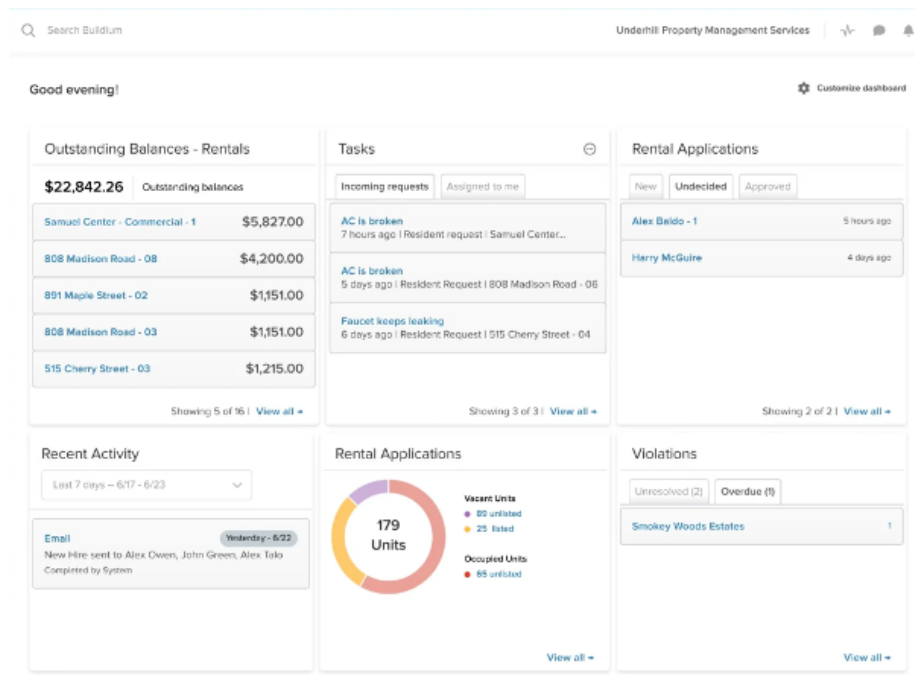


Рис. 1.1 Інтерфейс системи управління нерухомістю **Buildium**

На цьому фото ми можемо бачити головну інформацію про орендні угоди та графік застрахованих та не застрахованих угод. Також присутній блок з завданнями, який допомагає ефективно управляти усіма процесами та налагоджувати їх.

Далі розглянемо сторінку Draft leases.

Тут ми можемо бачити, що сервіс надає зручну таблицю з основною інформацією про кожну угоду, її статус, дати початку та закінчення договору та агента з нерухомості для кожної угоди.

Draft leases

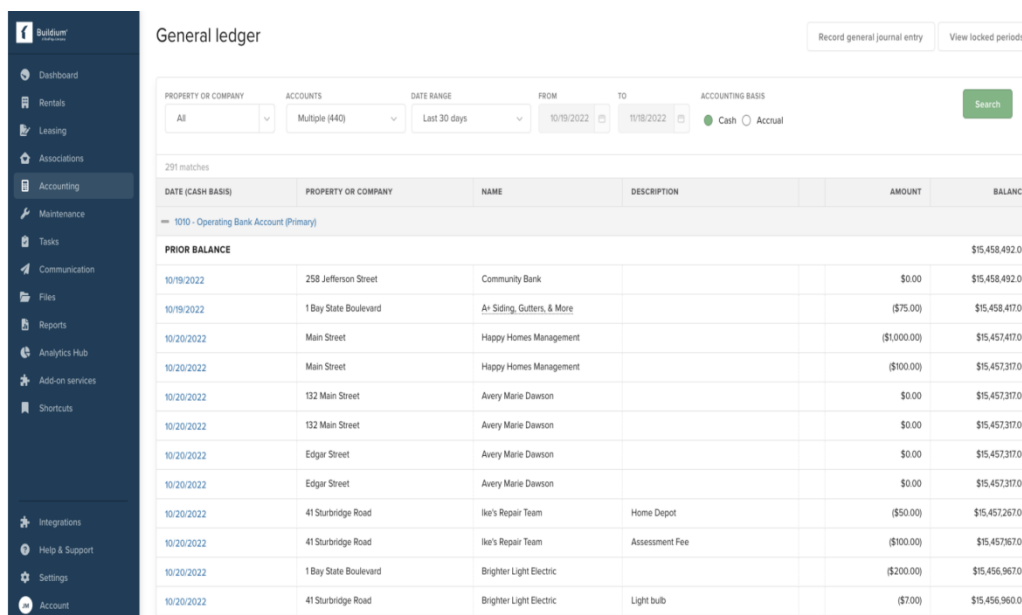
33 matches

LEASE	ESIGNATURE STATUS	LEASE STATUS	AGENT	START DATE	END DATE
312 Oak Street - 02 Jenna McDermott	Awaiting signatures	Not executed	Jenna McDermott	12/01/2022	11/30/2023
22 Main Street - 3 - Room Rental Brian Post	Awaiting signatures	Not executed	Derek Reed	06/01/2022	05/31/2023
Section 8 Property - 51 Jenna McDermott	Awaiting signatures	Not executed	David Logue	11/01/2022	10/31/2023
22 Main Street - 3 - Room Rental Brian Post	Awaiting signatures	Not executed	Christina Hurley	05/01/2022	04/30/2023
312 Oak Street - 02 Jason Evans	Awaiting signatures	Not executed	Brian Gendron	06/01/2022	05/31/2023
22 Main Street - 3 - Room Rental Eric Rondeau	Awaiting signatures	Not executed	Alex Andre	08/01/2022	07/31/2023
Section 8 Property - 51 eric rondeau	Awaiting signatures	Not executed	Alex Andre	10/03/2022	10/02/2023
22 Main Street - 3 - Room Rental Brian Post	Awaiting signatures	Not executed	Aden Kadri	06/01/2022	09/30/2022
312 Oak Street - 02 Mookie Betts	Awaiting signatures	Not executed	Aaron Cohen	05/01/2022	04/30/2023
1 Long Wharf - 11 Joe Samuelman	Awaiting signatures	Not executed		02/15/2022	02/14/2023
1 Parking Way - 21 Joe Samuelman	Awaiting signatures	Not executed		02/15/2022	02/14/2023
11 Lincoln Street - 2 Joe Samuelman	Awaiting signatures	Not executed		02/16/2022	02/15/2023
1000 rehab fake property - 11 Joe Samuelman	Awaiting signatures	Not executed		02/22/2022	02/21/2023
100 Clarendon Street - 11 Joe Samuelman	Awaiting signatures	Not executed		02/22/2022	02/21/2023
1000 Massachusetts Avenue - 11 Joseph Samuelman	Awaiting signatures	Not executed		02/23/2022	02/22/2023

Рис. 1.2 Сторінка Draft leases системи управління нерухомістю **Buildium**

Наступною ми розглянемо сторінку General ledger.

Тут ми можемо бачити список компаній з якими заключені угоди оренди, їхні назви, опис та баланс.



General ledger

Record general journal entry View locked periods

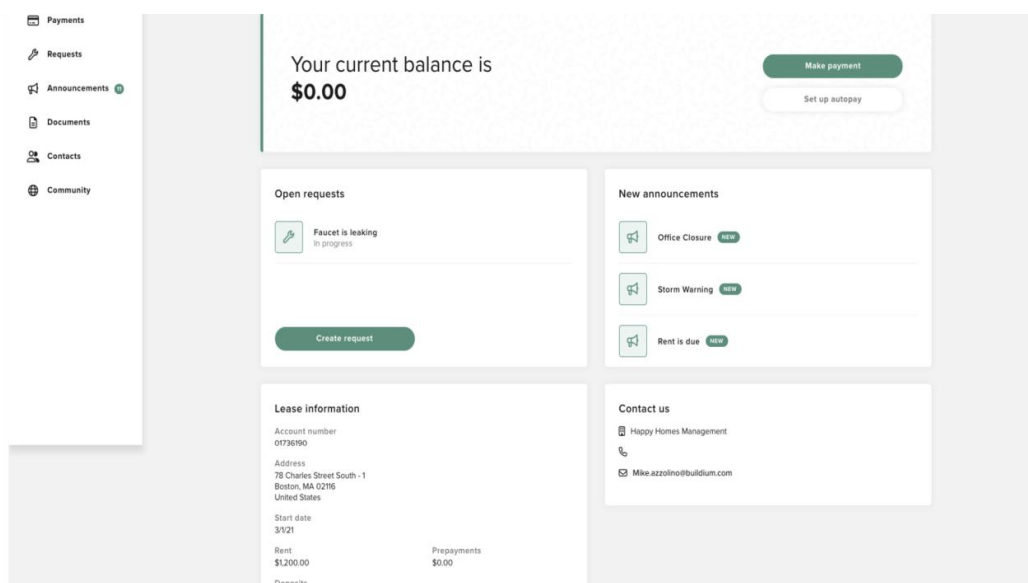
PROPERTY OR COMPANY: All ACCOUNTS: Multiple (40) DATE RANGE: Last 30 days FROM: 10/19/2022 TO: 11/18/2022 ACCOUNTING BASIS: Cash Accrual

291 matches

DATE (CASH BASIS)	PROPERTY OR COMPANY	NAME	DESCRIPTION	AMOUNT	BALANCE
PRIOR BALANCE					\$15,458,492.00
10/19/2022	258 Jefferson Street	Community Bank		\$0.00	\$15,458,492.00
10/19/2022	1 Bay State Boulevard	Ar Siding, Gutters, & More		(\$75.00)	\$15,458,417.00
10/20/2022	Main Street	Happy Homes Management		(\$1,000.00)	\$15,457,417.00
10/20/2022	Main Street	Happy Homes Management		(\$100.00)	\$15,457,317.00
10/20/2022	132 Main Street	Avery Marie Dawson		\$0.00	\$15,457,317.00
10/20/2022	132 Main Street	Avery Marie Dawson		\$0.00	\$15,457,317.00
10/20/2022	Edgar Street	Avery Marie Dawson		\$0.00	\$15,457,317.00
10/20/2022	Edgar Street	Avery Marie Dawson		\$0.00	\$15,457,317.00
10/20/2022	41 Sturbridge Road	Ile's Repair Team	Home Depot	(\$50.00)	\$15,457,267.00
10/20/2022	41 Sturbridge Road	Ile's Repair Team	Assessment Fee	(\$100.00)	\$15,457,167.00
10/20/2022	1 Bay State Boulevard	Brighter Light Electric		(\$200.00)	\$15,456,967.00
10/20/2022	41 Sturbridge Road	Brighter Light Electric	Light bulb	(\$700)	\$15,456,960.00

Рис. 1.3 Сторінка General ledger системи управління нерухомістю **Buildium**

Останньою розглянемо домашню сторінку об'єкта нерукозності, на якій представлений поточний баланс компанії, інформація з договору про оренду та кнопки для оплати оренди та налаштування автоматичної оплати.



Payments Requests Announcements Documents Contacts Community

Your current balance is **\$0.00**

Make payment Set up autopay

Open requests

Faucet is leaking In progress

Create request

New announcements

Office Closure New

Storm Warning New

Rent is due New

Lease information

Account number: 01736190

Address: 78 Charles Street South - 1 Boston, MA 02116 United States

Start date: 3/9/21

Rent: \$1,200.00

Prepayments: \$0.00

Deposits

Contact us

Happy Homes Management

Mike azzolino@buildium.com

Рис. 1.4 Сторінка Номе об'єкта нерухомості в системі управління нерухомістю **Buildium**

2. **DoorLoop:** Це програмне забезпечення для управління нерухомістю, яке допомагає автоматизувати та структурувати багато основних бізнес-процесів. Воно дозволяє автоматизувати процеси, такі як створення договорів оренди, відслідковування платежів та управління витратами на обслуговування. Однак, ця система може бути складною для користувачів, які не мають достатнього досвіду роботи з подібними системами.

Переваги:

- Інтуїтивний інтерфейс, який легко використовувати для початківців та досвідчених менеджерів нерухомості.
- Багатий набір функцій, який враховує як потреби орендарів, так і власників.
- Спрощення фінансових аспектів управління нерухомістю, таких як збір орендної плати, відстеження витрат та генерація детальних фінансових звітів.
- Автоматизація робочого процесу та завдань, таких як надсилання нагадувань, створення документів та відповідь на запити орендарів.

Недоліки:

- Обмеження зберігання документів.
- Немає можливості створювати власні звіти.

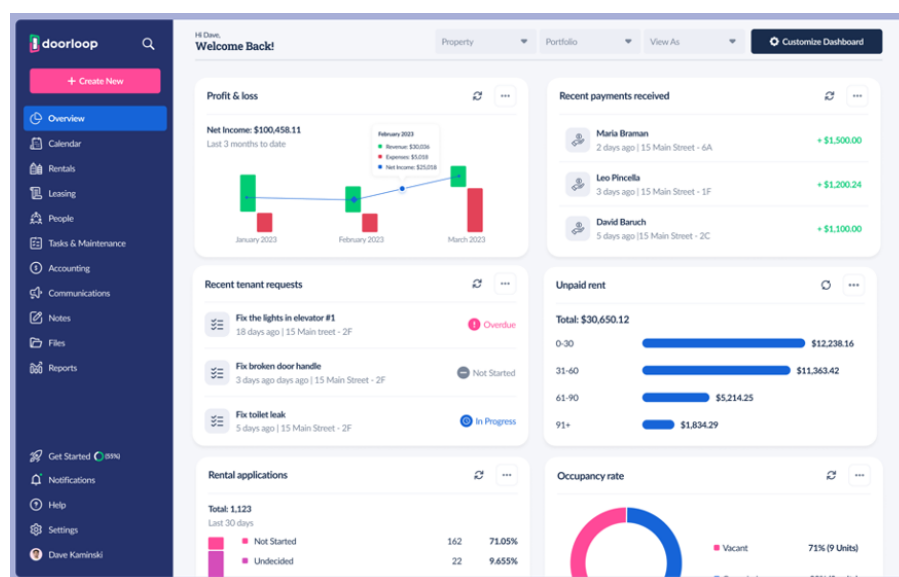


Рис. 1.5 Інтерфейс системи управління нерухомістю **DoorLoop**

На цій сторінці нам представлено основну інформацію для власника об'єкта нерухомості. Серед них: графік доходів підприємства, останні отримані платежі, запити від орендарів з приводу орендних приміщень, неоплачених договорів, графік співвідношень зайнятих та вільних орендних приміщень.

Наступною розглянемо сторінку з орендарями. Тут ми можемо бачити картки орендарів, які вміщують статус, адресу, персональні та контактні дані. Також є можливість фільтрувати орендарів по групі, статусу та об'єктам нерухомості.

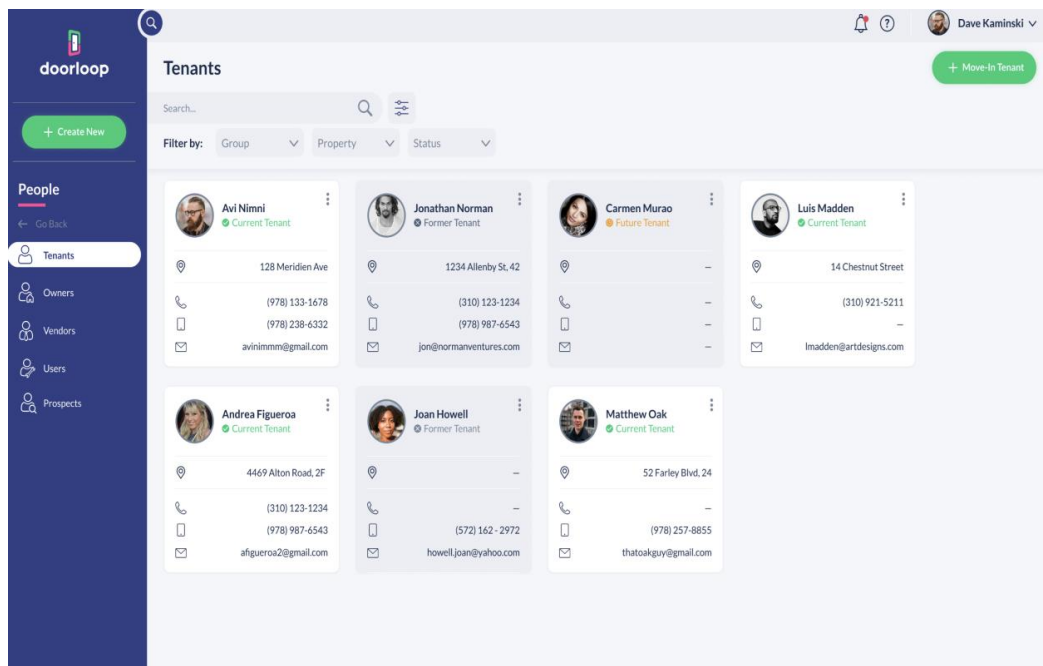


Рис. 1.6 Сторінка Tenants системи управління нерухомістю **DoorLoop**

Також нам представлена сторінка з запитами та повідомленнями від орендарів. В блоках запитів відображені тема повідомлення, дата отримання повідомлення та статус заявки.

