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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система управління орендною житловою нерухомістю»

Виконав:
студент IV курсу, групи ІС-12
Галактіонов Максим Олександрович _____

Керівник:
доцент кафедри ІСТ, к.ф.-м.н., доцент
Рибачук Людмила Віталіївна _____

Рецензент:
доцент кафедри ІПІ, к.т.н.
Олійник Юрій Олександрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

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

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Галактіонову Максиму Олександровичу

1. Тема проєкту «Інформаційна система управління орендною житловою нерухомістю», керівник проєкту Рибачук Людмила Віталіївна, к.ф.-м.н., доцент, затверджені наказом по університету від «23» травня 2025 р. № 1705-с.

2. Термін подання студентом проєкту: «9» червня 2025 року.

3. Вихідні дані до проєкту: мови програмування Python, JavaScript, бібліотека React, фрейморк FastAPI, база даних PostgreSQL, редактор програмного коду Visual Studio Code, система контролю версій Git.

4. Зміст пояснювальної записки: аналіз предметної області і існуючих рішень, вимоги до системи, аналіз технологій для розробки, технічний опис і тестування системи, математичне забезпечення.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): діаграма варіантів використання, ER-діаграма, діаграма компонентів, діаграма послідовності.

6. Дата видачі завдання: 10.03.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз предметної області	20.04.2025	
2	Аналіз готових рішень	27.04.2025	
3	Формування вимог до системи	04.05.2025	
4	Дослідження та розробка математичного забезпечення	07.05.2025	
5	Розробка серверної частини ПЗ	11.05.2025	
6	Розробка клієнтської частини ПЗ	14.05.2025	
7	Тестування ПЗ	18.05.2025	

Студент Максим ГАЛАКТІОНОВ

Керівник Людмила РИБАЧУК

АНОТАЦІЯ

Інформаційна система управління орендною житловою нерухомістю.

Проєкт містить 91 с. тексту, 3 рисунки, 15 таблиць, посилання на 15 літературні джерела, 2 додатки та 4 конструкторських документа.

АЛОРИТМ З ПРИЗНАЧЕННЯ КОЛЬОРІВ, ВЕБЗАСТОСУВАННЯ, ЖИТЛОВА НЕРУХОМОСТЬ, КЛІЄНТ-СЕРВЕРНА АРХІТЕК-ТУРА, ОРЕНДА, СИСТЕМА УПРАВЛІННЯ, УПРАВЛІННЯ ОРЕНДНИМИ ПРОЦЕСАМИ, ФІНАНСОВА СТАТИСТИКА.

Об’єктом розробки є система управління орендною житловою нерухомістю на базі вебзастосування.

Мета розробки – спрощення і автоматизація процесу управління орендною житловою нерухомістю.

У дипломному проєкті розроблено вебзастосування, що реалізує систему управління орендною нерухомістю, яка спрощує і частково автоматизує процеси призначення обов’язків для мешканців орендного житла, надання їм актуальної інформації загального характеру, створення і відслідковування їх запитів, створення і контроль над фінансовими потоками, що включать доходи і витрати. В ході роботи над дипломним проєктом було проаналізовано поточний стан ринку нерухомості, актуальний ринок сервісів і застосунків в сфері оренди. Проведено аналіз програмних засобів для реалізації API серверної частини, клієнтського інтерфейсу і бази даних. Значну уваго приділено розробці алгоритму зі створення колірного градієнту для візуалізації фінансової статистики. Було вивчено і досліджено специфіку роботи вебзастосунків на основі клієнт-серверної архітектури.

Отримані результати можуть бути використані для автоматизації схожих процесів в інших індустріях.

SUMMARY

Information system for residential rental property management.

The project contains 91 pages. text, 3 figures, 15 tables, links to 15 literary sources, 2 annexes and 4 design documents.

Keywords: color assignment algorithm, web application, residential real estate, client-server architecture, rent, management system, rental process management, financial statistics.

The object of the development is a rental housing management system based on a web application. The purpose of the development is to simplify and automate the process of managing rental residential property.

In this thesis project, a web application was developed that implements a rental property management system. This system simplifies and partially automates processes such as assigning responsibilities to tenants, providing them with up-to-date general information, creating and tracking their requests, and managing financial flows, including income and expenses. During the work on the project, an analysis was conducted of the current state of the real estate market and the relevant services and applications in the rental sector. Software tools were analyzed for implementing the backend API, the client interface, and the database. Significant attention was given to the development of an algorithm for generating a color gradient to visualize financial statistics. The specifics of web applications based on client-server architecture were also studied and explored.

The results obtained can be used to automate similar processes in other industries.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС12.080БАК.005 ПЗ	Інформаційна система управління	91		
6			орендною житловою нерухомістю.			
7			Пояснювальна записка			
8						
9	А3	ІС12.080БАК.005 Д1	Інформаційна система управління	1		
10			орендною житловою нерухомістю.			
11			Діаграма варіантів використань			
12						
13	А3	ІС12.080БАК.005 Д2	Інформаційна система управління	1		
14			орендною житловою нерухомістю.			
15			ER-діаграма			
16						
17	А3	ІС12.080БАК.005 Д3	Інформаційна система управління	1		
18			орендною житловою нерухомістю.			
19			Діаграма компонентів			
20						
21	А3	ІС12.080БАК.005 Д4	Інформаційна система управління	1		
22			орендною житловою нерухомістю.			
23			Діаграма послідовності			
24						
25						
26						
27						
28						

Зм.	Лист	№ докум.	Підпис		ІС12.080БАК.005 ТП		
Розробив		Галактіонов			Інформаційна система управління орендною житловою нерухомістю. Відомість дипломного проєкту	Літ.	Арк.	Аркушів
Перевірив		Рибачук				Т	1	1
						КПІ ім. Ігоря Сікорського Група ІС-12		
Затв.								

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система управління орендною житловою нерухомістю»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

API – Application Programming Interface.

CSS – Cascading Style Sheets.

HSL – Hue, Saturation, Lightness.

HTML – HyperText Markup Language.

HTTP – Hypertext Transfer Protocol.

JSON – JavaScript Object Notation.

REST – Representational State Transfer.

SPA – Single Page Application.

URL – Uniform Resource Locator.

ВСТУП

В сучасному світі з кожним роком все більше і більше людей живуть не у власному житлі, а в орендованому. Цьому сприяють як і достатньо відомі довгострокові процеси урбанізації, що почалися ще на наприкінці XIX ст., збільшення потоків міграції людей до окремих країн, загальне збільшення населення світу, а також значною мірою сучасні економічні проблеми, що розпочалися приблизно після 2008-року в більшості розвинених країн світу і призвели до значного подорожчання нерухомості, особливо у населених пунктах, що є важливими фінансовими і промисловими центрами [1 – 2].

Подорожчання нерухомості є проблемою, що торкається людей різного соціального складу: молодих сімей, студентів, тимчасових робітників і робітників з низьким доходом, переселенців. Результатом складного становища ринку нерухомості стає стрімке зростання популярності орендованого житла як закономірної відповіді на виклики сучасного світу. Відповідно, оренда житла стала важливою і дуже вагомою економічною діяльністю у багатьох кранах світу.

Зараз здача житла в оренду стає комплексним, багатофакторним економічним процесом. Власник нерухомості може здавати в оренду одразу декілька об’єктів, при тому одразу декільком людям. Крім того, специфікою сучасного ринку нерухомості є той факт, що одне житло можуть орендувати одразу декілька малознайомих людей, що можуть брати не себе обов’язки перед орендодавцем в рівнозначній мірі. В склад таких обов’язків можуть входити, наприклад, прибирання приміщення і сплата рахунків. Водночас, орендодавець також може мати обов’язки перед орендарям: заміна інфраструктури об’єкту, ремонт сантехніки, купівля нової електроніки, тощо. Все це значною мірою ускладнює контроль над всіма процесами, що стосуються як орендаря, так і орендарів [3].

Відповіддю на проблеми галузі оренди житла є спеціалізоване програмне забезпечення, що має мету спростити або автоматизувати контроль зазначених процесів. Однак актуальні рішення в області програмного забезпечення здебільшого фокусуються на інших напрямках оренди – зазвичай на короткостроковій оренді в

сфері туризму, а не на довгостроковій оренді постійного житла. Саме тому створена в межах дипломного проєкту інформаційна система управління орендною житловою нерухомістю є актуальною розробкою, що фокусується на базових аспектах класичної оренди основного житла, націлена на взаємодію власника житла і людей, що його орендують, а також на фінансову складову управління цим житлом як орендного об’єкту.

Отже, метою дипломного проєкту є розробка програмного рішення як інформаційної системи, що спрощує процеси управління житлом, що здається в оренду. Для досягнення цієї мети необхідно було виконати такі завдання, як аналіз предметної області і існуючих в цій області рішень, вибір програмних засобів для розробки і планування архітектури системи.

Наразі розроблена інформаційна система працює в межах сфери оренди житла, однак в майбутньому ця, або інші системи на її основі зможуть бути використані для вирішення проблем управління в таких галузях як короткострокова оренда житла і складських приміщень, оренда транспорту, спортивного чи інструментального інвентарю. Цьому сприяє архітектура розробленої системи і спосіб її реалізації.

Цей дипломний проєкт складається з наступних розділів: вступ, опис предметної області, аналіз існуючих рішень, формування вимог до системи, вибір технологій розробки, розробка інформаційної системи, математичне забезпечення, тестування ПЗ, висновки, перелік використаних джерел із 15 найменувань, двох додатків. Графічна частина включає 4 креслеників формату А3 . Загальний обсяг становить 91 сторінку.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Предметним середовищем дипломного проєкту «Інформаційна система управління орендною нерухомістю» є управління і контроль над процесами, що пов'язані з індустрією оренди нерухомості (як довгострокової, так і короткострокової), а конкретно – менеджмент процесів, що відбуваються власне після заключення контракту між орендарями та орендодавцем.

Такі процеси включать в себе відслідковування сплати регулярних і нерегулярних платежів з боку як орендарів, так і орендодавця, комунікація між орендарями та орендодавцем, метою якої є отримання актуальної інформації щодо стану нерухомості, її конкретних об'єктів чи інфраструктури, що мають відношення до поточної оренди, а також відслідковування бажань і скарг осіб, що орендують житло, задля своєчасного вирішення пов'язаних з ними проблем.

Окремо слід зазначити важливість фактору часу для зазначених процесів – важливим є не тільки сам факт сплати того чи іншого рахунку чи задоволення запитів мешканців, а ще і час, коли необхідну суму було сплачено чи запит задоволено. Це є критично важливим, оскільки в реальних умовах часові проміжки для зазначених бізнес-процесів мають суттєвий вплив на дохідність і стабільність економічної діяльності в сфері оренди житла.

Менеджмент цих процесів є рутинною, але водночас комплексною задачею, саме тому вона потребує рішень в області програмного забезпечення для її спрощення і часткової автоматизації [4].

Система, що працює в межах описаної предметної області, передбачає рольову модель взаємодії: орендодавець має доступ до конкретних взаємодій і процесів, що є недоступними до для орендарів, і навпаки.

1.1. Опис процесу діяльності

Розроблена система управління орендною нерухомістю передбачає виконання багатьох процесів, що включать в себе як власне взаємодію користувачів із

системою, так і взаємодію користувачів між собою через функціонал системи. Доступність і специфіка роботи тих чи інших процесів залежить від ролі користувача.

В системі наявні певні сценарії взаємодії/діяльності, які можна умовна поділити на категорії в залежності від типу інформаційних об’єктів, які будуть задіюватися під час їх виконання.

До сценаріїв взаємодії, пов’язаних з обліковими записами користувачів відносяться:

– реєстрація в системі нового користувача;

– авторизація у системі існуючого користувача;

– редагування інформації власного облікового запису користувача.

До сценаріїв взаємодії, пов’язаних з нерухомістю відносяться:

– перегляд об’єктів нерухомості у власності орендаря;

– додавання нових об’єктів нерухомості;

– видалення обраних об’єктів нерухомості;

– редагування інформації про обрані об’єкти нерухомості.

Діяльності, пов’язані з орендарями:

– перегляд орендарів у нерухомості;

– додавання нових орендарів до нерухомості;

– видалення орендарів з нерухомості.

Діяльності, пов’язані з транзакціями:

– перегляд транзакцій, що відносяться до конкретної нерухомості;

– додавання нових транзакцій;

– видалення обраних транзакцій;

– редагування обраних транзакцій;

– підтвердження сплати по транзакції користувачами;

– перегляд фінансової статистики нерухомості орендодавцем.

Діяльності, пов’язані з запитами орендарів:

– перегляд запитів від орендарів конкретної нерухомості;

– додавання нових запитів від орендарів;

– видалення обраних запитів від орендарів;

– редагування обраних запитів від орендарів;

– підтвердження про задоволення запитів орендарів користувачами.

Діяльності, пов’язані з оголошеннями:

– перегляд оголошень, що відносяться до конкретної нерухомості;

– додавання нових оголошень;

– редагування обраних оголошень.

Діяльності, пов’язані з обов’язками:

– перегляд обов’язків для орендарів;

– додавання нових обов’язків для орендарів;

– видалення обраних обов’язків для орендарів;

– редагування обраних обов’язків для орендарів.

Можливість виконувати певні діяльності контролюється шляхом авторизації користувача – спочатку користувач має автентифікувати себе, а потім перевіряється його роль і належність конкретній сутності, в даному випадку – його належність певній нерухомості, яку він орендує або якою володіє.

Деякі варіанти використання системи можуть бути схожими за природою і реалізацією. Наприклад, підтвердження транзакції і підтвердження запиту мешканців є досить близькими за реалізацією подіями. Це також відноситься і до діяльностей, що передбачають додавання, редагування і видалення орендарів, транзакцій, обов’язків, оголошень, запитів.

З наведених вище причин не будуть наводитися діаграми діяльностей для всіх способів взаємодії. Власне сам принцип взаємодії користувача із системою показано на прикладі діаграми діяльності на рисунку 1.1, що описує процес підтвердження транзакції, який є досить демонстративним і дає вичерпне уявлення про діяльність, пов’язану з системою в рамках предметної області.

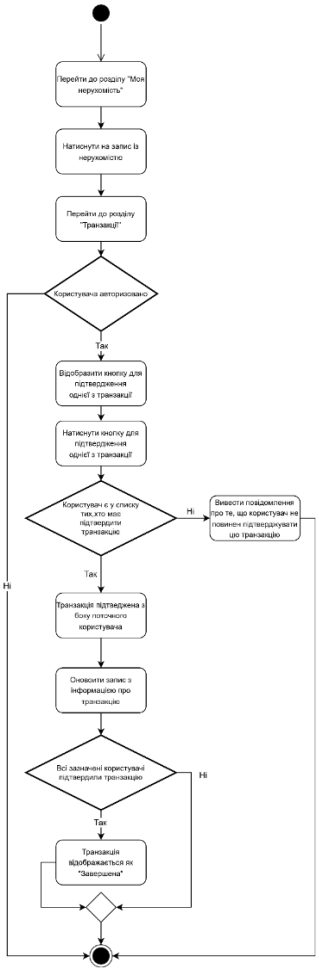

Рисунок 1.1 – Діаграма діяльності "Підтвердження транзакції"

Інші основні процеси взаємодії користувачів із системою описано у вигляді діаграм активності на рисунках Б.1 – Б.5 у додатку Б.

