МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп’ютерних технологій Кафедра комп’ютерних наук

# КВАЛІФІКАЦІЙНА РОБОТА

на тему:

***«Дослідження методів та технологій уникнення колізій при онлайн бронюванні з урахуванням підходів до оптимізації ресурсів та управління попитом»***

Рівень вищої освіти другий (магістерський) Спеціальність 122 Комп’ютерні науки Освітня програма Комп’ютерні науки

Виконала: студентка групи МгІТ1-23 Марія ПОСТОЄНКО

Науковий керівник

К.т.н., доц.Мар’яна ГОЛЬДБЕРГ Рецензент

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет Мехатроніки та комп'ютерних технологій Кафедра Комп’ютерних наук

Рівень вищої освіти другий (магістерський) Спеціальність 122 Комп’ютерні науки Освітня програма Комп’ютерні науки

# ЗАТВЕРДЖУЮ

Завідувач кафедри КН

Наталія ЧУПРИНКА

# ЗАВДАННЯ

«\_\_\_\_» \_листопада\_2024 р.

# НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Постоєнко Марії Олексіївні

1. **Тема роботи:** Дослідження методів та технологій уникнення колізій при онлайн бронюванні з урахуванням підходів до оптимізації ресурсів та управління попитом.

Науковий керівник роботи: *Гольдберг Мар’яна Ігорівна , к.т.н. доц.*

затверджені наказом КНУТД від «03» вересня 2024 року № 188-уч

1. **Вихідні дані до роботи:** Вихідні дані до кваліфікаційної роботи (проєкту) Розробки кафедри комп’ютерних наук. Літературні джерела з тематики дипломної роботи.
2. **Зміст дипломної роботи:** Розділ 1(Теоретичні основи систем онлайн- бронювання з урахуванням вирішення колізій); Розділ 2(Огляд використаних інструментів та технологій в рамках проектування програмної реалізації); Розділ 3 (Програмна реалізація системи онлайн-бронювання).
3. **Дата видачі завдання:** 08.2024

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапу кваліфікаційної роботи | Терміни виконання  етапів | Примітка про виконання |
| 1 | Вступ | 09.10.2024 |  |
| 2 | Розділ 1 Теоретичні основи систем онлайн-  бронювання з урахуванням вирішення колізій. | 12.10.2024 |  |
| 3 | Розділ 2 Огляд використаних інструментів та технологій в рамках проектування програмної  реалізації . | 20.10.2024 |  |
| 4 | Розділ 3 Програмна реалізація системи онлайн-  бронювання | 25.10.2024 |  |
| 5 | Висновки | 28.10.2024 |  |
| 6 | Оформлення (чистовий варіант) | 11.11.2024 |  |
| 7 | Подача кваліфікаційної роботи науковому керівнику  для відгуку (за 14 днів до захисту) | 15.11.2024 |  |
| 8 | Здача дипломної магістерської роботи на кафедру  для рецензування | 18.11.2024 |  |
| 9 | Перевірка дипломної магістерської роботи на  наявність ознак плагіату |  |  |
| 10 | Подання дипломної магістерської роботи на затвердження завідувачу кафедри |  |  |

|  |  |  |
| --- | --- | --- |
| **Студент** |  | Марія ПОСТОЄНКО |
| **Науковий керівник роботи** |  | Мар’яна ГОЛЬДБЕРГ |
|  |  |  |

# АНОТАЦІЯ

**Постоєнко М.О. *Дослідження методів та технологій уникнення колізій при онлайн бронюванні з урахуванням підходів до оптимізації ресурсів та управління попитом.* Дипломна магістерська робота за спеціальністю 122 –**

## «Комп’ютерні науки» - Київський національний університет технологій та дизайну, Київ, 2024 рік.

У роботі проаналізовано методи та технології уникнення колізій при проєктуванні та розробці систем онлайн-бронювання. Проведено дослідження алгоритмів запобігання колізіям, враховуючи оптимізацію ресурсів та управління попитом. Окрему увагу було зосереджено на методах та технологіях забезпечення ефективної взаємодії компонентів системи онлайн-бронювання клієнт-серверної архітектури.

На основі проведених досліджень, було створено систему онлайн- бронювання земельних ділянок та заміських будинків з впровадженням методів забезпечення уникнення колізій, управління попитом та оптимізації ресурсів. В роботі описано використані при розробці інструменти та продемонстровано інтерфейс користувача.

**Ключові слова:** система онлайн-бронювання, клієнт-серверна архітектура, колізії, управління попитом, оптимізація ресурсів, бронювання земельних ділянок, бронювання будинків.

# ANNOTATION

**Postoienko M. O. *Research of methods and technologies for avoiding collisions during online booking, including resource optimization and demand management*. Master’s Thesis, Specialty 122 – "Computer Science" – Kyiv National University of Technologies and Design, Kyiv, 2024.**

The work analyses methods and technologies for avoiding collisions development and implementation of online booking systems. A study of conflict prevention algorithms was conducted, taking into account resource optimization and demand management. Particular attention was paid to methods and technologies for ensuring effective interaction between the components of the online booking system with the client-server architecture.

Based on the research, an online booking system for land plots and country houses was created with the usage of methods to ensure conflict avoidance, demand management and resource optimization. The work describes the tools used in the development and demonstrates the user interface of the system.

**Keywords:** online booking system, client-server architecture, collisions, demand management, resource optimization, land plot booking, house booking.

# ЗМІСТ

[АНОТАЦІЯ 4](#_Toc183443261)

[ANNOTATION 5](#_Toc183443263)

[ЗМІСТ 6](#_Toc183443264)

[ВСТУП 7](#_Toc183443265)

[РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ОНЛАЙН-БРОНЮВАННЯ З УРАХУВАННЯМ ВИРІШЕННЯ КОЛІЗІЙ 10](#_Toc183443266)

[1.1 Історія розвитку систем онлайн-бронювання 10](#_Toc183443267)

[1.2. Основні поняття в розрізі онлайн-бронювання. 11](#_Toc183443268)

[1.2.1. Колізії 11](#_Toc183443269)

[1. Блокування ресурсів: 11](#_Toc183443270)

[2. Використання транзакцій: 12](#_Toc183443271)

[3. Регулювання перенавантаження: 13](#_Toc183443272)

[4. Прогнозування попиту: 15](#_Toc183443273)

[5. Алгоритми черговості: 15](#_Toc183443274)

[1.2.2 Онлайн-бронювання. 21](#_Toc183443275)

[1.2.3 Оптимізація ресурсів та управління попитом 23](#_Toc183443276)

[1.3 Огляд існуючих систем онлайн-бронювання 26](#_Toc183443277)

[1.4 Висновки до розділу 1 29](#_Toc183443278)

[РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ В РАМКАХ ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 30](#_Toc183443279)

[2.1 Середовище розробки 30](#_Toc183443280)

[2.2 Серверна частина 31](#_Toc183443281)

[2.2.1 Методології серверної розробки 33](#_Toc183443282)

[2.2.2 Технології серверної розробки 34](#_Toc183443283)

[2.3 Клієнтська частина 38](#_Toc183443284)

[2.4 Методологія вирішення колізій при клієнт-серверній архітектурі систем онлайн-бронювання 44](#_Toc183443285)

[2.5 Висновок до розділу 2 47](#_Toc183443286)

[РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОНЛАЙН- БРОНЮВАННЯ 48](#_Toc183443287)

[3.1 Використані інструменти 48](#_Toc183443288)

[3.2 Основний функціонал системи 50](#_Toc183443289)

[3.3 Перспективи адаптації розробленої системи онлайн-бронювання під використання в інших галузях. 57](#_Toc183443290)

[3.4 Висновки до розділу 3 58](#_Toc183443291)

[ВИСНОВОК 59](#_Toc183443292)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61](#_Toc183443293)

# ВСТУП

На сьогоднішній день, з розвитком сучасних інформаційних технологій, інтернет-комунікацій та глобалізації ринку відбулися кардинальні зміни у здійсненні бронювання послуг, таких як транспорт, готелі, подорожі та заходи. Онлайн-системи бронювання грають важливу роль в повсякденному житті, забезпечуючи швидкий і зручний доступ до ресурсів для мільйонів користувачів по всьому світу. Проте стрімке зростання попиту та обмеженість ресурсів створює нові виклики та проблеми для розробників систем бронювання. Основною проблемою при онлайн-бронюванні є необхідність забезпечення алгоритму, який буде попереджати виникнення колізій, які можуть виникнути при одночасному бронюванні одного ресурсу різними користувачами. Відсутність продуманої структури попередження колізій може викликати збої в системі, як-то дублювання бронювання, відмова у доступі, фінансові втрати, що може призвести до втрати довіри користувача до сервісу бронювання.

Протягом розвитку, створення та впровадження різних систем онлайн- бронювання досліджувалися аспекти проблем, пов’язаних з прогнозуванням попиту та уникненням конфліктів. Зокрема, при розробці систем використовувалися методи динамічного ціноутворення, системи нейронних мереж, технології розподілених баз даних, блокчейн-технології, тощо. У багатьох випадках оптимізація ресурсів виявляється ключовим аспектом у запобіганні колізіям. Вона дозволяє ефективніше розподіляти обмежені ресурси між клієнтами, враховуючи актуальний попит та наявність доступних можливостей. Управління попитом є важливим інструментом для фільтрування кількості запитів на послугу та її доступності. Ці процеси можуть включати динамічне ціноутворення, регулювання доступу, прогнози попиту та інші стратегії, спрямовані на підвищення ефективності роботи системи та мінімізацію колізій. Проте практичне запровадження даних технологій не може забезпечити повноцінне уникнення колізій та потребує подальших досліджень та апробацій.

Актуальність даної теми зумовлена зростаючою потребою у забезпеченні надійної та ефективної роботи онлайн-систем бронювання, які повинні динамічно реагувати на зміну попиту, розподіляти обмежені ресурси та забезпечувати швидке і надійне обслуговування користувачів. Запобігання колізіям допомагає знизити ризики фінансових втрат як для споживачів, так і для постачальників послуг. У зв’язку із зростанням конкуренції у сфері онлайн-бронювання при розробці системи бронювання є потреба забезпечити підвищення рівня задоволеності клієнтів, що може бути досягнене з використанням сучасних методів оптимізації ресурсів і управління попитом.

Метою даного дослідження є аналіз методів та технологій уникнення колізій у системах онлайн-бронювання з урахуванням сучасних тенденцій та підходів до оптимізації ресурсів. Задля досягнення мети дослідження необхідно проаналізувати існуючі методи та технології управління ресурсами у системах бронювання, дослідити інструменти, що можуть використовуватись для виявлення можливих колізій, розробити та протестувати алгоритм з урахуванням проаналізованої інформації.

Об’єктом дослідження є системи онлайн-бронювання ресурсів. Предметом дослідження є технології та методи уникнення колізій при онлайн-бронюванні з урахуванням управління попитом та оптимізації обмежених ресурсів.