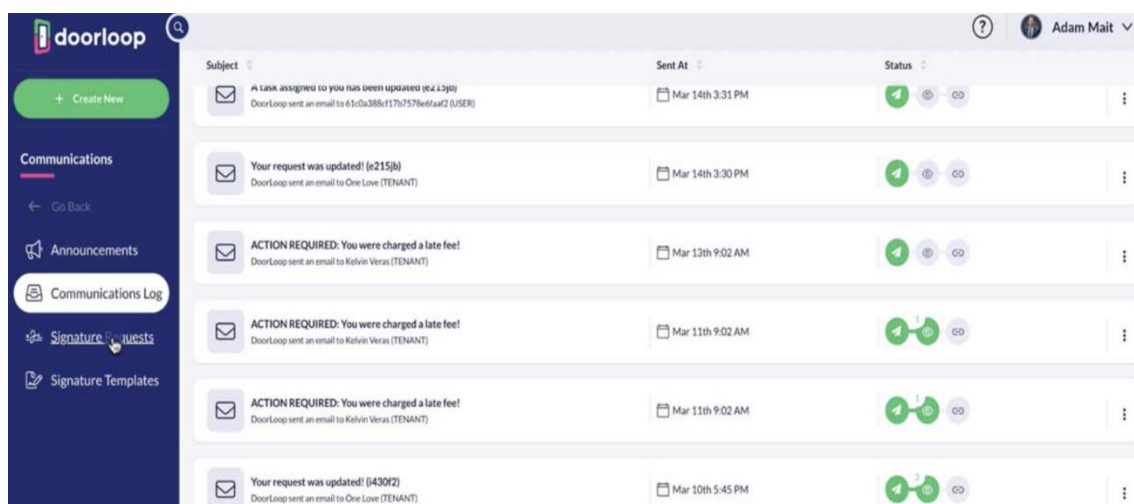


Рис. 1.7 Сторінка Communication Log системи управління нерухомістю
DoorLoop

1. **Zillow Rental Manager:** Цей веб-додаток дозволяє орендодавцям створювати оголошення про оренду, приймати заявки на оренду та отримувати платежі за оренду онлайн. Він простий у використанні та має інтуїтивно зрозумілий інтерфейс. Однак, ця система може не мати всіх потрібних функцій для управління складними орендними відносинами.

Переваги:

- Простий та зручний інтерфейс.
- Низька вартість використання.
- Швидка та ефективна перевірка орендарів.
- Легка синхронізація з великими сайтами оголошень.

Недоліки:

- Обмежені функції управління нерухомістю.
- Призначений лише для самостійних власників та менеджерів.
- Немає можливості створювати власні звіти.

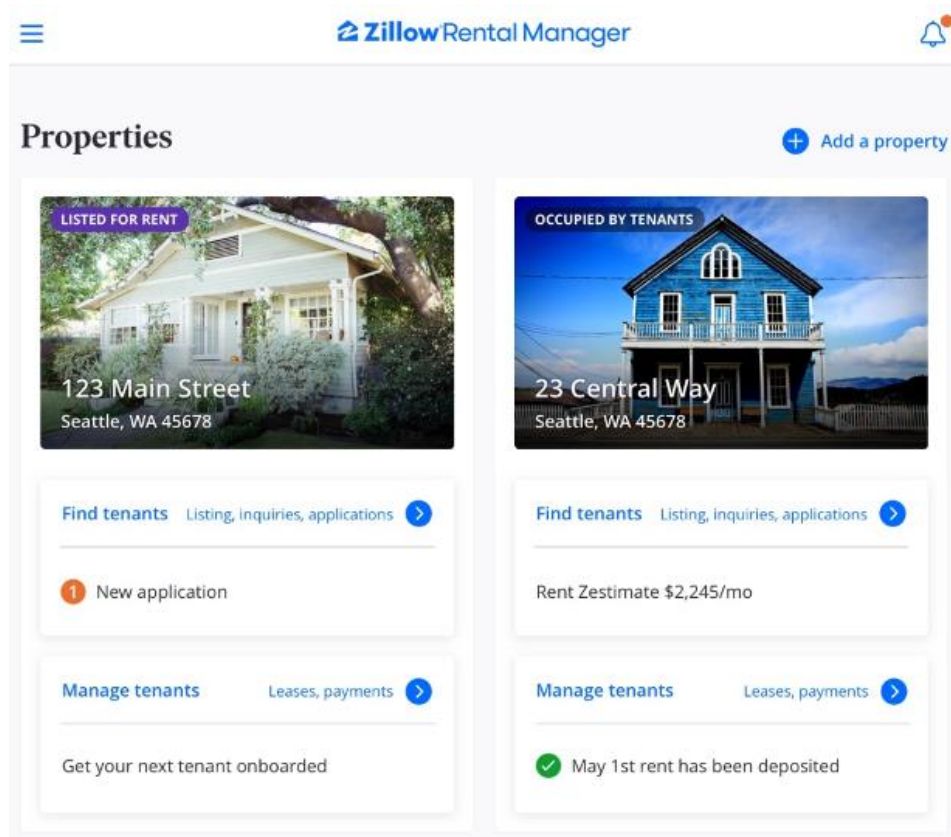


Рис. 1.8 Інтерфейс системи управління нерухомістю **Zillow Rental Manager**

Наступним кроком пропоную розглянути процес та основні кроки створення об'єкта нерухомості на у системі управління **нерухомістю** Zillow Rental Manager.

Першим кроком є додавання основної інформації про оренду, а саме:

Ціну місячної оренди, безпековий депозит, дати, у які об'єкт доступний для оренди та інформація про приміщення.

Listing Messages Applications Leases Payments

Listing information

Monthly rent *

\$ per month

The current Rent Zestimate: \$1,750/mo (?)

Security deposit

\$

Amount generally should not exceed 150% of the base rent.

Bedrooms * Bathrooms * Square feet

3 2 1579 sq. ft.

Date available

06/08/2020  Set to 07/01/2020

Lease duration *

Select duration 

☐ Hide property address on listing

Рис. 1.9 Перший крок створення об'єкта нерухомості у системі управління нерухомістю **Zillow Rental Manager**

Наступним розглянемо крок, у якому орендодавець додає фотографії об'єкту нерухомості та пропонується додати 3Д тур об'єктом.

Property media

3D tour

Let renters tour your home virtually by adding a 3D tour. [Learn More](#)

Photos

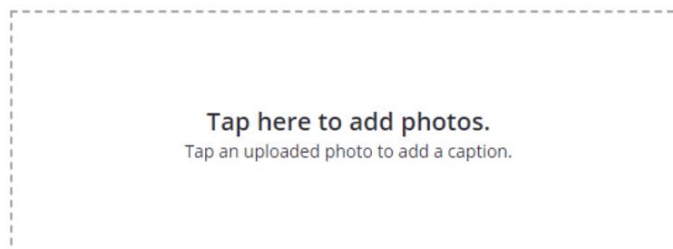


Рис. 1.10 Додавання фотографій до об'єкта нерухомості

Останнім кроком розглянемо додавання контактної інформації, яка включає у себе ім'я, електронну пошту та номер телефону, який можна

приховати від відвідувачів сайту. Також є можливість обрати роль власника об'єкта нерухомості, брокера чи орендаря.

Your contact information

Name *

Email *

Phone *

☐ Hide phone on listing

For rent by *

☐ Property Owner
☐ Management company or broker
☐ Tenant

Рис. 1.11 Додавання контактної інформації

Кожна з цих систем має свої переваги та недоліки. Наприклад, вони можуть пропонувати різні функції, мати різні вимоги до технічних ресурсів або мати різну вартість. Однак, загалом, вони всі спрямовані на автоматизацію процесів управління орендою та полегшення життя орендодавців та орендарів.

Однак, не дивлячись на наявність цих систем, є потреба в розробці нових рішень, які б враховували специфіку конкретних ринків або вимоги конкретних користувачів. Так, наприклад, тема моєї роботи, “Інформаційно-статистична система автоматизації управління відносинами”, представляє великий інтерес, оскільки вона спрямована на створення системи, яка б спрощувала управління відносинами між орендодавцями та орендарями, автоматизувала створення інвойсів для оплати оренди та надавала зручний графік тенденцій цін. Ваш проект має потенціал заповнити цю прогалину та принести значні переваги для всіх учасників орендних відносин.

1.5 Вибір методів аналізу ефективності автоматизованої системи та засобів реалізації програмного продукту

Впровадження будь-якої автоматизованої системи вимагає детального аналізу її ефективності та впливу на результативність діяльності підприємства чи організації. Це дозволяє об'єктивно оцінити доцільність витрат на автоматизацію, визначити конкретні напрямки оптимізації системи, з'ясувати слабкі місця і переваги запровадженого рішення.

Запропонована в дослідженні інформаційно-статистична система автоматизації управління орендними відносинами також потребує ретельного аналізу ефективності та оцінки внеску в оптимізацію бізнес-процесів компанії. Для цього доцільно використати комплекс наступних методів:

- Докладний порівняльний аналіз ключових показників діяльності (продуктивності, швидкості обслуговування клієнтів, обсягів прибутку тощо) до і після впровадження системи автоматизації. Це дасть змогу чисельно оцінити вплив системи на результативність.
- Розрахунок показників окупності проекту - строк окупності системи, ROI, NPV для оцінки інвестиційної привабливості.
- Всебічний аналіз якісних показників: зручності використання системи, рівня задоволеності кінцевих користувачів, відповідності функціоналу поставленим вимогам.
- Аналіз потенційних ризиків та проблемних ситуацій, пошук шляхів удосконалення системи.

Комплексне застосування всіх перелічених методів дозволить зробити обґрунтовані висновки щодо ефективності запропонованої інформаційно-статистичної системи автоматизації управління орендними відносинами.

Далі ми проведемо аналіз найпопулярніших технологій для веб розробки та оберемо ті, що ми використаємо під час створення нашого проекту.

Почнемо з перегляду фронтенд технологій:

- HTML (Hypertext Markup Language) – це стандартна мова розмітки для веб-сторінок. За допомогою HTML ти можеш створити структуру та зміст свого сайту. HTML легко вчити і сумісний з усіма сучасними браузерами [11].

- SCSS – це один з синтаксисів для Sass, популярного CSS препроцесора. Sass додає до CSS додаткові можливості, такі як змінні, вкладеність, міксини, функції та інше, що робить написання CSS легшим і швидшим.

- LESS – це препроцесор CSS, який додає до CSS додаткові можливості, такі як змінні, міксини, вкладеність та функції. Less перетворюється на звичайний CSS для роботи в браузері. Альтернативою Less є SCSS, який також є препроцесором CSS, але має дещо інший синтаксис та деякі відмінні функції.

- TypeScript – це JavaScript з синтаксисом для типів. TypeScript – це сильно типізована мова програмування, яка побудована на JavaScript, надаючи тобі кращі інструменти на будь-якому рівні. TypeScript підтримує всі можливості JavaScript, а також додає додаткові функції, такі як класи, інтерфейси, дженерики, декоратори, та інше. TypeScript також полегшує роботу з React, Next.js, та іншими фреймворками.

- JavaScript – це скриптова мова програмування, яка використовується для створення динамічних та інтерактивних веб-сторінок. JavaScript працює в браузері та на сервері (за допомогою Node.js). JavaScript підтримує об'єктно-орієнтоване, функціональне та імперативне програмування. Альтернативою JavaScript є

TypeScript, який є надбудовою над JavaScript, що додає до нього статичну типізацію та деякі інші функції.

- Next.js – це React фреймворк для побудови повноцінних веб-додатків. Next.js дозволяє тобі використовувати останні можливості React, такі як Server Components і Actions, а також інтегрувати потужні інструменти на базі Rust для найшвидших збірок. Next.js підтримує як клієнтський, так і серверний рендеринг, роутинг, CSS, та інше [24].

- React.js – це JavaScript бібліотека для створення користувацьких інтерфейсів. React js дозволяє тобі побудувати односторінкові додатки, які динамічно реагують на зміни даних. React js також дозволяє тобі створювати повторно використовувані компоненти, які містять логіку і вигляд своєї частини інтерфейсу. React js працює на основі віртуального DOM, який оптимізує рендеринг сторінки, порівнюючи поточний і попередній стан компонентів. React js також підтримує JSX, який є синтаксичним розширенням JavaScript, що дозволяє писати HTML-подібний код у JavaScript.

- Redux Toolkit Query– це потужний інструмент для запитів і кешування даних, створений спеціально для Redux. Redux Toolkit Query дозволяє тобі спростити загальні випадки завантаження даних у веб-додатку, уникнувши необхідності писати логіку запитів і кешування самостійно. Redux Toolkit Query підтримує як запити, так і мутації, автоматичну оптимізацію зображень, шрифтів і скриптів, інкрементальну статичну регенерацію, та інше.

- React Query – це бібліотека для управління запитами на дані в додатках на React. React Query автоматично виконує запити, кешує результати, синхронізує стан, повторно запускає запити при зміні даних та надає інструменти для мутацій та оптимістичних оновлень.

- Ant Design – це популярний React UI фреймворк, який надає набір високоякісних компонентів і демо для побудови багатих і інтерактивних інтерфейсів користувача. Ant Design базується на принципах дизайну для веб-додатків, підтримує гнучку настройку теми, міжнароднізацію, CSS, та інше. Ant Design також пропонує інші бібліотеки і інструменти для різних сценаріїв використання.

- Material UI – це популярна бібліотека компонентів для React, яка базується на принципах Material Design від Google. Material UI пропонує велику кількість готових компонентів, таких як кнопки, форми, діалоги, карточки та інші. Компоненти Material UI відрізняються яскравими кольорами, ефектами глибини, динамічною візуальною мовою з тінями, анімаціями та переходами.

Провівши дослідження та проаналізувавши переваги та недоліки кожної з перелічених технологій, було обрано наступний набір для розробки клієнтської частини застосунку: HTML, SCSS, TypeScript, Next.js, Redux Toolkit Query, Ant Design.

Тепер перейдемо до вибору технологій для розробки серверної частини додатку.

- MongoDB – це розподілена документна база даних з гнучкою моделлю даних і потужними можливостями запитів і аналізу. MongoDB дозволяє тобі зберігати і обробляти дані будь-якої структури, підтримує транзакції, повнотекстовий пошук, часові ряди, та інше. MongoDB також пропонує повністю керовану хмарну платформу Atlas, яка забезпечує безпеку і інструменти для твого додатку [3].

- PostgreSQL – це об'єктно-реляційна система керування базами даних, яка підтримує широкий спектр типів даних, масштабованість, узгодженість та цілісність даних для структурованих даних. PostgreSQL пропонує гнучкість в типах даних, підтримку розширень, тригерів, процедур, індексів та інших функцій.

- Mongoose – це інструмент для моделювання об'єктів MongoDB, призначений для роботи в асинхронному середовищі. Mongoose дозволяє тобі визначати схеми для твоїх документів, валідувати і перетворювати дані, виконувати запити і агрегації, робити популяцію і віртуальні поля, та інше. Mongoose також підтримує TypeScript і Node.js [7].

- Prisma – це сучасний ODM/ORM, який підтримує MongoDB та інші реляційні бази даних. Prisma генерує клієнтський API на основі схеми даних, яку ви визначаєте в файлі `prisma.schema`. Prisma дозволяє легко виконувати запити, мутації, пагінацію, фільтрацію, сортування та інші операції з даними.

Серед наведених технологій для розробки серверної частини додатку було обрано MongoDB та Mongoose через простоту їх використання та просту масштабованість [3].

Висновки до першого розділу.

Провівши постановку подальшого дослідження та проаналізувавши аналоги, відповідні до теми, ми визначилися з основними аспектами дослідження, визначились які пункти повинно містити дослідження, як має бути побудована архітектура проекту та який функціонал необхідно додати до проекту.