1.2 Постановка задачі

1.2.1 Призначення розробки

Система призначена для автоматизації і оптимізації процесів, пов’язаних з управлінням орендною житловою нерухомістю і контролем окремих показників, пов’язаних з економічною діяльністю в сфері оренди нерухомості зі сторони власника. Також система призначена для спрощення і пришвидшення процесів інформаційно-ознайомлювальної складової, що відбуваються під час оренди зі сторони мешканців.

До об’єктів автоматизації належать:

– фінансовий облік;

– відслідковування часових термінів виконання окремих процесів в контексті оренди;

– комунікація орендарів і орендодавця;

– контроль сплати витрат і рахунків;

– контроль виконання запитів орендарів;

– формування і відправка запитів орендодавцю.

1.2.2 Цілі та задачі розробки

Метою розробки є створення інформаційної системи, що спрощує процес управління житловою орендною нерухомістю її власником і процес оренди для орендарів шляхом надання обом сторонам орендних відносин зручного інтерфейсу для комунікації, відстеження і контролю пов’язаних з орендою процесів.

Для досягнення поставленої мети необхідно виконати такі задачі:

– проаналізувати потреби і проблеми учасників орендних відносин на ринку нерухомості;

– дослідити існуючі інструменти для вирішення виявлених проблем і задоволення виявлених потреб;

– дослідити існуючі рішення в області програмного забезпечення;

– визначити основний функціонал і переваги конкретних рішень в області ПЗ;

– визначити недоліки і слабкі місця розглянутих рішень в області ПЗ;

– сформувати основний принцип і напрям роботи розроблюваного ПЗ;

– визначити функціонал ПЗ і вимоги до нього;

– сформувати вимоги до системи в цілому;

– сформувати вимоги до функціональних характеристик;

– сформувати вимоги до видів забезпечення;

– спроєктувати архітектуру ПЗ;

– спроєктувати архітектуру серверної частини;

– спроєктувати архітектуру клієнтської частини;

– визначити технології і засоби, необхідні для розробки ПЗ;

– розробити програмне забезпечення відповідно до визначеної архітектури;

– провести тестування розробленого ПЗ;

– перевірити відповідність ПЗ функціональним вимогам;

– перевірити ПЗ на надійність;

– перевірити ПЗ на складову інформаційної безпеки.

Таким чином, наведений перелік послідовних завдань є основою для досягнення поставленої мети – розробки якісного і надійного програмного продукту, що спростить і оптимізує рутинний і комплексний процес управління орендною нерухомістю.

Висновки до розділу 1

В цьому розділі був проведений опис і аналіз предметної області, її окремих складових і особливостей. Була сформульована постановка задачі: визначено призначення розробки, сформульована мета розробки і послідовність задач, виконання яких є необхідною умовою для її досягнення.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

На ринку програмних продуктів існує суттєва кількість застосувань і сервісів, що працюють в сфері оренди нерухомості. Більша частина з них націлена на пошук житла для короткострокової оренди, наприклад, на час відпустки або відрядження до іншого міста або країни. Найпопулярніші з таких сервісів – Booking.com та AirBnb.

Названі сервіси мають на меті завдання задоволення короткочасного попиту на житло, більшою мірою в індустрії туризму. Звісно, вони передбачають комунікацію орендодавця і орендарів, але не призначені для вирішення питань і проблем, що виникають при довгостроковій оренді. Наприклад, там немає окремого інтерфейсу і функціоналу для надсилання запитів зі сторони орендарів (наприклад, на ремонт інфраструктури). Крім того, хоча в цих сервісах і є фінансовий облік, він досить обмежений і враховує лише дохід від нерухомості, можливість додавати і відстежувати витрати там відсутня [5 – 6].

Сервіси і застосунки для контролю різних аспектів оренди, подібні за функціоналом до дипломного проєкту, існують, але представлені в значно меншій кількості і мають ряд недоліків чи відмінностей. Нижче надано перелік деяких з них, проаналізовано їх основні переваги і недоліки.

LandlordStudio – мобільний Andriod-застосунок, призначений для обліку фінансів і зберігання та систематизованого відображення різного роду релевантної інформації, що відноситься до нерухомості. Дозволяє додавати статті доходів і витрат, інформацію про мешканців і стан житла.

Переваги:

– широкий функціонал для збору статистики;

– гнучкість налаштування;

– великий вибір типів інформації для зберігання і відображення;

– не потребує доступу в Інтернет для роботи;

– можливість створювати фінансові звіти у вигляді окремих pdf-файлів.

Недоліки:

– обмеженість однією платформою – мобільними пристроями на операційній системи Andriod;

– перевантаженість інтерфейсу великою кількістю інтерактивних елементів, деталей та анімацій;

– відсутність можливості комунікації з мешканцями, мешканці – лише абстрагований запис в локальному сховищі, а не реальні актори системи;

– вся інформація зберігається локально на пристрої, що може призвести до швидкого заповнення пам’яті та зниження продуктивності роботи на старих пристроях.

Hostaway – потужний PMS-сервіс для менеджменту нерухомості, що здається у короткочасну оренду. Дозволяє орендодавцю слідкувати і керувати бронюванням своїх об’єктів, що були зарезервовані через сервіси-партнери AirBnb та Booking.com, переглядати фінансову аналітику, створювати звіти з ефективності. Має багато додаткових модулів для налаштування реклами, автоматичних платежів, повідомлень [7].

Переваги:

– має потужні інструменти для глибокої фінансової аналітики;

– має можливість налаштування автоматичних платежів для орендарів;

– глибока інтеграція з сервісами оплати;

– зручний інтерфейс для відстежування бронювань і задач з обслуговування нерухомості;

– наявність мобільного застосунку для доступу до системи.

Недоліки:

– складний, перевантажений інтерфейс. Комплексний функціонал, що підходить лише для професіоналів в орендному бізнесі, що мають велику кількість об’єктів;

– чіткий фокус на короткострокову оренду;

– пряма залежність від сервісів AirBnb та Booking.com;

– комунікація з користувачами лише у вигляді чатів, що делегуються від зовнішніх сервісів AirBnb та Booking.com, та через email, SMS та сторонні месенджери;

– орендарі мають дуже обмежений доступ до системи: немає функціоналу для підтвердження транзакцій і створення запитів мешканцями;

– в системі може бути наявний лише один мешканець, відповідальний за бронювання, тому систему неможливо використовувати для колективної оренди.

Lodgify – ще один популярний PMS-сервіс, що є аналогом і конкурентом Hostaway. Він так само призначений для менеджменту нерухомості, що здається у короткочасну оренду. Дозволяє орендодавцю слідкувати і керувати бронюванням своїх об’єктів, що були зарезервовані через сервіси-партнери AirBnb та Booking.com, але також має можливість додавати нові об’єкти через сам сервіс. Також можна переглядати фінансову аналітику і бізнес-складові ефективності, але аналітика не така глибока, як у Hostaway [8].

Переваги:

– має зручні інструменти для достатньо просунутої фінансової аналітики;

– має можливість налаштування автоматичних повідомлень і платежів;

– зручний інтерфейс для відстежування бронювань і задач з обслуговування нерухомості;

– можливість створити власний сайт-візитівку для реклами і бронювання нерухомості;

– досить лаконічний і зрозумілий інтерфейс; низький поріг входу: застосунок простий у використанні і може підходити для використання приватними орендодавцями з невеликою кількістю об’єктів та без команди професіоналів;

– наявність мобільного застосунку для доступу до системи.

Недоліки:

– чіткий фокус на короткострокову оренду;

– комунікація з користувачами лише у вигляді чатів, що делегуються від зовнішніх сервісів AirBnb та Booking.com, та через email, SMS та сторонні месенджери;

– орендарі мають дуже обмежений доступ до системи: немає функціоналу для підтвердження транзакцій і створення запитів мешканцями;

– в системі може бути наявний лише один мешканець, відповідальний за бронювання, тому систему неможливо використовувати для колективної оренди.

Існує ще низка систем, подібних за функціоналом, серед них: Guesty, Smoobu, iGMS, Hospitable та ін. Вони мають свої особливості і унікальні характеристики, але в цілому мають схожі недоліки і фокус функціоналу, здебільшого – на фінансову складову і процес пошуку мешканців і бронювання, а не довгострокову взаємодію.

TenantCloud – сервіс пошуку орендного житла і мешканців, а також управління процесами, пов'язаними з контрактом оренди, призначений як для орендодавців, так і для мешканців. Має сильний фокус саме на пошук житла і автоматизацію процесу оформлення документів і юридичних довідок [9].

Переваги:

– сучасний, адаптивний інтерфейс;

– орендар – повноцінний актор системи;

– наявність мобільного застосунку;

– автоматизація процесів оформлення контракту і заселення мешканців, а також створення і верифікація різноманітних юридичних документів;

– широкий функціонал для створення запитів на обслуговування житла від орендарів;

– широкі можливості для автоматизації платежів і обліку фінансових потоків.

Недоліки:

– відсутня можливість підтвердження транзакцій вручну, лише після сплати через платіжні сервіси;

– відсутня можливість колективного підтвердження транзакцій і запитів декількома обраними орендарями;

– немає функціоналу для управління колективними обов'язками для мешканців.

Таким чином, розроблювана система управління орендною житловою нерухомістю є не прямим конкурентом існуючих аналогів, а займає конкретну нішу і має фокус на побутову взаємодію власника нерухомості і мешканців, що її орендують, а також на спрощення управління орендними процесами у випадку, коли житло орендують колективно декілька людей. Цим пояснюється актуальність і доцільність розробки цієї системи.

Висновки до розділу 2

В цьому розділі було розглянуто існуючі аналоги розроблюваної системи, як сервіси короткочасної оренди AirBnb, Booking.com, мобільний застосунок LandlordStudio, сервіси Lodgify, Hostaway та TenantCloud. Було проаналізовано їх функціонал, напрямок і сферу роботи, визначено їх головні переваги і недоліки. На основі отриманої про існуючі рішення інформації обґрунтовано доцільність і актуальність розробки системи в межах дипломного проєкту.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

У розроблюваній системі виокремлено такі підсистеми, що забезпечують основу її функціоналу:

– система автентифікації;

– система авторизації;

– база даних для зберігання користувацької та іншої релевантної інформації;

– серверна частина та API;

– клієнтська частина, користувацький інтерфейс для взаємодії;

– система збереження стану поточного користувача на клієнтській стороні.

Основною вимогою до перелічених підсистем є їх взаємозв’язок і послідовність в їх роботі, що передбачає постійну взаємодію. Кожен компонент системи загалом є самодостатнім в межах свого цільового призначення та конкретно визначеного функціоналу, але є залежним від інших компонентів як складова частина розроблюваної системи в цілому.

Система автентифікації відповідає за процеси підтвердження особистості користувача, що необхідно для надання користувачу доступу до системи загалом, а також до певного функціоналу. Автентифікація є невід’ємною складовою авторизації.

Система авторизації відповідає за надання автентифікованому користувачу прав на доступ до окремого функціоналу системи. В даному проєкті авторизація є фундаментом для рольової взаємодії в системі з такими акторами як «Орендодавець» і «Орендар», відповідно кожен з акторів має доступ лише де певних варіантів взаємодії в системі, що визначені предметною областю, а авторизація підтримує цю чітку рольову взаємодію.

Сховище даних є важливим компонентом майже будь-якого сучасного застосунку, сховища можуть бути різних типів і мати різне призначення. Оскільки більшість систем передбачають маніпуляцію інформацією, що має бути збережена на час більший, ніж одна користувацька сесія, спосіб збереження даних і її отримання має ключове значення для розробки. Його важливість і сильний зв'язок з іншими компонентами дозволяє вважати його повноцінною підсистемою в межах інфраструктури програмного продукту. В даному дипломному проєкті використовується реляційна база даних, де зберігається особиста інформація про користувачів та дані, що представляють об'єкт прямого інтересу користувачів в контексті орендних відносин, а саме – записи транзакцій, запитів, оголошення, інформація про житло і мешканців.

Серверна частина в межах проєкту «Інформаційна система управління житловою орендною нерухомістю» є ключовою підсистемою, що відповідає за основну логіку обробки клієнтських запитів, взаємодію з базою даних, передачу коректної і персоналізованої інформації, отриманої з бази даних, до клієнтської сторони. Для серверної частини було розроблено API, що надає інтерфейс взаємодії для клієнтської сторони у вигляді HTTP-запитів через визначені на серверній частині адреси і процедури обробки доступу до них.

Підсистема клієнтської сторони відповідає за взаємодію кінцевого користувача із системою. Вона регулює те, яким чином буде отримуватися інформація від серверу, як вона буде оброблюватися і як буде подаватися користувачу. Крім того, вона також відповідає за первинну обробку інформації, отриманої від користувача, та її відправку до серверу. Клієнтська сторона також визначає загальний вигляд інтерфейсу, наданого користувачу, включаючи дизайн і стильове оформлення, що забезпечують доступний, зрозумілий і вичерпний доступ до інформації. В розробленій системі клієнтська сторона формується на основі вебзастосунку, що має бути доступний з будь-якого сучасного браузеру.

Система збереження стану поточного користувача є досить невеликим компонентом, що відповідає за тимчасове збереження деякої ключової інформації про

поточного користувача системи на клієнтській стороні, що дозволяє уникнути зайвих запитів до серверу на час однієї сесії.

Інформаційна система управління житловою орендною нерухомістю передбачає використання двома акторами: «Орендарем» та «Орендодавцем». Основною вимогою щодо варіантів використання системи є розподілення і розмежування взаємодій всередині системи. Користувач-орендар не повинен мати доступ до функціоналу, передбаченого виключно для користувача-орендаря, і навпаки. Відмінності в доступі мають забезпечуватися авторизацією, а несанкціонований доступ до конкретного фунціоналу блокуватися на серверній стороні. Крім того, ще однією вимогою є надання різних візуальних вказівок щодо доступності певних функцій різним користувачам за допомогою динамічного користувацького інтерфейсу. Це може досягатися відображенням тих чи інших візуальних елементів в залежності від ролі авторизованого користувача.

Структура системи надає міцний фундамент для впровадження нових компонентів. Існуючий функціонал може бути значно розширений, а рольову модель доповнено. Зокрема, в майбутньому можна запровадити інтеграцію з платіжними і банківськими системами для автоматизації певних транзакцій та їх підтвердження, можливість додавати і підписувати договір оренди між сторонами. Рольову модель може бути доповнено такими акторами, як «Ріелтор», «Бухгалтер» та «Спеціаліст з обслуговування інфраструктури», які також можуть виступати в якості учасників процесів економічної і управлінської діяльності під час оренди, відповідно бути окремими користувачами системи з різними ролями і відповідними їхній спеціалізації можливостями для взаємодії. Користувацький інтерфейс також може бути значно покращений: можуть бути додані нові, більш вичерпні варіанти відображення ключової інформації, можливості для її фільтрації і сортування. Тобто, можна впевнено затвердити, що створена інформаційна система має реальні перспективи для подальшого розвитку.

3.1.2 Вимоги до надійності

Система повинна опрацьовувати деякі аварійні та непередбачувані ситуації таким чином, щоб забезпечувалась надійність роботи системи в цілому і безпека користувацьких даних. Аварійні або просто проблемні ситуації можуть бути різного характеру і виникати з різних причин. Всі можливі випадки передбачити неможливо, однак основні, найпоширеніші проблемі ситуації мають послідовно оброблюватися системою.

До вірогідних помилок і збоїв під час роботи системи можна віднести:

– втрата підключення до бази даних;

– проблеми з інтернет-з'єднанням користувача;

– програмні помилки і виключні ситуації під час роботи серверу;

– фізичне пошкодження серверу;

– користувач намагається отримати інформацію, що не відноситься до нього;

– особисті дані користувача було викрадено.