Наукова новизна роботи полягає в розробці адаптованої системи бронювання з алгоритмом запобігання колізіям, що базується на прогнозуванні попиту та оптимізації розподілу ресурсів.

Практична значущість дослідження полягає у можливості використання результатів дослідження в подальшому процесі розробки систем бронювання транспорту, готелей, білетів, тощо. Проаналізована інформація дозволить зменшити кількість конфліктів при розробці систем бронювання та підвищити ефективність розподілу і використання доступних ресурсів.

Під час дослідження було застосовано комплексний підхід до вирішення поставлених завдань. Зокрема, проведено аналіз доступних літературних джерел, що включає аналіз наукових статей, монографій, технічної документації та інтернет джерел пов’язаних з тематикою уникнення колізій та оптимізації ресурсів. Також було розглянуто існуючі методи математичного моделювання, що дало можливість формалізувати процеси бронювання і оцінити ефективність застосування різних методів до розподілу ресурсів та оптимізації попиту. Алгоритми оптимізації, такі як генетичні алгоритми, методи гілок і меж, а також стохастична оптимізація, застосовувалися для ефективного розподілу обмежених ресурсів між користувачами. Це сприяло зменшенню кількості колізій та підвищенню загальної ефективності роботи систем.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ОНЛАЙН-БРОНЮВАННЯ З УРАХУВАННЯМ ВИРІШЕННЯ КОЛІЗІЙ

## Історія розвитку систем онлайн-бронювання

Розвиток систем онлайн-бронювання пройшов численні етапи, видозмінюючись відповідно до розвитку цифровізації та прогресу інформаційних технологій. Інтегруючи інноваційні технології та методи системи бронювання розвинулися з простих ручних методів обліку до сучасних багатофункціональних сервісів. Крім того, методи уникнення колізій, прогнозування попиту та оптимізації ресурсів значно змінили свою структуру та підхід до вирішення проблем.

На ранніх етапах розвитку сфери бронювання всі процеси здійснювалися вручну, з використанням телефонів та паперових журналів запису. Адміністратори та відповідальні за реєстрацію вручну перевіряли наявність доступних ресурсів, що часто викликало помилки пов’язані з дублюванням броні або недоступністю ресурсу.

Початковий процес впровадження автоматизації процесу бронювання відбувся у 1950-х роках в авіаційній галузі. Однією з перших автоматизованих систем бронювання стала Semi-Automated Business Research Environment (SABRE) створена компаніями American Airlines та IBM. Ця система дала можливість автоматизувати розподіл місць на рейсах та продаж квитків, що вплинуло на ефективність та швидкість обслуговування пасажирів і дало поштовх для розвитку сфери автоматизованого бронювання. У 1980-х роках почали з’являтися глобальні дистрибутивні системи (GDS), такі як Amadeus, Galileo, Travelport, що надали можливість з’єднувати постачальників послуг з даними про доступність ресурсів та пришвидшити процес пошуку доступних для клієнтів варіантів.

З розвитком Інтернету в 1990-х роках виникли онлайн туристичні агентства (OTA) (Expedia, Booking.com), які надали можливість клієнтам самостійно робити бронювання. Згодом, метапошукові системи (наприклад, Skyscanner) зробили

оптимізацію доступу до даних з кількох джерел, що надало можливість клієнтам порівнювати наявні ресурси та обирати найкращі пропозиції.

На сьогоднішній день, сучасні системи онлайн-бронювання використовують алгоритми штучного інтелекту (AI) з метою прогнозування попиту, персоналізації пропозицій та оптимізації динамічного ціноутворення. Метапошукові системи та чат-боти налаштовуються для можливості надання персоналізованих пропозицій користувачам, що підвищує ефективність роботи системи і зменшує ризик дублювання бронювань. Наприклад, система Airbnb використовує штучний інтелект для прорахування ціни на послуги залежно від попиту та актуальності в регіоні.

## Основні поняття в розрізі онлайн-бронювання.

## Колізії

**Колізія** – в різних сферах це поняття має розширене значення. В сфері інформаційних технологій, зокрема в онлайн-бронюванні, колізія означає конфлікт між двома (або більше) запитами або ресурсами. Найбільш поширеною ситуацією виникнення колізії є бронювання одного ресурсу двома (або більше) користувачами одночасно. Виникнення колізій може бути наслідком перенавантаження системи або неправильної синхронізації даних.

В онлайн-бронюванні можна виділити декілька суттєвих типів колізій: ресурсні – виникають при одночасному резервуванні ресурсу декількома користувачами, колізії баз даних – виникають при неправильно налаштованих базах даних і можуть спричиняти дублювання або втрату запитів, системні – виникають при перенавантаженні серверів або відсутності достатніх характеристик системи для обробки запитів.

Задля вирішення колізій в розрізі онлайн-бронювання використовуються методи запобігання колізіям[13,14]:

## Блокування ресурсів:

* + під час бронювання користувачем ресурс блокується для інших користувачів;
  + може бути реалізоване «за таймером» (якщо користувач не закінчить процес бронювання протягом певного заданого часу, ресурс буде розблокований для інших користувачів) та жорстким методом (поки транзакція за ресурсом не буде завершена, він залишається заблокованим);
  + забезпечує запобігання конфліктам одночасних запитів, проте при реалізації цього методу можуть створюватися «мертві зони» при тимчасовій недоступності ресурсу, що може призвести до незадоволення та незручності з боку користувачів.

## Використання транзакцій:

* + забезпечують повне завершення операції бронювання (в тому числі і відмову від операції), що дає можливість досягнути узгодженості запитів та даних у базах;
  + надійність транзакцій забезпечується використанням ACID принципів (Atomicity, Consistency, Isolation, Durability). ACID принципи складаються з набору властивостей:
    - Атомарність (Atomicity) – транзакція є неподільним процесом, який виконується повністю або не виконується взагалі. Коли користувач бронює ресурс, етапи операції (запит до бази щодо доступності, оплата, запис у базу) виконуються як одне ціле, а результатом є або успішне резервування, або скасування резервування.
    - Узгодженість (Consistency) – процес транзакції змінює базу даних з одного узгодженого стану в інший. Після завершення транзакції не створюються дублювання одного ресурсу.
    - Ізольованість (Isolation) – одночасні транзакції виконуються незалежно. Коли декілька користувачів бронюють однаковий ресурс, відбувається ізолювання процесів, кожен запит обробляється окремо і послідовно.
    - Довговічність (Durability) – результати транзакцій зберігаються в системі певний час (збереження інформації також відбувається і під час збоїв у процесі транзакції). Коли бронювання завершено дані зберігаються в базі навіть при помилках на сервері.
    - забезпечує бронювання ресурсу без дублювання даних, надійність та збереженні інформації про бронювання навіть у випадку збоїв системи, відновлення ресурсу до початкового стану у випадку скасування бронювання;
    - реалізація методу можлива за допомогою використання реляційних баз даних (MySQL, PostgreSQL), реалізації логування (write ahead logging), застосування спеціалізованих протоколів для узгодженості між вузлами (Two-Phase Commit (2PC), Paxos);
    - при використанні методу існують певні недоліки: через багатофункціональність ACID принципів робота системи може здійснюватися повільніше, що може призвести до втрати зацікавленості користувачів до системи;

## Регулювання перенавантаження:

* + забезпечує продуктивність та доступність системи в періоди високого навантаження;
  + реалізується шляхом розподілу запитів між серверами таким чином, щоб ресурси системи не були перевантажені;
  + при використанні цього методу можуть бути використані механізми балансування навантаження:
  + **DNS (Domain Name System)** – розподілення запитів між різними серверними вузлами. Цей механізм дає можливість розподілити навантаження, але не враховує стан серверів в реальному часі, що може спричиняти перенавантаження на окремих серверних вузлах;
  + **балансування на рівні мережевого трафіку** – реалізується з використанням апаратних (ADC F5 BIG-IP, Citrix ADC) або програмних (Nginx, HAProxy, AWS Elastic Load Balancer) рішень. Використання цього

механізму дозволяє проводити аналіз даних в реальному часі та перенаправляти запити на сервери з урахуванням їхньої продуктивності;

* + **балансування за алгоритмами –** такий механізм враховує використання статичних та динамічних алгоритмів. Статичні алгоритми: Round Robin (забезпечує відправку запитів на сервер по черзі), Least Connections (забезпечує розподіл запитів так, що сервери із найменшою кількістю активних з’єднань першочергово отримують новий запит). Динамічні алгоритми: Weighted Round Robin (проводить аналіз продуктивності серверів та розподіляє запити згідно їх потужності), Adaptive Load Balancing (аналізує завантаженість серверів у реальному часі та динамічно перенаправляє запити);
  + **балансування у хмарних середовищах** – запити розподіляються між віртуальними серверами за допомогою можливостей хмарних середовищ (AWS, Azure, Google Cloud). Цей механізм дозволяє забезпечити масштабування в залежності від завантаженості серверів та провести глобальний розподіл навантаження для підтримки доступності ресурсів**;**
  + **CDN (Content Delivery Network) –** розподілення запитів через мережу серверів, які розташовані у різних місцях світу. Такий механізм дозволяє зменшити час проведення процесів за рахунок передачі даних з географічно найближчого до користувача сервера. Також цей механізм дозволяє швидко обробляти статичні ресурси (файли, зображення);
  + при використанні методу система отримує можливість функціонувати навіть при збоях на окремих серверах, перенаправляти запити до серверів відповідно до їх завантаженості, швидше обробляти запити за рахунок перенаправлення нових запитів на вільні сервери. Також завдяки оптимальному розподілу навантаження знижуються витрати на серверну інфраструктуру та забезпечується стабільність роботи системи;
  + існують певні недоліки цього механізму: інфраструктурне та програмне забезпечення можуть бути достатньо дорогими для невеликих компаній.