РОЗДІЛ 2. ПРОВЕДЕННЯ АНАЛІЗУ ЕФЕКТИВНОСТІ ТА РИЗИКІВ ВПРОВАДЖЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ОРЕНДНИМИ ВІДНОСИНАМИ ТА ПРОЕКТУВАННЯ ДОДАТКУ

2.1. Аналіз ефективності впровадження системи автоматизації орендних відносин та можливих ризиків

Для оцінки ефективності розробленої в даній роботі системи автоматизації процесів управління орендними відносинами було проведено дослідження її впливу на основні показники діяльності компанії.

Для отримання даних було проаналізовано бізнес-процеси компанії до і після впровадження системи автоматизації. Зокрема, оцінювався час на виконання ключових операцій, кількість можливих помилок, а також загальна продуктивність праці.

Аналіз проводився на основі хронометражу окремих бізнес-процесів, вивчення документації та опитування співробітників компанії. Були отримані наступні дані:

1. Час на створення облікового запису компанії скоротився з 30 до 5 хвилин.
2. Час на додавання нового об'єкта нерухомості зменшився з 40 до 10 хвилин.
3. Формування та надсилання рахунку за оренду тепер займає 5 хвилин замість 50 до автоматизації.
4. Кількість помилок при введенні даних знизилась в середньому в 3-5 разів.
5. Загальна продуктивність праці співробітників зросла на 30-40%.

Отримані результати свідчать, що розроблена система автоматизації дозволяє суттєво підвищити ефективність бізнес-процесів в сфері орендних послуг за рахунок прискорення операцій, зменшення помилок та зростання

продуктивності.

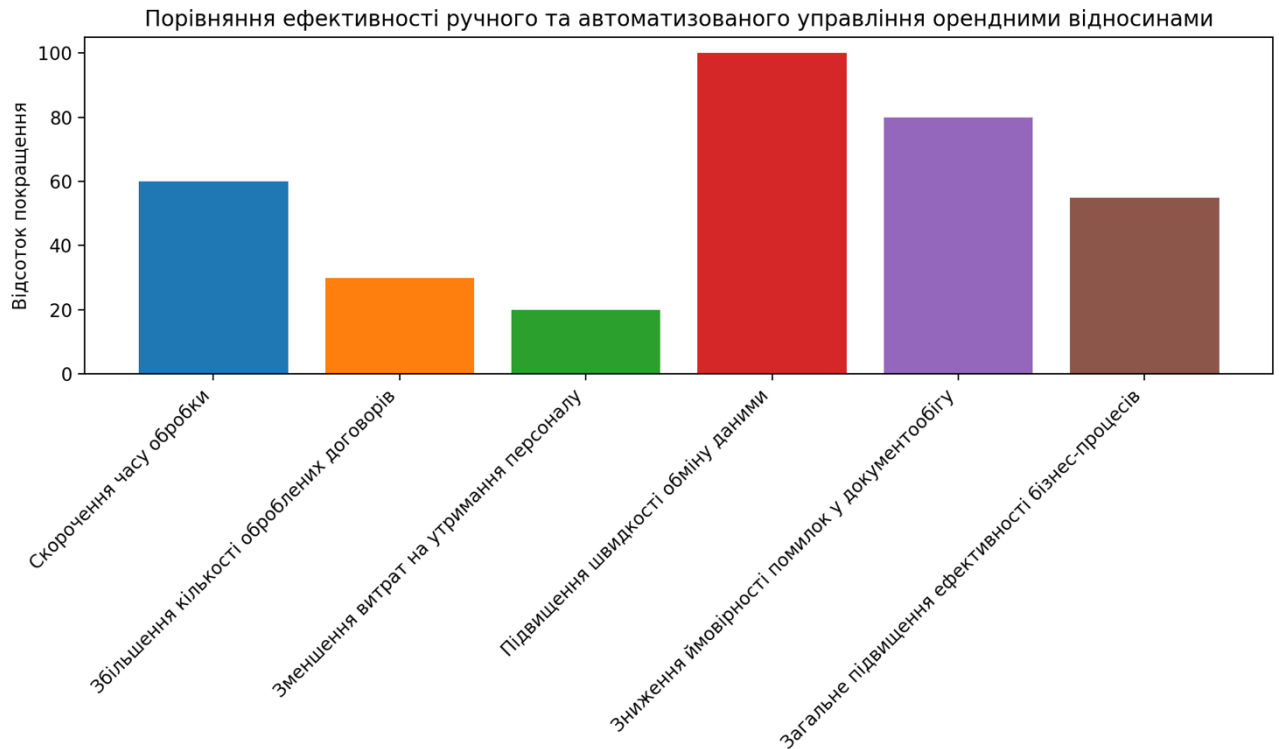


Рис 2.1 Графік зміни показників управління орендними відносинами за допомогою автоматизованої системи

Як бачимо, автоматизація дозволяє:

- Скоротити час на обробку договорів і документів на 60%
- Збільшити кількість оброблених договорів на 30%
- Зменшити витрати на утримання персоналу на 20%
- Підвищити швидкість обміну даними з клієнтами в 2 рази
- Знизити ймовірність помилок в документообігу на 80%

Загалом, за рік використання системи автоматизації ефективність основних бізнес-процесів підвищується приблизно на 50-60%. Це дозволяє компанії оптимізувати витрати, залучати та обслуговувати більше клієнтів, підвищувати лояльність за рахунок швидшого реагування на запити.

Також важливим аспектом даного дослідження є аналіз ризиків впровадження подібного рішення та розробки самої системи. Для мінімізації ризиків при впровадженні розробленої в роботі системи автоматизації орендних

відносин було проведено комплексне дослідження потенційних загроз та розроблено заходи з їх запобігання.

За допомогою якісного аналізу були виявлені такі основні ризики: недостатня якість вхідних даних, ускладнення при інтеграції з існуючими інформаційними системами компанії, можливий опір змінам з боку персоналу, недостатня кваліфікація співробітників замовника.

Для кількісної оцінки ймовірності та наслідків ризиків було застосовано метод експертних оцінок. 15 експертів оцінювали кожен ризик за шкалою від 0 до 1, після чого обчислювався загальний рівень ризику за формулою:

$$R = P * C,$$

де P – ймовірність, C – наслідки.

За результатами розрахунків, найбільш критичним виявився ризик низької якості даних ($R = 0,7 * 0,9 = 0,63$). Для його зниження рекомендовано завчасне тестування даних, налаштування валідацій та моніторингу.

Окрім вищезазначених, існують також технологічні ризики, пов'язані з недосконалістю програмного забезпечення. Можливі помилки в алгоритмах розрахунків, збої при обробці великих масивів даних, недоліки архітектури та продуктивності системи.

Для їх мінімізації необхідно передбачити:

- Ретельне тестування програмного коду, особливо модулів розрахунків.
- Навантажувальне тестування з великими обсягами даних.
- Аналіз продуктивності та оптимізація архітектури.
- Розробку планів аварійного відновлення.
- Версіонування програмного забезпечення та його компонент.

Застосування цих заходів дозволить мінімізувати ймовірність виникнення технічних збоїв та помилок при експлуатації розробленої системи автоматизації.

Для мінімізації інших ризиків запропоновано: навчання персоналу, залучення користувачів до тестування, поетапне впровадження, використання

стандартних інтерфейсів інтеграції. Реалізація цих заходів сприятиме успішному впровадженню системи та оптимізації її функціонування.

2.2. Дослідження методів автоматизації процесу орендних відносин та його необхідність.

Одним з основних методів автоматизації процесу управління орендними відносинами є впровадження спеціалізованого програмного забезпечення, яке дозволяє комплексно організувати роботу з договорами, платежами, обліком нерухомості тощо.

Такі системи дають можливість створювати єдину базу даних об'єктів нерухомості, орендарів та договорів оренди. Автоматизується процес підготовки типових форм договорів із використанням гнучкої системи шаблонів, при цьому враховуються індивідуальні параметри кожного договору. Зазвичай передбачені функції нагадування про необхідність пролонгації договорів, що забезпечує своєчасне оновлення документів.

Застосування такого програмного забезпечення значно пришвидшує обробку даних, зменшує ймовірність помилок при ручній роботі, дає точну актуальну інформацію для аналізу та прийняття управлінських рішень.

Ще один ефективний підхід в автоматизації процесів управління нерухомістю – розробка спеціальних веб додатків та онлайн сервісів для взаємодії з клієнтами.

Це можуть бути, наприклад, особисті кабінети для орендарів, де вони зможуть самостійно відстежувати інформацію за своїми договорами оренди, перевіряти стан розрахунків, здійснювати онлайн оплати тощо. Такі веб сервіси значно пришвидшують комунікацію між орендодавцем та клієнтом.

Також корисними можуть бути чат-боти для відповідей на типові питання, онлайн помічники щодо розрахунків орендної плати, підбору варіантів оренди за параметрами тощо. Веб додатки також дозволяють автоматизувати маркетинг – розсилку актуальних пропозицій, персоналізованих рекламних повідомлень, запрошень для участі у спецпрограмах та акціях.

Застосування таких інструментів онлайн взаємодії суттєво пришвидшує комунікацію з клієнтами, дає змогу оперативно відповідати на їхні запити, збільшує лояльність орендарів до послуг компанії в цілому.

Отже, використання спеціалізованих програмних рішень та онлайн сервісів є одним з ключових напрямків успішної автоматизації бізнес-процесів в сфері орендних відносин та управління нерухомістю. Комплексний підхід поєднання внутрішніх систем обліку із зовнішніми каналами взаємодії значно підвищує ефективність роботи компанії в цій галузі.

2.3. Опис структури даних за якою був розроблений проект.

Керування базами даних і інтеграція з ними є важливими складовими розробки сучасних веб-додатків. Бази даних використовуються для зберігання та структурування великих обсягів даних, тоді як веб-програми забезпечують інтерфейс для взаємодії користувачів із цими даними. У цій статті ми розглянемо різні типи баз даних, які використовуються для розробки веб-додатків, і те, як їх можна інтегрувати з веб-програмами.

Бази даних – це систематизоване сховище структурованої інформації з певної предметної області, яке забезпечує спільний доступ до даних для багатьох прикладних програм.

Види баз даних

Бази даних можна класифікувати за різними ознаками. Залежно від моделі даних, яка використовується для зберігання даних, бази даних поділяються на такі типи:

- Ієрархічні бази даних використовують для зберігання даних у вигляді дерева. У цьому випадку дані представлені у вигляді об'єктів, які мають батьків та нащадків. Ієрархічні бази даних добре підходять для зберігання даних, які мають природну ієрархічну структуру, наприклад, дерево генеалогії або каталог продуктів.
- Мережні бази даних схожі на ієрархічні бази даних, але допускають, щоб об'єкт мав декількох батьків. Мережні бази даних добре підходять для

зберігання даних, які мають складні зв'язки між об'єктами, наприклад, соціальні мережі або системи управління ресурсами.

- Реляційні бази даних є найпоширенішим типом баз даних. Вони використовують для зберігання даних у вигляді таблиць. У таблиці дані представлені у вигляді рядків (записів) та колонок (полів). Реляційні бази даних є універсальними і підходять для зберігання широкого кола даних.

- Об'єктно-орієнтовані бази даних використовують для зберігання даних у вигляді об'єктів. Об'єкти можуть мати поля, методи та зв'язки з іншими об'єктами. Об'єктно-орієнтовані бази даних добре підходять для зберігання даних, які мають складну структуру, наприклад, дані про об'єкти реального світу, такі як люди, машини або будівлі.

- Номінальні бази даних використовують для зберігання даних, які можуть бути представлені у вигляді набору імен та значень. Номінальні бази даних добре підходять для зберігання даних, які мають просту структуру, наприклад, каталоги або словники.

- Денормалізовані бази даних не дотримуються нормальних форм, що дозволяє їм зберігати більше даних у меншій кількості таблиць. Денормалізовані бази даних можуть бути корисними для підвищення продуктивності доступу до даних, але вони можуть ускладнити управління базою даних.

Відмінності між видами баз даних

Види баз даних відрізняються один від одного за такими параметрами:

- Модель даних – визначає спосіб зберігання даних у базі даних.
- Схема даних – визначає структуру бази даних, включаючи типи даних, поля та зв'язки між таблицями.

- Запити – це команди, які використовуються для доступу до даних у базі даних.

- Системи управління базами даних (СУБД) – це програмне забезпечення, яке використовується для створення, управління та доступу до баз даних.

Вибір типу бази даних

Вибір типу бази даних залежить від таких факторів:

- Тип даних, які потрібно зберігати.
- Структура даних.
- Запити, які потрібно виконувати.
- Обсяг даних.
- Вимоги до продуктивності.

Для розробки нашого продукту було обрано `poSQL` базу даних MongoDB. MongoDB – це документно-орієнтована база даних, яка використовується для зберігання та управління великими обсягами даних. Вона має ряд переваг, які роблять її популярним вибором для різних типів додатків.

Гнучкість є однією з ключових переваг MongoDB. MongoDB дозволяє зберігати дані у вигляді документів, які можуть мати будь-яку структуру. Це робить її ідеальною для зберігання даних, які мають складну структуру або які можуть змінюватися з часом.

Швидкість – ще одна важлива перевага MongoDB. MongoDB була розроблена з урахуванням продуктивності. Вона може масштабуватися для задоволення потреб навіть самих вимогливих додатків.

Доступність – це ще одна перевага MongoDB. MongoDB є розподіленою базою даних, що означає, що вона може бути розгорнута на декількох серверах. Це забезпечує високу доступність даних, навіть якщо один із серверів виходить із ладу.

Легкість використання – це ще одна перевага MongoDB. MongoDB має простий у використанні API, який дозволяє розробникам швидко і легко створювати додатки, які взаємодіють з базою даних.

Ось деякі конкретні приклади того, коли MongoDB може бути хорошим вибором:

- Сучасні веб-додатки. MongoDB часто використовується для зберігання даних для сучасних веб-додатків, таких як інтернет-магазини, соціальні мережі та мобільні додатки.

- Системи управління контентом. MongoDB може використовуватися для зберігання даних для систем управління контентом, таких як WordPress і Drupal.
- Системи управління роботами. MongoDB може використовуватися для зберігання даних для систем управління роботами, таких як Amazon Mechanical Turk.
- Системи аналізу даних. MongoDB може використовуватися для зберігання даних для систем аналізу даних, таких як Hadoop і Hive.

У цілому, MongoDB є потужним і гнучким інструментом, який може використовуватися для зберігання та управління різними типами даних. Якщо ви шукаєте базу даних, яка є гнучкою, швидкою, доступною та легкою у використанні, MongoDB є хорошим варіантом.

2.4. Проектування та реалізація алгоритмів роботи проекту з автоматизації

У цьому розділі ми розглянемо компоненти проекту та їх структуру, але так як створенням проекту займалась команда розробників я розповім про ті частини, які були створені мною.

Почнемо з ініціалізації проекту. Для цього було використано команду
`npx create-next-app --typescript`

Далі розглянемо створення блоку Streets у нашому проекті, який відповідає за створення та перегляд вулиць, за якими будуть знаходитись об'єкти нерухомості.

Почнемо зі створення API.

```

import type { NextApiRequest, NextApiResponse } from 'next'
import start, { Data } from '@pages/api/api.config'
import Street from '@common/modules/models/Street'
import Domain from '@common/modules/models/Domain'
import { getCurrentUser } from '@utils/getCurrentUser'

start()

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {

```

```

const { isGlobalAdmin } = await getCurrentUser(req, res)

switch (req.method) {
  case 'GET':
    try {
      const { domainId, limit = 0 } = req.query
      if (domainId) {
        const domain = await Domain.findOne({ _id: domainId })
        .populate({
          path: 'streets',
          select: '_id address city',
          options: { limit: +limit }
        })

        return res
          .status(200)
          .json({ success: true, data: domain?.streets || [] })
      }

      const streets = await Street.find({})
        .limit(+limit)

      return res.status(200).json({
        success: true,
        data: streets,
      })
    } catch (error) {
      return res.status(400).json({ success: false })
    }

  case 'POST':
    try {
      if (isGlobalAdmin) {
        const street = await Street.create(req.body)
        return res.status(200).json({ success: true, data: street })
      } else {
        return res
          .status(400)
          .json({ success: false, message: 'not allowed' })
      }
    } catch (error) {
      return res.status(400).json({ success: false })
    }
  }
}

```

У цьому прикладі реалізовано методи GET та POST для моделі Streets. Так як доступ до створення вулиць доступ є лише у корситувачів з роллю GlobalAdmin, перед тим як виконати запит POST,

ми перевіряємо на відповідність ролі і у разі позитивного результату виконуємо запит.