Задля надійного функціонування системи до процесу опрацювання наведених випадків висуваються вимоги.

При втраті підключення до бази даних, сервер повинен просигналізувати про відсутність можливості доєднатися до бази даних, спровокувати виключну ситуації і спробувати обробити її. Після цього має бути проведена повторна спроба підключення. Якщо підключення не було відновлено, сервер має сповістити клієнтську сторону про виникнення технічних проблем. Всі помилки повинні логуватися.

Якщо у швидкість інтернет-з’єднання на стороні користувача було знижено, система все одно повинна постачати користувачу необхідну йому інформацію і можливість взаємодії з більшою часовою затримкою. Якщо Інтернет-з’єднання було втрачено, користувач повинен мати змогу маніпулювати даними, що були отримані до втрати з’єднання, на клієнтській стороні до тих пір, поки поточна ситуація не зміниться.

Якщо на серверній частині виникли помилки або виключні ситуації, якомога більша частина з них має бути оброблена серверною частиною задля впровадження безперервної роботи. Якщо виключну ситуацію не вдалось обробити, клієнтська сторона має бути проінформована про це – користувач має отримати повідомлення про наявність помилки і її характер, без деталізації технічної складової. Виключні ситуації повинні логуватися. Якщо клієнтська сторона не може отримати доступу до ресурсів серверу, користувач повинен мати змогу маніпулювати даними, що були отримані до моменту виникнення проблеми, поки поточна ситуація не зміниться.

Для уникнення ситуацій втрати інформаційної складової серверу і його налаштувань у випадку його фізичного пошкодження, вся логіка роботи серверу має бути збережена на сторонньому надійному сховищі, хмарному сервісі. Серверний програмний код повинен систематизуватися і відслідковуватися системою контролю версій.

Ситуації, коли користувач намагається отримати дані, що не відносяться до його облікового запису (наприклад, отримати дані інших користувачів) мають опрацьовуватися на сервері шляхом автентифікації та авторизації. Між клієнтом і базою даних повинен існувати прошарок у вигляді API, що має гарантувати безпеку від несанкціонованого доступу до чутливої інформації шляхом надання строго керованого інтерфейсу для отримання даних.

Для випадків, коли особисті дані користувача були отримані зловмисником, або коли доступ до облікового запису було отримано небажаною стороною, користувач повинен мати можливість видалити свій акаунт, тобто стерти свої дані з системи і створити новий акаунт.

До надійності технічних засобів та програмного забезпечення також висуваються вимоги:

– система повинна бути розгорнута на надійних серверних потужностях із можливістю резервування;

– програмне забезпечення, що використовується системою, повинно бути актуальним і мати дійсну на поточний час команду розробників для його підтримки;

– технічні засоби розробки повинні мати інструменти для відлагодження помилок і тестування;

– програмне забезпечення має бути протестоване на витривалість і стійкість до збоїв.

3.1.3. Вимоги до збереження інформації

Зберігання інформації в системі повинно забезпечуватися програмно-технічними засобами, що мають інструменти для резервного копіювання та протидії аварійним ситуаціям і несанкціонованому доступу до бази даних. Це є основною вимогою до засобів збереження інформації, оскільки в система постійно оперує даними, що зберігаються і дістаються зі сховища, а також чутливими даними, пов’язаними з особою користувача.

Оскільки система має бути в достатній степені надійною, а інформація є ключовим об’єктом роботи системи, ще однією вимогою до системи є збереження користувацької інформації при виникненні наступних проблемних ситуацій:

– збій у роботі фізичного серверу;

– збої в серверному програмному забезпеченні (структурні і логічні помилки програмного коду);

– збої в клієнтському програмному забезпеченні (структурні і логічні помилки програмного коду);

– переривання з'єднання з базою даних;

– потрапляння до серверу необроблених, некоректних даних;

– потрапляння до сховища даних некоректних записів.

Таким чином, виконання наведених вимог гарантує як якісний користувацький досвід роботи з системою, так і безпеку цілісності важливих даних.

3.2 Вимоги до функціональних характеристик

Перелік функцій та вимоги до якості реалізації кожної з них, до форми представлення вихідної інформації, характеристики необхідної точності та часу виконання наведені в таблиці 3.1.

Таблиця 3.1 – Функціональні вимоги

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
Реєстрація	Користувача зареєстровано в системі; користувач може обрати ім’я, email та пароль	Точність виконання: висока; час виконання: 10-15 секунд	Користувачу надається форма для реєстрації; користувачу надається повідомлення про успішну чи невдалу реєстрацію
Вхід у систему (Аутентифікація)	Якщо користувач був зареєстрований, він може увійти в систему; незареєстрований користувач не може отримати доступ до системи	Точність виконання: висока; час виконання: 10-15 секунд	Користувачу надається форма для входу у систему; користувачу надається повідомлення про успішний чи невдалий вхід у систему
Вихід із системи	Користувач виходить з системи; після виходу доступ до системи обмежується	Точність виконання: висока; час виконання: 5-10 секунд	Для виходу із системи надається відповідна кнопка на сторінці облікового запису

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
Перегляд нерухомості, що управляється	Користувачу надається список нерухомості, що знаходиться під його управлінням; для кожного запису з нерухомістю відображається його назва, місцеположення і опис; функція доступна лише користувачу з роллю «Орендодавець»	Точність виконання: середня; час виконання: 3-5 секунд	Список нерухомості можна прокручувати; список коректно відображається на мобільних пристроях; кожний запис з нерухомістю у списку відображено як інтерактивну картку
Вибір нерухомості для управління	Користувачу надається можливість обрати нерухомість зі списку для переходу на панель управління, що відноситься до обраної нерухомості	Точність виконання: висока; час виконання: 5-7 секунд	Натискання на картку з нерухомістю зі списку нерухомості переносить користувача на панель управління, що містить весь функціонал для управління списками мешканців, транзакцій, запитів, обов'язків і оголошень;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
			Списки мешканців, транзакцій, запитів, обов'язків і оголошень відображаються у виділеній зоні; перехід між списками виконується переключенням вкладок; Оновлення і зміна списків не викликає оновлення сторінки
Перегляд мешканців в обраній нерухомості	Користувачу надається список мешканців, що орендують обрану нерухомість; для кожного запису з мешканцем відображається його ім'я, email та терміни оренди	Точність виконання: висока; час виконання: 3-5 секунд	Список мешканців можна прокручувати; список коректно відображається на мобільних пристроях; Кожний запис з мешканцем у списку відображено як картку з інформацією
Додати мешканця до нерухомості	Надається можливість додати мешканця (орендаря)	Точність виконання: висока;	У списку мешканців обраного об'єкту не-

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	до обраного об’єкту нерухомості, використовуючи унікальний код запрошення, призначений кожному користувачу-орендарю; функція доступна лише користувачам з роллю «Орендодавець»	час виконання: 5-7 секунд	рухомості існує зелена кнопка, натискання на яку відкриває поле для введення коду-запрошення; якщо користувача з введеним кодом не було знайдено, виводиться відповідне повідомлення; після додавання мешканця, список оновлюється
Видалити мешканця з нерухомості	Надається можливість видалити мешканця (орендаря) з обраного об’єкту нерухомості; функція доступна лише користувачам з роллю «Орендодавець»	Точність виконання: висока; час виконання: 2-3 секунди	У списку мешканців обраного об’єкту нерухомості у правому верхньому куті кожної картки з мешканцем є червона кнопка для видалення відповідного мешканця;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
			система потребує від користувача підтвердження видалення мешканця
Перегляд нерухомості, що орендується	Користувачу надається список нерухомості, яку він орендує; для кожного запису з нерухомістю відображається його назва, місцеположення і опис; функція доступна лише користувачу з роллю «Орендар»	Точність виконання: середня; час виконання: 3-5 секунд	Список нерухомості можна прокручувати; список коректно відображається на мобільних пристроях; кожний запис з нерухомістю у списку відображено як інтерактивну картку
Перегляд транзакцій	Користувачу надається список транзакцій, що відносяться до обраної нерухомості; транзакції з маркером «лише для орендодавця» не відо-	Точність виконання: висока; час виконання: 3-5 секунд	Список транзакцій можна прокручувати; список коректно відображається на мобільних пристроях; кожний запис з транзакцією у списку відображено як інтерактивну картку;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	бражаються орандарям обраної нерухомості; для кожного запису з транзакцією відображається тип транзакції, її сума, дата сплати, кількість днів до сплати, список користувачів, що мають підтвердити її сплату		якщо транзакцію підтвердили всі обрані користувачі, картка з транзакцією змінює свій колір на зелений; якщо дата сплати на момент перегляду вже пройшла, на картці відображається червоний надпис «Overdue»
Перегляд запитів	Користувачу надається список запитів від орендарів, що відноситься до обраної нерухомості; для кожного запису з запитом відображається текст запиту, дата його створення, список користувачів, що	Точність виконання: висока; час виконання: 3-5 секунд	Список запитів можна прокручувати; список коректно відображається на мобільних пристроях; кожний запис з запитом у списку відображено як інтерактивну картку; якщо всі обрані користувачі підтвердили виконання за-

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	мають підтвердити його виконання		питу, картка з запитом змінює свій колір на зелений; якщо дата сплати на момент перегляду вже пройшла, на картці відображається червоний надпис «Overdue»
Перегляд обов’язків	Користувачу надається список обов’язків для орендарів, що відноситься до обраної нерухомості; для кожного запису з обов’язком відображається назва обов’язку і його опис, дата його створення	Точність виконання: середня; час виконання: 3-5 секунд	Список обов’язків можна прокручувати; список коректно відображається на мобільних пристроях; кожний запис з обов’язком у списку відображено як інтерактивну картку
Перегляд оголошень	Користувачу надається список оголошень, що відноситься до обраної нерухомості;	Точність виконання: середня; час виконання: 3-5 секунд	Список оголошень можна прокручувати;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	для кожного запису з оголошенням відображається тема оголошення і його текст		список коректно відображається на мобільних пристроях; кожний запис з обов'язком у списку відображено як інтерактивну картку
Підтвердження транзакції, запиту	Користувач може підтвердити певну транзакцію або запит у списку транзакцій або запитів, якщо він є у відповідному списку користувачів, відповідальних за підтвердження	Точність виконання: висока; час виконання: 2-3 секунди	Якщо користувач є в списку, на картці з транзакцією або запитом наявна кнопка з написом «Підтвердити транзакцію», що відповідає за процес підтвердження; користувач отримує повідомлення про успішне підтвердження; поруч з ім'ям користувача на картці з транзакцією або запитом стоїть маркер, що відображає пото-

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
			чний статус підтвердження цим користувачем
Додавання нового інформаційного запису до нерухомості: транзакція, обов'язок, запит, оголошення	Користувачу надається можливість додати новий запис з транзакцією, обов'язком, запитом, оголошенням до відповідного списку; користувачу надається можливість вказати всю інформацію, що відноситься до запису; користувач має вказати всю обов'язкову для конкретного запису інформацію; додавати нові обов'язки і транзакції може лише користувач з роллю «Орендодавець»;	Точність виконання: висока; час виконання: 2-3 секунди	У списку, що відповідає конкретним об'єктам у верхній частині наявна кнопка, що дозволяє відкрити форму для додавання нового запису; форма дозволяє заповнити всю необхідну для запису інформацію і додати цей запис до списку шляхом натискання на відповідну кнопку знизу форми; користувач отримує повідомлення, якщо не заповнив всі обов'язкові поля форми;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	додавати нові запити може лише користувач з роллю «Орендар»		після того як користувач відправив форму, список із записами оновлюється
Редагування інформаційного запису для обраної нерухомості: транзакція, обов’язок, запит, оголошення	Користувачу надається можливість редагувати існуючий запис з транзакцією, обов’язком, запитом, оголошенням з відповідного списку; користувачу надається можливість змінити всю інформацію, що відноситься до обраного запису; користувач має вказати всю обов’язкову для конкретного запису інформацію;	Точність виконання: висока; час виконання: 2-3 секунди	При натисканні на картку, що відповідає конкретному запису у списку відкривається форму для редагування; форма дозволяє змінити всю релевантну для запису інформацію і підтвердити зміни шляхом натискання на відповідну кнопку знизу форми; користувач отримує повідомлення, якщо не заповнив всі обов’язкові поля форми;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	редагувати обов’язки і транзакції може лише користувач з роллю «Орендодавець»; редагувати запити може лише користувач з роллю «Орендар»		після того як користувач відправив форму, список із записами оновлюється
Видалення інформаційного запису для обраної нерухомості: транзакція, обов’язок, запит, оголошення	Користувачу надається можливість видалити існуючий запис з транзакцією, обов’язком, запитом, оголошенням з відповідного списку; видаляти обов’язки і транзакції може лише користувач з роллю «Орендодавець»;	Точність виконання: висока; час виконання: 2-3 секунди	У верхньому правому куті кожної картки, що відповідає конкретному запису у списку, є кнопка для видалення відповідного їй запису; якщо користувач не має право видаляти запис, кнопка для видалення не відображається;

Функція	Загальна вимога до результату	Вимога до якісних характеристик	Вимоги до форми представлення інформації
	видаляти запити може лише користувач з роллю «Орендар»		система потребує від користувача підтвердження видалення запису; після того як користувач видалив запис, відповідний список оновлюється

3.3 Вимоги до видів забезпечення

3.3.1 Вимоги до математичного забезпечення

Для розробки програмного продукту в межах проєкту інформаційної системи управління орендною житловою нерухомістю дозволяється використовувати різноманітні математичні моделі і алгоритми. Спосіб роботи алгоритмів з пам’яттю в даному проєкті не обмежується, дозволяється використання алгоритмів сортування і модифікації структур даних, що працюють як зі статичною, так і з динамічною пам’яттю, з тимчасовими даними та постійними.

Основною вимогою щодо алгоритмів при розробці є першочергове використання вбудованих алгоритмів, що постачаються з програмним забезпеченням нативно, якщо технічна реалізація того чи іншого функціоналу не потребує розробки нових алгоритмів і методів. Це пояснюється тим, що алгоритми, що постачаються програмними засобами та математичними модулями з існуючою підтримкою, відрізняються високою ефективністю та високим рівнем оптимізації. Крім того, їх використання спрощує розробку і підтримку програмного продукту, його подальше розширення і модифікацію.

В розроблюваній системі вимагається не використовувати стохастичні типові алгоритми, такі як генетичні і ройові алгоритми, методи відпалу та ін. Концепція роботи таких алгоритмів не відповідає предметній області і меті розроблюваного продукту. Такі алгоритми можуть давати неочікуваний і негарантований результат, а використання значних часових і обчислювальних ресурсів погіршувати користувацький досвід.

Вимогою щодо розроблюваних алгоритмів є заборона на використання новітніх і експериментальних концепцій і методів, ефективність яких ще не була перевірена, а можливість їх використання в реальних продуктах не протестовано. Це може призвести до отримання неочікуваних результатів, зниження продуктивності роботи системи і ускладнення її підтримки.

3.3.2 Вимоги до інформаційного забезпечення

Інформаційна система оперує даними як даними, що мають прямо відношення до сфери оренди, так і персональними даними, що відносяться до користувачів. Для цих даних визначається чітка структура, що відповідає за їх технічне використання в системі і пряму користь для користувача як об’єкт інтересу.

В системі основні операції відбуваються з даними такого складу:

– загальні данні користувачів;

– дані складової автентифікації;

– дані про об’єкти нерухомості;

– дані про орендарів;

– дані про фінансові операції / транзакції;

– дані про запити орендарів;

– дані про обов’язки орендарів.

