**НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

| | |
|---|---|
| **ПОГОДЖЕНО** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** |
| **Декан факультету Інформаційних технологій** | **Завідувач кафедри інформаційних систем і технологій** |
| **_____Ігор БОЛБОТ** | **_____Михайло ШВИДЕНКО** |
| **"___"_____2026 р.** | **"___"__ _____2026 р.** |

# БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: **«Інформаційна система пошуку оголошень оренди житла»**

Спеціальність 126 «Інформаційні системи і технології»

Освітньо-професійна програма «Інформаційні системи і технології»

**Гарант освітньої програми**
к.е.н., доцент **Максим МОКРІЄВ**

**Керівник бакалаврської кваліфікаційної роботи**
д.ф з к.н., старший викладач **Ярослав ПОНЗЕЛЬ**

**Консультант**
к.п.н., доцент **Тетяна ВОЛОШИНА**

**Виконала** **Софія КРИВША**

**КИЇВ-2026**

**НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
**Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
**інформаційних систем і технологій**

**к.е.н., доцент_____Михайло ШВИДЕНКО**

«___» _____ 2026 р.

**З А В Д А Н Н Я**
**ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**ЗДОБУВАЧУ**

**Кривші Софії Володимирівни**

Спеціальність 126 «Інформаційні системи і технології»
Освітньо-професійна програма «Інформаційні системи і технології»

Тема бакалаврської кваліфікаційної роботи «Інформаційна система пошуку оголошень оренди житла» затверджена наказом від «19» грудня 2025 року № 3205 «С»

Термін подання завершеної роботи на кафедру «___» _____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи:
відкриті веб-ресурси з оголошеннями оренди житла (OLX, DOM.RIA, Flatfy та інші); статистичні дані ринку нерухомості України; технічна документація мов програмування Python та JavaScript; документація фреймворків для розробки веб-застосунків.
Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області та ринку оренди житла в Україні
2. Дослідження існуючих систем пошуку оголошень оренди
3. Проєктування архітектури та структури даних інформаційної системи

Дата видачі завдання «19» січня 2026 р.

**Керівник бакалаврської**
**кваліфікаційної роботи** **Ярослав ПОНЗЕЛЬ**

**Консультант** **Тетяна ВОЛОШИНА**

**Завдання взяла до виконання** **Софія КРИВША**

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

API (Application Programming Interface) - прикладний програмний інтерфейс

BP (Blood Pressure) - артеріальний тиск

CRUD (Create, Read, Update, Delete) - базові операції роботи з даними: створення, читання, оновлення, видалення

DTO (Data Transfer Object) - об’єкт передавання даних

ER-діаграма (Entity–Relationship Diagram) - діаграма «сутність–зв’язок»

HTTPS (HyperText Transfer Protocol Secure) - захищений протокол передавання гіпертексту

ISO (International Organization for Standardization) - Міжнародна організація зі стандартизації

JWT (JSON Web Token) - маркер доступу у форматі JSON

RBAC (Role-Based Access Control) - керування доступом на основі ролей

REST (Representational State Transfer) - архітектурний стиль побудови веб-сервісів

TLS (Transport Layer Security) - протокол криптографічного захисту транспортного рівня

UML (Unified Modeling Language) - уніфікована мова моделювання

WHO (World Health Organization) - Всесвітня організація охорони здоров’я

ІС - інформаційна система

ПЗ - програмне забезпечення

# ВСТУП

Інформаційні технології відіграють ключову роль у цифровій трансформації сфери послуг, зокрема у галузі оренди приміщень, де ефективне управління ресурсами, автоматизація процесів бронювання та забезпечення зручної взаємодії між користувачами є критично важливими факторами. Зростання попиту на гнучкі робочі простори, конференц-зали, коворкінги та інші типи приміщень, а також розвиток бізнесу й віддаленої роботи зумовлюють необхідність створення сучасних програмних рішень, які забезпечують швидкий пошук, бронювання та управління орендними об’єктами. У традиційних підходах до організації оренди часто спостерігаються проблеми неефективного використання ресурсів, відсутності централізованого обліку, складності координації між орендодавцями та орендарями, а також обмежені можливості аналітики. У цьому контексті особливого значення набуває використання мікросервісної архітектури, яка забезпечує масштабованість, гнучкість, надійність та незалежність компонентів системи. Тому актуальним є розроблення веб-платформи бронювання оренди приміщень, що дозволяє інтегрувати всі ключові процеси в єдиному інформаційному середовищі, забезпечити їх автоматизацію, підвищити ефективність використання ресурсів і покращити користувацький досвід.

**Метою роботи є** дослідження та розробка веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури, яка забезпечує автоматизацію процесів пошуку, бронювання та управління приміщеннями, а також підвищує ефективність взаємодії між користувачами системи.

Для досягнення поставленої мети в роботі необхідно розв’язати такі основні **завдання:**

1. проаналізувати предметну область бронювання оренди приміщень та визначити її ключові особливості;
2. дослідити існуючі програмні рішення у сфері оренди та бронювання і визначити їхні переваги та обмеження;

3. сформувати функціональні, нефункціональні та безпекові вимоги до веб-платформи;

4. побудувати модель системи бронювання приміщень та UML-моделі її функціонування;

5. розробити архітектуру системи на основі мікросервісного підходу з урахуванням масштабованості та відмовостійкості;

6. реалізувати основні функціональні модулі системи, зокрема управління користувачами, каталогом приміщень, бронюванням і оплатою;

7. забезпечити механізми інтеграції із зовнішніми сервісами та обробки подій;

8. оцінити ефективність та практичну доцільність запропонованого рішення.

**Об’єктом дослідження є** процеси організації та автоматизації бронювання оренди приміщень із використанням інформаційних систем.

**Предметом дослідження є** методи, моделі та програмні засоби розроблення веб-платформи бронювання приміщень на основі мікросервісної архітектури, що забезпечують збір, оброблення, зберігання та аналіз відповідних даних.

У процесі виконання роботи використовуються методи системного аналізу для дослідження предметної області та формування вимог, методи об’єктно-орієнтованого моделювання для побудови UML-діаграм, методи проєктування розподілених систем для реалізації мікросервісної архітектури, а також інженерні методи розроблення програмного забезпечення для створення веб-платформи. Для оцінювання результатів застосовуються методи порівняльного аналізу, тестування та експериментальної перевірки функціонування системи.

# РОЗДІЛ: 1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ БРОНЮВАННЯ ОРЕНДИ ПРИМІЩЕНЬ

## 1.1 Актуальність роботи та аналіз проблеми системи бронювання оренди приміщень

Актуальність дослідження зумовлена зростанням попиту на цифрові сервіси оренди приміщень (офіси, коворкінги, конференц-зали), що вимагають швидкого пошуку, бронювання та централізованого управління ресурсами. Існуючі підходи, засновані на ручному обліку або ізольованих програмних рішеннях, не забезпечують узгодженості даних про доступність, що призводить до конфліктів бронювання, дублювання записів і обмеженої аналітики.

Сучасні інформаційні системи повинні забезпечувати автоматизацію життєвого циклу бронювання: пошук приміщень за параметрами, перевірку доступності в реальному часі, створення та оброблення бронювання, інтеграцію з платіжними сервісами та підсистемами сповіщень, а також збереження історичних даних для подальшого аналізу. Використання мікросервісної архітектури дозволяє декомпозувати систему на незалежні сервіси (користувачі, приміщення, бронювання, платежі, сповіщення), що забезпечує масштабованість, відмовостійкість і можливість незалежного розгортання компонентів.

Предметна область включає оброблення структурованих даних про користувачів, об’єкти оренди, часові інтервали доступності, транзакції та події системи. Процес бронювання реалізується як послідовність операцій: формування запиту, перевірка доступності, створення бронювання, ініціація оплати, фіксація результату транзакції та генерація сповіщень. Такий підхід забезпечує цілісність даних, узгодженість станів системи та підтримку подальшої аналітики використання ресурсів.

На рисунку 1.1 представлено контекстну модель предметної області у вигляді діаграми потоків даних, яка відображає взаємодію основних сутностей системи:

користувача (орендаря), власника приміщення, адміністратора, платіжного сервісу та сервісу сповіщень.


*Рисунок 1.1 – Контекстна діаграма потоків даних веб-платформи бронювання оренди приміщень*

Представлена модель демонструє, що ключовими процесами є пошук приміщень за заданими параметрами, оброблення заявок на бронювання, управління доступністю ресурсів, проведення платіжних операцій, надсилання сповіщень та формування аналітичної інформації. Потоки даних між зовнішніми сутностями та системою забезпечують повний цикл обробки інформації «запит – обробка – підтвердження – збереження – інформування», що є характерною особливістю сучасних інформаційних систем управління ресурсами.

Крім того, предметна область передбачає розподіл ролей і прав доступу, оскільки різні категорії користувачів взаємодіють із системою з різною метою: орендар здійснює пошук і бронювання приміщень, власник управляє ресурсами та обробляє заявки, а адміністратор забезпечує контроль, модерацію та налаштування роботи системи.

Для узагальнення основних компонентів предметної області та їх функціонального призначення у таблиці 1.1 наведено перелік ключових сутностей системи бронювання оренди приміщень.

Таблиця 1.1 – Основні сутності та функціональне призначення предметної області

| Сутність | Опис та призначення |
|---|---|
| Орендар (користувач) | Користувач системи, який здійснює пошук приміщень, створює бронювання та виконує оплату |
| Власник приміщення | Користувач, який додає приміщення, управляє їх доступністю та обробляє заявки |
| Адміністратор | Користувач, що здійснює керування системою, модерацію та контроль її функціонування |
| Приміщення | Об’єкт оренди з характеристиками (адреса, місткість, ціна, опис) |
| Доступність (розклад) | Дані про вільні та зайняті часові інтервали для бронювання |
| Бронювання | Інформація про заявку користувача на оренду приміщення |
| Платіж | Дані про фінансову операцію, пов’язану з бронюванням |
| Сервіс сповіщень | Підсистема для інформування користувачів про статуси операцій |
| Платіжний сервіс | Зовнішня система для обробки фінансових транзакцій |

Предметна область досліджуваної системи характеризується складністю, розподіленістю та орієнтацією на автоматизацію бізнес-процесів у сфері оренди приміщень. Вона поєднує процеси управління ресурсами, взаємодії користувачів і оброблення фінансових операцій у єдиному інформаційному середовищі, що забезпечує підвищення ефективності використання ресурсів і якості обслуговування користувачів.

## 1.2 Аналіз існуючих програмних рішень бронювання оренди приміщень

Сучасні програмні рішення у сфері бронювання оренди приміщень представлені веб-платформами, що забезпечують пошук об’єктів, керування доступністю, оброблення бронювань та інтеграцію платіжних сервісів. Аналіз таких систем дозволяє визначити їх функціональні можливості та обмеження, а також обґрунтувати необхідність розроблення власної платформи.

Одним із найбільш поширених рішень є платформа Booking.com, інтерфейс якої наведено на рисунку 1.2. Система забезпечує пошук об’єктів за параметрами, підтримує онлайн-бронювання та інтеграцію платіжних сервісів. Основною перевагою є висока масштабованість та надійність оброблення запитів, однак функціональність системи орієнтована переважно на туристичну сферу і не забезпечує гнучкого управління ресурсами в межах організації.

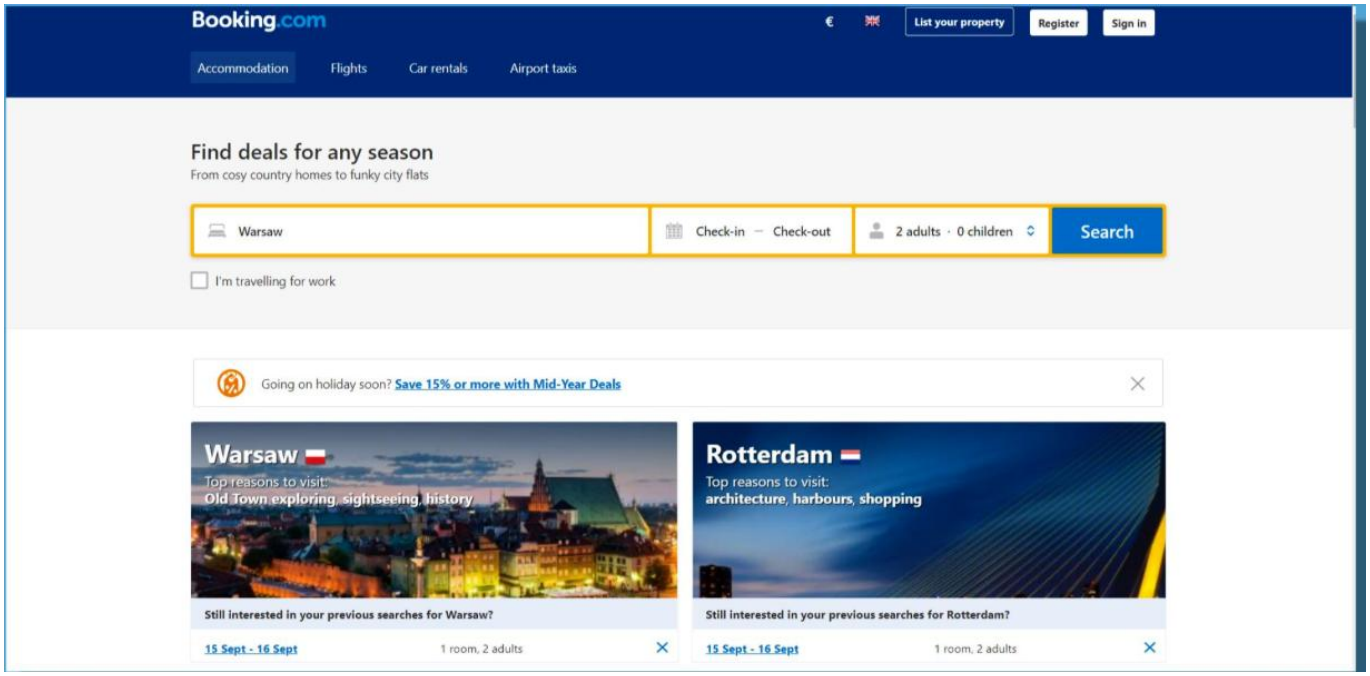


*Рисунок 1.2 – Інтерфейс платформи Booking.com для бронювання об’єктів*

Іншим прикладом є платформа Airbnb, що дозволяє здійснювати короткострокову оренду приміщень, інтерфейс якої наведено на рисунку 1.3. Система реалізує календар доступності, механізм підтвердження бронювання та

взаємодію між користувачами. Водночас аналітичні можливості системи обмежені, а її функціональність не орієнтована на автоматизацію внутрішніх бізнес-процесів.

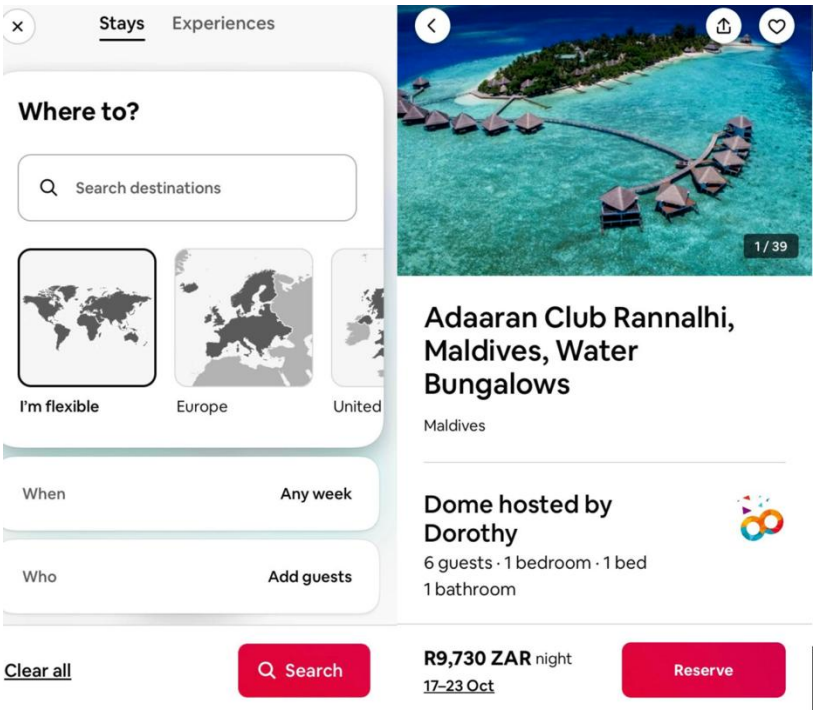


*Рисунок 1.3 – Інтерфейс платформи Airbnb для бронювання приміщень*

Спеціалізованим рішенням для управління приміщеннями є система Skedda, інтерфейс якої наведено на рисунку 1.4. Дана система забезпечує управління календарем бронювання, налаштування правил доступу та базову аналітику використання ресурсів. Перевагою є орієнтація на бізнес-середовище, однак система має обмежені можливості інтеграції та масштабування.

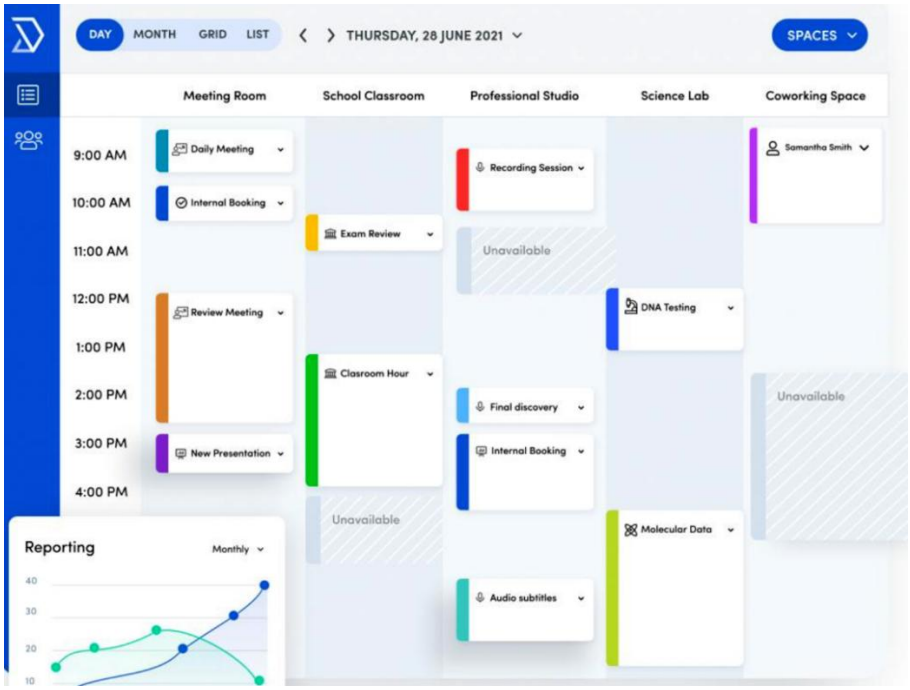


*Рисунок 1.4 – Інтерфейс системи Skedda для бронювання приміщень*

Для управління коворкінгами та офісними просторами використовується платформа OfficeRnD, приклад інтерфейсу якої наведено на рисунку 1.5. Система підтримує бронювання, облік користувачів та фінансові операції. Разом з тим, її впровадження є складним, а функціональність часто надлишкова для задач середнього рівня складності.