Також у деяких випадках перешкодою може стати неможливість забезпечити надійну синхронізацію між серверами;

## Прогнозування попиту:

* + дозволяє вчасно передбачити пікові періоди популярності ресурсу, оптимізувати ціноутворення на основі очікуваного попиту та оптимізувати можливості сервісу (наприклад, графіки рейсів, розподіл персоналу, об’єми замовлення продукції);
  + реалізується шляхом використання аналітичних інструментів для оцінки необхідних ресурсів на майбутні періоди;
  + можна виділити певні інструменти та методи прогнозування в розрізі онлайн-бронювання:
  + аналіз попередніх бронювань, сезонності, ситуації в регіоні та змін попиту

(SQL, Tableau, Excel);

* + використання алгоритмів машинного навчання, які розраховують можливі фактори впливу (Python (бібліотеки Scikit-learn, TensorFlow), R);
  + використання Big Data аналітики задля управління великими обсягами даних, які були зібрані з різних відкритих джерел (Apache Spark, Hadoop);
  + використання регресійних моделей для виявлення аномалій або актуальності у попиті (ARIMA, SARIMA);
  + недоліки методу: чутливість даних до новизни, актуальності та повноти (наприклад, застарілі або неповні дані можуть призводити до помилкових прогнозів), динамічність ринку (раптові зміни попиту можуть ускладнити прогнозування або негативно вплинути на попередньо розраховані прогнози);

## Алгоритми черговості:

* + виконують обробку запитів з забезпеченням уникнення перенавантаження системи та надають послідовний доступ до ресурсів;
  + дозволяють обробляти запити в логічному порядку, з урахуванням пріоритетів, мінімізують затримки в роботі системи, дають можливість адаптувати систему до різних подій;
  + реалізуються шляхом використання різних типів:
  + **FIFO (First In, First Out) –** забезпечує обробку запитів у порядку їх надходження. Цей алгоритм забезпечує прозорість і справедливість (логічний порядок обробки запитів) для користувачів та простий в реалізації, проте недолік полягає в відсутності врахування пріоритетності запитів;
  + **LIFO (Last In, First Out)** – алгоритм першочергово обробляє найновіші запити. Цей алгоритм надає швидкий відгук для останніх запитів та підходить для короткотермінових завдань, де нові запити мають більший пріоритет ніж старі**.** В той же час першочергова обробка нових запитів може створювати затримку реалізації старих запитів;
  + **Priority queue –** забезпечує для запитів різні рівні пріоритетності і обробляє їх за чергою пріоритетів, незалежно від часу надходження (запити, які мають вищий пріоритет обробляються першими). Даний алгоритм гнучкий і продуктивний, бо забезпечує першочергову обробку для критично важливих запитів та дозволяє адаптувати систему до різних бізнес-логік (наприклад, VIP користувачі можуть отримати перевагу в наданні певних послуг). Проте даний алгоритм важкий в реалізації та вимагає більш складної логіки для оцінки пріоритету. Також існує ймовірність зловживання можливостями системи користувачами, шляхом штучного збільшення пріоритету задля швидкого отримання результатів запиту;
  + **Round Robin –** алгоритм обробляє запити по черзі, приділяючи кожному однакову частку часу. Такий алгоритм не вимагає складної логічної побудови, забезпечує рівність всіх запитів та гарантує обробку всіх запитів. Проте алгоритм не враховує складність запитів, що може зупинити процес обробки інших запитів в черзі;
  + недоліком використання алгоритмів черговості є можливість виникнення затримки у черзі при обробці запиту з великими ресурсними вимогами;

Стрімкий розвиток сфери онлайн-бронювання потребує використання ефективних методів запобігання колізіям, які можуть виникати через перевантаженість системи, помилки у синхронізації даних, дублювання запитів та інших факторів.

Колізії в сфері онлайн-бронювання є багатовимірним явищем, на яке можуть вплинути багато як внутрішніх, так і зовнішніх чинників. Колізії можна розділити на: технічні, операційні та організаційні.

Технічні колізії пов’язані зі збоями в системах. Такі колізії виникають при проблемах з інтеграцією системи, обробкою даних в реальному часі, виникненні втручання до системи з боку сторонніх осіб, атак на бази даних тощо. Найбільш критичною проблемою пов’язаною з технічними колізіями є витік конфіденційної інформації користувачів, що може спричинити фінансові втрати та юридичні конфлікти.

Операційні колізії пов’язані з нестачею ресурсів у реальному часі. Такі колізії можуть виникати через неправильне планування розподілу ресурсів, використання неефективних алгоритмів уникнення колізій або неправильного прогнозування попиту.

Організаційні помилки пов’язані з конфліктами між попитом і пропозицією. Такі колізії виникають при неорганізованому внесенні даних щодо доступних ресурсів, а також при невідповідності між доступними ресурсами та потребами користувачів.

Ще одним важливим фактором при проєктуванні системи онлайн-бронювання є визначення чинників, які можуть вплинути на виникнення колізій[14]:

1. Зовнішні чинники:
   * сезонність: у святкові періоди на певні ресурси (готельні номери, авіаквитки, квитки на концерти у періоди новорічних та літніх відпусток) попит значно зростає, перевищуючи звичайні обсяги. При недостатній

підготовці ресурсів до таких періодів може статися перенавантаження системи;

* + несподіваний попит: на раптові зміни в актуальності ресурсів можуть бути спричинені природними катастрофами(наприклад, після урагану може підвищитися попит на бронювання житла), масовими заходами (наприклад, в регіоні зростає попит на бронювання під час концертів, спортивних подій тощо), кризовими ситуаціями (наприклад, під час COVID-19 багато компаній зіткнулися з масовими скасуваннями бронювань та змінами бронювань);

1. Внутрішні чинники:
   * недосконалість використаних алгоритмів: невірна оцінка попиту через старі дані, відсутність адаптивності до змін в реальному часі, невчасне реагування на тенденції в суспільстві тощо може призвести до недієвості системи та фінансових втрат компанії.
   * неналежна якість даних: помилки у введених даних або неактуальна інформація призводить до виникнення ситуацій, коли користувачі не можуть зробити бронювання.
   * недостатня інтеграція між платформами: розсинхронізація між системами компанії (сайт, мобільний додаток тощо) створює інформаційний розрив і ускладнює обробку запитів.
   * невідповідність між автоматизованими системами та людським фактором.

Враховуючи можливі ризики та чинники, які впливають на роботу системи онлайн-бронювання, під час розробки варто ефективно використовувати ресурси майбутньої системи та впровадити якісне управління бронюванням задля уникнення ризиків[1,14]:

1. Погане обслуговування через перевантажений персонал:

Через відсутність в системі алгоритму вирішення проблем пов’язаних з дублюванням бронювань, вирішення конфліктів з клієнтами лягає на персонал, що може спричинити зниження якості обслуговування.

Наприклад, некоректна робота системи може призвести до витрати додаткового часу на спілкування з користувачами, вимушене надання компенсацій чи відмови від бронювань. В результаті, через непродуманість системи онлайн-бронювання можуть страждати користувачі, персонал та компанія, яка надає послуги. Крім цього, конфлікти спричиняють збільшення недовіри до компанії та ризик негативних відгуків.

1. Фінансові втрати:

Через постійні колізії невдоволені клієнти можуть скасовувати бронювання та відмовлятися від послуг компанії, що напряму зменшує дохід компанії. Наприклад, у готелях для залагодження конфліктів запроваджена практика пропонування компенсацій у вигляді знижок чи безкоштовного надання послуг. Також у випадку колізій пов’язаних з розкриттям персональних даних під час бронювання вимагають коштів для вирішення конфлікту (юридичні консультації для вирішення проблеми, вкладення у PR-кампанії для відновлення репутації).

1. Негативна репутація на ринку:

Часті колізії в роботі системи можуть спричинити поширення негативних відгуків на онлайн-платформах, що може відштовхнути нових користувачів та зменшити кількість бронювань.

1. Втрати через невірно спрогнозований попит:

Відсутність якісної оптимізації ресурсів і управління попиту може призвести до хибного прогнозування попиту та поганого розподілу ресурсів. Наприклад, в періоди «пікового» попиту через відсутність заздалегідь підготовленої потрібної кількості ресурсів можливе перевантаження системи та, в результаті, втрата клієнтів через відсутність доступних варіантів.

1. Юридичні конфлікти:

Через недотримання умов договору з користувачами (несвоєчасне надання послуг, відсутність ресурсу, який вже був заброньований), компанії можуть втратити ліцензії, стикнутися з фінансовими санкціями або судовими позовами.

1. Невідповідність даних у системі:

При відсутності налагодженого синхронізування даних між платформами може виникнути інформаційний хаос. Це може призвести до додаткових операційних витрат, втрати інформації користувачів та «витіку» даних у відкритий доступ.

Задля запобігання наведених ризиків при проєтуванні системи онлайн- бронювання важливо застосовувати актуальні підходи для уникнення колізій. Серед існуючих підходів можна виділити основні[14]:

1. Використання сучасних автоматизованих систем: автоматизовані системи надають можливість забезпечити миттєву синхронізацію даних між різними вузлами серверів. Використання хмарних платформ (наприклад, Mews) забезпечує можливість інтегрування управління ресурсами у єдину систему, що дозволяє зменшити ймовірність помилок через застарілу інформацію або дублювання даних. Програмні засоби Fieldproxy та Baluu дозволяють додавати буферний час між бронюваннями, що допомагає враховувати затримки чи технічні збої та попередити перевантаження системи.
2. Використання політик управління ресурсами: динамічне ціноутворення залежно від попиту допомагає зменшити кількість невигідних бронювань, зберігаючи доступність до ресурсів для пріоритетних користувачів. Також обмеження доступу до ресурсу для певних користувачів у «пікові» періоди допомагає ефективно розподілити ресурси системи (наприклад, корпоративним бронюванням буде надаватися пріоритетність).
3. Інтеграція прогнозування алгоритмів: для аналізу минулих даних про бронювання використовується машинне навчання, аналітичні алгоритми

прогнозують «пікові» періоди, що дозволяє попередити колізії пов’язані з нестачею або недоступністю ресурсів. Наприклад, програмна платформа SynXis надає можливість коригувати доступність ресурсів в реальному часі, що допомагає підлаштувати систему на основі потреб користувачів.

1. Використання мультиканальних підходів: інтеграція даних через канальні менеджери, використовуючи декілька платформ бронювання, дозволяє уникати дублювань. Наприклад, Expedia (онлайн туристичне агентство) використовує спеціальні модулі щоб синхронізувати об’єкти розміщення на різні канали в режимі реального часу.
2. Тестування та аудит процесів: регулярне проведення тестування та аудиту системи допомагає виявити слабкі сторони та уникнути подальших колізій. Запровадження сценарних тестів може допомогти перевірити, як система реагує на різні ситуації та удосконалити її. Аналіз попередніх виявлених колізій дає можливість вчасно зреагувати на можливі помилки та уникнути збоїв в системі.
3. Використання блокчейн-технологій: блокчейн-технології допомагають зберігати інформацію у розподіленій мережі, що виключає ризик втрати даних або маніпуляції з даними сторонніми особами.

## Онлайн-бронювання.

**Онлайн-бронювання** – автоматизований процес резервування послуг та ресурсів за допомогою використання відповідних систем. Системи онлайн- бронювання допомагають користувачам перевірити доступність необхідних ресурсів, порівняти ціни та швидко зробити бронювання самостійно в будь-який час.

На сьогоднішній день онлайн-бронювання широко застосовується в різних сферах: готельний бізнес (резервування номерів, додаткових послуг), транспорт (бронювання квитків, літаків, автобусів; оренда автомобілів), культурно- розважальні заходи (резервування квитків у театр, кіно чи концерти), харчування

(бронювання столиків у ресторанах), здоров’я (запис до лікарів, проведення аналізів).

