ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних домах за допомогою nest.js»

на здобуття освітнього ступеня магістра

зі спеціальності 124 Системній аналіз

освітньо-професійної програми «Інтелектуальні системи управління»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Кирило ЛОЛЕНКО
(підпис)

Виконав: здобувач вищої освіти гр. САДМ-61
Кирило ЛОЛЕНКО
Керівник: Михайло КУЗЬМІЧ
к.т.н., доцент
Рецензен:
науковий ступінь, вчене звання
Ім’я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра: Інформаційних систем та мереж

Ступінь вищої освіти: Магістр

Спеціальність: 124 «Системний аналіз»

Освітньо-професійна програма: Інтелектуальні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ Каміла СТОРЧАК
«___» _____ 20_____ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лоленко Кирило Олексійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: «Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних домах за допомогою Nest.js»
Керівник кваліфікаційної роботи: Михайло КУЗЬМІЧ, Доктор філософії (PhD)
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
навчального закладу від «_____»_____ 20_____ р. № ____.
2. Строк подання студентом роботи: «_____»_____ 20_____ р.
3. Вихідні дані до роботи:
   1. Опис проблематики автоматизації обліку розрахунків у багатоквартирних будинках;
   2. Вимоги до функціоналу платформи (збір даних, обробка платежів, звітність);
   3. Технічні характеристики платформи, середовище виконання (Nest.js, база даних, інтеграція з API);
   4. Інформація про типові користувачі платформи (адміністратори, мешканці, керуючі компанії).
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   1. Аналіз предметної області та вимог до платформи;
   2. Розробка архітектури системи та бази даних;
   3. Реалізація основних модулів платформи: облік розрахунків, аутентифікація, звіти;
   4. Тестування, оптимізація та оцінка ефективності платформи;
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «_____»_____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Характеристика та аналіз предметної області		
2	Дослідження глобальних тенденцій		
3	Позитивний ефект автоматизації обліків		
4	Прогнозування розвитку технологій		
5	Аналіз зібраної інформації		
6	Ідеї для покращення сервісів		
7	Розробка математичних моделей		
8	Обґрунтування вибору інструментів		
9	Проектування архітектури системи		
10	Реалізація інформаційної підсистеми		

Здобувач вищої освіти _____ Кирило ЛОЛЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Михайло КУЗЬМІЧ
(підпис)

РЕФЕРАТ

Кваліфікаційний проект містить 73 стор., 13 рис., 2 табл., 30 джерел., 4 додатків, 4 формул.

Мета роботи – створення цифрової платформи для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою NestJS, яка задовольняє потреби користувачів та забезпечує ефективне управління.

Об'єкт дослідження – функціональність, структура платформи та процес автоматизації обліку розрахунків у багатоквартирних будинках.

Предмет дослідження – розробка цифрової платформи для автоматизації обліку розрахунків із використанням технологій NestJS, PostgreSQL та Prisma ORM.

Короткий зміст роботи:

У рамках проєкту було виконано такі завдання:

- проаналізовано потреби мешканців та адміністраторів житлових комплексів;
- розроблено структуру платформи та її функціональність;
- інтегровано сучасні методи аутентифікації та захисту даних;
- оптимізовано роботу з базою даних PostgreSQL;
- створено інтуїтивно зрозумілий інтерфейс для користувачів.

Для реалізації проєкту використовувалися такі інструменти: мова програмування TypeScript, фреймворк NestJS, Prisma ORM, база даних PostgreSQL, інструменти для побудови RESTful API, а також платформи для тестування й розгортання.

Результати розробки демонструють переваги у порівнянні з існуючими рішеннями. Система відзначається легкістю використання, високою продуктивністю та гнучкістю в інтеграції нових функцій.

Одержані результати можуть бути використані компаніями, що управляють багатоквартирними будинками, для автоматизації розрахунків та покращення взаємодії з мешканцями. Також платформа стане корисною для власників нерухомості, які прагнуть ефективно управляти об'єктами та здійснювати облік платежів.

КЛЮЧОВІ СЛОВА: цифровізація, інформаційні системи, сервіси, аналіз, показники, ефективність, інформаційна система, next.js, tailwind, prisma, postgresql, веб-сайт, інтерфейс, облік, розрахунки, діаграма.

ABSTRACT

Text part of the master's qualification work: 73 pages, 13 pictures, 2 table, 30 sources, 4 appendices, 4 formulas.

Objective of the work – to create a digital platform for automating payment accounting in multi-apartment buildings using NestJS, aimed at meeting user needs and ensuring effective management.

Research object – the functionality, structure of the platform, and the process of automating payment accounting in multi-apartment buildings.

Research subject – the development of a digital platform for automating payment accounting using NestJS, PostgreSQL, and Prisma ORM technologies.

Brief description of the work:

As part of the project, the following tasks were completed:

- analysis of the needs of residents and administrators of residential complexes;
- development of the platform’s structure and functionality;
- integration of modern authentication methods and data security measures;
- optimization of operations with the PostgreSQL database;
- creation of an intuitive user interface.

The project was implemented using the following tools: TypeScript programming language, NestJS framework, Prisma ORM, PostgreSQL database, tools for building RESTful APIs, as well as platforms for testing and deployment.

The results of the development demonstrate advantages compared to existing solutions. The system is characterized by ease of use, high performance, and flexibility in integrating new features.

The obtained results can be utilized by companies managing multi-apartment buildings to automate payment accounting and improve interactions with residents. Additionally, the platform can be useful for property owners who aim to efficiently manage their properties and keep track of payments.

KEYWORDS**:** digitalization, information systems, services, analysis, indicators, efficiency, information system, next.js, tailwind, prisma, postgresql, website, interface, accounting, calculations, diagram.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Лоленко К.О. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 124 Системний аналіз
(код, найменування спеціальності)
освітньо-професійної програми «Інтелектуальні системи управління»

на тему: «Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних домах за допомогою Nest.js».
Кваліфікаційна робота і рецензія додаються.

Директор ННІ ІТ _____ Катерина НЕСТЕРЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

Висновок керівника магістерської роботи

Здобувач Кирило ЛОЛЕНКО успішно виконав магістерську роботу на тему "Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою Nest.js". У роботі він продемонстрував високий рівень професійних знань, розробивши функціональну та ефективну платформу з використанням сучасних технологій NestJS, Prisma ORM та PostgreSQL. Розв'язання поставлених задач, таких як інтеграція методів аутентифікації, оптимізація бази даних та створення зручного інтерфейсу, було здійснено на високому рівні. Робота має практичну цінність і може бути застосована в управлінні багатоквартирними будинками.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Кирила ЛОЛЕНКА на оцінку «_____» та присвоїти йому кваліфікацію магістр з системного аналізу.

Керівник кваліфікаційної роботи _____ Михайло КУЗЬМІЧ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«_____» _____ 20_____ року

Висновок кафедри про кваліфікаційної роботу

Кваліфікаційна робота розглянута. Здобувач Кирило ЛОЛЕНКО допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри ІСТ _____ Каміла СТОРЧАК
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня магістра

здобувача вищої освіти Лоленка Кирила Олексійовича
(прізвище, ім'я, по батькові)
на тему «Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних домах за допомогою Nest.js»

Актуальність:

Магістерська робота «Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою Nest.js» є актуальною через зростаючу потребу в автоматизації управління багатоквартирними будинками. Платформа вирішує важливі задачі, такі як автоматизація розрахунків, прозорість даних і зручність взаємодії з користувачами. Використання технологій NestJS, PostgreSQL та Prisma ORM підтверджує інноваційність і технічну цінність проєкту, який корисний як для компаній-управителів, так і для власників нерухомості.

Позитивні сторони:

1. Використання сучасних технологій (NestJS, Prisma ORM, PostgreSQL), що забезпечують високу продуктивність і масштабованість платформи.
2. Чітко визначена архітектура платформи та зручний, інтуїтивно зрозумілий інтерфейс для кінцевих користувачів.
3. Високий рівень деталізації і системний підхід до вирішення задач, включаючи безпеку та оптимізацію бази даних.

Недоліки:

1. Відсутні детальні рекомендації щодо масштабування платформи в умовах значного зростання кількості користувачів.
2. Деякі аспекти взаємодії з користувачами, наприклад, інтеграція з мобільними платформами, висвітлено недостатньо.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи магістерської.

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку «_____», а здобувач Кирило ЛОЛЕНКО заслуговує присвоєння кваліфікації: магістр з системного аналізу*

Рецензент:

науковий ступінь, вчене звання	*підпис*	*Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	(англ. Application Programming Interface) – прикладний програмний інтерфейс;
CTA	(англ. Call To Action) – заклик до дії;
IoT	(англ. Internet of Things) – інтернет речей;
JWT	(англ. JSON Web Token) – веб-токен JSON;
MVP	(англ. Minimum Viable Product) – мінімально життєздатний продукт;
UX	(англ. User Experience) – дизайн користувацького досвіду;
ІС	інформаційні системи;
ЖКГ	житлово-комунальне господарство;
ЖКП	житлово-комунальні послуги;
ЖЕК	житлово-експлуатаційна контора.

ВСТУП

Проєктування цифрової платформи для автоматизації обліку розрахунків у багатоквартирних будинках є актуальною темою, враховуючи сучасні потреби в цифровізації процесів управління житловим фондом. Швидкий розвиток технологій, зокрема засобів для обробки даних та інтеграції з різними платформами, сприяє зростанню попиту на автоматизовані системи, які забезпечують зручність у використанні та ефективність управління.

Предметною галуззю дослідження є проєктування цифрової платформи, спрямованої на автоматизацію обліку розрахунків та взаємодії між мешканцями й адміністрацією багатоквартирних будинків. Основним об’єктом дослідження є розробка функціоналу та інтерфейсу платформи, яка забезпечить швидкий і зручний доступ до даних, проведення розрахунків та комунікацію з користувачами.

Метою даного проєктування є створення цифрової платформи, яка надає інструменти для автоматизації обліку розрахунків, мінімізує ручну роботу й помилки, а також забезпечує комфортне використання для мешканців і керуючих організацій.

Для досягнення цієї мети передбачені наступні завдання:

- аналіз потреб та вимог мешканців і адміністрації багатоквартирних будинків;
- розробка структури бази даних postgresql для зберігання та обробки інформації про розрахунки;
- визначення основних функціональних можливостей платформи, зокрема обліку платежів, автоматичного нарахування заборгованостей, генерації звітів тощо;
- інтеграція сучасних технологій для забезпечення безпеки даних і стабільної роботи платформи;
- проєктування інтуїтивно зрозумілого інтерфейсу для користувачів із різними рівнями технічної підготовки;

- тестування платформи з подальшим удосконаленням на основі отриманих результатів.

Для вирішення поставлених завдань будуть використовуватися такі методи дослідження:

- аналіз літературних джерел і сучасних підходів у проєктуванні систем автоматизації;
- збір та аналіз вимог користувачів шляхом опитування й дослідження практичного досвіду;
- розробка прототипів та їх тестування для визначення оптимальної структури й функціоналу;
- впровадження та інтеграція платформи з подальшою перевіркою її продуктивності.

Огляд літературних джерел дозволяє визначити актуальність теми автоматизації обліку розрахунків та розкрити переваги сучасних технологій, таких як PostgreSQL у поєднанні з NestJS та Prisma ORM. Дослідження подібних рішень показує, що попри наявність аналогів, існує потреба у створенні платформ, які краще адаптовані до специфічних вимог локальних користувачів.

Усвідомлення теоретичної та практичної значущості роботи полягає у створенні функціональної платформи, яка спрощує процес обліку розрахунків, підвищує його точність і знижує витрати часу та ресурсів.

Структура роботи передбачає наступні розділи:

- аналіз літератури та сучасних технологій для автоматизації облікових процесів;
- визначення вимог користувачів та функціональних особливостей платформи;
- проєктування структури бази даних postgresql та архітектури системи;
- реалізація функціоналу платформи та інтеграція з існуючими системами;
- тестування і вдосконалення розробленої платформи.

Результатом проєкту стане функціональна платформа, яка задовольнить потреби користувачів, сприятиме підвищенню ефективності обліку та матиме перспективу для масштабування в майбутньому.

РОЗДІЛ 1 ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія процесу внесення показників лічильників

20–30 років тому в Україні процес внесення показників лічильників (води, газу, електроенергії) суттєво відрізнявся від сучасного. Він був менш автоматизованим і потребував активної участі людей. Основним способом передачі показників були спеціальні паперові квитанції. Люди заповнювали ці квитанції вручну, вказуючи попередні та поточні показники лічильників, а також розраховували спожитий обсяг ресурсу. Такі квитанції зазвичай вкладали у поштові скриньки або передавали працівникам ЖЕКів (житлово-експлуатаційних контор).

Люди приносили квитанції до місцевих ЖЕКів або абонентських відділів постачальників комунальних послуг. У цих відділах працівники вручну фіксували дані в паперових журналах чи базах. Часто біля кас чи офісів утворювалися довгі черги, особливо під час періодів активної передачі показників або сплати рахунків.

Працівники ЖЕКів або постачальників комунальних послуг (контролери) періодично здійснювали обхід квартир. Вони особисто знімали показники з лічильників і записували їх у спеціальні журнали. Це було особливо поширено для газових та електролічильників, розташованих у під'їздах чи безпосередньо у квартирах.

У 1990-х і на початку 2000-х деякі постачальники комунальних послуг почали впроваджувати передачу показників через телефон. Для цього потрібно було зателефонувати на спеціальний номер і продиктувати оператору дані лічильника. Такий метод був прогресивним на той час, але через обмежену кількість телефонних ліній нерідко виникали труднощі – телефон був зайнятий, і додзвонитися було складно.

В окремих випадках люди надсилали квитанції поштою. Вони вкладали заповнені форми до конверта і відправляли їх до абонентського відділу. Цей спосіб був повільним, але користувався популярністю у віддалених районах, де важко було дістатися до офісів ЖЕКів.

Люди часто самостійно розраховували споживання та суму для оплати, базуючись на тарифах. Відсутність автоматизованих систем і доступу до актуальної інформації іноді спричиняла помилки в розрахунках.

Процес був тривалим через ручне введення даних і залежність від людського фактора. Помилки у фіксації показників або втрати квитанцій ускладнювали подальші розрахунки. Всі операції були ручними, що сприяло виникненню помилок та затримок.

1.2 Сучасні проблеми обліку рахунків

У сфері житлово-комунального господарства (ЖКГ) України актуальна ситуація показує, що попри розвиток нових житлових комплексів та ініціативу старих будинків, які відмовились від послуг ЖЕКів на користь самостійного обслуговування (приблизно 20-30%), значна частина житлового фонду (близько 70%) все ще користується послугами ЖЕКів. ЖЕКи досі використовують паперові квитанції для сплати комунальних послуг, які розповсюджуються через поштові ящики у будинках.

Цей процес створює значні незручності для населення, особливо для людей похилого віку, яким доводиться особисто відвідувати Укрпошту або інші установи для здійснення платежів. Водночас молодь, яка більш обізнана з цифровими технологіями, здійснює платежі через банківські додатки, що значно спрощує процес.

Основною проблемою є відсутність єдиної платформи, яка б дозволила централізовано управляти усіма аспектами житлово-комунальних послуг. Така платформа мала б функціонал для перегляду історії платежів, заборгованостей, подачі заявок на ремонт, або інших послуг, таких як посипання прибудинкових доріг піскосоляною сумішшю у зимовий час. Крім того, вона могла б забезпечити проведення зборів мешканців онлайн або організацію голосувань, що є суттєвою перевагою для управління багатоквартирними будинками.

Наявність такої платформи сприяла б не тільки зручності у користуванні послугами, а й покращенню комунікації між мешканцями та керуючою установою.

Можливість подання заявок чи питань безпосередньо на платформі або через чатбот значно прискорила б вирішення проблем. Також важливою є функція попереднього внесення номерів машин гостей для контролю доступу на приватну прибудинкову територію.

На сьогоднішній день процес комунікації з ЖЕКами, керуючими компаніями чи ОСББ є тривалим, що може призводити до загострення проблем у випадку затягування розгляду питань. Таким чином, інтеграція сучасних цифрових рішень у сферу ЖКГ здатна значно підвищити ефективність та зручність взаємодії між усіма учасниками процесу[30].

1.3 Переваги автоматизації розрахунку обліків

З розвитком інформаційних технологій та впровадженням автоматизованих систем обліку, процес збору та обробки даних значно спростився. Замість традиційних паперових квитанцій, мешканці тепер можуть передавати показники лічильників онлайн, використовуючи спеціальні додатки або веб-портали. Це дозволяє уникнути помилок, пов'язаних з ручним введенням даних, та забезпечує точність і прозорість розрахунків.

Автоматизація також має значний вплив на зменшення витрат на обслуговування, оскільки керуючі компанії можуть ефективніше управляти ресурсами та оперативно реагувати на будь-які проблеми. Мешканці отримують актуальну інформацію про стан своїх рахунків та можуть здійснювати платежі онлайн, що значно спрощує процес взаємодії з керуючими компаніями[29].

Переваги автоматизації обліку розрахунків у багатоквартирних будинках полягають у наступному:

– Ефективність управління: Автоматизовані системи дозволяють швидко та точно обробляти великі обсяги даних, що значно підвищує ефективність управління житловим фондом. Це включає облік платежів, контроль заборгованостей, формування звітності та інші аспекти фінансового управління;

- Зменшення витрат: Використання автоматизованих систем дозволяє зменшити витрати на обслуговування житлового фонду. Це досягається за рахунок зменшення кількості помилок, скорочення часу на обробку даних та зниження потреби в ручній праці;
- Покращення якості послуг: Автоматизація дозволяє покращити якість надання послуг мешканцям. Це включає швидке та точне оброблення запитів, своєчасне інформування про стан рахунків та можливість онлайн-оплати;
- Прозорість та контроль**:** Автоматизовані системи забезпечують прозорість фінансових операцій та контроль за їх виконанням. Це дозволяє уникнути корупційних схем та забезпечити чесність у розрахунках;
- Інтеграція з іншими системами: Сучасні автоматизовані системи можуть інтегруватися з іншими інформаційними системами, що дозволяє створити єдину інформаційну базу для управління житловим фондом;
- Безпека даних: Автоматизовані системи забезпечують високий рівень захисту даних мешканців. Використання сучасних технологій шифрування та захисту інформації дозволяє мінімізувати ризики несанкціонованого доступу та втрати даних;
- Екологічність: Перехід на автоматизовані системи обліку сприяє зменшенню використання паперу та інших ресурсів, що позитивно впливає на навколишнє середовище. Це також сприяє зменшенню викидів вуглекислого газу, пов'язаних з виробництвом та транспортуванням паперових документів;
- Аналіз та прогнозування: Автоматизовані системи дозволяють збирати та аналізувати великі обсяги даних, що дає можливість прогнозувати майбутні потреби та витрати. Це допомагає керуючим компаніям

приймати обґрунтовані рішення та планувати свою діяльність на основі реальних дани;

- Підвищення задоволеності мешканців**:** Завдяки автоматизації мешканці можуть отримувати більш якісні та швидкі послуги, що підвищує їх задоволеність. Можливість оперативно отримувати інформацію про стан рахунків та здійснювати платежі онлайн робить процес взаємодії з керуючими компаніями більш зручним та комфортним;
- Гнучкість та масштабованість: Автоматизовані системи легко адаптуються до змін у законодавстві та вимогах ринку. Вони можуть бути масштабовані для обслуговування як невеликих житлових комплексів, так і великих багатоквартирних будинків, що робить їх універсальним інструментом для управління житловим фондом.

Автоматизація обліку розрахунків у житловому фонді є важливим кроком до підвищення ефективності управління, зменшення витрат та покращення якості послуг для мешканців. Інтеграція з іншими смарт-технологіями та системами дозволяє створити єдину інформаційну базу, що забезпечує прозорість, контроль та безпеку фінансових операцій.

Повний і точний опис умов розрахунків, включаючи тарифи, доступність послуг, правила оплати та додаткові послуги, є важливою складовою інформації для користувачів. Крім того, платформа повинна надавати можливість пробної підписки на продукт, що дозволить отримати об'єктивну оцінку якості послуг.

Іншим важливим елементом інформації, що необхідна для прийняття розумних рішень, є можливості порівняння різних пропозицій. Платформа має забезпечити зручний інтерфейс для порівняння різних послуг за різними параметрами, такими як ціна, якість, зручності та інші фактори.

Загальна ідея полягає в тому, щоб цифрова платформа для автоматизації обліку розрахунків надавала усю необхідну інформацію на одному місці, забезпечуючи користувачам можливість здійснювати обґрунтовані рішення щодо

оплати послуг. Це допомагає економити час і зусилля, а також сприяє підвищенню задоволеності користувачів та ефективності процесу обліку розрахунків.

1.4 Глобальні тенденції цифровізації в ЖКГ

Глобальні тенденції свідчать про те, що впровадження цифрових технологій у різні сфери життя значно підвищує ефективність управління та задоволеність користувачів. Наприклад, у багатьох країнах Європи та США вже давно впроваджено автоматизовані системи обліку та управління комунальними послугами, які дозволяють користувачам здійснювати платежі онлайн, подавати заявки на ремонт та отримувати актуальну інформацію про стан своїх рахунків.

Приклади цифровізації у світі

- **Німеччина**: У Німеччині активно використовуються платформи для управління житловим фондом, такі як "Allthings", що дозволяє мешканцям управляти всіма аспектами свого житла через єдину платформу. Вона включає функції для подання заявок на ремонт, проведення опитувань, організації зборів мешканців та багато іншого.;
- **Швеція**: У Швеції платформа "Tmpl" забезпечує мешканцям доступ до різних послуг, таких як онлайн-оплата комунальних послуг, подання заявок на ремонт, отримання новин від керуючої компанії та участь у голосуваннях. Ця платформа дозволяє ефективно комунікувати між мешканцями та керуючими компаніями;
- **Сінгапур**: Сінгапур є лідером у впровадженні "розумних" рішень для управління містами. Платформа "Smart Nation" об'єднує різні аспекти управління, включаючи житловий фонд, транспорт, охорону здоров'я та енергетику. Вона дозволяє мешканцям здійснювати онлайн-платежі, подавати заявки на послуги та отримувати актуальну інформацію про стан своїх рахунків;
- **США**: У Сполучених Штатах платформа "Zego" пропонує широкий спектр послуг для мешканців багатоквартирних будинків. Це включає онлайн-оплату комунальних послуг, подання заявок на ремонт,

отримання повідомлень від керуючої компанії та організацію заходів для мешканців;

– **Ізраїль**: В Ізраїлі платформа "Home365" використовує штучний інтелект для управління житловим фондом. Вона забезпечує автоматизацію обліку розрахунків, управління заявками на ремонт, аналітику даних та прогнозування потреб мешканців. Ця платформа допомагає зменшити витрати на обслуговування та підвищити якість послуг.