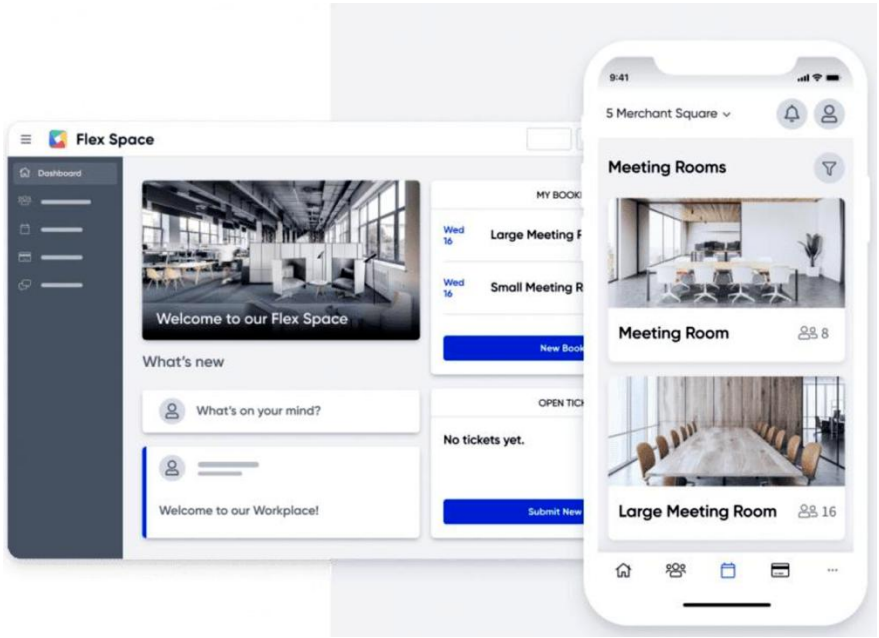


*Рисунок 1.5 – Інтерфейс системи OfficeRnD для управління просторами*

Крім спеціалізованих платформ, у практиці застосовуються універсальні інструменти, зокрема Google Calendar у поєднанні з додатковими сервісами, приклад використання якого наведено на рисунку 1.6. Такі рішення дозволяють реалізувати базове резервування часу, однак не підтримують повноцінну логіку бронювання, управління ресурсами та оброблення платежів.

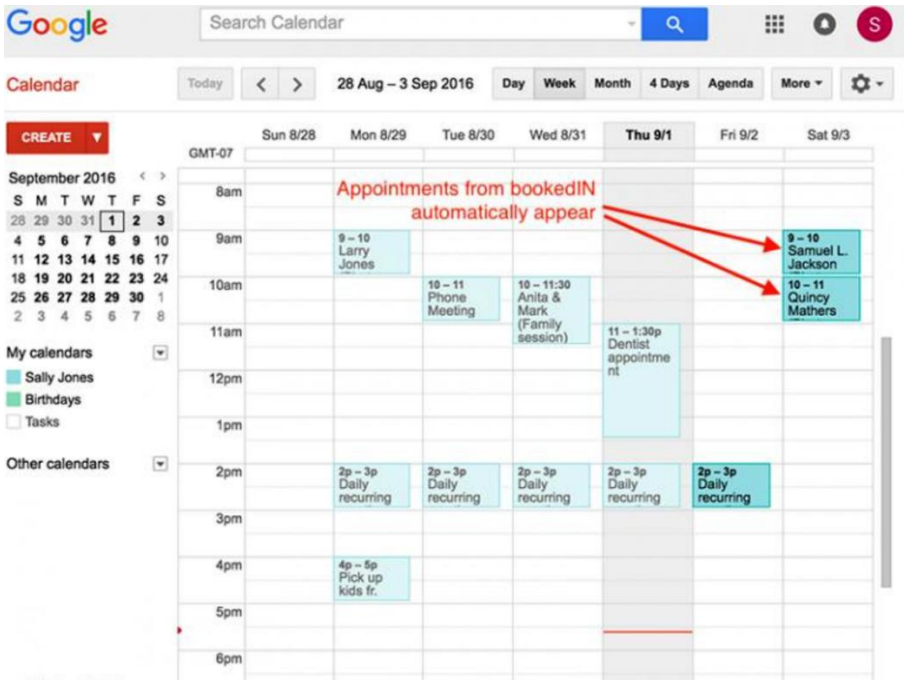


*Рисунок 1.6 – Приклад використання Google Calendar для бронювання*

Для узагальнення результатів аналізу у таблиці 1.2 наведено порівняльну характеристику розглянутих систем.

Таблиця 1.2 – Порівняльний аналіз систем бронювання приміщень

| Критерій | Booking.com | Airbnb | Skedda | OfficeRnD | Проєктована система |
|---|---|---|---|---|---|
| Пошук приміщень | Так | Так | Так | Так | Так |
| Управління доступністю | Частково | Так | Так | Так | Так |
| Онлайн-бронювання | Так | Так | Так | Так | Так |
| Інтеграція платежів | Так | Так | Частково | Так | Так |
| Аналітика | Обмежено | Обмежено | Частково | Так | Так |
| Масштабованість | Так | Так | Обмежено | Так | Так |
| Гнучкість налаштування | Ні | Ні | Частково | Так | Так |

Проведений аналіз показав, що існуючі системи забезпечують базовий функціонал бронювання, однак не реалізують повною мірою гнучку архітектуру, інтеграцію та аналітичні можливості. Це обґрунтовує доцільність розроблення власної веб-платформи на основі мікросервісної архітектури, що дозволяє забезпечити масштабованість, відмовостійкість та розширюваність системи.

## 1.3 Огляд методів і технологій для розробки веб-платформи бронювання оренди приміщень

Розроблення веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури потребує використання методів проєктування розподілених програмних систем, засобів моделювання бізнес-процесів, технологій веброзробки, механізмів інтеграції із зовнішніми сервісами та засобів надійного збереження даних. Основними вимогами до такої системи є коректна обробка запитів користувачів, синхронізація доступності приміщень, запобігання дублюванню бронювань, підтримка онлайн-оплати, формування сповіщень і забезпечення стабільної роботи за умов одночасного доступу багатьох користувачів.

У системі бронювання оренди приміщень доцільно застосовувати об’єктно-орієнтований підхід, оскільки предметна область містить чітко визначені сутності: користувач, власник приміщення, адміністратор, приміщення, часовий слот, бронювання, платіж і сповіщення. Кожна з цих сутностей має власні атрибути, стани та операції, тому їх зручно описувати у вигляді класів, зв’язків і сценаріїв взаємодії. Об’єктно-орієнтоване моделювання дозволяє формалізувати структуру системи до початку програмної реалізації та зменшити ризик помилок під час розроблення.

Для опису логіки роботи системи використовується UML-моделювання. Діаграма прецедентів дає змогу визначити основні функції платформи та ролі користувачів. Діаграма послідовності відображає порядок взаємодії між користувачем, інтерфейсом, сервісом бронювання, сервісом оплати та сервісом

сповіщень. Діаграма активності описує алгоритм виконання бронювання від пошуку приміщення до отримання підтвердження. Діаграма класів визначає структуру основних об’єктів системи, а діаграма компонентів і розгортання описують програмну та фізичну архітектуру платформи.

Основою архітектури проєктованої системи є мікросервісний підхід. На відміну від монолітної архітектури, у якій уся логіка розміщується в одному застосунку, мікросервісна архітектура передбачає поділ системи на окремі незалежні сервіси. Для даної предметної області доцільно виділити сервіс користувачів, сервіс приміщень, сервіс бронювання, сервіс платежів і сервіс сповіщень. Такий поділ дозволяє незалежно розробляти, тестувати, оновлювати та масштабувати окремі частини системи без повної зупинки платформи.

Взаємодія між клієнтською частиною та серверними модулями може реалізовуватися через REST API. Такий підхід забезпечує стандартизований обмін даними між вебінтерфейсом, API Gateway та окремими мікросервісами. API Gateway виконує роль єдиної точки входу для клієнтських запитів, маршрутизує їх до відповідних сервісів, перевіряє права доступу та спрощує взаємодію користувача з розподіленою системою.

Для збереження структурованих даних доцільно використовувати реляційну базу даних. У ній зберігаються облікові записи користувачів, ролі, інформація про приміщення, категорії, часові слоти, бронювання, платежі, відгуки та сповіщення. Реляційна модель забезпечує підтримку первинних і зовнішніх ключів, цілісність зв’язків між таблицями та можливість виконання складних запитів для формування звітів і аналітики. Для системи бронювання особливо важливо забезпечити унікальність часових слотів і недопущення одночасного бронювання одного приміщення на той самий період.

Клієнтська частина веб-платформи має забезпечувати зручну взаємодію користувача із системою. Через вебінтерфейс орендар повинен мати змогу переглядати список доступних приміщень, фільтрувати їх за параметрами, відкривати картку приміщення, обирати дату й час, створювати бронювання та переглядати його статус. Власник приміщення повинен отримати інструменти для

додавання об’єктів, редагування характеристик, керування доступністю та перегляду заявок. Адміністратор повинен мати доступ до модерації користувачів, контролю бронювань і перегляду звітної інформації.

Важливою складовою системи є інтеграція з платіжними сервісами. Платіжний модуль має приймати запит на оплату, передавати дані до зовнішнього платіжного шлюзу, отримувати статус транзакції та передавати результат до сервісу бронювання. У разі успішної оплати бронювання отримує статус підтвердженого, а у разі помилки користувач отримує відповідне повідомлення. Такий механізм дозволяє автоматизувати фінансову частину процесу та зменшити кількість ручних операцій.

Сервіс сповіщень використовується для інформування користувачів про ключові події системи. До таких подій належать створення бронювання, підтвердження оплати, зміна статусу заявки, скасування бронювання або оновлення даних приміщення. Сповіщення можуть надсилатися через електронну пошту, SMS або внутрішню систему повідомлень платформи. Це підвищує прозорість процесу бронювання та зменшує ризик втрати важливої інформації.

Для узагальнення основних методів і технологій, які доцільно використовувати під час розроблення веб-платформи бронювання оренди приміщень, у таблиці 1.3 наведено їх характеристику та призначення.

Таблиця 1.3 – Основні методи та технології розробки веб-платформи бронювання оренди приміщень

| Метод / технологія | Призначення | Особливості використання |
|---|---|---|
| Об’єктно-орієнтоване програмування | Побудова програмної логіки системи | Дозволяє описати користувачів, приміщення, бронювання, платежі та сповіщення як окремі об’єкти |
| UML-моделювання | Проєктування структури та поведінки системи | Використовується для побудови діаграм прецедентів, класів, послідовності, активності, компонентів і розгортання |
| Мікросервісна архітектура | Розподіл функціональності між незалежними сервісами | Забезпечує масштабованість, відмовостійкість і незалежне оновлення окремих модулів |
| API Gateway | Єдина точка входу до серверної частини | Маршрутизує клієнтські запити до відповідних мікросервісів |
| REST API | Обмін даними між клієнтом і сервером | Забезпечує стандартизовану взаємодію між вебінтерфейсом і серверними компонентами |
| Реляційна база даних | Збереження структурованої інформації | Використовується для зберігання користувачів, приміщень, слотів, бронювань і платежів |
| Вебтехнології | Реалізація інтерфейсу користувача | Забезпечують доступ до системи через браузер |
| Платіжні API | Оброблення фінансових операцій | Дозволяють інтегрувати оплату бронювання із зовнішніми платіжними сервісами |
| Сервіси сповіщень | Інформування користувачів | Використовуються для надсилання повідомлень про статус бронювання та оплати |
| Контейнеризація | Розгортання окремих сервісів | Забезпечує ізольоване середовище виконання для мікросервісів |
| Засоби тестування | Перевірка працездатності системи | Використовуються для перевірки коректності бронювання, оплати, авторизації та обробки помилок |

Застосування зазначених методів і технологій дозволяє побудувати платформу з чітким розподілом відповідальності між компонентами. Сервіс користувачів відповідає за реєстрацію, авторизацію та ролі доступу. Сервіс приміщень забезпечує зберігання характеристик об’єктів оренди та керування їх доступністю. Сервіс бронювання виконує перевірку часових слотів, створює заявки та змінює їх статус.

Сервіс платежів відповідає за обробку фінансових операцій, а сервіс сповіщень - за інформування користувачів.

Використання мікросервісної архітектури, REST API, реляційної бази даних та засобів UML-моделювання створює технічну основу для побудови масштабованої веб-платформи бронювання оренди приміщень. Обрані технології забезпечують структуровану реалізацію бізнес-логіки, підтримку інтеграцій, контроль цілісності даних і можливість подальшого розвитку системи.

## 1.4 Аналіз підґрунтя для розробки веб-платформи бронювання оренди приміщень

Аналіз підґрунтя для розроблення веб-платформи бронювання оренди приміщень передбачає визначення організаційних, технічних і функціональних факторів, які обумовлюють необхідність створення такого програмного рішення. До таких факторів належать зростання попиту на короткострокову оренду приміщень, потреба в автоматизації бронювання, необхідність централізованого управління даними, розвиток вебтехнологій, поширення онлайн-платежів і потреба у масштабованих архітектурних рішеннях.

Однією з основних передумов створення системи є складність ручного управління бронюванням. У випадку використання таблиць, телефонних заявок або окремих месенджерів виникає ризик дублювання бронювань, втрати інформації про заявки, несвоєчасного оновлення доступності приміщень і помилок під час узгодження часу оренди. Такий підхід не забезпечує прозорої історії операцій і ускладнює контроль завантаженості об’єктів.

Другою передумовою є необхідність оперативної перевірки доступності приміщень. У системах бронювання важливо, щоб інформація про вільні та зайняті часові інтервали оновлювалася одразу після створення або скасування заявки. Якщо цей процес не автоматизований, користувач може бачити неактуальні дані, що призводить до конфліктів бронювання. Тому система повинна підтримувати

централізоване зберігання слотів доступності та механізм перевірки зайнятості перед створенням бронювання.

Третьою важливою передумовою є потреба власників приміщень у контролі ресурсів. Власник або менеджер повинен мати змогу додавати нові приміщення, змінювати їх характеристики, встановлювати ціну, визначати доступні часові інтервали, переглядати заявки та контролювати фінансові операції. Без спеціалізованої системи ці дії виконуються вручну, що збільшує навантаження на персонал і знижує точність обліку.

Окреме значення має інтеграція з платіжними сервісами. Онлайн-оплата дозволяє автоматично фіксувати факт фінансової операції, змінювати статус бронювання та зменшувати кількість ручних підтверджень. Для платформи бронювання це є важливим елементом, оскільки оплата часто є умовою остаточного підтвердження заявки. Система повинна підтримувати обробку успішних і неуспішних транзакцій, збереження платіжного статусу та зв’язок платежу з конкретним бронюванням.

Наступною передумовою є необхідність автоматичного інформування користувачів. Під час роботи платформи користувачі повинні отримувати повідомлення про створення заявки, підтвердження бронювання, зміну статусу, оплату або скасування. Реалізація сервісу сповіщень дозволяє підвищити прозорість взаємодії між орендарем і власником приміщення та зменшити кількість ручних повідомлень.

Технічним підґрунтям для розроблення системи є розвиток вебтехнологій, хмарної інфраструктури та контейнеризації. Вебплатформа може бути доступною з різних пристроїв через браузер, а серверна частина може бути розгорнута у вигляді набору незалежних сервісів. Це забезпечує можливість масштабування окремих компонентів залежно від навантаження. Наприклад, сервіс бронювання може потребувати більшої кількості ресурсів у періоди активного попиту, тоді як сервіс сповіщень або адміністрування може працювати з меншим навантаженням.

Мікросервісна архітектура є доцільною для цієї предметної області, оскільки функції системи мають чіткий розподіл. Управління користувачами, каталог

приміщень, бронювання, платежі та сповіщення є відносно незалежними підсистемами. Такий поділ знижує складність розроблення, спрощує тестування та дозволяє оновлювати окремі компоненти без повного перерозгортання всієї платформи.

Для систематизації основних факторів, які формують підґрунтя для розроблення веб-платформи бронювання оренди приміщень, у таблиці 1.4 наведено їх характеристику.

Таблиця 1.4 – Основні передумови розроблення веб-платформи бронювання оренди приміщень

| Фактор | Характеристика | Значення для розробки системи |
|---|---|---|
| Потреба в автоматизації бронювання | Ручна обробка заявок призводить до помилок і дублювання записів | Обґрунтовує необхідність створення централізованої системи бронювання |
| Необхідність контролю доступності | Інформація про вільні часові слоти має бути актуальною | Вимагає реалізації механізму перевірки доступності перед створенням бронювання |
| Управління приміщеннями | Власник повинен керувати характеристиками, цінами та розкладом | Потребує окремого модуля керування об’єктами оренди |
| Онлайн-оплата | Оплата є частиною процесу підтвердження бронювання | Потребує інтеграції з платіжним сервісом |
| Сповіщення користувачів | Користувачі мають отримувати інформацію про зміну статусу заявки | Потребує реалізації сервісу повідомлень |
| Вебдоступ | Користувачі працюють із системою через браузер | Забезпечує доступність платформи з різних пристроїв |
| Масштабованість | Кількість користувачів і бронювань може зростати | Обґрунтовує використання мікросервісної архітектури |
| Аналітика використання ресурсів | Власнику потрібна інформація про завантаженість і доходи | Потребує накопичення історичних даних і формування звітів |

Зазначені фактори свідчать про наявність технічних і організаційних передумов для створення веб-платформи бронювання оренди приміщень. Система повинна не лише забезпечувати створення бронювань, але й підтримувати повний інформаційний цикл: реєстрацію користувача, пошук приміщення, перевірку

доступності, оформлення заявки, оплату, підтвердження, сповіщення та подальший аналіз даних.