Оскільки в системі передбачається активна маніпуляція даними та комунікація між клієнтською та серверною стороною, структурна організація цих даних має базуватися на концепції сутностей або об’єктів. Ця концепція відповідає сучасним методологіям розробки і спрощує управління даними в системі, особливо під час

розробки. Структури мають містити лише ту інформацію, що необхідна для забезпечення описаного функціоналу системи. Вони мають бути вичерпні в контексті розробки системи, але в той самий час не обтяжувати їх підтримку зайвими факторами. Відповідно, основну структуру даних як сутностей з ключовими описовими характеристиками, що визначають їх суть і зв'язок між ними, наведено в таблиці 3.2 в якості вимог до структури даних.

Таблиця 3.2 – Вимоги до структури даних

Вид даних	Структура
Данні користувачів	Ключові складові: – унікальний номер користувача; – ім'я; – email; – пароль; – роль; – код запрошення.
Данні для автентифікації	Ключові складові: – зашифрований ключ; – зашифрована роль.
Дані про об'єкт нерухомості	Ключові складові: – унікальний номер об'єкту нерухомості; – назва; – місцеположення; – опис; – унікальний номер орендодавця об'єкту.
Данні про транзакцію	Ключові складові: – унікальний номер транзакції; – унікальний номер об'єкту нерухомості, до якої відноситься транзакція; – тип (назва);

Вид даних	Структура
	– сума; – дата сплати; – сплатник (орендар / орендодавець); – видимість для орендарів (так / ні).
Дані про запит орендаря	Ключові складові: – унікальний номер запиту; – унікальний номер відправника запиту (орендаря); – унікальний номер об’єкту нерухомості, до якої відноситься запит; – назва; – опис; – дата створення.
Дані про обов’язки орендарів	Ключові складові: – унікальний номер обов’язку; – унікальний номер об’єкту нерухомості, до якої відноситься обов’язок; – назва; – опис; – дата виконання обов’язку.
Дані про належність мешканців орендним відносинам	Ключові складові: – унікальний номер даних про належність; – унікальний номер користувача, що орендує нерухомість; – унікальний номер об’єкту нерухомості, що орендується; – дата початку оренди; – дата закінчення оренди.

Вид даних	Структура
Дані про підтвердження транзакції	Ключові складові: – унікальний номер підтвердження; – унікальний номер транзакції, що підтверджується; – унікальний номер користувача, що має підтвердити транзакцію; – поточний статус транзакції (підтверджено, очікується).
Дані про підтвердження виконання запиту	Ключові складові: – унікальний номер підтвердження; – унікальний номер запиту, виконання якого підтверджується. – унікальний номер користувача, що має підтвердити запит; – поточний статус запиту (підтверджено, очікується).

Перелічені вище структури, звісно, повинні зберігатися в системі на тому чи іншому етапі роботи, щоб досягти бажаної якості потоку роботи застосунку. В розроблюваному програмному продукті дані зберігатимуться як на стороні клієнта в певні проміжки часу, так і у повноцінній базі даних.

Дані, що мають зберігатися довгостроково, повинні зберігатися в базі даних. Враховуючи цілі дипломного проєкту і специфіку роботи застосунку, використовувана база даних має бути реляційною. В базі даних інформація повинна зберігатися в таблицях, що відповідатимуть сутностям, описаним в таблиці 3.2. Ключова інформація зберігається в рядках, склад яких відповідає описаній структурі.

Дані, що отримуються з бази даних первинно потрапляють до серверу, з яким комунікує клієнтська сторона. Оскільки пряме з’єднання з базою даних виконується зі сторони серверу, структури для збереження даних повинні бути пристосованими для зберігання інформації також типу і складу, як і інформації з БД. Крім

того, вони повинні мати інтерфейс для взаємодії з ними на основі об’єктної структури, дублюючи склад таблиць реляційної бази даних. Ще одна вимога до організації інформації на сервері відноситься до способу отримання і подальшого зберігання даних: використовувані структури повинні мати прямий інтерфейс для взаємодії з БД і забезпечувати серіалізацію і переміщення інформації від БД до серверу без кроків, що включать зайве ресурсозатратне високорівневе перетворення типів даних. Однією з вимог до використовуваних методів збереження даних на сервері є можливість прямої, швидкої і, за можливості, автоматичної серіалізації до типів даних, що використовуються клієнтською стороною.

На стороні клієнта інформація, що отримується від сервера, повинна зберігатися в структурах даних, що постачаються програмними засобами, в даному випадку – мовою програмування або фреймворком / бібліотекою. Це можуть бути як масиви, списки, так і інші логічні структури. Головною вимогою для таких структур для збереження інформації є швидкий доступ до контенту і можливість динамічного перезапису інформації, оптимізація для постійних змін і оновлень, оскільки на стороні клієнта будуть проводитися значні і часті маніпуляції з цими структурами, що будуть необхідні для сортувань, фільтрацій і відображення у інтерфейсі. Ще однією вимогою до таких способів збереження на клієнтській стороні є можливість визначити об’єктну структуру, що відповідає об’єктам, що постачаються з серверу. Це спрощує роботу з даними на клієнтській частині і робить програмний код зрозумілішим, що спрощує підтримку і масштабування.

До систем управління базами даних також висуваються конкретні вимоги:

– СУБД є реляційною;

– СУБД повинна підтримувати транзакції, реплікацію та резервне копіювання;

– повинна забезпечувати цілісність та консистентність даних;

– має бути доступною у хмарному середовищі для масштабування системи.

– у серверному програмному забезпеченні повинні бути інструменти для інтеграції з обраною СУБД;

– повинна мати інструменти для візуалізації сутностей бази даних.

3.3.3 Вимоги до програмного забезпечення

До програмного забезпечення системи висувається ряд вимог. В першу чергу, всі компоненти програмного забезпечення, що використовуються у розробці, повинні відповідати умовам ліцензування, які дозволяють використання у навчальних або некомерційних цілях, але водночас мати можливість для переведення статусу проєкту, що використовує це забезпечення, в комерційний в майбутньому. Це стосується як і засобів розробки, таких як інтегровані середовища програмування і інструменти розробника, так і окремих функціональних складових продукту, такі як бібліотеки, модулі та фреймворки.

До мови програмування, що використовується для розробки основної логіки на backend-частині проєкту, висуваються такі вимоги:

– наявність бібліотек і модулів для створення гнучкого і швидкого API;

– наявність широкого спектру модулів і бібліотек для перетворення типів даних, серіалізації;

– наявність інструментів для роботи з віддаленою базою даних і встановлення прямого підключення до неї;

– наявність оптимізованих і гнучких структур даних для зберігання і роботи з даними;

– мова повинна відповідати парадигмі об’єктно-орієнтованого програмування;

– версія мови програмування повинна бути актуальною і мати поточну підтримку від розробників, компанії або підтримуватися у форматі open-source;

– мова повинна мати простий, лаконічний синтаксис.

Основні процеси, що відбуваються на клієнтській частині системи, включать запити до API, обробку даних, отриманих від користувача і серверу, роботу з інтерфейсом. Відповідно мова, що використовується для програмування клієнтської частини, повинна відповідати таким вимогам:

– наявність широкого спектру модулів і бібліотек для перетворення типів даних, серіалізації;

– наявність засобів для маніпуляції структурою розмітки вебсторінок та таблицями каскадних стилів;

– наявність можливості для хостингу клієнтської частини на окремому сервері;

– наявність фреймворків для впровадження чіткої структури організації файлів в процесі розробки клієнтської частини;

– наявність фреймворків і бібліотек для впровадження динамічних елементів до користувацького інтерфейсу;

– наявність оптимізованих модулів і бібліотек для створення HTTP-запитів до API;

– наявність засобів для обробки і збереження даних, що надходять від користувача;

– наявність оптимізованих і гнучких структур даних для зберігання і роботи з даними;

– мова повинна відповідати парадигмі об’єктно-орієнтованого програмування;

– версія мови програмування повинна бути актуальною і мати поточну підтримку від розробників, компанії або підтримуватися у форматі open-source;

– мова повинна мати простий, лаконічний синтаксис.

Інтегроване середовище програмування, яке виступає як основний інструмент для розробки і відповідає за такі процеси як написання програмного коду і відлагодження, також повинне відповідати основним вимогам. Серед них є візуальне форматування програмного коду на основі синтаксису (включаючи сигналізацію про синтаксичні помилки), інтеграція з засобами контролю версій, можливість генерації вичерпних звітів про наявні в програмному коді помилки, наявність інструментів для спрощення розробки вебзастосунків, наявність засобів для контролю і редагування файлової структури проєкту.

Система контролю версій, що використовується під час розробки, повинна мати фунціонал для створення окремих версійних відгалужень, можливість для ко-

лаборації з іншими розробниками. Також система контролю версій повинна передбачати наявність хмарного сервісу для зберігання проєкту і ознайомлення з ним іншими людьми.

3.3.4 Вимоги до технічного забезпечення

В рамках вимог до технічного забезпечення наводяться мінімально необхідні характеристики електро-обчислювальної машини для запуску і роботи двох основних складових системи: backend-серверу, та серверу, що поставлятиме клієнтську частину. Також наводяться вимоги до технічного складу кінцевого пристрою користувача, на якому буде відбуватися пряма експлуатація системи і вимоги до пристрою розробки, на якому можна запустити всі компоненти системи локально з метою розробки, відлагодження, тестування і ознайомлення з принципом роботи системи. Ці вимоги наведені нижче у таблиці 3.3.

Таблиця 3.3 – Вимоги до технічного забезпечення пристроїв системи

Технічний пристрій	Мінімальні технічні вимоги до пристрою
Backend-сервер	CPU: 1 vCPU / одноядерний 2.0 GHz Оперативна пам’ять: 1 GB Диск: 10 GB SSD Мережа: 100 Mbps Операційна система: Ubuntu 20.04 LTS або аналогічний Linux-дистрибутив
Frontend-сервер	CPU: 1 vCPU / одноядерний 1.8 GHz Оперативна пам’ять: 512 MB Диск: 5 GB SSD Мережа: 100 Mbps Операційна система: будь-яка Linux-подібна система або Docker-хостинг.

Технічний пристрій	Мінімальні технічні вимоги до пристрою
Клієнтський пристрій кінцевого користувача	Браузер: Chrome 90+, Firefox 85+, Safari 14+, Edge 90+ CPU: двоядерний 1.5 GHz Оперативна пам’ять: 2 GB Мережа: 5 Mbps (завантаження) / 1 Mbps (віддача) Роздільність екрану: 720×1280 px Операційна система: Windows 7-10 / macOS 10.15 / будь-яка сучасна Linux-дистрибуція / Android 9 (Pie) / iOS 13
Пристрій для розробки і локального розгортання	CPU: двоядерний 1.5 GHz Оперативна пам’ять: 4 GB (рекомендовано 8 GB) Диск: 20 GB вільного місця на SSD/HDD (для коду, залежностей, бази даних) Операційна система: Windows 10 / macOS 10.15+ / будь-яка сучасна Linux-дистрибуція

Висновки до розділу 3

В цьому розділі було сформульовано вимоги до системи в цілому: до її структури і функціональних можливостей, до надійності і до способів збереження інформації в різних умовах. Було сформовано вимоги до функціональних характеристик системи: перелічено всі можливі функції системи і вимоги до їх результату їх виконання, їх якості і форми представлення пов’язаних з ними складових. Сформовано також вимоги до всіх видів забезпечення інформаційної системи: математичного, інформаційного, програмного і технічного.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

Відповідно до сформованих у розділі 3 вимог, обґрунтовується вибір технологій розробки інформаційної системи управління орендною житловою нерухомістю.

4.1 Вибір технологій для розробки серверної частини

Основною мовою програмування для серверної частини було обрано Python. Python – сучасна інтерпретована мова програмування, яка застосовується в багатьох напрямках інформаційних технологій, включаючи аналіз даних, створення нейронних мереж, імітаційне моделювання і, що важливо для розроблюваного проєкту, – веброзробка. Python вже давно використовується для створення серверного коду для багатьох застосунків різного призначення [10].

Основною перевагою мови програмування Python є наявність великої кількості користувацьких бібліотек, що використовуються для вирішення широкого спектру задач в області інформаційних технологій. Серед наявних бібліотек існують рішення, що вирішують проблему створення гнучкого і ефективного API для систем з архітектурою типу «клієнт-сервер» і для роботи з базою даних. Це і є головною причиною вибору мови Python як основного компоненту backend-частини проєкту [11].

До інших переваг, що вплинули на вибір технології, можна віднести:

– найпростіший для розуміння і читання коду синтаксис серед наявних аналогів;

– наявність значної кількості інструментів для роботи з математичними операціями і великими даними;

– універсальність: може використовуватися як для програмування у функціональному стилі, так і в парадигмі ООП;

– можливість використання як динамічної, так і строгої типізації даних;

– наявність широкого інструментарію для відлагодження і тестування: автоматичні тести, функціонал для перехоплення і обробки помилок, бібліотеки для аналізу швидкодії і безпеки програмного коду;

– кросплатформеність: використовується для розробки на операційних системах Windows, Linux і MacOS;

– велика спільнота користувачів-розробників, що забезпечує вичерпну доступність інформації, включаючи форуми, інструкції, документацію.

До недоліків мови можна віднести відносно низьку швидкодію в порівнянні з іншими технологіями, що можуть використовуватися для серверного програмного забезпечення, такими як C#, PHP, Node.js. Однак цей недолік значною мірою компенсується підходом, що часто використовується для створення бібліотек на Python – кореневий функціонал більшості бібліотек розроблено на низькорівневих, високошвидкісних мовах, таких як C і C++.

Важливою складовою розробки інформаційної системи на основі вебзастосунку є вибір бібліотеки або фреймворку для налагодження серверної інфраструктури і створення API. В рамках дипломного проєкту було обрано відносно новий Python-фреймворк FastAPI. Не дивлячись на те, що фреймворк з’явився на ринку фреймворків відносно нещодавно, він вже встиг зарекомендувати себе як надійний, а головне – швидкий фреймворк для розробки RESTful API. Головною перевагою FastAPI є його швидкість в порівнянні з іншими популярними фреймворками, такими як Django та Flask, що досягається завдяки ASGI (Asynchronous Server Gateway Interface), що забезпечує високу продуктивність і асинхронність застосунку. Крім того, FastAPI відомий простотою розробки – він надає зручний і не перевантажений інтерфейс для налагодження кінцевих точок маршрутів API, обробки запитів і налаштування системи автентифікації. Архітектура FastAPI-застосунків легко масштабується [12].

Однією з важливих особливостей FastAPI є глибока інтеграція з такими бібліотеками як SQLAlchemy та Pydantic. SQLAlchemy – це бібліотека для роботи з реляційними СУБД, що працює на базі ORM (Object–relational mapping). Фактично

вона дозволяє створити репрезентацію таблиць створеної бази даних у вигляді спеціалізованих моделей як об’єктів мови програмування Python, надаючи розробнику зручний інтерфейс для взаємодії з сутностями БД. SQLAlchemy дозволяє не тільки визначати структуру таблиць у серверному коді, але і проводити стандартні маніпуляції з БД через визначені моделі, такі як пошук і вибір даних, їх фільтрація, видалення і додавання записів і навіть створення нових таблиць. Функціонал, що постачається бібліотечними класами, дозволяє проводити вказані маніпуляції без необхідності включення текстових фрагментів традиційних SQL-запитів у програмний код, що значно підвищує загальну безпеку системи, запобігаючи SQL-ін’єкціям. Використання SQLAlchemy спрощує масштабування проєкту, покращує читабельність коду і дозволяє контролювати структуру даних, що отримуються з БД.