Звернення до міжнародного досвіду, такого як приклади з Німеччини, Швеції, Сінгапуру, США та Ізраїлю, показує, що впровадження цифрових платформ є ефективним способом вирішення проблем, пов'язаних з обліком та управлінням комунальними послугами. Ці приклади демонструють, як сучасні технології можуть підвищити якість життя мешканців багатоквартирних будинків та забезпечити більш ефективне управління житловим фондом.

1.5 Аналіз ситуацій прийняття рішень

Під тему "Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою Nest.JS" можна проаналізувати ситуації прийняття рішень за наступними аспектами:

Види ситуацій прийняття рішень:

– Ситуації закритих задач:

1) Вибір технологій і фреймворків для розробки платформи: У цьому випадку можна використати досвід та рекомендації експертів, а також аналізувати популярність та функціональні можливості різних технологій;
2) Вибір бази даних для зберігання інформації про розрахунки: Для розробки було обрано PostgreSQL.

– Ситуації відкритих задач:

1) Вибір дизайну та інтерфейсу платформи: У цьому випадку не існує однозначного правильного рішення, оскільки дизайн є

суб'єктивним. Тут можна провести дослідження з метою зрозуміти потреби та вподобання цільової аудиторії та провести тестування користувачів для визначення оптимального дизайну;

2) Вибір способу оплати та інтеграція платіжних систем: В залежності від потреб і вимог користувачів можна провести аналіз різних платіжних систем, таких як УкрКарт або рішення від Monobank, та вибрати ту, яка найкраще відповідає потребам проекту.

– Кризові ситуації:
  – Відмова сервера або проблеми з доступом до бази даних: У такій ситуації можна розглянути використання резервних серверів або реплікацію бази даних для забезпечення неперервної роботи платформи;
  – Кібератаки або витоки даних: У таких ситуаціях важливо прийняти швидкі та ефективні заходи для захисту конфіденційної інформації та запобігання подібним інцидентам. Можна використати заходи безпеки, такі як шифрування даних та захист від DDoS-атак.
– Ситуації за умов ризику:
  1) Вибір партнерів для співпраці: Оцінка ризиків, пов'язаних з потенційними партнерами, такими як надійність, досвід та репутація. Врахування ризику некомпетентності партнера або можливості невиконання угоди;
  2) Вибір маркетингових стратегій та рекламних каналів: Аналіз ризиків, пов'язаних з вибором певної стратегії або каналу реклами. Наприклад, ризик неефективності рекламної кампанії або низької конверсії.
– Ситуації за умов невизначеності:

1) Вибір цінової політики: Невизначеність щодо оптимального рівня цін на послуги, який задовольнятиме клієнтів та забезпечуватиме дохід для бізнесу. Проведення дослідження ринку та попиту, врахування конкуренції для прийняття рішення.

– Ситуації за умов нечітких цілей:

1) Вибір функціональності платформи: Різні точки зору щодо функцій, які повинна мати платформа для автоматизації обліку розрахунків. Наприклад, додавання функції онлайн-оплати або можливості відгуків користувачів. Врахування потреб та очікувань цільової аудиторії.

– Посилені відкриті рішення:

1) Вибір алгоритмів для обліку та фільтрації розрахунків: Розгляд різних алгоритмів, таких як алгоритм обліку з використанням геолокації або алгоритм рекомендацій на основі попередніх виборів користувачів. Врахування ефективності та точності алгоритмів при роботі з великим обсягом даних.

– Ситуації з вимогами до безпеки даних:

1) Збереження конфіденційності та захист особистої інформації: Врахування вимог до захисту даних та дотримання правил та норм, таких як Загальний регламент про захист даних (GDPR). Використання шифрування даних та інших заходів безпеки для забезпечення конфіденційності.

– Ситуації з вимогами до масштабованості:

1) Збільшення обсягу користувачів та транзакцій: Врахування можливості масштабування системи для забезпечення стабільної роботи при зростанні навантаження. Використання горизонтального масштабування, розділення навантаження на кілька серверів, а також кешування та інші оптимізації для покращення продуктивності.

– Ситуації з вимогами до мобільності:

1) Розробка мобільного додатку або адаптація веб-сайту для мобільних пристроїв: Мобільна доступність є ключовою для багатьох користувачів. Розгляд розробки окремого мобільного додатку або адаптації веб-сайту для роботи на різних типах пристроїв (респонсивний дизайн).

– Ситуації з вимогами до аналітики та звітності:

1) Збір та аналіз даних про користувачів та їх поведінку: Для вдосконалення платформи та покращення користувацького досвіду збір даних про взаємодію користувачів з платформою, їх вибори та перегляди. Використання аналітичних інструментів, таких як Google Analytics, та розробка звітів для аналізу та прийняття відповідних рішень.

1.6 Методологія та підходи до розробки цифрової платформи

Методології розробки програмного забезпечення є важливим інструментом для забезпечення успішного виконання проектів. Вони допомагають структурувати процес розробки, забезпечуючи чітке розуміння вимог, ефективне проектування, розробку, тестування та впровадження системи. Використання методологій дозволяє зменшити ризики, підвищити якість продукту та забезпечити задоволення потреб користувачів.

Нижче наведено основні етапи розробки програмного забезпечення, які були використані при розробці платформи і як вони сприяють кінцевому результату:

– Аналіз вимог:

1) Визначення функціональних вимог: Це включає в себе опис функцій, які повинна виконувати система. Наприклад, які дії користувач може виконувати, які дані вводити та отримувати;

2) Визначення нефункціональних вимог: Це вимоги, які визначають, як система повинна працювати. Вони включають

продуктивність, безпеку, масштабованість, надійність та інші характеристики;

3) Документування вимог: Створення детальної документації, яка описує всі вимоги до системи. Це допоможе забезпечити, що всі учасники проекту мають однакове розуміння вимог.

– Архітектурний дизайн:

1) Вибір технологій: Визначення технологій, які будуть використовуватися для розробки системи. Це може включати мови програмування, фреймворки, бази даних та інші інструменти;
2) Структура бази даних: Проектування структури бази даних, включаючи таблиці, зв'язки між ними та індекси. Це забезпечить ефективне зберігання та доступ до даних;
3) Моделі даних: Створення моделей даних, які відображають структуру даних у системі. Це допоможе зрозуміти, як дані будуть зберігатися та оброблятися;
4) Інтеграція з іншими системами: Визначення способів інтеграції системи з іншими системами та сервісами. Це може включати використання API, веб-сервісів та інших методів.

– Розробка MVP:

1) Створення MVP (Мінімально життєздатний продукт): Розробка початкової версії системи, яка включає основні функції. Це дозволить перевірити, чи відповідає система вимогам[27];
2) Тестування прототипу: Проведення тестування прототипу для виявлення можливих проблем та недоліків. Це допоможе виправити їх на ранніх етапах розробки[28];
3) Отримання зворотного зв'язку: Збір зворотного зв'язку від користувачів, які тестують прототип. Це допоможе зрозуміти, чи задовольняє система потреби користувачів.

– Тестування та валідація:
    1) Функціональне тестування: Перевірка, чи працюють всі функції системи відповідно до вимог[28];
    2) Інтеграційне тестування: Перевірка взаємодії між різними компонентами системи. Це допоможе виявити проблеми, які можуть виникнути при інтеграції[28];
    3) Навантажувальне тестування: Перевірка, як система працює під високим навантаженням. Це допоможе визначити, чи здатна система обробляти велику кількість користувачів та даних[28];
    4) Валідація системи: Перевірка, чи відповідає система всім вимогам. Це включає перевірку функціональних та нефункціональних вимог.
– Впровадження та підтримка:
    1) Впровадження системи: Введення системи в експлуатацію. Це включає встановлення системи на сервери, налаштування та запуск;
    2) Навчання користувачів: Проведення навчання для користувачів, щоб вони могли ефективно використовувати систему;
    3) Технічна підтримка: Надання технічної підтримки користувачам. Це включає вирішення проблем, які можуть виникнути під час використання системи;
    4) Оновлення та вдосконалення: Регулярне оновлення системи на основі зворотного зв'язку від користувачів. Це допоможе забезпечити, що система залишається актуальною та відповідає потребам користувачів.

1.7 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Аналіз літературних джерел та практичного досвіду використання інформаційних систем (ІС) і технологій в предметній галузі автоматизації обліку

розрахунків у багатоквартирних будинках за допомогою Nest.JS допоможе зрозуміти сучасні тенденції, найкращі практики та інструменти, що можуть бути застосовані в цьому проекті.

Теорії та концепції:

- Цифровізація управління житловим фондом: Цифровізація є ключовим трендом у сучасному управлінні житловим фондом. Вона включає використання інформаційних систем для автоматизації процесів обліку, управління та контролю. Цифрові платформи дозволяють зменшити витрати, підвищити ефективність управління та забезпечити прозорість фінансових операцій;
- Автоматизація обліку розрахунків: Автоматизація обліку розрахунків у багатоквартирних будинках дозволяє зменшити кількість помилок, пов'язаних з ручним введенням даних, та забезпечити точність і своєчасність розрахунків. Вона також сприяє зменшенню витрат на обслуговування та підвищенню якості послуг для мешканців;
- Інтеграція з іншими системами: Сучасні цифрові платформи можуть інтегруватися з іншими інформаційними системами, що дозволяє створити єдину інформаційну базу для управління житловим фондом. Це забезпечує прозорість, контроль та безпеку фінансових операцій.

Критичний огляд наукових джерел:

- Дослідження цифрових платформ: У дослідженнях зазначається, що цифрові платформи для управління житловим фондом дозволяють значно підвищити ефективність управління та зменшити витрати. Наприклад, платформа "Наш Дім" в Україні сприяє зменшенню корупційних ризиків та мінімізації впливу людського чинника;
- Практичний досвід використання: Практичний досвід використання цифрових платформ показує, що вони дозволяють зберігати документацію у захищеному вигляді, спрощують процедуру

управління будинками та впроваджують онлайн-голосування для мешканців.

На основі проведеного аналізу літературних джерел можна зробити висновок, що цифрові платформи для автоматизації обліку розрахунків у багатоквартирних будинках є ефективним інструментом для підвищення ефективності управління житловим фондом. Вони дозволяють зменшити витрати, забезпечити прозорість фінансових операцій та підвищити якість послуг для мешканців. Однак, важливо враховувати можливі технічні збої та проблеми з безпекою даних, які можуть вплинути на точність та своєчасність розрахунків.

Проблемні питання:

– Як забезпечити безпеку даних у цифрових платформах для автоматизації обліку розрахунків?;
– Які методи можна використовувати для інтеграції цифрових платформ з іншими інформаційними системами?;
– Як мінімізувати вплив технічних збоїв на роботу цифрових платформ?;
– Які заходи можна вжити для підвищення доступності цифрових платформ для всіх мешканців?.

Ці питання потребують подальшого дослідження та обговорення, щоб забезпечити успішне впровадження та використання цифрових платформ для автоматизації обліку розрахунків у багатоквартирних будинках.

1.8 Виклики та ризики цифровізації

Цифровізація у сфері житлово-комунального господарства (ЖКГ) несе в собі низку викликів та ризиків, які необхідно враховувати для забезпечення ефективного впровадження та функціонування цифрових систем. Серед основних викликів можна виділити кібербезпеку, технічні проблеми та необхідність у навчанні персоналу.

Кібербезпека є одним з найбільш критичних аспектів цифровізації. З розвитком цифрових технологій, зростає кількість кіберзагроз, що можуть поставити під удар конфіденційність та цілісність даних. Системи ЖКГ можуть

стати об'єктами атак з боку хакерів, що намагаються отримати доступ до особистих даних мешканців або до критично важливої інфраструктури. Для запобігання таким загрозам необхідно впроваджувати сучасні засоби захисту інформації, такі як шифрування, багатофакторна аутентифікація та постійне моніторинг систем на наявність вразливостей[26].

Технічні проблеми також можуть стати значним викликом під час впровадження цифрових систем у сфері ЖКГ. Складні програмні забезпечення та інтеграція нових технологій з існуючими інфраструктурами можуть викликати технічні збої та проблеми з сумісністю. Це може призвести до непередбачуваних перебоїв у роботі систем, що може негативно вплинути на якість обслуговування мешканців. Щоб уникнути таких проблем, важливо проводити детальне планування та тестування систем до їх впровадження, а також мати в наявності резервні плани на випадок збоїв.

Ще одним важливим аспектом є необхідність навчання персоналу**.** Впровадження нових цифрових технологій вимагає наявності кваліфікованих фахівців, здатних обслуговувати та підтримувати ці системи. Недостатня підготовка персоналу може призвести до неправильного використання технологій та виникнення технічних проблем. Для запобігання цьому необхідно інвестувати у навчання та професійний розвиток працівників, створюючи умови для підвищення їхньої кваліфікації та ознайомлення з новітніми технологіями.

Крім зазначених аспектів, варто також враховувати ризики, пов'язані з фінансовими витратами **та** управлінням змінами**.** Цифровізація вимагає значних інвестицій, які можуть бути непосильними для деяких організацій ЖКГ, особливо у невеликих містах чи селищах. Важливо ретельно оцінювати витрати та визначати джерела фінансування перед початком впровадження цифрових систем.

Управління змінами також є важливим фактором, оскільки впровадження нових технологій може зустріти опір з боку співробітників та мешканців. Комунікація з усіма зацікавленими сторонами, залучення їх до процесу змін та

забезпечення розуміння переваг цифровізації можуть сприяти успішній інтеграції нових систем.

Таким чином, впровадження цифрових систем у сфері ЖКГ супроводжується низкою викликів та ризиків, які необхідно враховувати та ретельно планувати. Забезпечення кібербезпеки, подолання технічних проблем, навчання персоналу, фінансові витрати та управління змінами є ключовими аспектами, від яких залежить успішність цифровізації. Приділяючи увагу цим питанням, можна створити ефективні та безпечні цифрові системи, що забезпечать покращення якості обслуговування мешканців та підвищення ефективності управління ресурсами ЖКГ.

Окрім викликів, пов'язаних з кібербезпекою, технічними проблемами та необхідністю навчання персоналу, важливо враховувати й соціально-психологічні аспекти впровадження цифрових платформ у житлово-комунальне господарство (ЖКГ). Одним із ключових ризиків є небажання мешканців переходити на нові системи обліку розрахунків, що часто пов'язано з недовірою до цифрових технологій або недостатнім розумінням їх переваг.

Соціально-психологічний опір є природною реакцією на зміни, особливо у тих груп населення, які мають обмежений досвід користування цифровими інструментами. Згідно з дослідженнями, значна частина мешканців багатоквартирних будинків, особливо літнього віку, відчувають складнощі при використанні електронних систем для розрахунків та обліку комунальних послуг. Для подолання цього виклику важливо проводити інформаційні кампанії, які детально пояснюють, як працюють цифрові платформи, а також надають підтримку користувачам на початкових етапах їх впровадження. Важливим кроком може стати створення гарячих ліній підтримки або центрів допомоги, де мешканці зможуть отримати відповіді на свої запитання та практичну допомогу.

Проблеми з доступністю інтернету також є важливим викликом, особливо в невеликих містах та сільських регіонах, де якість мережевої інфраструктури може бути недостатньою для повноцінного функціонування цифрових платформ.

Недоступність швидкого інтернету може призвести до проблем з обробкою даних, виникнення затримок у роботі систем та загального зниження ефективності платформи. Для вирішення цієї проблеми необхідно співпрацювати з провайдерами інтернет-послуг з метою забезпечення стабільного зв'язку, а також створювати офлайн-режими для цифрових платформ, щоб мешканці могли взаємодіяти з ними навіть за відсутності доступу до мережі.

Юридичні та регуляторні ризики також слід брати до уваги під час цифровізації сфери ЖКГ. Законодавство, яке регулює роботу цифрових систем, може відрізнятися в різних регіонах, що створює додаткові труднощі для розробників та впроваджувачів таких платформ. Наприклад, вимоги щодо зберігання персональних даних мешканців чи обробки платежів повинні відповідати національним та міжнародним стандартам, таким як GDPR. Недотримання регуляторних норм може призвести до штрафних санкцій або зупинки роботи систем. Для мінімізації цих ризиків необхідно залучати фахівців з юридичних питань на етапі розробки та впровадження цифрових рішень.

Крім того, важливо враховувати економічну доцільність впровадження цифрових систем**.** Незважаючи на довгострокові переваги цифровізації, такі як зниження витрат на облік розрахунків, оптимізація ресурсів та підвищення прозорості, початкові витрати на розробку та впровадження платформи можуть бути значними. Це створює ризик перевищення бюджету або затримок у реалізації проекту, особливо для підприємств ЖКГ з обмеженим фінансуванням. У цьому контексті важливо проводити детальний аналіз вартості впровадження, оцінювати потенційну окупність інвестицій та шукати джерела додаткового фінансування, такі як гранти чи партнерські програми з приватним сектором.

Стійкість та надійність цифрових платформ також відіграє ключову роль у їх ефективному функціонуванні. Цифрові системи повинні мати високу стійкість до збоїв, технічних проблем та зовнішніх загроз. Ненадійна робота платформ може призвести до втрати даних, помилок у розрахунках або навіть повного припинення обслуговування мешканців, що негативно вплине на рівень довіри до системи. Для

забезпечення стійкості необхідно використовувати сучасні технології резервного копіювання даних, впроваджувати стратегії відновлення після збоїв (disaster recovery plans) та регулярно проводити аудит систем на предмет їх надійності.

Таким чином, цифровізація у сфері ЖКГ є складним процесом, який вимагає системного підходу до вирішення численних викликів та ризиків. Успішне впровадження цифрових платформ, зокрема на базі **NestJS**, залежить від забезпечення кібербезпеки, вирішення технічних та соціальних проблем, юридичної відповідності, фінансової стійкості та надійності роботи систем. Приділяючи увагу цим аспектам, можна досягти значного підвищення ефективності управління житлово-комунальним господарством та створити зручні умови для мешканців багатоквартирних будинків.

1.9 Економічний ефект від автоматизації обліків

Автоматизація обліків[2] у сфері житлово-комунального господарства (ЖКГ) є важливим кроком для підвищення ефективності управління, оптимізації фінансових процесів та зменшення витрат як для компаній, що надають послуги, так і для мешканців багатоквартирних будинків. Сучасні цифрові платформи дозволяють досягти суттєвого економічного ефекту за рахунок оптимізації ресурсів, зниження операційних витрат та підвищення прозорості фінансових операцій. У цьому підрозділі розглядаються ключові аспекти та показники економічного ефекту від впровадження автоматизованих систем обліку, зокрема на основі цифрових платформ, таких як **NestJS**.

1.9.1 Зниження операційних витрат

Одним із основних економічних ефектів автоматизації є зниження витрат на ведення обліку та управління фінансами. Традиційні методи обліку вимагають значних трудових ресурсів, що включають оплату праці бухгалтерів, інженерів та інших фахівців, а також витрати на паперовий документообіг. Автоматизовані системи дозволяють значно скоротити ці витрати завдяки:

- Мінімізації ручної праці: Процес обробки рахунків, нарахування платежів та відстеження заборгованостей відбувається автоматично, що знижує навантаження на працівників.
- Скороченню витрат на паперовий документообіг**:** Всі операції переводяться в електронний формат, що дозволяє зменшити витрати на друк, зберігання та доставку документів.
- Оптимізації використання часу: Завдяки автоматизації облікові завдання виконуються швидше та з мінімальною кількістю помилок.

За попередніми дослідженнями, впровадження цифрових систем обліку дозволяє скоротити операційні витрати підприємств ЖКГ на **15-25%** протягом першого року після впровадження.

1.9.2 Підвищення прозорості та контроль за фінансовими операціями

Автоматизація забезпечує повну прозорість облікових процесів, що є важливим фактором для підвищення довіри мешканців до надавачів комунальних послуг. Зокрема:

- Прозорість нарахувань та платежів: Кожен мешканець має доступ до особистого кабінету, де відображаються детальні розрахунки за комунальні послуги. Це зменшує кількість суперечок і помилок у платіжних документах;
- Контроль за заборгованістю: Система автоматично генерує звіти про заборгованості мешканців, надсилає нагадування про необхідність оплати, що дозволяє знизити рівень заборгованості;
- Аудит та моніторинг фінансових операцій: Впровадження автоматизованих платформ дозволяє керівництву компаній у реальному часі моніторити фінансовий стан та звітувати про ефективність використання ресурсів.

Таким чином, прозорість та контроль сприяють підвищенню фінансової дисципліни як серед мешканців, так і серед працівників, що призводить до економії коштів та більш ефективного управління бюджетами.

1.9.3 Скорочення заборгованості та покращення фінансової дисципліни

Заборгованість за комунальні послуги є однією з головних проблем у сфері ЖКГ. Впровадження автоматизованих систем дозволяє суттєво знизити рівень заборгованості завдяки:

- Своєчасному інформуванню: Автоматизовані платформи забезпечують регулярне надсилання нагадувань про оплату через SMS, електронну пошту або мобільні додатки;
- Гнучким методам оплати: Інтеграція платіжних систем дозволяє мешканцям швидко та зручно сплачувати рахунки онлайн, що підвищує ймовірність своєчасної оплати;
- Автоматичному нарахуванню пені: У разі прострочення платежів система може автоматично нараховувати штрафи, що стимулює мешканців виконувати свої фінансові зобов’язання вчасно.

