Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут
комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: «Аналітичний застосунок для моделювання та аналізу попиту на оренду апартаментів м. Львова»

Виконав: студент 6 курсу групи КН-61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Пецкович В. І.
(прізвище та ініціали)

Керівник Крошний І. М.
(прізвище та ініціали)

Рецензент Сторонсук О. І.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістреський)

Спеціальність 122 "Комп'ютерні науки"

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувачка кафедри КН

_____ Борецька І. Б.

"10" грудня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пецковичу Владиславу Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Аналітичний застосунок для моделювання та аналізу попиту на оренду апартаментів м. Львова

керівник роботи *Крошний Ігор Миколайович, к.т.н., доцент*,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "29" квітня 2025 року № С-288

2. Термін подання студентом роботи 10 грудня 2025 р.

3. Вихідні дані до роботи Розробити аналітичний застосунок для моделювання та аналізу попиту на оренду апартаментів м. Львова. Реалізувати програмний застосунок у вигляді сайту, бота та адмін-панелі керування, що забезпечить контроль та аналіз бронювань.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

1. Стан проблемної області

2. Інформаційне забезпечення

3. Математичне забезпечення

4. Програмне забезпечення

5. Розробка стартап-проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалів до доповіді

6. Дата видачі завдання 1 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	02.05.25 – 17.05.25	Виконано
2	Розроблення сайту лендінгу з модулем бронювання	18.05.25 – 02.06.25	Виконано
3	Створення чат боту для покращення досвіду використання у користувачів	03.06.25 – 19.06.25	Виконано
4	Розробка серверної частини системи	20.06.25 – 13.07.25	Виконано
5	Створення адмін-панелі керування	14.07.25 – 21.08.25	Виконано
6	Розробка аналітичної частини і машинного навчання на справжніх даних	22.08.25 – 18.09.25	Виконано
7	Тестування системи та виправлення помилок	19.09.25 – 29.10.25	Виконано
8	Оформлення пояснювальної записки	30.10.25 – 25.11.25	Виконано

Студент _____ (підпис) Пецкович В. І. (прізвище та ініціали)

Керівник роботи _____ (підпис) Крошний І. М. (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 76 сторінок пояснювальної записки, 26 рисунків, 2 додатки, 2 формули, 20 джерел.

У роботі представлено розробку застосунку для бронювання апартаментів у місті Львові та повноцінної системи аналізу цін на короткострокову оренду нерухомості для компанії Royal Apart. Метою проєкту було створення зручного інтерфейсу для користувачів (вебсайту та чат-бота), а також адміністративної панелі для персоналу з інструментами аналітики, що сприяють оптимізації цінової політики та підвищенню прибутковості компанії.

У роботі сформовано власний датасет, який включає 54 апартаментів, розроблено сайт з власним дизайном, створено інноваційного чат-бота, орієнтованого на локальний ринок, та багатофункціональну адмін-панель керування.

Ключові слова: нерухомість, вебсайт, машинне навчання, аналіз цін, бронювання, чат-бот, короткострокова оренда.

ABSTRACT

The thesis contains 76 pages of the explanatory report, 26 figures, 2 appendixes, 2 formulas, 20 references.

This thesis presents the development of a website for booking apartments in the city of Lviv, as well as a comprehensive system for analyzing prices and demand in the short-term rental market for the company Royal Apart. The aim of the project was to create a user-friendly interface (website and Telegram bot) and an administrative panel for company staff equipped with analytical tools that support price optimization and improve the company’s profitability.

A custom dataset containing 54 apartments has been created, a website with an original design has been developed, an innovative chatbot tailored to the local market has been implemented, and a multifunctional administration panel has been built.

Keywords: real estate, website, machine learning, price analysis, booking, chatbot, short-term rental.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити комплексну систему для бронювання апартаментів та аналітики попиту на короткострокову оренду нерухомості для компанії Royal Apart, що включає вебсайт, чат-бот та адміністративну панель, а також модулі обробки даних і машинного навчання, а саме:

1. Розробити структуру та сформувати власний датасет, що включає усі апартаменти із параметрами розташування, характеристиками об’єктів та ціновими даними;
2. Необхідно створити вебсайт з індивідуальним дизайном, який забезпечує перегляд каталогу апартаментів, фільтрацію та пошук, перегляд доступності, оформлення онлайн-бронювання.
3. Потрібно розробити чат-бота, орієнтованого на локальний ринок, який підтримує пошук і бронювання апартаментів, перегляд доступності, отримання консультацій, клієнтську підтримку та взаємодію з API серверної частини.
4. Необхідно створити багатофункціональну адміністративну панель, що забезпечує управління апартаментами та їх характеристиками, роботу з користувачами, контроль даних чат-бота й вебсайту та доступ до аналітичних даних.
5. Потрібно реалізувати модуль машинного навчання для аналізу попиту та цін, який включає попередню обробку даних, формування ознак, навчання моделі прогнозування попиту або оптимальної ціни та оцінку її якості за регресійними метриками MAE, MSE, RMSE і MAPE.
6. Необхідно створити серверний модуль (API) для обробки запитів вебсайту та чат-бота, взаємодії з базою даних і датасетом, виконання прогнозів моделі та забезпечення її перенавчання за потреби.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API — Application Programming Interface (програмний інтерфейс взаємодії);

БД — база даних;

UI — User Interface (користувацький інтерфейс);

UX — User Experience (досвід користувача);

ML — Machine Learning (машинне навчання);

PMS – Property Management System (Система управління майном)

DBMS — Database Management System (система керування базами даних);

MongoDB — документно-орієнтована NoSQL база даних;

Node.js — серверне середовище виконання JavaScript;

Express.js — фреймворк для створення серверних застосунків Node.js;

React — бібліотека JavaScript для побудови інтерфейсів;

Vite — інструмент для швидкої розробки фронтенда;

Tailwind CSS — утилітарний CSS-фреймворк;

REST — Representational State Transfer (архітектурний стиль API);

JSON — JavaScript Object Notation (формат обміну даними);

MAE — Mean Absolute Error (середня абсолютна помилка);

MSE — Mean Squared Error (середня квадратична помилка);

RMSE — Root Mean Squared Error (корінь середньої квадратичної помилки);

MAPE — Mean Absolute Percentage Error ;

ВСТУП

У сучасних умовах цифровізації ринок короткострокової оренди житла активно трансформується, переходячи від традиційних підходів до повністю автоматизованих систем бронювання. Зростання конкуренції, потреба у швидкому реагуванні на поведінку користувачів та вимоги до точного управління цінами стимулюють появу комплексних технологічних рішень, що поєднують сучасні вебплатформи, мобільні інтерфейси, інтеграцію з професійними PMS-системами та елементи машинного навчання. Саме тому важливими стають системи, здатні забезпечити одночасно зручність користувачам, швидку взаємодію з сервісами бронювання, інтерактивну аналітику попиту і підтримку бізнес-процесів у сфері оренди нерухомості.

Проте інтеграція таких алгоритмів у повноцінну систему бронювання вимагає створення архітектури, здатної об’єднати вебсайт, мобільні сервіси, адміністративний функціонал та стабільну роботу з PMS, що робить подібні рішення технічно складними та унікальними.

Актуальність проблеми визначається низкою важливих чинників: зростанням конкуренції на ринку подобової оренди житла, необхідністю забезпечення швидкої та зручної взаємодії для гостей та менеджерів, потребою у централізованому управлінні апартаментами в межах єдиного цифрового середовища, збільшенням вимог до автоматизації календарів, тарифів та обробки заявок, розвитком PMS-платформ, таких як Wubook, що відкривають доступ до API для сторонніх інтеграцій, а також появою інструментів машинного навчання, здатних прогнозувати попит і оптимізувати цінову політику. Недостатність систем, які одночасно забезпечують бронювання, гнучке керування даними та можливості аналітики, підвищує цінність комплексних рішень на ринку.

Об’єктом дослідження є процес автоматизації системи бронювання апартаментів та управління нерухомістю в цифровому середовищі.

Предметом дослідження є методи та засоби створення вебсистеми бронювання з інтеграцією PMS Wubook та використанням моделей машинного навчання для прогнозування попиту й оптимізації ціни.

Метою роботи є розроблення комплексної системи для бронювання апартаментів та аналітики попиту, яка включає вебсайт, чат-бот, адміністративну панель, серверний модуль і підсистему машинного навчання, інтегровану з Wubook PMS та реалізовану на основі сучасних вебтехнологій.

Для реалізації необхідно виконати такі завдання: проаналізувати ринок короткострокової оренди та можливості інтеграції з Wubook PMS; сформувати власний датасет апартаментів і підготувати дані до використання в системі; розробити вебсайт для перегляду, фільтрації, доступності та бронювання апартаментів; створити чат-бота; реалізувати адміністративну панель для керування апартаментами; забезпечити коректну інтеграцію з Wubook API для отримання та синхронізації даних; розробити та навчити модель машинного навчання для прогнозування оптимальної ціни та оцінки попиту; створити серверну частину, що забезпечує взаємодію між усіма модулями системи; здійснити комплексне тестування та оцінити ефективність роботи системи.

Наукова новизна роботи полягає в розробленні інтегрованого рішення, яке поєднує класичні інструменти веброзробки з машинним навчанням та можливістю динамічного ціноутворення. Система не лише автоматизує бронювання, а й розширює можливості PMS Wubook, доповнюючи її кастомним інтерфейсом, гнучкою системою керування та модулем прогнозування попиту, що забезпечує прийняття обґрунтованих бізнес-рішень.

Практичне значення отриманих результатів полягає у створенні масштабованої платформи, яка може бути використана малими компаніями, мережами апартаментів або приватними власниками. Вона забезпечує автоматизацію бронювань, скорочення навантаження на персонал, підвищення точності управління цінами, а також пропонує інструменти аналітики, що дозволяють швидко адаптуватися до змін ринку.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Динамічність ринку та проблеми ціноутворення

У сучасному ринку короткострокової оренди нерухомості інформація про ціни, доступність і попит змінюється надзвичайно швидко. Туристичні міста, такі як Львів, характеризуються високою сезонністю, залежністю від подій, свят, ділової активності та зовнішніх факторів. У таких умовах вартість апартаментів може коливатися не лише щотижня, а й щодня. Ціни формуються динамічно, змінюються під впливом конкурентів, рівня заповненості, сезонних трендів, очікуваних подій та багатьох інших параметрів. Це створює складність для компаній, які працюють на локальному ринку оренди та намагаються підтримувати стабільну прибутковість у мінливому середовищі.

З огляду на швидкість змін традиційні ручні методи встановлення цін стають неефективними. Менеджери, які щоденно працюють з бронюваннями, календарями та заявками, не завжди можуть оперативно та раціонально оцінити оптимальну ціну для кожного конкретного дня чи апартаменту. Вони часто покладаються на особистий досвід або загальні ринкові припущення, які не враховують усю складність факторів: попередні бронювання, рівень зайнятості конкурентів, прогнозовану завантаженість чи локальні події. Це призводить до цінових втрат — як у вигляді недоотримання доходу, так і у вигляді низької заповненості у періоди високого попиту.

Додатковою проблемою є підвищена конкуренція на локальному ринку. У Львові працює значна кількість компаній та окремих власників, які пропонують апартаменти різних класів і вартості категорій. Незважаючи на це, навіть найбільші гравці ринку часто не впроваджують інноваційні технологічні рішення та працюють за застарілими моделями управління. Їхні переваги ґрунтуються здебільшого на масштабах, а не на сучасних підходах до автоматизації чи аналітики. Це створює додатковий простір для впровадження інтелектуальних систем, які дозволяють молодшим або середнім компаніям вирівнювати конкуренцію за рахунок технологічних інструментів.

Важливу роль на ринку відіграють PMS (Property Management System) — системи управління апартаментами, календарями та бронюваннями. Однією з таких платформ є Wubook [14], яка широко використовується компаніями для синхронізації каналів продажу, обліку бронювань та управління доступністю. Проте, попри високий рівень автоматизації операційних процесів, базовий функціонал PMS не завжди забезпечує розширену аналітику чи гнучкі інструменти оцінювання попиту. Системи такого типу рідко враховують локальні особливості ринку або автоматично аналізують історію бронювань із метою прогнозування оптимальної ціни. Як наслідок, навіть із використанням Wubook значна частина рішень щодо ціноутворення покладається на менеджера, що не гарантує точності чи ефективності.

Проблема ускладнюється тим, що ринок короткострокової оренди надзвичайно динамічний: ціни конкурентів змінюються непередбачувано, особливі піки попиту виникають раптово, а поведінка туристів та гостей міста стає все менш стабільною. Змінюються пріоритети клієнтів, підходи до вибору житла, спосіб взаємодії з сервісами бронювання. У цих умовах компаніям потрібно не лише підтримувати високий рівень сервісу, а й мати інструменти, що дозволяють оперативно реагувати на зміни та приймати рішення на основі даних.

Надмірна залежність від ручних процесів, недостатня аналітична підтримка, відсутність автоматизованого прогнозування попиту та обмеженість традиційних PMS-рішень створюють реальну загрозу втрати конкурентоспроможності. Підприємства, які не впроваджують інноваційні технології, стикаються з ризиком неефективного використання ресурсів, нерівномірного завантаження апартаментів та зниження прибутку. Навіть компанії з великим портфоліо нерухомості можуть зазнавати втрат, якщо не використовують інтелектуальні інструменти для обробки даних та аналізу ринку.

У рамках цього дослідження проведено аналіз сучасного стану проблемної області управління цінами та попитом у сфері короткострокової оренди. Вивчено роль PMS-систем схожих на Wubook, визначено їхні сильні сторони та обмеження, а також проаналізовано проблеми традиційних методів ціноутворення. Отримані результати

дозволили сформувати вимоги до системи, що розробляється: вона повинна забезпечувати автоматизований аналіз попиту, оперативність, точність, можливість оновлення даних, синхронізацію з PMS та підтримку зручної взаємодії для клієнтів і персоналу.

Таким чином, стан проблемної області підтверджує актуальність створення інтелектуальної системи, яка поєднує можливості PMS, сучасні вебтехнології та алгоритми прогнозування. Подальша розробка такої системи дозволяє зменшити навантаження на менеджерів, підвищити точність ціноутворення, забезпечити конкурентні переваги та покращити ефективність роботи у сфері короткострокової оренди.

1.2 Організаційні труднощі та людський фактор у роботі менеджерів

У сфері короткострокової оренди нерухомості значна частина ефективності бізнесу залежить від роботи менеджерів, які щодня повинні оновлювати ціни, контролювати заповненість апартаментів і слідкувати за актуальністю інформації на всіх каналах бронювання. У компанії Royal Apart це особливо складно, адже менеджер працює з 54 апартаментами, кожен із яких має власну сезонність, рівень попиту та окремий календар доступності.

Ручне встановлення цін для такої кількості об’єктів є важким процесом, що вимагає постійного аналізу попередніх бронювань, динаміки ринку та дій конкурентів. Це займає багато часу, створює високе навантаження та супроводжується ризиком помилок. Менеджер фізично не встигає своєчасно реагувати на всі зміни, що призводить як до недоотримання прибутку при занижених цінах, так і до втрати потенційних гостей при завищених тарифах.

Додатково ситуацію ускладнює динамічність ринку, де рівень попиту може змінитися навіть протягом одного дня, а конкуренти швидко коригують власні пропозиції. За таких умов ручні методи аналізу стають неефективними, а людський фактор суттєво впливає на якість прийняття рішень.

1.3 Конкурентне середовище ринку короткострокової оренди у Львові

Ринок короткострокової оренди у Львові є динамічним і дуже конкурентним, а якість цифрових рішень компаній суттєво впливає на їхню привабливість для клієнтів. Серед сильних гравців варто відзначити «Інші Апартментс», які мають сучасний сайт і власний модуль бронювання, що забезпечує швидку роботу, зручний інтерфейс і високий рівень конверсії. Їхній технологічний підхід вирізняє компанію серед конкурентів.

Разом із тим на ринку присутні й компанії зі застарілими рішеннями. Наприклад, «Емілі Ресорт» використовує шаблонний сайт із низькою продуктивністю та застарілим дизайном. Незважаючи на це, компанія утримує стабільний потік клієнтів завдяки масштабам і великій базі постійних гостей, а не завдяки інноваціям.

Загальна тенденція свідчить: підприємства, які інвестують у сучасні цифрові сервіси та покращення користувацького досвіду, отримують конкурентну перевагу. Водночас бізнеси, що не оновлюють свої системи, поступово втрачають ефективність. Це формує виразну потребу у нових рішеннях, які поєднують автоматизацію, гнучкість та якісну презентацію об’єктів нерухомості — саме такі вимоги враховано у розроблюваній системі.

1.4 Технологічні обмеження існуючих систем бронювання та PMS-рішень

Більшість сучасних систем бронювання та PMS-рішень у сфері короткострокової оренди забезпечують лише базові функції — управління календарями, бронюваннями та каналами продажу. Проте вони практично не підтримують глибоку аналітику, прогнозування попиту чи автоматичне ціноутворення, що є критично важливим у динамічному ринковому середовищі. Wubook PMS ефективно виконує операційні задачі, але не надає розширених інструментів для оцінки сезонності, аналізу ринку чи адаптивного формування цін. Схожі обмеження мають і глобальні платформи Airbnb та Booking.com, чиї рекомендації є загальними та не враховують індивідуальні дані компанії.

Через відсутність автоматизованої аналітики менеджери змушені покладатися на ручні розрахунки, що уповільнює прийняття рішень і знижує точність ціноутворення. Це підкреслює необхідність створення систем, здатних прогнозувати попит і оптимізувати ціни на основі реальних даних [20].