Проведений аналіз підґрунтя підтверджує доцільність використання мікросервісної архітектури. Такий підхід відповідає структурі предметної області, оскільки дозволяє виділити окремі сервіси для роботи з користувачами, приміщеннями, бронюваннями, оплатами та повідомленнями. Це підвищує керованість системи, спрощує її супровід і забезпечує можливість подальшого розширення функціональних можливостей.

## 1.5 Постановка завдання веб-платформи бронювання оренди приміщень

На основі проведеного аналізу предметної області, існуючих програмних рішень, методів проєктування та технологічного підґрунтя постає завдання розроблення веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури. Основною ідеєю дослідження є створення програмної системи, яка забезпечує автоматизацію процесів пошуку, бронювання, оплати та управління приміщеннями в єдиному інформаційному середовищі.

У межах поставленого завдання система повинна забезпечувати взаємодію між орендарями, власниками приміщень та адміністратором. Орендар повинен мати можливість зареєструватися, виконати пошук приміщень за заданими параметрами, переглянути інформацію про об’єкт, обрати доступний часовий інтервал, створити бронювання, здійснити оплату та отримати підтвердження. Власник приміщення повинен мати змогу додавати об’єкти оренди, редагувати їх параметри, керувати календарем доступності, переглядати заявки та контролювати статуси бронювань. Адміністратор повинен забезпечувати модерацію користувачів і приміщень, контроль коректності роботи системи та перегляд аналітичної інформації.

Система повинна підтримувати повний цикл оброблення бронювання. Спочатку користувач формує пошуковий запит із зазначенням параметрів приміщення, дати, часу та інших критеріїв. Після цього система перевіряє наявність

доступних об’єктів і відображає результати. Під час створення бронювання виконується повторна перевірка доступності часового слота, що дозволяє уникнути конфліктів. Після створення заявки система ініціює оплату, фіксує результат транзакції та змінює статус бронювання. Після завершення операції користувач і власник отримують відповідні сповіщення.

Особливу увагу необхідно приділити забезпеченню цілісності даних. Для цього в базі даних повинні бути визначені зв’язки між користувачами, приміщеннями, часовими слотами, бронюваннями та платежами. Кожне бронювання має бути пов’язане з конкретним користувачем, приміщенням і часовим інтервалом. Платіж повинен бути пов’язаний із відповідним бронюванням, а сповіщення — з користувачем і подією, що його спричинила.

Архітектура системи повинна бути побудована за мікросервісним принципом. До складу платформи доцільно включити сервіс автентифікації та користувачів, сервіс керування приміщеннями, сервіс бронювання, сервіс платежів, сервіс сповіщень та API Gateway. Такий підхід дозволяє розподілити відповідальність між компонентами, спростити тестування та забезпечити можливість незалежного масштабування окремих сервісів.

Для формалізації задачі у таблиці 1.5 наведено перелік вхідних і вихідних даних системи, які визначають основні інформаційні потоки між користувачами, програмними модулями та зовнішніми сервісами.

Таблиця 1.5 – Вхідні та вихідні дані веб-платформи бронювання оренди приміщень

| Тип даних | Опис |
|---|---|
| Вхідні дані | Облікові дані користувача для реєстрації та автентифікації |
| Вхідні дані | Дані профілю користувача, власника або адміністратора |
| Вхідні дані | Характеристики приміщення: назва, адреса, категорія, місткість, ціна, опис |
| Вхідні дані | Дані про доступність приміщення: дата, час початку, час завершення, статус слота |
| Вхідні дані | Параметри пошуку: локація, дата, час, місткість, вартість, категорія приміщення |
| Вхідні дані | Дані бронювання: користувач, приміщення, часовий інтервал, статус заявки |
| Вхідні дані | Платіжні дані та результат оброблення транзакції |
| Вхідні дані | Дані для надсилання сповіщень користувачам |
| Вихідні дані | Список доступних приміщень відповідно до параметрів пошуку |
| Вихідні дані | Детальна інформація про вибране приміщення |
| Вихідні дані | Підтвердження створення або відхилення бронювання |
| Вихідні дані | Статус оплати та результат фінансової операції |
| Вихідні дані | Сповіщення про зміну статусу бронювання |
| Вихідні дані | Аналітичні звіти щодо завантаженості приміщень і кількості бронювань |

У рамках виконання поставленого завдання необхідно реалізувати такі основні етапи: проєктування логічної структури бази даних; побудову UML-моделей системи; розроблення мікросервісної архітектури; створення вебінтерфейсу для орендаря, власника та адміністратора; реалізацію сервісу керування приміщеннями; реалізацію сервісу бронювання з перевіркою доступності; інтеграцію з платіжним сервісом; реалізацію механізму сповіщень; тестування основних сценаріїв роботи системи.

Результатом виконання поставленого завдання має бути програмне рішення, яке забезпечує централізоване управління процесами бронювання оренди приміщень. Система повинна зменшити кількість ручних операцій, підвищити точність обліку доступності, забезпечити контроль статусів бронювання та створити основу для подальшої аналітики використання ресурсів.

Постановка завдання визначає межі розроблення веб-платформи та формує основу для подальшого проєктування інформаційного, програмного й архітектурного забезпечення системи.

**Висновки до першого розділу**

У першому розділі виконано аналіз предметної області бронювання оренди приміщень і визначено основні проблеми, які виникають під час використання ручного обліку або розрізнених програмних рішень. Встановлено, що ключовими недоліками таких підходів є неузгодженість даних про доступність приміщень, ризик дублювання бронювань, складність контролю статусів заявок, відсутність централізованого збереження інформації та обмежені можливості аналітики.

Проведено огляд існуючих програмних рішень для бронювання приміщень і управління просторами. Аналіз показав, що наявні системи забезпечують базові функції пошуку, бронювання та керування ресурсами, однак не завжди відповідають вимогам гнучкої інтеграції, масштабування та адаптації під конкретні бізнес-процеси. Це обґрунтовує доцільність розроблення власної веб-платформи, орієнтованої на автоматизацію повного циклу бронювання.

Розглянуто методи та технології, які доцільно використовувати під час розроблення системи. Визначено, що основою проєктування є об’єктно-орієнтований підхід, UML-моделювання, REST API, реляційна база даних та мікросервісна архітектура. Мікросервісний підхід дозволяє розподілити функціональність між незалежними сервісами користувачів, приміщень, бронювання, платежів і сповіщень, що підвищує масштабованість і керованість системи.

У межах аналізу підґрунтя встановлено основні технічні та організаційні передумови створення системи: потребу в автоматизації бронювання, необхідність актуального контролю доступності приміщень, інтеграцію з платіжними сервісами, автоматичне інформування користувачів, захист персональних даних і формування

звітної інформації. Визначено, що ці фактори безпосередньо впливають на архітектуру та функціональний склад майбутньої платформи.

На основі проведеного аналізу сформульовано постановку завдання, визначено основні вхідні та вихідні дані системи, а також описано ключові етапи її реалізації. Отримані результати створюють технічне підґрунтя для подальшого проєктування архітектури, структури бази даних, програмних модулів і механізмів взаємодії компонентів веб-платформи бронювання оренди приміщень.

# РОЗДІЛ: 2 АНАЛІЗ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

## 2.1 Моделювання бізнес-процесів веб-платформи бронювання оренди приміщень

Моделювання бізнес-процесів є одним з основних етапів проєктування веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури, оскільки дозволяє формалізувати функціональні сценарії системи, визначити ролі користувачів, послідовність оброблення запитів і взаємодію між окремими сервісами. Для опису предметної області використано UML-моделі, які відображають поведінку системи на рівні прецедентів, послідовності повідомлень і алгоритму виконання основного процесу бронювання.

На рисунку 2.1 наведено діаграму прецедентів веб-платформи бронювання оренди приміщень. Діаграма відображає основні функціональні можливості системи та акторів, які беруть участь у процесі бронювання. У межах моделі виокремлено п’ять акторів: гість, орендар/клієнт, власник приміщення, адміністратор, платіжний сервіс і сервіс сповіщень. Такий склад акторів дозволяє показати як дії користувачів платформи, так і взаємодію із зовнішніми сервісами, необхідними для оброблення платежів і надсилання повідомлень.

Гість взаємодіє із системою на початковому етапі та має доступ до реєстрації, входу, пошуку приміщень і перегляду детальної інформації про об’єкти оренди. Після авторизації користувач переходить до ролі орендаря/клієнта та отримує можливість оформлювати заявки на бронювання, керувати власними бронюваннями, скасовувати їх і здійснювати оплату. Перед створенням заявки система виконує перевірку доступності приміщення, що є обов’язковою умовою для уникнення конфліктів бронювання.

Власник приміщення взаємодіє із системою з метою керування об’єктами оренди. Він може додавати та редагувати приміщення, задавати доступність і ціну,

переглядати заявки, підтверджувати або відхиляти бронювання, а також формувати звіти щодо бронювань. Адміністратор забезпечує контроль роботи платформи, зокрема адміністрування користувачів та оголошень. Платіжний сервіс використовується для оброблення оплати, а сервіс сповіщень — для інформування користувачів про створення, підтвердження, скасування або зміну статусу бронювання.

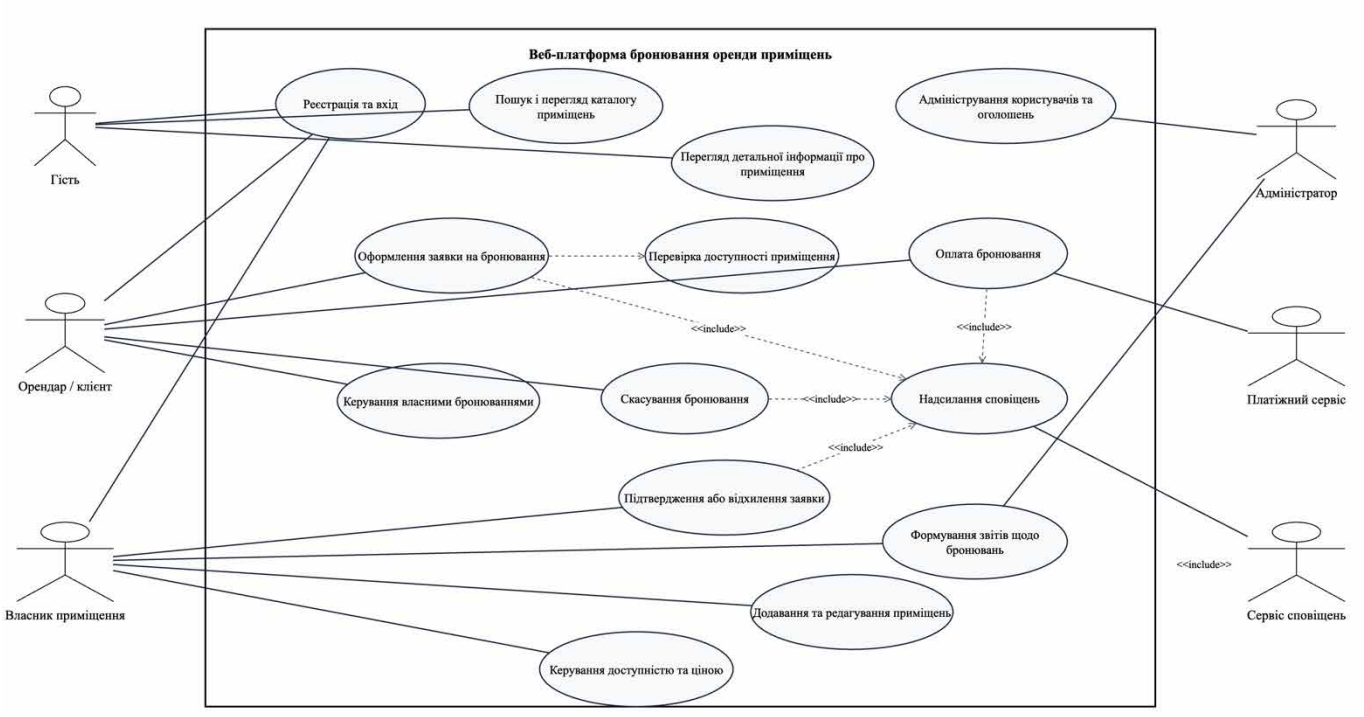


*Рисунок 2.1 – Діаграма прецедентів веб-платформи бронювання оренди приміщень*

Діаграма прецедентів показує, що центральним процесом системи є оформлення заявки на бронювання. Цей сценарій пов’язаний із перевіркою доступності приміщення, оплатою бронювання та надсиланням сповіщень. Використання зв’язків include є доцільним, оскільки перевірка доступності та інформування користувачів є обов’язковими підпроцесами під час виконання основних сценаріїв. Така модель дозволяє визначити межі функціональності системи та перейти до деталізації взаємодії між компонентами.

Для уточнення порядку виконання основного сценарію побудовано діаграму послідовності, наведену на рисунку 2.2. Вона описує процес бронювання

приміщення з урахуванням мікросервісної архітектури. У моделі показано взаємодію між орендарем, вебінтерфейсом, Auth Service, Room Service, Booking Service, Payment Service та Notification Service. Такий поділ відповідає архітектурі системи, у якій кожен сервіс відповідає за окрему частину бізнес-логіки.

На початку сценарію орендар вводить параметри пошуку у вебінтерфейсі. Вебінтерфейс звертається до Auth Service для перевірки активної сесії користувача. Після підтвердження сесії запит передається до Room Service, який повертає перелік доступних приміщень відповідно до заданих параметрів. Далі користувач обирає приміщення та часовий період, після чого система переходить до перевірки доступності та створення попереднього бронювання.

У діаграмі використано альтернативний фрагмент alt, який відображає два можливі варіанти виконання сценарію: приміщення доступне або приміщення недоступне. Якщо приміщення доступне, вебінтерфейс передає дані до Booking Service, який створює попереднє бронювання. Якщо приміщення недоступне, користувач отримує повідомлення про неможливість бронювання вибраного часу. Такий підхід дозволяє формалізувати логіку запобігання дублюванню бронювань.

Після створення попереднього бронювання система формує платіж через Payment Service. Другий альтернативний фрагмент описує результати оброблення оплати. Якщо оплата успішна, Booking Service підтверджує бронювання, після чого Notification Service надсилає користувачу відповідне повідомлення. Якщо оплату відхилено, попереднє бронювання скасовується, а користувач отримує інформацію про помилку. Така послідовність забезпечує узгодженість станів між бронюванням, оплатою та сповіщеннями.

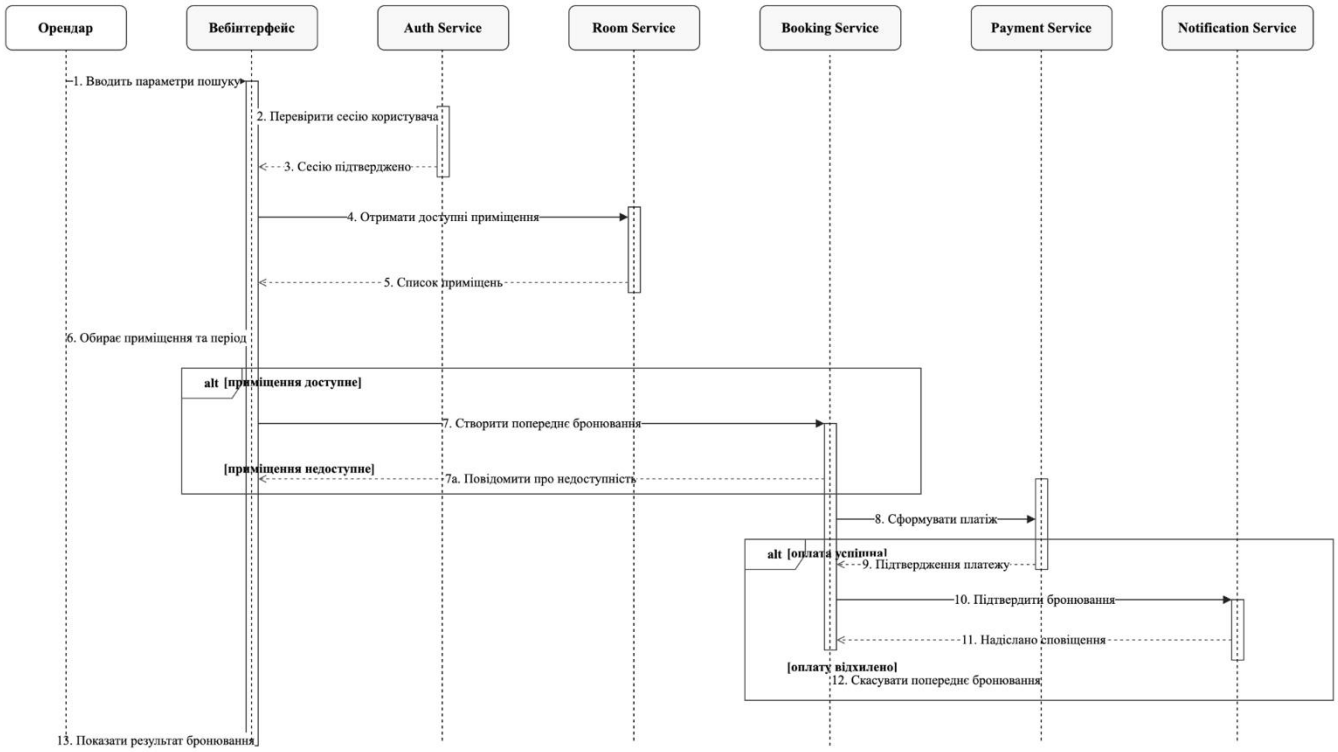


*Рисунок 2.2 – Діаграма послідовності процесу бронювання приміщення*

Діаграма послідовності деталізує інформаційні потоки між сервісами та демонструє розподіл відповідальності між компонентами системи. Auth Service відповідає за перевірку сесії, Room Service - за отримання інформації про доступні приміщення, Booking Service - за створення та зміну статусу бронювання, Payment Service — за оброблення фінансової операції, а Notification Service - за інформування користувача. Такий підхід відповідає принципам мікросервісної архітектури та спрощує подальшу реалізацію програмної системи.