Pydantic в свою чергу є бібліотекою для валідації даних. Pydantic також дозволяє створювати об’єкти на основі бібліотечних класів з потрібною структурою. Однак ці об’єкти використовуються для валідації даних, що надходять з бази даних, зі сторони клієнта, або надходять з серверу до клієнта. Тобто бібліотека дозволяє бути впевненим, що дані, які переміщуються між компонентами системи мають чітко визначену структуру і що ніяких важливих фрагментів інформації не буде втрачено в процесі.

У SQLAlchemy та Pydantic є недолік. Часто моделі, що визначаються для валідації даних і для роботи з таблицею БД мають майже однакову структуру, що провокує дублювання коду, зменшуючи його читабельність і ускладнюючи підтримку. Саме тому в дипломному проєкті використовується бібліотека SQLModel, що вирішує цю проблему. SQLModel працює як «надбудова» над об’єктами SQLAlchemy та Pydantic, дозволяючи поєднати функціонал і переваги обох бібліотек в одному класі. Таким чином, SQLModel обирається як основна бібліотека для взаємодії з базою даних і валідації користувацьких даних на стороні серверу в розроблюваній системі.

Для автентифікації і авторизації користувача, тобто для контрольованого обмеження певних ресурсів API для клієнтської сторони, використовується стандартний протокол OAuth 2.0. Цей протокол було обрано, оскільки він є надійним і розповсюдженим методом запровадження авторизації для вебзастосунків. FastAPI реалізує цей протокол завдяки спеціальній схемі, що надається класом OAuth2PasswordBearer. Відповідно, OAuth2 працює за принципами підпису і перевірки автентичності так званих токенів, або JWT (JavaScript Web Token), які використовуються для кодування інформації про користувача. JWT токени – це безпечний і простий спосіб організації передачі даних від клієнта до серверу і навпаки для автентифікації і авторизації користувача, а FastAPI в свою чергу надає зручний інструментарій для роботи з ними. Саме тому метод OAuth2 було обрано як основу для впровадження безпеки і цілісності даних в системі.

4.2 Вибір СУБД

Як реляційну СУБД для проєкту системи управління орендною нерухомістю було обрано PostgreSQL. Основна причина – глибока інтеграція фреймворку FastAPI саме з цією СУБД. Методи і класи для роботи з БД, що надаються нативними модулями FastAPI та SQLModel, оптимізовано для роботи з PostgreSQL. PostgreSQL доцільно використовувати для систем з сутностями, які мають чітку виражену структуру. В розроблюваному проєкті наявні такі сутності: користувачі, транзакції, нерухомість, оголошення і обов’язки [13].

Крім цього, PostgreSQL відома своїми перевагами:

– надійність і масштабованість: висока стабільність при роботі з транзакціями та зв’язками між таблицями;

– підтримка складних запитів і перевірок: наприклад, CHECK, FOREIGN KEY, UNIQUE;

– створення власних типів даних, агрегатних функцій, операторів;

– реалізує повноцінні ACID-властивості, а також підтримує MVCC (Multi-Version Concurrency Control) для одночасної роботи багатьох користувачів без блокувань.

– можливість створення CTE (WITH-запити), віконних функцій, рекурсивних запити і матеріалізованих подань.

– гнучке налаштування індексів, що дає змогу покращити продуктивність у складних запитах.

PostgreSQL також показує високу ефективність роботи у складі хмарних сервісів. Відповідно, для розгортання БД в рамках проєкту було обрано безкоштовний сервіс хмарний сервіс Supabase. Це дозволяє уникнути необхідності розгортання БД на локальній машині, відповідно – спрощує розробку і зменшує навантаження на процесор під час тестування програмного забезпечення. Використання хостингу Supabase надає також фундамент для розширення проєкту майбутньому. Крім того, Supabase надає корисний і зручний інструментарій для роботи з базою і її налаштування, забезпечує стабільне з'єднання з базою даних, а також надає функціонал для візуалізації сутностей БД, що суттєво спрощує процес проектування структури таблиць.

4.3 Вибір технологій для розробки клієнтської сторони

Основною мовою програмування логіки клієнтської частини і взаємодії було обрано JavaScript як стандарт сучасної веброзробки. На основі JavaScript написано безліч фреймворків, що спрощують розробку frontend-складової вебзастосунку, особливо – створення динамічного інтерфейсу, роутинг та взаємодію з API.

Для токенів запровадження системи автентифікації і авторизації JWT-токени, що створюються і перевіряються на сервері, зберігаються локальному сховищі браузеру і включаються у HTTP-запити, що надходять до API, в спеціальному заголовку.

Як головний фреймворк, який визначає структуру клієнтської частини інформаційної системи, обрано React.

React – сучасний фреймворк з відкритим вихідним кодом, розроблений компанією Facebook, який використовується для створення користувацьких інтерфейсів. Сьогодні він активно підтримується і розвивається великою спільнотою розробників. Основна ідея React – створення користувацького інтерфейсу на базі реактивних, незалежних компонентів зі своїм специфічним функціоналом засобами JavaScript [14].

Вибір React як фреймворку для розробки обумовлено такими перевагами і особливостями:

– компонентна структура: інтерфейс розділяється на спеціалізовані частини, що можуть використовуватися повторно, такі як форми, стилістичні компоненти для відображення, таблиці і списки, які оголошуються в окремих JavaScript-файлах;

– компоненти підтримують реактивні стани, тобто компонент може реагувати відповідним чином на дії користувача і змінювати свій поточний стан;

– компоненту структуру легко підтримувати і масштабувати;

– віртуальний DOM: оновлення UI виконується швидко і без перезавантаження сторінки, що забезпечує кращий досвід користувача;

– розвинена технічна екосистема – велика кількість додаткових бібліотек для розширення функціоналу, наприклад, для маршрутизації сторінок, валідації даних і формування запитів;

– JSX-синтаксис (JavaScript XML), який дозволяє писати HTML-подібний синтаксис прямо у JavaScript-функціях, що надає гнучкість у створенні динамічних компонентів.

Окрім React для розробки клієнтської частини також використовуються допоміжні бібліотеки: Axios та React Router.

React Roter – бібліотека, що постачається від самого фреймворку React. Дозволяє емулювати класичну маршрутизацію HTML-сторінок через адресу URL, але з ключовими React-компонентами, які у реактивному застосунку виступають замість HTML-файлів зі статичною розміткою. Має лаконічний і читабельний синтаксис для оголошення маршрутів.

Axios – бібліотека для створення HTTP-запитів до API, яка надає маж більш гнучкий функціонал в порівнянні зі стандартною бібліотекою JavaScript Fetch. Axios спрощує синтаксис в порівнянні з Fetch, а головною його перевагою є автоматична серіалізація даних, отриманих від серверу, у формат JSON. Крім того, він надає зручний інструментарій для обробки помилок, встановлення глобальних заголовків, перехоплювачів і тайм-аутів.

4.4 Вибір інструментів розробника

Ключовою складовою якості кінцевого програмного продукту є вибір правильного інструментарію для розробки, тестування і налаштування системи.

Оскільки інформаційна система розроблюється на базі вебзастосунку, середовищем програмування було обрано Visual Studio Code – популярний, кросплатформенний редактор коду від Microsoft, відомий своєю зручністю для розробки вебсервісів.

Visual Studio Code впроваджує високий ступінь інтеграції з основними frontend-фреймворками, такими як Vue.js та React.js, що критично важливо, оскільки для розробки клієнтської частини використовується React. Це включає як і контекстуальне форматування програмного коду для кращої читабельності коду на основі синтаксису мови чи бібліотеки, так і засоби для ідентифікації і обробки помилок. VS Code має вбудований менеджер терміналів, що дозволяє контролювати віртуальні середовища, запускати і керувати локальними серверами, відслідковувати помилки в одному місці, незалежно від типу компонентів, з яких складається система. VS Code має також менеджер файлового середовища проєкту, який дозволяє переглядати, редагувати, видаляти файли з проєкту, відслідковувати зміни в них і переключатися між ними. Однією зі значних переваг VS Code в порівнянні з іншими редакторами коду – наявність великої кількості користувацьких модулів для гнучкого налаштування редактору і розширення його функціоналу, відповідно до потреб проєкту. Серед таких модулів, наприклад, використовувався Live Server для автоматичного оновлення сторінки під час розробки інтерфейсу.

Як система контролю версій в проєкті використовується Git – універсальне рішення для проєктів будь-якого масштабу і призначення. Git має весь основний функціонал для варіювання програмного продукту: можливість створювати версійні відгалуження чи навпаки поєднувати різні версії продукту, відслідковувати кожну зміну в програмному коді кожним розробником, відкатуватися до попередніх версій. Система Git також надає сервіс для зберігання вихідного коду продукту у хмарі для командної роботи з іншими розробниками та його публікації у відкритий доступ через репозиторії – GitHub. Розповсюдженість, універсальність і потужність Git та GitHub відіграють ключову роль у виборі цих інструментів для розробки дипломного проєкту.

4.5 Короткий перелік обраних технологій

Технології, вибір яких для розробки інформаційної системи було обґрунтовано в попередніх підпунктах, наведено у таблиці 4.1 із вказанням складових системи, в яких вони використовуються та їхнім призначенням.

Таблиця 4.1 – Обрані технології

Складова системи	Призначення	Обрана технологія
Серверна частина	Основна мова програмування	Python
	Фреймворк для створення API	FastAPI
	Бібліотека для взаємодії з таблицями БД на базі моделей, валідації даних	SQLModel на основі SQLAlchemy та Pydantic
	Автентифікація і авторизація користувачів	OAuth2, Java Web Token
База даних	СУБД	PostgreSQL
	Сервіс для розгортання бази даних	Supabase

Складова системи	Призначення	Обрана технологія
Клієнтська частина	Основна мова програмування	JavaScript, HTML, CSS
	Фреймворк для створення користувацького інтерфейсу	React
	Бібліотека для налаштування маршрутизації	Rect Router
	Бібліотека для створення HTTP-запитів	Axios
	Автентифікація і авторизація користувачів	Java Web Token, локальне сховище JavaScript
Інструменти розробки	Середовище програмування	Visual Studio Code
	Система контролю версій	Git
	Хмарний сервіс системи контролю версій	GitHub

Висновки до розділу 4

В цьому розділі було обґрунтовано вибір таких технологій для розробки інформаційної системи управління орендною нерухомістю, як мови програмування Python, JavaScript, фреймворки FastAPI та React, бібліотеки SQLModel, React-router та Axios, а також СУБД PostgreSQL і хмарний сервіс для її розгортання Supabase. Було розглянуто особливості і переваги зазначених технологій, які забезпечують доцільність їх вибору.

5 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Структура системи

Розроблювана система складається з таких трьох ключових компонентів як серверна частина, база даних і клієнтська частина, що формують її загальну структуру.

До складових серверної частини належать:

– API на базі фреймворку FastAPI;

– програмні моделі для взаємодії з БД на базі SQLModel;

– контроль сесій для з'єднання з БД;

– системи авторизації і автентифікації через JWT.

До складових бази даних на основі PostgreSQL належать:

– розгорнута на хмарному сервісі Supabase БД;

– таблиці із логічними зв'язками для інформаційного забезпечення системи.

До складових клієнтської частини належать:

– вебзастосунок на основі фреймворку React;

– окремі React-компоненти як основа логіки і вигляду інтерфейсу;

– сторінки браузерного застосунку на основі React-компонентів та навігація між ними;

– HTTP-запити до ресурсів API;

– контроль облікового запису користувача.

Відповідно до поставлених у пункті 3.1.1. вимог до структури і підсистем, було створено технічну реалізацію описаних взаємозв'язаних компонентів, що формують систему управління орендною нерухомістю.

5.2 Функціональна модель системи

Система передбачає різні варіанти використання в залежності від ролі актора: «Орендар» та «Орендодавець». Ці варіанти використання відображено у вигляді діаграми варіантів використання на кресленику ІС12.08БАК.005 Д1.

Для актора-орендодавця передбачається функціонал, пов’язаний з цільовою діяльністю управління своєю нерухомістю. Це включає в себе можливість додавати, видаляти і редагувати об’єкти нерухомості, так само як і додавати до них користувачів, що виступають акторами-орендарями. Також надається можливість додавати, редагувати і видаляти обов’язки для орендарів конкретної нерухомості та фінансові потоки (транзакції), створювати, редагувати і видаляти оголошення. Також цей актор може переглядати фінансову статистику для кожного об’єкту своєї нерухомості.

Для актора-орендаря передбачена взаємодія з орендарем шляхом запитів, орендар може створювати, редагувати і видаляти запити. Орендарі можуть переглядати транзакції в межах об’єкту, до якого вони були додані, але лише якщо транзакція була помічена як «видима для орендарів».

Як орендар, так і орендодавець можуть переглядати такі об’єкти цільового інтересу системи як обов’язки, транзакції, оголошення і запити. Орендарі і орендодавці можуть підтверджувати виконання запитів і транзакцій, якщо їх було додано до списку тих користувачів, що повинні їх підтвердити.

5.2 Модель бази даних

Реляційна база даних, яка використовується системою, має такі пов’язані між собою таблиці:

– users;

– rental_properties;

– tenancies;

– announcements;

– responsibilities;

– transactions;

– tenant_requests;

– transaction_resolutions;

– request_resolutions.

ER-діаграму бази даних, що відображає структуру цих таблиць і зв’язки між ними, зображено на кресленику ІС12.08БАК.005 Д2.

Таблиця tenancies виступає зв’язна таблиця для вирішення проблеми зв’язку «багато-до-багатьох» між користувачами з таблиці users та об’єктами нерухомості з таблиці properties. Зв’язок «багато-до-багатьох» мав би місце бути в розробленій БД, оскільки відповідно до предметної області користувач-орендар може належати декільком об’єктам нерухомості (тобто орендувати декілька об’єктів). Таким чином, таблиця tenancies містить посилання на користувачів-орендарів та орендною нерухомість – фактично містить id з таблиці users та id з таблиці properties.

Таблиці transaction_resolutions та request_resolutions містять інформацію про підтвердження транзакцій та запитів орендарів відповідно. Таблиці містять поле status, що може приймати лише два значення: pending та resolved. Якщо status для запису з цих таблиць приймає значення pending, то це означає, що певну транзакцію або запит ще не було підтверджено певним користувачем. Якщо status має значення resolved, то це означає, що користувач транзакцію чи запит вже підтвердив.

5.3 Передавання та обробка даних

Вхідні та вихідні дані, якими оперує система, залежать від конкретної дії, яку виконує користувач під час взаємодії з системою. З цієї причини далі буде описано вхідні та вихідні дані системи в контексті дій, передбачених системою. Вхідні та вихідні дані, що наведені в цьому розділі, описують взаємодію кінцевого користувача з системою, тобто фактично описується обмін інформацію між клієнтською стороною та сервером. Обмін даними між сервером та БД в свою чергу вважається внутрішньою взаємодією в середині системи.

5.3.1 Вхідні дані

Вхідні дані, які система приймає в ході виконання окремих дій наведені в таблиці 5.1.