1.5 Проблеми масштабування при збільшенні кількості апартаментів

У міру зростання бізнесу навантаження на менеджерів збільшується пропорційно кількості апартаментів. Ручне ведення календарів, моніторинг попиту, оновлення цін та контроль каналів стають надзвичайно складними при розширенні портфоліо. Багато компаній стикаються з тим, що збільшення кількості об’єктів призводить до хаосу в управлінні, розпорошення інформації та зниження ефективності через відсутність автоматизованих бізнес-процесів. Це підкреслює потребу у системах, здатних централізувати управління, масштабуватися разом із бізнесом та мінімізувати людський фактор.

Висновки до розділу

У результаті аналізу встановлено, що ринок короткострокової оренди у Львові є динамічним, конкурентним та чутливим до технологічного рівня компаній. Ручне управління цінами, календарями та великою кількістю апартаментів створює значне навантаження на менеджерів і підвищує ризик помилок, що негативно впливає на заповненість та прибутковість. Дослідження конкурентів показало, що компанії, які інвестують у сучасні цифрові рішення та власні модулі бронювання, отримують суттєву перевагу над тими, що використовують застарілі або шаблонні інструменти.

Аналіз існуючих PMS і систем бронювання, зокрема Wubook, підтвердив їхні технологічні обмеження: вони забезпечують лише базові операції та не пропонують достатньої аналітики, прогнозування чи автоматичного ціноутворення. У поєднанні зі швидкими ринковими змінами та відсутністю інструментів моніторингу конкурентів це ускладнює прийняття ефективних управлінських рішень.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Загальна характеристика інформаційного забезпечення

Інформаційне забезпечення є ключовим елементом розроблюваної системи бронювання та аналітики для компанії Royal Apart, оскільки саме правильна організація даних, потоків інформації та технічної інфраструктури визначає ефективність роботи сайту, Telegram-бота, адмін-панелі та цінової аналітики. Система оперує великою кількістю різнорідних даних — від інформації про 54 апартаменти до статистики бронювань, календарів доступності, параметрів цін і показників попиту, що потребує чіткої структури їх збереження, обробки та актуалізації.

Основу інформаційного забезпечення становлять декілька груп даних. До статичних належать характеристики апартаментів, їх фотографії, опис, базові тарифи, технічні параметри та налаштування інтеграції з PMS Wubook. Динамічні дані включають бронювання, зміни цін, оновлення календарів, дані з каналів продажу та результати аналітичних розрахунків. Окрему категорію формують дані, необхідні для функціонування цінового модуля: історія бронювань, показники завантаженості, сезонність, облік конкурентних цін, а також службові структури для побудови моделей прогнозування.

Важливою складовою є взаємодія системи з зовнішніми сервісами, зокрема PMS Wubook, яка забезпечує синхронізацію календарів, отримання нових бронювань і передачу оновлених цін. Така комунікація потребує стабільних API-запитів, правильної структури даних та механізмів обробки можливих збоїв. Внутрішній інформаційний потік пов’язує фронтенд (React/Vite) [13], бекенд (Node.js/Express), базу даних MongoDB, модуль аналітики та інтерфейс Telegram-бота, забезпечуючи єдиний цикл роботи системи в реальному часі.

Значну роль відіграє також система логування: фіксація змін цін, дій адміністратора, помилок API, оновлень календарів і взаємодій користувачів із ботом.

Це дозволяє контролювати працездатність системи, аналізувати поведінку менеджерів, вчасно знаходити неполадки та підтримувати безперервність роботи.

Окрему увагу приділено структурованому зберіганню даних у MongoDB, що забезпечує високу швидкість обробки, масштабованість та гнучкість. Поділ колекцій на блоки апартаментів для різних платформ, користувачів і службових параметрів забезпечує логічність роботи та спрощує експлуатацію системи.

Таким чином, інформаційне забезпечення в проєкті формує повноцінну інфраструктуру, що охоплює збір, обробку, зберігання та аналітичну оцінку даних у режимі реального часу. Воно забезпечує узгоджену роботу всіх компонентів — сайту, бота, адмін-панелі, аналітичного модуля та інтеграцій з PMS — і є основою надійного функціонування та масштабованості створюваної системи.

2.2 Вхідні та вихідні дані системи

Вхідними даними для системи є контактні дані користувача та його вибір доступної квартири. Інформація вводиться через інтерфейс, у якому користувач зазначає своє ім’я, номер телефону, а також обирає житло зі списку актуально вільних варіантів. Дані можуть мати різні формати та не завжди бути структурованими, тому система повинна коректно обробляти неточності, помилки введення та варіації у формі подання. Це забезпечує гнучкість та зручність взаємодії для різних категорій користувачів. Перед збереженням вхідна інформація проходить первинну обробку: перевірку коректності контактних даних, валідацію вибраної квартири, а також нормалізацію формату введених полів. Такий підхід дозволяє зменшити кількість помилок і гарантувати, що в базі даних опиняються повні та коректні записи. Зокрема, перевіряється актуальність статусу квартири, щоб уникнути дублювання бронювань або конфліктів між клієнтами. Вихідними даними системи є сформоване бронювання, яке містить узагальнену інформацію про користувача, вибрану квартиру та час створення заявки. Цей результат надсилається менеджеру, який отримує доступ до контактів клієнта та може оперативно з ним зв’язатися. Таким чином забезпечується можливість уточнити особливі побажання, обговорити умови заселення та

підтвердити деталі бронювання. Окрім основного результату, система генерує службові вихідні дані: записи журналу подій, інформацію про помилки, статуси успішних та невдалих операцій. Ці дані не відображаються користувачеві, але використовуються адміністраторами для підтримки коректної роботи сервера та оптимізації процесів. У кінцевому результаті формується деталізований запис про бронювання, що зберігається в базі і може бути використаний для контролю завантаженості, аналітики та подальшого управління житловим фондом. Саме комбінація вхідних даних від користувача, автоматизованих перевірок і вихідних інформаційних структур забезпечує повноцінну та надійну роботу системи бронювання.

2.3 Використані програмні інструменти, фреймворки та технології

Інформаційне забезпечення програмного комплексу охоплює низку технологій, що формують повну інфраструктуру для роботи сайту, адміністративної панелі, серверної частини, телеграм-бота та модуля машинного навчання. Під час розробки було обрано інструменти, які забезпечують швидкодію, масштабованість та можливість зручної інтеграції між окремими компонентами. Клієнтська частина сайту реалізована з використанням React та Vite, що забезпечують швидке рендерування та оптимізовану збірку. Для стилізації застосовано Tailwind CSS разом із додатковими UI-бібліотеками, а система інтернаціоналізації побудована на i18next. Карти Google, анімації та SEO-компоненти інтегровані через профільні React-бібліотеки, що підвищує функціональність і зручність взаємодії користувачів із системою. Чат-бот взаємодіє із сервером через HTTP-запити. Середовище виконання Node.js та фреймворк Express [12] забезпечують обробку повідомлень і стабільну роботу під час високого навантаження. Додаткові утиліти відповідають за форматування дат, керування змінними середовища та підтримку процесів, що робить бот модульним і легким у розширенні. Адміністративна панель, як і сайт, побудована на React і Vite, а інтерактивні компоненти реалізовані за допомогою Tailwind та tw-

elements-react. Для представлення статистики застосовується Recharts, а підтримка різних форматів даних забезпечується бібліотеками роботи з таблицями та XML.

Підсистема машинного навчання реалізована мовою Python із використанням scikit-learn, pandas та numpy [18]. Моделі прогнозування зберігаються у форматі PKL, що дозволяє швидко завантажувати їх у Flask-сервер та інтегрувати з Node.js [2]. Такий підхід забезпечує стабільну роботу алгоритмів у режимі реального часу та можливість подальшого розширення навчальної вибірки.

Центральний сервер побудований на Express та підключений до MongoDB через Mongoose [1]. Він відповідає за бізнес-логіку, авторизацію , обробку файлів та інтеграцію із зовнішніми сервісами. Додаткові модулі забезпечують роботу з даними, планування задач, відправку email та взаємодію з Python-модулем через python-shell [3]. Загальна архітектура є мікросервісною, кожен компонент працює автономно, а маршрутизація запитів здійснюється через Nginx. Загальна структура контейнерів реалізована за допомогою Docker, а база даних розміщена у хмарному середовищі MongoDB Atlas. Сукупність використаних технологій формує надійну, гнучку та продуктивну систему, яка може масштабуватися відповідно до потреб сервісу.

2.4 Джерела та структура текстових даних

Система використовує два основних типи даних із сервісу WuBook: тарифні ціни та інформацію про реальні бронювання. Вони формують єдину базу, на основі якої створюються ознаки та навчаються моделі машинного навчання. Тарифні дані автоматично отримуються через WuBook API, тоді як дані про фактичні бронювання експортуються вручну. Обидва набори зберігаються у текстових CSV-файлах, що забезпечує простоту обробки та можливість швидкого оновлення.

Тарифні ціни завантажуються за допомогою API-запитів у щотижневому розрізі на рік уперед. Дані включають назву кімнати, дату та вартість, після чого проходять фільтрацію та нормалізацію, а некоректні значення відсіюються. Файли з реальними бронюваннями містять інформацію про створення броні, період проживання та загальну ціну. Під час обробки такі записи конвертуються у денний

формат: кожна доба проживання отримує власну ціну, що дозволяє точніше порівнювати фактичні та тарифні значення. Після завантаження обидва джерела даних уніфікуються в Python-середовищі. На їх основі формуються ознаки, що описують сезонність, попит, коливання цін та заповнюваність кімнат. Система розраховує метрики попиту, аналізує схожі бронювання, визначає тенденції зміни цін та загальні патерни заповнення за попередні місяці. Таке поєднання сирих числових даних із агрегованими статистичними показниками створює збалансоване навчальне середовище для моделей. На основі сформованих ознак створюються вибірки для класифікаційної та регресійної моделей, які визначають адекватність тарифної ціни та прогнозують оптимальну вартість для вибраної дати. Під час навчання застосовується стандартизація ознак та збереження моделі у форматі PKL для подальшого швидкого використання під час передбачень. Таким чином, структура даних системи охоплює автоматично отримані тарифні значення, реальні історичні бронювання та створені на їх основі аналітичні ознаки. Використання CSV-формату, формування денних даних та попереднє опрацювання забезпечують сталість, повторюваність і якість підготовки даних, що є ключовим чинником для побудови точних та надійних моделей прогнозування цін.

2.5 Особливості формування ознак та роботи моделі

У процесі побудови моделі враховано низку важливих технічних нюансів, пов’язаних із якістю джерел, відсутністю окремих даних та специфікою короткострокової оренди. Насамперед було розроблено механізми обробки пропусків: якщо для певної дати або кімнати не знайдено схожих реальних бронювань, система використовує тарифну ціну як базове значення, а відсутні метрики заповнюються нульовими значеннями. Це унеможливлює появу NaN-значень та забезпечує стабільність навчання моделі.

Додатковою проблемою виявилася неоднорідність назв апартаментів у різних джерелах. Для цього реалізовано нормалізацію назв, що дозволяє коректно зіставляти дані навіть за різної пунктуації або форматування. Аналіз часових рядів ґрунтується

на декількох вікнах: 90 днів для оцінки зайнятості та ±30 днів для визначення попиту і пошуку схожих бронювань. Таке комбінування дає змогу моделі враховувати як короткострокові, так і довгострокової тенденції. Для задачі класифікації цін визначено пороги відхилення у межах ±15% від реальної або медіанної ціни, що дозволяє визначати статус ціни як низький, нормальний або завищений. У процесі розрахунку орієнтирів для моделі використовується медіана, оскільки вона стійка до разових аномально дорогих чи дешевих бронювань.

Висновки до розділу

У межах другого розділу було розглянуто структуру інформаційного забезпечення системи динамічного ціноутворення, визначено джерела даних, особливості їх формування та методи попередньої обробки. Аналіз показав, що результативність роботи моделі безпосередньо залежить від узгодженої взаємодії всіх програмних складових — від збору тарифних та реальних цін до створення ознак, навчання моделей і подальшого формування прогнозів. Використання сучасних інструментів обробки даних, машинного навчання та API WuBook забезпечує надійну основу для побудови алгоритмів, здатних працювати з різними типами цінової інформації та часових патернів.

Завдяки такій архітектурі забезпечується масштабованість рішення, його стійкість до змін попиту та можливість функціонування в режимі реального часу, що повністю відповідає вимогам до сучасних систем динамічного ціноутворення в сфері короткострокової оренди.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Формалізація задачі динамічного ціноутворення

Задачу динамічного ціноутворення у сфері короткострокової оренди можна розглядати як процес керованого впливу на ціну, де кожне нове бронювання, зміна календарних умов чи оновлення ринкової ситуації призводить до переходу системи в інший стан. У цьому сенсі цінова модель має виражену математичну структуру: ціна не є випадковим значенням, а формується як результат послідовних перетворень над множиною історичних та поточних параметрів попиту. Це дозволяє описувати роботу системи формальними моделями, які зв’язують між собою календарні фактори, динаміку зайнятості, статистичні властивості минулих цін та поведінкові особливості гостей.

Система бронювання Wubook та її PMS-модуль виступають джерелом станів, що безперервно змінюються. Кожен день для кожного апартаменту можна розглядати як окремий об’єкт із власними характеристиками: рівнем завантаженості, наявністю бронювань, діючою ціною, сезонністю та актуальними параметрами ринку. Додатково на стан впливають події, які зовні не пов’язані з користувачем: відміни бронювань, перенесення дат, поява нових гостей, автоматичні оновлення OTA-платформ. Таким чином, система перебуває у постійному русі, і саме зміна станів визначає математичний зміст задачі прогнозування.

Математичні моделі дозволяють перетворити цей складний та нерівномірний потік даних у керовану структуру. Розбиття бронювання на денні записи, нормалізація цін, облік сезонних коливань і корекція даних створюють основу для формування векторів ознак. На їх основі моделі класифікації та регресії відтворюють логіку ціноутворення на ринку. У підсумку кожний прогноз оптимальної ціни можна інтерпретувати як композицію функцій, що перетворюють вхідний стан апартаменту у новий, економічно обґрунтований ціновий стан. Такий підхід дає змогу розглядати систему ціноутворення не як набір емпіричних рішень, а як формальну модель,

спрямовану на досягнення цільової конфігурації — максимально можливого доходу за умов поточного ринку.

3.2 Математична модель представлення вхідних даних

Формування математичної моделі ціноутворення починається з правильного подання вихідних даних, оскільки саме структура даних визначає подальшу якість прогнозів. Базою для аналізу виступають бронювання, отримані з PMS-системи Wubook, однак самі по собі вони не є зручними для моделювання. Бронювання містять інтервальні дані — дату заїзду, дату виїзду та загальну вартість — тоді як модель працює з дискретними спостереженнями, прив'язаними до кожного календарного дня. Тому першим етапом є перехід від інтервальної структури до денного представлення. Кожне бронювання розбивається на множину елементів, де кожен елемент відповідає окремій добі проживання. Для кожного дня обчислюється ціна за добу, яка визначається як частка від ділення повної вартості бронювання на його тривалість. Такий підхід усуває нерівномірність інтервалів і дозволяє моделі працювати з однорідними даними, у яких усі спостереження мають однакову часову поширеність. Я обрав дані про бронювання за минулий рік. Після первинної обробки дані завантажуються для подальшого аналізу (рис 3.1).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Code	Created	Agency	Country	Price	From	To	Nights	Room daily pri	Type Code	Type Nam	Room Cod	Room Name
2	77-2285	30.08.2024			23400.0	01.09.2024	08.09.2024	7	3342.86	fr14	Франка 14	fr14	Франка 14
3	77-2290	31.08.2024			3000.0	01.09.2024	02.09.2024	1	3000.0	mos4	Мосяжна	mos4	Мосяжна 4
4	99-1317	01.09.2024			1600.0	01.09.2024	02.09.2024	1	1600.0	vr13	Вірменськ	vr13	Вірменська 13
5	99-1318	01.09.2024			1400.0	01.09.2024	02.09.2024	1	1400.0	S381	Шевченка	S381	Шевченка 38/1
6	99-1326	01.09.2024			1500.0	01.09.2024	02.09.2024	1	1500.0	hm37	Хмельниц	hm37	Хмельницького 37
7	AM-0086	01.09.2024	Booking [3431808]	Ukraine	1350.0	01.09.2024	02.09.2024	1	1350.0	ts9	Цехова 9	ts9	Цехова 9
8	AP-0084	01.09.2024	Airbnb [429214186]		1066.0	01.09.2024	02.09.2024	1	1066.0	kz26	Ковжуна 2	kz26	Ковжуна 2/6
9	DK-0052	26.08.2024	Booking [7445021]	Ukraine	1400.0	01.09.2024	02.09.2024	1	1400.0	sr6	Сербська	sr6	Сербська 6
10	HD-0003	01.09.2024	Booking [7476763]	Ukraine	13600.0	01.09.2024	06.09.2024	5	2720.0	b23	Балабана	b23	Балабана 23
11	KR-0010	29.08.2024	Booking [12522356]	Ukraine	1250.0	01.09.2024	02.09.2024	1	1250.0	kz28	ковжуна 2	kz28	ковжуна 2/8
12	OL-0034	30.08.2024	Airbnb [429214186]		1804.12	01.09.2024	02.09.2024	1	1804.12	ch1	Чорновол	ch1	Чорновола 1
13	OT-0034	29.08.2024	Booking [10075590]	Ukraine	1700.0	01.09.2024	02.09.2024	1	1700.0	sh60	Шевченка	sh60	Шевченка 60
14	PZ-0004	29.08.2024			134000.0	01.09.2024	31.10.2024	60	2233.33	gl5	Глібова 5	gl5	Глібова 5
15	RP-0010	31.08.2024	Booking [6423774]	Ukraine	1500.0	01.09.2024	02.09.2024	1	1500.0	pd24	Підмурна	pd24	Підмурна 24
16	VF-0010	21.08.2024	Booking [7544655]	Ukraine	2500.0	01.09.2024	02.09.2024	1	2500.0	nl13	Наливайк	nl13	Наливайка 13
17	99-1305	28.08.2024			3000.0	02.09.2024	03.09.2024	1	3000.0	gr31	Городоць	gr31	Городоцька 31
18	99-1319	01.09.2024			1800.0	02.09.2024	03.09.2024	1	1800.0	ts9	Цехова 9	ts9	Цехова 9
19	99-1323	01.09.2024			1600.0	02.09.2024	03.09.2024	1	1600.0	pd24	Підмурна	pd24	Підмурна 24
20	99-1324	01.09.2024			3200.0	02.09.2024	03.09.2024	1	3200.0	kl47	Куліша 47	kl47	Куліша 47а
21	99-1327	02.09.2024			4000.0	02.09.2024	04.09.2024	2	2000.0	ch26	Чайковськ	ch26	Чайковського 26
22	99-1331	02.09.2024			19400.0	02.09.2024	08.09.2024	6	3233.33	fr14	Франка 14	fr14	Франка 14
23	99-1333	02.09.2024			22200.0	02.09.2024	09.09.2024	7	3171.43	fr14	Франка 14	fr14	Франка 14
24	99-1335	02.09.2024			1800.0	02.09.2024	03.09.2024	1	1800.0	fr28	Франка 28	fr28	Франка 28
25	MH-0020	01.09.2024	Airbnb [429214186]		2706.0	02.09.2024	04.09.2024	2	1353.0	gz10	Газова 10	gz10	Газова 10
26	NF-0004	25.08.2024	Booking [3431808]	Ukraine	1665.0	02.09.2024	03.09.2024	1	1665.0	bal3	Балабана	bal3	Балабана 3

Рисунок 3.1 – Приклад реальних даних після первинної обробки

Наступною проблемою є наявність пропусків. Частина днів не має реальних цін, оскільки апартаменти могли бути вільними або бронювання не збереглося в історії. Для таких випадків використовується комбінація методів: заповнення пропусків значенням тарифної ціни (рис 3.2), встановленої в PMS-системі, а також заміна некоректних або відсутніх значень нулями. Такий підхід забезпечує стабільність подальшого навчання моделі, оскільки алгоритми машинного навчання не працюють з NaN-значеннями.