Досвід впровадження подібних систем показує, що рівень заборгованості може бути знижено на **20-30%** протягом перших двох років після автоматизації процесів обліку.

1.9.4 Оптимізація ресурсів та підвищення продуктивності

Автоматизація дозволяє оптимізувати використання ресурсів, як людських, так і фінансових. У результаті:

- Підвищується продуктивність працівників: Замість виконання рутинних завдань, персонал може зосередитися на стратегічних завданнях та підвищенні якості послуг;
- Оптимізується використання енергетичних ресурсів: Цифрові системи обліку дозволяють аналізувати споживання води, газу та електроенергії, виявляти аномалії та оптимізувати витрати;
- Зменшуються втрати ресурсів: Автоматизовані системи дозволяють контролювати витрати ресурсів та оперативно реагувати на можливі втрати або аварійні ситуації.

Таким чином, автоматизація сприяє підвищенню ефективності роботи підприємств ЖКГ та оптимізації їх ресурсів.

1.9.5 Економія для мешканців

Важливим аспектом економічного ефекту автоматизації є пряма економія для мешканців багатоквартирних будинків:

- Точний облік спожитих ресурсів: Використання автоматизованих систем дозволяє забезпечити точний облік споживання ресурсів, що запобігає помилкам у нарахуваннях та завищенню рахунків;
- Зниження витрат на адміністративні послуги: Автоматизація процесів дозволяє зменшити витрати компаній ЖКГ, що у перспективі призводить до зниження тарифів для кінцевих споживачів;
- Зручність оплати та економія часу: Мешканці можуть швидко здійснювати платежі онлайн, уникати штрафів за прострочення та економити свій час.

1.9.6 Показники економічного ефекту

Для оцінки економічного ефекту від автоматизації обліків можуть використовуватися такі показники:

- зменшення операційних витрат (у %);
- скорочення заборгованості (у %);
- підвищення продуктивності праці (у кількості оброблених заявок на одного працівника);
- скорочення часу на обробку платежів та розрахунків (у год/місяць);
- зниження втрат ресурсів (у %).

Таким чином, впровадження автоматизованих систем обліку у сфері ЖКГ створює значний економічний ефект, що проявляється у зниженні витрат, підвищенні продуктивності працівників, оптимізації ресурсів та економії коштів для мешканців. Враховуючи ці переваги, цифровізація є необхідним кроком для

розвитку сучасного житлово-комунального господарства та забезпечення його сталого функціонування.

1.10 Прогнози розвитку технологій у ЖКГ

Сфера житлово-комунального господарства (ЖКГ) є однією з ключових для забезпечення комфортного життя населення та ефективного управління ресурсами. Проте традиційні методи управління в цій галузі часто характеризуються неефективністю, застарілою інфраструктурою та високими витратами. У контексті сучасної цифровізації та розвитку інноваційних технологій, очікується, що найближчі роки принесуть суттєві зміни у функціонуванні систем ЖКГ. Цей підрозділ представить прогноз розвитку технологій у ЖКГ, визначить ключові тренди та їх можливий вплив на галузь.

Одним із найпотужніших трендів є розвиток концепції Smart City (Розумного міста)[24], яка передбачає інтеграцію цифрових технологій для автоматизації та оптимізації управління міськими ресурсами. Основні технологічні рішення у цьому напрямку включають:

- Інтернет речей (IoT): Смарт-сенсори та лічильники, що автоматично збирають та передають дані про споживання ресурсів (електроенергії, води, газу)[25]. Це дозволяє оперативно виявляти втрати та аномалії, а також оптимізувати використання ресурсів;
- Централізовані системи моніторингу: Єдині платформи для моніторингу інфраструктури міста в реальному часі. Вони об’єднують дані з усіх житлових будинків, лічильників і сенсорів для швидкого прийняття рішень;
- Автоматизація управління комунальними послугами: Завдяки смарт-технологіям мешканці зможуть контролювати споживання ресурсів через мобільні додатки та керувати енергоспоживанням у режимі реального часу.

Прогнозується, що вже до 2030 року до 60% великих міст світу запровадять елементи Smart City, що призведе до суттєвого підвищення енергоефективності та скорочення операційних витрат у секторі ЖКГ.

Використання штучного інтелекту та аналітики великих даних (Big Data) у ЖКГ є наступним важливим кроком до підвищення ефективності управління:

- Прогнозування споживання ресурсів: Алгоритми штучного інтелекту аналізуватимуть історичні дані про споживання ресурсів і прогнозуватимуть їхні витрати з урахуванням сезонних, погодних та соціальних факторів. Це дозволить уникати перевантаження систем та оптимізувати їх роботу;
- Автоматичне виявлення проблем: Системи на основі AI зможуть виявляти витоки води, несправності в мережах або несанкціоноване підключення, що знизить втрати ресурсів;
- Оптимізація витрат на обслуговування: Розумні алгоритми допоможуть планувати ремонтні роботи та технічне обслуговування інфраструктури, що дозволить зменшити витрати на аварійні ситуації.

Очікується, що застосування AI дозволить підвищити ефективність управління комунальними ресурсами на 30-40% у найближчі 5-10 років.

Сектор ЖКГ відіграє ключову роль у зниженні викидів вуглецю та переході до зеленої енергетики. Основні інновації у цій сфері включають:

- Смарт-мережі (Smart Grids): Інтеграція розподілених джерел енергії (сонячних панелей, вітрових установок) та розумних систем розподілу дозволить оптимізувати споживання електроенергії;
- Енергоефективні будівлі: Використання сучасних технологій утеплення, систем рекуперації тепла та енергоощадних матеріалів значно зменшить витрати на опалення та кондиціонування;
- Автоматизоване управління енергією: Системи енергоефективного керування (BMS) автоматично регулюватимуть освітлення, опалення та вентиляцію в залежності від потреб.

За оцінками, до 2040 року інновації у сфері енергоефективності дозволять скоротити споживання енергії в ЖКГ на 40-50%.

Майбутнє ЖКГ також пов’язане з удосконаленням платіжних систем і мобільних додатків:

- Цифрові гаманці та онлайн-оплата: Повна інтеграція онлайн-платежів спростить процедуру оплати комунальних послуг;
- Інтеграція з мобільними додатками: Сучасні додатки дозволять мешканцям відслідковувати споживання ресурсів, управляти послугами та здійснювати оплату в один клік;
- Мікроплатежі та передплати: Нові платіжні моделі, як-от системи передплати на певний обсяг послуг або «плати за фактичне споживання», дозволять гнучкіше підходити до розрахунків;

Розвиток технологій у ЖКГ вплине не лише на економічні показники, а й на соціальну сферу:

- Зниження витрат для мешканців: Оптимізація витрат на управління ресурсами дозволить зменшити тарифи на комунальні послуги.
- Підвищення якості життя: Автоматизація процесів забезпечить комфорт, оперативність у вирішенні проблем та кращий доступ до інформації.
- Створення нових робочих місць: Розвиток технологій потребуватиме фахівців з IT, енергетики та інноваційних систем управління.

Отже, технології у сфері ЖКГ розвиватимуться в напрямку автоматизації, цифровізації, енергоефективності та прозорості. Впровадження Smart City**,** IoT, штучного інтелекту та блокчейну дозволить оптимізувати ресурси, знизити витрати та покращити якість послуг для населення. Ці зміни створять умови для сталого розвитку міської інфраструктури, підвищать комфорт життя та сприятимуть ефективному управлінню ЖКГ у майбутньому.

РОЗДІЛ 2 АНАЛІЗ І ОЦІНКА ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ ЦИФРОВИХ ПЛАТФОРМ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ РОЗРАХУНКІВ

2.1 Аналіз зібраної інформації

Цей розділ є невід’ємною частиною практичної діяльності, яка спрямована на глибоке вивчення та аналіз обраної тематики. Мета цього розділу полягає у систематизації, узагальненні та інтерпретації зібраної інформації, а також у розробці пропозицій і рекомендацій, які сприятимуть вирішенню визначених проблем або вдосконаленню існуючих процесів. Зібрані дані та проведений аналіз дозволяють більш чітко зрозуміти ключові аспекти досліджуваної теми, що, своєю чергою, формує основу для подальших наукових чи практичних розробок.

Практика, яка проходила в компанії, що спеціалізується на обслуговуванні багатоквартирних будинків, стала важливим етапом у глибшому розумінні досліджуваної сфери. Під час проходження практики вдалося отримати унікальну можливість ознайомитися з роботою житлово-комунального підприємства (ЖКП) зсередини, зрозуміти його структуру, організаційні процеси, а також основні виклики, з якими стикається підприємство при обслуговуванні багатоквартирних будинків.

Роль багатоквартирних будинків у досліджуваній сфері

Багатоквартирні будинки є ключовою частиною міської інфраструктури та потребують постійного обслуговування, яке включає широкий спектр послуг: забезпечення водопостачанням і водовідведенням, теплопостачанням, електроенергією, утримання спільних приміщень (під'їзди, дахи, ліфти), прибирання територій та багато іншого. Практика в компанії, яка займається обслуговуванням цих будинків, надала можливість зібрати вичерпну інформацію про механізми управління, комунікацію з мешканцями, облік платежів і вирішення проблем, пов'язаних із заборгованостями.

Одна з найважливіших сфер, що була охоплена під час практики, – це фінансовий облік та система розрахунків за комунальні послуги. Зокрема, вдалося

зрозуміти, як формується фінансова модель обслуговування будинків, які труднощі виникають через заборгованості, а також як впровадження автоматизованих систем обліку могло б покращити ситуацію.

Інформація, зібрана під час практики, була отримана з різних джерел, які можна умовно розділити на три основні категорії:

– Офіційна документація підприємства:
    1) фінансові звіти, які демонструють структуру доходів і витрат;
    2) документи, що регламентують процеси обслуговування будинків (контракти з мешканцями, тарифи на послуги, графіки ремонтів тощо);
    3) статистичні дані щодо заборгованостей мешканців.
– Спостереження за роботою підприємства:
    1) безпосередня участь у процесах, пов'язаних із нарахуванням оплат і розсилкою квитанцій;
    2) ознайомлення з роботою відділу, відповідального за зв'язок із мешканцями;
    3) аналіз використання застарілих програм обліку, які суттєво впливають на ефективність роботи підприємства.
– Комунікація з мешканцями та персоналом:
    1) опитування співробітників компанії, які щоденно взаємодіють із мешканцями;
    2) спілкування з мешканцями, які розповіли про свої враження та проблеми, пов’язані з оплатою послуг та обслуговуванням.

Завдяки такому комплексному підходу вдалося не лише зібрати багато інформації, але й отримати глибше уявлення про проблеми, які виникають у взаємодії між підприємством і мешканцями багатоквартирних будинків.

Основні аспекти, охоплені під час практики

– Система обліку платежів. Одна з ключових проблем, виявлених під час практики, – це неефективність системи обліку платежів. У багатьох

випадках використовуються застарілі локальні програми, які не мають інтеграції з банківськими системами. Через це інформація про оплату надходить із запізненням, а в деяких випадках взагалі губиться. Це створює недовіру з боку мешканців, які можуть отримувати помилкові повідомлення про заборгованість, хоча оплата була здійснена вчасно.

Наприклад, у документах підприємства були виявлені численні випадки, коли інформація про оплату надходила із затримкою у 5-7 днів. Це спричиняло конфлікти між мешканцями та підприємством, оскільки останні отримували повідомлення про борги навіть після здійснення оплати.

– Заборгованості мешканців. Аналіз статистики заборгованостей показав, що близько 20% мешканців багатоквартирних будинків мають борги за комунальні послуги. Основними причинами цього є:
  1) низький рівень автоматизації нагадувань про строки оплати;
  2) складність розрахунків у квитанціях, яка викликає нерозуміння з боку мешканців;
  3) технічні помилки через ручне внесення даних.

Практика дозволила побачити, як саме застарілі підходи до ведення обліку впливають на накопичення заборгованостей і загальну фінансову стабільність підприємства.

– Комунікація з мешканцями. Інший важливий аспект, який вдалося дослідити, – це рівень комунікації з мешканцями. Було з’ясовано, що більшість скарг пов’язана з недостатньо прозорою системою нарахувань та відсутністю зручних каналів зв'язку. Наприклад:
  1) мешканці скаржаться на неможливість отримати оперативну інформацію про стан своїх платежів;
  2) підприємство часто використовує традиційні методи комунікації (наприклад, друковані квитанції або телефонні дзвінки), які є застарілими й неефективними.

– Використання технологій. Одним із важливих висновків стало розуміння того, як сучасні технології можуть допомогти покращити роботу підприємства. Наприклад, впровадження автоматизованих систем обліку, які мають інтеграцію з банківськими системами, дозволило б:
    1) зменшити кількість помилкових повідомлень про заборгованості;
    2) автоматично надсилати нагадування мешканцям про строки оплати;
    3) підвищити прозорість нарахувань і довіру до системи.

Практика допомогла не лише отримати нові знання, але й краще зрозуміти функціонування компанії, яка обслуговує багатоквартирні будинки. Зібрана інформація стала основою для подальшого аналізу та розробки рекомендацій, спрямованих на вдосконалення процесів. Зокрема, вдалося ідентифікувати ключові проблеми, які заважають ефективному управлінню багатоквартирними будинками, і визначити шляхи їх вирішення.

2.2 Аналіз існуючих рішень

Аналіз цифрових платформ для автоматизації обліку розрахунків у багатоквартирних будинках допоміг виявити ключові аспекти, необхідні для створення ефективного і сучасного рішення, яке відповідає потребам користувачів. У цій роботі було здійснено детальний аналіз сучасних підходів, функціональності, дизайну та технологій, які використовуються в розробці подібних платформ.

Користувацький досвід (UX) і дизайн. Одним із ключових висновків було те, що зручність користувацького досвіду є одним із визначальних факторів успіху цифрової платформи для обліку розрахунків. Важливо забезпечити інтуїтивно зрозумілий і простий інтерфейс, який дозволяє мешканцям багатоквартирних будинків легко отримувати доступ до важливої інформації, таких як стан рахунків, платежі та звіти про використання ресурсів.

Основні аспекти, які були враховані:

- Адаптивність дизайну: Веб-платформа повинна бути доступною на різних пристроях – від настільних комп'ютерів до смартфонів. Адаптивний дизайн дозволяє забезпечити зручність використання незалежно від розміру екрана;
- Мінімалістичний підхід: Використання мінімалістичного дизайну допомагає сфокусувати увагу користувачів на головному функціоналі платформи. Простий і елегантний вигляд створює позитивне враження і підвищує довіру до платформи;
- Інтуїтивна навігація: Логічно організована структура платформи дозволяє користувачам швидко знаходити потрібну інформацію. Наприклад, функціонал для перегляду балансу, сплати рахунків або перегляду звітів має бути доступний у кілька кліків;
- Використання мультимедійних елементів: Графіки, діаграми, інтерактивні звіти надають користувачам можливість отримувати дані в зрозумілому та наочному форматі;
- Функціональність цифрових платформ.

Для автоматизації обліку розрахунків важливо впроваджувати функції, які забезпечують повний спектр послуг, необхідних для управління багатоквартирними будинками[18].

Основні функції, які були визначені під час аналізу:

- Модуль обліку платежів: Ця функція дозволяє мешканцям здійснювати оплату рахунків онлайн через інтегровані платіжні сервіси, а також отримувати повідомлення про стан рахунку чи наявність заборгованості;
- Звітування: Забезпечення доступу до звітів про використання коштів, споживання енергоносіїв чи інших послуг будинку є важливим для прозорості;

- Система сповіщень**:** Автоматичні нагадування про терміни оплати, повідомлення про збори мешканців чи аварійні ситуації дозволяють підвищити рівень комунікації;
- Персоналізація**:** Користувачі можуть отримувати налаштовані рекомендації, доступ до їхніх персональних даних (наприклад, історії оплат) і використовувати фільтри для швидкого пошуку інформації;
- Підтримка клієнтів: Інтеграція чат-ботів або систем зворотного зв’язку для вирішення запитів мешканців.

Достовірність та безпека: Користувачі повинні мати довіру до веб-сайту, тому важливо забезпечити безпеку та захист особистих даних. Необхідно використовувати захищене з'єднання (SSL), реалізувати механізми перевірки достовірності об'єктів проживання, забезпечити чітку політику конфіденційності та врахувати вимоги до захисту даних.

Мобільна сумісність: З урахуванням зростаючого використання мобільних пристроїв, веб-сайт повинен бути адаптивним та оптимізованим для різних пристроїв і платформ. Він повинен забезпечувати зручний користувацький досвід як на комп'ютерах, так і на смартфонах та планшетах.

Відгуки та оцінки: Можливість залишати відгуки та оцінки про сервіс дозволяє користувачам отримати більше інформації про якість та задоволення попередніх гостей. Це допомагає збільшити довіру до веб-сайту та об'єктів проживання.

При аналізі було виведено те, що в сучасній розробці дуже важливо мати приємний і зручний дизайн, сумісний з усіма пристроями, бо правильний дизайн буде тримати нових користувачів на сайті значно довше, тому в ході розробки було проведено аналіз конкурентів і прийняття рішення щодо найкращого дизайну.

Веб-дизайн та користувацький досвід (User Experience, UX) важливі аспекти при розробці веб-сайту для обліку розрахунків, оскільки вони впливають на сприйняття, зручність використання та задоволення користувачів. Нижче наведено

кілька прикладів сучасних тенденцій у веб-дизайні, що можуть бути використані для поліпшення користувацького досвіду під час під час використання сервісу:

Мінімалістичний дизайн: Стилізований та простий дизайн з мінімальною кількістю елементів сприяє легкому сприйняттю і фокусуванню на важливій інформації. Чисті простори, прості форми та обмежене використання кольорів створюють сучасний і елегантний вигляд.

Адаптивний дизайн: Врахування різних пристроїв та платформ (настільні комп'ютери, планшети, смартфони) і надання оптимального користувацького досвіду на кожному з них. Резинові макети, гнучкі шаблони та адаптивні зображення дозволяють веб-сайту адаптуватися до різних розмірів екранів.

Інтуїтивний навігаційний дизайн: Логічна та зрозуміла структура навігації, яка допомагає користувачам швидко зорієнтуватися на сайті та знайти необхідну інформацію. Зручне розміщення меню, логотипу та пошукової функції забезпечує зручний доступ до різних розділів веб-сайту.

Мультимедійні елементи: Використання відео, анімації та інших мультимедійних елементів може зробити веб-сайт більш цікавим та привабливим для користувачів. Наприклад, відеоогляди останніх оновлень, галереї зображень з можливістю збільшення, візуалізація тощо.

Ці приклади тенденцій у веб-дизайні можна застосувати для створення зручного веб-сайту для обліку розрахунків, як для звичайного мешканця, так і для менеджера управляючої компанії, що покращить користувацький досвід та задоволення від використання системи.

Іншим висновком було те, що швидкодія і коректна робота сайту при різних ситуаціях є дуже важливою складовою, адже повільний сайт може відвернути користувачів. Саме тому в даному випадку було використано наступні технології:

Аналіз технологій для розробки платформи

Для розробки сучасної цифрової платформи було обрано стек технологій, які забезпечують високу продуктивність, гнучкість і масштабованість. Особливу увагу

приділено використанню **Nest.JS** – прогресивного фреймворка для серверної розробки.

Використані технології:

– Nest.JS для бекенду:
  1) Це фреймворк на основі TypeScript, який забезпечує структуру додатка, модульність і гнучкість;
  2) Його інтеграція з Prisma ORM дозволяє легко працювати з базами даних і забезпечує високу швидкість обробки запитів;
  3) Фреймворк сприяє впровадженню концепцій мікросервісів, що дозволяє масштабувати систему.
– TypeScript:
  1) Використання статичної типізації сприяє зниженню кількості помилок і підвищує якість коду.
  2) Інструмент дозволяє команді швидше розробляти функціонал і легше підтримувати його в майбутньому.
– Next.js для фронтенду:
  1) Забезпечує інтерактивність і динамічний інтерфейс платформи.
  2) Фреймворк дозволяє реалізувати складні UI-компоненти для зручного відображення фінансових даних.
  3) Фреймворк допомагає створювати швидкі веб-додатки
– PostgreSQL: Як реляційна база даних використовується PostgreSQL, яка забезпечує високу продуктивність і безпеку даних.
– Tailwind CSS: Фреймворк для швидкої розробки сучасного дизайну, що дозволяє створити адаптивний і естетичний інтерфейс.
– Модулі безпеки: Інтеграція захищених механізмів авторизації та аутентифікації, таких як JWT (JSON Web Tokens), гарантує безпечний доступ до системи.

Ці технології були використані з метою створення сучасного, швидкого та зручного веб-сайту з бронювання житла. Вони сприяють розширюваності,

підтримці високої продуктивності та забезпечують комфортну розробку та підтримку веб-сайту.

2.2.1 Наявні рішення в Україні

Для дослідження інформаційних систем в предметній області було обрано:

- Мій дім онлайн;

Порівняння ІС конкурентів наведено в таблиці 1.1.1

Мій дім онлайн[17] – Українська компанія-розробник програмного комплексу для ведення побудинкового обліку та комунікації з мешканцями. Програма створена для ОСББ, управляючої компанії, комунального підприємства, садового товариства чи котеджного містечка.

Бухгалтерам та керівництву обслуговуючих організацій Мій Дім Онлайн допомагає автоматизувати абонентський облік, формувати квитанції, розподіляти оплат мешканців та спілкуватися з ними.

Жителям Мій Дім Онлайн надає у користування Особистий кабінет, мобільний додаток мій Дім Онлайн та Телеграм бот, за допомогою яких можна отримувати квитанції, оплачувати платіжки, передавати показники лічильників та писати звернення до правління.

Ключові функції платформи.

Трохи нижче на головній сторінці знаходяться інформаційні блоки, які детально описують основні функції платформи. Вони включають:

- Автоматизацію обліку розрахунків: можливість прозорого ведення фінансових даних та спрощення процесу нарахування оплат;
- Зручну взаємодію між мешканцями та управлінськими організаціями: чат, форма зворотного зв’язку чи інші інструменти комунікації;
- Персоналізований доступ**:** кожен користувач має власний акаунт, де може перевірити баланс, подивитися історію платежів чи отримати інформацію про обслуговування будинку.