Поведенкову логіку основного процесу відображено на діаграмі активності, наведеній на рисунку 2.3. Діаграма описує повний цикл бронювання з розподілом відповідальності між трьома доріжками: користувач/орендар, веб-платформа бронювання, мікросервіси та зовнішні сервіси. Такий поділ дозволяє чітко визначити, які дії виконує користувач, які операції реалізуються на рівні платформи, а які передаються окремим сервісам.

Процес починається з відкриття веб-платформи та введення критеріїв пошуку. Веб-платформа обробляє параметри запиту й передає їх до мікросервісів для отримання переліку доступних приміщень. Після повернення результатів система

відображає користувачу доступні варіанти. Далі орендар обирає приміщення, дату й час, після чого платформа виконує перевірку доступності вибраного об’єкта.

Ключовим елементом діаграми активності є умовний вузол «Приміщення доступне?». Якщо вибраний часовий інтервал недоступний, система повідомляє користувача про недоступність і повертає його до вибору іншого приміщення або часу. Якщо приміщення доступне, система формує заявку на бронювання та передає її до мікросервісів для створення попереднього бронювання. Після цього користувач підтверджує умови бронювання, а платформа передає дані для оплати.

Наступним етапом є оброблення платежу. Умовний вузол «Оплату підтверджено?» визначає подальший перебіг процесу. Якщо платіж не підтверджено, система скасовує попереднє бронювання та відображає повідомлення про помилку. Якщо оплата успішна, мікросервіси фіксують бронювання, надсилають підтвердження, а користувач отримує кінцевий результат операції. Завершення процесу відбувається після отримання підтвердження бронювання.

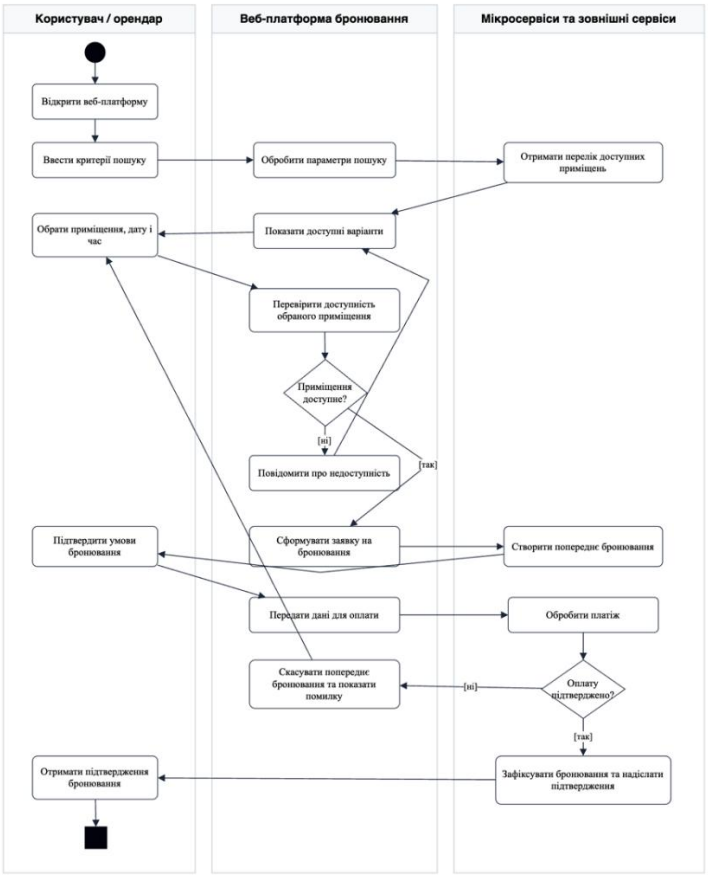


*Рисунок 2.3 – Діаграма активності процесу бронювання оренди приміщення*

Діаграма активності дозволяє визначити основні контрольні точки процесу: перевірку доступності приміщення, створення попереднього бронювання, підтвердження умов, оброблення платежу та фіксацію фінального статусу. Саме ці етапи є критичними для коректної роботи системи, оскільки помилки на будь-якому з них можуть призвести до некоректного бронювання або невідповідності між статусом платежу та статусом заявки.

Узагальнюючи результати моделювання, можна зробити висновок, що побудовані UML-діаграми забезпечують формальне представлення основних процесів веб-платформи бронювання оренди приміщень. Діаграма прецедентів визначає функціональні межі системи та ролі учасників, діаграма послідовності деталізує взаємодію між користувачем і мікросервісами, а діаграма активності описує алгоритм виконання основного бізнес-процесу. Отримані моделі створюють основу для подальшого проєктування структури даних, програмних компонентів і архітектури розгортання системи.

## 2.2 Аналіз вимог веб-платформи бронювання оренди приміщень

Аналіз вимог є необхідним етапом проєктування веб-платформи бронювання оренди приміщень, оскільки визначає функціональні можливості системи, якісні характеристики її роботи та обмеження безпеки. Сформовані вимоги є основою для побудови архітектури, проєктування бази даних, реалізації мікросервісів і розроблення користувацького інтерфейсу.

### 2.2.1 Функціональні вимоги веб-платформи бронювання оренди приміщень

Функціональні вимоги визначають набір операцій, які повинна виконувати система для підтримки процесів пошуку, бронювання, оплати та адміністрування приміщень. Узагальнений перелік функціональних вимог наведено в таблиці 2.1.

Таблиця 2.1 – Функціональні вимоги системи

| Позначення | Опис вимоги |
|---|---|
| FR1 | Реєстрація та автентифікація користувачів |
| FR2 | Розмежування ролей: орендар, власник приміщення, адміністратор |
| FR3 | Додавання та редагування інформації про приміщення |
| FR4 | Пошук приміщень за датою, часом, місткістю, ціною та категорією |
| FR5 | Перегляд детальної інформації про приміщення |
| FR6 | Керування доступністю приміщень за часовими слотами |
| FR7 | Перевірка доступності приміщення перед створенням бронювання |
| FR8 | Створення, підтвердження та скасування бронювання |
| FR9 | Оброблення оплати через зовнішній платіжний сервіс |
| FR10 | Збереження статусу бронювання та результату платежу |
| FR11 | Надсилання сповіщень про створення, оплату, підтвердження або скасування бронювання |
| FR12 | Формування звітів щодо бронювань, завантаженості приміщень і платежів |
| FR13 | Адміністрування користувачів, оголошень і системних налаштувань |

### 2.2.2 Нефункціональні вимоги веб-платформи бронювання оренди приміщень

Нефункціональні вимоги визначають якість роботи системи: швидкодію, надійність, масштабованість, доступність і зручність використання. Основні нефункціональні вимоги подано в таблиці 2.2.

Таблиця 2.2 – Нефункціональні вимоги системи

| Категорія | Опис вимоги |
|---|---|
| Продуктивність | Час оброблення пошукового запиту не більше 2 с |
| Продуктивність | Час створення бронювання після підтвердження користувачем не більше 3 с |
| Надійність | Недопущення дублювання бронювань для одного приміщення на один часовий слот |
| Доступність | Робота системи 24/7 з рівнем доступності не менше 99 % |
| Масштабованість | Можливість незалежного масштабування сервісів користувачів, приміщень, бронювання, платежів і сповіщень |
| Зручність використання | Інтерфейс має бути зрозумілим для орендаря, власника приміщення та адміністратора |
| Сумісність | Підтримка роботи в сучасних браузерах |
| Інтегрованість | Можливість взаємодії з платіжними сервісами та сервісами сповіщень |
| Супроводжуваність | Можливість оновлення окремих мікросервісів без зупинки всієї платформи |
| Відмовостійкість | Коректна обробка помилок оплати, недоступності сервісів і повторних запитів |

### 2.2.3 Вимоги до безпеки системи

Оскільки система обробляє персональні дані користувачів, інформацію про бронювання та платіжні операції, необхідно забезпечити автентифікацію, розмежування прав доступу та захист даних під час передавання і зберігання. Вимоги безпеки наведено в таблиці 2.3.

Таблиця 2.3 – Вимоги до безпеки системи

| Позначення | Опис вимоги |
|---|---|
| SR1 | Захист персональних даних користувачів |
| SR2 | Автентифікація користувачів перед виконанням операцій бронювання |
| SR3 | Розмежування прав доступу за ролями: орендар, власник, адміністратор |
| SR4 | Захищене передавання даних між клієнтом і сервером через HTTPS |
| SR5 | Обмеження доступу до платіжних даних і результатів транзакцій |
| SR6 | Захист бази даних від несанкціонованого доступу |
| SR7 | Журналювання критичних дій: створення бронювання, оплата, скасування, зміна доступності |
| SR8 | Перевірка вхідних даних для запобігання некоректним або шкідливим запитам |

Сформовані функціональні, нефункціональні та безпекові вимоги визначають основні характеристики веб-платформи бронювання оренди приміщень. Вони є основою для подальшого проєктування мікросервісної архітектури, структури бази даних, програмних компонентів і механізмів взаємодії із зовнішніми сервісами.

## 2.3 Логічна модель бази даних системи

Логічна модель бази даних веб-платформи бронювання оренди приміщень визначає структуру збереження даних, зв’язки між основними сутностями та правила забезпечення цілісності інформації. База даних спроєктована у реляційному вигляді та приведена до третьої нормальної форми, що дозволяє усунути надлишкове дублювання даних, забезпечити однозначність зв’язків між таблицями та спростити подальше розширення системи.

У межах логічної моделі виділено сутності, які відповідають основним бізнес-об’єктам платформи: користувачам, ролям, приміщенням, категоріям приміщень, часовим слотам доступності, бронюванням, платежам, відгукам і сповіщенням. Кожна сутність має первинний ключ, а зв’язки між таблицями реалізовані через зовнішні ключі. Такий підхід забезпечує контроль посилальної цілісності та дозволяє коректно відстежувати повний цикл бронювання від вибору приміщення до оплати й отримання сповіщення.

Логічну модель бази даних веб-платформи бронювання оренди приміщень наведено на рисунку 2.4.

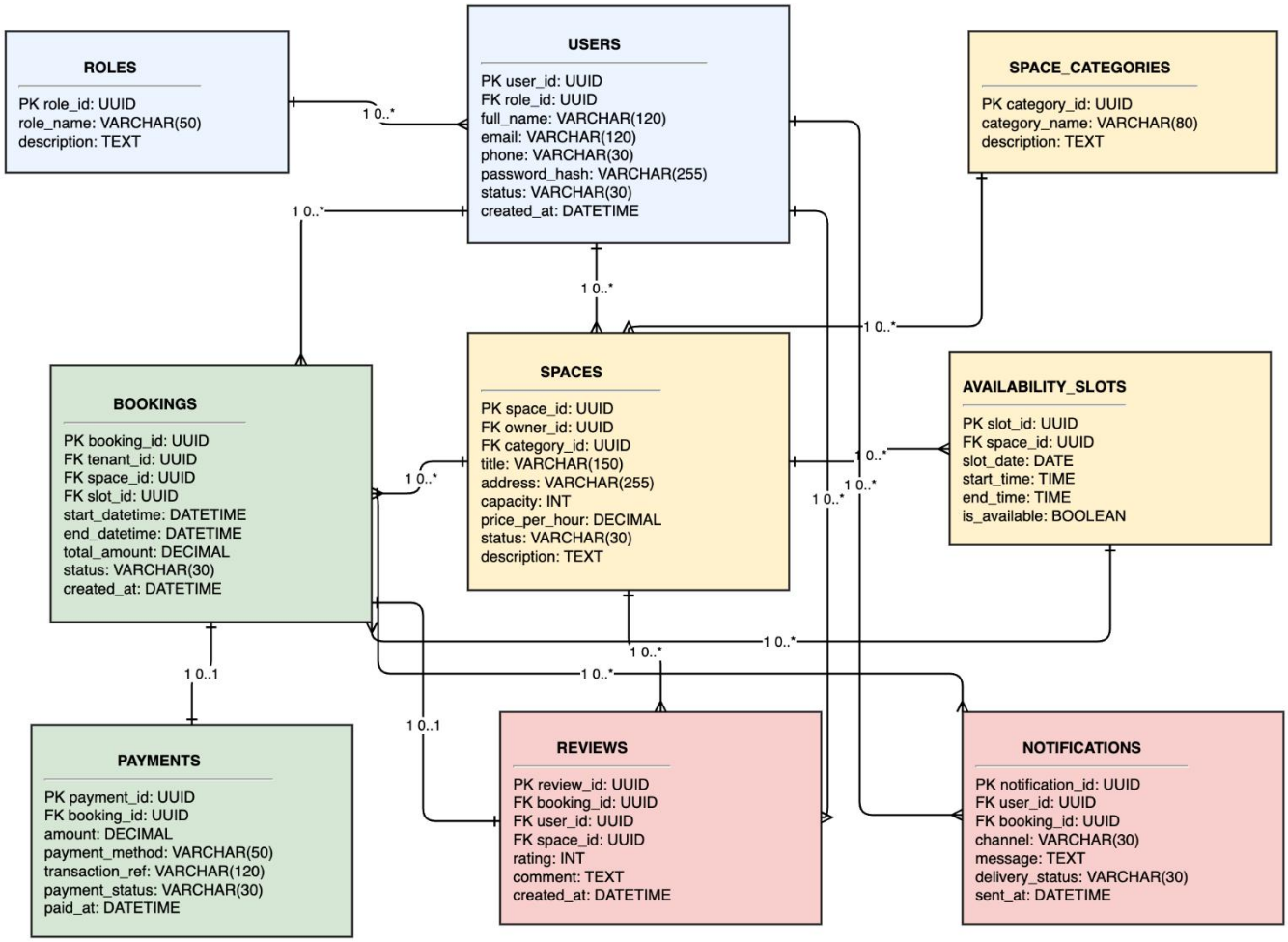


*Рисунок 2.4 – Логічна модель бази даних веб-платформи бронювання оренди приміщень*

Основою структури є таблиця USERS, яка зберігає облікові дані користувачів і пов’язана з таблицею ROLES. Це дозволяє реалізувати розмежування прав доступу між орендарем, власником приміщення та адміністратором. Таблиця SPACES

містить інформацію про об’єкти оренди та пов’язана з користувачем-власником і категорією приміщення. Таблиця AVAILABILITY_SLOTS використовується для збереження часових інтервалів доступності конкретного приміщення, що дає змогу уникати дублювання бронювань на один і той самий період.

Таблиця BOOKINGS є центральною у процесі бронювання, оскільки поєднує користувача-орендаря, приміщення та вибраний часовий слот. Таблиця PAYMENTS зберігає результат платіжної операції, пов’язаної з конкретним бронюванням. Таблиці REVIEWS і NOTIFICATIONS використовуються для збереження відгуків користувачів та службових повідомлень про зміну статусів бронювання.

Для узагальнення структури логічної моделі у таблиці 2.4 наведено основні сутності бази даних та їх призначення.

Таблиця 2.4 – Основні сутності логічної моделі бази даних

| Сутність | Призначення |
|---|---|
| ROLES | Зберігання ролей користувачів системи |
| USERS | Зберігання облікових даних користувачів і зв’язку з роллю |
| SPACE_CATEGORIES | Класифікація приміщень за типами |
| SPACES | Зберігання інформації про приміщення, їх характеристики, ціну та власника |
| AVAILABILITY_SLOTS | Зберігання часових інтервалів доступності приміщень |
| BOOKINGS | Зберігання заявок на бронювання та їх статусів |
| PAYMENTS | Збереження інформації про оплату бронювання |
| REVIEWS | Зберігання оцінок і відгуків користувачів |
| NOTIFICATIONS | Збереження повідомлень про події системи |

Нормалізація структури бази даних до третьої нормальної форми забезпечується тим, що кожна таблиця описує окрему сутність предметної області, усі неключові атрибути залежать від первинного ключа, а транзитивні залежності винесені в окремі таблиці. Наприклад, ролі користувачів зберігаються окремо в таблиці ROLES, категорії приміщень — у таблиці SPACE_CATEGORIES, а інформація про оплату винесена в таблицю PAYMENTS і пов’язана з бронюванням

через зовнішній ключ. Це зменшує дублювання даних і підвищує узгодженість інформації під час оновлення записів.

Запропонована логічна модель забезпечує структуроване збереження даних, підтримку зв’язків між основними об’єктами системи та можливість формування аналітичних запитів щодо бронювань, завантаженості приміщень, платежів і активності користувачів. Така структура є достатньою для реалізації основних функцій веб-платформи та може бути розширена у подальших версіях системи без порушення цілісності бази даних.

## 2.4 Діаграма класів і кооперації веб-платформи бронювання оренди приміщень

На етапі проєктування веб-платформи бронювання оренди приміщень важливим є формалізований опис структури програмного забезпечення та взаємодії його компонентів. Для цього використано UML-діаграму класів і діаграми кооперації, які дозволяють узгоджено представити статичну модель даних і динаміку обміну повідомленнями між об’єктами системи.

На рисунку 2.5 наведено діаграму класів веб-платформи бронювання оренди приміщень, яка відображає основні сутності предметної області, їх атрибути, операції та зв’язки між ними.

Базовим класом системи є User, який містить атрибути ідентифікації та автентифікації користувача. Від нього успадковуються класи Tenant, Owner та Administrator, що відповідають ролям орендаря, власника приміщення та адміністратора. Такий підхід реалізує механізм узагальнення/спеціалізації та забезпечує розмежування функціональності залежно від ролі користувача.