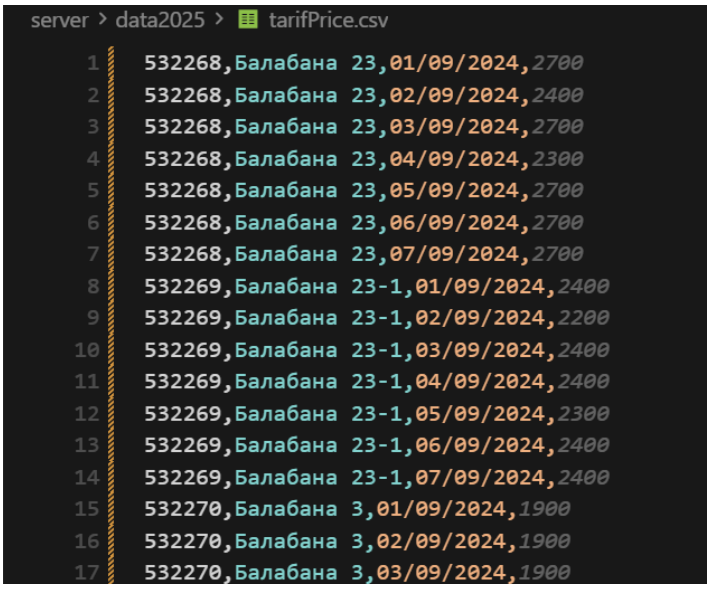


Рисунок 3.2 – Приклад тарифних даних після первинної обробки

Паралельно здійснюється нормалізація структурованих даних, що включає приведення назв кімнат до єдиного формату. Оскільки в Wubook назва однієї і тієї ж кімнати може містити різні варіації написання, регістру чи роздільників, система застосовує правила стандартизації, що дозволяють стабільно ідентифікувати об’єкти. Це є критично важливим для коректного підбору схожих бронювань і формування статистичних ознак.

У підсумку формується математично узгоджена структура даних: кожен день для кожного апартаменту представлений у вигляді векторного запису з набором характеристик, що включають денну ціну, параметри попиту, ознаки сезонності та

інформацію про зайнятість. Така однорідна та систематизована база стає основою для подальшого побудування моделей прогнозування та класифікації.

3.3 Методологія побудови ознак

Побудова ефективної моделі прогнозування оптимальної ціни та класифікації тарифних значень неможлива без ретельно спроєктованого набору ознак. У межах даного дослідження було сформовано 20 інформативних ознак, що описують як календарний контекст, так і поведінкові характеристики попиту, історичні закономірності ціноутворення та стан зайнятості апартаментів. Методологія їх формування базується на комбінуванні часових факторів, статистичних властивостей реальних цін та динамічних метрик ринку.

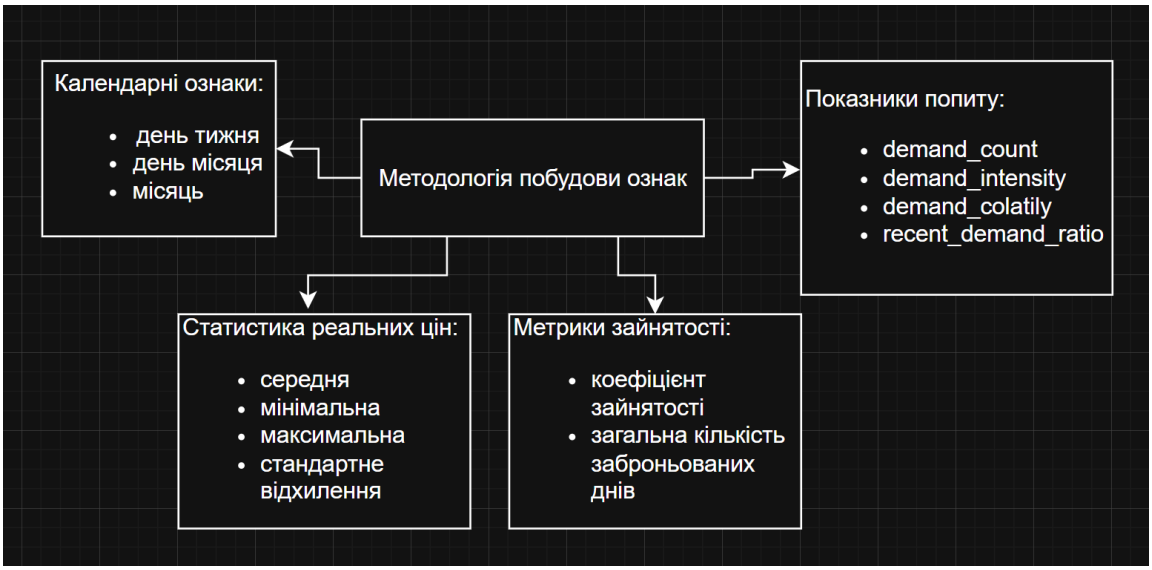


Рисунок 3.3 – Приклад тарифних даних після первинної обробки

Першу групу становлять календарні ознаки, що відображають сезонні й тижневі патерни попиту. До них належать день тижня, день місяця та місяць, які дозволяють моделі враховувати періодичні зміни попиту, характерні для індустрії короткострокової оренди. Наприклад, вихідні та святкові періоди часто супроводжуються підвищенням цін, тоді як будні дні — стабільнішими тарифами.

Друга група містить статистики реальних цін, отриманих із вікна схожих бронювань за ±365 днів. Серед них — середня, мінімальна, максимальна ціна, стандартне відхилення та кількість доступних спостережень. Ці показники

дозволяють моделі орієнтуватися на історичний контекст кожної конкретної дати, а також розуміти, наскільки стабільною є цінова політика у певний період. Показник has_real_price фіксує факт наявності або відсутності реальних даних, що зменшує ризик помилок у випадках, коли система змушена використовувати fallback.

Третя група охоплює метрики зайнятості, що розраховуються у вікні 90 днів назад. Коефіцієнт зайнятості, загальна кількість заброньованих днів та середня тривалість бронювання дозволяють встановити зв'язок між рівнем заповненості та цінами, оскільки високий попит у минулому часто корелює з підвищенням вартості.

Окремий блок становлять показники попиту, що враховують кількість активних бронювань у часовому вікні ±365 днів. До них належать: demand_count, demand_intensity, demand_volatility та recent_demand_ratio. Ці ознаки характеризують амплітуду коливань попиту, його стабільність і відносну силу поточної тенденції. Наприклад, співвідношення попиту в останні 30 днів до попиту у попередні 30 днів дозволяє моделі реагувати на зміну сезонності або нові ринкові умови [11].

Остання група включає цінові тренди, що відображають динаміку зміни цін у часі. Ознака price_trend показує, наскільки середня ціна зростає або падає порівняно з попереднім періодом, і тим самим дозволяє моделі враховувати не лише статичні характеристики, а й напрям розвитку цін.

Усі ці ознаки формуються автоматизовано, із застосуванням процедур очищення та нормалізації. Така комплексна методологія дозволяє звести багатовимірні дані про ринок короткострокової оренди до компактного та інформативного набору числових параметрів, який забезпечує високу точність і стійкість моделі до шумів та коливань у вихідних даних.

3.4 Математичні моделі класифікації тарифної ціни

Класифікація тарифної ціни в рамках побудованої системи передбачає визначення статусу кожної прогнозованої ціни відносно історичного та ринкового контексту. Метою цього етапу є не лише передбачення числового значення, але й встановлення, чи є ціна заниженою, типовою або завищеною для конкретної дати та

апартаментів. Такий підхід дозволяє автоматизувати контроль цінової політики, уникати недоотримання доходу та своєчасно реагувати на зміни ринкового попиту (3.1).

Для цілей дослідження було введено три класи тарифного статусу:

- низька ціна (Low) – якщо прогнозована ціна істотно нижча за історично типовий рівень;
- нормальна ціна (Normal) – якщо значення відповідає очікуваному діапазону;
- висока ціна (High) – якщо встановлена вартість суттєво перевищує середній рівень для відповідних умов.

Для формального поділу цін використовується індекс відхилення I_p, що порівнює поточну ціну p із середньою історичною p_{mean} у вікні 30 днів. Статус тарифу визначається саме за значенням цього індексу.

$$I_p = \frac{p}{p_{mean}} \qquad (3.1)$$

Пороги класифікації: Low, p < 0.85 * p_mean, Normal, 0.85 * p_mean ≤ p ≤ 1.15 * p_mean, High, p > 1.15 * p_mean

Для розв'язання задачі багатокласової класифікації було обрано алгоритм RandomForestClassifier, що реалізує ансамблевий підхід і об'єднує множину незалежних дерев рішень. Процес навчання моделі базується на мінімізації сумарної втрати ансамблю, яка є похідною від невідповідностей між прогнозованими та реальними класами (3.2). Кожне дерево в ансамблі виконує послідовне розбиття ознак за критерієм максимальної інформативності, яку вимірюють через зменшення імпурності. У дослідженні як матрицю нечіткості використовується критерій Джині:

$$G = 1 - \sum_{i=1}^{3} p_i^2 \qquad (3.2)$$

де p_i — ймовірність належності об'єктів у вузлі до класу i.

Зменшення цього показника визначає, наскільки розбиття покращує структуру даних.

RandomForestClassifier формує прогноз шляхом агрегації голосів усіх дерев. Остаточний клас призначається за принципом максимальної частоти або за усередненими ймовірностями.

Для оцінювання роботи моделі застосовуються такі метрики:

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix
- Feature Importance

Для підвищення надійності оцінювання використовувалася крос-валідація, що зменшує вплив вибірки та дозволяє отримати узагальнені результати.

3.5 Модель регресійного прогнозування оптимальної ціни

Побудова моделі прогнозування оптимальної ціни є ключовим елементом математичного забезпечення системи, оскільки саме цей компонент визначає здатність алгоритму рекомендувати коректну, ринково обґрунтовану вартість для заданої дати [9]. На відміну від задачі класифікації (рис. 3.4), де визначається лише статус тарифного значення, задача регресії передбачає обчислення точного числового прогнозу, що потребує відповідного підходу до структурування ознак, вибору моделі та оцінювання її якості.

Метою регресійного прогнозу є визначення такої ціни, яка найбільш точно відповідає поведінковим патернам ринку короткострокової оренди. У моделі використовується той самий набір із 20 сформованих ознак, проте їхня роль у прогнозуванні є більш безпосередньою — кожен параметр розглядається як незалежна змінна, що впливає на кінцеве значення ціни.

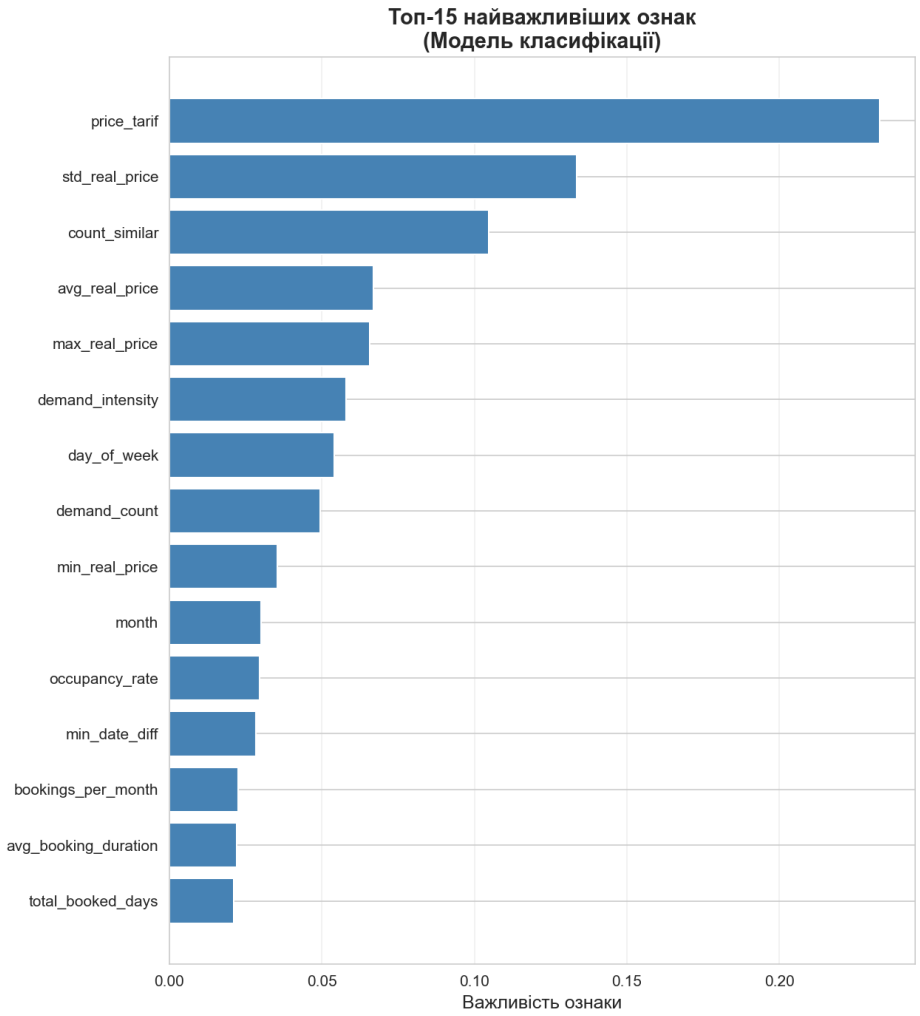


Рисунок 3.4 – Важливість ознак для моделі класифікації

3.6 Оцінювання якості математичної моделі класифікації

Нормалізація й очищення ознак є критичним етапом підготовки даних, оскільки забезпечують стабільну роботу моделей та зменшують вплив шумів. У межах проєкту виконано кілька узгоджених процедур, спрямованих на уніфікацію форматів і підвищення якості ознак.

Для числових параметрів застосовано StandardScaler, який приводить дані до стандартного нормального розподілу. Це дозволяє уникнути домінування ознак із більшими діапазонами та забезпечує коректне навчання моделей, особливо чутливих до масштабів даних [8].

Проблема пропусків вирішувалася за допомогою fallback-значень: числові ознаки заповнювалися медіаною або нульовими значеннями, а категоріальні — узагальненим класом «unknown». Такий підхід дозволив зберегти повноту вибірки та уникнути викривлення статистичних характеристик.

Важливим елементом була нормалізація назв кімнат та категорій проживання. Різні формати записів (“Standard”, “std”, “Std room”) призводять до дублювання категорій, тому було виконано очищення від символів, приведення до нижнього регістру та мапінг на єдиний набір стандартних назв.

Для забезпечення надійності цінових ознак здійснювалася обробка аномалій і викидів. Виявлення виконувалося за статистичними правилами (IQR, z-score) у поєднанні з доменними порогами тарифів. Аномальні записи коригувалися або виключалися, що дозволило усунути вплив некоректних або випадкових значень.

Застосування зазначених технік забезпечило формування чистого, узгодженого та масштабованого набору ознак, який став основою для подальшої побудови моделей класифікації та регресійного прогнозування.

3.7 Математична інтерпретація навчання та перенавчання моделі

Алгоритм інтегрального прогнозу поєднує результати класифікаційної та регресійної моделей, що дозволяє отримати як статус ціни (низька, нормальна, висока), так і точне значення оптимального тарифу. Класифікатор визначає загальну категорію ціни, тоді як регресійна модель формує числовий прогноз, що враховує динаміку попиту, історичні дані та поведінкові закономірності [7].