Дизайн сайту виконаний у мінімалістичному стилі, що є популярним трендом у сучасній розробці веб-ресурсів. Основні кольори – білий фон, сині акценти для виділення ключових елементів та іконок. Такий вибір кольорів створює відчуття довіри та прозорості.

Шрифти чіткі та добре читаються, що важливо для відвідувачів різного віку. Пропорції тексту, іконок та зображень добре збалансовані, а кожен розділ чітко відокремлений один від одного, завдяки чому сторінка виглядає структуровано.

Сайт “Мій Дім Online” залишає позитивне враження завдяки своєму сучасному вигляду та зручності використання. Він демонструє професійний підхід до розробки, враховуючи потреби як мешканців багатоквартирних будинків, так і управлінських компаній. Головна сторінка не лише приваблює відвідувачів, але й одразу надає всю необхідну інформацію, спрямовуючи користувача до взаємодії із сервісом. Головна сторінка «Мій Дім Онлайн» зображена на рисунку 1.3.

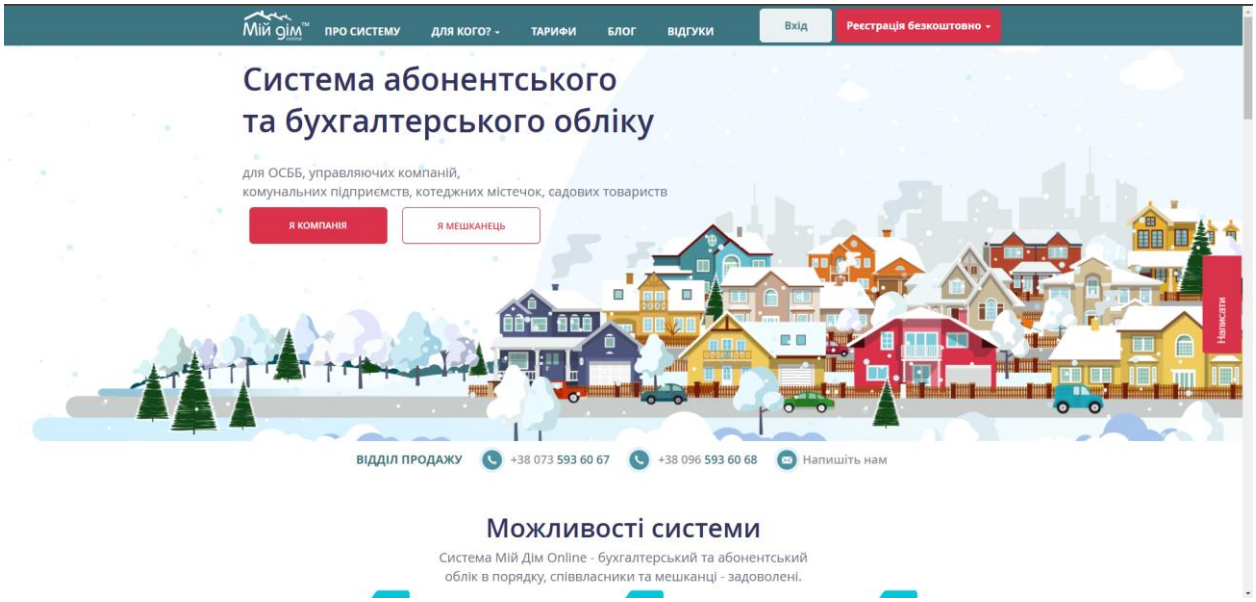


Рис. 1.3 Головна сторінка «Мій Дім Онлайн»

2.2.2 Наявні рішення в Європі

Для дослідження інформаційних систем в предметній області було обрано такі наявні рішення з Європи:

– hausperfekt.de;

– facilityapps.

Hausperfekt.de – це платформа, яка пропонує рішення для цифрового управління багатоквартирними будинками та нерухомістю. Його головна сторінка виконана у професійному стилі, із сучасним дизайном, що акцентує увагу на зручності, технологічності та ефективності роботи з системою

Одразу при завантаженні сайту відвідувач бачить великий, якісно виконаний банер із ключовим повідомленням, що відображає головну ідею платформи: “Digitale Hausverwaltung leicht gemacht” (Цифрове управління нерухомістю стало простим). Банер займає центральне місце на сторінці та супроводжується закликом до дії, наприклад, кнопкою для безкоштовного тестування ("Jetzt kostenlos testen"). Цей елемент одразу підкреслює користь сервісу та мотивує відвідувача діяти.

Фонове зображення або ілюстрація передає професійність і технічний характер платформи, викликаючи довіру до компанії.

У верхній частині сайту розташоване компактне та зручне меню, яке дозволяє швидко переходити до основних розділів:

– Startseite (Головна сторінка) – повернення на головну сторінку.
– Funktionen (Функції) – опис функціоналу платформи.
– Preise (Ціни) – інформація про тарифні плани.
– Kontakt (Контакти) – для зв’язку з представниками компанії.
– Jetzt testen (Спробувати зараз) – кнопка для реєстрації та тестування платформи.

Меню фіксується при прокрутці сторінки, забезпечуючи постійний доступ до основної навігації.

Інформаційні блоки. Нижче банера на головній сторінці розташовані кілька блоків, що структуровано подають інформацію про переваги платформи. Кожен блок оформлений у вигляді секцій із чіткими заголовками, короткими текстами та ілюстраціями:

– Опис основного функціоналу:
    1) можливість автоматизованого управління фінансами будинку (баланси, рахунки, платежі);

2) зручна комунікація з мешканцями через інтегровані канали зв’язку;
3) спрощення обліку витрат і доходів через систему цифрових звітів.

– Переваги використання**.** Цей блок підкреслює, як HausPerfekt дозволяє заощаджувати час, автоматизуючи рутинні процеси, і знижує адміністративні витрати для керуючих компаній;
– Технологічні особливості.

Інтеграція з сучасними платіжними системами та захищеність персональних даних користувачів виділяються як ключові технологічні переваги.

Візуальний стиль. Головна сторінка виконана в нейтральних і приємних кольорах – білий, сірий і синій. Білий фон створює відчуття простору та легкості, синій колір використовується для акцентів, що підкреслює професійність і довіру до сервісу. Всі елементи виглядають гармонійно та збалансовано.

Іконки, ілюстрації та блоки з текстом мають чіткий дизайн, що сприяє швидкому сприйняттю інформації. Усі елементи розташовані таким чином, щоб користувачеві було легко орієнтуватися навіть без попереднього знайомства із сайтом.

Заклики до дії (CTA). Головна сторінка побудована так, щоб постійно мотивувати відвідувача почати використовувати платформу. Заклики до дії, такі як "Jetzt kostenlos testen" ("Спробувати безкоштовно зараз"), розташовані в кількох місцях, включно з банером, окремими блоками та нижньою частиною сторінки.

Додаткові елементи. У нижній частині головної сторінки (футері) відображено:

– контактні дані компанії;
– посилання на політику конфіденційності та умови використання;
– інформація про компанію.

Головна сторінка сайту HausPerfekt створює враження надійної, платформи, яка орієнтована на автоматизацію та спрощення процесів управління нерухомістю.

Дизайн застарілий і на екрані водночас знаходиться забагато інформації, але багато німецьких сайтів мають схожі проблеми, тому це скоріше специфіка німецького ринку аніж недолік, але дизайн варто оновити до останніх тенденцій ринку.

Facilityapps – це платформа, яка пропонує інноваційні цифрові рішення для управління процесами в різних галузях, таких як управління об'єктами, ландшафтний дизайн, прибирання, технічне обслуговування та багато інших. Головна сторінка сайту акцентує увагу на технологічності, гнучкості та можливості адаптації продукту до потреб клієнтів.

Одразу при завантаженні сторінки видно великий банер із центральним заголовком, що пояснює основну пропозицію компанії: “All-in-one Field Service Software” (Універсальне програмне забезпечення для польових послуг). Цей елемент акцентує увагу на головній ідеї: цифрові рішення для оптимізації процесів.

Банер супроводжується інтерактивним елементом – кнопкою з чітким закликом до дії, наприклад, "Book a Demo" або "Learn More", що спрямовує користувача на наступний крок для взаємодії з продуктом.

Фонове зображення або відео (залежно від пристрою) підкреслює сучасність і технологічність пропозиції.

– Навігаційне меню. Меню розташоване у верхній частині сайту та включає наступні розділи:
    1) Solutions (Рішення) – перелік галузей і напрямків, де використовуються їхні програми;
    2) Features (Функції) – опис основних можливостей системи;
    3) Pricing (Ціни) – інформація про вартість продукту;
    4) About Us (Про нас) – розповідь про компанію;
    5) Contact (Контакти) – для зв’язку з представниками компанії;
    6) Book a Demo – кнопка для бронювання демонстрації продукту;

Меню адаптоване до мобільних пристроїв і легко доступне в будь-якій точці сторінки завдяки фіксації при прокрутці.

– Ключові переваги. На головній сторінці чітко виділено блок із перевагами продукту. Серед них:
  1) економія часу завдяки автоматизації рутинних процесів;
  2) гнучкість і можливість адаптації продукту до конкретних бізнес-потреб;
  3) збільшення продуктивності завдяки мобільному доступу до даних у режимі реального часу.

Кожна перевага супроводжується іконкою або невеликим зображенням для кращого сприйняття.

– Демонстрація продуктую. Центральна частина сторінки присвячена показу того, як працює система. Представлені приклади інтерфейсу мобільних і десктопних додатків, зокрема, графічні елементи, що демонструють управління завданнями, контроль процесів і звітність;
– Галузеві рішення. В окремих секціях головної сторінки представлені рішення для різних галузей:
  1) ландшафтний дизайн;
  2) клінінг (прибирання);
  3) управління об'єктами;
  4) технічне обслуговування.

Кожен напрямок має короткий опис із підкресленням специфічних функцій, які можуть бути корисними для цієї галузі.

– Соціальні докази. На головній сторінці також розміщені відгуки клієнтів і логотипи компаній, які вже використовують FacilityApps. Це створює довіру до продукту та показує його успішність у реальних умовах;
– Форма для контакту. В кінці сторінки розташована форма для збору контактних даних користувачів, які зацікавлені у продукті. Це може бути запит на демонстрацію, питання щодо функціоналу або консультація.

FacilityApps справляє враження добре організованої платформи, орієнтованої на вирішення складних бізнес-задач за допомогою цифрових інструментів. Головна сторінка чітко презентує основні можливості системи, підкреслюючи її універсальність і технологічність. І хоч дизайн не є надсучасним, він простий, не перевантажений інформацією і показує основні переваги сервісу.

2.2.3 Порівняння існуючих рішень

Нижче в таблицях 2.1 та 2.2 буде показаний порівняльний аналіз існуючих рушень.

Таблиця 2.1

Загальний опис існуючих платформ

Критерій	Мій Дім Онлайн	Haus Perfekt	FacilityApps
Основна мета	Автоматизація обліку, розрахунків і управління багатоквартирними будинками.	Управління фінансами та послугами в багатоквартирних будинках, орієнтоване на Німеччину.	Загальна автоматизація польових послуг для різних галузей, включаючи техобслуговування.
Цільова аудиторія	ОСББ (об'єднання співвласників багатоквартирних будинків), керуючі компанії.	Керуючі компанії, менеджери нерухомості.	Компанії, що займаються обслуговуванням будівель, технічним обслуговуванням, ландшафтами.
Технологічний стек	Microsoft .NET + JavaScript	WordPress	WordPress
Підтримка мобільних пристроїв	Веб-платформа з адаптивним дизайном, можливе часткове використання на мобільних пристроях.	Переважно десктопне використання, з мінімальними мобільними функціями.	Висока мобільність, доступ через додатки для смартфонів і планшетів.

Таблиця 2.2

Основні функції існуючих платформ

Критерій	Мій Дім Онлайн	Haus Perfekt	FacilityApps
Облік розрахунків	Облік комунальних послуг, автоматизація розрахунків, онлайн-оплата.	Контроль фінансів, виставлення рахунків, управління платежами.	Не спеціалізується на оплатах у багатоквартирних будинках, але має функції для моніторингу завдань.
Робота з мешканцями	Особисті кабінети мешканців із доступом до інформації про рахунки, звернення до керуючих ОСББ.	Інформування мешканців про фінансовий стан через звіти або розсилки.	Немає прямої роботи з мешканцями; фокусується на оптимізації роботи керуючого персоналу.

Продовження таблиці 2.2

Критерій	Мій Дім Онлайн	Haus Perfekt	FacilityApps
Автоматизація процесів	Повна автоматизація обліку споживання комунальних послуг і виставлення рахунків.	Напівавтоматичні функції для розрахунків і фінансового моніторингу.	Автоматизація польових робіт, облік виконаних завдань, мобільна інтеграція.
Інтеграції	Інтеграція з платіжними системами та банківськими сервісами.	Інтеграція з банківськими рахунками, бухгалтерськими системами.	Широка інтеграція з мобільними додатками та системами управління завданнями.

Схожості:

- Автоматизація**:** Усі три платформи мають за мету автоматизацію процесів, що стосуються управління;
- Цільова аудиторія: "Мій Дім Онлайн" і "Haus Perfekt" орієнтовані на багатоквартирні будинки, тоді як FacilityApps більше підходить для технічного обслуговування й управління завданнями;
- Цифрові рішення: Усі сервіси надають зручний доступ до даних у цифровому форматі.

Відмінності:

- Цільова функціональність:
  1) "Мій Дім Онлайн" спеціалізується на фінансовому обліку комунальних послуг;
  2) "Haus Perfekt" акцентує на контролі загального фінансового стану та управлінні послугами для багатоквартирних будинків у Німеччині;
  3) FacilityApps надає більш універсальні рішення для польових робіт, але без акценту на житловий сектор.
- Технологічна спрямованість:
  1) FacilityApps використовує передові хмарні технології та мобільні додатки;

2) "Мій Дім Онлайн" та "Haus Perfekt" орієнтовані на роботу у веб-форматі з обмеженою мобільною інтеграцією.

– Географічна орієнтація:
  1) "Мій Дім Онлайн" орієнтований на ринок України;
  2) "Haus Perfekt" створений для ринку Німеччини;
  3) FacilityApps – глобальна платформа з широким спектром галузей застосування.

Отже, проаналізувавши дані сервіси, можна відмітити досить високу якість сервісів але не всіх, HausPerfekt не має такого рівню зручності інтерфейсу, як у інших двох варіантів. Кожен має стандартний перелік функцій та певні особливості, які вирізняють її серед конкурентів. Основними можливостями є автоматизація обліку, розрахунків і управління.

Недоліки існуючих систем:

– Сервіс "Мій Дім Онлайн" має кілька недоліків, які впливають на його ефективність і зручність використання. Крім того, функціонал платформи недостатньо гнучкий для обробки нестандартних фінансових ситуацій, таких як розподіл витрат на екстрені потреби чи додаткові послуги. Інтерфейс також не можна назвати інтуїтивно зрозумілим для користувачів із низьким рівнем технічної підготовки, що може вимагати додаткового навчання. Ще одним недоліком є вузька спеціалізація сервісу, орієнтованого лише на багатоквартирні будинки, що обмежує його використання в інших сферах.
– Сервіс "Haus Perfekt", орієнтований переважно на ринок Німеччини, також має свої недоліки. Через мовну специфіку та відповідність місцевому законодавству його важко адаптувати для інших країн. Ще одним мінусом є відсутність сучасного мобільного додатку або адаптивної веб-версії, що значно ускладнює використання для тих, хто звик працювати з мобільних пристроїв. Інтерфейс платформи виглядає застарілим у порівнянні з конкурентами, що може негативно впливати

на досвід користувачів, особливо молодшого покоління. Окрім того, сервіс недостатньо інтегрований із сучасними технологіями, як-от хмарні рішення чи API, що обмежує можливості розширення функціоналу та інтеграції з іншими програмами.

– Сервіс "FacilityApps" має свої унікальні недоліки. Перш за все, він не спеціалізується на управлінні багатоквартирними будинками, адже його основна мета – забезпечення технічного обслуговування об’єктів. Надмірно широкий функціонал може створювати труднощі для невеликих ОСББ або керуючих компаній, які потребують простих і зрозумілих рішень. Крім того, цей сервіс є значно дорожчим у порівнянні з платформами, які спеціалізуються лише на багатоквартирних будинках, що може відлякати невеликі організації. Основний акцент "FacilityApps" спрямований на польові послуги й технічне обслуговування, що робить його менш привабливим для користувачів, які шукають інструменти для автоматизації фінансового обліку.

2.3 Ідеї для покращення існуючих сервісів

Можливі застосування:

– Мій Дім Онлайн**:** Може розширити функціонал за рахунок мобільного додатку для Android та iOS, використовуючи NestJS у поєднанні з Ionic або React Native;
– Haus Perfekt: Можна додати хмарні сервіси для зберігання даних та інтеграції з популярними бухгалтерськими системами через API;
– FacilityApps: Взяти приклад із мобільного доступу та додати автоматизацію управління послугами в ОСББ через універсальні системи.

Ключові аспекти аналізу:

- Платформи на кшталт "Мій Дім Онлайн" можуть слугувати базою для розробки нових цифрових платформ, що поєднують сучасні технології на зразок NestJS для створення масштабованих і гнучких рішень;
- Інтеграція з хмарними сервісами, мобільними платформами та API платежів – це основний тренд у подібних розробках.

2.3.1 Релевантність NestJS для подібних систем

NestJS – це потужний фреймворк для побудови бекенд-сервісів. У випадку подібних платформ використання NestJS може бути виправданим завдяки:

- **Модульності:** Легко розподілити різні частини системи (облік, автоматизація, інтеграція з платіжними системами) в окремі модулі;
- **Мікросервісної архітектури:** NestJS дозволяє створювати мікросервіси, що можуть масштабуватися залежно від потреб платформи[23];
- **Інтеграції:** Простота інтеграції з базами даних (PostgreSQL, MongoDB), а також зовнішніми API, які можуть використовуватись для автоматизації (наприклад, платіжні системи чи облік послуг).

2.4 Математичні моделі та алгоритми для автоматизації розрахунків

В роботі цифрової платформи для автоматизації обліку розрахунків у багатоквартирних будинках можна використовувати низку математичних формул та алгоритмів, які допомагають автоматизувати фінансові розрахунки, оптимізувати ресурси та забезпечити точність обліку. Ось деякі з них:

Розрахунок суми платежу для кожного мешканця будинку:

$$S = \frac{(Z * Pk)}{Zp} \quad (1)$$

де:

S – сума платежу мешканця;

Z – загальна сума рахунку;

Pk – площа квартири мешканця;

Zp – загальна площа всіх квартир.

Розподіл витрат за кількістю осіб у квартирі

$$Pm = \frac{(Zv * Km)}{Kz} \quad (2)$$

де:

Pm – платіж мешканця;

Zv – загальна сума витрат;

Km – кількість мешканців у квартирі;

Kz – загальна кількість мешканців у будинку.

Розрахунок відсотка оплачених рахунків:

$$V = (\frac{So}{Zs}) * 100 \quad (3)$$

де:

V – відсоток оплати;

So – сума оплачених рахунків;

Zs – загальна сума рахунків.

Розрахунок економії після впровадження енергозберігаючих технологій:

$$E = (\frac{Vd - Vp}{Vd}) * 100 \quad (4)$$

де:

E – економія;

Vd – витрати до впровадження;

Vp – витрати після впровадження.

2.5 Дослідження динаміки заборгованостей у сфері ЖКГ

За останні два роки проблема заборгованостей за комунальні послуги в Україні залишалася однією з найбільш актуальних у сфері житлово-комунального господарства (ЖКГ)[19]. За даними Міністерства розвитку громад та територій України, станом на кінець 2023 року загальна сума заборгованостей за житлово-

комунальні послуги перевищила 70 мільярдів гривень. Це величезна сума, яка свідчить про низку проблем у сфері обліку та взаємодії між мешканцями багатоквартирних будинків і житлово-комунальними підприємствами (ЖКП).

Основні дані щодо заборгованостей:

- за постачання та розподіл газу становлять понад 35% від загальної суми заборгованості;
- борги за опалення та гаряче водопостачання складають близько 25%;
- заборгованість за водопостачання та водовідведення – близько 20%;
- інші види послуг, зокрема електричне постачання та обслуговування будинків, займають решту частину.

Такі дані свідчать про те, що традиційні методи обліку та взаємодії з мешканцями не дозволяють ефективно контролювати стан заборгованості. Багато житлово-комунальних підприємств досі використовують застарілі системи обліку, які не мають можливості для автоматичного оновлення інформації або інтеграції з банківськими системами. Це призводить до того, що мешканці не завжди отримують актуальні квитанції або не отримують їх вчасно.

Основні причини накопичення боргів:

- Відсутність автоматизованих нагадувань для мешканців про строки оплати послуг. Багато ЖКП не використовують сучасні технології для автоматичного нагадування про необхідність здійснити платіж, що може призвести до затримок і помилок.
- Непрозорість нарахувань. Мешканці не завжди розуміють, за що саме вони платять, особливо в умовах змінюваних тарифів або сумнівних розрахунків, що створює додаткові конфлікти.
- Технічні помилки під час внесення даних вручну. Відсутність автоматизації призводить до помилок при введенні даних або розрахунках, що в свою чергу викликає недовіру мешканців до системи.

– Неефективна комунікація через застарілі канали зв'язку. Більшість підприємств ЖКГ досі використовують звичайні листи, телефонні дзвінки або повідомлення через інші канали зв'язку, що значно сповільнює процес інформування.

Ці фактори впливають на ефективність системи обліку і сприяють збільшенню заборгованості, оскільки мешканці не мають достатньої мотивації або розуміння, щоб платити вчасно.