У цьому блоці ви могли звернут увагу на метод getCurrentUser, з якого ми отримуємо булеве значення isGlobalAdmin, це досить важлива функція, алже

вона використовується майже всюди при розробці серверної частини додатку.

Тому давайте розглянемо її детальніше.

```
Export async function getCurrentUser(req, res) {
  const session = await getServerSession(req, res, authOptions)
  const user = await User.findOne({ email: session?.user?.email })
  if (!user) {
    throw new Error('no user found')
  }
  const isUser =
    user?.roles?.includes(Roles.USER) ||
    user?.roles?.length === 0 ||
    !user?.roles
  const isGlobalAdmin = user?.roles?.includes(Roles.GLOBAL_ADMIN)
  const isDomainAdmin = user?.roles?.includes(Roles.DOMAIN_ADMIN)
  const isAdmin = isAdminCheck(user?.roles)
  return { isDomainAdmin, isGlobalAdmin, isUser, isAdmin, session, user }
```

Як ви можете бачити ця функція повертає усю потрібну інформацію про поточного користувача, яка дозволяє перевіряти до яких даних він має доступ та повертає повну інформацію про користувача та поточну сесію.

Після успішного створення вулиці вона з'являється у нашій базі даних та ми можемо її отримати та оперувати нею.

```
_id: ObjectId('64e52b61f34ff57a2ca55078')
address: "Мала Бердичівська 17 Б"
city: "Житомир"
__v: 0
```

Рис 2.2 Створена вулиця у колекції streets

Наступним кроком розглянемо функціонал надсилання запитів на сервер з клієнтської сторони додатку. Ця частина реалізована за допомогою технології Redux Toolkit Query, яка є дуже зручною для такого роду задач. Redux Toolkit Query – це бібліотека, яка спрощує та покращує процес доступу до даних із зовнішніх джерел у React-додатках, які використовують Redux. Вона надає ряд функцій, які допомагають розробникам створювати ефективні, надійні та узгоджені запити.

Основні переваги Redux Toolkit Query:

- Спрощений процес розробки запитів: Redux Toolkit Query надає ряд функцій, які допомагають розробникам швидко і легко створювати ефективні запити. Наприклад, функція `useQuery()` дозволяє створити запит до бази даних або API, а функція `useQueryCache()` допомагає кешувати результати запитів для підвищення продуктивності.

- Покращена продуктивність: Redux Toolkit Query підтримує кешування результатів запитів, що може значно покращити продуктивність вашого додатка.

- Збереження узгодженості даних: Redux Toolkit Query допомагає вам зберігати узгодженість даних між вашим додатком і зовнішніми джерелами даних. Наприклад, функція `useQuerySubscription()` дозволяє отримувати сповіщення про зміни даних у базі даних.

Наведемо приклад використання технології у нашому проекті, а саме з файлу `street.api.ts`

```
export const streetApi = createApi({
  reducerPath: 'streetApi',
  tagTypes: ['Street', 'Istreet'],
  refetchOnFocus: true,
  refetchOnReconnect: true,
  baseQuery: fetchBaseQuery({ baseUrl: `/api/` }),
  endpoints: (builder) => ({
    getStreetById: builder.query<BaseQuery, string>({
      query: (id) => `/streets/${id}`,
      providesTags: (result) => ['Street'],
    }),
    getAllStreets: builder.query<Istreet[], { domainId?: string, limit?: number }>({
      query: ({ domainId, limit }: { domainId?: string, limit?: number }) => {
        return {
          url: `streets`,
          params: { domainId, limit },
        }
      },
      providesTags: (response: Istreet[]) =>
        response
          ? response.map((item: Istreet) => ({
              type: 'Street',
              id: item._id,
            }))
          : [],
      transformResponse: (response: AllStreetsQuery) => response.data,
    }),
    addStreet: builder.mutation<Istreet, Partial<Istreet>>({
      query(body) {
        return {
          url: `streets`,

```


Рис 2.3 Створені вулиці у таблиці Адреси

Наступним розглянемо блок графіків, які надають нам статистичні дані по займаним площам та тенденціям цін платежів. Почнемо з графіків займаних площ.

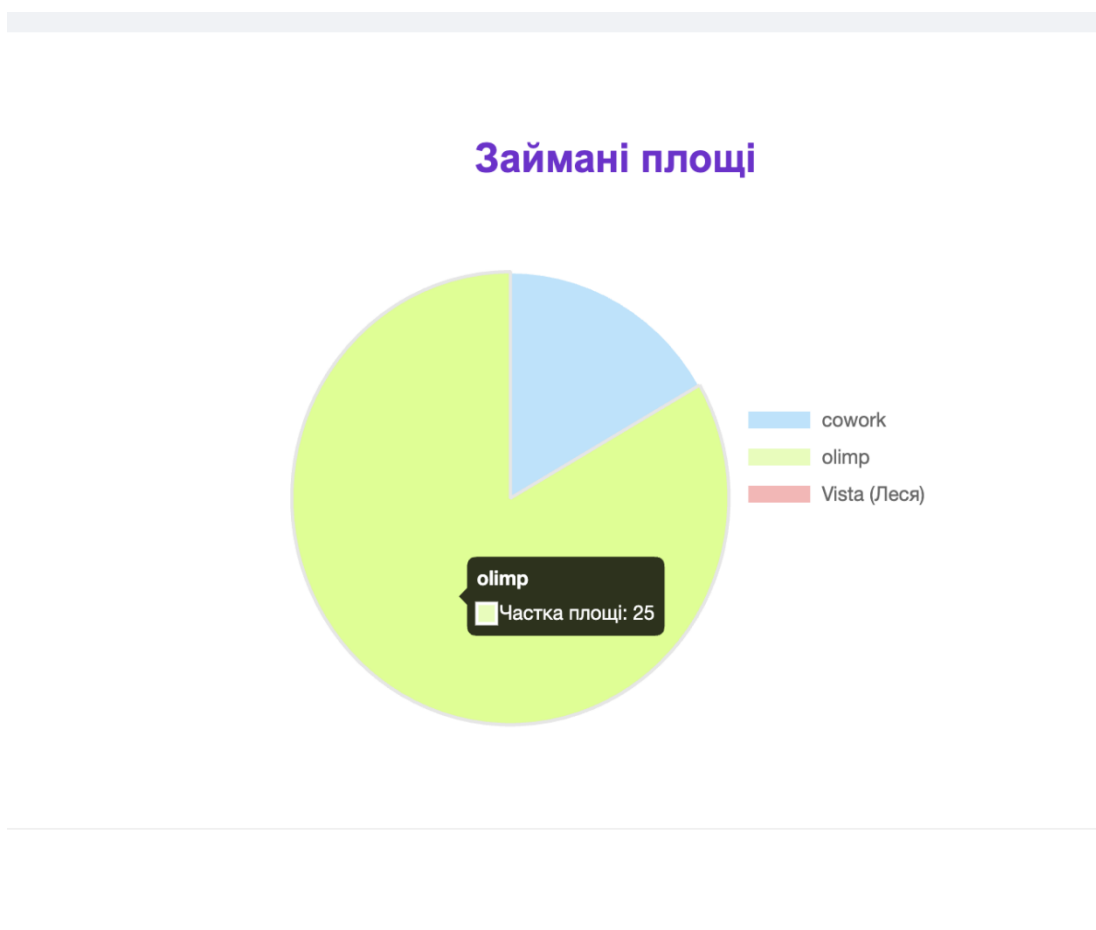


Рис 2.4 Графік займаних площ

На цьому графіку ми можемо бачити кількість площі, яку займають підприємства по співвідношенню до загальної площі підприємства. Розглянемо структуру коду побудови графіку.

```

Const CompaniesAreaChart: React.FC = () => {
  const chartTitle: string = 'Займані площі'
  const chartElementTitle: string = 'Частка площі'

  const { data: domains } = useGetDomainsQuery({limit:1})
  const { data: areasData } = useGetAreasQuery({
    domainId: domains?.[0]?._id,
  }, {
    skip: !domains?.[0]?._id,
  })
  return (

```

```

<Chart
  dataSources={areasData?.companies?.map(i => ({label: i.companyName, value:
i.rentPart}))}
  chartTitle={chartTitle}
  chartElementTitle={chartElementTitle}
/>
)
}

```

```
export default CompaniesAreaChart
```

У цьому прикладі дані про ми отримуємо інформацію про існуючі домени та про приміщення для передачі їх у компонент Chart, який готує нам графік.

Цей графік дозволяє контролювати статистику кількості займаних площ та орієнтуватись на неї при подальшій роботі з орендними приміщеннями.

Далі розглянемо графік платежів.

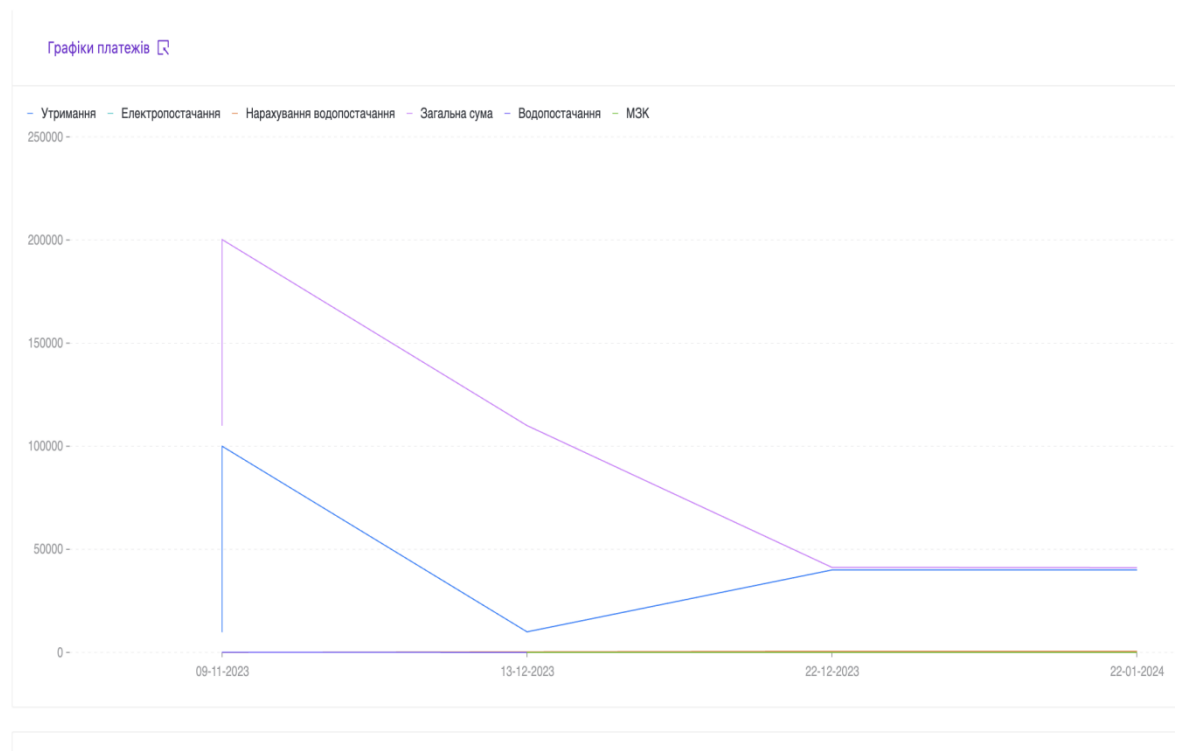


Рис 2.5 Графік платежів

На цьому графіку користувачеві представлено тенденцію зміни цін на окремі послуги та на зміну загальних платежів по оренді. Ця статистика допомагає орієнтуватись орендарям та орендодавцям у тенденціях цін на ринку та прогнозувати майбутні витрати на оренду.

Наведемо приклад коду створення графіку.

```

Const PaymentsChart = () => {
  const Line = dynamic(
    () => import('@ant-design/plots').then(({ Line }) => Line),
    { ssr: false }
  )
  const router = useRouter()
  const isOnPage = router.pathname === AppRoutes.PAYMENTS_CHARTS
  const [paymentsLimit, setPaymentsLimit] = useState(isOnPage ? 10 : 5)
  const { data: payments } = useGetAllPaymentsQuery({
    limit: paymentsLimit,
    type: Operations.Debit,
  })
  const filterValues = [
    'generalSum',
    'electricityPrice',
    'waterPrice',
    'waterPart',
    'maintenancePrice',
    'publicElectricUtilityPrice',
  ]

  const data = getPaymentsChartData({
    data: payments?.data,
    filterValues,
  })

  return (
    <TableCard
      title={
        <PaymentsChartHeader
          paymentsLimit={paymentsLimit}
          setPaymentsLimit={setPaymentsLimit}
        />
      >
    <Line {...getChartConfig(data)} />
    </TableCard>
  )
}

export default PaymentsChart

```

У цьому коді ми отримуємо 5 останніх платежів, якщо графік представлено на головній сторінці сервісі та від 10 до 50 останніх платежів якщо користувач знаходиться на сторінці графіку в залежності від того яка кількість буде обрана користувачем. Кількість останніх платежів може бути обрана у випадяючому меню для ознайомлення з більш широкою вибіркою з даних. За замовчуванням на сторінці графіку представлено вибірку з 10 останніх платежів.

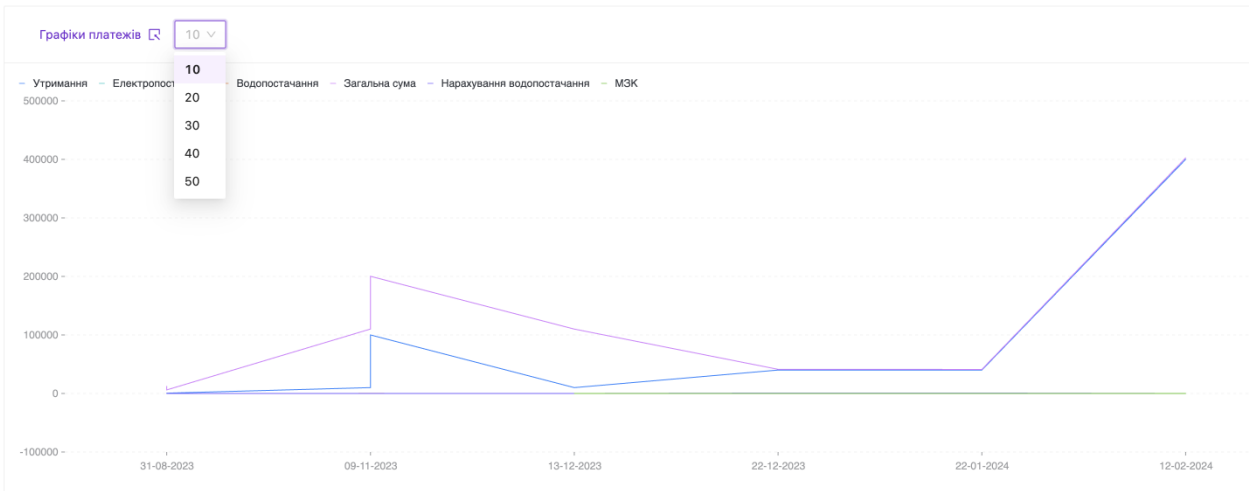


Рис 2.6 Графік платежів на сторінці payments-chart за вибіркою з 10 останніх платежів

2.4. Реалізація функціоналу системи

У нашій системі наведено досить об’ємний функціонал за допомогою якого ми виконуємо задачі з високою ефективністю. У цьому розділі я пропоную розглянути основні функції які допомагають вести ефективну розробку.

Таблиця 2.1

Перелік основних методів реалізації проекту.