При створенні системи онлайн-бронювання можна виділити ключові компоненти, які будуть впливати на продуктивність та успішність роботи системи[26]:

1. Інтерфейс користувача (UI/UX): інтерфейс має бути зручним, зрозумілим, візуально приємним для користувача, включати функції фільтрації, пошуку, вибору дат та часу, інформацію про ресурс, що бронюється та інформацію щодо правил бронювання;
2. Модуль управління бронюваннями: в системі має бути реалізована логіка управління бронюванням (створення, редагування, скасування), яка буде обробляти запити в реальному часі;
3. Синхронізація з базою даних: дані про ресурси мають зберігатися в надійно захищеній централізованій базі даних. База даних має підтримувати алгоритми запобігання колізіям для забезпечення ефективності та продуктивності.
4. Система сповіщень: інформування користувачів про підтвердження, зміну або скасування замовлення дозволить оптимізувати та полегшити процес бронювання для користувачів;
5. Аналітичні інструменти: використання аналітичних алгоритмів та збирання відгуку від користувачів дозволить покращувати продукт та робити його конкурентоспроможним;

Системи онлайн-бронювання дозволяють користувачам швидко робити бронювання 24/7, обирати найбільш вигідні пропозиції та отримувати рекомендації щодо майбутніх бронювань. Проте такий вид резервування ресурсів має свої недоліки: системи не можуть функціонувати без доступу до Інтернету, завжди існує ризик крадіжки даних, кібератак або шахрайства, деякі системи не підтримують зміну або скасування бронювань. Також певні групи населення можуть мати труднощі з використанням систем онлайн-бронювання.

## Оптимізація ресурсів та управління попитом

**Оптимізація ресурсів** – процес максимально ефективного та продуктивного використання наявних можливостей системи.

**Управління попитом** – процес аналізу, прогнозування та впливу на рівень попиту, що дозволяє збільшити ефективність системи, уникнути перенавантаження та забезпечити найкраще використання доступних ресурсів.

Оптимізація ресурсів та управління попитом є важливими аспектами систем онлайн-бронювання. Без забезпечення цих аспектів неможливо ефективно керувати обмеженими або недоступними ресурсами, що може негативно вплинути на задоволення потреб користувачів. У системах онлайн-бронювання це особливо актуально, оскільки зростання кількості користувачів та запитів може призвести до перевантажень, коли попит перевищує наявні ресурси, спричиняючи колізії. Управління попитом та оптимізація ресурсів дає змогу не лише уникати таких ситуацій, але й збільшувати ефективність системи в цілому, забезпечуючи баланс між попитом і наявністю послуг.

У сфері бронювання існують різні стратегії, які забезпечують оптимізацію ресурсів та управління попитом, а також знижують ризик колізій[16]:

1. Управління доступністю: система онлайн-бронювання має забезпечувати фунціонал автоматичного оновлення статусів доступних ресурсів після кожного процесу бронювання задля мінімізації ризику дублювання.
2. Прогнозування попиту: прогнозування є критично важливою складовою функціонування системи онлайн-бронювання. Для реалізації прогнозування найчастіше використовують алгоритми машинного навчання, алгоритми завантаженості та реалізацію різних сценаріїв розвитку подій.
3. Ціноутворення: цей аспект є одним з важливих при формуванні вартості ресурсів, які надаються. Для ефективного встановлення цін використовуються наступні методи: динамічне ціноутворення – ціни змінюються залежно від попиту, доступності ресурсів, часу, події або

популярності (наприклад, підвищення вартості бронювання столиків під час свят), yield management – забезпечення максимізації доходу шляхом диференціації цін (наприклад, запровадження спеціальних пропозицій перед рейсом при великій кількості вільних місць).

1. Розподіл ресурсів: головна ідея розподілу ресурсів це призначення об’єктів (квитки, номери, місця) на основі запитуваних критеріїв (бюджет користувача, наявність місць, умови реалізації).
2. Сегментація користувачів: цей метод полягає у визначенні пріоритетних груп користувачів та наданні їм індивідуальних пропозицій відповідно до потреб. Персоналізація послуг допомагає знизити ризик одночасного звернення до одного ресурсу та дозволяє краще управляти доступністю ресурсів.
3. Обмеження доступу до ресурсів: блокування доступу до дефіцитних ресурсів для певної категорії користувачів також допомагає уникнути колізій та встановити різні правила для певних категорій користувачів.
4. Масштабування системи: метод полягає у використанні хмарних технологій для динамічного збільшення/зменшення можливостей системи в залежності від навантаження. Також масштабування може надавати можливість кешувати дані, що зменшує навантаження на базу даних.

Під час прогнозування попиту та оптимізації ресурсів використовуються різноманітні аналітичні інструменти[17, 18]:

* 1. Машинне навчання (ML): використання для аналізу великих обсягів даних дає можливість точно передбачити попит та адаптувати стратегії бронювання. Найбільш поширено застосовуються методи кластеризації, регресійного аналізу та нейронних мереж.
  2. Інструменти бізнес-аналітики (BI): платформи бізнес-аналітики (Power BI, Tableau, Google Data Studio) дають можливість візуалізації даних для розробки прогнозів. Такі інструменти дають можливість

бізнес-аналітикам швидко реагувати на зміни попиту, виявляти потенційні проблеми в системі і коригувати доступність ресурсів.

* 1. Yield Management Systems (YMS): системи управління доходами допомагають реалізовувати динамічне ціноутворення залежно від попиту, доступності ресурсів та поведінкових чинників. Ці системи активно використовуються в готельному бізнесі, авіаперевезеннях та автоперевезеннях для ефективного керування цінами на номери та місця.
  2. Програмне забезпечення для оптимізації цін: програмне забезпечення (наприклад, Revionics, RStudio, Pricefx) дозволяє автоматично адаптувати ціни, враховуючи аналітичні дані та зміни в попиті. Це дозволяє знизити ризик перевантаження та стимулювати бронювання не в «пікові» періоди.
  3. CRM-системи: системи управління відносинами з користувачами (Salesforce, HubSpot, Zoho) дозволяють збирати та аналізувати інформацію про користувачів та створювати для них персоналізовані пропозиції. Такі системи спрощують управління попитом, оптимізацію ресурсів та дозволяють зрозуміти потреби користувачів.
  4. Аналіз поведінки користувачів: технології для збору даних про поведінку користувачів (Google Analytics, Hotjar, Mixpanel) допомагають виділити типові сегменти користувачів, визначити цільову аудиторію та адаптувати доступність ресурсів. Це дозволяє ефективно розподіляти запити на обмежені ресурси.
  5. ERP-системи: системи планування ресурсів дають можливість відстежувати наявність ресурсів в реальному часі, враховувати попит та динамічно встановлювати доступність ресурсів. Це допомагає знизити ризик колізій, оскільки система може адаптуватися до змін попиту шляхом розподілення ресурсів.
  6. Системи оптимізації розкладу: такі системи (When I Work, Humanity, Shiftboard) дають інформацію щодо ефективного управління

розкладом і розподілом ресурсів залежно від прогнозованого попиту. Для компаній з обмеженим персоналом або обладнанням це може допомогти розподілити можливості під час «пікових» періодів.

* 1. Використання хмарних сервісів: сучасні хмарні сервіси (Amazon Web Services(AWS), Google Cloud Platform(GCP), Microsoft Azure) пропонують можливості динамічного масштабування, що дозволяє збільшувати обчислювальну потужність системи та упорядковувати місце для зберігання даних. Таке використання забезпечує гнучкість системи та знижує ризик виникнення колізій.
  2. Використання контейнеризації та мікросервісної архітектури: використання контейнерів (Docker) дозволяє масштабувати окремі компоненти системи, враховуючи навантаження на сервери. Це дає можливість забезпечити надійність та продуктивність онлайн- бронювань навіть при різких коливаннях попиту.

## Огляд існуючих систем онлайн-бронювання

На сьогоднішній день існує велика кількість систем онлайн-бронювання в сферах готельного бізнесу, авіаперевезень, харчування, здоров’я, дозвілля тощо. Розглянемо декілька систем онлайн-бронювання, що набули популярності по всьому світу:

1. Google Travel – платформа створена компанією Google для подорожей. Ця система використовує можливості пошукової системи Google та інтегрує дані з різних джерел, дозволяючи користувачам порівнювати ціни на готелі, авіаквитки та оренду автомобілів. Задля уникнення колізій використовуються алгоритми прогнозування цін та попиту. Google Travel аналізує потреби користувачів та надає персоналізовані пропозиції, що дозволяє уникнути перенавантаження системи через високий попит.
2. Expedia – одна з найбільш популярних платформ для бронювання під час подорожей. Система пропонує бронювання готелей, авіаквитків, прокату автомобілів та турів. Для мінімізації колізій в системі використовуються

інструменти аналізу попиту та пропозиції, що дає можливість продуктивно розподіляти ресурси та управляти навантаженнями. Проте за відгуками користувачів можна дізнатись, що система має певні проблеми з процесом скасування бронювань та повернення коштів.

1. Booking.com – ще одна популярна платформа для бронювань. Система дозволяє забронювати кімнату в готелі, житло, квитки на заходи та орендувати авто. Тут використовується механізм резервування в реальному часі, що забезпечує надання альтернативних варіантів у випадку недоступності ресурсів. Проте за відгуками користувачів ціни, вказані в системі, можуть бути вищими ніж в інших схожих системах.
2. Agoda – популярна платформа на азійському ринку. Agoda пропонує конкурентоспроможні ціни на готелі та житло в різних країнах. Тут задля уникнення колізій використовуються алгоритми для управління попитом та обмеженням доступу до ресурсів для певних груп користувачів.
3. Kayak – система, що спеціалізується на доступі до порівняння цін на авіаквитки, готелі та оренду автомобілів. Система використовує багатофункціональні пошукові інструменти, що дозволяє користувачам підібрати оптимальні варіанти відповідно до їхніх потреб. Цей механізм допомагає зменшити ризики технічних збоїв та колізій при онлайн- бронюванні.
4. Skyscanner – популярна метапошукова система та туристична агенція. Система дає можливість користувачам порівнювати ціни на авіаквитки, готелі та оренду транспорту. Для оптимізації ресурсів та управління попитом Skyscanner використовує алгоритми прогнозування попиту та алгоритми відслідковування цінових тенденцій, що дозволяє зменшити кількість колізій.
5. Rentalcars.com – популярний онлайн-сервіс для оренди автомобілів. Система пропонує користувачам порівнювати ціни на автомобілі та робити вибір бронювання за типом та ціновим діапазоном автомобілів. Тут

використовуються гнучкість системи та зручний інтерфейс для забезпечення задоволення потреб користувачів.