Випадки неплатежів через відсутність належної автоматизації обліку

Одним із основних факторів, які сприяють накопиченню заборгованості в багатоквартирних будинках, є відсутність належної автоматизації обліку. Відсутність сучасних програмних рішень для обробки даних та здійснення нарахувань веде до помилок, затримок в отриманні квитанцій і непорозумінь між мешканцями та ЖКП.

Ключові фактори, що впливають на неплатежі:

– Ручне ведення даних. У багатьох ЖКП досі застосовують старі методи ведення обліку, зокрема Excel-таблиці або навіть паперові реєстри. Ці методи не можуть забезпечити високий рівень точності і прозорості, що призводить до помилок при розрахунках сум до оплати. Більше того, вручну складати квитанції або вести облік заборгованості – це дуже часозатратно і неефективно.

– Невчасне оновлення даних. У разі використання застарілих методів або ручного вводу даних часто відбувається затримка в оновленні інформації про оплату або заборгованість, що може викликати плутанину серед мешканців і збільшення кількості неплатежів.

– Відсутність інструментів для автоматичних нагадувань. Оскільки традиційні системи обліку не здатні автоматично генерувати нагадування про строки оплати, мешканці часто забувають про необхідність здійснити платіж або дізнаються про заборгованість занадто пізно.

Приклади використання застарілих методів у конкретних організаціях ЖКГ

Незважаючи на прогрес у технологіях і прагнення до автоматизації, багато житлово-комунальних підприємств досі використовують застарілі методи ведення обліку, що негативно впливає на процеси збору платежів та комунікацію з мешканцями.

Приклади таких методів:

- Паперові журнали обліку. У деяких малих населених пунктах житлово-комунальні підприємства досі ведуть облік платежів за допомогою паперових журналів або зошитів. Це не лише забирає багато часу, а й збільшує ймовірність втрати даних або виникнення помилок при введенні інформації.
- Excel-таблиці. Хоча Excel дозволяє деяку автоматизацію обліку, цей метод не може забезпечити необхідний рівень точності і прозорості, особливо при великих обсягах даних. Крім того, таблиці не дозволяють легко здійснювати інтеграцію з іншими системами або генерувати автоматичні квитанції для мешканців.
- Локальні програми без інтеграції. Деякі ЖКП досі використовують програмне забезпечення, яке не має інтеграції з банківськими системами або онлайн-платежами, а також не дозволяє мешканцям здійснювати оплату через Інтернет. Це спричиняє додаткові проблеми, такі як затримки у обробці платежів та неможливість автоматично нараховувати пільги чи знижки.

Ці приклади чітко показують, як важливо для ЖКП впроваджувати сучасні автоматизовані системи обліку, які зможуть зменшити заборгованість, підвищити рівень довіри мешканців і покращити загальну ефективність роботи.

РОЗДІЛ 3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ ІНСТРУМЕНТІВ АВТОМАТИЗАЦІЇ ОБЛІКУ ТА РОЗРАХУНКІВ

3.1 Обгрунтування вибору інструментів для розробки

Nest.JS – це прогресивний фреймворк для розробки серверних застосунків на JavaScript/TypeScript, що базується на принципах модульності, масштабованості та зручності використання. Завдяки своїй архітектурі, яка слідує шаблону MVC (Model-View-Controller) та використовує декоратори для опису логіки, Nest.js забезпечує високий рівень структурованості коду, що є важливим для створення складних і стабільних систем, таких як цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках.

Переваги Nest.js для розробки системи:

- Модульна структура. Nest.js дозволяє розбивати систему на окремі модулі, кожен із яких відповідає за певний функціонал (наприклад, облік платежів, обробка користувачів, управління будинками). Це спрощує розробку, тестування та подальше масштабування проєкту;
- TypeScript за замовчуванням. Використання TypeScript забезпечує строгість типізації, що дозволяє виявляти помилки на етапі компіляції та полегшує роботу з великими командами розробників. Це важливо для зменшення ризиків появи багів у системі, яка повинна обробляти фінансові дані та гарантувати їхню точність;
- Висока продуктивність. Nest.js побудований поверх Node.js, що забезпечує високу швидкість виконання запитів і підтримку багатопотокової архітектури. Це ідеально підходить для роботи з великим обсягом даних і високою кількістю одночасних запитів, які можуть виникати в багатокористувацькій системі;
- Підтримка різних типів баз даних. Nest.js інтегрується з популярними ORM, такими як TypeORM і Prisma, що дозволяє легко працювати з базами даних, як-от PostgreSQL, MongoDB або MySQL. Це дає

можливість обрати найкраще рішення для зберігання даних залежно від потреб системи;

- Інтеграція з іншими інструментами. Завдяки вбудованій підтримці GraphQL, WebSocket і REST API[16], Nest.js дозволяє легко реалізувати різні типи комунікації між компонентами системи. Наприклад, REST API може використовуватися для обміну даними між бекендом і фронтендом, а WebSocket – для реального часу (наприклад, сповіщення про зміну стану рахунків);
- Безпека. Nest.js надає можливості для легкої інтеграції механізмів безпеки, таких як аутентифікація через JWT (JSON Web Token), захист від CSRF (Cross-Site Request Forgery) та налаштування CORS (Cross-Origin Resource Sharing). Це важливо для захисту даних користувачів і запобігання несанкціонованому доступу;
- Підтримка мікросервісів. Nest.js має вбудовану підтримку архітектури мікросервісів, що дозволяє розділити систему на незалежні сервіси для виконання специфічних завдань. Наприклад, обробка платежів може бути винесена в окремий мікросервіс, що зменшить навантаження на основний сервер;
- Активна спільнота та документація. Nest.js має добре розвинену спільноту, велику кількість готових рішень і чудову документацію, що значно спрощує процес розробки, особливо для великих проєктів.

Також Nest.js використовує принципи SOLID[12]. Використання принципів SOLID у фреймворку має суттєве значення для структурованості, підтримки та масштабованості програмного забезпечення.

SOLID принципи в NestJS та їх переваги:

- Single Responsibility Principle (Принцип єдиної відповідальності). У NestJS модулі, контролери, сервіси та інші компоненти чітко розділені за їхньою відповідальністю.

- Open/Closed Principle (Принцип відкритості/закритості). Код у NestJS розробляється таким чином, щоб бути відкритим для розширення, але закритим для змін. Це досягається через використання інтерфейсів, залежностей та декораторів.
- Liskov Substitution Principle (Принцип підстановки Барбари Лісков). У NestJS класи легко замінюються їхніми підтипами або реалізаціями завдяки використанню інтерфейсів і залежностей через DI (Dependency Injection).
- Interface Segregation Principle (Принцип розподілу інтерфейсів). У NestJS сервіси та модулі розробляються з мінімальними, вузько орієнтованими інтерфейсами, що відповідають лише за необхідну функціональність.
- Dependency Inversion Principle (Принцип інверсії залежностей). Завдяки вбудованому механізму DI (Dependency Injection) NestJS дозволяє залежностям задаватися через абстракції, а не через конкретні реалізації.

Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках має обробляти великий обсяг фінансових транзакцій, взаємодіяти з користувачами, забезпечувати високу безпеку та гнучкість. Використання принципів SOLID у NestJS допомагає створювати масштабовані, гнучкі й надійні серверні застосунки. Це особливо цінно для складних проєктів, таких як системи автоматизації або платформи з обробкою великого обсягу даних. Завдяки своїй продуктивності, модульності та можливостям інтеграції Nest.js ідеально підходить для таких задач. Цей фреймворк дозволяє створювати стабільну й масштабовану систему, яка може обробляти дані різного типу (рахунки, оплати, звіти) та забезпечувати комфортну роботу кінцевих користувачів.

PostgreSQL – це одна з найпопулярніших об'єктно-реляційних систем керування базами даних (СКБД), яка має відкритий вихідний код та велику

кількість функцій для обробки даних. Вона широко використовується для проєктів, які потребують високої надійності, продуктивності та гнучкості в роботі з даними.

Переваги PostgreSQL:

- Надійність і стабільність. PostgreSQL забезпечує високу стабільність роботи завдяки механізмам ACID (атомарність, узгодженість, ізоляція, довговічність), які гарантують коректність обробки транзакцій. Це є критично важливим для систем, що працюють із фінансовими розрахунками, де будь-яка помилка в даних може призвести до значних збитків;
- Масштабованість. PostgreSQL дозволяє працювати з великими обсягами даних і забезпечує масштабованість як вертикальну (додавання потужностей одного сервера), так і горизонтальну (шардування). Це ідеально для багатокористувацьких систем, таких як цифрова платформа для автоматизації розрахунків у багатоквартирних будинках;
- Розширені можливості запитів. PostgreSQL підтримує складні SQL-запити, індекси, функції та вбудовані процедури, що спрощує обробку великих обсягів інформації та дозволяє ефективно управляти фінансовими даними, зокрема підрахунками балансу, платежів і заборгованостей;
- Гнучкість у роботі з типами даних. PostgreSQL підтримує широкий набір типів даних, включаючи JSON, JSONB, масиви та типи, створені користувачем. Це дозволяє зберігати як структуровані, так і неструктуровані дані, що корисно для зберігання специфічної інформації, наприклад, історії оплат чи звітів;
- Безпека. PostgreSQL має вбудовані механізми аутентифікації та шифрування, що забезпечують захист даних на рівні СКБД. Це дозволяє зберігати конфіденційні фінансові дані користувачів без ризику несанкціонованого доступу;

– Активна спільнота та підтримка. PostgreSQL має велику спільноту розробників, добре документовану інфраструктуру та безліч готових рішень, що спрощує її інтеграцію та використання.

PostgreSQL забезпечує високу продуктивність і стабільність для роботи з фінансовими даними. Вона дозволяє ефективно управляти даними різного типу, реалізовувати складну логіку та забезпечувати масштабованість системи, що є критичним для багатокористувацької платформи.

Prisma ORM

Prisma ORM – це сучасний інструмент для роботи з базами даних, який надає зручний і типобезпечний інтерфейс для взаємодії з СКБД. Prisma підтримує роботу з PostgreSQL, що дозволяє поєднати його потужність із простотою використання.

Переваги Prisma ORM:

– Типобезпека. Prisma генерує автоматично типи для всіх моделей і запитів, що забезпечує захист від помилок під час компіляції. Це важливо для системи, яка працює з фінансовими даними, оскільки дозволяє уникнути помилок у запитах до бази даних;
– Простота використання. Prisma використовує декларативний підхід для створення моделей і взаємодії з базою даних. Розробник працює з моделями на рівні коду, а Prisma автоматично генерує SQL-запити, що значно прискорює розробку;
– Міграції баз даних. Prisma надає інструменти для легкого управління міграціями. Зміни в схемі бази даних описуються у вигляді моделей, після чого Prisma генерує відповідний SQL-код. Це полегшує оновлення структури даних і забезпечує узгодженість;
– Гнучкість. Prisma підтримує складні відношення між моделями, що дозволяє легко реалізувати зв'язки між таблицями, наприклад, між користувачами, будинками та платежами;
– Підтримка реального часу. Prisma має інтеграцію з GraphQL та інструменти для роботи з даними в реальному часі, що може бути

корисним для сповіщення користувачів про зміну стану рахунків чи нові платежі;

– Інтеграція з Nest.js. Prisma легко інтегрується з Nest.js, завдяки чому розробник може швидко налаштувати взаємодію між ORM і бекендом, зберігаючи структурованість коду.

Prisma ORM забезпечує зручну та ефективну роботу з базою даних PostgreSQL, дозволяє легко управляти міграціями, уникати помилок завдяки типобезпеці та швидко реалізовувати складну логіку роботи з даними. Завдяки інтеграції з Nest.js Prisma ідеально підходить для розробки цифрової платформи, яка має забезпечувати облік і автоматизацію розрахунків у багатоквартирних будинках.

Next.js – це популярний фреймворк для React, який надає потужні можливості для розробки як серверних, так і клієнтських веб-додатків. Завдяки широкому набору вбудованих функцій, таких як рендеринг на стороні сервера (SSR), статичне генерування сторінок (SSG), а також гнучкості у використанні API-роутів, Next.js є одним з найбільш зручних і ефективних інструментів для створення сучасних веб-додатків.

Переваги Next.js 14:

– рендеринг на стороні сервера (SSR)[22] Однією з ключових переваг Next.js є його підтримка рендерингу на сервері (SSR), що дозволяє згенерувати HTML-сторінки на сервері до того, як вони будуть передані клієнту. Це важливо для SEO-оптимізації, адже пошукові системи можуть індексувати контент сторінок ще до їх повного завантаження. Крім того, SSR забезпечує кращу продуктивність і зменшує час завантаження сторінок, що покращує користувацький досвід;

– статичне генерування сторінок (SSG)[22] Next.js підтримує статичне генерування сторінок (SSG), що дозволяє створювати сторінки під час розробки, які потім можуть бути розміщені на сервері або у хмарному

сховищі. Цей підхід значно покращує швидкість завантаження сторінок, оскільки вони вже готові до відправки користувачу, що особливо важливо для платформи, яка потребує швидкої реакції, наприклад, для обробки фінансових даних;

- інтеграція з API-роутами Next.js надає можливість створення API-роутів прямо в межах додатку. Це дозволяє легко налаштувати серверну логіку без необхідності в окремому серверному коді. Зв'язок між фронтендом і бекендом можна організувати через ці API-роути, що забезпечує зручну та гнучку інтеграцію з серверною частиною платформи;
- генерація статичних ресурсів для SEO Оскільки Next.js підтримує автоматичну генерацію мета-тегів і відкритих графічних тегів для кожної сторінки, це допомагає значно покращити SEO-позиціювання вебсайту. Оскільки платформа для автоматизації розрахунків у багатоквартирних будинках має потенційно широку аудиторію, зокрема з різними мовами і локаціями, SEO є важливою складовою для залучення нових користувачів;
- простота інтеграції з бекендом Оскільки для бекенд-частини використовується Nest.js, інтеграція між фронтендом і сервером буде зручною завдяки чітко визначеним API-інтерфейсам та стандартам. Next.js дозволяє легко створювати клієнтські запити до серверу, забезпечуючи зручну взаємодію між різними частинами системи;
- оптимізація продуктивності Next.js включає в себе безліч автоматичних оптимізацій, таких як автоматичне розбиття коду, що дозволяє завантажувати тільки ті частини JavaScript, які необхідні для конкретної сторінки. Це значно покращує швидкість завантаження та знижує час відгуку веб-додатка, що є важливим для користувачів, які працюють з даними в реальному часі або мають високі вимоги до продуктивності;

– міжплатформність та підтримка мобільних версій Next.js дозволяє швидко розробляти додатки, які автоматично адаптуються під різні розміри екрану завдяки вбудованим засобам для мобільних версій. Враховуючи, що багато користувачів платформи можуть звертатися до неї через мобільні пристрої, підтримка респонсивного дизайну є необхідною.

Next.js 14 є відмінним вибором для фронтенду платформи автоматизації обліку розрахунків у багатоквартирних будинках завдяки його потужним можливостям для оптимізації продуктивності, гнучкості в розробці та інтеграції з бекендом. Його підтримка рендерингу на сервері (SSR) і статичного генерування сторінок (SSG) дозволяє створити швидкий та ефективний інтерфейс для користувачів, а автоматичні оптимізації коду покращують загальну продуктивність веб-додатка. Також, завдяки зручній інтеграції з API-роутами та іншими технологіями, Next.js дозволяє легко обробляти запити та дані, що робить його ідеальним інструментом для створення цифрових платформ.

Tailwind CSS – це утилітарний CSS-фреймворк, який дозволяє швидко створювати кастомізовані, адаптивні та ефективні інтерфейси. Його особливість полягає в тому, що він базується на використанні класів для безпосереднього застосування стилів до елементів без необхідності написання власних CSS-правил. Це дає розробникам значну гнучкість у налаштуванні вигляду додатка, дозволяючи уникати дублювання коду і забезпечуючи чистоту та зручність при розробці.

Переваги Tailwind CSS

– Швидкість розробки. Завдяки великій кількості утилітних класів, Tailwind CSS дозволяє значно скоротити час на створення і налаштування інтерфейсів. Розробники можуть моментально застосовувати необхідні стилі до елементів, без необхідності додатково писати окремі CSS-правила для кожного компонента.

– Легкість у налаштуванні. Tailwind дозволяє користувачам легко змінювати стилі за допомогою класів, що дозволяє швидко адаптувати

інтерфейс до вимог проекту або бренду. Це особливо корисно, коли потрібні регулярні зміни або оптимізації в дизайні без великого клопоту.

- Адаптивний дизайн. Один з найбільших плюсів Tailwind CSS – підтримка мобільної адаптивності. Розробники можуть використовувати інтуїтивно зрозумілі класи для налаштування вигляду інтерфейсу на різних пристроях (мобільних, планшетах, десктопах), що дозволяє створювати повністю адаптивні веб-сайти та додатки.
- Мінімізація стилів. Tailwind підтримує вбудовані інструменти для видалення невикористовуваних стилів, що дозволяє значно зменшити розмір фінального CSS-файлу. Це є важливим аспектом для продуктивності, оскільки зменшується час завантаження сторінок.
- Спільнота та підтримка. Tailwind має велику і активну спільноту розробників, що надає можливість отримати допомогу або знайти готові рішення для різних задач. Також існує велика кількість плагінів та компонентів, які можна інтегрувати для ще більшого покращення розробки.

ShadCN/UI[21] – це бібліотека компонентів, яка була створена для спрощення розробки інтерфейсів у проектах на базі React та інших фреймворків. Вона надає набір адаптованих і готових до використання компонентів UI, які можна швидко інтегрувати в проект і налаштувати під потреби розробника.

Переваги ShadCN/UI:

- Готові компоненти для використання. ShadCN/UI пропонує набір готових компонентів, таких як кнопки, модальні вікна, меню, таблиці тощо. Це дозволяє зекономити час, оскільки розробник може використовувати ці компоненти, не витрачаючи час на їх розробку з нуля;
- Інтуїтивно зрозумілий дизайн. Компоненти бібліотеки мають сучасний і мінімалістичний дизайн, що відповідає стандартам і вимогам

сучасних веб-додатків. Це дозволяє швидко створювати інтерфейси, які виглядають професійно, без необхідності витрачати багато часу на дизайн;

- Інтеграція з Tailwind. CSS Shadcn/ui була розроблена з урахуванням роботи з Tailwind CSS, що дозволяє легко налаштовувати стилі компонентів через утиліти Tailwind. Це робить інтеграцію цих двох технологій надзвичайно простою, що дає можливість швидко налаштувати інтерфейс під потреби проекту;
- Адаптивність і доступність. Бібліотека компонентів автоматично оптимізована для адаптивних дизайнів і забезпечує високий рівень доступності для користувачів з обмеженими можливостями. Це є важливим фактором для створення інклюзивних веб-додатків;
- Швидка інтеграція. Завдяки простоті інтеграції ShadCN/UI в проект, розробники можуть швидко почати використовувати компоненти без додаткових налаштувань або конфігурацій. Це допомагає прискорити розробку фронтенду, особливо коли важливо забезпечити швидке і якісне створення інтерфейсів;

Обрання Tailwind CSS для стилізації фронтенду було зумовлено його гнучкістю, простотою в використанні та можливістю швидко налаштовувати адаптивні інтерфейси. Це дозволяє значно скоротити час розробки та забезпечити високу продуктивність додатку. Вибір ShadCN/UI доповнює Tailwind CSS, надаючи розробникам готові компоненти для швидкого створення інтерфейсів з мінімумом зусиль. Завдяки інтеграції цих двох технологій, розробка веб-додатку є швидкою, гнучкою та ефективною, що є важливим аспектом при створенні цифрової платформи для автоматизації обліку розрахунків у багатоквартирних будинках.

3.1 Архітектура системи

3.2.1 Діаграма взаємодії осіб

Особами що приймають рішення виступають: адміністратори сайта, мешканці, менеджери. Взаємодію цих осіб з веб-сайтом зображено на рис. 3.1.

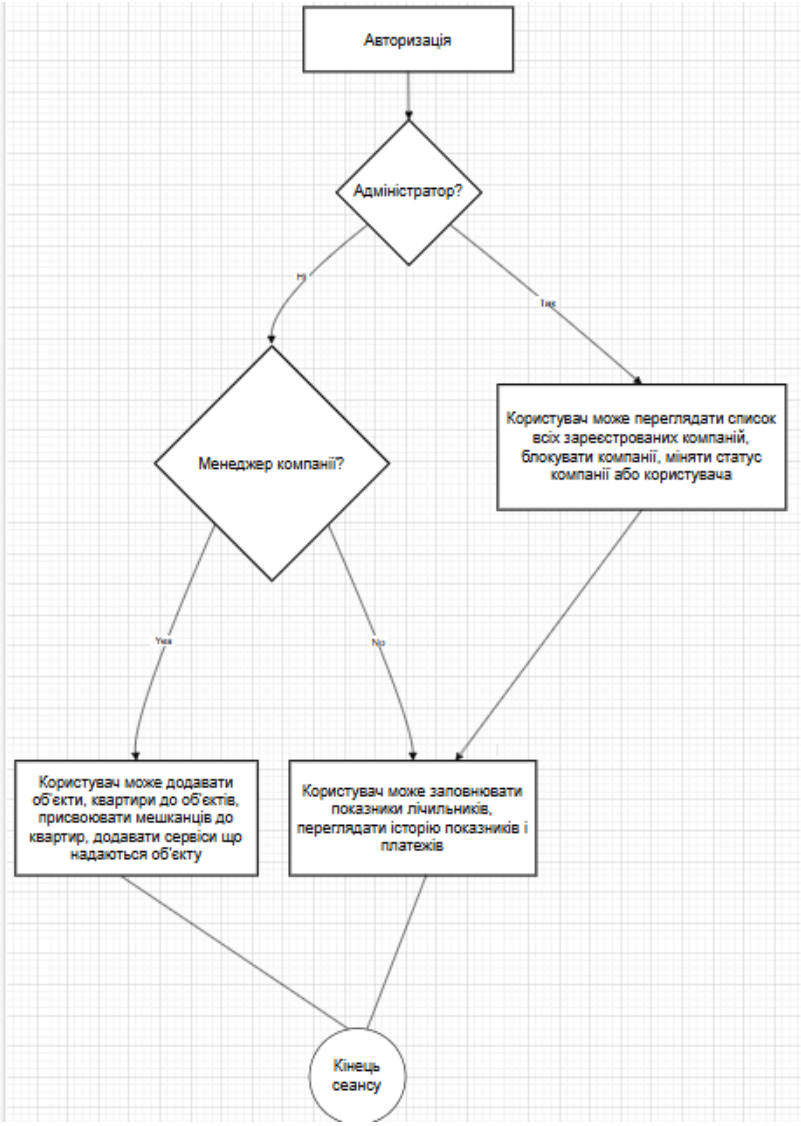


Рис 3.1 – Діаграма взаємодії осіб, що приймають рішення з веб-сайтом

На діаграм взаємодії осіб, зображена можлива частина взаємодій користувачів із сервісом, також зображена перевірка на те ким є Користувач: Адміністратором сервісу, Менеджером Компанії або ж звичайним мешканцем.