Таблиця 5.1 – Вхідні дані системи

Вид взаємодії	Вхідні дані
Реєстрація / вхід	email, пароль
Створення об'єкта нерухомості	JWT-токен користувача, назва, місцезнаходження, опис
Додавання орендаря до об’єкту нерухомості	JWT-токен користувача, код запрошення користувача
Створення / редагування транзакції	JWT-токен користувача, сума, тип, дата, роль платника, список користувачів, що мають підтвердити виконання транзакції
Створення / редагування запиту орендаря	JWT-токен користувача, назва, опис, дата, список користувачів, що мають підтвердити виконання запиту
Створення / редагування відповідальностей	JWT-токен користувача, заголовок, опис
Створення / редагування оголошень	JWT-токен користувача, заголовок, текст
Підтвердження транзакції	JWT-токен користувача

5.3.2 Вихідні дані

Вихідні дані, отримувані користувачем в ході взаємодії з системою наведені в таблиці 5.2.

Таблиця 5.2 – Вихідні дані системи

Вид взаємодії	Вихідні дані
Реєстрація / вхід в систему	JWT-токен користувача
Перегляд об’єктів нерухомості	Список об’єктів нерухомості, для кожного об’єкту: назва, опис, локація

Вид взаємодії	Вихідні дані
Перегляд орендарів нерухомості	Список орендарів нерухомості, для кожного орендаря: ім’я, email
Перегляд транзакцій	Список транзакцій, для кожної транзакції: сума, тип, статус
Перегляд запитів	Список запитів, для кожного запиту: заголовок, опис, статус
Перегляд оголошень	Список оголошень для нерухомості, для кожного оголошення: заголовок, текст, дата створення
Перегляд обов’язків	Список обов’язків для мешканців нерухомості, для кожного обов’язку: заголовок, опис
Перегляд фінансової статистики	Місячний баланс, діаграма витрат і доходів на основі типів транзакцій

5.3.1 Опис обробки та передачі даних

На серверній частині маніпулювання даними, що витягуються з бази даних і отримуються від клієнта, відбуваються за допомогою стандартних списків Python (клас List), класів Pydantic та SQLModel, що визначають структуру специфічних користувацьких даних, таких як транзакції чи користувачі. Однак коли ці дані передаються клієнту, вони серіалізуються в універсальний формат JSON і в такому форматі надаються клієнту.

На клієнтській частині JSON-дані, що отримано від серверу, перетворюються у звичайний JavaScript-об’єкт, а з цими об’єктами вже працює код React-компонентів. З цими об’єктами можна проводити всі необхідні для клієнтської частини маніпуляції: доступ до полів, їх динамічна зміна, утворення масивів і наборів «ключ-значення», арифметичні операції з чисельними полями.

JavaScript Web Token, що передається з кожним запитом користувача у HTTP-заголовку для підтвердження його автентичності, фактично представляє собою довгу строку в звичайному текстовому форматі, що містить певну ключову

інформацію. Ця строка дешифрується на сервері за допомогою спеціалізованих алгоритмів – таким чином відбувається підтвердження автентичності токену.

5.4 Архітектура програмного забезпечення

5.4.1 Загальний опис архітектури

Архітектуру інформаційної системи побудовано на основі моделі «клієнт-сервер». Це надійний і безпечний спосіб організації архітектури систем різного призначення, який дуже часто використовується для розробки систем на базі вебзастосувань [15].

Модель працює за таким принципом: клієнт надсилає запит до сервера (наприклад, відкриття веб-сторінки), сервер приймає цей запит, обробляє його і відправляє відповідь клієнту (це може бути як HTML-файл, так і текстові або чисельні дані різних форматів), клієнт в свою чергу приймає ці дані, оброблює їх на своїй стороні і відображує кінцевому користувачу. Обмін запитами і відповідями регулюються протоколами взаємодій мережі Інтернет.

Як основу для архітектури проєкту таку модель було обрано з таких міркувань:

– розповсюдженість, наявність великої кількості теоретичних матеріалів;

– простота реалізації і доцільність для систем на базі вебзастосувань;

– наявність широкого інструментарію для впровадження;

– централізоване управління даними;

– легкість обслуговування (оновлення на сервері одразу для всіх клієнтів);

– підтримка великої кількості клієнтів.

В дипломному проєкті архітектуру реалізовано шляхом створення API засобами FastAPI для серверу, який надає інтерфейс для відправки запитів до цього серверу з клієнтської частини. Клієнт в даному випадку – це пристрій, який надсилає запити до серверу через мережу Інтернет, звертаючись до спеціальних адрес, визначених на сервері. Виконання запитів за цими адресами і обробка відповідей від серверу виконується за допомогою frontend-застосунку на базі React.

Вебінтерфейс як React-застосунок також збирається на сервері, а користувач отримує лише HTML-файл та JavaScript-файл, що відповідає за логіку інтерфейсу і його зовнішній вигляд. Frontend-частину даного проєкту було реалізовано як SPA (Single Page Application), тобто клієнтський пристрій одразу отримує один html-файл, розмітка якого динамічно змінюється JavaScript-кодом. У розробленій системі коли клієнту необхідно отримати інформацію від серверу і змінити зміст сторінки в залежності від цієї інформації, клієнт запитує від серверу лише текстові і чисельні дані у форматі JSON, які оброблюються і відображаються JavaScript-кодом у браузері. Таким чином відсікається необхідність постійного запиту цілісних HTML, JavaScript та CSS файлів від серверу.

Як JSON-дані від backend серверу, так і html-файлі можуть отримуватися з одного серверу. Однак архітектура розробленої системи з React-застосунком передбачає використання двох серверів: backend-сервер для основної логіки, такої як авторизація, доступ до ресурсів і взаємодія з БД, та окремий frontend-сервер, який відповідає лише за надання користувачу вебінтерфейсу для взаємодії з API, що працює на backend-сервері. Такий підхід дозволяє зменшити навантаження на backend-сервер і спрощує підтримку інфраструктури системи.

Схему архітектури розробленої інформаційної системи зображено на рисунку 5.1.

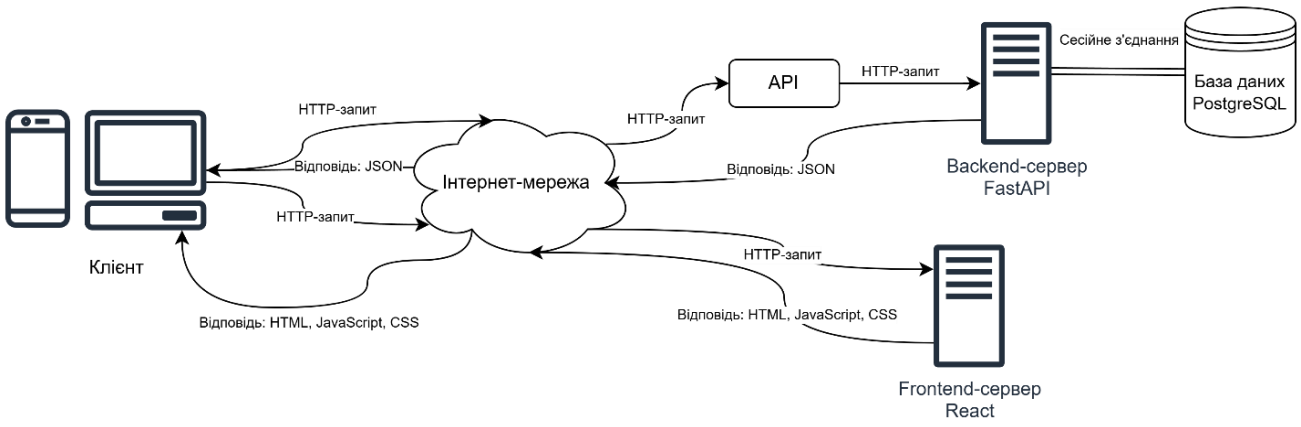

Рисунок 5.1 – Схема архітектури системи

Відповідно до моделі «клієнт-сервер» для backend-серверу доречно створювати API за архітектурним стилем REST (REpresentational State Transfer). Згідно цього архітектурного стилю, вся взаємодія побудована навколо ресурсів, доступ до яких виконується через унікальні URL-адреси. Ресурсами в розробленій системі виступають дані про користувачів, об’єкти нерухомості, оголошення, транзакції, обов’язки та запити користувачів, які надсилаються як відповідь від серверу у єдиному форматі – в даному випадку у форматі JSON. Це дозволяє запровадити передбачувану взаємодію між компонентами системи через надання єдиного, стандартизованого інтерфейсу.

5.4.2 Специфікація функцій серверної частини

Більшість розроблених користувацьких функцій на серверній частині використовуються як обробники запитів на маршрути API, також наявні функції для створення і автентифікації токенів. Специфікацію цих розроблених функцій backend-частини наведено у таблиці 5.3.

Таблиця 5.3. – Специфікація функцій серверної частини

Назва функції	Параметри	Опис дії
hash_password	password: str	Хешує пароль за допомогою bcrypt
authenticate_user	email: str, password: str, session: Session	Перевіряє автентичність користувача
create_access_token	data: dict, expires_delta: Optional[timedelta] = None	Створює JWT токен з даними та часом дії
get_current_user	token: str = Depends(oauth2_scheme), session: Session	Декодує токен і повертає поточного користувача або помилку

Назва функції	Параметри	Опис дії
	= Depends(get_session)	
get_tenants_for_property	property_id: int, session: Session, current_user: User	Повертає всіх орендарів для заданої нерухомості
register_user	user: User, session: Session	Реєструє нового користувача після перевірки email та хешування пароля
login	form_data: OAuth2PasswordRequestForm, session: Session	Автентифікує користувача та повертає токен доступу
read_users_me	current_user: User	Повертає дані поточного автентифікованого користувача
regenerate_invite_code	session: Session, current_user: User	Генерує новий інвайт-код для користувача та зберігає його в базі даних
get_rental_properties	session: Session, current_user: User	Повертає список об'єктів нерухомості для поточного користувача
get_property	property_id: int, session: Session, current_user: User	Повертає дані про обраний об'єкт нерухомості
add_property	property: RentalProperty, session: Session, current_user: User	Додає новий об'єкт нерухомості від імені орендодавця

Назва функції	Параметри	Опис дії
delete_property	property_id: int, session: Session, current_user: User	Видаляє об'єкт нерухомості, якщо він належить поточному орендодавцю
add_tenant_to_property	property_id: int, invite_code: str, session: Session, current_user: User	Додає орендаря до об'єкта нерухомості за кодом запрошення
remove_tenant_from_property	property_id: int, tenant_id: int, session: Session, current_user: User	Видаляє орендаря з об'єкта нерухомості
leave_property	property_id: int, session: Session, current_user: User	Орендар залишає об'єкт нерухомості
get_transactions	property_id: int, session, current_user	Отримує транзакції для вказаного об'єкта з урахуванням ролі користувача
get_resolved_transactions	session, current_user	Повертає всі транзакції для об'єктів, що належать орендодавцю, якщо вони повністю підтверджені
get_transaction_resolutions	transaction_id: int, session, current_user	Отримує всі резолюції транзакції з інформацією про користувачів
create_transaction	property_id: int, transaction: Transaction, session, current_user	Створює нову транзакцію для вказаного об'єкта

Назва функції	Параметри	Опис дії
update_transaction	transaction_id: int, updated_transaction: Transaction, session, current_user	Оновлює існуючу транзакцію, якщо користувач є власником
delete_transaction	transaction_id: int, session, current_user	Видаляє транзакцію та всі пов’язані з нею резолюції
add_transaction_resolution	resolution: TransactionResolution, session, current_user	Додає нову резолюцію для транзакції та користувача
remove_transaction_resolution	transaction_id: int, user_id: int, session, current_user	Видаляє резолюцію транзакції для конкретного користувача
resolve_transaction	transaction_id: int, session, current_user	Перемикає статус резолюції користувача між "pending" і "resolved"
get_tenant_requests	property_id: int, session, current_user	Отримує всі запити орендарів для заданої нерухомості
get_request_resolutions	request_id: int, session, current_user	Отримує всі резолюції запиту разом з іменем та роллю користувача
add_tenant_request	property_id: int, tenant_request: TenantRequest, session, current_user	Додає новий запит орендаря на об'єкт
update_tenant_request	request_id: int, updated_request:	Оновлює існуючий запит орендаря

Назва функції	Параметри	Опис дії
	TenantRequest, session, current_user	
delete_tenant_request	request_id: int, session, current_user	Видаляє запит орендаря разом з його резолюціями
add_request_resolution	resolution: RequestResolution, session, current_user	Додає нову резолюцію до запиту, якщо вона ще не існує
remove_request_resolution	request_id: int, user_id: int, session, current_user	Видаляє резолюцію користувача до запиту
resolve_tenant_request	request_id: int, session, current_user	Змінює статус резолюції на "resolved" або "pending"
get_announcements	property_id: int, session, current_user	Отримує всі оголошення для вказаного об'єкта
add_announcement	property_id: int, announcement: Announcement, session, current_user	Додає нове оголошення від орендодавця до об'єкта
update_announcement	announcement_id: int, updated_announcement: Announcement, session, current_user	Оновлює існуюче оголошення, якщо воно належить орендодавцю
delete_announcement	announcement_id: int, session, current_user	Видаляє оголошення, якщо воно належить орендодавцю
get_responsibilities	property_id: int, session, current_user	Повертає список обов’язків для заданої нерухомості

Назва функції	Параметри	Опис дії
add_responsibility	property_id: int, responsibility: Responsibility, session, current_user	Додає новий обов’язок для нерухомості (тільки для орендодавця)
update_responsibility	responsibility_id: int, updated_responsibility: Responsibility, session, current_user	Оновлює існуючий обов’язок (тільки для орендодавця)
delete_responsibility	responsibility_id: int, session, current_user	Видаляє обов’язок (тільки для орендодавця)

5.4.3 Специфікація компонентів інтерфейсу клієнтської частини

Як було описано вище, клієнтська частина представляє собою React-застосунок, що використовує для побудови об’єктів інтерфейсу так звані React-компоненти, кожен з яких описується в окремому JavaScript-файлі. Ці компоненти є багатофункціональними і містять програмний код для маніпулювання інтерфейсом, обробки і надсилання запитів, валідації даних. Специфікацію наявних в системі React-компонентів наведено у таблиці 5.4.

Таблиця 5.4 – Специфікація компонентів інтерфейсу клієнтської частини

Назва компоненту	Опис
Navbar	Відображає навігаційну панель з посиланнями на різні сторінки
Home	Головна сторінка, показує список нерухомості залежно від ролі
Profile	Сторінка профілю користувача з можливістю входу/виходу з системи, реєстрації та оновлення коду запрошення

Назва компоненту	Опис
StatisticsPanel	Панель статистики з діаграмами доходів та витрат за місяць
PropertyPanel	Панель управління конкретною нерухомістю з вкладками
ManagementTabs	Вкладки для управління орендарями, обов'язками, платежами, дошкою оголошень та запитами
TenantList	Список орендарів, з можливістю додавання та видалення
TenantCard	Картка орендаря з інформацією та кнопкою видалення
ResponsibilitiesList	Список обов'язків з можливістю додавання, редагування та видалення
ResponsibilitiesForm	Форма для додавання або редагування обов'язків
TransactionsList	Список транзакцій з можливістю додавання, редагування та видалення
TransactionCard	Карточка транзакції з інформацією та кнопками дій
TransactionForm	Форма для додавання або редагування транзакцій
RequestsList	Список запитів орендарів з можливістю додавання та редагування
RequestCard	Карточка запиту з інформацією та кнопками дій
RequestForm	Форма для додавання або редагування запитів
AnnouncementsList	Список оголошень з можливістю додавання та редагування
AnnouncementCard	Карточка оголошення з інформацією та кнопкою видалення
AnnouncementForm	Форма для додавання або редагування оголошень
PropertyList	Список нерухомості з можливістю додавання та видалення

Назва компоненту	Опис
PropertyCard	Картка нерухомості з інформацією та кнопкою видалення
PropertyForm	Форма для додавання нової властивості
Card	Універсальна картка для відображення інформації, базисний компонент для спеціалізованих карток
NotFound	Сторінка для відображення помилки 404

Програмний код розробленого програмного забезпечення як для клієнтської, так і для серверної частини можна знайти в репозиторії на GitHub за посиланням на QR-коді, зображеному на рисунку А.1 у додатку А.