Інтеграція результатів здійснюється через перевірку їх узгодженості. Якщо класифікатор сигналізує про завищення або заниження ціни, а регресійна модель дає суттєво інше значення, застосовується коригування. Такий підхід дозволяє уникнути нестабільних або нереалістичних прогнозів та забезпечує найбільш збалансовану рекомендацію.

Обчислення втрат доходу виконується на основі різниці між оптимальною та фактичною ціною. Якщо поточна ціна нижча за рекомендовану — виникає

недоотриманий дохід; якщо вища — збільшується ризик втрати попиту. Значення втрат подається користувачу як індикатор економічної ефективності прогнозу.

Після оновлення даних система автоматично перераховує інтегральний прогноз та потенційні втрати, забезпечуючи адаптивність моделі та постійне удосконалення рекомендацій у змінних ринкових умовах.

Висновки до розділу

У цьому розділі було сформовано математичну основу системи прогнозування оптимальної ціни та класифікації тарифних значень для ринку короткострокової оренди. Було розроблено комплексний набір ознак, що охоплює календарні параметри, статистики реальних цін, показники попиту, метрики зайнятості та цінові тренди. Їх поєднання дозволяє моделі враховувати як статичні закономірності ринку, так і динамічні коливання, що забезпечує високу інформативність простору ознак.

У межах розділу були розглянуті математичні моделі класифікації та регресійного прогнозування, методи нормалізації даних, обробки аномалій і підходи до інтеграції результатів обох моделей. Застосування алгоритмів RandomForestClassifier та регресійних моделей надало можливість отримати точні рекомендації щодо тарифу, а механізм обчислення втрат доходу — оцінити економічний ефект прийняття рішень.

Ретельно розроблені математичні процедури забезпечують стабільність, гнучкість та адаптивність системи, дозволяючи їй ефективно реагувати на зміни ринкової динаміки. Побудована модель є придатною для використання в реальних умовах, забезпечує високий рівень точності прогнозування та створює підґрунтя для подальшого розширення можливостей системи.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Архітектура та загальна структура програмної системи

Розроблена система управління бронюванням апартаментів «Royal Apart» побудована за мікросервісною архітектурою з використанням контейнеризації Docker (рис 4.1). Такий підхід забезпечує незалежність компонентів, масштабованість і зручність супроводу. Система складається з чотирьох основних рівнів: клієнтського інтерфейсу, серверної логіки, сервісу машинного навчання та рівня даних.

Рисунок 4.1 – Важливість ознак для моделі класифікації

Клієнтський рівень включає вебсайт для гостей (React SPA), адміністративну панель (React) та чат-бот. Усі клієнтські застосунки працюють через Nginx і взаємодіють із системою виключно через REST API.

Рівень бізнес-логіки реалізований у вигляді окремого Node.js/Express API-сервера, що відповідає за обробку запитів, управління апартаментами,

користувачами, бронюваннями, цінами, акціями, а також забезпечує інтеграцію з WuBook і відправку email-повідомлень.

Модуль машинного навчання побудований на Python/Flask і функціонує як незалежний сервіс. Він виконує підготовку даних, формування ознак, навчання моделей прогнозування попиту й цін, а також надає API для отримання передбачень. Для регресії використовуються RandomForestRegressor, StandardScaler та збережені моделі (.pkl).

Рівень даних складається з MongoDB Atlas (оперативні дані), CSV-файлів (історичні тарифні та реальні ціни) та файлового сховища зображень. Усі компоненти об'єднані у Docker-середовище, де Nginx виступає як reverse proxy з повною підтримкою HTTPS/SSL.

Архітектура забезпечує прозору взаємодію компонентів через HTTP-запити, дозволяє масштабувати ключові модулі та підтримує автоматизацію робочих процесів (оновлення цін, синхронізація бронювань, генерація кешованих датасетів). Така структура відповідає вимогам до комплексної системи аналітики та бронювання апартаментів.

4.2 Реалізація серверного застосунку

Серверна частина реалізована на Node.js з використанням Express.js. Сервер надає REST API для взаємодії з клієнтськими додатками та інтеграцією з сервісами машинного навчання. Серверна частина програмної системи «Royal Apart» реалізована на платформі Node.js із використанням фреймворку Express.js, що забезпечує гнучку побудову REST API та високу продуктивність при обробці паралельних запитів. Сервер виступає центральним елементом бізнес-логіки та координує взаємодію клієнтських застосунків, модулів машинного навчання, бази даних і зовнішніх сервісів [3].

Уся серверна логіка розміщена в каталозі server/ і структурована за модульним принципом. Основними складовими є:

- server.js – головний файл ініціалізації;
- routes/ – каталог маршрутизаторів REST API;
- models/ – Mongoose-моделі MongoDB;
- data2025/ – історичні CSV-дані для ML та аналітики;
- imgs/, imgsRoyal/ – файлове сховище зображень;

Головний файл server.js ініціалізує Express-додаток, налаштовує middleware та підключає маршрути (рис. 4.2)

Рисунок 4.2 – Фрагмент ініціалізації сервера та підключення шляхів

На етапі ініціалізації відбувається налаштування ключових компонентів системи. Спершу конфігурується CORS-політика для безпечної взаємодії з фронтендом, а також підключається middleware для обробки JSON-запитів. Система налаштовує видачу статичних файлів (зображень та документів) і забезпечує логування запитів через morgan або власні інструменти. Далі, відповідно до бізнес-логіки, підключаються маршрутизатори та встановлюється з'єднання з MongoDB через Mongoose. Завершує конфігурацію активація централізованого обробника помилок (global error handler) [4].

Для зберігання інформації про апартаменти, користувачів, бронювання, акції та зовнішні інтеграції у системі використовується хмарна база даних MongoDB Atlas [6].

Усі операції з базою виконуються за допомогою ODM-бібліотеки Mongoose, що забезпечує: типізацію документів, валідацію даних, автоматичне створення колекцій, зручні CRUD-операції, підтримку timestamps та індексів. Найважливішою моделлю збереження даних є структура збереження окремих апартаментів (рис. 4.3).

```
4
5   const roomSchema = mongoose.Schema({
6     name: { type: String, required: true },
7     numrooms: { type: Number, required: true },
8     description: { type: String, required: true },
9     price: { type: Number, required: true },
10    wubid: { type: Number },
11    imgurl: [],
12    currentbookings: []
13  }, { timestamps: true });
14
15  const rmModel = mongoose.model("aparts", roomSchema);
```

Рисунок 4.3 – Модель для збереження апартаментів у mongo db

В силу особливостей роботи з даними через чат бота, я вирішив усі апартаменти розділити на 2 колекції. В першій колекції для швидкої відповіді чат-боту викликається колекція з одним фото квартири, яке зберігається локально на сервері (рис 4.4). А у іншій колекції зберігаються апартаменти з усіма зображеннями, як

посилання на сторонній сервіс. Це допомогло мені оптимізувати швидкість. Також для правильної роботи чат боту я зберігаю дані користувача. Такі як його ім’я, айді чату, номер, індекс останнього повідомлення і тд.

server > models > JS users.js > ...

```
const mongoose = require("mongoose");

const userSchema = mongoose.Schema(
  {
    chatId: {
      type: String,
      required: true,
    },
    checkedRooms: {
      type: Object,
    },
    markup: {
      type: String,
    },
    lastMessage: { type: String },
    lastMessageId: {
      type: Number,
    },
    chkin: { type: String },
    chkout: { type: String },
    name: { type: String },
    coment: { type: String },
    context: { type: String },
    phone: { type: Number },
    currentroom: { type: Object },
    roomsid: [],
  },
  { timestamps: true }
);

const urModel = mongoose.model("users", userSchema);
module.exports = urModel;
```

Рисунок 4.4 – Модель для збереження апартаментів у mongo db

Маршрут отримання всіх апартаментів за адресою GET /api/aparts/ виконує роль базового запиту для завантаження каталогу нерухомості (рис. 4.5). Він повертає колекцію для чат-боту. Під час виклику цього маршруту сервер звертається до колекції aparts у MongoDB та за допомогою Mongoose виконує операцію find(), що повертає всі наявні записи без фільтрації. Результат формується у вигляді масиву об’єктів, кожен з яких містить повний набір характеристик квартири: назву, кількість кімнат, опис, базову ціну, зображення та ідентифікатор у системі WuBook. Отримані

дані повертаються клієнту у форматі JSON, що дозволяє чат-боту відображати повний список доступних апартаментів.

```
router.get("/", async (req, res) => {
  try {
    // Знаходимо всі квартири в базі даних
    let rooms = await Roomsr.find({});

    return res.json({ data: rooms });
  } catch (error) {
    console.log("Error fetching rooms:", error);
    res.status(500).json({
      error: "Server error"
    });
  }
});
```

Рисунок 4.5 – Запит для отримання усіх апартаментів для чатботу

Також я створив окремий запит для створення апартаментів (рис 4.6). Завдяки цьому я маю повний контроль над додаванням нових квартир.

```
router.post("/newRoom", upload.single("image"), async (req, res) => {
  try {
    const {
      address, body, category, roomcount, price, floor, beds, guests, square, wubid,
    } = req.body;
    if (!req.file) {
      return res.status(400).json({ message: "Image file is required" });
    }
    const newRoom = new Roomsr({
      name: address,
      numrooms: roomcount,
      description: body,
      category: category,
      price: price,
      imgurl: req.file.filename,
      beds: beds,
      guests: guests,
      floor: floor,
      surface: square,
      wubid: wubid,
    });
    await newRoom.save();
    res.status(201).json({ message: "Room created successfully" });
  } catch (error) {
    console.error("Error saving room:", error);
    res.status(500).json({ message: "Failed to save room" });
  }
});
```

Рисунок 4.6 – Запит для додавання нових апартаментів

Не менш важливим є правильне видалення апартаментів, не лише з бази даних і колекцій, але й сам файл зображення. Тому я прибираю непотрібні фото і не заповню сервер не актуальними апартаментами (рис. 4.7). Видалення здійснюється по айді і це допомагає уникнути видалення не тієї квартири.

```
router.delete("/:id", async (req, res) => {
  try {
    const roomId = req.params.id;

    // Знаходимо квартиру
    const room = await Roomsr.findById(roomId);

    if (!room) {
      return res.status(404).json({
        error: "Room not found"
      });
    }

    // Видаляємо зображення з файлової системи
    const imagePath = path.join(
      __dirname,
      "../imgs",
      room.imgurl[0]
    );

    if (fs.existsSync(imagePath)) {
      fs.unlinkSync(imagePath);
    }

    // Видаляємо квартиру з бази даних
    await Roomsr.findByIdAndDelete(roomId);

    res.json({
      message: "Room deleted successfully"
    });
  } catch (error) {
    console.error("Error deleting room:", error);
    res.status(500).json({
      error: "Server error"
    });
  }
});
```

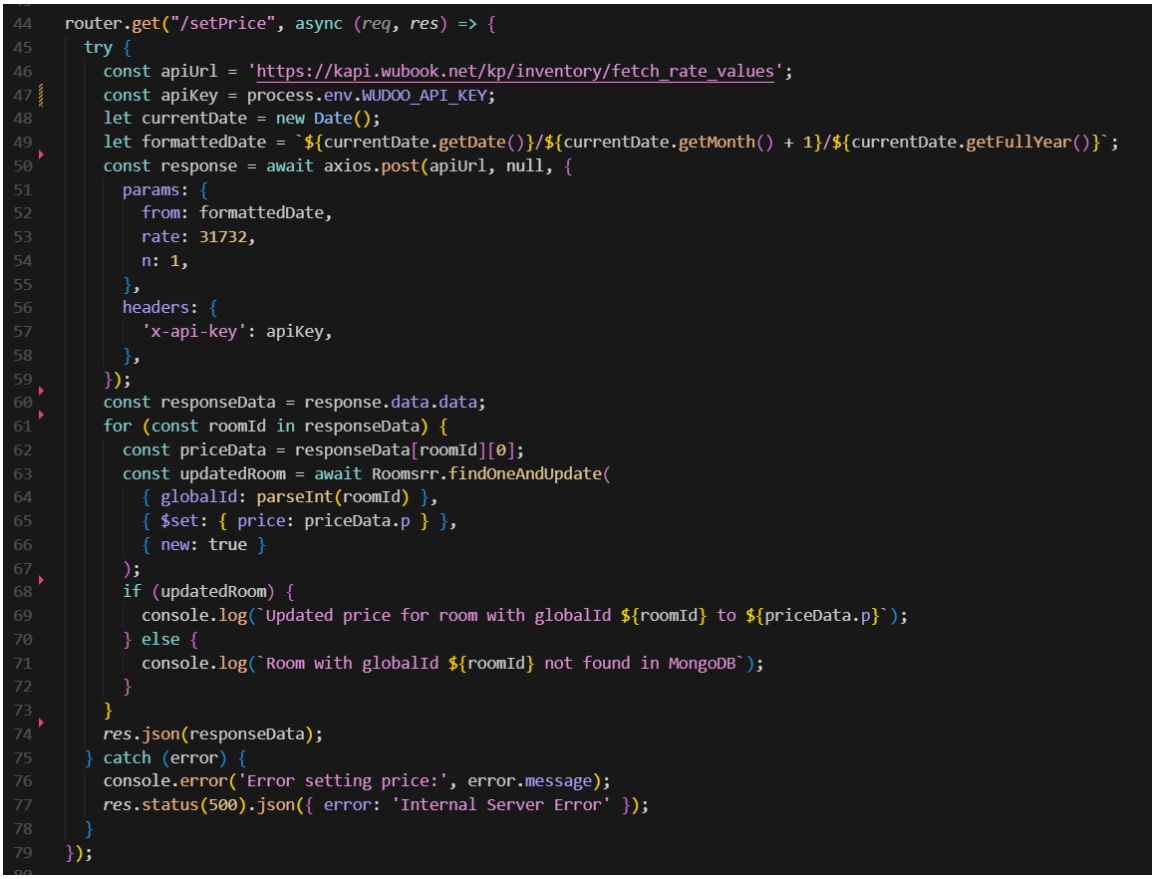
Рисунок 4.7 – Запит для видалення нових апартаментів

Маршрут GET /api/aparts/roomType використовується для отримання списку типів кімнат із зовнішньої системи WuBook. Під час виклику цей запит надсилає POST-запит до ендпоінту fetch_rooms сервісу WuBook API, передаючи API-ключ у заголовках для автентифікації. WuBook повертає повну інформацію про всі апартаменти, що належать об’єкту: їх ідентифікатори, назви, місткість, мінімальні та максимальні ціни, типи розміщення, доступність та інші параметри. Сервер просто повертає відповідь назад клієнту в форматі JSON. Цей маршрут використовується для синхронізації внутрішньої бази апартаментів із зовнішньою системою бронювань, що дозволяє оновлювати структуру номерів та підтримувати узгодженість інформації між Royal Apart і WuBook (рис. 4.8).

```
router.get("/roomType", async (req, res) => {
  try {

    const response = await axios.post(
      "https://kapi.wubook.net/kp/property/fetch_rooms",
      {},
      {
        headers: {
          "x-api-key": process.env.WUDOO_API_KEY,
          "Content-Type": "application/json",
          "Access-Control-Allow-Origin": "*",
          "Access-Control-Allow-Methods": "GET,PUT,POST,DELETE,PATCH,OPTIONS",
        },
      }
    );

    res.json(response.data);
  } catch (error) {

    console.error("Error:", error.message);
    res.status(500).json({ error: "Internal Server Error" });
  }
});
```

Рисунок 4.8 – Запит для синхронізації апартаментів з Wubook

Оскільки у чат бот не можна інтегрувати модуль бронювання wubook, мені довелося робити його самому. І для цього я мушу регулярно оновлювати ціни на актуальні і зберігати їх у колекції з апартаментами (рис. 4.9). Таким чином забезпечується автоматична підтримка актуального тарифу, який використовується як у логіці бронювання, так і в модулі машинного навчання, що базує свої прогнози на фактичних значеннях. Регулярне оновлення тарифів гарантує коректність бізнес-процесів, мінімізує ризик помилок та дозволяє системі працювати незалежно від сторонніх сервісів.