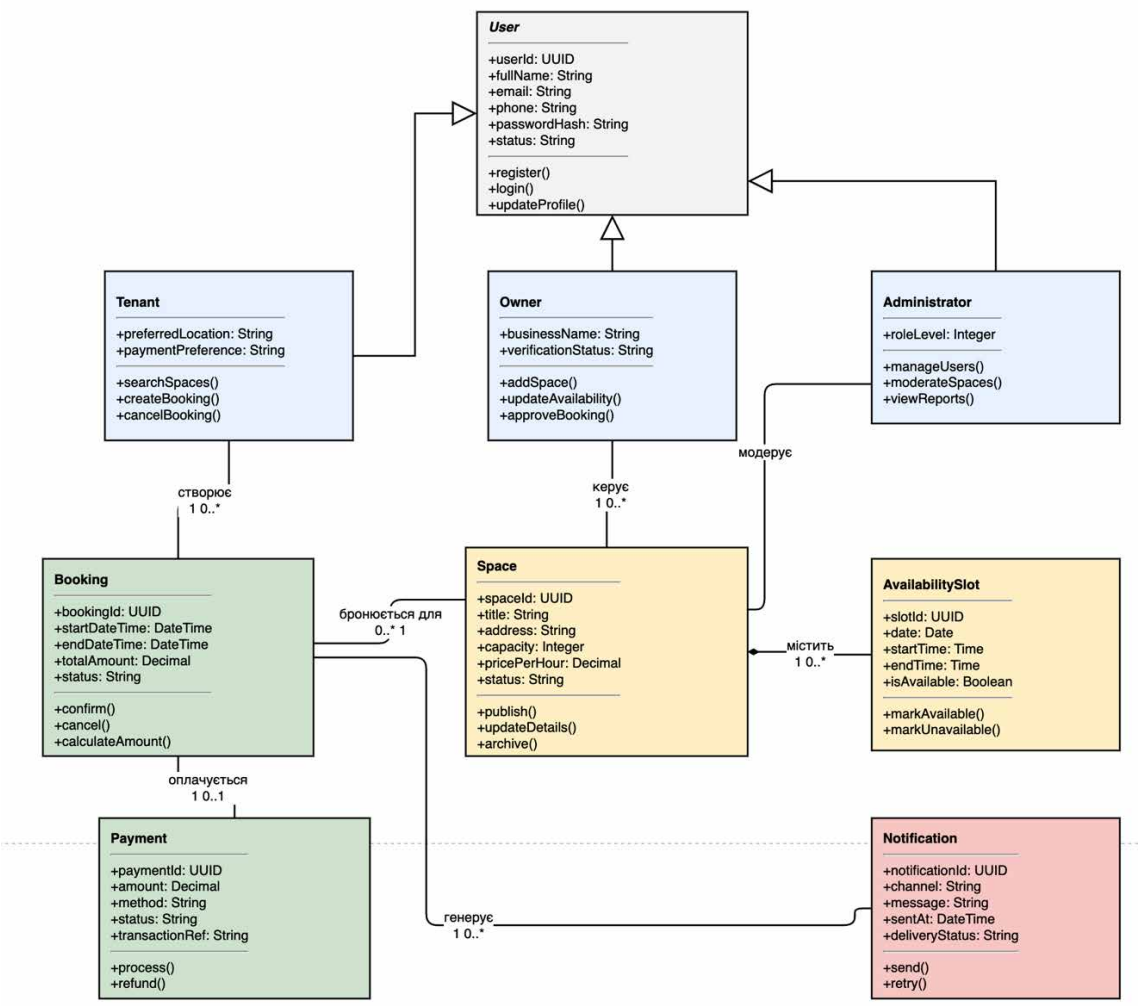


*Рисунок 2.5 – UML-діаграма класів веб-платформи бронювання оренди приміщень*

Клас Tenant реалізує операції пошуку приміщень, створення та скасування бронювання. Клас Owner відповідає за додавання приміщень, керування їх доступністю та підтвердження заявок. Клас Administrator виконує функції керування користувачами, модерації об’єктів і перегляду звітної інформації.

Центральними доменними класами є Space, Booking, AvailabilitySlot, Payment і Notification. Клас Space описує приміщення, включаючи його характеристики, місткість і ціну. Він пов’язаний із класом Owner та агрегує AvailabilitySlot, що представляє часові інтервали доступності. Такий зв’язок забезпечує можливість керування доступністю приміщення без дублювання інформації.

Клас Booking реалізує процес бронювання і пов’язує орендаря, приміщення та часовий інтервал. У ньому визначено операції підтвердження, скасування та розрахунку вартості. Клас Payment відповідає за оброблення фінансових операцій і

містить методи виконання платежу та повернення коштів. Клас Notification використовується для формування та надсилання повідомлень користувачам про зміни стану бронювання.

Побудована діаграма класів забезпечує чітке розмежування відповідальностей між об’єктами системи, відповідає вимогам модульності та дозволяє реалізувати мікросервісну архітектуру, у якій кожен клас або група класів відповідає окремому сервісу.

Для відображення динаміки взаємодії між об’єктами системи розроблено діаграми кооперації, які демонструють послідовність обміну повідомленнями в межах типових сценаріїв використання.

На рисунку 2.6 подано діаграму кооперації процесу пошуку приміщень.

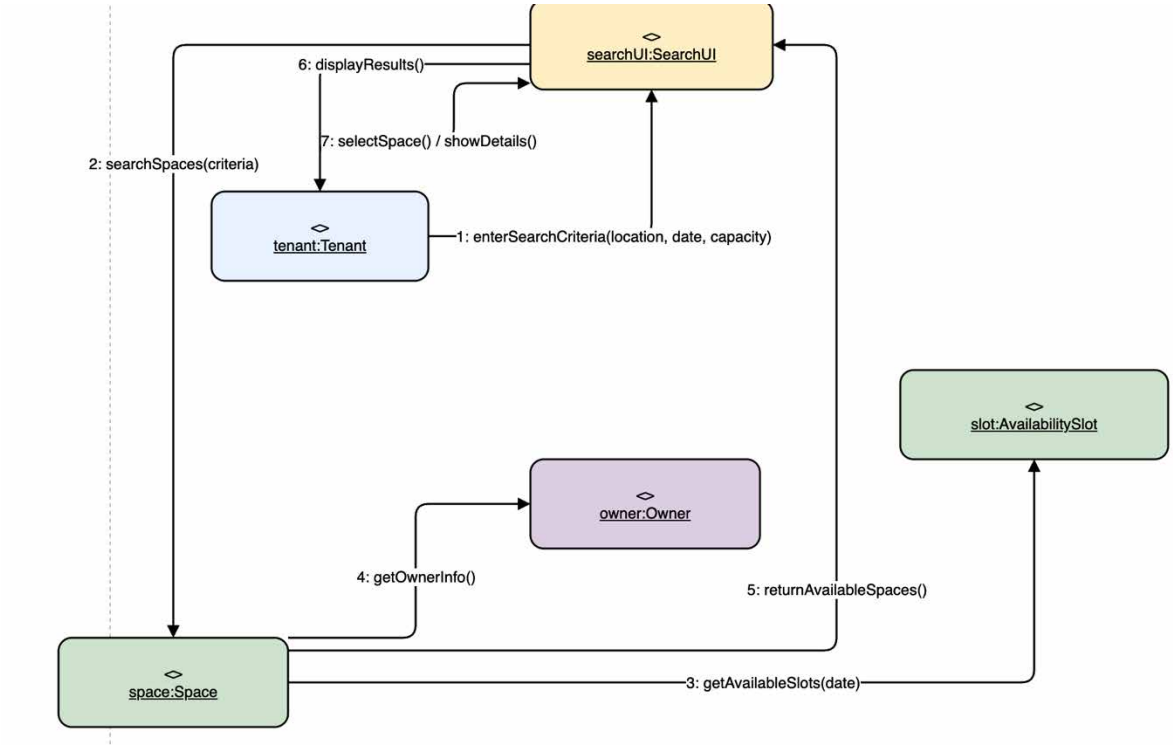


*Рисунок 2.6 – Діаграма кооперації процесу пошуку приміщень*

У даному сценарії орендар формує параметри пошуку через інтерфейс, після чого об’єкт Space виконує запит доступних приміщень і отримує інформацію про доступні часові слоти. Далі результати передаються у SearchUI для відображення

користувачу. Така взаємодія забезпечує розділення логіки пошуку та представлення даних.

На рисунку 2.7 наведено діаграму кооперації процесу створення бронювання.

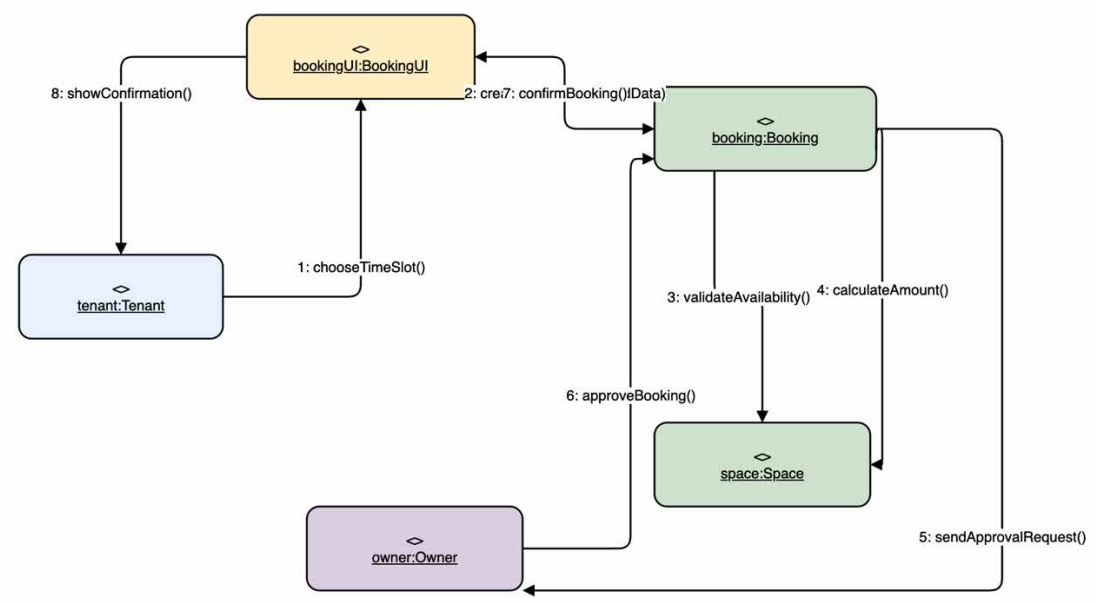


*Рисунок 2.7 – Діаграма кооперації процесу бронювання приміщення*

Сценарій починається з вибору орендарем часового слота, після чого формується об’єкт Booking. Далі виконується перевірка доступності приміщення та розрахунок вартості. Після цього запит передається власнику для підтвердження. Така модель відображає повний цикл створення бронювання з урахуванням перевірок і узгодження між сторонами.

На рисунку 2.8 подано діаграму кооперації процесу оброблення платежу та надсилання сповіщень.

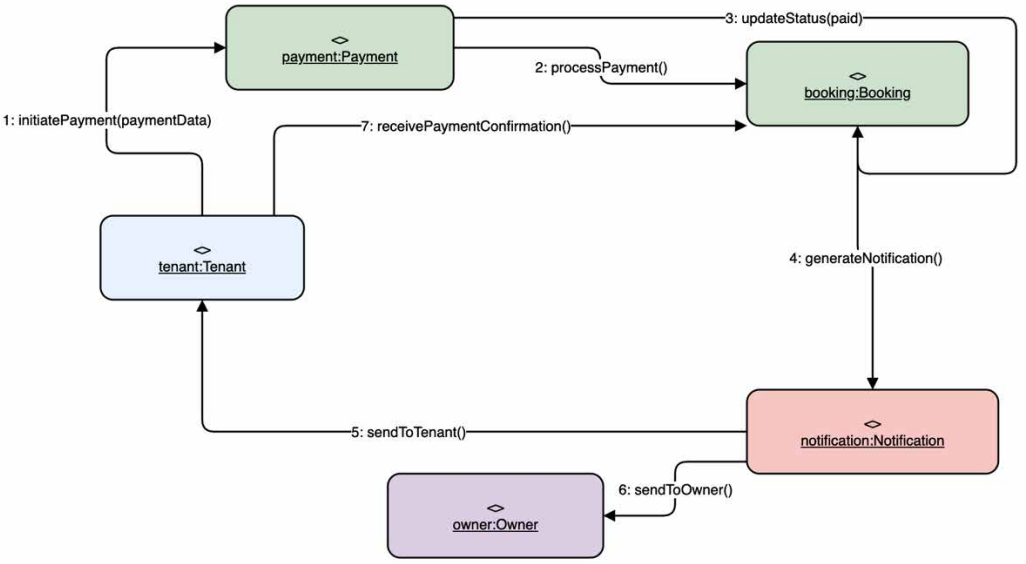


*Рисунок 2.8 – Діаграма кооперації процесу оплати та сповіщення*

У цьому сценарії орендар ініціює платіж, який обробляється класом Payment. Після успішного виконання транзакції статус бронювання оновлюється, а система формує повідомлення через клас Notification. Сповіщення надсилаються як орендарю, так і власнику приміщення. Така організація забезпечує узгодженість між фінансовими операціями та станом бронювання.

Для узагальнення відповідності UML-моделей функціональним вимогам системи у таблиці 2.5 наведено їх взаємозв’язок.

Таблиця 2.5 – Відповідність UML-діаграм функціональним вимогам системи

| Функціональна вимога | UML-артефакт |
|---|---|
| Реєстрація та автентифікація | Діаграма класів |
| Пошук приміщень | Кооперація пошуку |
| Перевірка доступності | Діаграма класів, кооперації |
| Створення бронювання | Кооперація бронювання |
| Оброблення платежу | Кооперація оплати |
| Надсилання сповіщень | Кооперація оплати |
| Керування приміщеннями | Діаграма класів |

Використання діаграми класів і діаграм кооперації дозволило сформувати цілісну модель веб-платформи бронювання оренди приміщень. Отримані UML-артефакти відображають як структуру системи, так і логіку взаємодії її компонентів,

що створює основу для подальшого проєктування архітектури та реалізації програмного забезпечення.

## 2.5 Діаграма пакетів веб-платформи бронювання оренди приміщень

З метою формалізації логічної структури програмного забезпечення веб-платформи бронювання оренди приміщень використано UML-діаграму пакетів. Даний тип діаграм дозволяє узагальнено представити організацію програмних модулів, їх групування за функціональним призначенням і залежності між ними. Це є критично важливим для проєктування масштабованої системи на основі мікросервісної архітектури.

На рисунку 2.10 наведено діаграму пакетів розроблюваної системи, побудовану відповідно до принципів багаторівневої архітектури та результатів аналізу вимог.

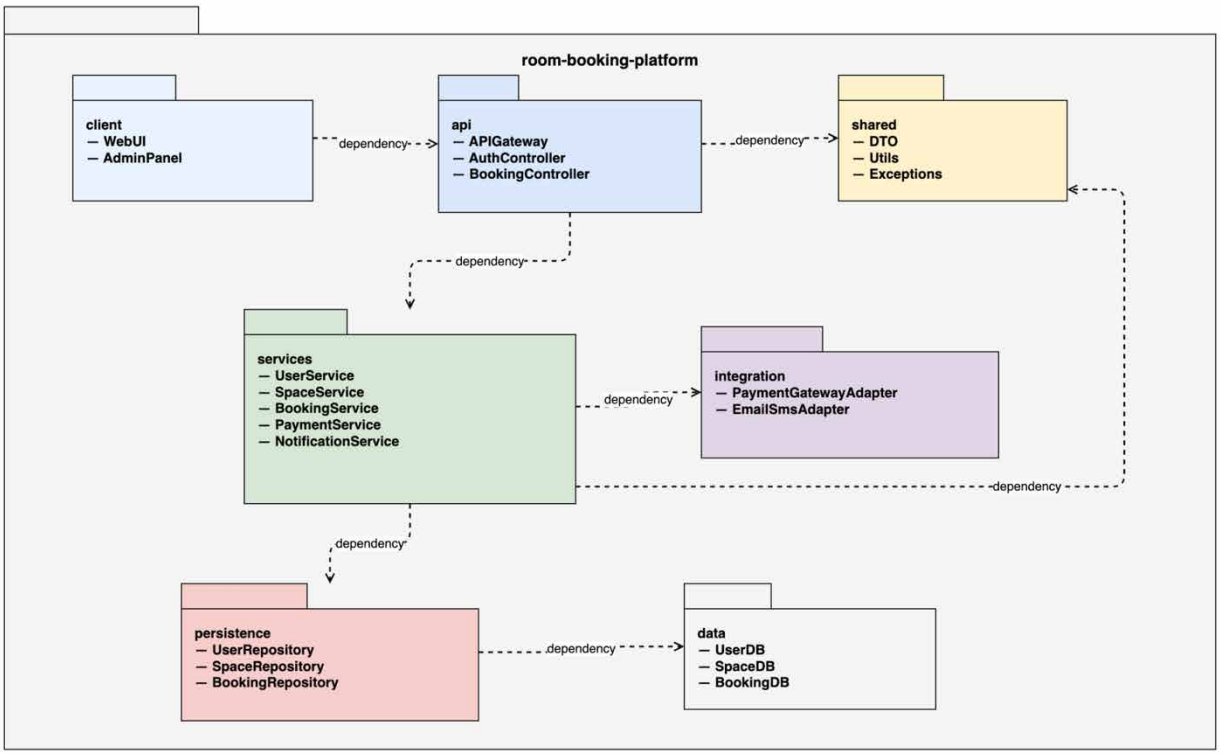


*Рисунок 2.10 – UML-діаграма пакетів веб-платформи бронювання оренди приміщень*

Запропонована структура передбачає поділ системи на логічні пакети: client, api, services, persistence, data, integration та shared. Такий підхід забезпечує розмежування відповідальностей між компонентами, зменшення зв’язаності та можливість незалежного розвитку окремих частин системи.

Пакет client містить модулі користувацького інтерфейсу, зокрема WebUI та AdminPanel. Він забезпечує взаємодію користувачів із системою (орендарів, власників приміщень та адміністраторів) і не містить бізнес-логіки. Взаємодія з серверною частиною здійснюється через API.

Пакет api виконує роль точки входу до системи та реалізує контролери (AuthController, BookingController) і API Gateway. Він приймає запити від клієнта, виконує базову валідацію, автентифікацію та передає їх до відповідних сервісів. Такий підхід дозволяє централізувати доступ до бізнес-логіки та спростити контроль безпеки.

Пакет services є центральним елементом архітектури та містить основні бізнес-сервіси: UserService, SpaceService, BookingService, PaymentService і NotificationService. Кожен сервіс відповідає за окрему функціональну підсистему: керування користувачами, оброблення даних про приміщення, виконання бронювання, оброблення платежів і формування сповіщень. Така декомпозиція відповідає принципам мікросервісної архітектури та дозволяє незалежно масштабувати окремі компоненти.

Пакет persistence реалізує доступ до даних через репозиторії (UserRepository, SpaceRepository, BookingRepository). Він виступає проміжним шаром між бізнес-логікою та фізичними сховищами даних, що забезпечує незалежність програмного коду від конкретної СУБД і спрощує зміну або масштабування системи зберігання.

Пакет data представляє фізичний рівень зберігання інформації (UserDB, SpaceDB, BookingDB). Він відповідає за організацію баз даних і збереження інформації про користувачів, приміщення, бронювання та пов’язані сутності.