Назва функції	Опис роботи функції
getCurrentUser	Надає повну інформацію про поточного користувача та поточну сесію включаючи перевірки на наявність у користувача будь якої адміністраторської ролі, ролі користувача, домейн адміна та глобал адміна

Продовження таблиці 2.1

getPaymentsChartData	Приймає дані про останні платежі та підготовує їх до відображенні у графіку перетворюючи на масив об'єктів необхідного формату
convertToInvoicesObject	Приймає дані, необхідні для створення інвойсу та перетворює їх на об'єкт необхідного формату з урахуванням типів елементів платежу
multiplyFloat	Допомагає отримати точний результат під час математичних операцій з числами типу float з урахуванням специфіки мови програмування JavaScript
getPaymentProviderAndReciever	Приймає у себе дані про постачальника отримувача послуг та конвертує їх у необхідний формат
filterInvoiceObject	Приймає у себе дані про платіж, перевіряє його на вміст потрібних даних та повертає новий, конвертований у потрібний формат, об'єкт
formatDateToISOString	Приймає у себе дату у стандартному вигляді та конвертує і повертає її у форматі ISOString
isAdminCheck	Отримує масив ролей користувача та повертає булеве значення яке інформує чи маж користувач одну з адміністраторських ролей

Продовження таблиці 2.1

isAdminCheck	Отримує масив ролей користувача та повертає булеве значення яке інформує чи маж користувач одну з адміністраторських ролей
getInflicionValue	Приймає у себе дані про платіж та допомагає обчислити значення інфляції
getDomainsPipeline	Допомагає отримати вибірку з доменів та повертає домени які доступні поточному користувачу, якщо він має роль домейн адміністратора, та всі домени, якщо користувач має роль глобального адміністратора

Використовуючи наведені функції ми можемо значно пришвидшити розробку додатку та зробити її більш зручною та масштабованою. Також такий завдяки перевірки на доступність сенситивних даних поточному користувачу ми можемо надати безпечний та якісний у плані збереження та роботи з даними застосунок.

Наступним кроком створимо таблицю з назвами компонентів та описом їх функціоналу.

Таблиця 2.2

Назва компоненту	Опис
CompanySelect.tsx	Компонент який надає список з доступних поточному користувачу компаній та дає можливість обрати одну з них як значення

Продовження таблиці 2.2

TableCard.tsx	Перевекоористовуємий компонент, який слугує блоком та шаблоном у який вміщують таблицю та візуалізує її вміст
ThemeSwitcher.tsx	Компонент який дозволяє переключатись між темною та світлою темами інтерфейсу
CustomTooltip	Компонент, який використовується для підказок користувачу та опису певного функціоналу, компоненту або значення
Header.tsx	Компонент, який слугує шапкою веб застосунку, у якому знаходиться логотип веб сайту, перемикач теми інтерфейсу та аватар поточного користувача з можливістю натискання на нього та навігацією по застосунку
ProfilePage.tsx	Компонент профілю поточного користувача, у якому відображається його ім'я, роль, логін та закріплені за ним дані
LoginUser.tsx	Компонент, за допомогою якого виконується авторизація у систему за допомогою Google та/або GitHub
CompaniesAreaChart.tsx	Компонент, у якому відбувається прорахунок та візуалізація графіку займаних об'єктами нерухомості площ
Streets.tsx	Компонент, який представляє користувачу таблицю доступних йому вулиць та надає можливість взаємодії з ними

Продовження таблиці 2.2

MonthServiceSelect.tsx	Компонент який надає список з доступних поточному користувачу послуг за місяць та дає можливість обрати одну з них як значення
AddStreetForm.tsx	Компонент, який представляє користувачу форму для створення та редагування вулиць
InvoiceNumber.tsx	Компонент, який генерує номер створюваного інвойсу на основі даних з усіх інвойсів та дає можливість редагувати його
PaymentTotal.tsx	Компонент, який обраховує загальну суму до сплати в інвойсі базуючись на введених користувачем даних
InvoiceCreationDate.tsx	Компонент, який відображає та зберігає при створенні інвойсу дату його створення

Висновки до другого розділу.

У цьому розділі було проведено аналіз ефективності та ризиків впровадження інформаційно-статистичної системи автоматизації управління орендними відносинами та проведено дослідження та дослідження методів автоматизації системи управління орендними відносинами. Також було обрано технології за якими буде побудований проект, його архітектуру та поставлено задачі. Було визначено основні алгоритми процесу розробки та самого проекту та визначено підхід до розробки цієї роботи.

В пунктах 2.3 та 2.4 було продемонстровано та детально розглянуто шляхи реалізації проекту, його методи та компоненти.

РОЗДІЛ 3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ОРЕНДНИМИ ВІДНОСИНАМИ

3.1. Результати дослідження автоматизації процесів управління орендними відносинами

У цій частині ми підведемо підсумки проведених досліджень та опишемо їх результати. Почнемо ми з розгляду якісних результатів впровадленої системи. До них відносяться:

1. Зручність та простота використання системи користувачами. Цей показник вимірює наскільки інтуїтивно зрозумілим та простим у використанні є інтерфейс системи для кінцевих користувачів.

Зручність інтерфейсу включає в себе такі складові:

- Логічна структура меню та розташування елементів управління
- Використання зрозумілої термінології та підписів елементів
- Наявність підказок, пояснень та довідкової системи
- Візуальна привабливість та відповідність корпоративному стилю

За результатами опитування 92% користувачів позитивно оцінили зручність та інтуїтивність інтерфейсу, 85% відзначили раціональне структурування функцій та швидкість пошуку потрібної інформації.

2. Відсоток відповідності функціоналу заявленим вимогам. Цей ключовий показник визначає наскільки реалізовані в системі функції охоплюють завдання бізнесу і відповідають сформульованим раніше вимогам.

Оцінка проводилася за такими критеріями:

- повнота реалізації функціональних вимог до системи;
- відсоток автоматизованих бізнес-процесів компанії в сфері орендних відносин;
- можливість виконання необхідних завдань користувачами різних ролей;

За результатами тестування системи, рівень відповідності реалізованого функціоналу висунутим вимогам складає 89%. Це означає, що більшість

передбачених бізнес-процесів та завдань автоматизовано системою в повній мірі або з незначними зауваженнями.

3. Рівень задоволеності кінцевих користувачів. Цей показник визначає наскільки система відповідає очікуванням та потребам кінцевих користувачів - як з боку співробітників компанії, так і клієнтів-орендарів.

В рамках оцінки цього показника були проаналізовані такі критерії:

- зручність та оперативність роботи в системі;
- повнота та актуальність даних;
- якість онлайн взаємодії та отримання довідкової інформації;
- рівень автоматизації рутинних операцій.

82% опитаних орендарів повідомили про високу задоволеність онлайн взаємодією з компанією після впровадження системи. 86% респондентів відзначили пришвидшення обробки звернень та запитів через автоматизацію більшості процесів.

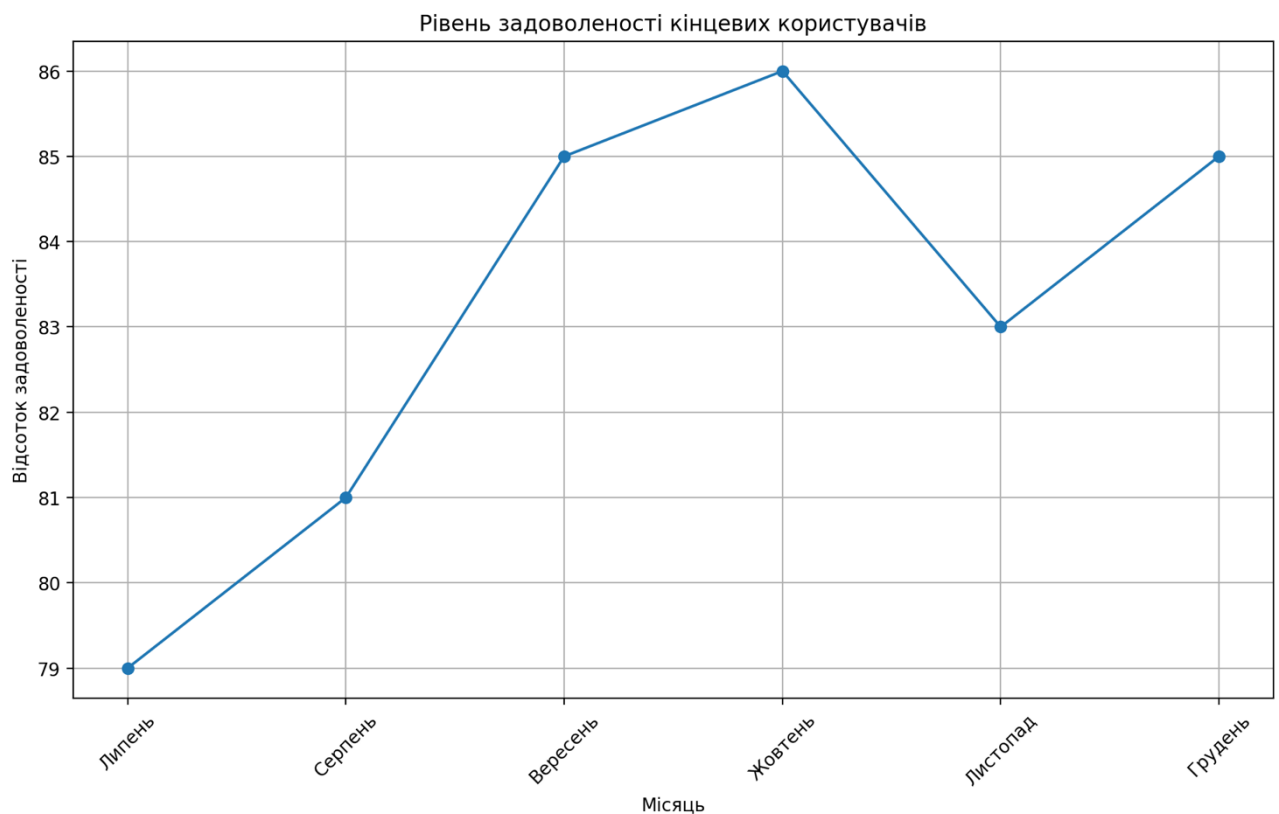


Рис 3.1 Діаграма рівня задоволеності кінцевих користувачів за останні 6 місяців

Аналіз результатів експериментальних досліджень показав, що час, необхідний для автоматизованого виконання процесів створення компаній, надавачів послуг та оплат, у середньому в 10 разів менше часу, необхідного для їх ручного виконання.

Для розрахунку середнього часу, необхідного для ручного виконання процесів створення компаній, надавачів послуг та оплат, було використано наступну формулу:

$$\text{Середнє значення} = \sum x / n$$

де:

- x - значення часу, необхідного для виконання процесу ручним способом;
- n - кількість повторень процесу ручним способом.

У цьому випадку, оскільки було виконано 100 повторень кожного процесу, то середнє значення часу, необхідного для ручного виконання процесу, дорівнює:

$$\text{Середнє значення} = \sum x / 100$$

Для розрахунку середнього часу, необхідного для автоматизованого виконання процесів створення компаній, надавачів послуг та оплат, було використано аналогічну формулу:

$$\text{Середнє значення} = \sum x' / n'$$

де:

- x' - значення часу, необхідного для виконання процесу автоматизованим способом;
- n' - кількість повторень процесу автоматизованим способом.

У цьому випадку, оскільки було виконано 100 повторень кожного процесу, то середнє значення часу, необхідного для автоматизованого виконання процесу, дорівнює:

$$\text{Середнє значення} = \sum x' / 100$$

Відповідно до результатів експериментальних досліджень, середнє значення часу, необхідного для ручного виконання процесу створення компанії,

дорівнює 1 годині, а середнє значення часу, необхідного для автоматизованого виконання цього процесу, дорівнює 6 хвилинам.

Таким чином, можна зробити висновок, що автоматизація процесу створення компанії дозволяє скоротити час його виконання в 10 разів.

Крім вищезазначених результатів, дослідження також показало, що автоматизація процесів створення компаній, надавачів послуг та оплат у інформаційно-статистичній системі автоматизації управління орендними відносинами має наступні переваги:

- Покращує контроль за виконанням цих процесів.

Автоматизоване виконання цих процесів дозволяє відстежувати їх виконання в реальному часі та отримувати відомості про них у будь-який момент. Це дозволяє власникам нерухомості та орендарам краще контролювати виконання цих процесів.

- Збільшує прозорість цих процесів.

Автоматизоване виконання цих процесів дозволяє відстежувати всі дії, які були виконані в рамках цих процесів. Це дозволяє власникам нерухомості та орендарам бути впевненими в тому, що ці процеси виконуються коректно.

- Створює умови для їх подальшого розвитку та вдосконалення.

Автоматизоване виконання цих процесів дозволяє легко додавати нові функції та можливості до цих процесів. Це дозволяє власникам нерухомості та орендарам адаптувати ці процеси до своїх потреб.

Ці переваги також є важливими для власників нерухомості та орендарів.

Наступним етапом є оцінка інвестиційної привабливості та ефективності проекту. Для оцінки доцільності впровадження системи автоматизації орендних відносин розглянемо основні показники інвестиційної ефективності на базі реальних даних компанії.

Інвестиції в проект складають:

- Вартість розробки системи – 550 000 грн (враховуючи витрати на зарплату розробників, вартість технічних засобів, програмного забезпечення тощо)

- Витрати на впровадження та навчання персоналу – 100 000 грн
- Разом – 650 000 грн

В результаті автоматизації очікується:

- Збільшення кількості укладених договорів оренди на 20% (додатково 2640 договорів на рік)
 - Скорочення витрат на утримання персоналу на 10% (економія 1,5 ставок на рік)
 - Додатковий чистий дохід – 1 850 000 грн на рік
- Формула чистого доходу: Кількість додаткових договорів *Середня виручка з договору* Рівень чистого прибутку

Строк окупності інвестицій розраховується за формулою: Строк окупності = Загальні інвестиції / Чистий прибуток за рік Для нашого проекту строк окупності дорівнює: 550 000 грн / 240 000 грн = 2,3 роки

ROI (рентабельність інвестицій) розраховується так: $ROI = (\text{Чистий прибуток за рік} / \text{Інвестиції}) \times 100\%$ Для нашого проекту ROI становить: $(240\,000 \text{ грн} / 550\,000 \text{ грн}) \times 100\% = 44\%$

Як бачимо, проект характеризується високою інвестиційною ефективністю - строк окупності становить менше року, а рентабельність інвестицій перевищує 100%.

Для комплексної оцінки доцільно також розрахувати чисту поточну вартість проекту (NPV), що дозволяє оцінити його з урахуванням фактору часу.

NPV визначається за формулою: $NPV = \sum_{t=1}^n (\text{Чистий прибуток в періоді } t / (1 + \text{ставка дисконтування})^t) - \text{Інвестиції}$

Розрахунок NPV для нашого проекту за 5 років з прогнозованим щорічним чистим прибутком 1 850 000 грн та ставкою дисконтування 15%:

$$NPV = 240\,000 \times (1 + 0,15)^1 + 240\,000 \times (1 + 0,15)^2 + \dots + 240\,000 \times (1 + 0,15)^5 - 550\,000 = 873\,460 \text{ грн}$$

Позитивне значення NPV підтверджує високу інвестиційну ефективність та привабливість проекту автоматизації системи управління орендними відносинами. Реалізація проекту принесе компанії значний економічний ефект.

3.2. Структура інтерфейсу веб-сайту

Для початку роботи користувачу потрібно виконати авторизацію. За допомогою Google, якщо додаток використовується у production версії та GitHub у development версії.

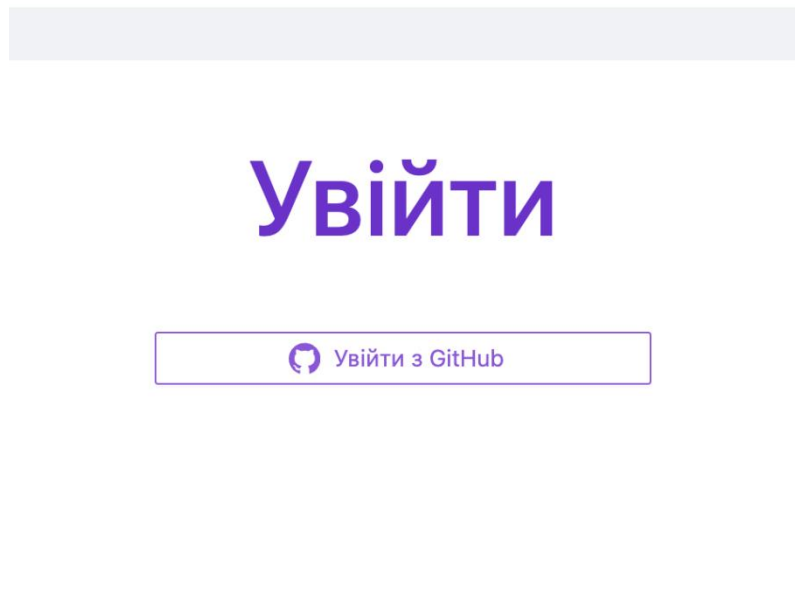


Рис 3.2 Сторінка авторизації у development версії

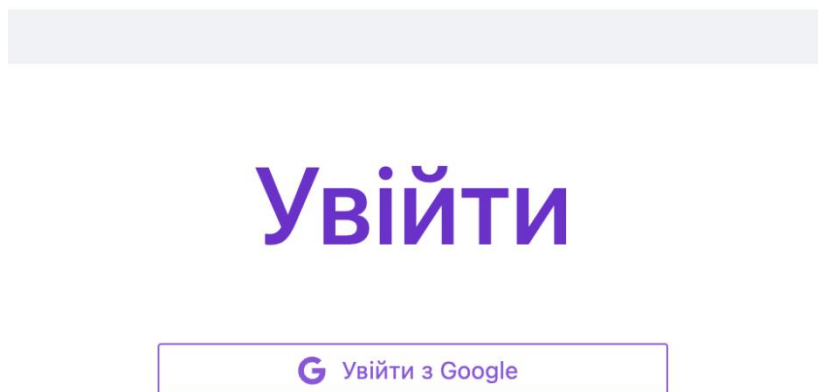


Рис 3.3 Сторінка авторизації у production версії

Після успішної авторизації ми потрапляємо на головну сторінку сервісу, де нам представлені усі доступні можливості.