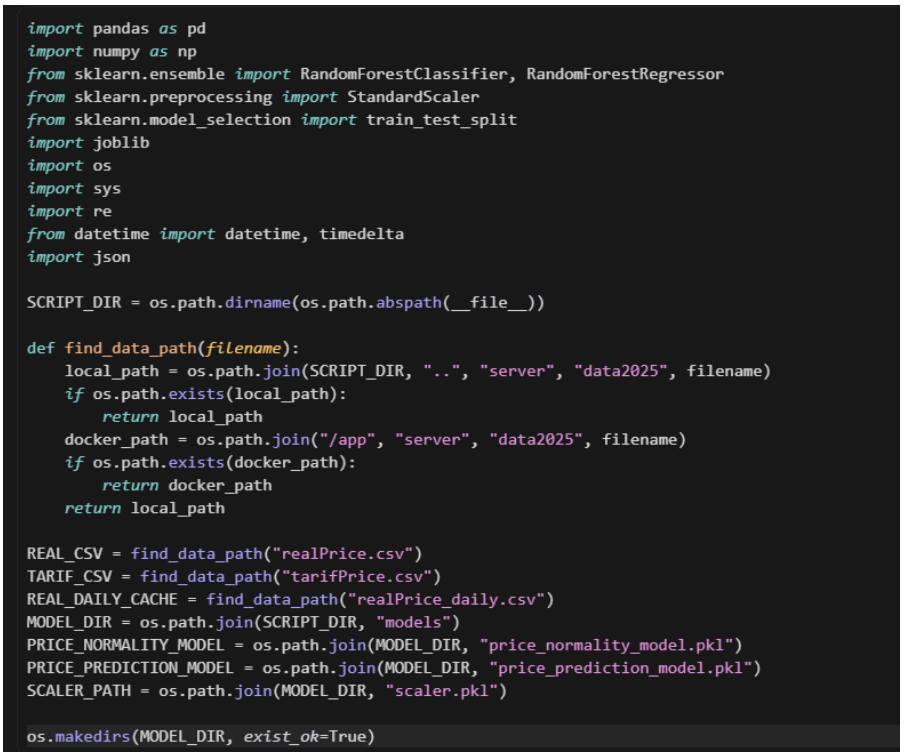


```
router.get("/setPrice", async (req, res) => {
  try {
    const apiUrl = 'https://kapi.wubook.net/kp/inventory/fetch_rate_values';
    const apiKey = process.env.WUDOO_API_KEY;
    let currentDate = new Date();
    let formattedDate = `${currentDate.getDate()}/${currentDate.getMonth() + 1}/${currentDate.getFullYear()}`;
    const response = await axios.post(apiUrl, null, {
      params: {
        from: formattedDate,
        rate: 31732,
        n: 1,
      },
      headers: {
        'x-api-key': apiKey,
      },
    });
    const responseData = response.data.data;
    for (const roomId in responseData) {
      const priceData = responseData[roomId][0];
      const updatedRoom = await Roomsrr.findOneAndUpdate(
        { globalId: parseInt(roomId) },
        { $set: { price: priceData.p } },
        { new: true }
      );
      if (updatedRoom) {
        console.log(`Updated price for room with globalId ${roomId} to ${priceData.p}`);
      } else {
        console.log(`Room with globalId ${roomId} not found in MongoDB`);
      }
    }
    res.json(responseData);
  } catch (error) {
    console.error('Error setting price:', error.message);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```

Рисунок 4.9 – Запит для синхронізації цін на апартаменти з Wubook

4.3 Модуль машинного навчання та робота з моделлю

Модуль машинного навчання у системі «Royal Apart» розпочинається з підключення необхідних бібліотек і формування базової інфраструктури для роботи з даними, навчання моделей, обробки ознак і збереження артефактів навчання. Основними інструментами є бібліотека pandas [17], яка використовується для зчитування та аналізу табличних даних, NumPy для роботи з масивами, а також фреймворк scikit-learn [16], який забезпечує реалізацію моделей класифікації та регресії, стандартизацію ознак і підготовку навчальних вибірок. Для збереження натренованих моделей застосовується бібліотека joblib, оскільки вона найбільш оптимальна для роботи з великими структурами даних та об’єктами моделей машинного навчання (рис 4.10).



```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import joblib
import os
import sys
import re
from datetime import datetime, timedelta
import json

SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))

def find_data_path(filename):
    local_path = os.path.join(SCRIPT_DIR, "..", "server", "data2025", filename)
    if os.path.exists(local_path):
        return local_path
    docker_path = os.path.join("/app", "server", "data2025", filename)
    if os.path.exists(docker_path):
        return docker_path
    return local_path

REAL_CSV = find_data_path("realPrice.csv")
TARIF_CSV = find_data_path("tarifPrice.csv")
REAL_DAILY_CACHE = find_data_path("realPrice_daily.csv")
MODEL_DIR = os.path.join(SCRIPT_DIR, "models")
PRICE_NORMALITY_MODEL = os.path.join(MODEL_DIR, "price_normality_model.pkl")
PRICE_PREDICTION_MODEL = os.path.join(MODEL_DIR, "price_prediction_model.pkl")
SCALER_PATH = os.path.join(MODEL_DIR, "scaler.pkl")

os.makedirs(MODEL_DIR, exist_ok=True)
```

Рисунок 4.10 – Початок модуля з налаштуванням шляхів і бібліотек

Функція find_room_matches відповідає за пошук відповідних квартир у датасеті на основі нормалізованої назви. Оскільки дані про апартаменти можуть містити різні варіанти написання (наприклад, відмінності в пробілах, регістрі, дефісах або скороченнях), функція виконує багатоступеневий механізм порівняння, що дозволяє знаходити потрібний запис навіть у разі часткової невідповідності текстових полів. Спочатку вона здійснює повний точний пошук: порівнює нормалізовану вхідну назву з нормалізованими значеннями у стовпці room (рис. 4.11).

У разі відсутності точного збігу функція переходить до часткового пошуку, де назва розбивається на окремі ключові слова. Далі для кожного слова виконується послідовне фільтрування датасету, і в результаті повертаються всі записи, у яких нормалізована назва містить усі основні компоненти пошукового запиту. Такий підхід дозволяє коректно обробляти ситуації, коли назви квартир відрізняються додатковими словами або мають іншу структуру. На жаль, при роботі з Wubook API я помітив дуже багато неточностей у зберіганні даних і їх відмінності.

```python
def find_room_matches(normalized_room, real_df):
    if not normalized_room:
        return pd.DataFrame()

    real_df_normalized = real_df.copy()
    real_df_normalized['room_normalized'] = real_df_normalized['room'].apply(normalize_room_name)

    exact_matches = real_df_normalized[real_df_normalized['room_normalized'] == normalized_room]

    if len(exact_matches) > 0:
        return exact_matches.drop(columns=['room_normalized'])

    room_words = [w for w in normalized_room.split() if len(w) > 1]

    if not room_words:
        return pd.DataFrame()

    mask = pd.Series([True] * len(real_df_normalized))
    for word in room_words:
        mask = mask & real_df_normalized['room_normalized'].str.contains(word, case=False, na=False, regex=False)

    partial_matches = real_df_normalized[mask]

    if len(partial_matches) > 0:
        return partial_matches.drop(columns=['room_normalized'])

    if len(room_words) > 0:
        first_word = room_words[0]
        first_word_matches = real_df_normalized[
            real_df_normalized['room_normalized'].str.startswith(first_word, na=False)
        ]
        if len(first_word_matches) > 0:
            return first_word_matches.drop(columns=['room_normalized'])

    return pd.DataFrame()
```

Рисунок 4.11 – Функція find_room_matches

Блок аналітики та машинного навчання виконує комплексну оцінку поведінки попиту та цін на апартаменти на основі історичних даних. Спочатку функція calculate_room_occupancy_metrics аналізує зайнятість конкретної квартири за останні 365 днів. Вона визначає частку днів, коли апартаменти були заброньовані, обчислює середню тривалість бронювань, кількість бронювань на місяць та волатильність попиту. Послідовні календарні дні автоматично об’єднуються в одне бронювання, що дозволяє моделі отримати більш реалістичну оцінку поведінки гостей та тривалості їхнього перебування.

Паралельно функція calculate_demand_metrics аналізує інтенсивність попиту навколо цільової дати у 30-денному вікні. Вона рахує кількість днів із бронюваннями, співвідносить недавній попит із попереднім, оцінює тренд зміни реальної ціни та загальну активність гостей у цей період. Завдяки цьому можна визначити, чи зростає попит, падає він, чи ціни мають тенденцію підвищуватися або знижуватися в найближчі тижні.

На основі цих даних функція create_features формує повний набір із приблизно двадцяти ознак — від календарних характеристик (день тижня, місяць, дата) до статистики подібних бронювань, відстані до найближчих реальних бронювань, метрик попиту, рівня волатильності та оцінки статусу ціни (рис 4.12). Також розраховується оптимальна ціна та можливі втрати доходу, якщо фактична ціна нижча за рекомендовану. Отримані ознаки використовуються для навчання моделей машинного навчання. Завершальний етап виконується у функції train_models, яка завантажує тарифні та історичні дані, генерує ознаки, нормалізує їх за допомогою StandardScaler та розділяє вибірку на тренувальну й тестову. Далі система навчає дві окремі моделі: RandomForestClassifier для визначення статусу ціни (низька, нормальна або завищена) та RandomForestRegressor для прогнозування ймовірної реальної вартості на вибрану дату. Після навчання моделі та скалер зберігаються у файлову систему для подальшого використання під час API-запитів.

```
feature_cols = [
    'day_of_week', 'month', 'day_of_month', 'price_tarif',
    'avg_real_price', 'min_real_price', 'max_real_price',
    'std_real_price', 'count_similar', 'min_date_diff', 'has_real_price',
    'occupancy_rate', 'total_booked_days', 'avg_booking_duration',
    'bookings_per_month', 'demand_volatility', 'demand_count',
    'demand_intensity', 'price_trend', 'recent_demand_ratio'
]

X = features_df[feature_cols].fillna(0)
y_classification = features_df['price_status']
y_regression = features_df['real_price']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train_cls, y_test_cls, y_train_reg, y_test_reg = train_test_split(
    X_scaled, y_classification, y_regression, test_size=0.2, random_state=42
)

print("Навчання моделі класифікації...")
clf_model = RandomForestClassifier(n_estimators=100, random_state=42, max_depth=10)
clf_model.fit(X_train, y_train_cls)

print("Навчання моделі регресії...")
reg_model = RandomForestRegressor(n_estimators=100, random_state=42, max_depth=10)
reg_model.fit(X_train, y_train_reg)

joblib.dump(clf_model, PRICE_NORMALITY_MODEL)
joblib.dump(reg_model, PRICE_PREDICTION_MODEL)
joblib.dump(scaler, SCALER_PATH)