Пакет integration призначений для взаємодії із зовнішніми сервісами. До його складу входять адаптери платіжного шлюзу та сервісів повідомлень (PaymentGatewayAdapter, EmailSmsAdapter). Винесення інтеграційної логіки в

окремий пакет дозволяє ізолювати зовнішні залежності та зменшити їх вплив на внутрішню архітектуру системи.

Пакет shared містить спільні компоненти, такі як DTO, утиліти та обробники винятків. Він використовується всіма іншими пакетами для забезпечення уніфікованого обміну даними та оброблення помилок.

Залежності між пакетами побудовані за принципом однонаправленого доступу: клієнт звертається до API, API — до сервісів, сервіси — до шару збереження та інтеграції. Така структура забезпечує слабку зв’язаність компонентів і підвищує керованість системи.

Для узагальнення функціонального призначення пакетів у таблиці 2.5 наведено їх відповідність ролям у системі.

Таблиця 2.5 – Функціональне призначення пакетів веб-платформи

| Пакет | Призначення |
|---|---|
| client | Реалізація користувацьких інтерфейсів (WebUI, AdminPanel) |
| api | Оброблення запитів, маршрутизація та контроль доступу |
| services | Реалізація бізнес-логіки системи |
| persistence | Доступ до бази даних через репозиторії |
| data | Фізичне збереження даних |
| integration | Взаємодія із зовнішніми сервісами |
| shared | Спільні компоненти (DTO, утиліти, обробка помилок) |

Побудована UML-діаграма пакетів відображає структуровану організацію програмного забезпечення веб-платформи бронювання оренди приміщень. Запропонований підхід забезпечує відповідність вимогам модульності, масштабованості та підтримуваності системи, а також створює основу для подальшої реалізації мікросервісної архітектури.

**Висновки до другого розділу**

У другому розділі виконано проєктування програмного забезпечення веб-платформи бронювання оренди приміщень із використанням об’єктно-орієнтованого підходу та UML-моделювання. Основну увагу приділено формалізації структури системи, визначенню взаємодії компонентів і побудові логічної моделі програмного забезпечення відповідно до сформованих вимог.

У межах діаграми класів визначено ключові сутності предметної області: користувачі, приміщення, часові слоти, бронювання, платежі та сповіщення. Встановлено їх атрибути, операції та зв’язки, що забезпечує узгодженість між структурою даних і бізнес-логікою системи. Отримана модель дозволяє реалізувати повний цикл бронювання з урахуванням перевірки доступності, розрахунку вартості та оброблення платежів.

Розроблені діаграми кооперації відображають типові сценарії функціонування системи: пошук приміщень, створення бронювання, оброблення платежу та надсилання сповіщень. Це дозволило формалізувати послідовність обміну повідомленнями між об’єктами, визначити точки інтеграції з зовнішніми сервісами та забезпечити узгодженість між функціональними вимогами і поведінкою системи.

Діаграма компонентів забезпечила представлення архітектури системи на концептуальному рівні. Виділення окремих сервісів (керування користувачами, приміщеннями, бронюванням, оплатою та сповіщеннями) відповідає принципам мікросервісної архітектури, що забезпечує масштабованість, відмовостійкість і можливість незалежного розвитку компонентів.

У межах діаграми пакетів узагальнено логічну організацію програмних модулів. Виділення пакетів client, api, services, persistence, data, integration і shared забезпечує чітке розмежування відповідальностей між рівнями системи, зменшення зв’язаності компонентів і підвищення супроводжуваності програмного забезпечення. Така структура відповідає вимогам до сучасних веб-платформ і створює основу для ефективної реалізації мікросервісного підходу.

Отже, результати другого розділу формують цілісну модель програмного забезпечення веб-платформи бронювання оренди приміщень. Побудовані UML-артефакти визначають структуру системи, логіку взаємодії компонентів і принципи організації архітектури, що є достатнім підґрунтям для переходу до етапу реалізації, вибору технологічного стеку та розроблення програмних модулів у наступному розділі.

# РОЗДІЛ: 3 РОЗРОБКА І ТЕСТУВАННЯ ПРОГРАМНО ЗАБЕЗПЕЧЕННЯ

## 3.1 Вибір мови та середовища програмування для реалізації веб-платформи

Вибір технологічного стеку є критичним етапом реалізації веб-платформи бронювання оренди приміщень, оскільки визначає продуктивність системи, масштабованість, відмовостійкість та можливість подальшого розвитку. З урахуванням вимог до мікросервісної архітектури, інтеграції із зовнішніми сервісами та підтримки великої кількості користувачів було обрано сучасний стек веб-технологій: TypeScript, Node.js (NestJS), Next.js, PostgreSQL, Redis, Docker та середовище розробки Visual Studio Code.

Обрані технології забезпечують розподілену обробку даних, підтримку REST API, масштабування окремих сервісів і зручність розробки. Узагальнення характеристик використаних технологій наведено в таблиці 3.1.

Таблиця 3.1 – Обґрунтування вибору технологій для реалізації веб-платформи

| Технологія | Призначення | Переваги використання |
|---|---|---|
| TypeScript | Основна мова програмування | Статична типізація, підвищена надійність коду, підтримка ООП |
| Node.js (NestJS) | Серверна частина, мікросервіси | Висока продуктивність, модульність, підтримка REST і мікросервісної архітектури |
| Next.js | Клієнтська частина (Web UI) | SSR/CSR, оптимізація продуктивності, зручна маршрутизація |
| PostgreSQL | Реляційна база даних | Надійність, підтримка складних запитів, транзакційність |
| Redis | Кешування та сесії | Висока швидкість доступу, зменшення навантаження на БД |
| Docker | Контейнеризація | Ізоляція сервісів, зручність розгортання та масштабування |
| REST API | Взаємодія компонентів | Стандартизований обмін даними між сервісами |
| Visual Studio Code | Середовище розробки | Розширення, інтеграція з Git, зручність роботи з TypeScript |

Аналіз наведеного стеку показує, що використання TypeScript дозволяє підвищити якість коду за рахунок статичної типізації та зменшити кількість помилок на етапі розробки. Платформа Node.js у поєднанні з фреймворком NestJS забезпечує побудову масштабованої серверної частини з чітким поділом на модулі та сервіси, що відповідає принципам мікросервісної архітектури.

Застосування Next.js дозволяє реалізувати клієнтську частину з підтримкою серверного рендерингу, що покращує швидкодію та SEO-оптимізацію веб-платформи. Використання PostgreSQL як основної СУБД забезпечує надійне збереження структурованих даних і підтримку транзакцій при виконанні операцій бронювання та оплати.

Додатково використання Redis дозволяє реалізувати кешування запитів і керування сесіями користувачів, що зменшує навантаження на базу даних і підвищує продуктивність системи. Контейнеризація за допомогою Docker забезпечує ізольоване розгортання кожного сервісу, що спрощує масштабування та підтримку системи в умовах реального використання.

Обраний стек технологій повністю відповідає функціональним і нефункціональним вимогам веб-платформи бронювання оренди приміщень. Його використання забезпечує реалізацію масштабованої, відмовостійкої та продуктивної системи з можливістю подальшого розширення функціональності та інтеграції нових сервісів.

## 3.2 Структурна схема реалізації програмної системи

Структурна схема реалізації веб-платформи бронювання оренди приміщень визначає загальну архітектуру системи, взаємодію між її компонентами та організацію потоків даних. Побудова такої схеми дозволяє формалізувати розподіл функціональності між мікросервісами, визначити точки інтеграції із зовнішніми сервісами та забезпечити узгодженість між програмною реалізацією і вимогами, сформованими у попередніх розділах.

На рисунку 3.1 наведено структурну схему реалізації веб-платформи, яка відображає поділ системи на клієнтський рівень, рівень мікросервісів, рівень зберігання даних та зовнішні сервіси.

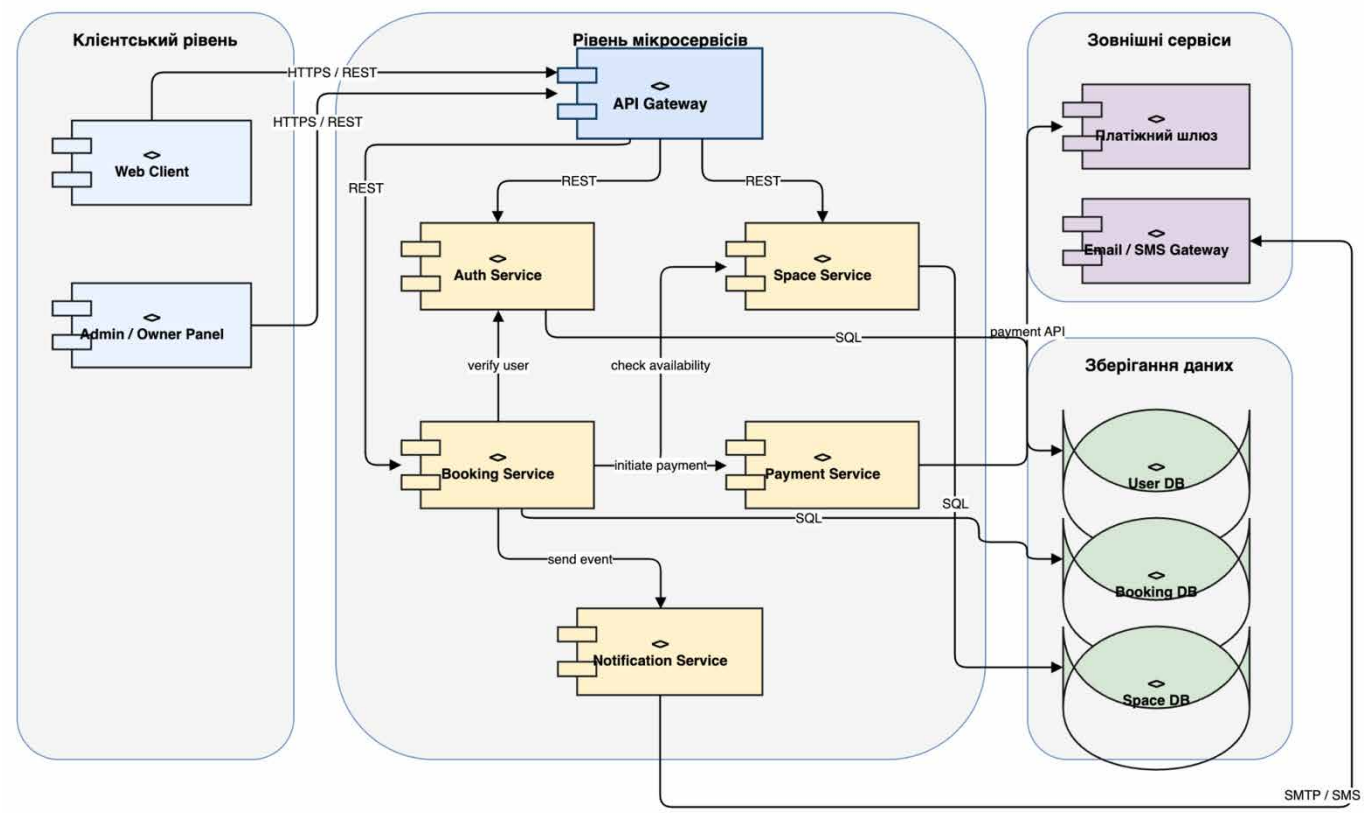


*Рисунок 3.1 – Структурна схема реалізації веб-платформи бронювання оренди приміщень*

Представлена схема демонструє, що взаємодія користувачів із системою здійснюється через клієнтський рівень, який включає Web Client та панель адміністратора/власника. Обмін даними між клієнтом і серверною частиною реалізується за протоколом HTTPS із використанням REST API, що забезпечує стандартизовану взаємодію компонентів.

Центральним елементом архітектури є API Gateway, який виконує маршрутизацію запитів, оброблення автентифікації та передачу даних до відповідних мікросервісів. Такий підхід дозволяє централізувати доступ до системи та зменшити зв’язаність між клієнтом і внутрішніми сервісами.

Рівень мікросервісів представлений набором незалежних компонентів: Auth Service, Space Service, Booking Service, Payment Service та Notification Service. Кожен із них виконує окрему функцію:

1) Auth Service - автентифікація та перевірка користувачів;
2) Space Service - управління даними про приміщення та їх доступність;
3) Booking Service - оброблення логіки бронювання;

4) Payment Service - ініціація та оброблення платежів;

5) Notification Service - формування та надсилання повідомлень.

Взаємодія між сервісами реалізується через REST-запити та внутрішні події, що дозволяє забезпечити асинхронність обробки та підвищити відмовостійкість системи.

Рівень зберігання даних включає окремі бази даних для користувачів, бронювань і приміщень (User DB, Booking DB, Space DB). Такий підхід відповідає принципу ізоляції даних у мікросервісній архітектурі та дозволяє масштабувати кожен компонент незалежно.

Зовнішні сервіси представлені платіжним шлюзом і сервісами сповіщень (Email/SMS Gateway). Інтеграція з ними здійснюється через API, що забезпечує виконання фінансових операцій і інформування користувачів про статуси бронювання.

Для узагальнення функціонального призначення основних компонентів системи у таблиці 3.2 наведено їх ролі в архітектурі.

Таблиця 3.2 – Основні компоненти структурної схеми та їх призначення

| Компонент | Призначення |
|---|---|
| Web Client | Інтерфейс користувача для пошуку та бронювання |
| Admin / Owner Panel | Керування приміщеннями та бронюваннями |
| API Gateway | Маршрутизація запитів і контроль доступу |
| Auth Service | Автентифікація користувачів |
| Space Service | Керування приміщеннями та доступністю |
| Booking Service | Оброблення бронювань |
| Payment Service | Оброблення платежів |
| Notification Service | Надсилання сповіщень |
| User DB | Збереження даних користувачів |
| Booking DB | Збереження даних бронювань |
| Space DB | Збереження даних про приміщення |
| Payment Gateway | Виконання платіжних операцій |
| Email/SMS Gateway | Надсилання повідомлень |

Аналіз структурної схеми показує, що використання мікросервісної архітектури забезпечує модульність системи, можливість незалежного масштабування сервісів і підвищення відмовостійкості. Розподіл функціональності між компонентами дозволяє ізолювати критичні частини системи (зокрема оброблення платежів і бронювання) та спростити їх супровід.

Отже, запропонована структурна схема реалізації повністю відповідає функціональним і нефункціональним вимогам веб-платформи бронювання оренди приміщень і створює технічне підґрунтя для подальшої реалізації програмних модулів, налаштування інфраструктури та проведення тестування системи.

## 3.3 Розробка програмних компонентів веб-платформи

### 3.3.1 Розробка компонента реєстрації та автентифікації користувача

Компонент реєстрації та автентифікації забезпечує вхід користувача до веб-платформи та визначення його ролі в системі. У межах даного компонента обробляються облікові дані орендаря, власника приміщення або адміністратора, виконується перевірка введеної інформації та створюється активна сесія користувача. Компонент взаємодіє з Auth Service і User DB, що дозволяє централізовано керувати доступом до функцій платформи.

На рисунку 3.2 наведено блок-схему алгоритму реєстрації та автентифікації користувача.

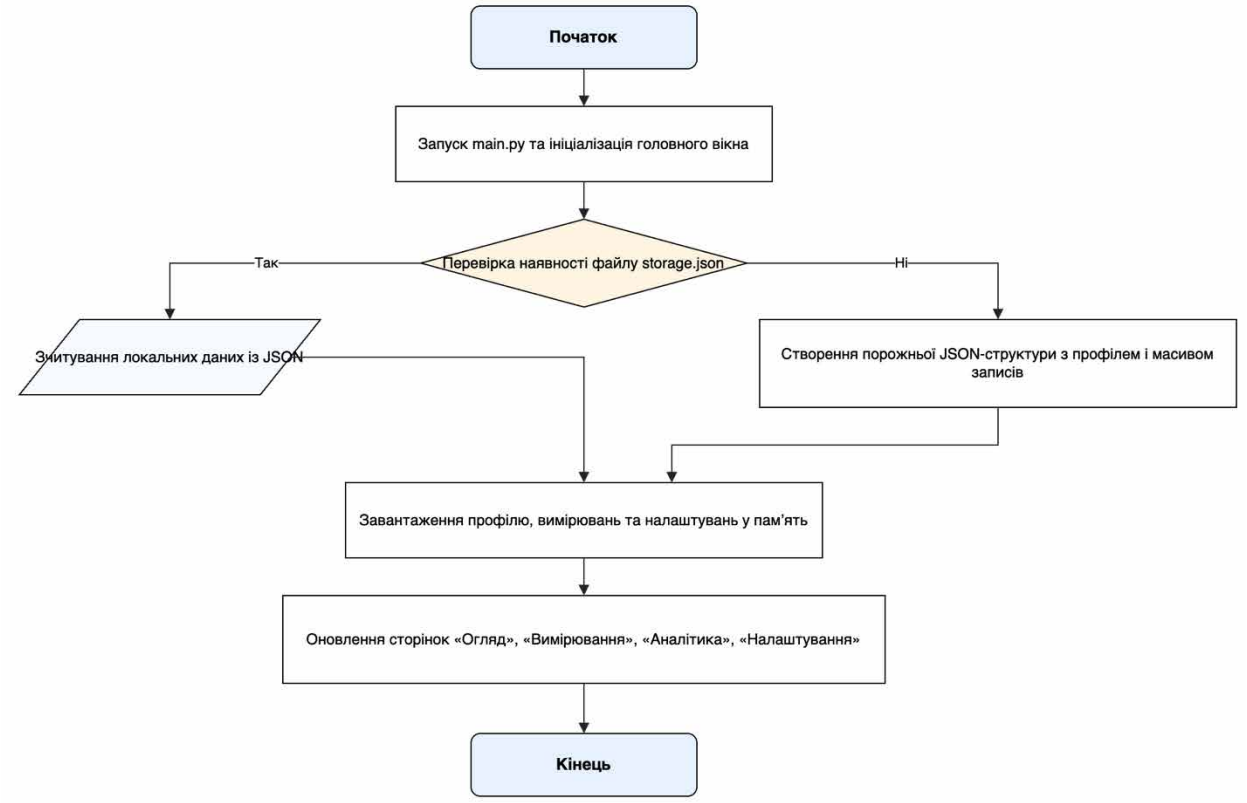


*Рисунок 3.2 – Блок-схема алгоритму реєстрації та автентифікації користувача*

Алгоритм передбачає введення облікових даних, перевірку їх коректності, звернення до сервісу автентифікації та визначення ролі користувача. Якщо дані підтверджено, система відкриває відповідний інтерфейс: для орендаря, власника або адміністратора. У разі помилки користувач отримує повідомлення про некоректні дані.