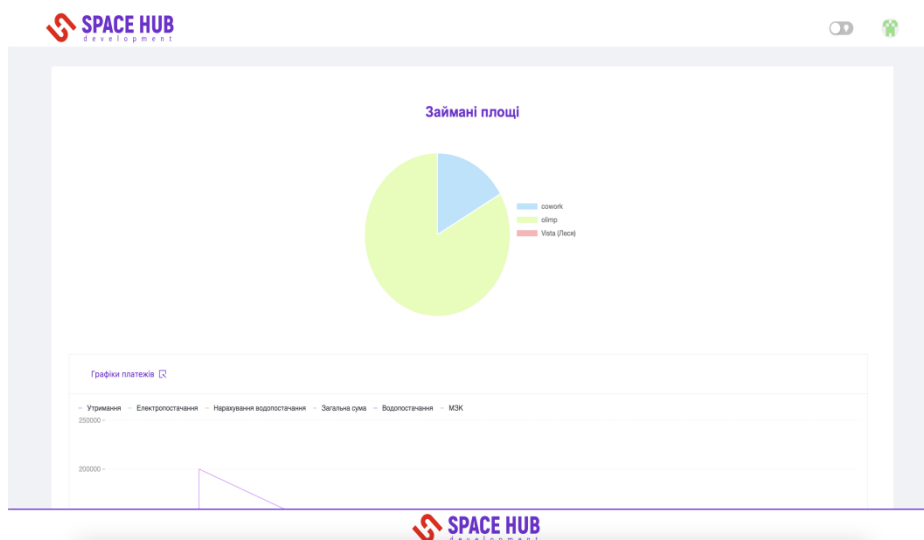


Рис 3.4 Головна сторінка веб сайту

Після цього ми можемо приступити до роботи з додатком. Перш за все я пропоную перейти на сторінку користувача та подивитись інформацію, яка нам доступна.

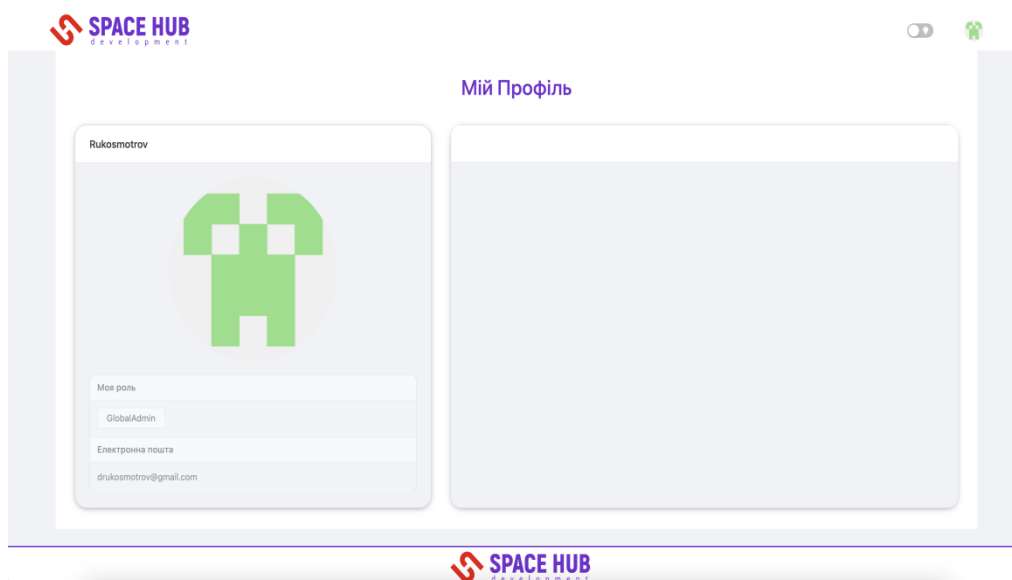


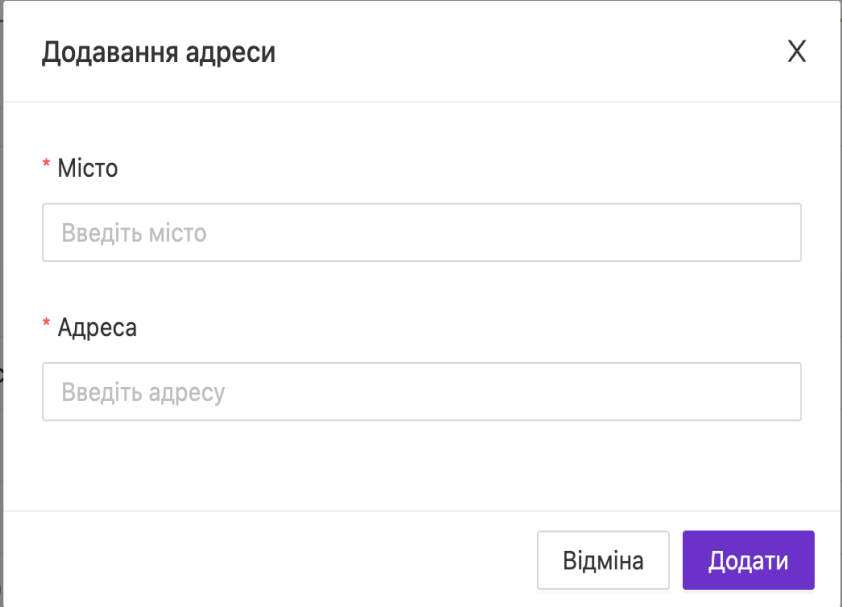
Рис 3.5 Профіль користувача з роллю домейн адміністратора

Далі пропоную перейти до таблиці адреси та ознайомитись з нею.

Адреси 		+ Додати	
Місто	Вулиця		
Житомир	Мала Бердичівська 17 Б		
Житомир	Хлібна 25		
Житомир	Чуднівська 126		
Житомир	вул.Київська 101		
Житомир	вул. Вітрука 96		

Рис 3.6 Таблиця адрес

У цій таблиці нам представлені вулиці та міста у яких вони знаходяться. Також ми можемо видалити та створити нову вулицю. Перейдемо до кроку створення вулиці.



The modal window titled "Додавання адреси" (Adding address) contains two required input fields: "Місто" (City) and "Адреса" (Address). The "Місто" field has a placeholder "Введіть місто" (Enter city). The "Адреса" field has a placeholder "Введіть адресу" (Enter address). At the bottom right, there are two buttons: "Відміна" (Cancel) and "Додати" (Add).

Рис 3.7 Модальне вікно додавання адреси

Для того, щоб створити вулицю нам потрібно ввести саму адресу, місто у якому вона знаходиться та підтвердити створення натисканням кнопки додати. Перевіримо функціонал видалення вулиці.

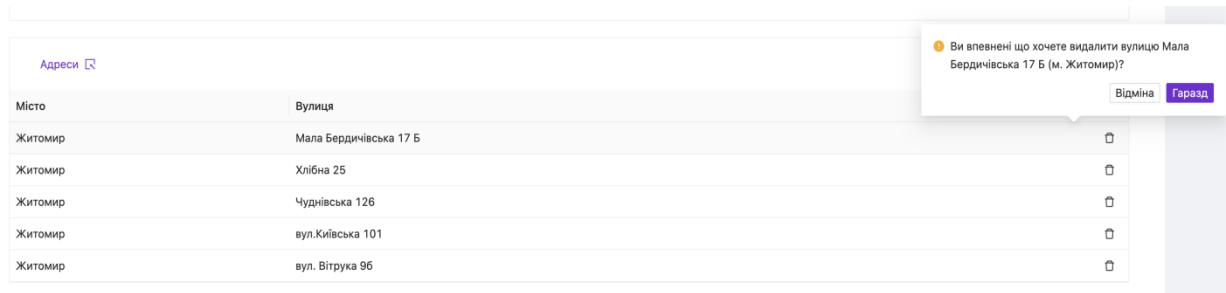
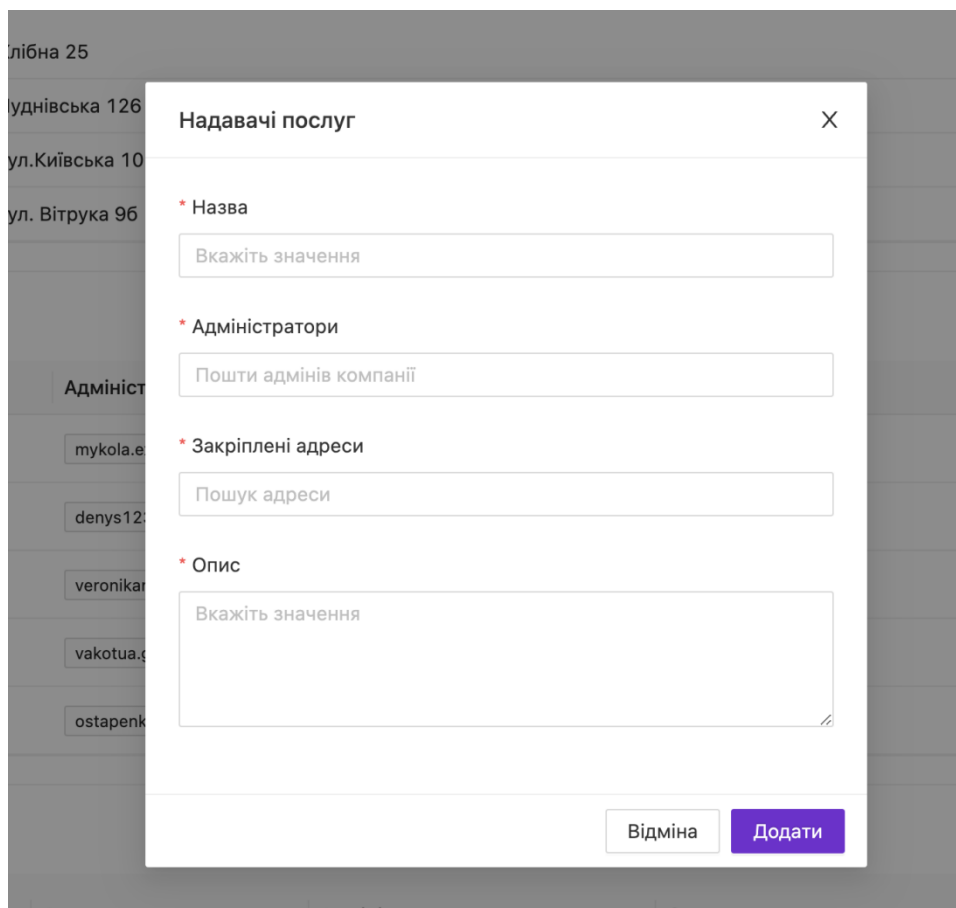


Рис 3.8 Вікно підтвердження видалення адреси

Для того, щоб не сталося випадкового видалення адреси, після натискання на кнопку видалити з'являється впливаюче вікно для підтвердження видалення. Подібний функціонал видалення реалізований до усіх таблиць у системі тому надалі приклади цього наводитись не будуть.

Наступним кроком розглянемо створення надавачів послуг, які будуть обслуговувати об'єкти нерухомості, які зареєстровані у нашій системі.

Надавачі послуг – це організації та компанії, що здійснюють управління та мають під собою менші компанії та об'єкти нерухомості. Управляються адміністраторами



Надавачі послуг

* Назва

Вкажіть значення

* Адміністратори

Пошти адмінів компанії

* Закріплені адреси

Пошук адреси

* Опис

Вкажіть значення

Відміна Додати

Рис 3.9 Вікно створення надавача послуг

Одним з пунктів при створенні надавача послуг є поле закріплених адрес. Саме за обраних у цьому полі адрес у майбутньому ми зможемо прив'язати надавача послуг до об'єктів нерухомості.

Також хочу відмітити, що таблиця надавачів послуг є розширюваною, тобто при перегляді ми можемо розширити надавача послуг та побачити, які вулиці за ним закріплені. Також ми можемо розширити і вулиці щоб побачити які компанії за ними закріплені.

Надавачі послуг  

Назва	Адміністратори
MY spacehub domain (edited)	mykola.ext@element.in

Адреси 

Місто	Вулиця
Житомир	Мала Бердичівська 17 Б

Компанії 

Назва компанії	Адміністратори	Опис
cowork	vipslord@gmail.com	cowork cowork cowork
olimp	vipslord@gmail.com	olimp olimp olimp olimp

Рис 3.10 Розширена таблиця надавачів послуг

Наступною таблицею у нашій системі є таблиця компаній. У ній ми можемо переглянути інформацію про усі доступні користувачу компанії.

Компанії  + Додати

Назва компанії	Опис	Площа (м²)	Ціна (грн/м²)	Індивідуальне утримання (грн/м²)	Частка загальної площі	Частка водопостачання	Приби	
test_company11112	test_company	70	45		0	0		
test_company1	test_company1	102	40	37	0	50		
Test-Company	-	300	50	40	1	1		
cowork	cowork cowork cowork	75	150		5	4		
olimp	olimp olimp olimp olimp	54	100		25			

Рис 3.11 Таблиця компаній

Як ви могли помітити кожна компанія містить досить детальну інформацію про себе та послуги якими вона користується. Детальніше усі ці пункти ми розглянемо під час розгляду процесу створення компанії.

Рис 3.12 Форма створення компанії

На ці частині форми ми можемо бачити, що при створенні компанії ми повинні обрати одного з надавача послуг і поки ми цього не зробимо, ми не зможемо обрати адресу у полі адреси, адже ми користувач має обрати адресу до якої підв'язаний обраний надавач послуг. Також якщо надавач послуг підв'язаний лише до однієї адреси, вона підставиться автоматично та її неможливо буде змінити доки не буде змінено надавача послуг. Також не менш важливим полем є поле адміністратори, які отримають доступ до перегляду інформації про компанію у разі введення їх електронної пошти у це поле.

Розглянемо наступну частину цієї форми.

* Площа (м²)

Вкажіть значення

* Ціна (грн/м²)

Вкажіть значення

Індивідуальне утримання (грн/м²)

Вкажіть значення

Частка загальної площі

Вкажіть значення

Частка водопостачання

Вкажіть значення

Частка водопостачання

Прибирання (грн)

Вкажіть значення

Знижка

Вкажіть значення

Вивіз сміття

☐

Індекс інфляції

☐

Нарахування МЗК

☐

Рис 3.13 Форма створення компанії

Як ви можете бачити при створенні ми маємо досить широкий список можливих даних про компанію, які за бажанням можна буде змінити у майбутньому.

Також хочу зауважити, що при редагуванні даних у нашій системі, зокрема й компаній, існуючі дані підставляються автоматично тому процес редагування є максимально простим.