clf_score = clf_model.score(X_test, y_test_cls)
reg_score = reg_model.score(X_test, y_test_reg)
```

Рисунок 4.12 – Функція create_features

4.4 Функціональність вебсайту Royal Apart

Вебсайт Royal Apart є центральним клієнтським компонентом усієї системи, оскільки саме через нього більшість користувачів здійснюють пошук апартаментів,

перегляд доступності та оформлення бронювання. Інтерфейс створено як багатосторінковий застосунок, на основі React, Vite та Tailwind CSS, що забезпечує високу швидкість роботи та миттєве оновлення даних без перезавантаження сторінки (рис 4.13).

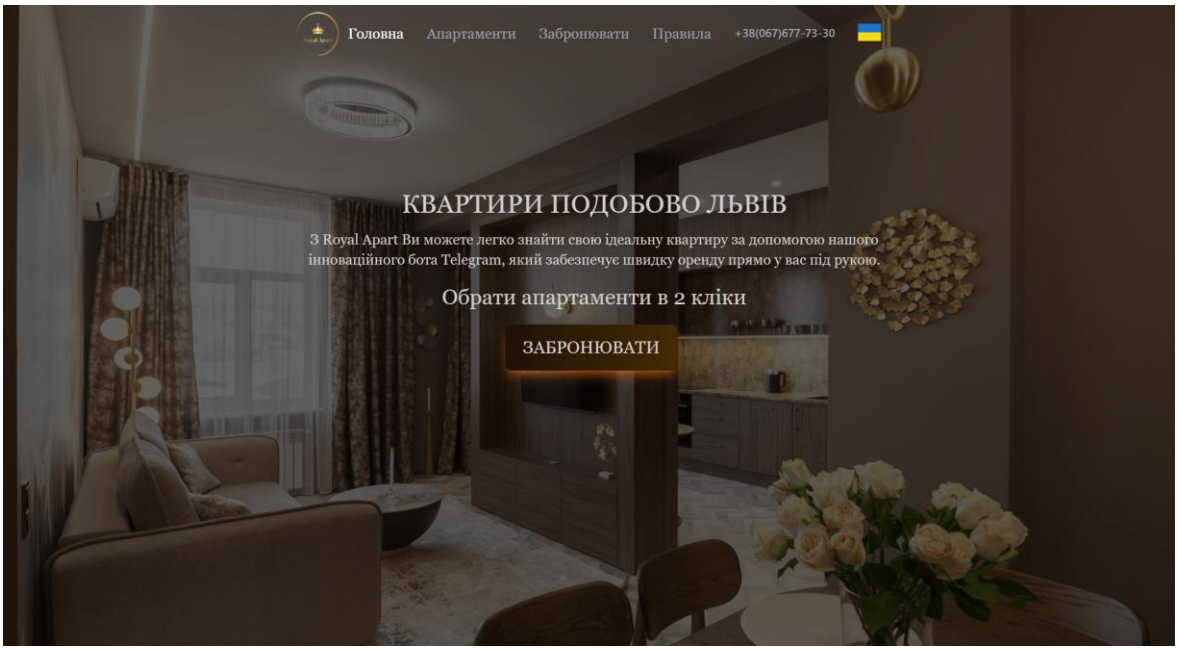


Рисунок 4.13 – Головна сторінка сайту Royal Apart

Однією з ключових можливостей вебсайту є динамічний каталог апартаментів (рис. 4.14), який формується на основі даних із MongoDB.

Під час завантаження сторінки сайт надсилає запит до API-сервера, отримує повний список апартаментів разом із характеристиками, фотографіями та прив’язкою до WuBook через глобальні ідентифікатори (globalId). Завдяки цьому будь-які зміни, зроблені адміністратором у панелі керування, миттєво відображаються на сайті.

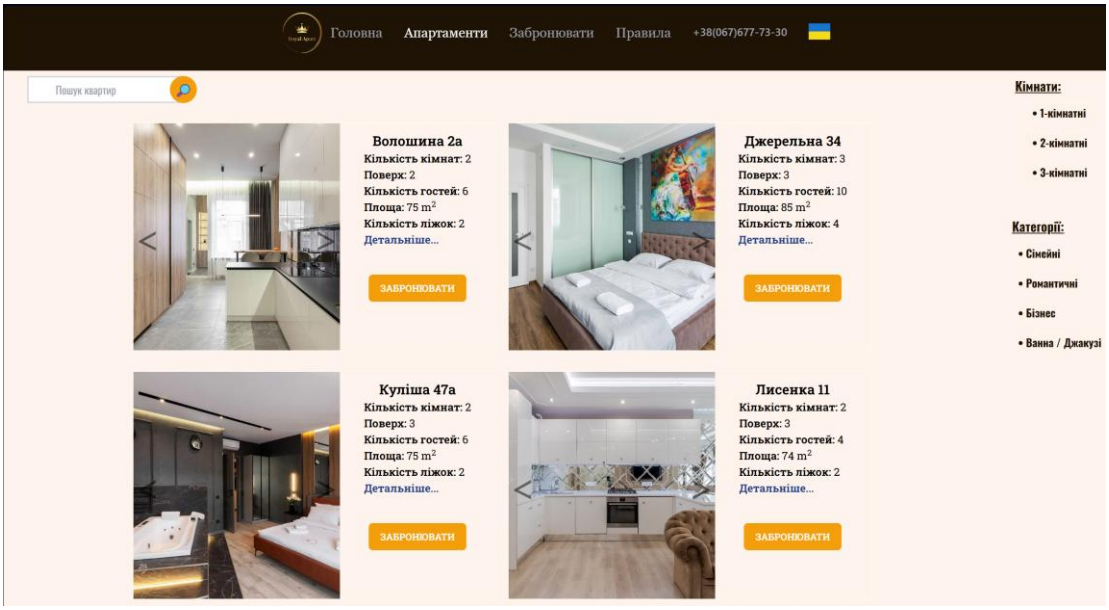


Рисунок 4.14 – Каталог апартаментів на сайті

На вебсайті Royal Apart використовується готовий модуль бронювання WuBook, який інтегрується безпосередньо через офіційний віджет та API сервіс (рис. 4.15).

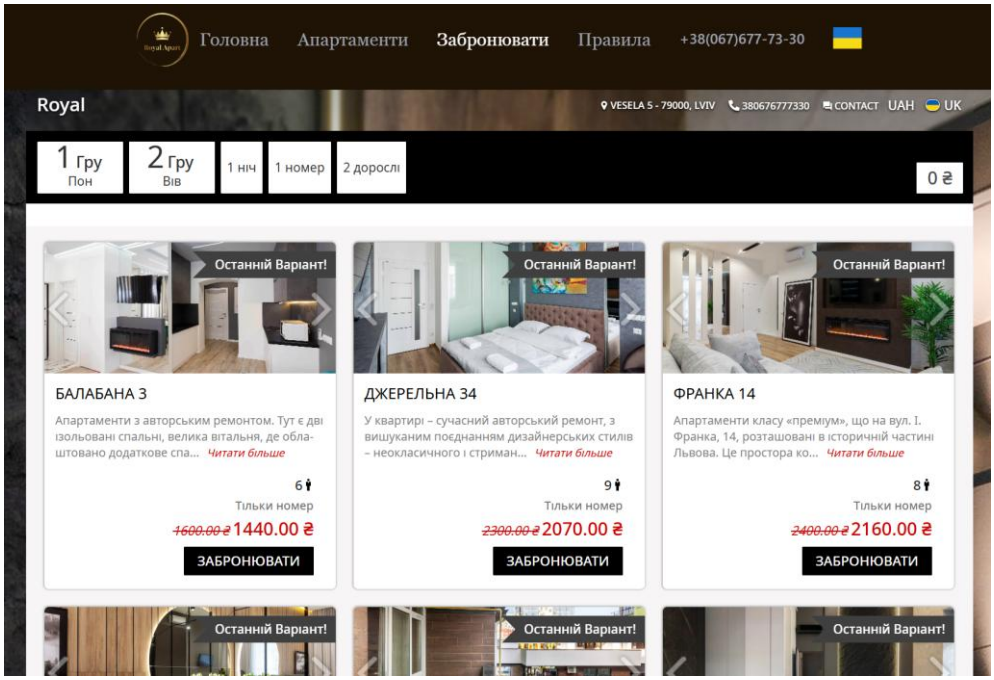


Рисунок 4.15 – Модуль бронювання Wubook

Завдяки цьому мені не обов’язково розробляти власну систему обробки резервацій, а використовувати повнофункціональний механізм WuBook з автоматичною перевіркою доступності, формуванням бронювань, синхронізацією

календаря та обробкою тарифів. Готовий модуль забезпечує надійну роботу, повну сумісність із зовнішніми каналами продажів і мінімізує ризик технічних помилок, оскільки всі операції бронювання виконуються на стороні WuBook, а сайт лише взаємодіє з ним через офіційні інтерфейси.

4.5 Чат бот з власним модулем бронювання

Telegram-бот Royal Apart виконує роль швидкого та зручного інструмента для взаємодії з клієнтами, які шукають житло та хочуть оперативно оформити бронювання без переходу на сайт (рис. 4.16). Він реалізований на Node.js і інтегрований із серверною частиною через REST API, що забезпечує синхронний доступ до бази апартаментів, актуальних цін і календаря зайнятості. Завдяки інтерактивній архітектурі бот працює як повноцінний користувацький інтерфейс, який супроводжує гостя на кожному кроці від пошуку до подання заявки на бронювання [4].

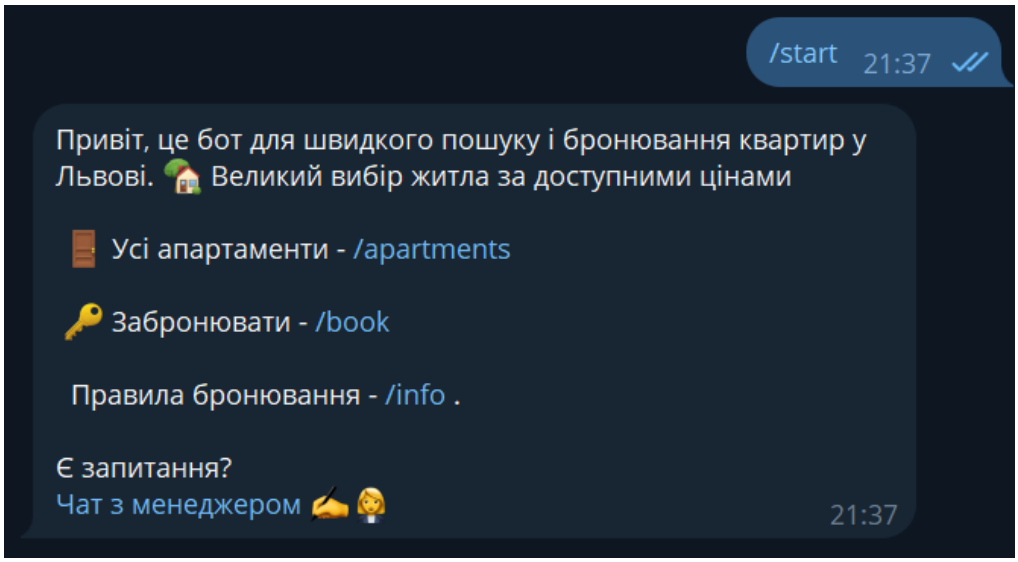


Рисунок 4.16 – Запуск чат-бота

Найважливішим елементом бота є модуль бронювання, який працює за принципом покрокового сценарію. Спочатку користувач обирає дату заїзду та дату виїзду: для цього бот формує інтерактивний календар із кнопками для кожного дня місяця. Після вибору дат бот надсилає запит на сервер, де відбувається перевірка доступності через WuBook API. Лише ті апартаменти, які справді вільні в заданий

період, показуються користувачу (рис. 4.17). Такий механізм гарантує точність і запобігає ситуаціям, коли клієнт намагається забронювати вже зайнятий об’єкт [5].

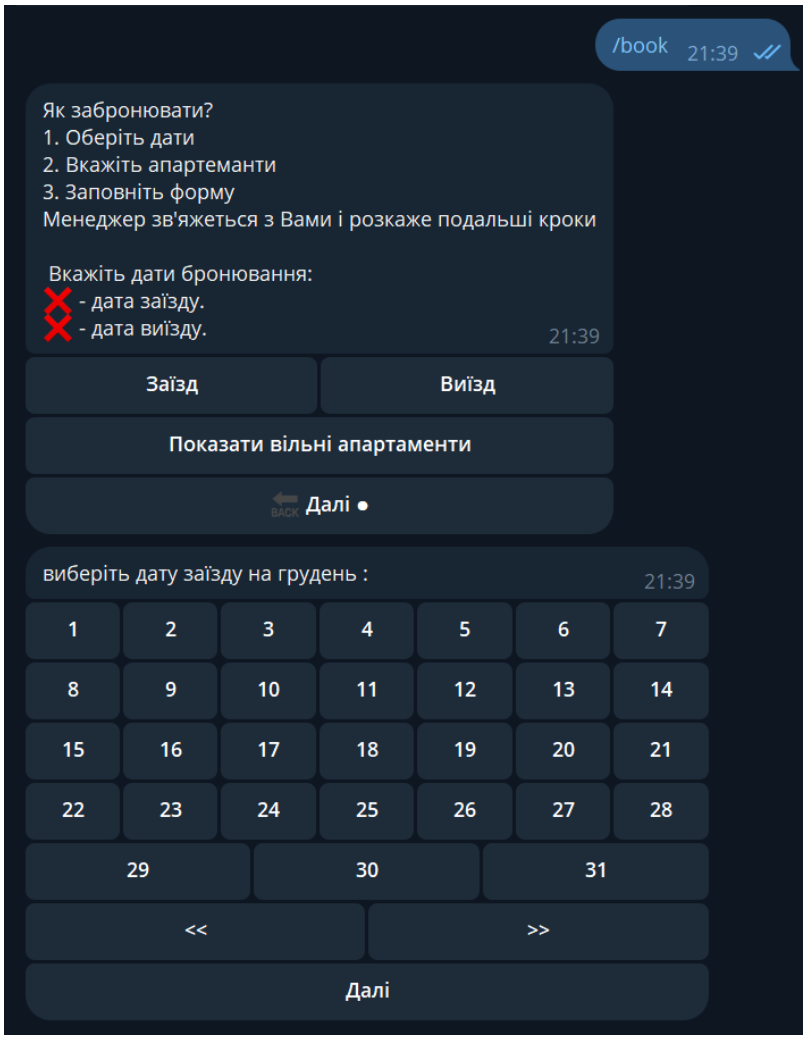


Рисунок 4.17 – Вибір дат бронювання

Фінальним кроком є заповнення форми, де користувач вводить ім’я, номер телефону та додаткові побажання. Після цього заявка надсилається адміністратору, а менеджер зв’язується з гостем для підтвердження. Хоча безпосереднє створення бронювання в WuBook не відбувається через обмеження інтеграції, бот працює як ефективний інструмент попередніх заявок, повністю автоматизуючи процес збору інформації та перевірки наявності житла (рис. 4.18).

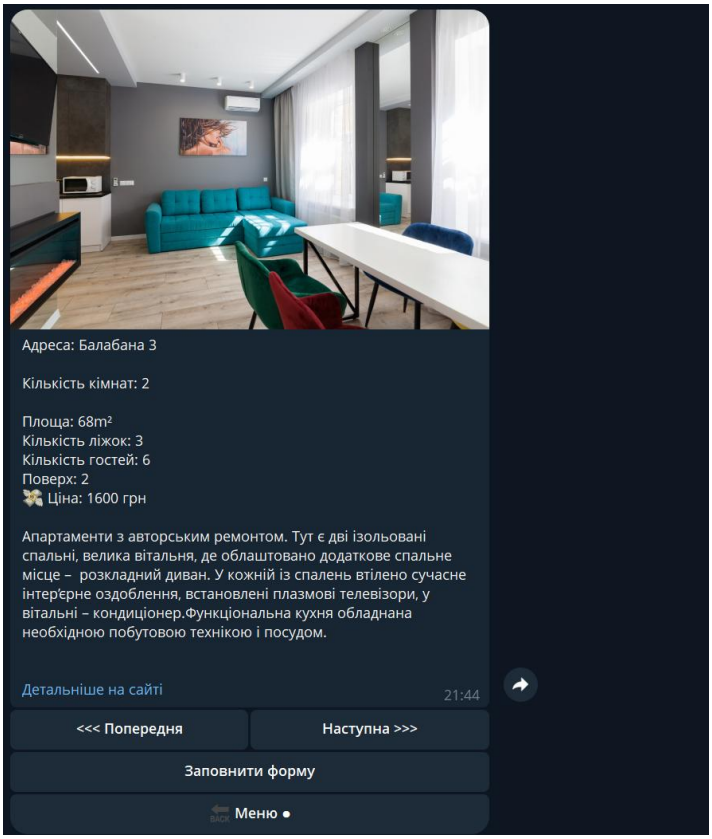


Рисунок 4.18 – Актуальні апартаменти з синхронізованою ціною

Чат-бот значно розширює канали взаємодії з клієнтами, забезпечуючи швидкий доступ до інформації, зручність використання, а також мобільність сервісу.

4.6 Адмін-панель керування та аналітичний модуль

Адмін-панель є центральним інструментом для менеджерів та власників системи, які здійснюють керування всіма об'єктами нерухомості, оновлюють інформацію на сайті та проводять аналітичні дослідження попиту. Панель реалізована у вигляді окремого React-застосунку з використанням Tailwind CSS, що забезпечує швидку роботу інтерфейсу, адаптивність та можливість швидко взаємодіяти з серверною частиною.

Після авторизації адміністратор отримує доступ до головної сторінки, де відображається повний перелік апартаментів у вигляді простих карток. Кожна картка містить фотографію житла, базові характеристики (ціна, площа, кількість кімнат, гостей, поверх), а також два основні інструменти — кнопку редагування та кнопку видалення (рис. 4.19).

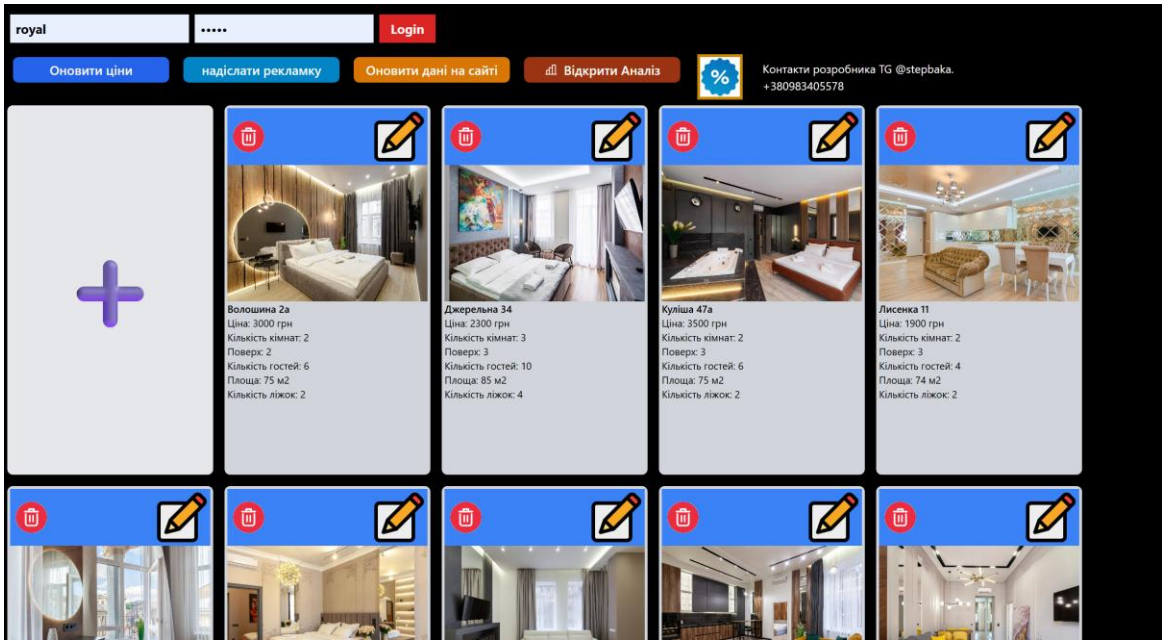


Рисунок 4.19 – Головна сторінка адмін-панелі керування

Це дозволяє швидко оновлювати інформацію або вилучати апартаменти з каталогу. Додатково присутній окремий елемент із кнопкою «+», що відкриває форму для створення нового запису, включно із завантаженням фотографій, описом та технічними параметрами квартири.

Перший ключовий компонент аналітичного модуля відповідає за відображення актуальних, попередніх та майбутніх тарифів на апартаменти, отриманих безпосередньо з сервісу WuBook. Цей інструмент призначений для менеджерів та адміністраторів, які мають контролювати динаміку зміни цін.

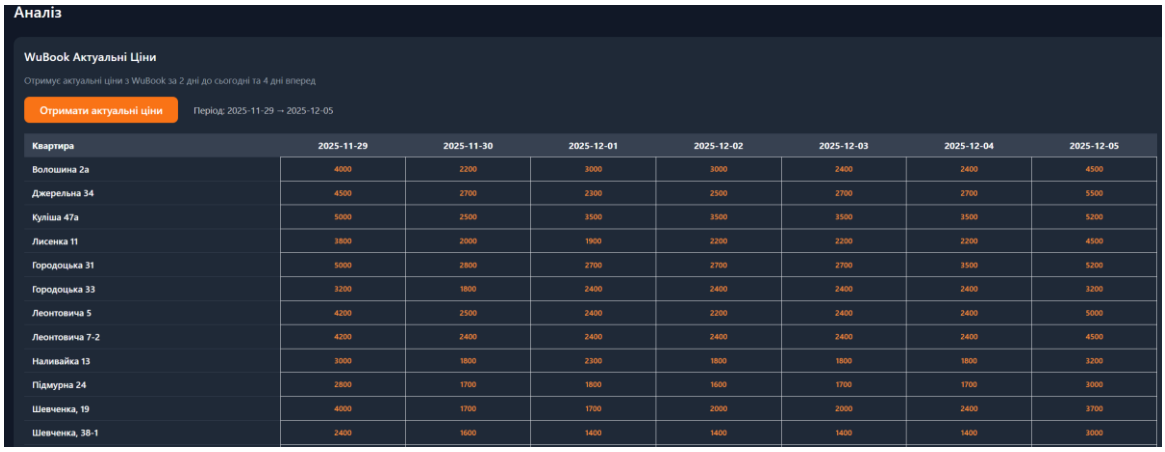


Рисунок 4.20 – Блок адмін-панелі керування з актуальними цінами

Другий компонент аналітичного розділу є найінформативнішим та найбільш технічно насиченим елементом системи. Він відповідає за повний аналіз тарифів на сьогодні, порівнюючи поточні ціни з WuBook із передбаченими значеннями моделі машинного навчання, а також з даними історичних бронювань. В результаті менеджер отримує не просто таблицю цін, а повноцінну систему підтримки прийняття рішень, яка пояснює, чому саме така ціна є коректною або потребує корекції.

У верхній частині блоку відображаються загальні показники: дата аналізу, кількість квартир, кількість успішно оброблених позицій та кількість помилок. Це дає змогу оцінити цілісність даних та якість синхронізації з мого сервера та WuBook.

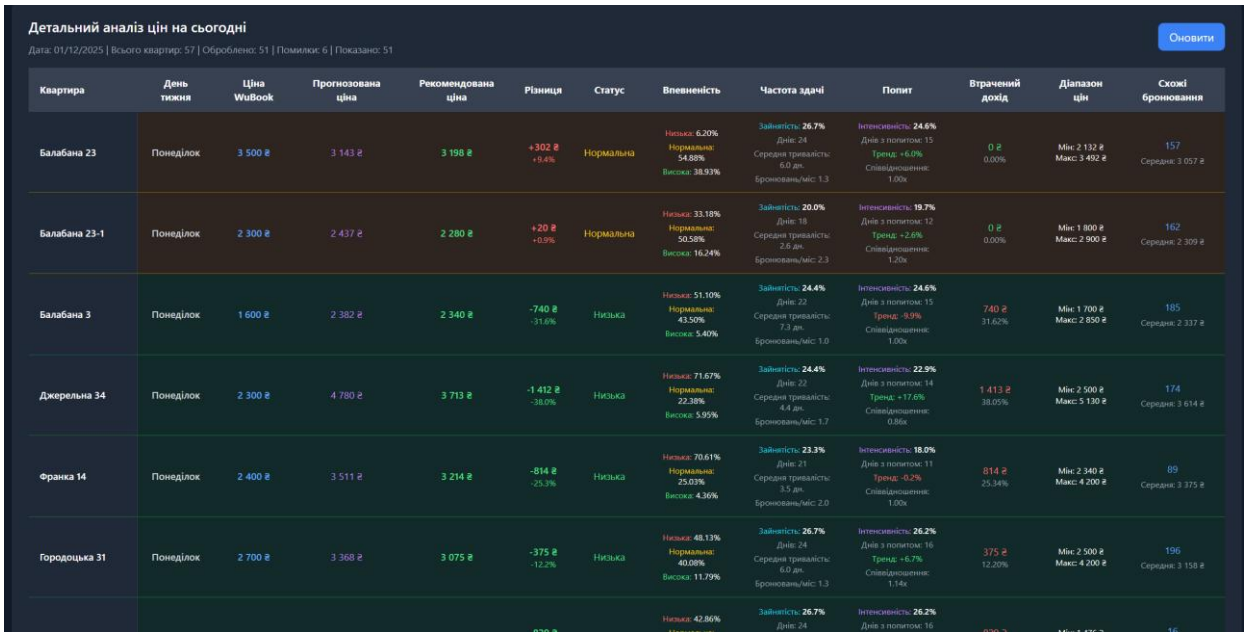


Рисунок 4.21 – Блок адмін-панелі керування з аналізом цін

Кожен рядок таблиці містить детальний опис однієї конкретної квартири, і ціни на неї. Спочатку подаються базові значення: день тижня, актуальна ціна WuBook, прогнозована ціна та рекомендована ціна. Далі система автоматично обчислює різницю між поточним тарифом і передбаченою вартістю, що дозволяє визначити статус ціни (нормальна, завищена або занижена). Окремо показаний рівень впевненості моделі, що базується на ймовірностях трьох сценаріїв: низька, нормальна або висока ціна.

Праворуч розташований блок з аналітичними метриками, сформованими на основі історичних бронювань. Тут відображаються ключові показники попиту та зайнятості: коефіцієнт зайнятості, середня тривалість бронювання, кількість бронювань на місяць та інтенсивність попиту навколо поточної дати. Також виводиться показник втраченого доходу, якщо фактична ціна нижча за оптимальну, та діапазон реальних цін на схожі бронювання у минулому.

Особливу цінність має індикатор «Схожі бронювання» — кількісне та середнє значення історичних бронювань, близьких до поточної дати та з подібними характеристиками. Завдяки цьому менеджер може швидко оцінити, наскільки моделлю враховано реальний попит.

Висновки до розділу

Архітектура моєї програмної системи «Royal Apart» демонструє комплексний та продуманий підхід до організації платформи керування бронюваннями. Використання мікросервісної структури, контейнеризації Docker та чіткої багаторівневої структури дозволило створити гнучку, масштабовану та зручну інфраструктуру. Кожен компонент — вебсайт, адмін-панель, чат-бот, серверна логіка, модуль машинного навчання та рівень даних — виконує власну ізольовану роль, але водночас ефективно взаємодіє через REST API.

Програмна модель повністю відповідає потребам сучасного сервісу бронювання та забезпечує високу якість користувацького досвіду.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис концепції та ціннісної пропозиції продукту

Ідея стартап-проєкту «Royal Apart» полягає у створенні комплексної цифрової системи для управління апартаментами у сфері добової оренди, яка поєднує можливості онлайн бронювання на різних платформах, аналітики, автоматизації бізнес-процесів та машинного навчання. На відміну від традиційних систем бронювання, які переважно зосереджуються лише на обробці запитів про бронювання, запропонований проєкт поєднує сайт, чат-бот, серверну частину, інструменти для динамічного ціноутворення, а також аналітичний модуль з прогнозування ціни. Така інтегрована система дозволяє власнику нерухомості не лише управляти бронюваннями, але й оптимізувати доходи, покращувати завантаженість апартаментів, контролювати динаміку цін і знижувати ризики малої заповненості житла.

Зміст ідеї полягає у забезпеченні єдиного простору, в якому клієнти можуть швидко знайти доступні апартаменти, переглянути характеристики та забронювати їх у зручний спосіб, тоді як адміністратор отримує можливість централізованого керування об’єктами, актуальними тарифами та історією бронювань. Особливе місце в системі займає чат-бот із власним модулем бронювання, який дозволяє клієнтам взаємодіяти із сервісом у максимально простій та знайомій формі, не переходячи на сторонні платформи. А аналітичний модуль є унікальним і основаним на справжніх даних. Саме ця функція стає однією з ключових переваг проєкту та його конкурентною відмінністю, оскільки більшість конкурентів працюють за застарілими методами бронювання.

Проєкт має широкі напрями застосування. Система може використовуватися апарт-готелями, малими готельними комплексами, окремими власниками нерухомості, компаніями подобової оренди, а також сервісами, що здійснюють посередницькі функції на ринку житла. Користувач отримує низку важливих переваг, серед яких оперативний доступ до інформації про апартаменти, можливість

здійснювати бронювання самостійно та без участі менеджера, а також прозорість умов оренди. Для власника нерухомості система забезпечує автоматизацію рутинних процесів, економію часу, зниження кількості помилок, а також можливість контролювати дохідність та отримувати дані для прийняття стратегічних рішень. Не менш важливим є зменшення кількості людей і часу для опрацювання усіх заявок.

5.2 Аналіз технологічних можливостей реалізації ідей проекту

Технологічна реалізація стартапу базується на сучасному стеку технологій, які забезпечують масштабованість, стабільність та зручність подальшого розвитку проєкту. Оскільки система вже частково функціонує, всі обрані технології підтвердили свою ефективність на практиці. Основою для клієнтської частини слугують React та Tailwind CSS, що дозволило створити швидкий, адаптивний та гнучкий інтерфейс. Адмін-панель також побудована на React і дає змогу оперативно керувати всіма об’єктами нерухомості, оновлювати інформацію, додавати нові апартаменти та переглядати статистику.

Серверна частина розроблена на Node.js з використанням Express.js, що забезпечує стабільну REST API взаємодію між усіма клієнтськими компонентами системи. Технологічність рішення полягає в тому, що сервер виконує роль центру обробки запитів, інтегрується з базою даних MongoDB Atlas, викликає модулі машинного навчання та взаємодіє з зовнішньою системою бронювання WuBook. Ця архітектура дозволила забезпечити високу ефективність та розподілити навантаження між різними сервісами.

Модуль машинного навчання реалізовано на Python із використанням бібліотек pandas, NumPy, scikit-learn та joblib. Модель навчається на основі історичних тарифів і реальних цін бронювання та виконує як класифікацію (оцінка статусу ціни), так і регресію (передбачення рекомендованої вартості). Технології машинного навчання є абсолютно доступними розробнику, і їх впровадження не потребувало створення нових алгоритмів, а лише адаптації та налаштування існуючих під специфіку даних.

Для забезпечення гнучкості та легкості розгортання вся система поміщена у контейнери за допомогою технології Docker. Це робить можливим швидке масштабування у разі потреби, а також спрощує підтримку й перенесення системи на інші сервери. Технології, використані в проєкті, є доступними, широко застосовуються на ринку та не потребують додаткової розробки, що свідчить про повну технологічну реалізованість ідеї та готовність проєкту до розширення.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Ринок подобової оренди в Україні продовжує активно розвиватися, незважаючи на вплив зовнішніх факторів. Через війну багато великих міст стали притулком для великої кількості людей. Також міста стають пунктами тимчасового перебування між подорожами. Особливо інтенсивне зростання демонструють великі міста, такі як Львів, Київ, Одеса, де існує значний потік туристів, бізнес-подорожей і короткострокових відряджень. Короткострокова оренда дуже важлива. Саме тому ринок створює сприятливі умови для впровадження автоматизованих систем бронювання та управління апартаментами, оскільки попит на швидке бронювання та якісний сервіс постійно зростає. Модульність системи дозволяє використовувати її і за межами однієї компанії.

Потенційні групи клієнтів проєкту включають власників індивідуальних апартаментів, апарт-готелі, малі готельні мережі, агенції нерухомості, а також менеджерів, які працюють із кількома об’єктами та потребують централізованої функціональності. Кожна з цих груп має власні вимоги до системи, однак усі вони є зацікавленими у швидкому доступі до інформації, точних тарифах та можливості автоматизувати частину процесів.

Ринкове середовище характеризується помірною конкуренцією, де значна частина рішень представлена великими міжнародними сервісами на кшталт Booking чи Airbnb. Проте їх головний недолік полягає в тому, що вони не забезпечують комплексного управління апартаментами, а орієнтуються виключно на клієнтоорієнтованість для гостей. Локальні сервіси часто не мають аналітичних

модулів чи функцій прогнозування цін, що створює сприятливу нішу для впровадження інноваційного рішення. Навіть найсильніші гравці на львівському ринку оренди апартаментів “Інші Апартментс”, мають сталі ціни, які рідко змінюються, і через це вони мають меншу заповненість.

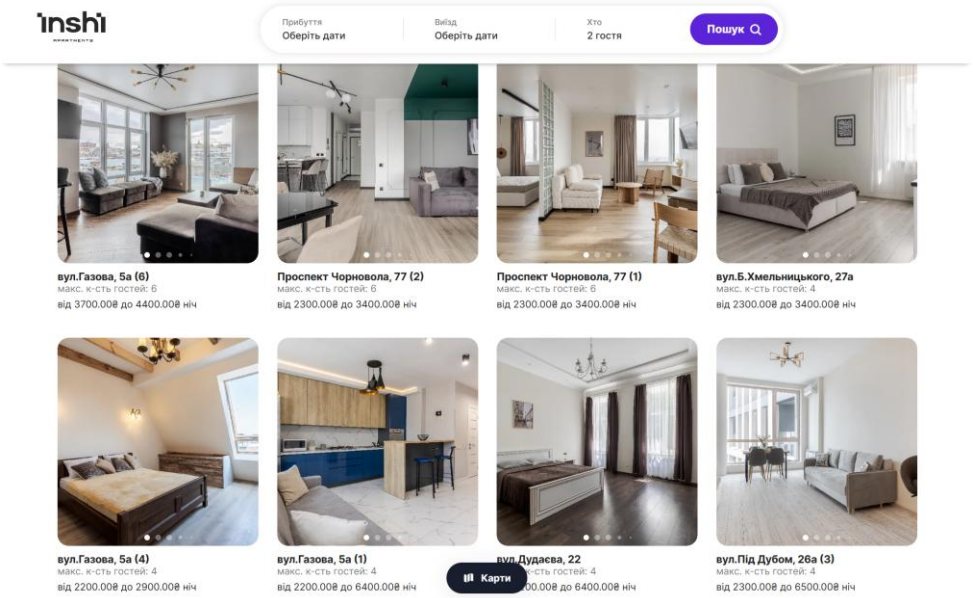


Рисунок 5.1 – Приклад сайту і цін конкурента

Серед сильних сторін «Royal Apart» варто виділити інтелектуальну аналітику, інтеграцію з чат-ботом, повну автономність роботи, а також можливість гнучкої адаптації під потреби конкретного об’єкта нерухомості. Слабкою стороною на початковому етапі є необхідність подальшого масштабування системи у випадку різкого зростання кількості об’єктів та клієнтів. Також серед недоліків це розширення і адаптування функціоналу для якісного виконання поставлених рішень. Водночас динаміка ринку вказує на високу ймовірність комерційного успіху, оскільки попит на цифрові рішення в туристичному секторі продовжує зростати.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії проєкту ґрунтується на детальному аналізі сегментації цільової аудиторії, що є ключовим фактором для максимального охоплення різних груп користувачів. Апартаменти «Royal Apart» поділяються на три

окремі категорії — романтичні, бізнес та сімейні. Такий поділ дозволяє більш точно задовольняти потреби різних сегментів ринку та створювати персоналізований користувацький досвід, що позитивно впливає на конкурентоспроможність продукту.

Стратегія диференційованого маркетингу є найбільш ефективною для цього проєкту, оскільки кожна категорія апартаментів має власну аудиторію та унікальні очікування. Романтичні апартаменти орієнтовані на пари та гостей, які шукають атмосферу затишку та приватності. Бізнес-категорія підходить для ділових мандрівників, яким важливі комфорт, наявність робочого простору та оптимальне розташування. Сімейні апартаменти розраховані на гостей з дітьми або великими групами, що вимагають більше простору, побутових зручностей і гнучких умов проживання. Така сегментація не лише полегшує просування, а й впливає на формування ціни, позиціонування, дизайн інтерфейсу та навіть структуру аналітичного модуля. чат-бот у цьому контексті є універсальним рішенням для швидкого підбору апартаментів за категорією та датами, особливо для романтичних і бізнес-клієнтів, які цінують простоту взаємодії. Адмін-панель відіграє важливу роль для управляючих компаній та менеджерів, які контролюють великі сімейні об’єкти та потребують детальної статистики, гнучкого керування цінами та аналізу завантаженості кожного сегмента.

Позиціонування системи ґрунтується на її багатофункціональності та інтелектуальності. «Royal Apart» не позиціонується лише як сервіс бронювання, а подається як інтегрована платформа управління нерухомістю, що здатна глибоко аналізувати поведінку ринку та формувати оптимальні тарифи з урахуванням категорії апартаментів. Окрему конкурентну перевагу формує модуль машинного навчання, який генерує окремі рекомендації цін для кожної квартири. Таким чином, ринкова стратегія проєкту враховує різноманітність цільових сегментів і забезпечує адаптивність продукту до потреб кожного з них.

Висновки до розділу

Проаналізовано концепцію продукту, визначено його унікальну цінність та ринкові перспективи, що базуються на поєднанні автоматизації процесів, зручності для користувачів та впровадженні інтелектуальних алгоритмів ціноутворення.

Система вирізняється тим, що охоплює одразу декілька ключових каналів взаємодії з клієнтами — вебсайт із вбудованим модулем бронювання WuBook, чат-бот із власним механізмом попереднього бронювання та повноцінну адміністративну панель. Такий підхід формує цілісну екосистему, здатну працювати як для невеликих приватних власників, так і для апарт-готелів чи агенцій нерухомості. Розподіл житла на три категорії — романтичні, бізнес та сімейні — дозволяє адаптувати систему під різні стилі використання та сегменти ринку.

Аналіз технологічних можливостей засвідчив, що всі необхідні компоненти вже реалізовані та функціонують у виробничому режимі. Платформа побудована на сучасному технологічному стеку з використанням Node.js, React, Docker, MongoDB та Python-модулів машинного навчання. Це підтверджує не лише технічну здійсненність проєкту, а й його готовність до масштабування та розширення. Особливу цінність становить ML-модуль, що формує прогнози цін для кожної категорії апартаментів, оцінює попит, зайнятість і динаміку ринку, створюючи реальні передумови для підвищення прибутковості.

У межах ринкового аналізу виявлено стійкий попит на цифрові продукти у сфері короткострокової оренди, а також недосконаліть існуючих інструментів управління. Більшість конкурентних рішень не пропонують комплексної аналітики чи гнучкої взаємодії через різні платформи. Це відкриває додаткові можливості для розвитку проєкту, оскільки «Royal Apart» поєднує функції бронювання, CRM-компоненти, машинне навчання та зручний інтерфейс для користувачів.

ВИСНОВКИ

У магістерській дипломній роботі я виконав повний цикл розроблення інтелектуальної багатокомпонентної системи управління апартаментами «Royal Apart», що об’єднує модуль бронювання, вебсайт, чат-бот, аналітичний інструментарій і сервіс машинного навчання. У ході роботи проведено аналіз сучасних тенденцій ринку подобової оренди, інструментів цифрової автоматизації та підходів до динамічного ціноутворення, що дозволило сформувати вимоги до функціональності, надійності та архітектури майбутнього програмного комплексу.

На етапі математичного та інформаційного забезпечення розроблено алгоритмічну систему прогнозування вартості проживання, що включає підготовку датасету, створення ознак, аналіз попиту та зайнятості, пошук релевантних історичних бронювань і формування рекомендацій щодо оптимальної ціни. Система базується на моделях RandomForestRegressor та RandomForestClassifier, доповнених механізмами нормалізації, оцінювання якості та обробки аномалій у даних. Проведені експерименти засвідчили стабільність роботи моделі та здатність точно прогнозувати статус і рівень ціни на основі комплексних історичних характеристик.

Програмна реалізація охопила створення мікросервісного середовища, що включає Node.js/Express сервер, Python-сервіс машинного навчання, хмарну базу даних MongoDB, модуль інтеграції з WuBook, механізми автоматичного оновлення тарифів та систему керування медіафайлами. Розроблено повний клієнтський стек — вебсайт на React із готовим модулем бронювання, чат-бот із власним сценарієм підбору апартаментів і попереднього бронювання, а також адміністративну панель із розширеною аналітикою попиту та динамічного ціноутворення.

Результати тестування підтвердили функціональну цілісність системи, її стабільність та здатність до подальшого масштабування. Проєкт продемонстрував практичну цінність завдяки автоматизації ключових бізнес-процесів, що охоплюють управління цінами, аналіз ринку, роботу з клієнтами та синхронізацію апартаментів із зовнішніми сервісами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. K. Chodorow (2020). MongoDB: The Definitive Guide. 516 p.
2. Ethan Brown (2019). Web Development with Node and Express: Leveraging the JavaScript Stack. 352 p.
3. Morgan, M. (2020). Node.js Design Patterns (Third Edition). 520 p.
4. Valerio G., Cantelon M., Harter D., Holowaychuk T. (2017). Node.js in Action (Second Edition). 330 p.
5. Alex Banks, Eve Porcello (2020). Learning React. 350 p.
6. Mongoose ODM Documentation. URL: https://mongoosejs.com/ (дата звернення: 18.03.2025).
7. Aurélien Géron (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 856 p.
8. Trevor Hastie, Robert Tibshirani, Jerome Friedman (2017). The Elements of Statistical Learning. 745 p.
9. Hyndman R., Athanasopoulos G. (2021). Forecasting: Principles and Practice. 398 p.
10. Fielding R., Taylor R. (2000). Principled Design of Modern Web Architecture. 78 p.
11. O'Reilly Media (2021). Designing Data Applications. Martin Kleppmann. 616 p.
12. Express.js documentation. URL: https://expressjs.com/ (дата звернення: 12.03.2025).
13. React documentation. URL: https://react.dev/ (дата звернення: 05.03.2025).
14. WuBook API Documentation. URL: https://wubook.net/wired/ (дата звернення: 10.03.2025).
15. MongoDB Atlas Documentation. URL: https://www.mongodb.com/docs/atlas/ (дата звернення: 15.03.2025).
16. Scikit-learn documentation. URL: https://scikit-learn.org/stable/ (дата звернення: 18.03.2025).

17. Pandas documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 18.03.2025).
18. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 20.03.2025).
19. Nginx Documentation. URL: https://nginx.org/en/docs/ (дата звернення: 02.04.2025).
20. ISO/IEC 25010:2011 Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE). 34 p.

ДОДАТКИ

ДОДАТОК А

app.py

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import joblib
import os
import sys
import re
from datetime import datetime, timedelta
import json
if sys.platform == 'win32':
    import io
    try:
        if hasattr(sys.stdout, 'buffer'):
            sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8', errors='replace')
        if hasattr(sys.stderr, 'buffer'):
            sys.stderr = io.TextIOWrapper(sys.stderr.buffer, encoding='utf-8', errors='replace')
        os.environ['PYTHONIOENCODING'] = 'utf-8'
    except:
        pass
SCRIPT_DIR = os.path.dirname(os.path.abspath(__file__))

def find_data_path(filename):

    local_path = os.path.join(SCRIPT_DIR, "..", "server", "data2025", filename)
    if os.path.exists(local_path):
        return local_path

    docker_path = os.path.join("/app", "server", "data2025", filename)
    if os.path.exists(docker_path):
        return docker_path
```

```
    return local_path

REAL_CSV = find_data_path("realPrice.csv")
TARIF_CSV = find_data_path("tarifPrice.csv")
REAL_DAILY_CACHE = find_data_path("realPrice_daily.csv")
MODEL_DIR = os.path.join(SCRIPT_DIR, "models")
PRICE_NORMALITY_MODEL = os.path.join(MODEL_DIR, "price_normality_model.pkl")
PRICE_PREDICTION_MODEL = os.path.join(MODEL_DIR, "price_prediction_model.pkl")
SCALER_PATH = os.path.join(MODEL_DIR, "scaler.pkl")
os.makedirs(MODEL_DIR, exist_ok=True)
def normalize_room_name(name):

    if not name or pd.isna(name):
        return ""
    normalized = str(name).strip()
    normalized = normalized.lower()
    normalized = re.sub(r'[/\-_.,;:]+', ' ', normalized)
    normalized = re.sub(r'\s+', ' ', normalized)
    normalized = normalized.strip()

    return normalized
def find_room_matches(normalized_room, real_df):

    if not normalized_room:
        return pd.DataFrame()

    real_df_normalized = real_df.copy()
    real_df_normalized['room_normalized'] = real_df_normalized['room'].apply(normalize_room_name)

    exact_matches = real_df_normalized[real_df_normalized['room_normalized'] == normalized_room]

    if len(exact_matches) > 0:
        return exact_matches.drop(columns=['room_normalized'])

    room_words = [w for w in normalized_room.split() if len(w) > 1]

    if not room_words:
        return pd.DataFrame()

    mask = pd.Series([True] * len(real_df_normalized))
```

```
    for word in room_words:
        mask = mask & real_df_normalized['room_normalized'].str.contains(word, case=False, na=False, regex=False)

    partial_matches = real_df_normalized[mask]

    if len(partial_matches) > 0:
        return partial_matches.drop(columns=['room_normalized'])

    if len(room_words) > 0:
        first_word = room_words[0]
        first_word_matches = real_df_normalized[
            real_df_normalized['room_normalized'].str.startswith(first_word, na=False)
        ]
        if len(first_word_matches) > 0:
            return first_word_matches.drop(columns=['room_normalized'])

    return pd.DataFrame()


def load_tarif_data():
    """Завантажує дані тарифів"""
    try:

        encodings = ['utf-8', 'utf-8-sig', 'cp1251', 'latin-1']
        df = None

        for encoding in encodings:
            try:

                df = pd.read_csv(TARIF_CSV, encoding=encoding)
                if 'price' in df.columns or 'Price' in df.columns:
                    price_col = 'price' if 'price' in df.columns else 'Price'
                    date_col = 'date' if 'date' in df.columns else 'Date'
                    room_col = 'roomName' if 'roomName' in df.columns else 'room'

                    df = df.rename(columns={
                        price_col: 'price_tarif',
                        date_col: 'date',
                        room_col: 'room'
                    })
```

```
                    break
                elif len(df.columns) == 4:

                    df = pd.read_csv(TARIF_CSV, header=None, names=["room_id", "room", "date", "price_tarif"], encoding=encoding)
                    break
                except (UnicodeDecodeError, pd.errors.EmptyDataError):
                    continue

            if df is None or len(df) == 0:

                df = pd.read_csv(TARIF_CSV, header=None, names=["room_id", "room", "date", "price_tarif"], encoding='latin-1')

            df["room"] = df["room"].astype(str).str.strip()
            df["date"] = pd.to_datetime(df["date"], dayfirst=True, errors='coerce')
            df["price_tarif"] = pd.to_numeric(df["price_tarif"], errors='coerce')

            df = df.dropna(subset=["date"])
            df = df[df["price_tarif"].notna() & (df["price_tarif"] > 0)]

            return df
        except Exception as e:
            print(f"Помилка завантаження тарифів: {e}")
            import traceback
            traceback.print_exc()
            return pd.DataFrame()
```