В залежності від того ким є користувач, він може мати різний набір можливих дій, у випадку адміністратора сервісу, список можливих дій розповсюджується на всі зареєстровані компанії на сервісі, в той час як список дій менеджера обмежується його компанією. Мешканець, в свою чергу, може тільки переглядати

інформацію пов’язану з ним, наприклад переглядати історію показників лічильників і платежів

3.2.2 Діаграма сутність-зв’язок

Діаграма сутність-зв'язок (Entity-Relationship Diagram, ERD) є важливим інструментом для моделювання даних у системах управління базами даних (СУБД). Вона дозволяє чітко описати структуру даних, які використовуються в додатку, та їх взаємозв'язки.

Призначення діаграми сутність-зв'язок:

- Визначення сутностей (Entities)**:** Сутність у діаграмі ERD – це об'єкт або концепт, що має певні атрибути (поля). Для цифрової платформи для автоматизації обліку розрахунків були створені такі сутності: “Service”, “Dwelling”, “Object”, “ACL”, “User”, “Company”, “DwellingService”, “UserCompany”
- Визначення зв'язків (Relationships)**:** Зв'язки описують, як сутності взаємодіють одна з одною. Наприклад, зв'язок "User" пов’язаний з "Company" (один користувач може належати до багатьох кампаній, а компанія може мати багато користувачів), а "Dwelling" може бути пов'язаний з "Object" (одна квартира належить одному будинку).
- Моделювання бази даних**:** ERD є важливою частиною проектування бази даних, оскільки допомагає визначити, які таблиці і поля потрібно створити, а також як ці таблиці будуть взаємодіяти між собою.
- Оптимізація запитів та структури бази даних**:** За допомогою ERD можна оцінити ефективність структури бази даних і можливі оптимізації, зокрема, щодо нормалізації.

Для даного сервісу, діаграма сутність-зв’язок зображена на рис. 3.2

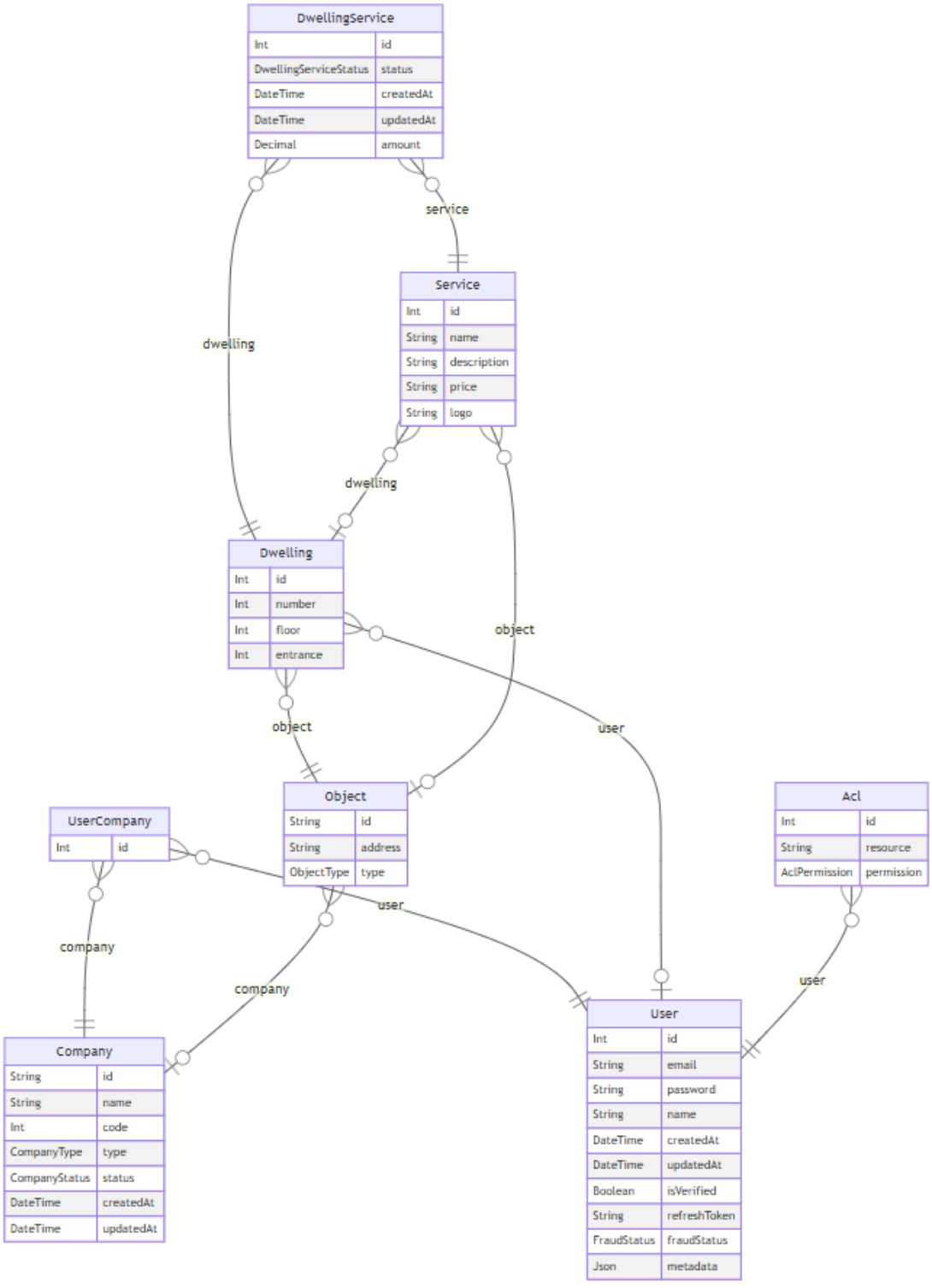


Рис. 3.2 Діаграма сутність-зв’язок для цифрової платформ для автоматизації обліку розрахунків у багатоквартирних домах

3.3 **Реалізація системи**

Інтерфейс головної сторінки системи, що забезпечує доступ до основних функцій платформи (див. рис. 3.3).

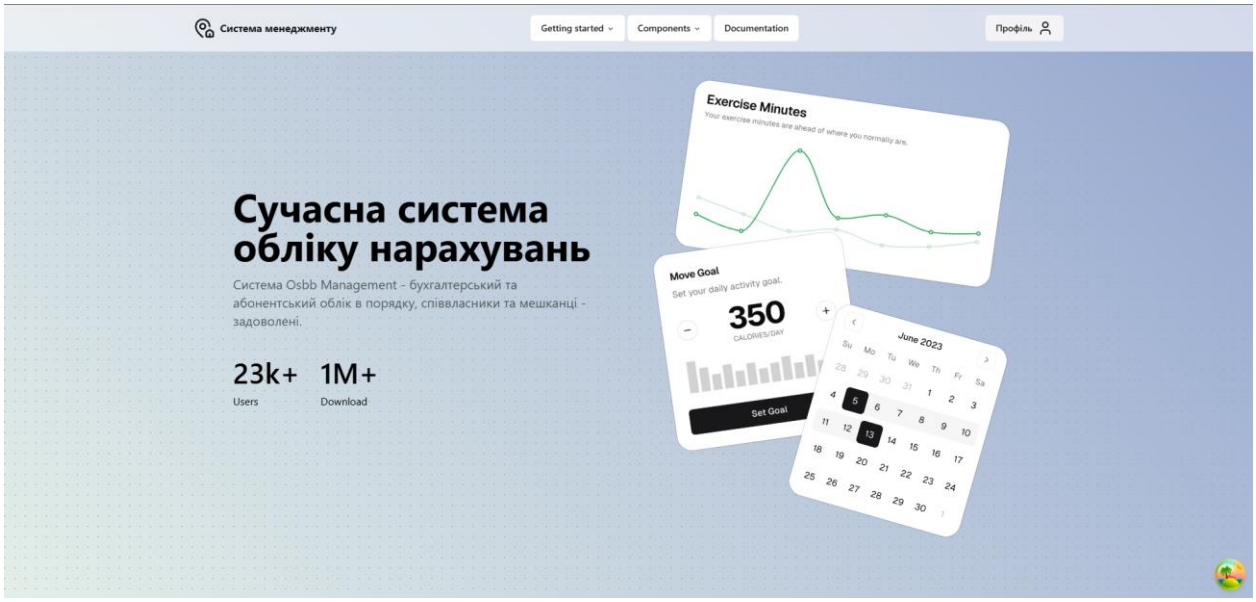


Рис. 3.3 Головна сторінка системи

Персональна сторінка користувача, що містить інформацію про його обліковий запис, а також вікно з налаштуваннями (див. рис. 3.4).

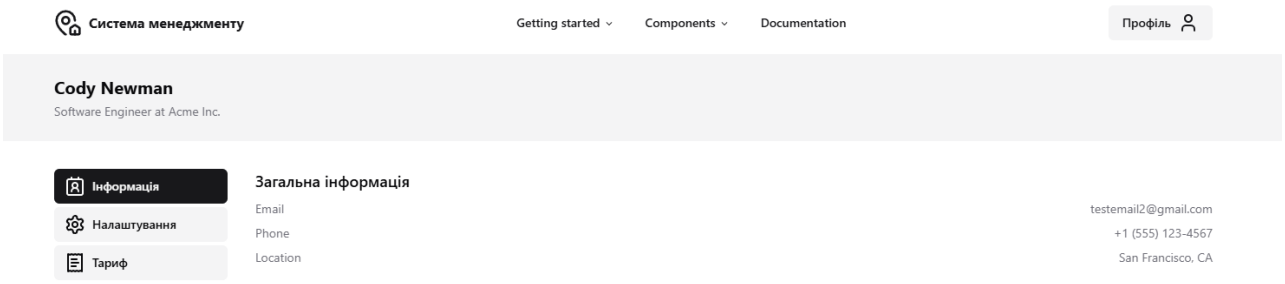


Рис. 3.4 Сторінка профілю

Список зареєстрованих компаній який бачить адміністратор сервісу зображено на рисунку 3.5.

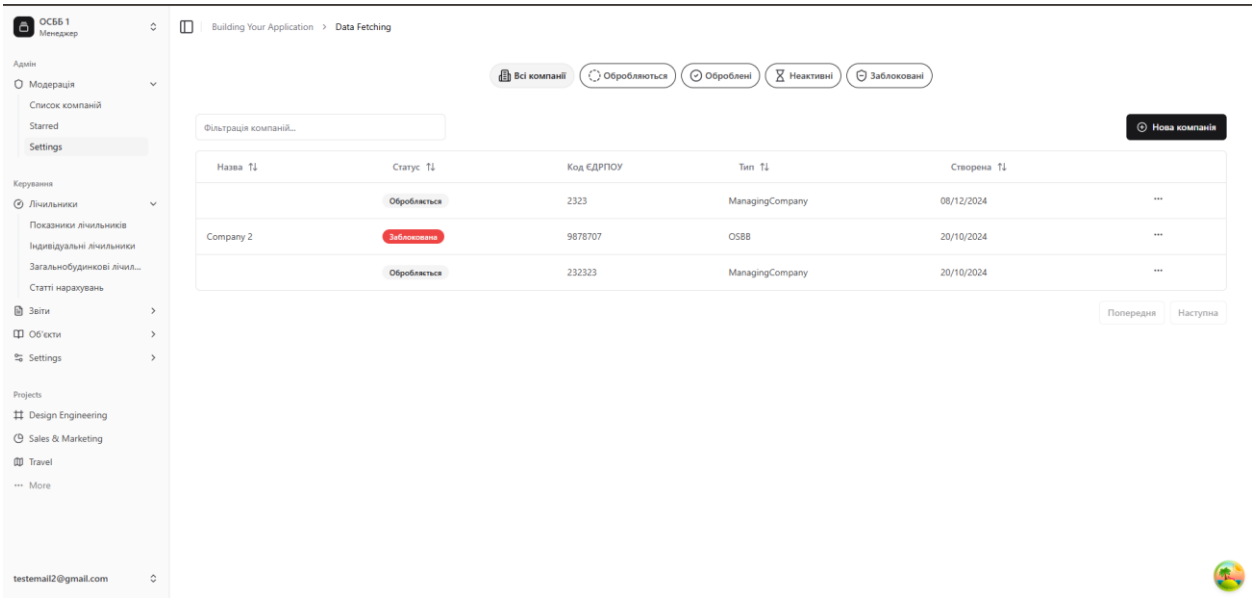


Рис. 3.5 Список зареєстрованих компаній який бачить адміністратор сервісу

Інтерфейс зі списком об’єктів, які належать компанії, доступний для менеджера компанії (див. рис. 3.6).

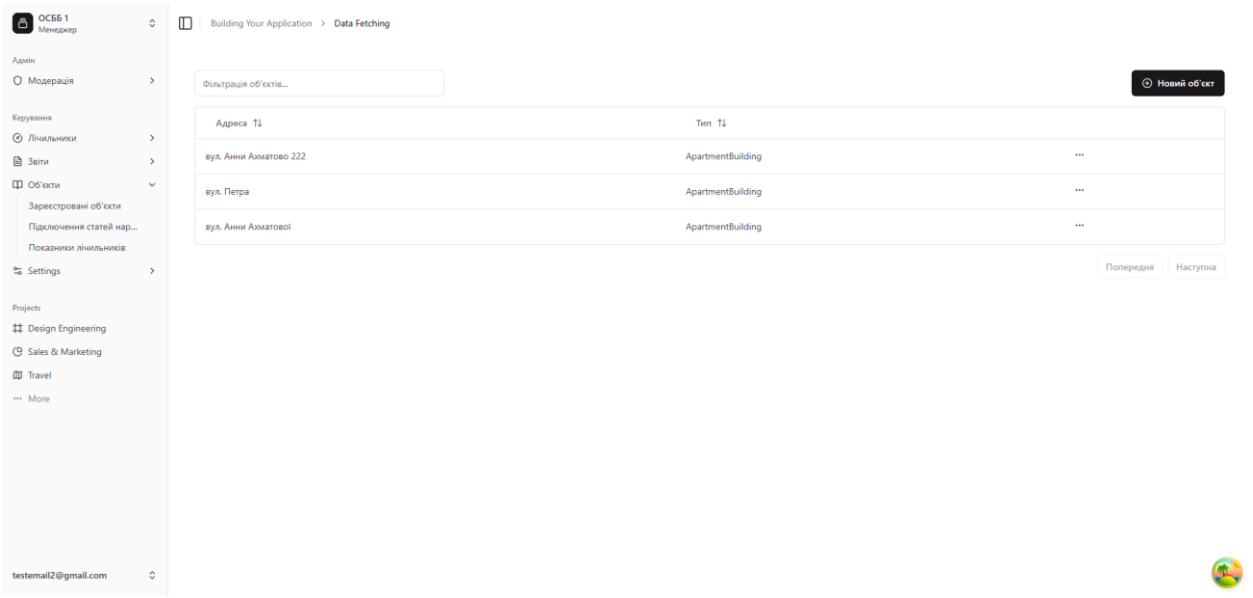


Рис. 3.6 Список зареєстрованих об’єктів який бачить менеджер компанії

Екран із формою для додавання нового об’єкту в систему, доступний менеджеру компанії (див. рис. 3.7).

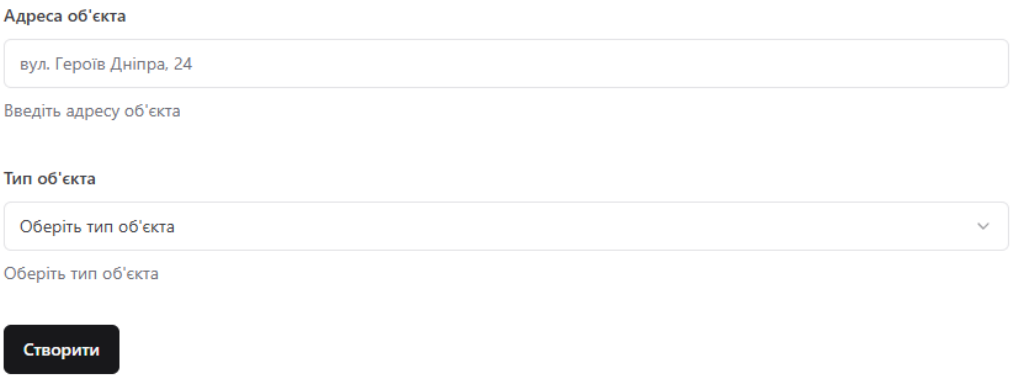


Рис. 3.7 Форма створення об’єкту

Детальна інформація про конкретний об’єкт із можливістю перегляду та редагування даних (див. рис. 3.8).

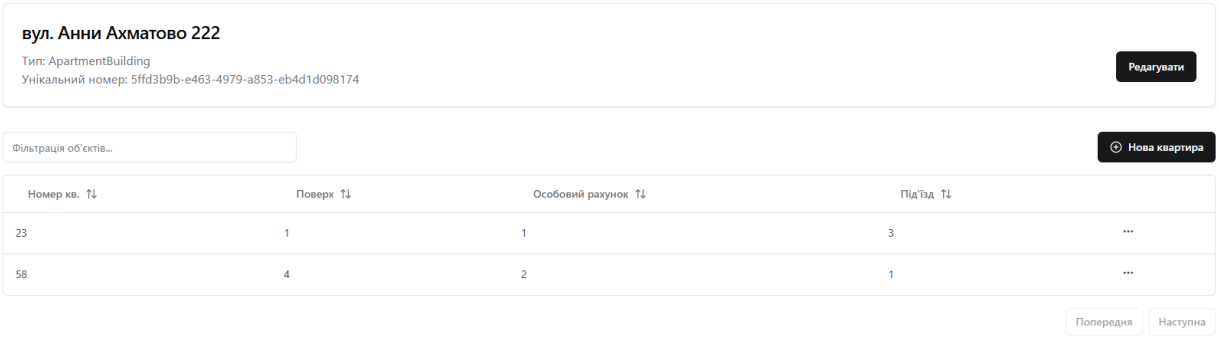


Рис. 3.8 Сторінка об’єкту

Екран із формою, яку нові користувачі використовують для створення облікового запису (див. рис. 3.9).

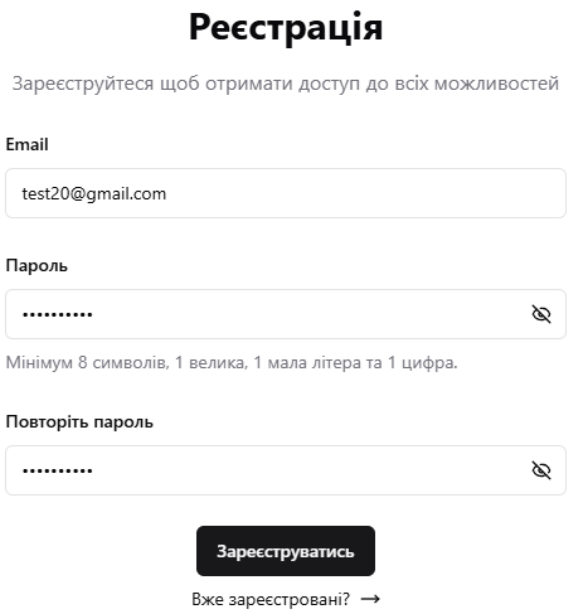


Рис. 3.9 Форма реєстрації

Сповіщення, яке інформує користувача про необхідність підтвердження облікового запису для доступу до захищених сторінок (див. рис. 3.10).

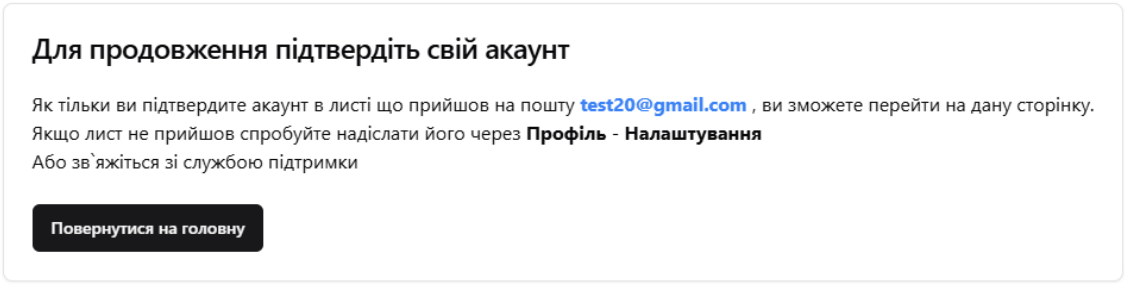


Рис. 3.10 Повідомлення про необхідність підтвердження акаунту при спробі перейти на захищенне посилання

Шаблон електронного листа, який надсилається користувачу для підтвердження облікового запису (див. рис. 3.11).