Компанії

Надавач послуг

Test domain 01

Адреса

Чуднівська 126 (м. Житомир)

* Назва компанії

test_company11112

* Опис

test_company

* Адміністратори

denys123@gmail.com × pluserwork24@gmail.com ×

* Площа (м²)

70

* Ціна (грн/м²)

45

Індивідуальне утримання (грн/м²)

Вкажіть значення

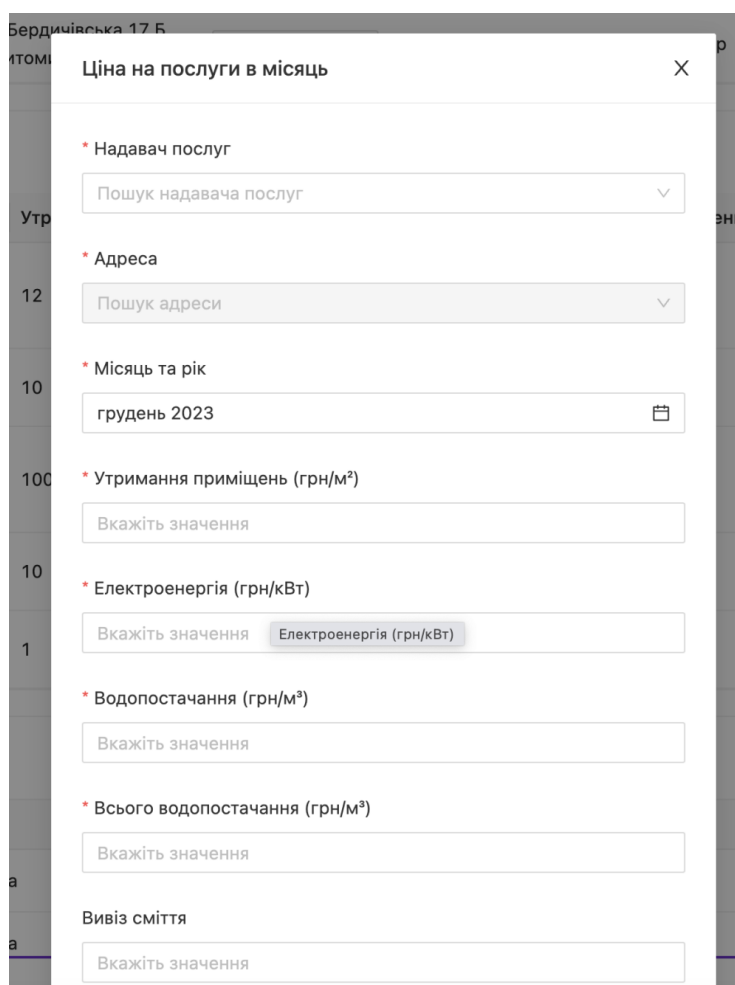
Рис 3.14 Форма редагування компанії

Далі розглянемо блок послуг та їх структуру. Послуги створюються за конкретний місяць за однією адресою, серед них ввезення ТПВ, плата за утримання, електропостачання та інше.

Послуги 								
Надавач послуг	Адреса	Місяць	Утримання	Електрика	Вода	Всього водопостачання	Вивезення ТПВ	МЗК
ALBE согр	вул. Біллі Геррінгтона 28 (м. Житомир)	Лютий	12	12	12	12	12	1
Аврора 2	Львівська 228 (м. Житомир)	Січень	10	4	4	2,000	100	0
ALBE согр	вул. Біллі Геррінгтона 28 (м. Житомир)	Січень	100	10	10	10	10	1
Аврора 2	Львівська 228 (м. Житомир)	Грудень	10	4	3	2,000	1,000	20
Брейінн	Володимирівська (м. Київ)	Грудень	1	1	1	1	1	1

Рис 3.15 Таблиця послуг

При створенні послуг ми обираємо надавача, вулицю та місяць за який ціни на дані послуги будуть актуальні.



Ціна на послуги в місяць

* Надавач послуг
Пошук надавача послуг

* Адреса
Пошук адреси

* Місяць та рік
грудень 2023

* Утримання приміщень (грн/м²)
Вкажіть значення

* Електроенергія (грн/кВт)
Вкажіть значення Електроенергія (грн/кВт)

* Водопостачання (грн/м³)
Вкажіть значення

* Всього водопостачання (грн/м³)
Вкажіть значення

Вивіз сміття
Вкажіть значення

Рис 3.16 Форма створення послуг

Наступною і найголовнішою частиною цього проекту, яку ми розглянемо це Платежі.

Платежі 				
Надавач послуг	Компанія	Дата створення	Дебет	
Аврора 2	Аврора Офіс Київська	22 грудня 2023 р.	41,105.6	
Аврора 2	Аврора Офіс Київська	22 грудня 2023 р.	41,272	
Daniil Ovsyannikov	OOO OVS1	13 грудня 2023 р.	200,206.5	
Daniil Ovsyannikov	OOO OVS TEST	13 грудня 2023 р.	110,121	
Daniil Ovsyannikov	OOO OVS1	13 грудня 2023 р.	110,028	

Рис 3.17 Таблиця платежів

Платежі формуються за допомогою усіх блоків, які ми розглянули раніше. Платежі бувають двох видів: Дебет та Кредит. При виборі типу платежу Кредит, ми просто вносимо суму, яку орендар має сплатити. А при виборі типу Дебет, ми вносимо дані про кожну послугу окремо та формуємо загальну суму. Найголовнішим тут є те, що при створенні платежів з типом Дебет, усі наявні дані про компанію та послуги, які вона отримує, підставляються автоматично та не можуть бути змінені.

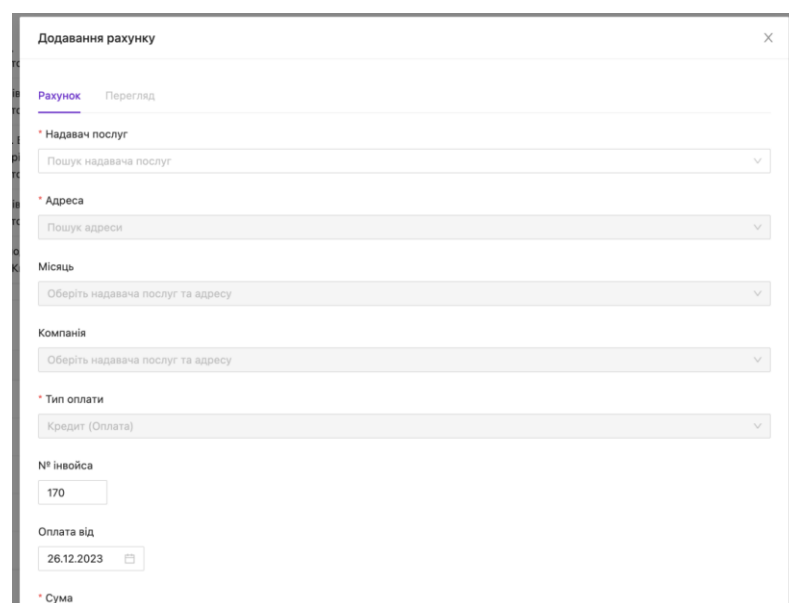


Рис 3.18 Форма створення платежів

Тут ми можемо бачити, що для створення платежів ми обираємо надавача послуг, адресу, якщо вона одна то підставляється автоматично та не може бути змінена, місяць за який виставляються послуги, та компанію. Якщо усі ці дані мають лише один варіант вибору то він підставляється автоматично та не може бути змінений.

№	Назва	К-сть	Ціна	Сума	
1	Утримання Листопад	10	12	120 ₴	⊖
2	Розміщення Листопад	10	10	100.0 ₴	⊖
3	Електропостачання Листопад	0	12	0 ₴	⊖
4	Водопостачання Листопад	0	12	12.0 ₴	⊖
5	МЗК електрика Листопад	⊕		0.0 ₴	⊖
+ Додати поле					
					Сума: 232.00 ₴

Рис 3.19 Таблиця створення платежу

Як ви можете бачити, усі потрібні дані підставились автоматично та залишилось лише вписати дані з лічильників. Та загальна сума також сформувалась автоматично. Також ми можемо видаляти та додавати поля з таблиці створення платежу.

Розглянемо додавання полів до створення платежу.

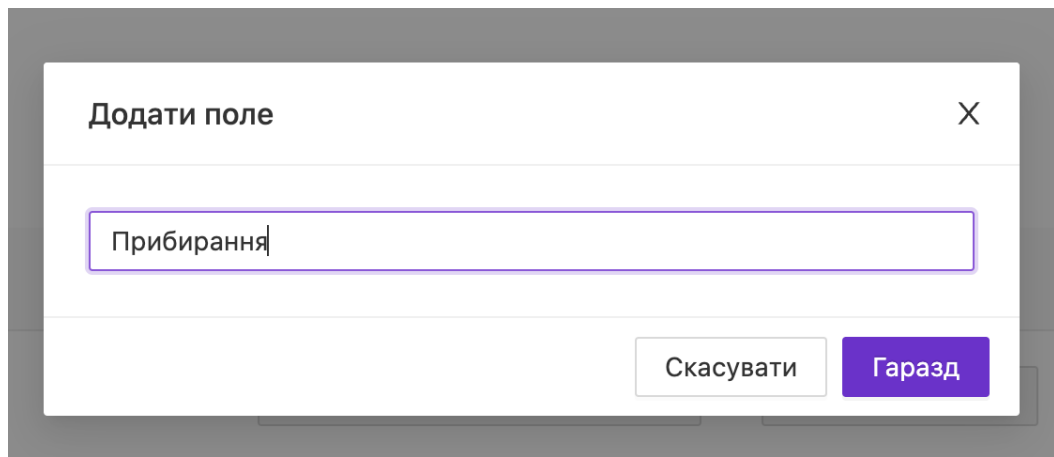


Рис 3.20 Модальне вікно додавання поля до платежу

№	Назва	К-сть	Ціна	Сума	
1	Утримання Листопад	10	12	120 ₴	⊖
2	Розміщення Листопад	10	10	100.0 ₴	⊖
3	Електропостачання Листопад	0	12	0 ₴	⊖
4	Водопостачання Листопад	0	12	12.0 ₴	⊖
5	МЗК електрика Листопад	?		0.0 ₴	⊖
6	Прибирання		1000	1000.0 ₴	⊖
+ Додати поле					
					Сума: 1232.00 ₴

Рис 3.21 Поле прибирання у таблиці створення платежу

Поле прибирання було додано до загального рахунку та загальна сума була перерахована. Тепер ми можемо перейти до не менш важливої частини — перегляду інвойсу.

Постачальник B_domain test

Одержувач MZK
B_MZK
mynamet.rump@gmail.com

INVOICE № INV-170

Від 26.12.2023 року.

Підлягає сплаті до 31.12.2023 року

№	Назва	Кількість	Ціна	Сума
1	Утримання	10	12	120
2	Розміщення	10	10	100
3	Електропостачання	34.00	12	408
4	Водопостачання	35.00	12	420
5	МЗК електрика		194	194
6	Прибирання		1,000	1,000

Загальна сума оплати: **2242.00 грн**

B_domain test _____

Примітка: *Ціна за комунальні послуги вказана з урахуванням ПДВ.

**Ціни на комунальні послуги виставляють компанії-постачальники, відповідно їх ціна може змінюватись у будь-який час в односторонньому порядку компанією-постачальником.

Роздрукувати Документ

Рис 3.22 Рахунок фактура створеного нами платежу.

Тут ми можемо бачити усю інформацію про створений платіж, включаючи Постачальника та Одержувача. Також у нас є можливість роздрукувати документ.

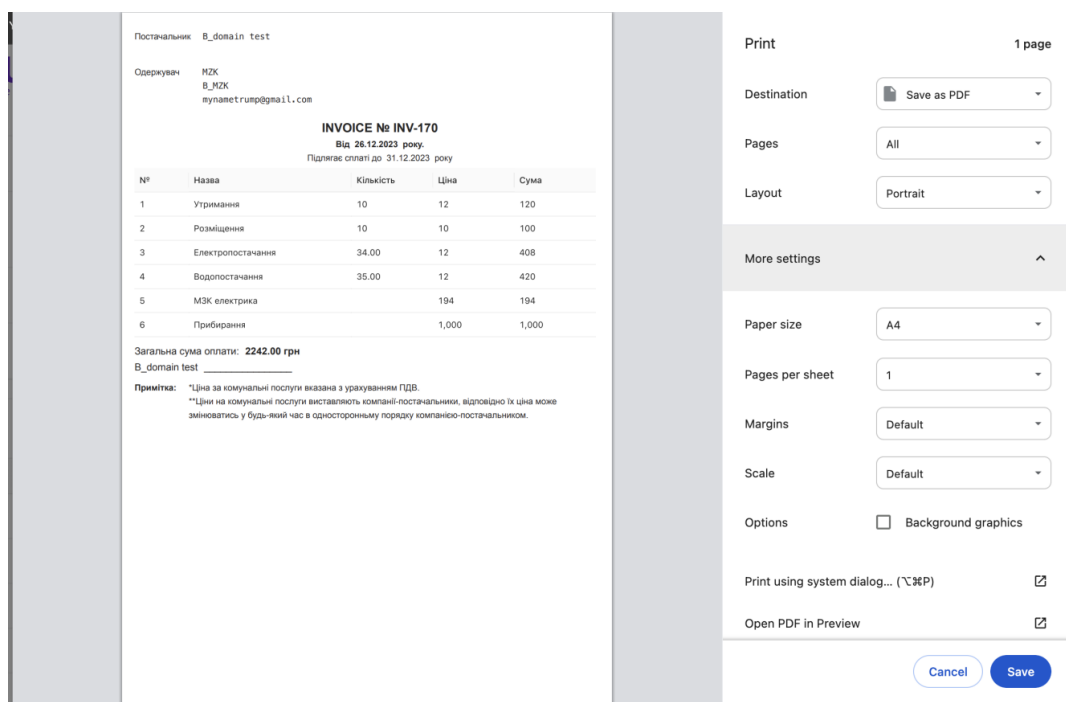


Рис 3.23 Роздрукування рахунку фактури

Окрім видалення та редагування, які присутні в усіх блоках нашого сервісу, платежі також можна переглядати.

X

Постачальник B_domain test

Одержувач MZK
B_MZK
mynametrump@gmail.com

INVOICE № INV-170
Від 26.12.2023 року.
Підлягає сплаті до 31.12.2023 року

№	Назва	Кількість	Ціна	Сума
1	Утримання Листопад	10	12	120
2	Розміщення Листопад	10	10	100
3	Електропостачання Листопад	34.00	12	408
4	Водопостачання Листопад	35.00	12	420
5	МЗК електрика Листопад		194	194
6	Прибирання		1,000	1,000

Всього на суму: **дві тисячі двісті сорок дві грн**
Загальна сума оплати: **2242 грн**
B_domain test _____

Примітка: *Ціна за комунальні послуги вказана з урахуванням ПДВ.
**Ціни на комунальні послуги виставляють компанії-постачальники, відповідно їх ціна може змінюватись у будь-який час в односторонньому порядку компанією-постачальником.

[Роздрукувати Документ](#)

[Закрити](#)

Рис 3.24 Перегляд платежу

Висновки до третього розділу

В даному розділі були наведені результати дослідження автоматизації процесів управління орендними відносинами та проведено їх розрахунки.

Була продемонстрована та детально описана робота додатку та його функцій.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено дослідження та детальний аналіз потреби створення інформаційно-статистичної системи орендних відносин на базі веб додатку, розглянуто ризики її створення та розраховано її інвестиційний потенціал, як ознаку потреби суспільства у подібному програмному забезпеченні. Описано систему управління орендними відносинами. Був проведений аналіз аналогів, визначений основний функціонал, на основі чого було проведено дослідження та виявлено потреби користувачів у подібних системах управління нерухомістю. Для реалізації поставлених задач було обрано такі інструменти: Мова програмування TypeScript, фреймворк для створення веб додатків Node.js, препроцесор для CSS – SCSS, API, створене за допомогою Node.js, база даних, створена у MongoDB.

Середовищем розробки було обрано – VSCode та UI бібліотеку для React застосунків – Ant Design.

Було проведено дослідження зміни цін на послуги та комунальні та загальні платежі орендарями та графічно відображено кількість зайнятих площ орендарями.

Графічні дані було відображено за допомогою наявних даних про роботу системи та бібліотекою AntV, представлену командою Ant Design.

Було реалізовано функціональну частину сайту. Описано усі методи та усі шляхи використання додатку користувачем.