1. Eent-A-Car – велика система онлайн-бронювання автомобілів на глобальному рівні. Система надає можливість бронювання автомобіля одноразово або як частину пакету для подорожей з можливістю повернення та продовження оренди.
2. Europcar – відома система бронювання автомобілів по всьому світу. Система підбирає персоналізовані пропозиції та адаптована під використання в мобільному додатку.
3. OpenTable – найбільша у світі платформа бронювання столиків у ресторанах. Система дозволяє користувачам обирати ресторани, переглядати відгуки, бронювати столики та отримувати персоналізовані пропозиції. Задля уникнення колізій в системі підтримується автоматична обробка запитів в реальному часі.
4. Bookatable – європейська система для бронювання столиків в ресторанах. Система надає можливість бронювання у ресторанах по всій Європі з можливістю інтегрування з мобільними додатками та сповіщеннями про бронювання.
5. Eatigo – ще одна система для онлайн-бронювання столиків в ресторанах. Система надає пропозиції для бронювання і спеціально підбирає знижки, які залежать від часу бронювання та вподобань користувача.
6. Eventbrite – система для онлайн-бронювання квитків на культурні та розважальні заходи. Система дозволяє користувачам бронювати квитки на концерти, фестивалі, виставки та театри. Eventbrite забезпечує обробку запитів з великою кількістю даних, до дає змогу уникнути колізій.
7. StubHub – одна з найбільших платформ для покупки та продажу квитків на спортивні та розважальні заходи. Система пропонує для користувачів можливість онлайн-бронювання квитків та надання персоналізованих пропозицій. Також система інтегрована з іншими сервісами, що забезпечує мінімізацію технічних збоїв.
8. Ticketmaster – міжнародна система онлайн-бронювання квитків на концерти, вистави та спортивні події. Система використовує алгоритми управління попитом, що дозволяє користувачам безпечно та швидко здійснювати бронювання.
9. Zocdoc – система онлайн-запису на прийом до лікарів. Користувачі мають можливість обрати медичного працівника, час та дату для запису на прийом. Система використовує гнучке управління розкладом задля можливості швидкої зміни бронювання та уникнення колізій.
10. Doctolib – ще одна система для онлайн-запису до лікарів в Європі. Система використовує спеціальні алгоритми, які інтегровані для роботи медичних установ і враховують механізми уникнення колізій, пов’язаних з перенавантаженням та дублюванням.

## Висновки до розділу 1

В розділі 1 було розглянуто методи та технології уникнення колізій при онлайн-бронюванні, чинники та компоненти, які впливають на ефективність та продуктивність систем онлайн-бронювання. Було з’ясовано причини виникнення колізій та можливі рішення їх усунення. Також було розглянуто існуючі системи онлайн-бронювання та проаналізовано методи та технології, які в них застосовуються.

Аналіз інформації, наведений у розділі 1, підкреслює необхідність аналізу та оцінки ефективності при проєктуванні систем онлайн-бронювання. Забезпечення оптимізованої та адаптованої системи передбачає простоту використання, оперативність реагування на зміни на ринку та використання актуальних і повних даних. Дотримання забезпечення функціонування системи з використанням актуальних механізмів та методів уникнення колізій дозволить значно підвищити якість, швидкість та рівень роботи системи онлайн бронювання, а також підвищить досвід користувачів, що сприятиме підвищенню доходів та конкурентоспроможності системи.

# РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ В РАМКАХ ПРОЕКТУВАННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Процес проектування та розробки системи онлайн-бронювання потребує підбору правильних інструментів, методологій та технологій, які забезпечать надійну роботу, продуктивність, гнучкість та адаптивність створюваної системи. Розробка систем, які розгортаються в браузерах, найчастіше складається з розробки серверної та клієнтської частин. Для забезпечення правильної комунікації між клієнтом та сервером необхідно ретельно підходити до вибору інструментів, технологій та методів для забезпечення оптимізованої, продуктивної та ефективної роботи системи[15].

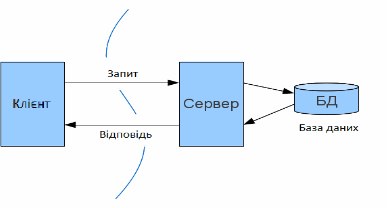


Рис 2.1. Схема процедури обміну даними між клієнтом та сервером

## Середовище розробки

При розробці системи онлайн-бронювання було проаналізовано ряд доступних середовищ розробки (IntelliJ IDEA, NetBeans) та обрано VS Code. Вибір цього середовища обґрунтовано його гнучкістю налаштувань, можливості інтеграції сучасних технологій та масштабування.

Visual Studio Code (VS Code) – є одним з найбільш використовуваних інструментів для розробки веб-додатків. Багатофункціональність та можливість інтеграції додаткових плагінів і сервісів забезпечує продуктивність та гнучкість розробки[10].

Під час вибору середовища було проаналізовано переваги VS Code[11]:

1. Швидкодія та легкість використання: VS Code простий в використанні та займає небагато системних ресурсів.
2. Можливість масштабування через підключення плагінів: дане інтегроване середовище розробки (IDE) підтримує підключення додаткових розширень, що дозволяє адаптувати роботу середовища під конкретний проєкт. Наприклад, плагін PHP IntelliSense дозволяє виявити помилки в коді та надає підказки для автозаповнення коду мовою PHP.
3. Універсальність: VS Code підтримує різні мови програмування, зокрема PHP, HTML, CSS, JavaScript тощо, що надає можливість розробляти різні компоненти системи (серверна частина та клієнтська частина) в одному середовищі.
4. Наявність інструментів тестування: вбудовані можливості VS Code дозволяють провести тестування коду та виявити помилки в процесі розробки. Наприклад, протягом розробки системи онлайн-бронювання було виявлено помилки в коді для функцій обробки бронювання та відправки запитів до бази даних.
5. Підтримка IntelliSense: для системи онлайн-бронювання, яка написана мовою PHP, зручним інструментом стало розширення PHP IntelliSense, яке надає підказки щодо коду та спрощує формування складних конструкцій коду.
6. Вбудований Emmet: вбудовані набори плагінів для текстових редакторів пришвидшують роботу з HTML шаблонами та дають можливість зосередитись на більш складних компонентах коду.
7. Зручність роботи з базами даних: завдяки розширенню SQL Server можливо легко працювати з MySQL для управління даними про бронювання.

## Серверна частина

Під час розробки серверної частини системи онлайн-бронювання варто враховувати забезпечення ефективної та надійної роботи бази даних. Серед

доступних варіантів для серверної розробки системи онлайн-бронювання для аналізу було обрано стек LAMP (Linux, Apache, MySQL, PHP).

Стек LAMP (Linux, Apache, MySQL, PHP) – вільне ПЗ з відкритим кодом, яке найчастіше використовується для створення серверних рішень. Цей стек забезпечує надійність та ефективність створюваної системи[9].

Задля зручного розгортання веб-серверу та бази даних додатково можливе використання багатоплатформної збірки веб-серверів XAMPP. XAMPP містить в собі HTTP-сервер Apache, бази даних (MariaDB, MySQL) та інтерпретатори скриптів PHP, Perl. Apache відповідає за обробку HTTP-запитів та їх передачу PHP інтерпретаторам і є одним з найбільш популярних веб-серверів, бо надає можливість легкої інтеграції з PHP, підтримки додаткових модулів для розширення функціональності, масштабування та надійності обробки одночасних запитів. Система управління базами даних MySQL є реляційною базою даних, що дозволяє обробляти, зберігати та керувати інформацією запитів. Перевагами MySQL є можливість підтримки складних реляційних шляхів, швидко обробляти запити та інтегрувати з PHP. Мова PHP допомагає розробити систему шляхом легкої інтеграції з MySQL, використання наявних фреймворків і бібліотек для роботи даними та підтримки інтерактивності контенту[12,19,22].

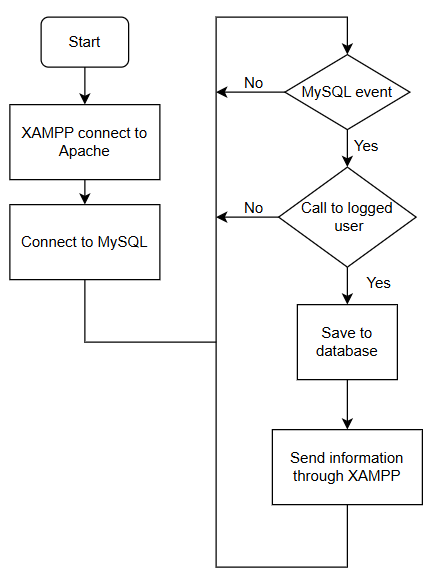


Рис 2.1.1 Схема роботи середовища XAMPP

При розробці систем онлайн-бронювання важливу роль грає застосування методологій, які дозволять забезпечити оптимізацію ресурсів, управління попитом та уникнення колізій. У рамках серверної розробки можливе застосування різних методологій та технологій для досягнення поставлених задач.

## Методології серверної розробки[8]

1. Concurrency Control (Керування паралелізмом) – набір методів, які дають можливість забезпечити коректне виконання одночасних транзакцій. Керування паралелізмом поділяється на Optimistic Locking (Оптимістичне блокування) та Pessimistic Locking(Песимістичне блокування).

Оптимістичне блокування передбачає, що колізії виникають не часто, тому під час обробки запитів не відбувається блокування, проте при збереженні інформації відбувається перевірка, чи не вплинули процеси обробки інших запитів на ресурси в поточному запиті.

Оптимістичне блокування може бути представлене математичною моделлю

(2.1):

де PS – ймовірність успішної обробки запиту, Pinit – початковий стан запиту, Ccoll –

кількість колізій, Ct – загальна кількість запитів.

Песимістичне блокування передбачає блокування ресурсів для інших запитів під час обробки поточного запиту. Такий механізм гарантує, що процеси не зможуть вносити зміни в ресурс під час виконання запиту.

1. Load Balancing (Балансування навантаження) – набір методів, що дають можливість рівномірно розподілити запити між серверами, забезпечуючи надійність системи та менший час обробки запиту.

Одними з методів балансування навантаження є Round Robin та Least Connections. Найпростіший метод Round Robin забезпечує рівномірне розподілення запитів по серверам, метод Least Connections розподіляє запити, враховуючи завантаженість серверів. За Least Connections сервери з найменшою кількістю з’єднань першочергово отримують нові запити, що допомагає оптимізувати роботу системи та надає можливість легкого масштабування системи при збільшенні попиту.

Розподіл запитів за методом Least Connections можна описати формулою (2.2):

де Ci – загальна кількість активних з’єднань на сервері.

1. Rate Limiting (Обмеження запитів) – метод, який обмежує кількість запитів для системи задля уникнення колізій пов’язаних з кібератаками та перевантаженням системи. Цей метод може бути реалізований шляхом використання алгоритмів Token Bucket (використання токенів) та Leaky Bucket (використання черг).
2. Demand Forecasting (Прогнозування попиту) – методологія, яка використовується для прогнозування попиту та майбутніх навантажень на систему, що дає можливість забезпечити уникнення колізій, пов’язаних з перевантаженням системи. Для реалізації цієї методології можуть бути використані алгоритми машинного навчання та лінійної регресії.

Використання лінійної регресії для прогнозування попиту може бути представлене формулою (2.3):

де Dt – прогнозований попит в період часу t, α0 та α1 – коефіцієнти лінійної регресії, e – помилка.

1. Caching (Кешування) – метод, який дозволяє зберігати результати запитів, які повторюються, у тимчасовому сховищі (кеші) задля зменшення навантаження на бази даних та можливості швидкого доступу до інформації.