5.4.4 Компоненти системи

Розроблена система складається з двох частин – frontend-частини та backend-частини, в складі кожної з яких виділяються основні компоненти, що формують їх призначення і фунціонал. Для складової frontend-частини виділено такі компоненти як React-застосунок, React-компоненти інтерфейсу, запити до API, сторінки на базі React-компонентів, React-роутер як модуль для керування рендером сторінок, локальне сховище браузеру та медіа-ресурси. Я компоненти backend-частини виділено маршрути FastAPI (FastAPI routes), обробники запитів, підсистему автентифікації, ORM-моделі та підсистему управління сесійним з’єднанням, що базується на класі Session. Базу даних також виділено як окремий компонент системи. Ці компоненти відображено у вигляді діаграми компонентів на кресленику ІС12.080БАК.005 Д3.

5.4.5 Діаграма послідовності

В межах розробленої системи передбачено багато варіантів використання ко-

ристувачами. В цьому розділі наведено діаграму послідовності, що відображає послідовність взаємодій між такими складовими системи як React-застосунок (фактично інтерфейс користувача), API, ORM-моделі та база даних, починаючи від дії зі сторони користувача і закінчуючи відгуком системи на цю дію на клієнтській стороні. Цю послідовність взаємодій відображено на діаграмі послідовності на кресленику ІС12.080БАК.005 Д4 на прикладі діяльності з редагування транзакції.

Висновки до розділу 5

В цьому розділі було описано систему з точки зору її технічної реалізації та обраних технологій розробки. Описано структуру і потоки даних в системі, її функціональну модель, створено модель бази даних. Деталізовано архітектуру розробленої системи з врахуванням технічних аспектів розробки, розроблено діаграму компонентів системи і діаграму послідовності, що демонструє послідовність взаємодій технічних складових системи під час діяльності з редагування користувачем транзакції.

6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Змістовна постановка задачі

У сучасних інформаційних системах, що оперують фінансовими потоками, важливу роль відіграє не лише зберігання та обробка даних, але й їх наочне подання для користувача. Одним із ключових елементів ефективної візуалізації фінансових даних є використання кольорового кодування, що дозволяє інтуїтивно оцінити структуру доходів і витрат, не вдаючись до детального аналізу числових значень.

Метою розробки є створення алгоритму, який автоматично генерує кольорове оформлення частин діаграми транзакцій на основі відносних значень окремих транзакцій у загальній структурі. При цьому очікується, що кольори будуть адаптовані до типу транзакції (дохід або витрата) і відображатимуть вклад кожної транзакції в загальну суму. Завдяки такому підходу користувач може швидко отримати уявлення про домінуючі категорії витрат або джерела доходу.

Поставлена задача має на меті підвищити ефективність візуального аналізу транзакцій через динамічне забарвлення елементів діаграми, яке враховує як абсолютні, так і відносні значення транзакцій. Це дає змогу забезпечити більш гнучке, інформативне й естетично привабливе відображення даних, що є важливим для прийняття рішень у фінансовому плануванні.

6.2 Математична постановка задачі

Нехай маємо множину транзакцій:

$$T = (t_1, t_2, \dots, t_n),$$

де кожна транзакція t_i характеризується числовим значенням v_i що представляє її суму. Кожній транзакції необхідно призначити колір у числовому представленні, що відображає вагомість транзакції на основі її суми по відношенню до сум інших транзакцій.

Відповідно, формується математична постановка задачі.

Вхідні дані:

n – кількість транзакцій,

$\vec{T} = (t_1, t_2, \dots, t_n)$ – вектор транзакцій,

$\vec{V} = (v_1, v_2, \dots, v_n)$ – вектор сум транзакції.

Обмеження:

$\forall v_i \in \vec{V}, v_i \in R^+$,

$\sum_{i=1}^{n} v_i > 0$,

$n \in N$.

Задачею є побудова відображення, наведеному у формулі (6.1).

$$f: T \to C, \tag{6.1}$$

де $C \subseteq [0,360] \times [0,100] \times [0,100]$ – простір допустимих значень кольору в моделі HSL (Hue, Saturation, Lightness), таке, що кожній транзакції t_i ставиться у відповідність колір $c_i = (h_i, s_i, l_i) \in C$, що має відображати вагомість транзакції по відношенню до інших.

6.3 Обґрунтування методу розв’язання

Завдання візуального подання фінансових транзакцій із використанням кольорового кодування активно вирішується в сучасних інформаційно-аналітичних системах, особливо у фінтех-застосунках, системах бухгалтерського обліку та персонального бюджетування. Ефективна візуалізація сприяє швидкому сприйняттю інформації, знижує когнітивне навантаження користувача та забезпечує інтуїтивно зрозумілий аналіз даних.

На сьогоднішній день найбільш поширені підходи до реалізації кольорового кодування даних включають:

– фіксоване кодування кольорів;

– кластеризація та градація кольору;

– масштабовані колірні градієнти.

Фіксоване кодування кольорів базується на ідеї, що кожній категорії транзакцій присвоюється сталий колір, незалежно від її кількісної характеристики. Цей підхід є простим у реалізації, проте не відображає відносної важливості або частки транзакції у загальній структурі.

При кластеризації та градації кольору застосовується сегментація транзакцій на групи за розміром або категорією, і кожній групі надається відповідний колір або градієнт. Це частково вирішує проблему відображення масштабу, але втрачається індивідуальна унікальність кожної транзакції.

Масштабовані колірні градієнти побудовані на основі математичного перетворення значень транзакцій у колірні параметри (наприклад, через нормалізацію, інтерполяцію, логарифмічне масштабування). Такий підхід дозволяє динамічно передавати розподіл значень і виділяти найбільш значущі транзакції.

Зважаючи на поставлену в пунктах 6.1 і 6.2 задачу, найбільш доцільним є використання саме масштабованого колірного градієнту з випадковим зсувом, який реалізується у вигляді функції, що трансформує відносну вагу транзакції у параметри HSL-моделі. Такий підхід має низку переваг:

– гнучкість – забезпечує адаптацію кольору до поточного набору даних незалежно від абсолютних значень;

– інформативність – дозволяє візуально виділяти транзакції з великою питомою вагою;

– випадкове варіювання – запобігає надмірній одноманітності кольорів для близьких значень, зберігаючи візуальну різноманітність;

– простота реалізації – алгоритм легко імплементується у сучасних мовах програмування (зокрема, JavaScript), що дозволяє інтегрувати його у фронтенд частину застосунку.

Враховуючи вищенаведене, для вирішення поставленої задачі обрано підхід, який поєднує лінійну інтерполяцію колірного відтінку, масштабовану насиченість

і яскравість, а також контрольований випадковий шум. Це дозволяє досягти бажаного ефекту візуального виділення важливих транзакцій без втрати загального стилю інтерфейсу користувача.

6.4 Опис методу розв’язання

Обраний метод базується на масштабованому колірному кодуванні із використанням HSL-моделі кольору, в якій колір кожної транзакції визначається відповідно до її питомої ваги в загальній сумі. Метод застосовується до масиву транзакцій і повертає набір кольорових значень, придатних для візуалізації (наприклад, у секторній або круговій діаграмі).

Основні принципи методу:

– Hue (відтінок): визначається залежно від типу транзакції (дохід або витрата) та масштабується пропорційно зменшенню питомої ваги;

– Saturation (насиченість): зростає з вагою транзакції, забезпечуючи виразніший вигляд для більших значень;

– Lightness (яскравість): зменшується з вагою транзакції, візуально акцентуючи великі значення.

Випадкове варіювання додає дрібні коливання до насиченості та яскравості, щоб уникнути повторення близьких кольорів для наборів даних, що мають близькі значення. Тобто, коли транзакції мають схожу або однакову питому вагу в загальній сумі, призначені їм кольори на діаграмі можуть бути надто близькими за спектром, через що деякі фрагменти діаграми стає важче відрізнити. Випадкове варіювання допомагає уникнути цієї проблеми.

Загальну логіку і принцип роботи обраного методу, що реалізовано в програмному коді на клієнтській частині, описано у вигляді псевдокоду на рисунку 6.1.

```
Функція generateColorsBasedOnValues(data, baseColor):
    shadeFactor ← якщо baseColor = 'green' тоді 120, інакше 0
    gradientColor ← якщо baseColor = 'green' тоді 240, інакше 60
    totalValue ← сума всіх item.value у data

Для кожного item у data:
    valueRatio ← item.value / totalValue
    saturation ← 30 + valueRatio * 70 + випадкове число від 0 до 15
    lightness ← 70 - valueRatio * 55 - випадкове число від 0 до 5
    mixedHue ← shadeFactor + (gradientColor - shadeFactor) * (1 -valueRatio)
    color ← HSL(mixedHue, saturation%, lightness%)
    додати color до результату

Повернути список кольорів
```

Рисунок 6.1 – Псевдокод методу

Відповідно, приклад застосування цього методу наведено нижче.

Нехай задано:

$n = 3$;

$\vec{T} = (t_1, t_2, t_3) =$ (витрата 1, витрата 2, витрата 3);

$\vec{V} = (v_1, v_2, v_3) = (300, 100, 600)$.

Розв’язок.

Крок 1. Обчислення загальної суми.

Загальна сума числових значень транзакцій обчислюється за формулою (6.2).

$$V_{\text{total}} = \sum_{i=1}^{n} v_i \,, \tag{6.2}$$

де V_{total} – загальна сума числових значень транзакцій, v_{i} – числове значення витрати t_i.

Відповідно, за формулою (6.2) обраховується загальна сума для поточних значень:

$$V_{\text{total}} = 300 + 100 + 600 = 1000.$$

Крок 2. Обчислення відносних ваг (часток).

Відносні ваги транзакцій обчислюються за формулою (6.3).

$$r_i = \frac{v_i}{V_{\text{total}}}, \qquad i = \overline{1, n}, \tag{6.3}$$

де r_i – відносна вага транзакції t_i, v_{i} – числове значення витрати t_i, V_{total} – загальна сума числових значень транзакцій.

Відповідно до формули (6.3) проводяться обчислення:

$$r_1 = \frac{300}{1000} = 0{,}3,$$
$$r_2 = \frac{100}{1000} = 0{,}1,$$
$$r_3 = \frac{600}{1000} = 0{,}6.$$

Крок 3. Встановлення параметрів відтінку (Hue).

Оскільки транзакції за умовою є витратами, то встановлюються відповідні значення базового і градієнтного кольору:

$$H_{\text{base}} = 0 \text{ (червоний)},$$
$$H_{\text{gradient}} = 60 \text{ (жовтий)}.$$

Відтінок для кожної транзакції обчислюється за формулою (6.4).

$$h_i = H_{\text{base}} + \left(H_{\text{gradient}} - H_{\text{base}}\right) \cdot (1 - r_i) = 0 + 60 \cdot (1 - r_i), \;\; i = \overline{1, n}, \tag{6.4}$$

де h_i – числове значення нового кольору для кожної транзакції t_i, H_{base} та H_{gradient} – це чисельні значення базового кольору та градієнтного кольору.

Тоді, відповідно до формули (6.4) обраховуються фактичні значення:

$$h_1 = 60 \cdot (1 - 0{,}3) = 60 \cdot 0{,}7 = 42,$$
$$h_2 = 60 \cdot (1 - 0{,}1) = 60 \cdot 0{,}9 = 54,$$
$$h_3 = 60 \cdot (1 - 0{,}6) = 60 \cdot 0{,}4 = 24.$$

Крок 4. Обчислення насиченості (Saturation).

Обчислення числових значень насиченості відбувається за формулою (6.5):

$$s_i = 30 + 70 \cdot r_i + \varepsilon_{1,i}, \;\; \varepsilon_{1,i} \in [0{,}15], \;\; i = \overline{1,n}, \tag{6.5}$$

де $\varepsilon_{1,i}$ – це випадкове число в заданому діапазоні, згенероване з деякою вірогідністю для транзакції t_i для додаткового варіювання насиченості.

Припускається, що випадкові числа приймають такі значення:

$$\varepsilon_{1,1} = 5,$$
$$\varepsilon_{1,2} = 2,$$
$$\varepsilon_{1,3} = 10.$$

Тоді обрахунки за формулою (6.5) виглядають наступним чином:

$$s_1 = 30 + 70 \cdot 0{,}3 + 5 = 30 + 21 + 5 = 56,$$
$$s_2 = 30 + 70 \cdot 0{,}1 + 2 = 30 + 7 + 2 = 39,$$
$$s_3 = 30 + 70 \cdot 0{,}6 + 10 = 30 + 42 + 10 = 82.$$

Крок 5. Обчислення яскравості (Lightness).

Обчислення числового значення яскравості відбувається за формулою (6.6).

$$l_i = 70 - 55 \cdot r_i - \varepsilon_{2,i}, \;\; \varepsilon_{2,i} \in [0{,}5], \;\; i = \overline{1,n}, \tag{6.6}$$

де $\varepsilon_{2,i}$ – це випадкове число в заданому діапазоні, згенероване з деякою вірогідністю для транзакції t_i для додаткового варіювання яскравості.

Припускається, що випадкові числа приймають такі значення:

$$\varepsilon_{2,1} = 2,$$
$$\varepsilon_{2,2} = 1,$$
$$\varepsilon_{2,3} = 3.$$

Тоді обрахунки за формулою (6.6) виглядають наступним чином:

$$l_1 = 70 - 55 \cdot 0{,}3 - 2 = 70 - 16{,}5 - 2 = 51{,}5,$$
$$l_2 = 70 - 55 \cdot 0{,}1 - 1 = 70 - 5{,}5 - 1 = 63{,}5,$$
$$l_3 = 70 - 55 \cdot 0{,}6 - 3 = 70 - 33 - 3 = 34.$$

Крок 6. Побудова відображення T → C.

Остаточні кольори в HSL-моделі будуються комбінуванням отриманих на кроках 2 – 5 значень за формулою (6.7).

$$f(t_i) = c_i = (h_i, s_i, l_i), \;\; i = \overline{1,n}, \tag{6.7}$$

де f – функція відображення, c_i – числове значення кінцевого кольору за моделлю HSL.

Відповідно, утворення фактичних значень кольору за формулою (6.7) наведено нижче:

$$f(t_1) = (h_1, s_1, l_1) = (42^\circ, 56\%, 51{,}5\%),$$
$$f(t_2) = (h_2, s_2, l_2) = (54^\circ, 39\%, 63{,}5\%),$$
$$f(t_3) = (h_3, s_3, l_3) = (24^\circ, 82\%, 34\%).$$

Таким чином було отримано відображення, що відповідає формулі (6.1):

$$f: T = \{t_1, t_2, t_3\} \rightarrow C = \{c_1, c_2, c_3\},$$

де кольори приймають такі значення у моделі HSL:

$$c_1 = \text{HSL}(42°, 56\%, 51{,}5\%),$$
$$c_2 = \text{HSL}(54°, 39\%, 63{,}5\%),$$
$$c_3 = \text{HSL}(24°, 82\%, 34\%).$$

Висновки до розділу 6

Було сформовано змістовну і математичну постановку задачі для проблеми з розподілу кольорів для сегментів кругових діаграми витрат і доходів розділу фінансової статистики розробленого програмного забезпечення. Було проаналізовано можливі варіанту розв’язку поставленої задачі, обрано тип алгоритму для реалізації. Відповідно до обраного методу розроблено алгоритм зі створення кольорів на основі колірного градієнту в залежності від ваги транзакцій однакового типу в загальній сумі. Розроблений алгоритм описано із наданням псевдокоду і прикладом використання.