predict_price.py

```
import joblib
import pandas as pd
import sys
import json
import os

if len(sys.argv) < 3:
    print("Usage: python predict_price.py <room_name> <input_price> [<date>]")
    sys.exit(1)

room_name = sys.argv[1]
```

```
input_price = float(sys.argv[2])
date_input = sys.argv[3] if len(sys.argv) > 3 else None

script_dir = os.path.dirname(os.path.abspath(__file__))

model_path = os.path.join(script_dir, 'models', 'price_model.pkl')
model = joblib.load(model_path)

   csv_path = os.path.join(script_dir, '..', 'server', 'data2025', 'tarifPrice.csv')
   if not os.path.exists(csv_path):
      csv_path = os.path.join('/app', 'server', 'data2025', 'tarifPrice.csv')
tarif_df = pd.read_csv(csv_path, header=None,
                  names=["room_id","room","date","price_tarif"])
tarif_df["room"] = tarif_df["room"].astype(str).str.strip()
tarif_df["date"] = pd.to_datetime(tarif_df["date"], dayfirst=True, errors='coerce')
tarif_df = tarif_df.dropna(subset=["date","price_tarif"])

date = pd.to_datetime(date_input)
tarif_row = tarif_df[(tarif_df["room"] == room_name) & (tarif_df["date"] == date)]
if len(tarif_row) == 0:
   tarif_price_input = tarif_df[tarif_df["room"]==room_name]["price_tarif"].mean()
else:
   tarif_price_input = tarif_row.iloc[0]["price_tarif"]

day_of_week = date.dayofweek
X_new = pd.DataFrame([{"tarif_price": tarif_price_input, "day_of_week": day_of_week}])

predicted_price = model.predict(X_new)[0]
coefficient = input_price / predicted_price if predicted_price != 0 else None

print(json.dumps({
   "room": room_name,
   "input_price": input_price,
   "tarif_price": round(tarif_price_input,2),
   "predicted_price": round(predicted_price,2),
   "coefficient": round(coefficient,2) if coefficient else None
}))
```