## Технології серверної розробки

Для реалізації методів серверної розробки необхідна наявність технологій, які зможуть забезпечити ефективну, стабільну та безпечну роботу систем. В рамках серверної розробки найчастіше використовуються серверні мови програмування, бази даних, фреймворки та контейнеризація[24].

1. Серверні мови та фреймворки[23]:
   * Node.js та Express.js: використовують асинхронну модель, що дає можливість швидко обробляти одночасні запити, забезпечує

комунікацію між клієнтською та серверною сторонами завдяки використанню JavaScript на сервері.

* + мови програмування Python, PHP, Java;
  + Django: фреймворк мови програмування Python, який надає інструменти для роботи з базами даних та забезпечує безпеку і масштабованість систем.
  + Lavarel: фреймворк мови програмування PHP, який дозволяє забезпечити зручну роботу з базами даних та авторизацією користувачів.

1. Бази даних[25]:
   * MySQL: реляційна база даних, яка підтримує використання транзакцій, SQL-запитів та інтеграцію з мовами програмування PHP, Python та JavaScript.
   * PostgresSQL: реляційна база даних з відкритим кодом, яка підтримує роботу зі складними та більшими за обсягом даними, включає аналіз різних типів даних та має складну систему забезпечення цілісності даних при обробці запитів.
   * MongoDB: нереляційна база даних, яка зберігає дані в форматі BSON (бінарний формат JSON), що забезпечує гнучкість і швидкість роботи системи та дозволяє зберігати дані за динамічними схемами.
2. Контейнеризація:
   * Docker: система для управління ізольованими контейнерами проєкту, що надає інструменти для адаптації додатків до роботи в різних середовищах. Задля зниження ризику несумісності в різних середовищах розробки Docker надає можливість розробниками створювати, тестувати та впроваджувати додатки в контейнерах[20].
   * Kubernetes: популярна платформа для оркестрації контейнерів, що дозволяє автоматизувати розгортання та управління додатками. Задля забезпечення доступності контейнерів та продуктивності систем Kubernetes автоматично розподіляє контейнери по доступним ресурсам[27].

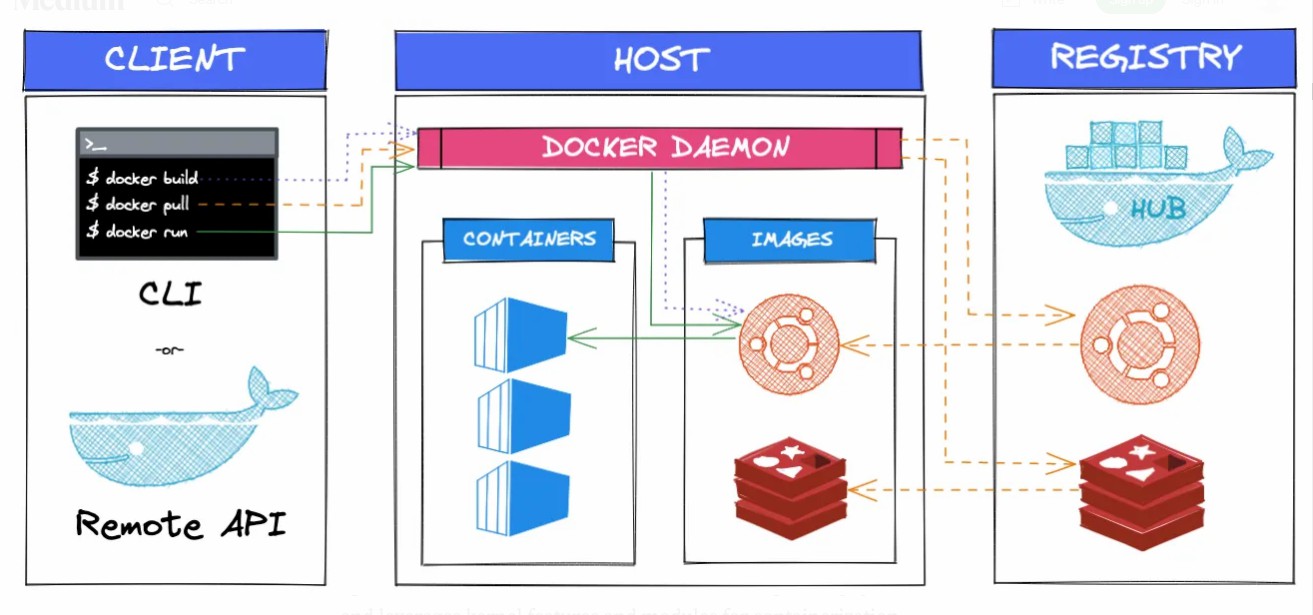


Рис 2.1.1. Принцип роботи високорівневої архітектури Docker

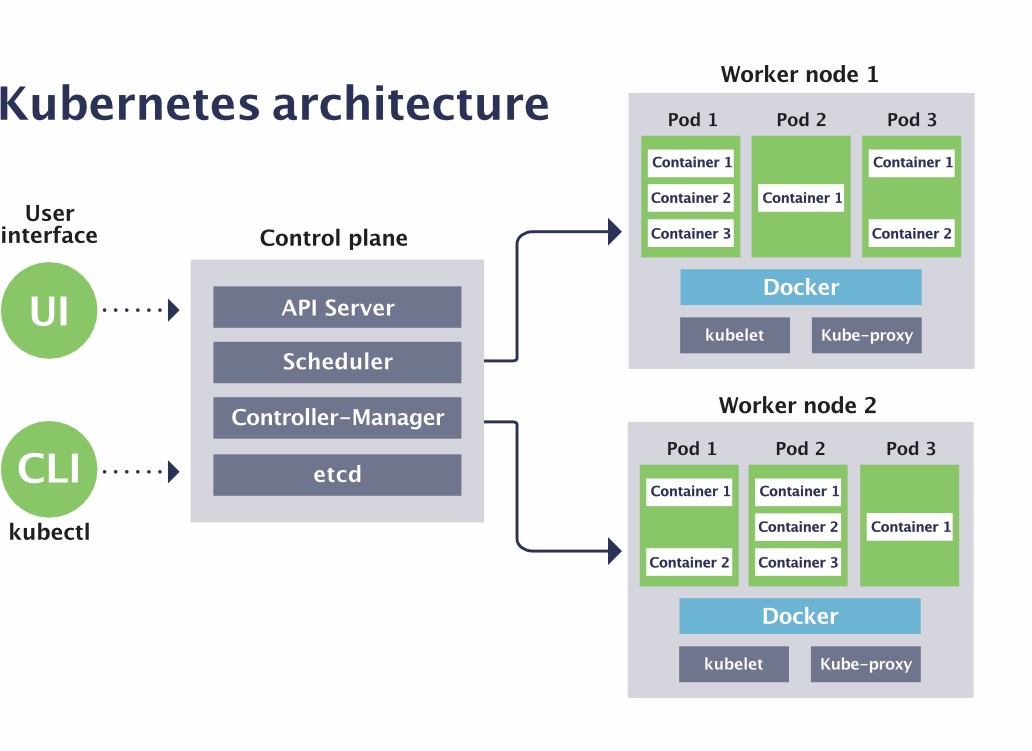


Рис 2.1.2. Принцип роботи Kubernetes

Важливою частиною серверної розробки є забезпечення безпеки даних, які надходять та зберігаються в базах даних. Задля забезпечення безпеки серверної частини використовуються протоколи шифрування, методи створення токенів та протоколів авторизації/аутентифікації[13]:

1. SSL/TLS (Secure Sockets Layer/Transport Layer Security) шифрування: протоколи шифрування, що забезпечують захищений потік передачі

даних між клієнтом та сервером. Використання цих протоколів дозволяє уникнути викрадення даних під час передачі та забезпечити конфіденційність інформації (наприклад, платіжна інформація, персональні дані користувачів, дані для авторизації). Ці протоколи використовують публічні і приватні ключі для забезпечення шифрування та аутентифікації веб-сервера[28].

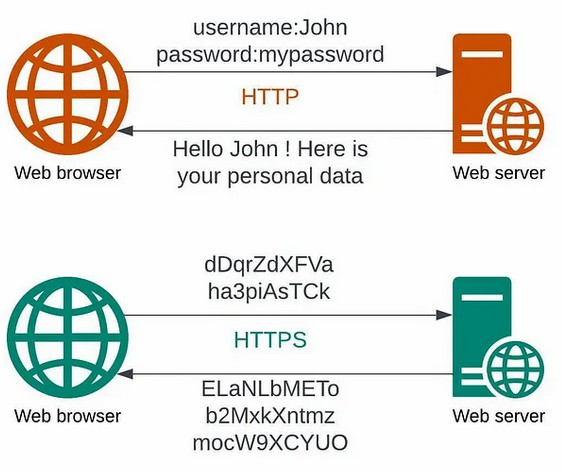


Рис 2.1.3. Порівняння відправлення запитів без використання та з використанням SSL/TLS протоколів

1. JWT (JSON Web Tokens): один з протоколів авторизації та аутентифікації, що є стандартом токена доступу на основі JSON. JWT застосовується для безпечної передачі даних між клієнтом та сервером шляхом підтвердження ідентичності користувачів без потреби зберігання паролів на серверах[28].
2. OAuth2: протокол-стандарт авторизації та аутентифікації, що надає можливість користувачам створювати та використовувати один обліковий запис для декількох додатків/сервісів. Також цей протокол дозволяє користувачам отримувати доступ до ресурсів через посередників (наприклад, Google, Facebook)[29].