### 3.3.2 Розробка компонента пошуку та перегляду приміщень

Компонент пошуку приміщень реалізує один із базових сценаріїв роботи системи. Він забезпечує введення параметрів пошуку, фільтрацію доступних об’єктів і відображення результатів у вебінтерфейсі. До параметрів пошуку належать дата, час, категорія приміщення, місткість, ціна та місце розташування.

На рисунку 3.3 наведено блок-схему алгоритму пошуку та перегляду приміщень.

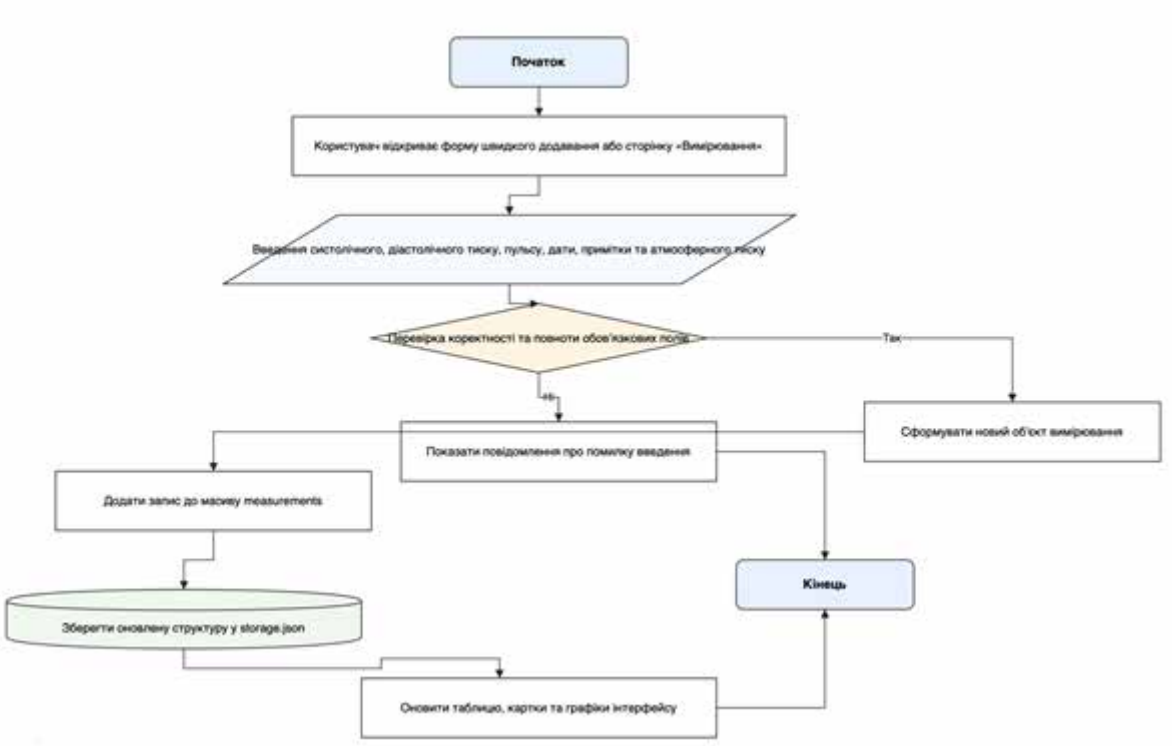


*Рисунок 3.3 – Блок-схема алгоритму пошуку та перегляду приміщень*

Алгоритм починається з введення користувачем критеріїв пошуку. Після цього система передає запит до Space Service, де виконується відбір приміщень відповідно до заданих параметрів. Якщо відповідні об’єкти знайдено, користувачу відображається перелік доступних приміщень. Якщо результатів немає, система формує повідомлення про відсутність доступних варіантів.

### 3.3.3 Розробка компонента перевірки доступності та створення бронювання

Компонент створення бронювання відповідає за перевірку доступності приміщення у вибраний період і формування заявки на бронювання. Його робота є критичною для системи, оскільки саме на цьому етапі необхідно запобігти дублюванню бронювань одного приміщення на один часовий слот.

На рисунку 3.4 наведено блок-схему алгоритму перевірки доступності та створення бронювання.

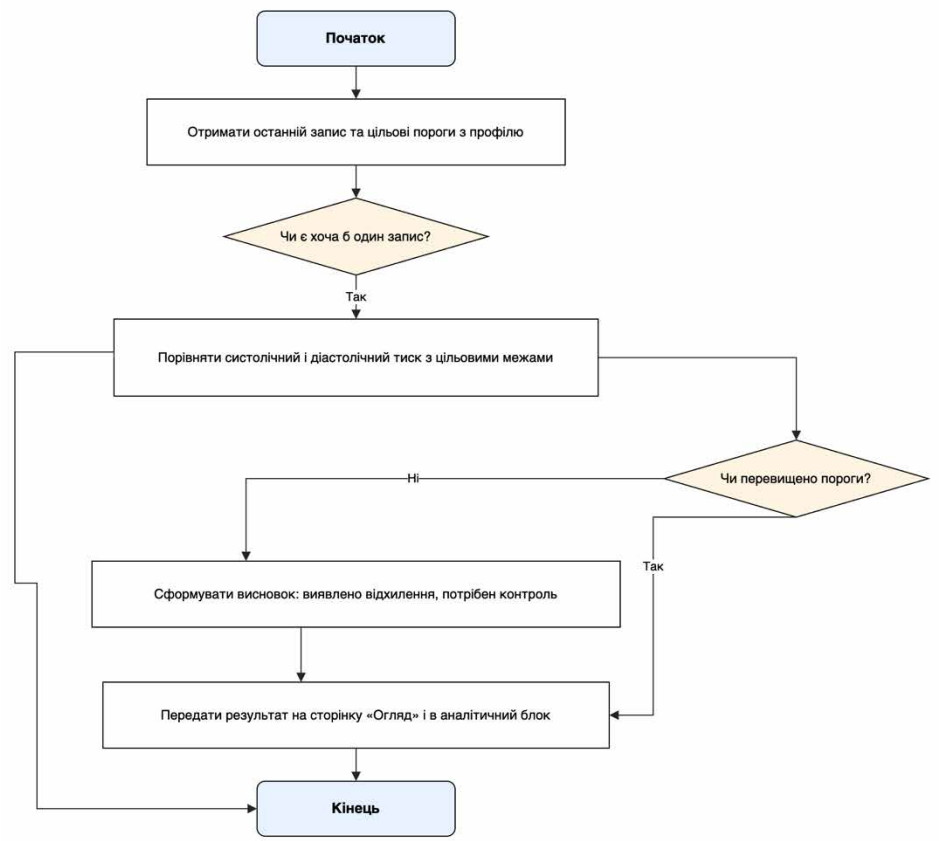


*Рисунок 3.4 – Блок-схема алгоритму перевірки доступності та створення бронювання*

Алгоритм передбачає вибір користувачем приміщення, дати та часу, після чого Booking Service виконує перевірку доступності вибраного слота. Якщо приміщення недоступне, система повідомляє користувача та повертає його до вибору іншого часу. Якщо слот доступний, формується попереднє бронювання зі статусом очікування оплати або підтвердження.

### 3.3.4 Розробка компонента оброблення платежу

Компонент оброблення платежу забезпечує інтеграцію системи з платіжним сервісом і фіксацію результату фінансової операції. Після створення попереднього бронювання система формує платіжний запит, передає його до Payment Service, а далі - до зовнішнього платіжного шлюзу.

На рисунку 3.5 наведено блок-схему алгоритму оброблення платежу.

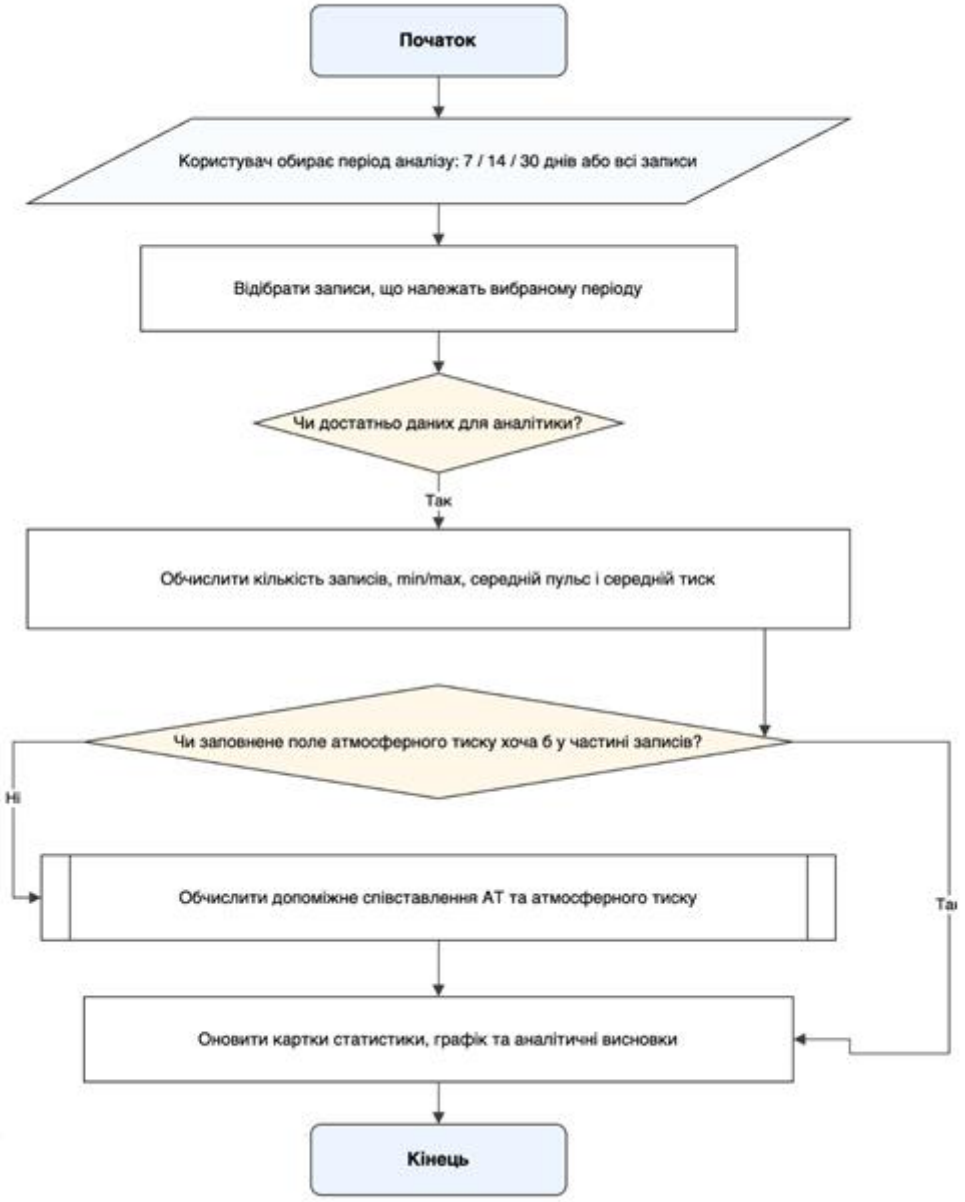


*Рисунок 3.5 – Блок-схема алгоритму оброблення платежу*

Алгоритм включає формування платіжного запиту, передавання суми й ідентифікатора бронювання до платіжного сервісу, отримання результату транзакції та оновлення статусу бронювання. Якщо платіж підтверджено, бронювання переходить у статус активного. Якщо оплату відхилено, попереднє бронювання скасовується або залишається у статусі помилки оплати.

### 3.3.5 Розробка компонента надсилання сповіщень

Компонент надсилання сповіщень призначений для інформування користувачів про зміну стану бронювання. Він активується після створення заявки,

підтвердження оплати, скасування бронювання або зміни його статусу власником приміщення.

На рисунку 3.6 наведено блок-схему алгоритму надсилання сповіщень.

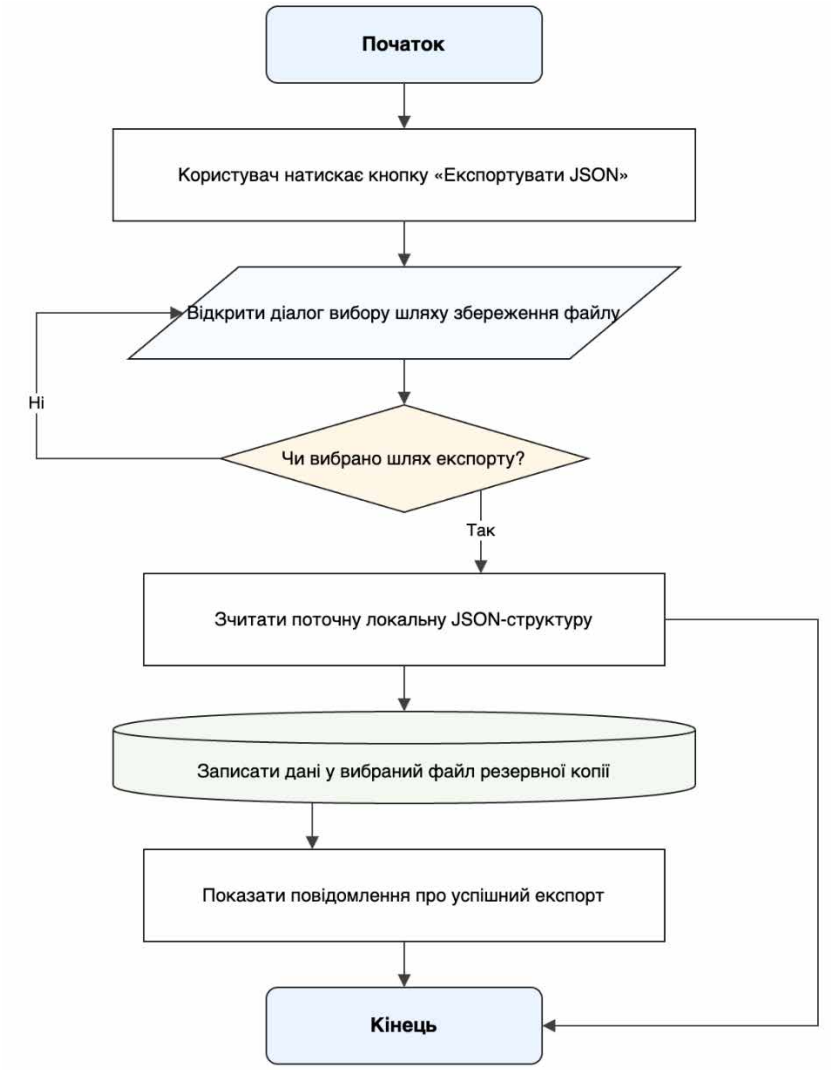


*Рисунок 3.6 – Блок-схема алгоритму надсилання сповіщень*

Алгоритм передбачає отримання події від Booking Service або Payment Service, формування тексту повідомлення, визначення отримувача та каналу доставки. Після цього Notification Service надсилає повідомлення через Email або SMS Gateway і зберігає статус доставки. Такий компонент забезпечує прозорість процесу бронювання та своєчасне інформування орендаря і власника приміщення.

### 3.4 Тестування системи

Тестування веб-платформи бронювання приміщень є завершальним етапом розроблення програмного забезпечення та спрямоване на перевірку коректності реалізації функціональних можливостей, відповідності системи визначеним вимогам і забезпечення стабільності її роботи в умовах реальної експлуатації. У межах даного етапу було виконано функціональне, інтеграційне та інтерфейсне тестування основних модулів системи.

Основну увагу приділено перевірці ключових сценаріїв роботи користувача: пошуку приміщень, оформлення бронювання, здійснення оплати, перегляду статистики та управління власними бронюваннями.

На рисунку 3.7 наведено інтерфейс головної сторінки системи з функцією пошуку приміщень.

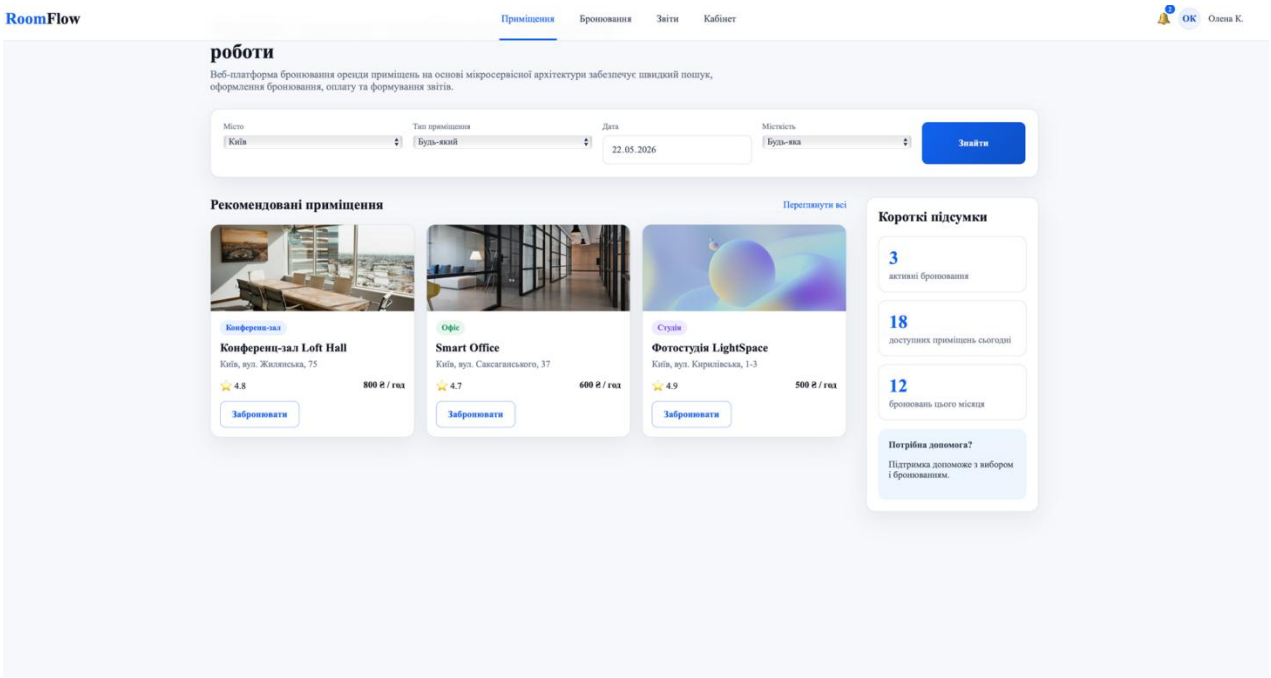


*Рисунок 3.7 – Головна сторінка веб-платформи з модулем пошуку приміщень*

Даний інтерфейс було протестовано на коректність роботи фільтрів пошуку, зокрема вибору міста, типу приміщення, дати та місткості. У результаті тестування встановлено, що система коректно обробляє введені критерії та відображає

релевантні результати у вигляді списку доступних приміщень. Також перевірено відображення рекомендованих об’єктів і статистичних показників, що формуються на основі даних системи.