7 ТЕСТУВАННЯ ПЗ

7.1 Мета випробувань

Метою випробувань є перевірка функціоналу розробленого програмного забезпечення, що реалізує систему управління орендною житловою нерухомістю, на відповідність сформованим вимогам, специфіці предметної області, а також на відсутність помилок в роботі програмного забезпечення під час взаємодії з системою користувача за запланованим сценаріями. Випробування представляють інтерес з точки зору оцінки якості розробленої системи і поточного стану її готовності, відповідно, надають основу для формування плану покращення і розширення функціоналу в майбутньому.

7.2 Спосіб проведення випробувань

Випробування проводяться шляхом тестування функціоналу з точки зору користувача, що взаємодіє з нею з кінцевого пристрою. Фактично, шляхом такого тестування перевіряється коректність роботи системи власне з точки зору її прямого призначення і дозволяє перевірити всі аспекти реалізації. Тестування функціоналу проводилося на розгорнутій локально системі з використанням веббраузеру Chrome. Для повноцінного тестування системи базу даних було заповнено тестовими даними такого характеру, що дають змогу оцінити роботу системи для всіх можливих варіантів взаємодій.

7.3 Результати випробувань

Результати випробувань подаються у вигляді таблиць, де відображені мета тестування, початковий стан системи (стан системи до початку виконання дій, що передбачені сценарієм тестування), вхідні дані, схема проведення тестування, очікуваний результат та стан системи після випробування. Тест вважається успішно пройденим, якщо кінцевий стан системи відповідає очікуваному результату.

Оскільки в системі передбачено схожі за принципом роботи і реалізацією взаємодії з різними об’єктами, що включають в себе обов’язки, транзакції, запити і оголошення в межах окремої нерухомості, результати тестів було вирішено згрупувати для всіх типів інформаційних об’єктів, розділивши їх лише за типом взаємодії.

В таблиці 7.1 відображено результати тестування функціоналу перегляду обов’язків, транзакцій, запитів і оголошень зареєстрованим користувачем.

Таблиця 7.1 – Перегляд інформаційних записів нерухомості

Параметр	Опис
Мета тестування	Перевірка коректності відображення релевантних обов’язків, транзакцій, запитів і оголошень
Початковий стан системи	Користувач увійшов в систему, обрано певну нерухомість і відкрито панель управління цією нерухомістю
Вхідні дані	Токен авторизації
Схема проведення тестування	На панелі управління обраної нерухомості обирається одна з вкладок для відображення списку інформаційних об’єктів
Очікуваний результат	На панелі управління виводиться список обраних об’єктів, що відповідають акаунту користувача з певною роллю і обраній нерухомості
Стан системи після випробування	Відповідає очікуваному результату

В таблицях 7.2 – 7.4 наведені результати тестування функціоналу з редагування, додавання і видалення обов’язків, транзакцій, запитів і оголошень авторизованого користувача. Авторизований користувач – це користувач, що увійшов до системи і має права на зазначену дію, що залежить від його ролі.

Таблиця 7.2 – Додавання нового запису до об’єкту нерухомості

Параметр	Опис
Мета тестування	Перевірка функціоналу з додавання нових обов’язків, транзакцій, запитів і оголошень авторизованим
Початковий стан системи	Користувач увійшов в систему, обрано певну нерухомість і відкрито панель управління цією нерухомістю
Вхідні дані	Токен авторизації, поля форми для додавання запису
Схема проведення тестування	На панелі управління обраної нерухомості обирається одна з вкладок для відображення списку інформаційних об’єктів, відкрито форму для додавання нових записів, заповнено всі необхідні поля, натиснуто кнопку для відправки форми
Очікуваний результат	Після підтвердження заповнену форму закрито, отримується відповідь від сервера, відповідний список оновлюється з урахуванням доданого об’єкту
Стан системи після випробування	Відповідає очікуваному результату

Таблиця 7.3 – Редагування записів об’єкту нерухомості

Параметр	Опис
Мета тестування	Перевірка функціоналу з редагування обов’язків, транзакцій, запитів і оголошень авторизованим користувачем
Початковий стан системи	Користувач увійшов в систему, обрано певну нерухомість і відкрито панель управління цією нерухомістю
Вхідні дані	Токен авторизації, поля форми для редагування запису

Параметр	Опис
Схема проведення тестування	На панелі управління обраної нерухомості обирається одна з вкладок для відображення списку інформаційних об’єктів, обрано певний запис, відкрито форму для редагування цього запису, змінено вміст необхідних полів, натиснуто кнопку для відправки форми
Очікуваний результат	Після підтвердження заповнену форму закрито, отримується відповідь від сервера, відповідний список оновлюється з урахуванням зміненого об’єкту
Стан системи після випробування	Відповідає очікуваному результату

Таблиця 7.4 – Видалення записів об’єкту нерухомості

Параметр	Опис
Мета тестування	Перевірка функціоналу з видалення обов’язків, транзакцій, запитів і оголошень авторизованим користувачем
Початковий стан системи	Користувач увійшов в систему, обрано певну нерухомість і відкрито панель управління цією нерухомістю
Вхідні дані	Токен авторизації
Схема проведення тестування	На панелі управління обраної нерухомості обирається одна з вкладок для відображення списку інформаційних об’єктів, обрано певний запис, натиснуто кнопку для видалення запису
Очікуваний результат	Система надсилає повідомлення про підтвердження дії, якщо дію підтверджено, то отримується відповідь від сервера, відповідний список оновлюється з урахуванням видаленого об’єкту

Параметр	Опис
Стан системи після випробування	Відповідає очікуваному результату

У таблиці 7.5 наведено результати випробування з перевірки того, як відбувається доступ до ресурсів системи не автентифікованого користувача, тобто користувача, що не увійшов в систему.

Таблиця 7.5 – Доступ до системи не автентифікованим користувачем

Параметр	Опис
Мета тестування	Перевірка відгуку системи і контрольованого доступу до її ресурсів для випадку, коли користувач не увійшов в систему
Початковий стан системи	Користувач не увійшов в систему, відкрито розділ «Profile»
Вхідні дані	Токен авторизації з вичерпаним терміном дії / пустий токен авторизації / невалідний токен авторизації
Схема проведення тестування	Спочатку відкривається розділ з профілем користувача, потім відкривається розділ «Home», де повинен відображатися список нерухомості. Після цього додатково перевіряється доступ до ресурсів з серверу за наявними URL як складових API
Очікуваний результат	На сторінці «Profile» відображається повідомлення про необхідність входу у систему, на сторінці «Home» не відображається жодних об’єктів нерухомості та наявне повідомлення про те, що користувача не автентифіковано. При спробі доступу будь-яких ресурсів системи за URL-адресами API сервер надси-

Параметр	Опис
	лає відповідь з кодом, що свідчить про невдалу автентифікацію користувача за токеном, жодної зайвої інформації не відправляється
Стан системи після випробування	Відповідає очікуваному результату

У таблиці 7.6 відображено результати перевірки коректності функціоналу розділу фінансової статистики нерухомості для орендодавців.

Таблиця 7.6 – Перегляд розділу фінансової статистики

Параметр	Опис
Мета тестування	Перевірка коректності відображення даних у розділі фінансової статистики
Початковий стан системи	Користувач увійшов в систему, відкрито розділ «Statistics», роль користувача – орендодавець
Вхідні дані	Токен авторизації
Схема проведення тестування	У розділі статистики обираються конкретні місяці для відображення статистики на основі доданих транзакцій
Очікуваний результат	На сторінці відображаються сумарний дохід, сумарні витрати та баланс для обраного місяцю. Знизу відображено дві кольорові кругові діаграми, що відображають відношення вкладу окремих типів транзакцій до сумарного місячного доходу і витрат. Сегменти кругових діаграм, що відображають типи фінансових операцій офарбовані відповідно до їх вкладу в загальну суму за розробленим в розділі 6 алгоритмом. Транзакції з роллю сплатника, що відповідає орендо-

Параметр	Опис
	давцю, відображаються у діаграмі витрат. До статистики включаються лише ті транзакції, що були підтверджені всіма зазначеними користувачами
Стан системи після випробування	Відповідає очікуваному результату

У таблиці 7.7 відображено результати з перевірки коректного відображення адаптивного інтерфейсу системи для пристроїв з різними розмірами екранів.

Таблиця 7.7 – Відображення інтерфейсу на екранах різного розміру

Параметр	Опис
Мета тестування	Перевірка коректності відображення інтерфейсу на різних типів пристроїв
Початковий стан системи	Проведено вхід у систему
Вхідні дані	Токен авторизації, розмір екрану
Схема проведення тестування	Виконується перехід по всім розділам системи: домашня сторінка, панель управління нерухомістю, розділ фінансової статистики, розділ профілю користувача. Для кожного розділу перевіряється адаптивність відображених в ньому компонентів за допомогою інструментів розробника браузеру Chrome для форматів екранів, що відповідають найпопулярнішим користувацьким пристроям
Очікуваний результат	Відображені компоненти інтерфейсу змінюють свій розмір і положення відповідного до обраного розміру екрану. Жодної необхідної інформації не зникає, елементи управління мають комфортний для взаємодії розмір, візуальні дефекти відсутні

Параметр	Опис
Стан системи після випробування	Відповідає очікуваному результату

Висновки до розділу 7

Створене програмне впровадження інформаційної системи управління орендною житловою нерухомістю було протестовано на відповідність поставленим вимогам і на відсутність помилок для сценаріїв, що максимально повно демонструють суть і особливості взаємодії з ним користувача. Було розглянуто такі сценарії як додавання, редагування і видалення обов’язків, транзакцій, запитів і оголошень, їх відображення, а також перевірено коректність відображення користувацького інтерфейсу на різних пристроях і обробку системою ситуацій доступу до неї користувачами, що не провели вхід у систему. Також було протестовано коректність відображення даних у розділі фінансової статистики. Результати випробувань було згруповано і відображено у вигляді таблиць. Згідно створеного плану і отриманих результатів всі випробування було пройдено успішно. Система відповідає поставленим вимогам.

ВИСНОВКИ

В ході роботи над створенням інформаційної системи управління орендною житловою нерухомістю було описано предметну область, в межах якої функціонує розроблена система, визначено особливості діяльності в сфері нерухомості, визначено призначення розробки, її цілі та задачі.

Було проаналізовано існуючі програмні рішення, що реалізують подібні системи, визначено їх переваги і недоліки, обґрунтовано доцільність власної розробки в межах дипломного проєкту.

До розробленої системи було сформовано вимоги до структури, надійності, до способів збереження інформації, до функціональних характеристик, а також вимоги до видів забезпечення.

На основі поставлених цілей і задач розробки та вимог до неї було обрано набір технологій для реалізації системи управління орендною нерухомістю як програмного продукту, що представляє з себе багатофункціональний вебзастосунок. Як основу для backend-частини застосунку було обрано фреймворк FastAPI, а для frontend-частини – React.js. Як базу даних було обрано PostgreSQL.

Було описано технічну складову реалізації системи як програмного продукту: описано структуру системи, створено її функціональну модель, створено модель бази даних, описано вхідні та вихідні дані в системі в процесі виконання різних функцій.

Було описано математичне забезпечення системи – поставлено специфічну для розроблюваної системи задачу з умовного форматування частин діаграм розділу фінансової статистики, було сформовану її математичну постановку, проаналізовано існуючі варіанти розв’язку цієї задачі, обрано тип алгоритму для її вирішення, а вибір обґрунтовано. На основі обраного типу було розроблено алгоритм для вирішення поставленої задачі на основі утворення масштабованого колірного градієнту з випадковим зсувом, цей алгоритм було описано. Також було наведено приклад розв’язання задачі з реальними числовими даними цим алгоритмом.

Створену систему було протестовано на відповідність поставленим вимогам, результати тестувань згруповано, класифіковано за типом взаємодії і наведено у вигляді таблиць.

Результатом дипломного проєкту є розроблена система управління орендною нерухомістю на базі вебсервісу. Отримані в ході роботи результати можна використовувати в дослідженнях в сфері розробки веб-застосувань на базі клієнт-серверної архітектури, а також в якості базису для створення систем різного призначення, що використовують серверні фреймворки мови програмування Python. Розроблена система може використовуватися як рішення в галузі оренди житлової нерухомості, а її подальший розвиток і модифікації надають основу для можливого використання в галузях, пов’язаних з орендою об’єктів різного формату і призначення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gilderbloom J. Rethinking rental housing / ed. by A. R. P. Philadelphia : Temple University Press, 1988. 280 p.

2. Kemp P. A. Private Renting After the Global Financial Crisis. Housing Studies. 2015. Vol. 30, no. 4. P. 601–620. URL: https://doi.org/10.1080/02673037.2015.1027671 (date of access: 05.03.2024).

3. Providing Rental Housing: A Systematic Literature Review of Residential Rental Property Owner Decision Making / T. M. Cook et al. Journal of Planning Literature. 2024. URL: https://doi.org/10.1177/08854122241239571 (date of access: 10.03.2025).

4. Pioneers Information Technology Ltd. The Role of Automation in Property Management Success. Pioneers IT. URL: https://pioneersit.com/the-role-of-automation-in-property-management-success/ (date of access: 20.03.2024).

5. Booking Holdings Inc. How we work. booking.com. URL: https://www.booking.com/content/how_we_work.html#hww_may_25_accommodations_item_9.

6. Airbnb, Inc. About AirBnb: what it is and how it works - AirBnb help center. Airbnb. URL: https://www.airbnb.com/help/article/2503.

7. Hostaway Oy. Features - Hostaway | short-term, airbnb & vacation rental management software. Hostaway - Airbnb & Vacation Rental Software. URL: https://www.hostaway.com/features/ (date of access: 01.04.2024).

8. Lodgify. Logify property management software. lodgify.com. URL: https://www.lodgify.com/property-management-software/.

9. TenantCloud LLC. TenantCloud - everything you need to manage your properties. tenantcloud.com. URL: https://www.tenantcloud.com/ (date of access: 09.04.2024).

10. De La Guardia C. Python Web Frameworks / ed. by A. MacDonald. Sebastopol, CA : O'Reilly Media, Inc., 2016. 72 p.

11. Python developers survey 2023 results. JetBrains: Developer Tools for Professionals and Teams. URL: https://lp.jetbrains.com/python-developers-survey-2023/ (date of access: 01.04.2025).

12. Lubanovic B. FastAPI: modern python web development / ed. by S. Wilkey. Sebastopol, CA : O'Reilly Media, Incorporated, 2023. 277 p.

13. Singh A. Mastering PostgreSQL with Python. Jabalpur, MP, 2023. 320 p.

14. Schwarzmüller M. React key concepts: consolidate your knowledge of React's core features. Packt Publishing, Limited, 2022. 590 p.

15. Бунке О.С. Клієнт-серверна архітектура. Серверні web-технології : навч. посіб. / ред. Новіков П.В. Київ, 2023.