Для проєктування серверної частини можуть бути використані ER-діаграми. ER-діаграми використовуються для візуалізації та моделювання структури бази даних. Система онлайн бронювання може бути представлена наступною діаграмою:

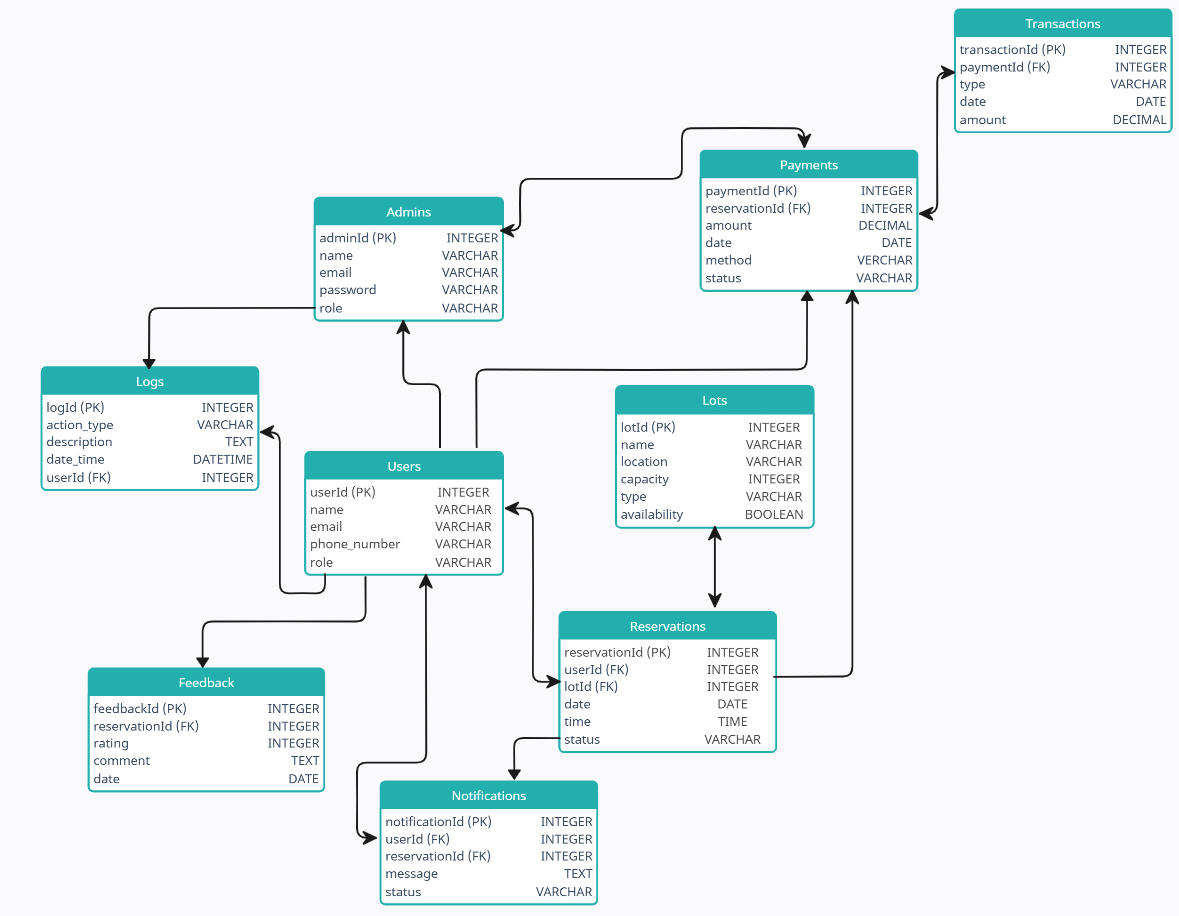


Рис 2.1.4 ER-діаграма системи онлайн-бронювання

* 1. **Клієнтська частина**

Під час розробки систем онлайн-бронювання не менш важливою частиною є клієнтська сторона. Для успішного впровадження системи та задоволення користувачів необхідно забезпечити інтерактивність, зручність та оптимізованість системи.

Розробка клієнтської сторони потребує використання сучасних методів та технологій, які дозволять розробити продуктивну систему з урахуванням потреби забезпечення злагодженої роботи з серверною частиною та уникнення колізій.

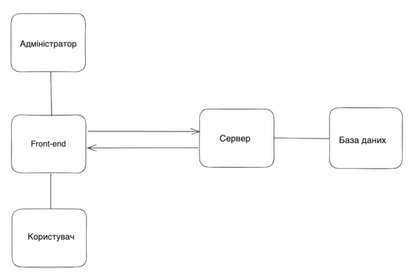


Рис 2.3.1 Структурна схема зв’язку клієнтської та серверної частин

Основними інструментами розробки клієнтської частини є HTML, CSS та

JavaScript[2, 4]:

1. HTML (HyperText Markup Language): мова розмітки гіпертексту є основою всіх сучасних веб-додатків та використовується для визначення структури контенту веб-сторінок. HTML підтримує вбудовані можливості реалізації мультимедіа на веб-сторінках (відео, аудіо, фото), а також рендерингу графіки та анімацій. HTML сумісний з різними платформами, а також підтримує реалізацію SEO (Search Engine Optimization) компонентів для пошукової оптимізації додатків. Проте використання HTML може бути громіздким при розробці великих проєктів та потребує використання сучасних браузерів для повної підтримки функціоналу[3].
2. CSS (Cascading Style Sheets): мова стилю сторінок, що забезпечує візуальну естетику веб-сторінок. Використання CSS дозволяє створити візуально привабливий продукт та забезпечити адаптивність веб-сервісів до різних розмірів девайсів, використовуючи мінімальну кількість ресурсів системи. Не дивлячись на поширеність використання, розробники можуть стикнутися з проблемою несумісності певного функціоналу між браузерами.
3. JS (JavaScript): динамічно, об’єктно-орієнтована мова програмування, що відповідає за інтерактивність та динамічність веб-сервісів. За допомогою JS можливо змінювати стани елементів веб-сторінок в реальному часі, обробляти події користувачів (кліки, наведення, виділення тощо), валідувати форми та інтегрувати API (Application Programming Interface). JS надає можливість використовувати різні бібліотеки та фреймворки для розробки більш складного функціоналу, підтримується у браузерах та забезпечує динамічність системи. Проте за відсутності оптимізованих підходів під час написання коду, можлива втрата продуктивності системи та виникнення проблем зі сторони безпеки.

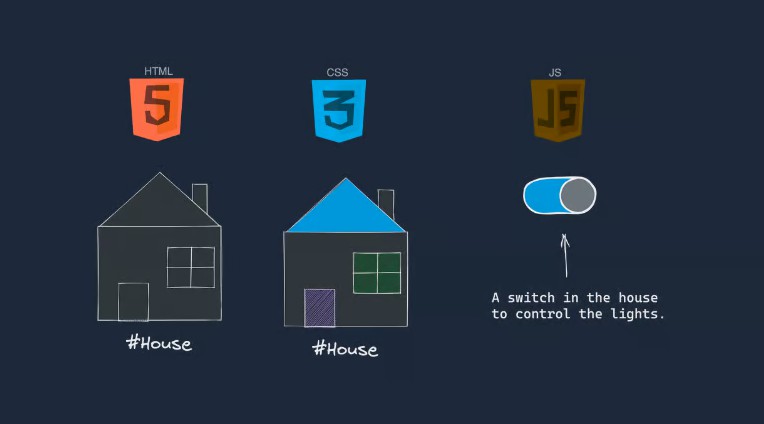


Рис 2.1.4. Візуалізація ролі HTML, CSS та JS при створенні веб-контенту Окрім використання базових інструментів, при розробці сучасних проєктів

використовують фреймворки і бібліотеки для оптимізації та пришвидшення процесу розробки[7].

Найпопулярнішими додатковими інструментами при розробці клієнтської частини є[5, 6]:

1. React.js: відкрита JS бібліотека для створення інтерфейсів користувача. React надає можливість «розбити» створювану систему на компоненти для оптимізації функціональності продукту та можливості перевикористання на різних веб-сторінках системи. Також ця бібліотека підтримує віртуальний DOM (Document Object Model), що дозволяє забезпечити швидкий рендеринг компонентів і швидше завантажувати

контент. Перевага цієї бібліотеки в тому, що її можливо інтегрувати в інші системи, використовуючи великий набір внутрішніх бібліотек та розширень. Проте є і недоліки при використанні бібліотеки React: відсутній вбудований функціонал для роботи з сервером та маршрутизацією, постійне оновлення в структурі та документації бібліотеки[30].

1. Angular: фреймворк односторінкових веб-додатків, який надає широкий функціонал для розробки систем. Angular забезпечує надійний двосторонній зв’язок даних, надає вбудовані інструменти для роутингу, HTTP-клієнта та підтримки форм, дозволяє впровадити модульність коду для полегшення тестування. Перевагою цього фреймворку є базування на мові програмування TypeScript, що розширює базові можливості JavaScript.
2. Vue.js: JS-фреймворк, що використовує шаблон MVVM (Model-View- View-Model) для створення інтерфейсів через реактивне зв’язування даних. Такий підхід забезпечує гнучкість та інтерактивність системи. На відміну від інших фреймворків, Angular потребує використання окремих розширень для реалізації складних функцій.

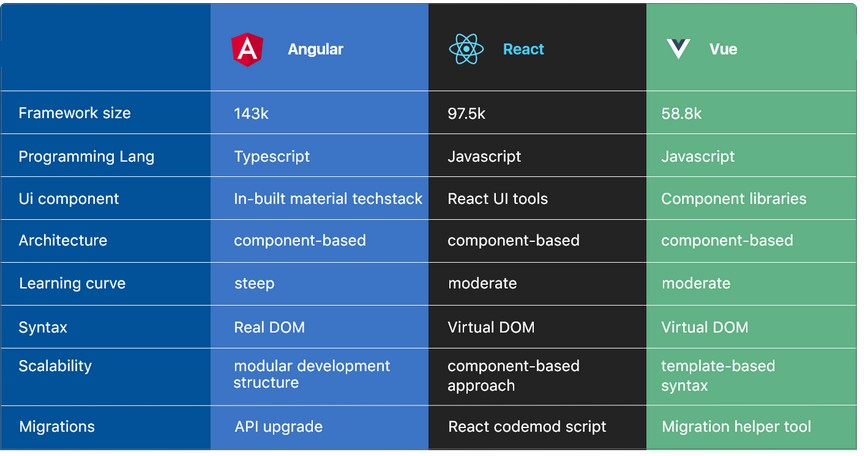


Рис 2.1.5. Порівняння характеристик React.js, Angular та Vue.js

1. Bootstrap: бібліотека з набором інструментів для створення веб-сервісів, яка містить шаблони HTML та CSS. Bootstrap надає готові компоненти для розробки інтерфейсу, що полегшує процес розробки.
2. jQuery: JavaScript-бібліотека, що надає інструменти для взаємодії з DOM,

анімаціями, обробки подій та роботи з Ajax.

Задля забезпечення продуктивності та ефективності системи з врахуванням оптимізації ресурсів, управління попитом та уникнення колізій використовуються різні методології[21]:

1. Мініфікація – процес зменшення розміру вихідного коду, шляхом видалення зайвих символів (коментарі, пробіли, переноси тощо).
2. Bundling (об’єднання файлів) – процес об’єднання імпортованих файлів в один файл-бандл.
3. Tree Shaking – видалення з коду невикористаних компонентів та елементів.
4. Застосування асемблерів та менеджерів пакетів. Застосування асемблерів (наприклад, Webpack, Parcel, Rollup) допомагає оптимізувати розподіл ресурсів та пришвидшити роботу системи.
5. Lazy loading – шаблон проєктування, що полягає у відтермінованому завантаженні необхідних ресурсів за вимогою. Такий метод дозволяє уникнути перевантаження системи.
6. Кешування ресурсів – процес зберігання раніше завантаженої інформації в локальних сховищах, що дозволяє уникнути колізій пов’язаних з перевантаженням системи та відсутності даних.
7. CDN (Content Delivery Network) – використання географічно розподіленої мережі для відправки запитів на сервер, що допомагає швидко доставити статичні файли до сервера та оптимізувати розподіл ресурсів.
8. Preloading – попереднє завантаження необхідних даних до початку відображення контенту дозволяє уникнути колізій з затримками обробки запитів.
9. Використання браузерних інструментів для аналізу продуктивності. Сучасні браузери надають можливість встановлення плагінів (наприклад, Google Lighthouse, PageSpeed Insights, WebPageTest тощо), які будуть проводити аудит продуктивності системи, виявляти недоліки та надавати рекомендації щодо оптимізації.
10. Critical Rendering Path Optimization – набір методів, які виконує браузер, для зменшення часу рендерингу компонентів шляхом зменшення об’єму CSS та JS.
11. Code Splitting – процес розмежування великих файлів JavaScript в проєкті на менші модулі, що дозволяє розставити пріоритетність ресурсів, що використовуються.
12. Використання Service Workers – скрипт, який дозволяє завантажувати дані в фоновому режимі та, завдяки кешуванню, забезпечувати доступ до даних без наявності підключення до Інтернету.
13. CSP (Content Security Policy) – стандарт комп’ютерної безпеки, що передбачає контроль над тим, які скрипти та ресурси дозволено завантажувати.
14. Забезпечення адаптивності системи – використання динамічних стилей, різних розмірів та форматів зображень для мобільних і десктопних пристроїв.