ДОДАТОК Б

App.jsx

```
import React, { useEffect, useState } from "react";
import { BrowserRouter, Route, Routes, useLocation } from "react-router-dom";
import Header from "./components/header";
import Footer from "./components/footer";
import Book from "./pages/book/book";
import MainPagebody from "./components/mainpage/mainPagebody";
import Aparts from "./pages/aparts/aparts";
import Rules from "./pages/rules/rules";
import Contact from "./pages/contact";
import RoomPage from "./pages/roomPage/roomPage";
import MiniHotel from "./pages/mini-hotel/MiniHotel";
import NotFound from "./components/utils/NotFound";
import Loader from "./components/utils/loader";
import TermsAndConditions from "./pages/thermsAndConditions";

function AppContent() {
 const location = useLocation();
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  setLoading(true);
  const timer = setTimeout(() => {
   setLoading(false);
  }, 1500);
  return () => clearTimeout(timer);
 }, [location]);

 return (
  <>
   {loading ? (
    <Loader />
   ) : (
    <>
     <Header />
     <Routes>
      <Route path="/" element={<MainPagebody />} />
      <Route path="/uk" element={<MainPagebody />} />
      <Route path="/en" element={<MainPagebody />} />
```

```
            <Route path="/aparts" element={<Aparts />} />
            <Route path="/rules" element={<Rules />} />
            <Route path="/book" element={<Book />} />
            <Route path="/contact" element={<Contact />} />
            <Route path="/mini-hotel" element={<MiniHotel />} />
            <Route path="/room/:roomId" element={<RoomPage />} />
            <Route
              path="/terms-and-conditions"
              element={<TermsAndConditions />}
            />
            <Route path="*" element={<NotFound />} />
          </Routes>
          <Footer />
        </>
      )}
    </>
  );
}

function App() {
  return (
    <BrowserRouter>
      <AppContent />
    </BrowserRouter>
  );
}

export default App;
```

aparts.jsx

```
import React, { useState } from "react";
import RoomCard from "./roomCard";
import PickCategory from "./pickCategory";
import PickNumRoom from "./pickNumRoom";

function aparts() {
  const [selectedCategory, setSelectedCategory] = useState("");
  const [selectedNumRooms, setSelectedNumRooms] = useState([]);
  return (
    <div className="h-full w-screen text-black bg-back flex flex-col ">
      <div className="bg-shit2 h-24 "></div>
```

```
    <div className=" flex items-center justify-center">
      <div className="flex flex-col-reverse  lg:flex-row ">
        <RoomCard
          selectedCategory={selectedCategory}
          selectedNumRoom={selectedNumRooms}
        />
        <div className="flex flex-row  m-1  md:flex-row lg:flex-col font-oswald font-bold items-center justify-center lg:justify-start">
          <PickNumRoom
            selectedNumRooms={selectedNumRooms}
            setSelectedNumRooms={setSelectedNumRooms}
          />
          <PickCategory setSelectedCategory={setSelectedCategory} />
        </div>
      </div>
    </div>
  );
}

export default aparts;
```

analis.jsx

```
import React, { useEffect, useState } from "react";
import axios from "axios";

import WuBookPanel from "./wuBookPanel";

import ForecastChart from "./forecastChart";
import PredictionTable from "./predictionTable";

function Analis() {
  const [rooms, setRooms] = useState([]);
  const [averagePrice, setAveragePrice] = useState(0);
  const [predictedData, setPredictedData] = useState([]);
  const [isLoading, setIsLoading] = useState(false);

  const fetchRooms = async () => {
    setIsLoading(true);
```

```
    try {
      const res = await axios.get(
        "https://royalapart.online/api/siteRoyal/get-all-wodoo"
      );
      setRooms(res.data.data || []);
    } catch (e) {
      console.error("API Error", e);
    }
    setIsLoading(false);
  };

  useEffect(() => {
    fetchRooms();
  }, []);

  return (
    <div className="p-6 text-white bg-gray-900 min-h-screen">
      <h1 className="text-2xl font-semibold mb-6">Аналіз</h1>

      <WuBookPanel rooms={rooms} setRooms={setRooms} />



      <ForecastChart predictedData={predictedData} />

      <PredictionTable />
    </div>
  );
}

export default Analis;
```

WubookPanel.jsx

```
import React, { useState } from "react";
import axios from "axios";
import RoomsTable from "./roomsTable";

export default function WuBookPanel({ rooms, setRooms }) {
  const [loading, setLoading] = useState(false);
  const [dateRange, setDateRange] = useState({ from: null, to: null });
```

```
const fetchCurrentPrices = async () => {
  if (rooms.length === 0) {
    alert("Немає кімнат для обробки");
    return;
  }

  setLoading(true);
  try {
    console.log("📡 Запит актуальних цін з WuBook API...");

    const apiUrl = getApiUrl();
    const res = await axios.get(apiUrl);

    console.log("✅ Відповідь від сервера:", {
      success: res.data?.success,
      rows: res.data?.rows,
      pricesCount: res.data?.prices?.length,
      dateRange: res.data?.dateRange,
    });

    const prices = res.data?.prices || [];
    const range = res.data?.dateRange || {};

    if (prices.length === 0) {
      alert("Не вдалося отримати ціни з WuBook");
      setLoading(false);
      return;
    }

    const updatedRooms = rooms.map((room) => {
      const roomPrices = prices.filter(
        (p) =>
          String(p.roomId).trim() ===
          String(room.wdid || room.wubid || "").trim()
      );

      return {
        ...room,
        pricesCsv: roomPrices,
      };
```

```
    });

    const totalFound = updatedRooms.reduce(
      (sum, r) => sum + r.pricesCsv.length,
      0
    );

    console.log(
      `✅ Оброблено ${updatedRooms.length} кімнат, знайдено ${totalFound} записів`
    );

    setRooms(updatedRooms);
    setDateRange(range);

    if (range.from && range.to) {
      const [d1, m1, y1] = range.from.split("/");
      const [d2, m2, y2] = range.to.split("/");
      setDateRange({
        from: `${y1}-${m1.padStart(2, "0")}-${d1.padStart(2, "0")}`,
        to: `${y2}-${m2.padStart(2, "0")}-${d2.padStart(2, "0")}`,
      });
    }
  } catch (err) {
    console.error("❌ Помилка завантаження:", err.message);
    console.error("❌ Деталі помилки:", err.response?.data || err);
    alert(`Помилка: ${err.message}`);
  } finally {
    setLoading(false);
  }
};

return (
  <div className="bg-gray-800 p-4 rounded-xl mb-6">
    <h2 className="text-lg font-semibold mb-3">WuBook Актуальні Ціни</h2>
    <p className="text-sm text-gray-400 mb-4">
      Отримує актуальні ціни з WuBook за 2 дні до сьогодні та 4 дні вперед
    </p>

    <div className="flex gap-2 mb-4">
      <button
```

```
        onClick={fetchCurrentPrices}
        disabled={loading}
        className="bg-orange-500 hover:bg-orange-600 disabled:bg-gray-600 px-6 py-2 rounded-lg font-semibold"
      >
        {loading ? "Завантаження..." : "Отримати актуальні ціни"}
      </button>
      {dateRange.from && dateRange.to && (
        <div className="px-4 py-2 text-sm text-gray-300 flex items-center">
          Період: {dateRange.from} → {dateRange.to}
        </div>
      )}
    </div>

    {dateRange.from && dateRange.to && (
      <RoomsTable rooms={rooms} dfrom={dateRange.from} dto={dateRange.to} />
    )}
  </div>
 );
}
```

Sales.js

```
const express = require("express");
const router = express.Router();
const Sale = require("../models/sales");
router.use(express.json());

router.post("/", async (req, res) => {
 const { roomId, discount, tillDate } = req.body;
 let sale = {
  roomId: Number(roomId),
  discount: Number(discount),
  tillDate: new Date(tillDate),
 };
 try {
  await Sale.create(sale);
  return res.status(200).send("Sale added");
 } catch (error) {
  console.error(error);
  return res.status(400).send("Error adding sale");
 }
```

```
});
router.get("/all", async (req, res) => {
  try {
    const allSales = await Sale.find({});
    return res.status(200).json(allSales);
  } catch (error) {
    console.error(error);
    return res.status(400).send("Error fetching sales");
  }
});
router.get("/:roomId", async (req, res) => {
  const roomId = Number(req.params.roomId);
  try {
    const roomOnSale = await Sale.findOne({ roomId: roomId });
    return res.status(200).json({
      roomOnSale,
    });
  } catch (error) {
    console.error(error);
    return res.status(400).send("Error fetching sale");
  }
});
router.delete("/delete-expired", async (req, res) => {
  const now = new Date();
  try {
    const result = await Sale.deleteMany({ tillDate: { $lt: now } });
    return res.status(200).send(`Deleted ${result.deletedCount} expired sales`);
  } catch (error) {
    console.error("Error deleting expired sales:", error);
    return res.status(400).send("Error deleting expired sales");
  }
});
router.delete("/:roomId", async (req, res) => {
  const roomId = Number(req.params.roomId);
  try {
    await Sale.findOneAndDelete({ roomId: roomId });
    return res.status(200).send("Sale deleted");
  } catch (error) {
    console.error(error);
    return res.status(400).send("Error deleting sale");
```

```
  }
});
module.exports = router;
```

AdvertModule.jsx

```
import React, { useRef } from "react";
import axios from "axios";
const ModalAdvert = ({ isOpen, onClose, onSubmit, children }) => {
  const textRef = useRef(null);
  const imageRef = useRef(null);

  const handleSubmit = async () => {
    const text = textRef.current.value;
    const image = imageRef.current.files[0];

    const formData = new FormData();
    formData.append("msg", text);
    formData.append("image", image);

    try {
      await axios.post("https://royalapart.online/api/advert/save", formData, {
        headers: {
          "Content-Type": "multipart/form-data",
        },
      });
      setTimeout(async () => {
        try {
          await axios.get("https://royalapart.online/api/server2/getData");
          alert("Дані надіслані успішно:");
          onClose();
        } catch (error) {
          console.error("Error fetching data:", error);
        }
      }, 3000);

      onClose();
    } catch (error) {
      console.error("Error saving data:", error);
    }
  };
```

```
 return (
  <>
   {isOpen && (
    <div className="fixed inset-0 z-10 overflow-y-auto">
     <div className="flex items-center justify-center min-h-screen">
      <div className="fixed inset-0 transition-opacity" onClick={onClose}>
       <div className="absolute inset-0 bg-gray-500 opacity-75"></div>
      </div>
      <div className="z-20 bg-white p-6 rounded-lg shadow-lg">
       <div className="mb-4">
        <label
         htmlFor="advertText"
         className="block text-sm font-medium text-gray-700"
        >
         Text:
        </label>
        <input
         type="text"
         ref={textRef}
         id="advertText"
         className="mt-1 focus:ring-indigo-500 focus:border-indigo-500 block w-full shadow-sm sm:text-sm border-gray-
300 rounded-md"
        />
       </div>
       <div className="mb-4">
        <label
         htmlFor="advertImage"
         className="block text-sm font-medium text-gray-700"
        >
         Image:
        </label>
        <input
         type="file"
         ref={imageRef}
         id="advertImage"
         className="mt-1 focus:ring-indigo-500 focus:border-indigo-500 block w-full shadow-sm sm:text-sm border-gray-
300 rounded-md"
        />
       </div>
```

```
            <div className="flex justify-between">
              <button
                onClick={handleSubmit}
                className="bg-blue-500 text-white py-2 px-4 rounded hover:bg-blue-600"
              >
                Надіслати дані
              </button>
              <button
                onClick={onClose}
                className="bg-gray-300 text-gray-800 py-2 px-4 rounded hover:bg-gray-400"
              >
                Закрити
              </button>
            </div>
            <div>{children}</div>
          </div>
        </div>
      </div>
    )}
  </>
 );
};

export default ModalAdvert;
```