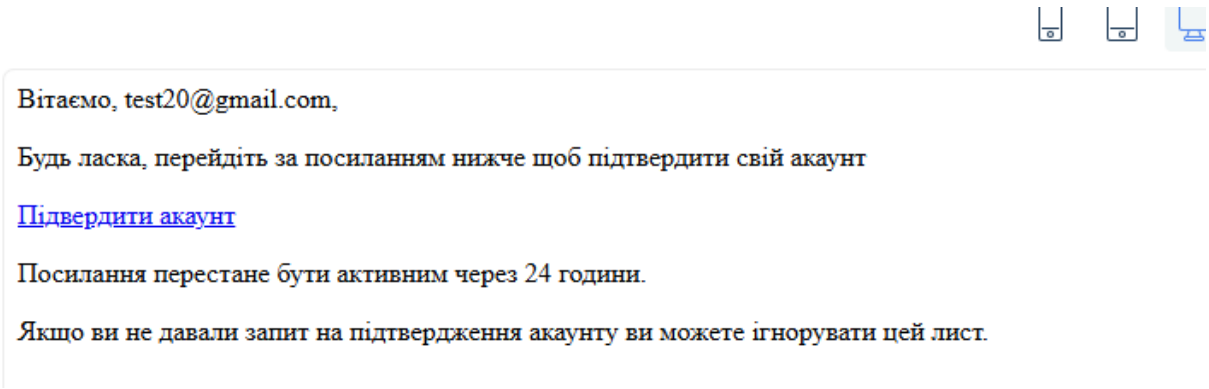


Рис 3.11 – Вигляд листу що приходить на пошту

Реалізація Auth Guard:

```
@Injectable()
export class AuthGuard implements CanActivate {
 constructor(
 private jwtService: JwtService,
 private reflector: Reflector,
 private alsProvider: AsyncLocalStorageProvider,
 private prisma: PrismaService,
 ) {}

 async canActivate(context: ExecutionContext): Promise<boolean> {
 const isPublic = this.reflector.getAllAndOverride<boolean>(IS_PUBLIC_KEY, [
  context.getHandler(),
  context.getClass(),
 ]);
 if (isPublic) {
  return true;
 }

 const request = context.switchToHttp().getRequest();

 const token = this.extractTokenFromHeader(request);

 if (!token) {
  throw new UnauthorizedException('No token provided!');
 }
 try {
  const payload = await this.jwtService.verifyAsync(token, {
  secret: jwtConstants.accessSecret,
  });

  const user = await this.prisma.user.findUnique({
```

```
  where: { id: payload.uId },
  select: { fraudStatus: true },
  });

  if (user?.fraudStatus === FraudStatus.BLOCKED) {
  throw new UnauthorizedException(
   'Акаунт заблоковано через підозрілу активність',
  );
  }

  request['user'] = payload;
  this.alsProvider.set('uId', payload.uId);
 } catch {
  throw new UnauthorizedException('Invalid access token');
 }
 return true;
 }

 private extractTokenFromHeader(request: Request): string | undefined {
 const [type, token] = request.headers.authorization?.split(' ') ?? [];
 return type === 'Bearer' ? token : undefined;
 }
}
```

Рис 3.12 – Реалізація Auth Guard, який перевіряє чи авторизований користувач

Код Auth Guard у NestJS, що забезпечує перевірку авторизації та захист доступу до ресурсів по JWT (Access Token) (див. рис. 3.13).

```
@Public()
@HttpCode(HttpStatus.CREATED)
@ApiOperation({ summary: 'Register a new user/user with assigned company' })
@ApiBody({ type: CreateUserDto })
@ApiResponse({
  status: 201,
  description: 'User successfully registered.',
})
@ApiResponse({ status: 400, description: 'Bad Request' })
@Post('sign-up')
async register(@Body() createUserDto: CreateUserDto): Promise<void> {
  const { email, password, company } = createUserDto;

  return await this.authService.register({ email, password }, company);
}
```

Рис 3.13 – Ендпойнт, який відповідає за реєстрацію користувача

Реалізація ендпойнта для створення нового користувача в системі.

3.4 Результати реалізації інформаційної підсистеми

В процесі реалізації інформаційної підсистеми було створено платформу для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою **NestJS**. Основні результати розробки включають наступні аспекти:

- **Головна сторінка системи:** Реалізовано головний інтерфейс, який забезпечує зручний доступ до ключових функцій системи. Він містить інформаційні блоки, навігацію та базовий огляд функціональності для різних типів користувачів;
- **Система управління профілями користувачів:** Створено інтерфейс сторінки профілю, який дозволяє користувачам переглядати та редагувати свої дані. Для адміністратора і менеджера реалізовано окремі функціональні можливості відповідно до їх ролей;
- **Модуль управління компаніями:** Для адміністратора реалізовано функціонал перегляду списку зареєстрованих компаній, що дозволяє контролювати їхню активність та виконувати адміністративні дії;
- **Модуль управління об’єктами:** Реалізовано інтерфейс для менеджера компанії, який забезпечує доступ до списку зареєстрованих об’єктів компанії. Крім того, створено функцію додавання нового об’єкта за допомогою інтерактивної форми;
- **Сторінка об’єкту:** Забезпечено можливість перегляду детальної інформації про кожен об’єкт. Користувачі з відповідними правами можуть редагувати дані об’єкту;
- **Система реєстрації та аутентифікації:** Реалізовано форму реєстрації для нових користувачів із валідацією даних;
- При спробі доступу до захищених сторінок без підтвердження акаунту, користувач отримує повідомлення із відповідною інструкцією;
- Автоматизовано надсилання листа з підтвердженням акаунту;
- **Механізм аутентифікації:** Розроблено Auth Guard для перевірки авторизації користувача. Реалізація забезпечує:

1) Перевірку токенів доступу, їх дійсності та відповідність прав користувача.
2) Захист ресурсів від несанкціонованого доступу.
3) Використання асинхронного локального сховища для зберігання ID авторизованого користувача.

– **Ендпойнти для роботи з користувачами:** Реалізовано ендпойнти для реєстрації, логіну, а також підтвердження електронної пошти. Забезпечено необхідну валідацію запитів та відповідну обробку даних. Також додано ендпойнти для роботи з компанією, додаванням об’єктів до компанії, квартир, також реалізовано логіку визначення прав користувача за допомогою ACL[20] – Access Control List;
– **Візуалізація роботи системи:** Усі ключові функціональні модулі супроводжуються скріншотами, що демонструють інтерфейси системи, приклади введення даних та результати обробки.

Розроблена інформаційна підсистема відповідає меті дослідження, забезпечуючи автоматизацію облікових процесів у багатоквартирних будинках. Вона забезпечує інтуїтивно зрозумілий інтерфейс, ефективне управління даними та надійний рівень захисту інформації завдяки використанню сучасних підходів у програмуванні та технологій NestJS.

ВИСНОВКИ

Підсумовуючи виконану роботу над дипломним проєктом "Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою NestJS", були зроблені наступні узагальнення та висновки, які відображають основні результати, переваги та недоліки проєкту, а також надають рекомендації щодо їх вдосконалення.

Розроблено сучасну цифрову платформу, архітектура якої побудована із використанням принципів модульності, масштабованості та об'єктно-орієнтованого програмування (ООП). Удосконалено процеси обліку розрахунків, що відповідають потребам різних категорій користувачів: мешканців, адміністраторів і менеджерів компаній.

Система впроваджує інтуїтивно зрозумілий інтерфейс, який забезпечує зручність взаємодії користувачів, та використовує сучасні технології, зокрема:

- NestJS для організації модульної архітектури;
- TypeScript для забезпечення строгого типізування і зменшення ризику помилок;
- Prisma для інтеграції з базами даних;
- PostgreSQL як надійну та продуктивну систему керування даними.

Впроваджено механізми безпеки, зокрема аутентифікацію на основі JWT, захист доступу через Auth Guard та ACL (Access Control List). Ці рішення забезпечують високий рівень безпеки даних.

У процесі роботи над проєктом було удосконалено процеси взаємодії з базою даних за допомогою Prisma та PostgreSQL, що сприяло підвищенню надійності та продуктивності системи. Також удосконалено логіку розробки через застосування принципів SOLI. Крім того, вдосконалено механізми інтеграції з зовнішніми сервісами, що розширює функціональні можливості платформи.

Розробка системи базувалася на впровадженні наукових методів, серед яких використання UML-діаграм для ефективного проектування структури системи. Додатково застосовано методології CI/CD, які дозволяють реалізувати безперервну інтеграцію та доставку оновлень, забезпечуючи гнучкість і швидкість розробки. Реалізація надійних інструментів для моніторингу та тестування гарантує стабільну роботу платформи, забезпечуючи високий рівень якості програмного забезпечення.

Проєкт дозволив удосконалити управління обліком розрахунків у багатоквартирних будинках та продемонстрував значний потенціал для подальшого розвитку. Платформа є зручною, продуктивною та безпечною, а її сучасна архітектура забезпечує можливість інтеграції нових функцій. Реалізація цього проєкту стала важливим етапом у професійному розвитку, дозволила застосувати сучасні методи розробки та створити інноваційне рішення для управління житловими комплексами.

ПЕРЕЛІК ПОСИЛАНЬ

1. MDN Web Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/
2. Бухгалтерський облік у житловому господарстві. [Електронний ресурс] – Режим доступу до ресурсу: https://osvita.ua/vnz/reports/accountant/17490/
3. UX-design [Електронний ресурс] – Режим доступу до ресурсу: https://www.nngroup.com/
4. Функціональні вимоги [Електронний ресурс] – Режим доступу до ресурсу: https://visuresolutions.com/uk/blog/functional-requirements/.
5. Бізнес вимоги [Електронний ресурс] – Режим доступу до ресурсу: https://what.com.ua/biznes-vimogi-rozrobka-ta-pri/.
6. Документація Next.js [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/
7. Документація Typescript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/
8. Як будувати UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/
9. Що таке PostgreSQL і для чого використовується? [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/postgresql-shcho-tse/
10. Що таке Tailwind [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/nuances-of-programming
11. Prisma documenation [Електронний ресурс] – Режим доступу до ресурсу: https://www.prisma.io/
12. SOLID: The First 5 Principles of Object Oriented Design [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
13. JWT (JSON Web Token) Specification [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/
14. Guards у Nest.js [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nestjs.com/guards

15. Continuous Integration and Continuous Deployment (CI/CD) [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment
16. Що таке REST API [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/shcho-take-rest-api/
17. Мій Дім Онлайн – Для кого? [Електронний ресурс] – Режим доступу до ресурсу: https://miydimonline.com.ua/osbb
18. Розроблення системи управління багатоквартирними будинками в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://www.easterneurope-ebm.in.ua/journal/32_2021/10.pdf
19. Напрями удосконалення механізмів державного управління реформуванням житлово-комунального господарства в україні [Електронний ресурс]
20. Access List (ACL) [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/access-lists-acl/
21. Shadcn/ui [Електронний ресурс] – Режим доступу до ресурсу: https://ui.shadcn.com/
22. Способи генерації сторінок [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/41585/
23. Що таке мікросервісна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/scho-take-mikroservisna-arhitektura-znachennya-skladovi-perevagi
24. Що таке Smart City і як виглядає в українських реаліях [Електронний ресурс] – Режим доступу до ресурсу: https://www.prostir.ua/?news=scho-take-smart-city-i-yak-vyhlyadaje-v-ukrajinskyh-realiyah
25. What is IoT (Internet of Things) [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/iot/?nc1=h_ls
26. Виклики та сучасні підходи до кібербезпеки [Електронний ресурс] – Режим доступу до ресурсу: http://journal-app.uzhnu.edu.ua/article/view/310999

27. Мінімально життєздатний продукт (MVP) [Електронний ресурс] – Режим доступу до ресурсу: https://ux.pub/zhmikhov/minimalno-zhittiezdatnii-produkt-minimum-viable-product-mvp-3if3
28. Огляд видів тестування [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/
29. Автоматизація бухгалтерського обліку [Електронний ресурс] – Режим доступу до ресурсу: https://www.netsoft.com.ua/articles-soft/bas-news/avtomatizacija-bukhgaltjerskogo-uchjeta-prjeimushhjestva-kritjerii-vybora-po-osobjennosti-vnjedrjenijaU.html
30. Облік доходів і витрат в ОСББ [Електронний ресурс] – Режим доступу до ресурсу: https://interbuh.com.ua/ua/documents/oneanalytics/34822