На рисунку 3.8 представлено інтерфейс оформлення бронювання.

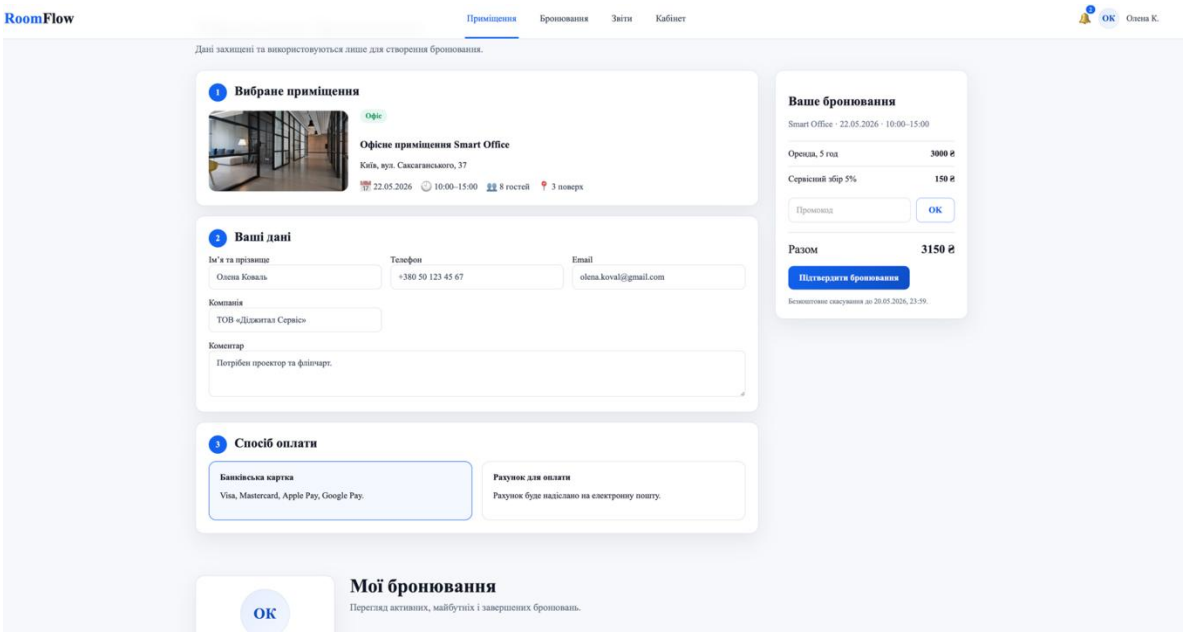


*Рисунок 3.8 – Сторінка створення та підтвердження бронювання*

У межах тестування даного модуля перевірено коректність передачі даних про вибране приміщення, введення персональної інформації користувача та вибору способу оплати. Окремо протестовано розрахунок вартості бронювання, включаючи сервісний збір, що підтвердило правильність обчислення загальної суми. Система забезпечує перевірку введених даних і не допускає створення бронювання у разі їх некоректності.

На рисунку 3.9 наведено інтерфейс формування звітів.

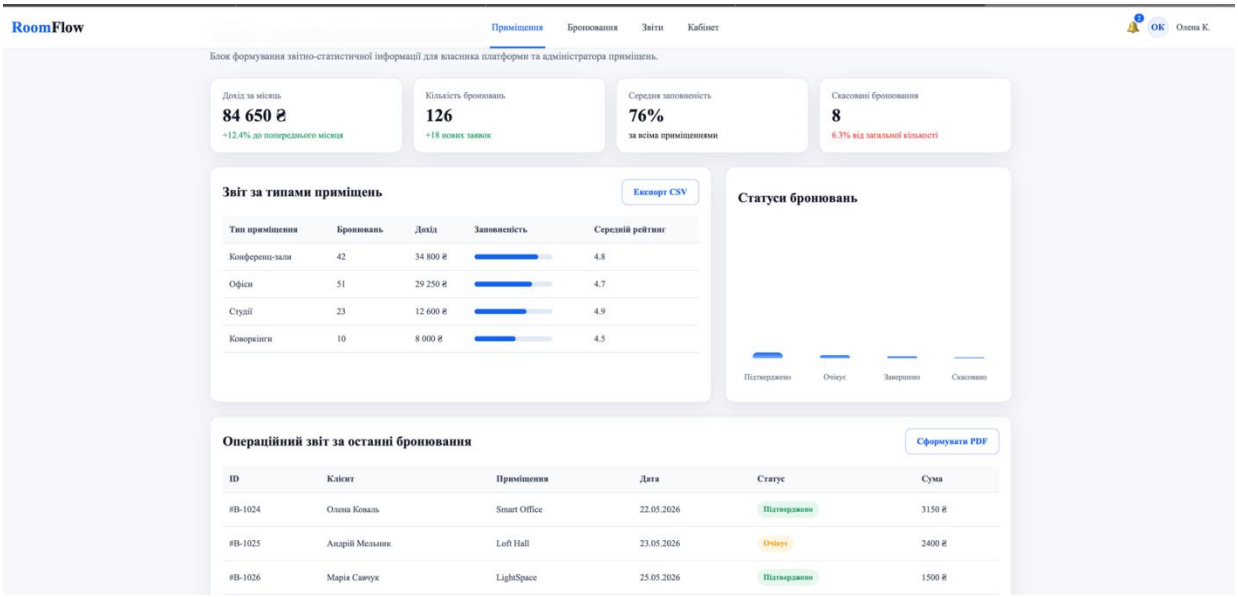


*Рисунок 3.9 – Сторінка аналітики та формування звітів*

Під час тестування аналітичного модуля перевірено правильність обчислення агрегованих показників, зокрема доходу, кількості бронювань, рівня заповненості та частки скасованих бронювань. Встановлено, що система коректно формує статистичні дані на основі записів бази даних, а також забезпечує відображення звітів у зручному для аналізу вигляді.

На рисунку 3.10 наведено інтерфейс перегляду бронювань користувача.

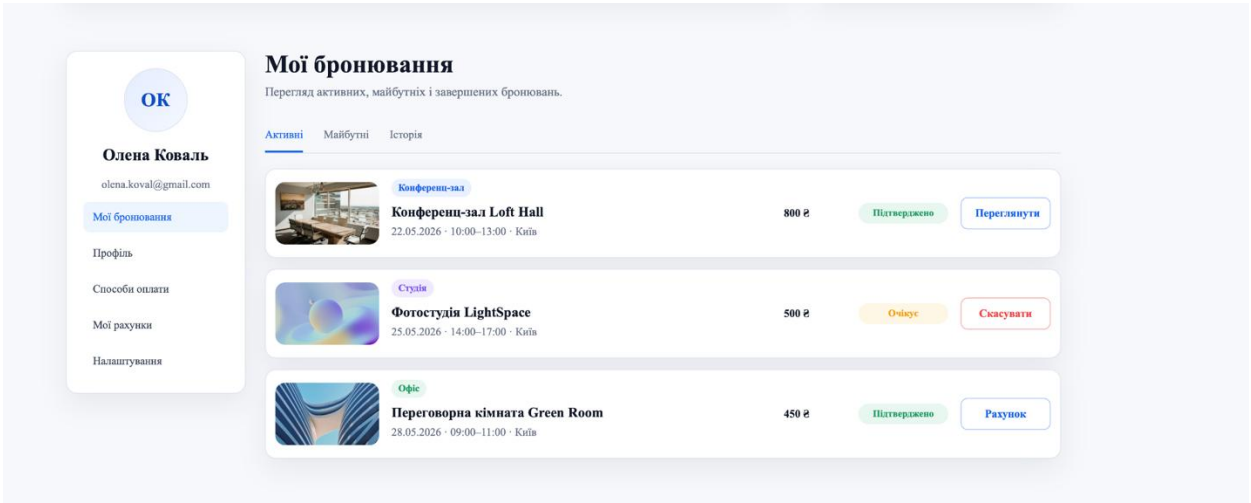


*Рисунок 3.10 – Сторінка «Мої бронювання»*

У процесі тестування перевірено відображення активних, майбутніх і завершених бронювань, а також можливість їх перегляду та скасування. Результати показали, що система коректно оновлює статуси бронювань та забезпечує взаємодію користувача з відповідними функціями без помилок.

Для узагальнення результатів тестування основних функціональних модулів у таблиці 3.2 наведено відповідність перевірених сценаріїв та отриманих результатів.

Таблиця 3.2 – Результати тестування функціональних модулів системи

| Модуль системи | Перевірений сценарій | Результат тестування |
|---|---|---|
| Пошук приміщень | Введення критеріїв та відображення списку | Успішно |
| Бронювання | Створення та підтвердження заявки | Успішно |
| Оплата | Розрахунок вартості та обробка платежу | Успішно |
| Аналітика | Формування статистичних звітів | Успішно |
| Управління бронюваннями | Перегляд та скасування бронювань | Успішно |

Аналіз результатів, наведених у таблиці 3.2, свідчить про те, що всі основні функціональні компоненти системи працюють коректно та відповідають визначеним вимогам. Помилки, що виникали під час тестування, мали незначний характер і були усунені на етапі налагодження.

Отже, проведене тестування підтвердило працездатність веб-платформи бронювання приміщень, коректність реалізації бізнес-логіки та стабільність взаємодії між компонентами системи. Отримані результати дозволяють зробити висновок про готовність програмного забезпечення до подальшої експлуатації та можливого розширення функціональності.

**Висновки до третього розділу**

У третьому розділі виконано проєктування та програмну реалізацію веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури. На основі сформульованих у попередніх розділах вимог обґрунтовано вибір сучасного

стеку технологій, зокрема використання TypeScript та Next.js для клієнтської частини, серверної логіки на базі REST-орієнтованих сервісів, а також застосування реляційної бази даних для збереження інформації. Встановлено, що обраний підхід забезпечує необхідний рівень продуктивності, масштабованості та зручності супроводу програмного забезпечення.

У межах реалізації системи сформовано структурну схему, яка відображає поділ на клієнтський рівень, рівень мікросервісів та рівень зберігання даних із підключенням зовнішніх сервісів. Запропонована архітектура забезпечує чітке розмежування відповідальності між модулями автентифікації, управління приміщеннями, оброблення бронювань, здійснення платежів і надсилання сповіщень. Це дозволяє підвищити гнучкість системи та забезпечити можливість незалежного масштабування окремих компонентів.

Реалізовано основні функціональні модулі платформи, зокрема підсистеми пошуку та фільтрації приміщень, оформлення та управління бронюваннями, оброблення платежів через зовнішні платіжні сервіси, формування аналітичних звітів і систему сповіщень користувачів. Для кожного з ключових процесів розроблено алгоритми у вигляді блок-схем, що формалізують послідовність оброблення даних і забезпечують узгодженість взаємодії між сервісами.

Особливістю реалізації є використання мікросервісного підходу з API Gateway, що забезпечує централізовану точку доступу до системи, а також інтеграцію із зовнішніми платіжними та комунікаційними сервісами. Це дозволяє мінімізувати зв’язаність компонентів і підвищити відмовостійкість системи в умовах зростання навантаження.

У процесі тестування перевірено працездатність основних сценаріїв функціонування платформи, зокрема виконання пошуку приміщень, створення та підтвердження бронювання, оброблення платежів, формування звітів і взаємодію користувача з інтерфейсом. Отримані результати підтвердили коректність реалізації бізнес-логіки, узгодженість роботи мікросервісів та стабільність функціонування системи.

Отже, у третьому розділі досягнуто поставленої мети щодо розроблення та програмної реалізації веб-платформи бронювання приміщень. Створене програмне забезпечення забезпечує ефективний пошук, бронювання, оплату та аналітичне опрацювання даних, що підвищує ефективність використання ресурсів і створює основу для подальшого розширення функціональності та інтеграції з іншими інформаційними системами.

# ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальну науково-практичну задачу розроблення веб-платформи бронювання оренди приміщень на основі мікросервісної архітектури. Актуальність дослідження обумовлена зростанням попиту на гнучкі простори (офіси, конференц-зали, коворкінги), необхідністю автоматизації процесів бронювання та недостатнім рівнем інтеграції і аналітики у більшості існуючих рішень.

У процесі виконання роботи проведено аналіз предметної області ринку оренди приміщень та визначено ключові проблеми: відсутність централізованого управління доступністю ресурсів, складність синхронізації даних, ризик конфліктів бронювання та обмежені можливості аналітичного опрацювання. Дослідження існуючих систем показало їх типові обмеження, що обґрунтувало доцільність розроблення власного програмного рішення з використанням сучасних архітектурних підходів.

На основі аналізу сформульовано функціональні, нефункціональні та безпекові вимоги до системи, що визначили її архітектуру та принципи функціонування. Виконано моделювання предметної області із застосуванням UML-діаграм (прецедентів, послідовності, активності, класів, пакетів), що дозволило формалізувати сценарії взаємодії користувачів із системою та логіку оброблення даних. Розроблено логічну модель бази даних, нормалізовану до третьої нормальної форми, що забезпечує цілісність, узгодженість та ефективність роботи з даними.

У межах роботи обґрунтовано вибір технологічного стеку, що включає використання веб-технологій (TypeScript, Next.js) для клієнтської частини, мікросервісної серверної архітектури з REST-взаємодією, а також реляційної бази даних для збереження інформації. Встановлено, що обраний підхід забезпечує масштабованість, гнучкість і можливість незалежного розвитку окремих компонентів системи.

Реалізовано веб-платформу, яка забезпечує пошук і фільтрацію приміщень, перевірку доступності, створення та управління бронюваннями, оброблення

платежів через зовнішні платіжні сервіси, формування аналітичних звітів і систему сповіщень користувачів. Особливістю розробленого рішення є застосування мікросервісної архітектури з API Gateway, що забезпечує централізовану точку доступу та ефективну взаємодію між компонентами системи.

Проведене тестування підтвердило коректність реалізації основних функціональних модулів системи, узгодженість роботи мікросервісів та стабільність функціонування платформи. Усі ключові сценарії використання, включаючи пошук приміщень, бронювання, оплату, формування звітів і управління даними, виконуються без помилок і відповідають очікуваним результатам.

Практична цінність роботи полягає у створенні програмного продукту, який може бути використаний для автоматизації процесів оренди приміщень, підвищення ефективності використання ресурсів і покращення якості обслуговування користувачів. Розроблена система може бути розширена шляхом інтеграції з додатковими сервісами, впровадження рекомендаційних алгоритмів, аналітики на основі великих даних та підтримки мобільних платформ.

Отже, поставлена мета роботи досягнута, а всі визначені завдання виконані у повному обсязі. Отримані результати можуть бути використані як основа для подальшого розвитку інформаційних систем у сфері оренди та управління ресурсами.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Уніфікована мова моделювання (Unified Modeling Language – UML). Офіційна специфікація OMG [Електронний ресурс] / Object Management Group. – Режим доступу: https://www.omg.org/uml/ (дата звернення: 10.02.2026).

2. UML-діаграми. Огляд структурних і поведінкових діаграм UML (UML Diagrams. UML structural and behavioral diagrams overview) [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/ (дата звернення: 15.02.2026).

3. Уніфікована мова моделювання (Unified Modeling Language) [Електронний ресурс] // Wikipedia. – Режим доступу: https://en.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 10.02.2026).

4. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Boston : Addison-Wesley, 2004. – 208 p.

5. Sommerville I. Software Engineering. – 10th ed. – Boston : Pearson Education, 2016. – 816 p.

6. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. – 2nd ed. – Boston : Addison-Wesley, 2005. – 496 p.

7. Архітектурний опис систем і програмного забезпечення (Systems and software engineering – Architecture description) : ISO/IEC/IEEE 42010:2011 [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/50508.html (дата звернення: 17.02.2026).

8. Bass L., Clements P., Kazman R. Software Architecture in Practice. – 3rd ed. – Boston : Addison-Wesley, 2013. – 624 p.

9. Richards M. Software Architecture Patterns. – Sebastopol : O’Reilly Media, 2015. – 162 p.

10. UML-діаграми: типи та приклади використання (UML diagrams) [Електронний ресурс] / Foxminded. – Режим доступу: https://foxminded.ua/uml-diagramy/ (дата звернення: 17.02.2026).

11. UML-діаграми для моделювання архітектури програмного забезпечення (UML diagrams for software architecture modeling) [Електронний ресурс] / Evergreens. – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 15.02.2026).

12. Системи управління інформаційною безпекою (Information security management systems) : ISO/IEC 27001:2022 [Електронний ресурс]. – Режим доступу: https://www.iso.org/isoiec-27001-information-security.html (дата звернення: 12.02.2026).

13. Основні ризики безпеки веб-застосунків OWASP Top 10 (OWASP Top 10 Web Application Security Risks) [Електронний ресурс] / OWASP Foundation. – Режим доступу: https://owasp.org/www-project-top-ten/ (дата звернення: 08.02.2026).

14. Інформаційні системи охорони здоров’я (Health Information Systems) [Електронний ресурс] / World Health Organization. – Режим доступу: https://www.who.int/data/health-information-systems (дата звернення: 09.02.2026).

15. Рекомендації з проєктування REST API (RESTful API Design Guidelines) [Електронний ресурс] / Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 20.02.2026).

**ДОДАТОК А**

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Декларація про академічну доброчесність та використання ШІ

Я, **Кривша Софія Володимирівна**, здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система пошуку оголошень оренди житла» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що:

- [✓ ] інструменти генеративного ШІ не використовувалися.
- [✓] інструменти ШІ (ChatGPT, Gemini, DeepL, були використані для:
  - [✓] перекладу іншомовних джерел;
  - [✓] стилістичного редагування та перевірки граматики;
  - [ ] допомоги у написанні/відлагодженні програмного коду;

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«__» _____2026 р. **Софія КРИВША**