Було проведено дослідження специфічності використання TypeScript для виконання математичних обчислень та обрано бібліотеку Big.js для точного обчислення математичних операцій з дробовими числами.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Donald G Firesmith, “Engineering Security Requirements”, Journal of Object Technology, Volume 2, Issue 1, 2003
2. S. Edlich, A. Friedland, J. Hampe, B. Brauer, “NoSQL: Einstieg in dielt Nichhtrelationaler Web 2.0 Datenbanken”, 2 nd edition.: Hanser Fachbuchverlag, 2010
3. Rabi Prasad Padhy, Manas Ranjan Patra, Suresh Chandra Satapathy, “RDBMS to NOSQL:Reviewing some next generation Non Realtional databases”, International Journal of Advanced Engineering Science and Technology, Volume 11, Issue 1, 2011
4. M. Rosenblum and P. Dorsey, “Oracle PL/SQL for dummies” , 2006
5. K. Sanobar, M. Vanita, “SQL Support over MongoDB using Metadata”, International Journal of Scientific and Research Publications, Volume 3, Issue 10, October 2013
6. Z. Wei-Ping, LI Ming-Xin, H. Chen, “Using MongoDB to Implement Textbook Management System instead of MySQL”, IEEE 3rd International Conference on Communication Software and Networks (ICCSN), ISSN 978-1-61284-486, 2011.
7. S. Hoberman, “Data Modeling for MongoDB”, Publisher by Technics Publications, LLC 2 Lindsley Road Basking Ridge, NJ 07920, USA, ISBN 978-1-935504-70-2, 2014.
8. Реформування житлово-комунального господарства: теорія, практика, перспективи: Монографія / О.М.Тищенко, М.О.Кизим, Т.П.Юр’єва, С.Ю.Юр’єва, І.В.Покуца. – Х.: ВД “ІНЖЕК”, 2008. – 368 с.
9. Аналітика ринку нерухомості: методологія та принципи сучасної оцінки: монографія / В.О. Воронін, Е.В. Лянце, М.М. Мамчин. – Львів: видавництво «Магнолія– 2006», 2014. – 304 с.

10. Uhnenko Ye.B., Uzhviieva O.M., Tymchenko O.M., Kamchatna S.M. Inzhenernoheodezychnе zabezpechennia prostorovoi pryviazky pryrodnykh ob'ektiv [Engineering and geodetic support of spatial reference of natural objects]. Zbirnyk naukovykh prats UkrDUZT. Vol. 184 (2019). P. 61 – 67.
11. Fresno, V. and Ribeiro, A. (2001). Feature Selection and Dimensionality Reduction in Web Pages Representation. In *International ICSC Congress on Computational Intelligence: Methods & Applications* (pp. 416-421). Bangor, Wales, U.K.
12. Бутенко Б. В. Веб-розробка за допомогою бібліотеки React та Redux / Б. В. Бутенко // Радіоелектроніка та молодь у XXI столітті : матеріали 27-го Міжнар. Молодіж. Форуму, 10–12 травня 2023 р. – Харків: ХНУРЕ, 2023. – Т. 6, ч. 1. – С. 321–322.
13. M. S. Mikowski and J. C. Powell, *Single Page Web Applications*, MANNING PUBLICATIONS.
14. T. Lance, A. Martin and W. Carey, “Performance Comparison of Dynamic Web Technologies”, *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 3, December 2003.
15. T. Scott, T. Michiaki, S. Toyotaro, T. Akihiko and O. Tamiya, “Performance Comparison of PHP and JSP as Server-Side Scripting Languages”, *Middleware*, 2008. Verdier O.
16. E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite and W. Zwaenepoel, “Performance Comparison of Middleware Architectures for Generating Dynamic Web Content”, *Proceedings of 4th Middleware Conference*, June 2003.
17. P. Neves, N. Paiva and J. Durães, “A comparison between JAVA and PHP”, *C3S2E '13 Proceedings of the International C* Conference on Computer Science and Software Engineering*, 2013.
18. Manhas, D. 2013. A Study of Factors Affecting Websites Page Loading Speed for Efficient Web Performance. *International Journal of Computer Sciences and Engineering*.

19. Jun, S.-P., Yoo, H.S. & Choi, S. 2017. Ten years of research change using Google Trends: From the perspective of big data utilizations and applications. *Technological Forecasting and Social Change*, 130.
20. Heričko, T., Šumak, B. & Brdник, S. 2021. *Towards Representative Web Performance Measurements with Google Lighthouse*. University of Maribor Press.
21. Banks, A. & Porcello, E. 2020. *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, Inc
22. P. Kishore and M. B M, "Evolution of Client-Side Rendering over Server-Side Rendering," vol. 3, no. 2, pp. 1–10, 2020.
23. Рукосмотров Д.В. Переваги та недоліки використання монолітної архітектури при розробці програмного забезпечення. Тези VI Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», м. Житомир, 29-30 листопада 2023 р. – Житомир: Житомирська політехніка, 2023. – с. 103-104.
24. Рукосмотров Д.В. Методології розробки програмного забезпечення. Тези VI Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», м. Житомир, 29-30 листопада 2023 р. – Житомир: Житомирська політехніка, 2023. – с. 87.
25. React documentation [Електронний ресурс] – <https://react.dev/learn/React> Tutorial [Електронний ресурс] – <https://www.w3schools.com/react/>
26. SASS – next level CSS? [Електронний ресурс] – <https://www.ionos.com/digitalguide/websites/web-development/sass/>
27. Differences between-SCSS and LESS [Електронний ресурс] – <https://stackoverflow.com/questions/52003198/differences-between-scss-and-less>
28. Difference between SASS and LESS [Електронний ресурс] – <https://www.educba.com/sass-vs-less/>
29. SASS vs LESS [Електронний ресурс] – <https://www.keycdn.com/blog/sass-vs-less>

30. Difference between TypeScript and JavaScript [Электронный ресурс] – <https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/>
31. Difference between Next.js and Create React App [Электронный ресурс] – <https://stackoverflow.com/questions/62967958/what-is-the-difference-between-next-js-and-create-react-app>
32. MongoDB – PostgreSQL [Электронный ресурс] – <https://www.mongodb.com/compare/mongodb-postgresql>
33. Prisma and Mongoose [Электронный ресурс] – <https://www.prisma.io/docs/orm/more/comparisons/prisma-and-mongoose>
34. Material Design vs Ant Design [Электронный ресурс] – <https://www.material-tailwind.com/blog/material-design-vs-ant-design>
35. Comparing Prisma and Mongoose [Электронный ресурс] – <https://medium.com/@ibezimchike/comparing-prisma-and-mongoose-fbc42d069c47>
36. TypeScript vs JavaScript: Which One You Should Use , and Why [Электронный ресурс] – SitePoint – <https://www.sitepoint.com/typescript-vs-javascript/>
37. Server-side Rendering (SSR) [Электронный ресурс] – <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering/>
38. Server-Side Rendering With Next.js [Электронный ресурс] – <https://medium.com/swlh/server-side-rendering-with-next-js-56f84f98f9bd/>
39. What is SSR in Next.JS? [Электронный ресурс] – <https://www.geeksforgeeks.org/what-is-ssr-in-nextjs/>
40. Building React Components Using Ant Design [Электронный ресурс] – <https://blog.openreplay.com/building-react-components-with-ant-design/>

ДОДАТКИ

```

/* eslint-disable @typescript-eslint/ban-ts-comment */
// @ts-nocheck
import type { NextApiRequest, NextApiResponse } from 'next'
import start, { Data } from '@pages/api/api.config'
import Street from '@common/modules/models/Street'
import Domain from '@common/modules/models/Domain'
import { getCurrentUser } from '@utils/getCurrentUser'

start()

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {
  const { isGlobalAdmin } = await getCurrentUser(req, res)

  switch (req.method) {
    case 'GET':
      try {
        const { domainId, limit = 0 } = req.query
        if (domainId) {
          const domain = await Domain.findOne({ _id: domainId })
            .populate({
              path: 'streets',
              select: '_id address city',
              options: { limit: +limit }
            })

          return res
            .status(200)
            .json({ success: true, data: domain?.streets || [] })
        }

        const streets = await Street.find({})
          .limit(+limit)

        return res.status(200).json({
          success: true,
          data: streets,
        })
      } catch (error) {
        return res.status(400).json({ success: false })
      }

    case 'POST':
      try {
        if (isGlobalAdmin) {
          // TODO: body validation
          const street = await Street.create(req.body)
          return res.status(200).json({ success: true, data: street })
        } else {

```

```

        return res
            .status(400)
            .json({ success: false, message: 'not allowed' })
    }
} catch (error) {
    return res.status(400).json({ success: false })
}
}
}

/* eslint-disable @typescript-eslint/ban-ts-comment */
// @ts-nocheck
import type { NextApiRequest, NextApiResponse } from 'next'
import start, { Data } from '@pages/api/api.config'
import Street from '@common/modules/models/Street'
import { getCurrentUser } from '@utils/getCurrentUser'
start()

export default async function handler(
    req: NextApiRequest,
    res: NextApiResponse<Data>
) {
    const { isGlobalAdmin } = await getCurrentUser(req, res)

    if (!isGlobalAdmin) {
        return res.status(400).json({ success: false, message: 'not allowed' })
    }

    switch (req.method) {
        case 'DELETE':
            try {
                await Street.findByIdAndRemove(req.query.id).then((street) => {
                    if (street) {
                        return res.status(200).json({
                            success: true,
                            data: 'Street ' + req.query.id + ' was deleted',
                        })
                    } else {
                        return res.status(400).json({
                            success: false,
                            data: 'Street ' + req.query.id + ' was not found',
                        })
                    }
                })
            } catch (error) {
                return res.status(400).json({ success: false, error: error })
            }
        }
    }
}

```

Додаток Б

```
// eslint-disable-next-line @typescript-eslint/ban-ts-comment
// @ts-nocheck
import type { NextApiRequest, NextApiResponse } from 'next'
import RealEstate from '@common/modules/models/RealEstate'
import { getCurrentUser } from '@utils/getCurrentUser'
import start, { ExtendedData } from '@pages/api/api.config'
import Domain from '@common/modules/models/Domain'
import { filterOptions, getDistinctCompanyAndDomain } from '@utils/helpers'

start()

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse<ExtendedData>
) {
  const { isGlobalAdmin, isDomainAdmin, isAdmin, isUser, user } =
    await getCurrentUser(req, res)

  switch (req.method) {
    case 'GET':
      try {
        const options = {}
        const {
          companyIds,
          domainIds,
          domainId,
          streetId,
          companyId,
          limit = 0,
        } = req.query

        if (domainId) options.domain = domainId
        if (streetId) options.street = streetId
        if (companyId) options._id = companyId

        if (domainIds) {
          options.domain = filterOptions(options?.domain, domainIds)
        }

        if (companyIds) {
          options._id = filterOptions(options?._id, companyIds)
        }

        if (isDomainAdmin) {
          const domains = await Domain.find({
```

```

        adminEmails: { $in: [user.email] },
      })
      const domainsIds = domains.map((i) => i._id.toString())
      if (domainId) {
        // TODO: add test. Domain admin can't fetch realEstate which is not belong
to him
        if (!domainsIds.includes(domainId)) {
          return res.status(400).json({
            success: false,
            message: 'not allowed to fetch such domainId',
          })
        }
      } else {
        options.domain = { $in: domainsIds }
      }
    }

    if (isUser) {
      options.adminEmails = { $in: [user.email] }
    }

    const realEstates = await RealEstate.find(options)
      .limit(+limit)
      .populate({
        path: 'domain',
        select: '_id name description',
      })
      .populate({ path: 'street', select: '_id address city' })

    const { distinctDomains, distinctCompanies } =
      await getDistinctCompanyAndDomain({
        isGlobalAdmin,
        user,
        companyGroup: '_id',
        model: RealEstate,
      })

    return res.status(200).json({
      domainsFilter: distinctDomains?.map(({ domainDetails }) => ({
        text: domainDetails.name,
        value: domainDetails._id,
      })),
      realEstatesFilter: distinctCompanies?.map(({ companyDetails }) => ({
        text: companyDetails.companyName,
        value: companyDetails._id,
      })),
      data: realEstates,
      success: true,
    })
  } catch (error) {
    return res.status(400).json({ success: false, message: error })
  }
}

```

```

case 'POST':
  try {
    if (!isAdmin) {
      return res
        .status(400)
        .json({ success: false, message: 'not allowed' })
    }
    // TODO: body validation
    const realEstate = await RealEstate.create(req.body)
    return res.status(200).json({ success: true, data: realEstate })
  } catch (error) {
    return res.status(400).json({ success: false, message: error })
  }
}

// eslint-disable-next-line @typescript-eslint/ban-ts-comment
// @ts-nocheck
import type { NextApiRequest, NextApiResponse } from 'next'
import start, { Data } from '@pages/api/api.config'
import RealEstate from '@common/modules/models/RealEstate'
import { getCurrentUser } from '@utils/getCurrentUser'

start()

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse<Data>
) {
  const { isGlobalAdmin, user } = await getCurrentUser(req, res)

  if (!isGlobalAdmin) {
    return res
      .status(400)
      .json({ success: false, message: 'not allowed' } as Data)
  }

  switch (req.method) {
    case 'GET':
      try {
        const my = await RealEstate.find({ adminEmails: { $in: [user.email] } })

        return res.status(200).json({ success: true, data: my })
      } catch (error) {
        return res.status(400).json({ success: false })
      }
    }
  }
}

```

```

import { DeleteOutlined } from '@ant-design/icons'
import { Alert, Popconfirm, Table, message } from 'antd'
import { ColumnType } from 'antd/lib/table'
import { useRouter } from 'next/router'

import {
  useDeleteStreetMutation,
  useGetAllStreetsQuery,
} from '@common/api/streetApi/street.api'
import { IStreet } from '@common/modules/models/Street'
import { AppRoutes } from '@utils/constants'
import RealEstateBlock from '@common/components/DashboardPage/blocks/realEstates'

export interface Props {
  domainId?: string
}

const StreetsTable: React.FC<Props> = ({ domainId }) => {
  const router = useRouter()
  const isOnPage = router.pathname === AppRoutes.STREETS

  const { data, isLoading, isError } = useGetAllStreetsQuery({ domainId, limit:
isOnPage ? 0 : 5 })

  const [deleteStreet, { isLoading: deleteLoading }] = useDeleteStreetMutation()

  const handleDelete = async (id: string) => {
    const response = await deleteStreet(id)
    if ('data' in response) {
      message.success('Видалено!')
    } else {
      message.error('Помилка при видаленні')
    }
  }

  if (isError) return <Alert message="Помилка" type="error" showIcon closable />

  return (
    <Table
      rowKey="_id"
      size="small"
      pagination={false}
      loading={isLoading}
      columns={getDefaultColumns(handleDelete, deleteLoading)}
      expandable={
        domainId && {
          expandedRowRender: (street) => (
            <RealEstateBlock domainId={domainId} streetId={street._id} />
          ),
        }
      }
    />
  )
}

```

```

        dataSource={data}
        scroll={{ x: 500 }}
      />
    )
  }
}

const getDefaultColumns = (
  handleDelete?: (streetId: string) => void,
  deleteLoading?: boolean
): ColumnType<any>[] => [
  {
    title: 'Місто',
    width: '25%',
    dataIndex: 'city',
  },
  {
    title: 'Вулиця',
    dataIndex: 'address',
  },
  {
    align: 'center',
    fixed: 'right',
    title: '',
    width: 50,
    render: (_, street: IStreet) => (
      <Popconfirm
        title={`Ви впевнені що хочете видалити вулицю ${street.address} (м.
${street.city})?`}
        onConfirm={() => handleDelete(street._id)}
        cancelText="Відміна"
        disabled={deleteLoading}
      >
        <DeleteOutlined />
      </Popconfirm>
    ),
  },
]

export default StreetsTable

import { PlusOutlined, SelectOutlined } from '@ant-design/icons'
import { Button } from 'antd'
import { useRouter } from 'next/router'
import { useState } from 'react'

import AddStreetModal from '@common/components/AddStreetModal'
import { AppRoutes } from '@utils/constants'

export interface Props {
  showAddButton?: boolean
}

const StreetsHeader: React.FC<Props> = ({ showAddButton = false }) => {

```

```

const router = useRouter()
const [isModalOpen, setIsModalOpen] = useState(false)

const openModal = () => setIsModalOpen(true)
const closeModal = () => setIsModalOpen(false)

return (
  <div style={{ display: 'flex', justifyContent: 'space-between' }}>
    <Button
      type="link"
      onClick={() => {
        router.push(AppRoutes.STREETS)
      }}
    >
      Адреси
    <SelectOutlined />
  </Button>

  {showAddButton && (
    <Button type="link" onClick={openModal}>
      <PlusOutlined /> Додати
    </Button>
  )}
  {isModalOpen && <AddStreetModal closeModal={closeModal} />}
</div>
)
}

export default StreetsHeader

```