ДОДАТКИ

```
@Injectable()
export class UsersService {
 constructor(
  private companiesDataService: CompaniesDataService,
  private userCompanyDataService: UserCompanyDataService,
  private userDataService: UserDataService,
  private alsProvider: AsyncLocalStorageProvider,
  private aclService: AclService,
 ) {}

 async create(data: CreateUserDto) {
  return await this.userDataService.create({
   ...data,
  });
 }

 async assignCompanyToUser(
  code: number,
  type: CompanyTypeEnum,
  userId: number,
 ) {
  const company = await this.companiesDataService.create({ code, type });

  await this.userCompanyDataService.create({ userId, companyId: company.id });
  return company;
 }

 async createAcl(input: CreateAclDto) {
  const { userId, resource, permission } = input;

  const acl = await this.aclService.createAcl({
   userId,
   resource,
   permission,
  });

  return acl;
 }

 async confirmEmail(email: string): Promise<void> {
  const user = await this.findOneByEmail(email);
  if (!user) {
   throw new BadRequestException('User not found');
  }
  if (user.isVerified) {
   throw new BadRequestException('Email already confirmed');
  }
```

```
  const updateUserDto: UpdateUserDto = { isVerified: true };
  await this.update(user.id, updateUserDto);
 }

 async findAll() {
  return await this.userDataService.findAll();
 }

 async findOneById(id: number) {
  return await this.userDataService.findOneById(id);
 }

 async findOneByEmail(email: string) {
  return await this.userDataService.findOneByEmail(email);
 }

 async update(id: number, updateUserDto: UpdateUserDto) {
  if (updateUserDto.password) {
   updateUserDto.password = await hashPassword(updateUserDto.password);
  }

  return await this.userDataService.update(id, updateUserDto);
 }

 async remove(id: number) {
  return await this.userDataService.remove(id);
 }
}
```

```
@Injectable()
export class AsyncLocalStorageProvider {
 private readonly storage = new AsyncLocalStorage<Map<string, any>>();

 run(store: Map<string, any>, callback: () => void) {
  this.storage.run(store, callback);
 }

 getStore(): Map<string, any> {
  return this.storage.getStore();
 }

 set(key: string, value: any) {
  const store = this.getStore();
  if (store) {
   store.set(key, value);
  } else {
   console.warn('Attempting to set ALS value outside of context');
  }
 }
```

```
 get(key: string): any {
  const store = this.getStore();
  return store ? store.get(key) : undefined;
 }
}
```

```
export class CreateObjectDto {
 @IsString()
 address: string;

 @IsEnum(ObjectType)
 type: ObjectType;

 @IsString()
 @IsOptional()
 companyId: string;
}

@ApiTags('Object')

@Controller('object')
export class ObjectController {
 constructor(private readonly objectService: ObjectService) {}

 @Post()
 @ApiOperation({ summary: 'Create a new object' })
 @ApiBody({ type: CreateObjectDto })
 @ApiResponse({ status: 201, description: 'Object successfully created' })
 async create(@Body() data: CreateObjectDto) {
  return await this.objectService.create(data);
 }

 @Get()
 @ApiOperation({ summary: 'Get all objects' })
 @ApiQuery({
  name: 'companyId',
  required: false,
  description: 'Filter objects by company ID',
 })
 @ApiResponse({
  status: 200,
  description: 'List of objects retrieved successfully',
 })
 async findAll(
  @Query() { companyId }: FindObjectsDto,
  @Query() paginationQuery: PaginationParamsDto,
  @Query() { objectSearch }: ObjectSearchParamsDto,
 ) {
  console.log('paginationQuery', paginationQuery);
```

```
  return await this.objectService.findAll(
   paginationQuery,
   companyId,
   objectSearch,
  );
 }

 @Get(':id')
 @ApiOperation({ summary: 'Get object by ID' })
 @ApiParam({ name: 'id', description: 'Object ID' })
 @ApiResponse({ status: 200, description: 'Object retrieved successfully' })
 @ApiResponse({ status: 404, description: 'Object not found' })
 async findOne(@Param('id') id: string) {
  return await this.objectService.findOne(id);
 }

 @Patch(':id')
 @ApiOperation({ summary: 'Update object' })
 @ApiParam({ name: 'id', description: 'Object ID' })
 @ApiBody({ type: UpdateObjectDto })
 @ApiResponse({ status: 200, description: 'Object updated successfully' })
 async update(@Param('id') id: string, @Body() data: UpdateObjectDto) {
  return await this.objectService.update(id, data);
 }

 @Patch(':id/company')
 @ApiOperation({ summary: 'Assign company to object' })
 @ApiParam({ name: 'id', description: 'Object ID' })
 @ApiBody({
  schema: {
   type: 'object',
   properties: {
    companyId: { type: 'string', description: 'Company ID to assign' },
   },
  },
 })
 @ApiResponse({
  status: 200,
  description: 'Company successfully assigned to object',
 })
 async assignCompany(
  @Param('id') id: string,
  @Body() data: { companyId: string },
 ) {
  return await this.objectService.assignCompany(id, data.companyId);
 }

 @Patch(':id/service')
 @ApiOperation({ summary: 'Assign service to object' })
 @ApiParam({ name: 'id', description: 'Object ID' })
 @ApiBody({
```

```
  schema: {
   type: 'object',
   properties: {
    serviceId: { type: 'number', description: 'Service ID to assign' },
   },
  },
 })
 @ApiResponse({
  status: 200,
  description: 'Service successfully assigned to object',
 })
 async assignService(
  @Param('id') id: string,
  @Body() data: { serviceId: number },
 ) {
  return await this.objectService.assignService(id, data.serviceId);
 }

 @Delete(':id')
 @ApiOperation({ summary: 'Delete object' })
 @ApiParam({ name: 'id', description: 'Object ID' })
 @ApiResponse({ status: 200, description: 'Object deleted successfully' })
 @ApiResponse({ status: 404, description: 'Object not found' })
 async remove(@Param('id') id: string) {
  return await this.objectService.remove(id);
 }
}
```

```
@Injectable()
export class ObjectService {
 constructor(
  private objectDataService: ObjectDataService,
  private prisma: PrismaService,
 ) {}

 async create(data: CreateObjectDto) {
  return await this.objectDataService.create(data);
 }

 async findAll(
  paginationParams: PaginationParamsDto,
  companyId?: string,
  objectSearch?: string,
 ) {
  const {
   offset = 0,
   limit = 10,
   sortBy = 'id',
   sortOrder = SortOrder.ASC,
  } = paginationParams;
```

```
 const take = Math.min(limit, 100);
 const skip = offset;

 const orderBy = {
  [sortBy]: sortOrder,
 };

 const where: Prisma.ObjectWhereInput = {
  ...(companyId ? { companyId } : {}),
 };

 if (objectSearch) {
  where.address = {
   contains: objectSearch,
   mode: 'insensitive',
  };
 }

 const [objects, total] = await this.prisma.$transaction(async (tx) => {
  const objects = await this.objectDataService.find(
   {
    where,
    take,
    skip,
    orderBy,
   },
   tx,
  );
  const total = await this.objectDataService.count(
   {
    where: { companyId },
   },
   tx,
  );
  return [objects, total];
 });

 return {
  data: objects,
  meta: {
   total,
   size: take,
   totalPages: Math.ceil(total / take),
  },
 };
}

async findOne(id: string) {
 const object = await this.objectDataService.find({
  where: { id },
```

```
    include: {
      company: true,
      services: true,
    },
  });

  if (!object) {
    throw new NotFoundException("Об'єкт не знайдено");
  }

  return object[0];
 }

 async assignCompany(objectId: string, companyId: string) {
  return this.objectDataService.update(objectId, {
    company: { connect: { id: companyId } },
  });
 }

 async assignService(objectId: string, serviceId: number) {
  return this.objectDataService.update(objectId, {
    services: { connect: { id: serviceId } },
  });
 }

 async update(id: string, data: UpdateObjectDto) {
  return await this.objectDataService.update(id, data);
 }

 async remove(id: string) {
  return await this.objectDataService.delete({ id });
 }
}
```

```
@ApiTags('Dwelling')
@Controller('dwelling')
export class DwellingController {
 constructor(private readonly dwellingService: DwellingService) {}

 @Post()
 @ApiOperation({ summary: 'Create a new dwelling' })
 @ApiResponse({ status: 201, description: 'The dwelling has been created.' })
 async create(@Body() data: CreateDwellingDto) {
  return await this.dwellingService.create(data);
 }

 @Get()
 @ApiOperation({ summary: 'Get all dwellings' })
 @ApiResponse({
  status: 200,
```

```
  description: 'Return paginated, sorted, and filtered list of dwellings.',
 })
 async findAll(
  @Query() query: FindDwellingsDto,
  @Query() paginationQuery: PaginationParamsDto,
 ) {
  return await this.dwellingService.findAll(query, paginationQuery);
 }

 @Get(':id')
 @ApiOperation({ summary: 'Get a dwelling by id' })
 @ApiResponse({ status: 200, description: 'Return the dwelling.' })
 async findOne(
  @Param('id', ParseIntPipe) id: number,
  @Query('services') services?: string,
 ) {
  if (services === 'true') {
   return await this.dwellingService.getDwellingServices(id);
  }
  return await this.dwellingService.findOne(id);
 }

 @Patch(':id')
 @ApiOperation({ summary: 'Update a dwelling' })
 @ApiResponse({ status: 200, description: 'The dwelling has been updated.' })
 async update(
  @Param('id', ParseIntPipe) id: number,
  @Body() updateDwellingDto: UpdateDwellingDto,
 ) {
  if (updateDwellingDto.userId) {
   await this.dwellingService.assignUser(id, updateDwellingDto.userId);
  }

  return await this.dwellingService.update(id, updateDwellingDto);
 }

 @Delete(':id')
 @ApiOperation({ summary: 'Delete a dwelling' })
 @ApiResponse({ status: 200, description: 'The dwelling has been deleted.' })
 async remove(@Param('id', ParseIntPipe) id: number) {
  return await this.dwellingService.remove(id);
 }
}
```

```
"use client";

import { ObjectDataTable } from "./_components/object-data-table";
import { usePagination } from "@/hooks/use-pagination";
import { useGetObjects } from "@/hooks/api/use-get-objects";
import { columns } from "./_components/objectColumns";
```

```tsx
import { useSorting } from "@/hooks/use-sorting";

function ObjectsPage() {
 const { limit, onPaginationChange, offset, pagination } = usePagination();
 const { sorting, onSortingChange, field, order } = useSorting();

 const {
  data: objects,
  isLoading,
  error,
 } = useGetObjects({ pagination: { offset, limit }, sort: { field, order } });

 if (isLoading) {
  return <div className="w-full text-center">Завантаження...</div>;
 }

 if (error) {
  return <div>Error: {error.message}</div>;
 }

 return (
  <div className="p-6 space-y-4">
   <ObjectDataTable
    data={objects.data}
    columns={columns}
    onPaginationChange={onPaginationChange}
    onSortingChange={onSortingChange}
    rowCount={objects.meta.total}
    pagination={pagination}
    sorting={sorting}
   />
  </div>
 );
}

export default ObjectsPage;

"use client";


import * as React from "react";
import {
 ColumnDef,
 OnChangeFn,
 PaginationState,
 SortingState,
 flexRender,
 getCoreRowModel,
 useReactTable,
} from "@tanstack/react-table";
```

```tsx
import {
 Table,
 TableBody,
 TableCell,
 TableHead,
 TableHeader,
 TableRow,
} from "@/components/ui/table";

import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import Link from "next/link";
import { PlusCircle } from "lucide-react";

interface DataTableProps<TData, TValue> {
 columns: ColumnDef<TData, TValue>[];
 data: TData[];
 rowCount: number;
 onPaginationChange: OnChangeFn<PaginationState>;
 pagination: {
  pageIndex: number;
  pageSize: number;
 };
 onSortingChange: OnChangeFn<SortingState>;
 sorting: {
  id: string;
  desc: boolean;
 }[];
 objectId: string;
}

export function DwellingDataTable<TData, TValue>({
 columns,
 data,
 onPaginationChange,
 pagination,
 rowCount,
 onSortingChange,
 sorting,
 objectId,
}: DataTableProps<TData, TValue>) {
 const table = useReactTable({
  data,
  columns,
  manualPagination: true,
  manualSorting: true,
  onPaginationChange,
  onSortingChange,
  state: { pagination, sorting },
  getCoreRowModel: getCoreRowModel(),
  rowCount,
```

```
});

return (
 <div className="w-full">
  <div className="flex items-center py-4 justify-between">
   <Input
    placeholder="Фільтрація об'єктів..."
    value={(table.getColumn("name")?.getFilterValue() as string) ?? ""}
    onChange={(event) =>
     table.getColumn("name")?.setFilterValue(event.target.value)
    }
    className="max-w-sm"
   />
   <Link href={`/dashboard/objects/${objectId}/create-dwelling`}>
    <Button>
     <PlusCircle className="h-4 w-4 mr-2" />
     Нова квартира
    </Button>
   </Link>
  </div>
  <div className="rounded-md border">
   <Table>
    <TableHeader>
     {table.getHeaderGroups().map((headerGroup) => (
      <TableRow key={headerGroup.id}>
       {headerGroup.headers.map((header) => {
        return (
         <TableHead
          key={header.id}
          {...(header.column.getCanSort()
           ? { onClick: header.column.getToggleSortingHandler() }
           : {})}
         >
          {header.isPlaceholder
           ? null
           : flexRender(
             header.column.columnDef.header,
             header.getContext()
            )}
         </TableHead>
        );
       })}
      </TableRow>
     ))}
    </TableHeader>

    <TableBody>
     {table.getRowModel().rows?.length ? (
      table.getRowModel().rows.map((row) => (
       <TableRow
        key={row.id}
```

```
          data-state={row.getIsSelected() && "selected"}
        >
          {row.getVisibleCells().map((cell) => (
           <TableCell key={cell.id}>
            {flexRender(
             cell.column.columnDef.cell,
             cell.getContext()
            )}
           </TableCell>
          ))}
        </TableRow>
       ))
      ) : (
       <TableRow>
        <TableCell
         colSpan={columns.length}
         className="h-24 text-center"
        >
         Нічого не знайдено.
        </TableCell>
       </TableRow>
      )}
     </TableBody>
    </Table>
   </div>
   <div className="flex items-center justify-end space-x-2 py-4">
    <Button
     variant="outline"
     size="sm"
     onClick={() => table.previousPage()}
     disabled={!table.getCanPreviousPage()}
    >
     Попередня
    </Button>
    <Button
     variant="outline"
     size="sm"
     onClick={() => table.nextPage()}
     disabled={!table.getCanNextPage()}
    >
     Наступна
    </Button>
   </div>
  </div>
 );
}

type TableLibType = {
 getState: () => any;
 setPageIndex: (pageIndex: number) => void;
 getCanPreviousPage: () => boolean;
```

```
 previousPage: () => void;
 getCanNextPage: () => boolean;
 nextPage: () => void;
 getPageCount: () => number;
 setPageSize: (pageSize: number) => void;
};

const Pagination = ({ tableLib }: { tableLib: TableLibType }) => (
 <footer className="pagination">
  <button
   disabled={!tableLib.getCanPreviousPage()}
   onClick={() => tableLib.setPageIndex(0)}
  >
   ⍰
  </button>
  <button
   disabled={!tableLib.getCanPreviousPage()}
   onClick={tableLib.previousPage}
  >
   ◀
  </button>
  <span>{`page ${
   tableLib.getState().pagination.pageIndex + 1
  } of ${tableLib.getPageCount()}`}</span>
  <button disabled={!tableLib.getCanNextPage()} onClick={tableLib.nextPage}>
   ▶
  </button>
  <button
   disabled={!tableLib.getCanNextPage()}
   onClick={() => tableLib.setPageIndex(tableLib.getPageCount() - 1)}
  >
   ⍰
  </button>
  <span>Show: </span>
  <select
   value={tableLib.getState().pagination.pageSize}
   onChange={(e) => tableLib.setPageSize(parseInt(e.target.value, 10))}
  >
   {[5, 10, 20].map((size) => (
    <option key={size} value={size}>
     {size}
    </option>
   ))}
  </select>
  <span> items per page</span>
 </footer>
);
```

```
"use client";
```

```tsx
import { zodResolver } from "@hookform/resolvers/zod";
import { useForm } from "react-hook-form";
import { z } from "zod";

import { Button } from "@/components/ui/button";
import {
 Form,
 FormControl,
 FormDescription,
 FormField,
 FormItem,
 FormLabel,
 FormMessage,
} from "@/components/ui/form";
import { Input } from "@/components/ui/input";
import { toast } from "@/components/ui/use-toast";
import { useRouter } from "next/navigation";
import { passwordValidation } from "@/lib/constants";
import { EyeIcon, EyeOffIcon, Link } from "lucide-react";
import { useState } from "react";
import {
 Select,
 SelectContent,
 SelectItem,
 SelectTrigger,
 SelectValue,
} from "@/components/ui/select";
import { createCompanyAction } from "@/actions/create-company-action";

const FormSchema = z
 .object({
  companyCode: z.coerce
   .number()
   .min(1, { message: "Код не може бути пустим" }),
  companyType: z.string(),
  email: z.string().email({
   message: "Введіть корректну Email адресу",
  }),
  password: z
   .string()
   .min(8, { message: "Має містити щонайменше 8 символів" })
   .regex(passwordValidation, {
    message: "Некоректний пароль",
   }),
  repeatPassword: z
   .string()
   .min(8, { message: "Має містити щонайменше 8 символів" }),
 })
 .refine((data) => data.password === data.repeatPassword, {
  path: ["repeatPassword"],
  message: "Паролі мають співпадати",
```

```
});

function RegisterCompanyForm() {
 const [showPassword, setShowPassword] = useState<boolean>(false);
 const [showRepeatPassword, setShowRepeatPassword] = useState<boolean>(false);

 const form = useForm<z.infer<typeof FormSchema>>({
  resolver: zodResolver(FormSchema),
  defaultValues: {
   email: "",
   password: "",
   repeatPassword: "",
  },
 });

 const router = useRouter();

 async function onSubmit(data: z.infer<typeof FormSchema>) {
  const result = await createCompanyAction(data);

  if (result?.error) {
   toast({
    variant: "destructive",
    title: "Не вдалося зареєструвати компанію",
    description: result.error,
   });
  } else {
   toast({
    title: "Реєстрація успішна",
    description: "Ви успішно зареєстрували компанію у системі",
   });
   router.push("/sign-in");
  }
 }

 return (
  <Form {...form}>
   <form className="w-2/3 space-y-6" onSubmit={form.handleSubmit(onSubmit)}>
    <FormField
     control={form.control}
     name="companyCode"
     render={({ field }) => (
      <FormItem>
       <FormLabel>Код ЄДРПОУ</FormLabel>
       <FormControl>
        <Input
         type="number"
         placeholder="Введіть код компанії"
         {...field}
        />
       </FormControl>
```

```
      <FormMessage />
     </FormItem>
    )}
   />
   <FormField
    control={form.control}
    name="companyType"
    render={({ field }) => (
     <FormItem>
      <FormLabel>Тип компанії</FormLabel>
      <Select onValueChange={field.onChange} defaultValue={field.value}>
       <FormControl>
        <SelectTrigger>
         <SelectValue placeholder="Оберіть тип компанії" />
        </SelectTrigger>
       </FormControl>
       <SelectContent>
        <SelectItem value="OSBB" className="cursor-pointer">
         OSBB
        </SelectItem>
        <SelectItem
         value="ManagingCompany"
         className="cursor-pointer"
        >
         ManagingCompany
        </SelectItem>
        <SelectItem value="CottageTown" className="cursor-pointer">
         CottageTown
        </SelectItem>
       </SelectContent>
      </Select>
      <FormMessage />
     </FormItem>
    )}
   />
   <FormField
    control={form.control}
    name="email"
    render={({ field }) => (
     <FormItem>
      <FormLabel>Email</FormLabel>
      <FormControl>
       <Input type="email" placeholder="Введіть Email" {...field} />
      </FormControl>
      <FormMessage />
     </FormItem>
    )}
   />
   <FormField
    control={form.control}
    name="password"
```

```
    render={({ field }) => (
     <FormItem>
      <FormLabel>Пароль</FormLabel>
      <FormControl>
       <div className="relative">
        <Input
         type={showPassword ? "text" : "password"}
         placeholder="Введіть Пароль"
         {...field}
        />
        <Button
         type="button"
         variant="ghost"
         size="sm"
         className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
         onClick={() => setShowPassword((prev) => !prev)}
        >
         {showPassword ? (
          <EyeIcon className="h-4 w-4" aria-hidden="true" />
         ) : (
          <EyeOffIcon className="h-4 w-4" aria-hidden="true" />
         )}
         <span className="sr-only">
          {showPassword ? "Hide password" : "Show password"}
         </span>
        </Button>
       </div>
      </FormControl>
      <FormDescription>
       Мінімум 8 символів, 1 велика, 1 мала літера та 1 цифра.
      </FormDescription>
      <FormMessage />
     </FormItem>
    )}
   />
   <FormField
    control={form.control}
    name="repeatPassword"
    render={({ field }) => (
     <FormItem>
      <FormLabel>Повторіть пароль</FormLabel>
      <FormControl>
       <div className="relative">
        <Input
         type={showRepeatPassword ? "text" : "password"}
         placeholder="Повторіть Пароль"
         {...field}
        />
        <Button
         type="button"
         variant="ghost"
```

```
          size="sm"
          className="absolute right-0 top-0 h-full px-3 py-2 hover:bg-transparent"
          onClick={() => setShowRepeatPassword((prev) => !prev)}
         >
          {showRepeatPassword ? (
           <EyeIcon className="h-4 w-4" aria-hidden="true" />
          ) : (
           <EyeOffIcon className="h-4 w-4" aria-hidden="true" />
          )}
          <span className="sr-only">
           {showRepeatPassword ? "Hide password" : "Show password"}
          </span>
         </Button>
        </div>
       </FormControl>
       <FormMessage />
      </FormItem>
     )}
    />
    <div className="w-full flex justify-center">
     <Button type="submit">Зареєструвати компанію</Button>
    </div>
   </form>
  </Form>
 );
}

export default RegisterCompanyForm;
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ: ПРЕЗЕНТАЦІЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Цифрова платформа для автоматизації обліку розрахунків у багатоквартирних будинках за допомогою Nest.JS

Виконав:
Студент групи САДМ-61
Лоленко Кирило
Керівник:
Кузьміч Михайло Юрійович
Доктор філософії (PhD), доцент кафедри Інформаційних систем та технологій

Актуальність та мета

Актуальність

- У сучасному світі цифровізація є ключовим напрямком розвитку різних галузей, включаючи житлово - комунальне господарство (ЖКГ).
- Застарілі методи обліку, такі як паперові квитанції та ручний облік, створюють проблеми: помилки, затримки, високі витрати часу.
- Жителі багатоквартирних будинків потребують зручного, прозорого та ефективного інструменту для взаємодії з керуючими компаніями.

Мета

- Створення цифрової платформи для автоматизації обліку розрахунків у багатоквартирних будинках
- Забезпечення мешканців сучасними інструментами для обліку та оплати послуг
- Спрощення комунікації між керуючою компанію і мешканцями

Проблематика та потреби

Традиційні методи

Паперові квитанції, ручне введення даних, особисті візити – це застарілі і незручні методи

Відсутність єдиної системи

Немає оптимального інтегрованого рішення для платежів, заявок, комунікації.

Ручна праця

Адміністрація витрачає багато часу на обробку платежів і створення звітів.

Архітектура системи

Бекенд

Nest.js для ефективної обробки запитів, підхід REST API для взаємодії з фронтендом.

База даних

PostgreSQL для зберігання даних, Prisma ORM для взаємодії з базою.

Фронтенд

Next.js для створення веб-сайту з серверним рендерингом сторінок та інтерактивності інтерфейсу.

nest

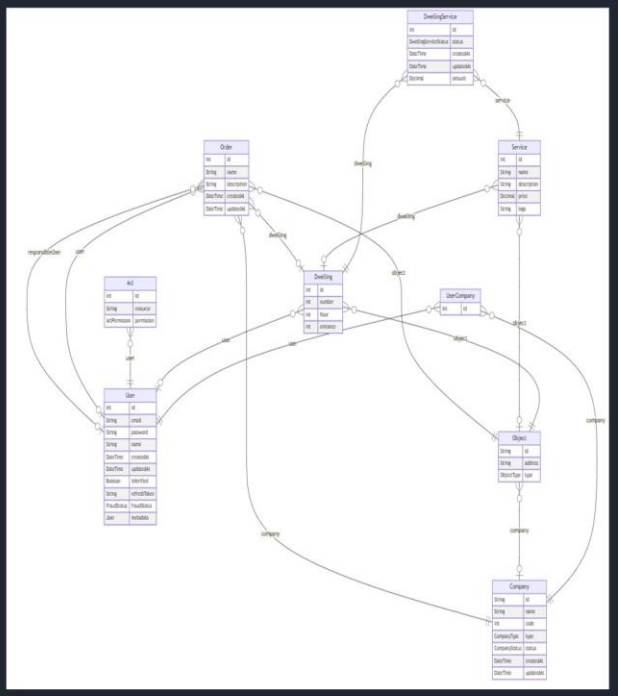
PostgreSQL

NEXT.JS

Структура бази даних

Ключові таблиці системи:

- **User (Користувачі):** Зберігає дані про мешканців, адміністрацію та їхні ролі, а також статуси верифікації.
- **Company (Компанії):** Інформація про компанії, що управляють житловим фондом (ОСББ, керуючі компанії).
- **Dwelling (Квартири):** Дані про номери, поверхи, під'їзди, а також зв'язок із мешканцями.
- **Object (Будинки):** Інформація про багатоквартирні будинки, адреси та типи об'єктів.
- **Service (Послуги):** Список послуг із тарифами та описами.
- **Order (Заявки):** Фіксує запити на ремонт або інші послуги.
- **DwellingService (Послуги у квартирах):** Відображає статуси та суми платежів за послуги.



Функціональність платформи

Облік платежів

Перегляд рахунків, історія платежів, фільтрація за датами.

Управліннязаявками

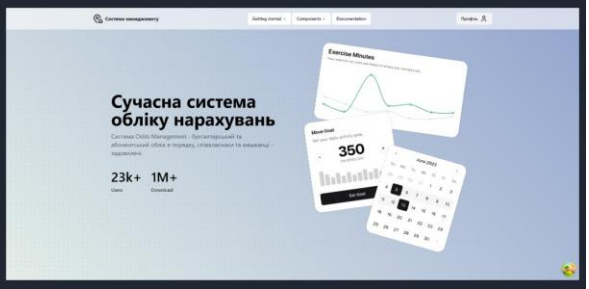
Подання заявок на ремонт, відстеження статусу виконання.

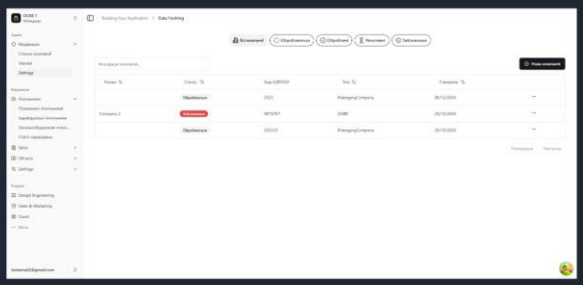
Сповіщення

Нагадування про оплату, нові рахунки, строки погашення заборгованості.

Аналітики

Візуалізація даних про витрати та споживання ресурсів, генерація звітів.





Переваги автоматизації

Для мешканців

Швидкий доступ до інформації, онлайн оплата, подання заявок через кілька кліків.

Для адміністрації

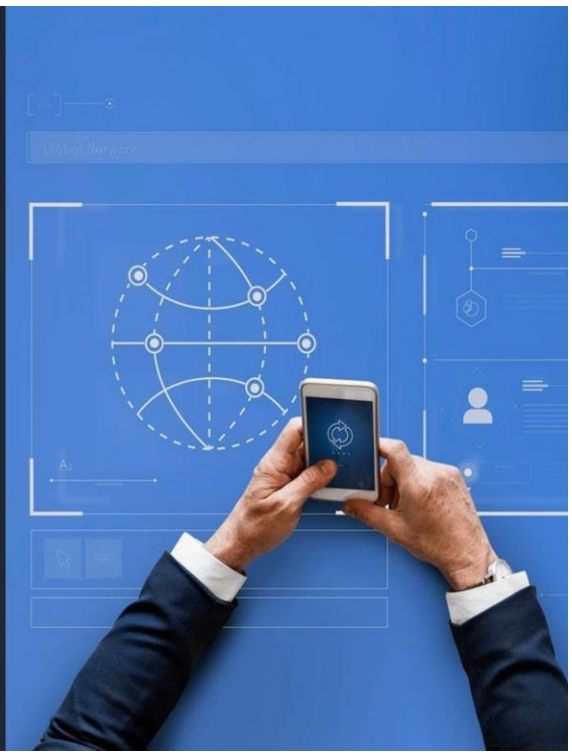
Скорочення часу на обробку заявок і платежів, прозорість фінансових операцій.

Економічний ефект

Зменшення витрат на паперовий документообіг, зниження операційних витрат.

Екологічність

Скорочення використання паперу, мінімізація викидів.



Виклики та ризики

1 — 2 — 3

Технічні виклики

Інтеграція з існуючими системами, забезпечення стабільної роботи.

Соціальні виклики

Небажання користувачів переходити на нові рішення, навчання персоналу.

Юридичні виклики

Відповідність законодавству щодо обробки персональних даних, прозорість фінансових транзакцій.

Платформа як рушій змін у ЖКГ

Покращення взаємодії:

- Зменшення проблемних питань між мешканцями та адміністрацією.
- Спрощення комунікації між усіма учасниками.

Сприяння цифровізації:

- Популяризація нових технологій у ЖКГ.
- Створення основи для інтеграції з іншими "розумними" системами, наприклад, IoT.

Довгостроковий вплив:

- Зменшення витрат і підвищення довіри мешканців до адміністрації будинків.
- Формування більш сучасного та ефективного управління житловим фондом.

Реалізація проєкту та очікувані результати

Розробка MVP (мінімально життєздатного продукту):

- Включення ключових функцій: облік платежів, автоматизовані сповіщення мешканцям про заборгованість та терміни оплати, зручна система подання заявок на послуги.
- Тестування базової версії платформи, збір зворотного зв’язку та внесення необхідних коректив.
- Розробка інтуїтивно зрозумілого інтерфейсу для користувачів з різним рівнем цифрової грамотності.

Очікуванірезультати:

- Зниження витрат на обслуговування завдяки автоматизації процесів.
- Підвищення ефективності комунікації з мешканцями, скорочення термінів розгляду заявок.
- Позитивний зворотній зв'язок після тестування, високий рівень задоволеності користувачів системою.

Висновки та майбутній розвиток платформи

У рамках виконання кваліфікаційного проєкту було досягнуто важливих результатів, які суттєво впливають на цифровізацію сфери житлово-комунального господарства. Створено цифрову платформу, яка автоматизує облік розрахунків у багатоквартирних будинках, що без сумніву вирішує ряд проблем пов'язаних із застарілими методами обліку.

Майбутній розвиток платформи спрямований на розширення її функціональності. Планується впровадження нових функцій, таких як голосування мешканців і аналітичні інструменти для прогнозування витрат та моніторингу споживання ресурсів. Також платформа адаптуватиметься до нових потреб, залежно від зворотного зв'язку від користувачів. У перспективі передбачається розширення географії використання платформи, включаючи її масштабування на інші міста та регіони.

Розроблена платформа є важливим кроком у модернізації ЖКГ в Україні. Вона спрощує управління житловими будинками, зменшує витрати, підвищує прозорість фінансових операцій і рівень задоволеності мешканців. Крім того, платформа створює надійне підґрунтя для подальшого розвитку галузі, сприяючи цифровізації та адаптації до сучасних викликів.

Дякую за увагу

Висловлюю щиру подяку за приділені увагу та час.