Для розрахунку продуктивності системи можна скористатися формулами, що надають розуміння щодо швидкості завантаження сторінок та ефективності кешування даних[21]:

де TDNS – час відгуку DNS (Domain Name System), Tc – час встановлення з’єднання, Tw – час очікування відповіді, Td – час завантаження.

де Cr – кількість запитів, які беруться з кешу, Ca – кількість всіх запитів.

## Методологія вирішення колізій при клієнт-серверній архітектурі систем онлайн-бронювання

Клієнт-серверна архітектура застосовується у багатьох сучасних системах онлайн-бронювання. При розробці таких систем важливо забезпечити чітке розподілення задач між клієнтською та серверною частинами. Підбираючи найбільш підходящі інструменти, методи та технології розробки, необхідно враховувати, що особливості цих двох складових. Клієнтська частина в основному відповідає за взаємодію з користувачами і не має бути перенасичена складними логічними функціями, а серверна частина забезпечує обробку запитів та управління базами даних і не має містити зайву логіку виведення інформації на клієнтську частину.

Загалом можна виділити декілька методів забезпечення механізмів уникнення колізій на стороні клієнта та сервера[24,25]:

1. Блокування ресурсів на сервері:
   * полягає у блокуванні ресурсів на стороні сервера для уникнення одночасного доступу декількома користувачами. Коли один користувач робить бронювання, сервер блокує відповідний ресурс для інших користувачів до тих пір, поки перший користувач не завершить процес бронювання;
   * можливе застосування різних стратегій блокування: Песимістичне блокування – на період бронювання користувачам блокується доступ до ресурсу. Такий метод підходить для систем з обмеженими ресурсами, коли кожен запит потребує спочатку підтвердження збоку сервера. Оптимістичне бронювання – перед фінальним підтвердженням бронювання сервер перевіряє чи відбулися зміни

даних, але ресурс під час бронювання не блокується. Цей метод знижує ризик виникнення конфліктів блокування, але збільшує ймовірність колізій пов’язаних з дублюванням.

* + наприклад, при використанні бази даних MySQL можна використовувати механізм транзакцій з блокуванням:

START TRANSACTION;

SELECT \* FROM bookings WHERE resource\_id = 1 FOR UPDATE;

-- функціонал створення або скасування бронювання

COMMIT;

1. Використання черг:
   * коли декілька користувачів бронюють однаковий ресурс, їхні запити потрапляють в чергу за часом надходження на сервер. Такий метод допомагає досягнути прозорості та справедливості розподілення ресурсів, а також уникнути колізій пов’язаних з перевантаженням системи;
   * реалізувати метод можливо за допомогою спеціалізованих інструментів (наприклад, RabbitMQ, Apache Kafka, Redis), які організовують потоки даних та обмежують доступ до одного ресурсу з декількох джерел;
   * наприклад, реалізація черг в RabbitMQ:

// Запит потрапляє у чергу channel.sendToQueue('booking\_request', Buffer.from(JSON.stringify(request)));

// Сервер обробляє запити по черзі channel.consume('booking\_request', function(msg) { const booking = JSON.parse(msg.content.toString());

// Обробка бронювання

channel.ack(msg);

});

1. Кешування та оновлення статусу ресурсів:
   * кешування дозволяє оперативно перевіряти статус ресурсу перед бронюванням. Якщо ресурс вже заброньований, при спробі повторного бронювання на клієнтську частину буде виводитися повідомлення щодо недоступності ресурсу і зайвий запит на сервер не буде відправлятися;
   * для реалізації кешування можна використовувати Redis, Memcached та

Varnish;

* + наприклад, зберігання статусу ресурсу в Redis:

// Збереження статусу

redis.set('Hotel room 1', 'booked', 'EX', 3600);

// Перевірка статусу перед бронюванням const status = await redis.get('Hotel room 1'); if (status === 'booked') {

// Логіка виводу повідомлення користувачеві

}

1. Створення інтерфейсу керування попитом:
   * на клієнтській частині можлива реалізація механізмів контролю навантаження та моніторингу попиту, наприклад, шляхом обмеження запитів або відкладення їхньої відправки на сервер;
   * даний механізм може бути досягнутий застосуванням методів Throttling та Debouncing з бібліотеки Lodash. Throttling може забезпечити обмеження кількості запитів від одного користувача за певний період часу. Debouncing може допомогти передбачити перевантаження системи шляхом відкладення виконання логіки обробки запиту на певний період часу;
   * наприклад, використання методу Throttling:

const throttle = require('lodash.throttle');

const sendBookingRequest = throttle(function(request) {

// Логіка обробки запиту на бронювання

}, 1000); // Обмеження надсилання запиту (максимум 1 запит на секунду)

## Висновок до розділу 2

В розділі 2 було розглянуто використання інструментів, методологій та технологій, які забезпечують механізми уникнення колізій, оптимізації ресурсів та управління попитом і можуть бути використані при розробці систем онлайн- бронювання. Особливу увагу було приділено аналізу методів, технологій та інструментів для забезпечення ефективної взаємодії клієнт-серверної моделі побудови і забезпечення стабільної роботи системи онлайн-бронювання.

Було розглянуто ключові аспекти забезпечення оптимізації ресурсів з боку клієнтської сторони з використанням таких інструментів як HTML, CSS, JS, React.js, Bootstrap тощо, а також з боку серверної сторони – PHP, MySQL, Redis, Docker, JWT тощо. Проаналізовано методи застосування транзакцій, блокувань, кешування та черг запитів для вирішення проблем колізій. Також було розглянуто питання забезпечення безпеки конфіденційних даних під час процесу бронювання і проаналізовано такі методи як шифрування даних, протоколи авторизації та аутентифікації.

Аналіз всієї інформації показав, що для створення сучасної, надійної та продуктивної системи онлайн-бронювання, необхідно ретельно обирати методи, технології, інструменти та стратегії, які будуть забезпечувати уникнення колізій, оптимізацію ресурсів та можливості управління попитом.

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОНЛАЙН- БРОНЮВАННЯ

Розроблена система онлайн-бронювання забезпечує зручний та продуктивний процес онлайн-бронювання земельних ділянок та заміських будинків. Основна мета даної системи – забезпечення доступності інформації про вільні ділянки/будинки для бронювання, мінімізувати колізії при бронюванні та вдосконалити використання ресурсів шляхом використання інструментів управління попиту.

## Використані інструменти

Для розробки системи було використано мову програмування PHP для реалізації логіки на серверній стороні, MySQL для зберігання інформації про користувачів, бронювання та події системи в базі даних, JavaScript, HTML та CSS для розробки інтерфейсу користувача, забезпечення динамічного завантаження системи та управління станами доступності ресурсів в реальному часі.

При розробці системи онлайн-бронювання було застосовано стек LAMP для забезпечення ефективності та надійності системи. Для зручного розгортання локального сервера для розробки використано комплекс програмного забезпечення XAMPP. XAMPP містить інструменти для розробки та тестування веб-продуктів локально на комп’ютері без розгортання на хостингу.

XAMPP складається з веб-серверу Apache, систем керування базами даних MySQL та MariaDB, мов програмування PHP та Perl, системи адміністрування баз даних phpMyAdmin. Завдяки цим інструментам XAMPP надає можливість запускати проєкт для розробки або тестування та оновлювати його в режимі реального часу без доступу до Інтернету. Ще одною перевагою XAMPP є можливість за потреби вмикати та вимикати вбудовані компоненти (наприклад, Apache та MySQL).

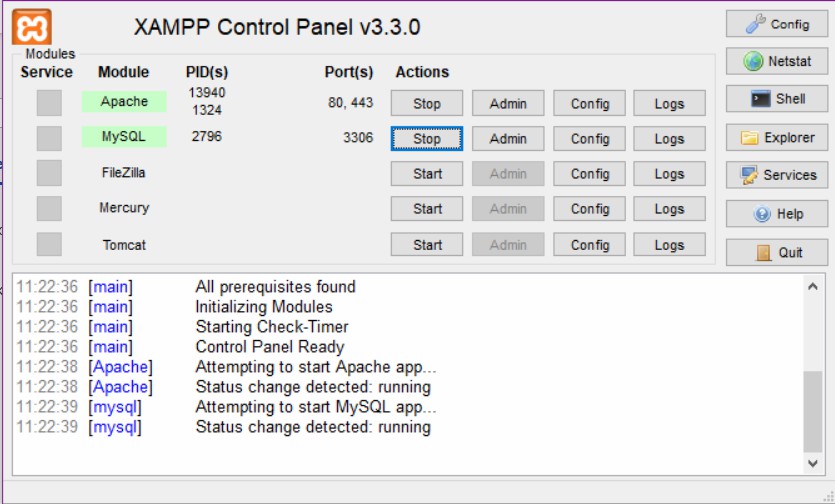


Рис 3.1.1. Інтерфейс XAMPP, де запущено компоненти Apache та MySQL

Для роботи з базою даних використовувалась система phpMyAdmin. Цей веб-додаток написаний на мові PHP та надає можливість використовувати інтерфейс для адміністрування баз даних MariaDB та MySQL. phpMyAdmin є альтернативою використанню команд в терміналі і спрощує роботу з базами даних, візуалізуючи всі необхідні інструменти. Основними перевагами phpMyAdmin є:

* можливість створення, редагування, видалення та забезпечення резервних копій баз даних;
* імпорт та експорт баз даних у форматах SQL, CSV, XML;
* керування таблицями з можливістю модифікування стовпців, індексів та ключів;
* збереження попередніх запитів та можливість їх повторного використання;
* наявність інструментів для проведення SQL-запитів вручну;
* можливість створення облікових записів користувачів та налаштувань прав доступу з бази даних;
* наявність інструментів для перегляду статистики використання системи, оптимізації таблиць та баз даних;

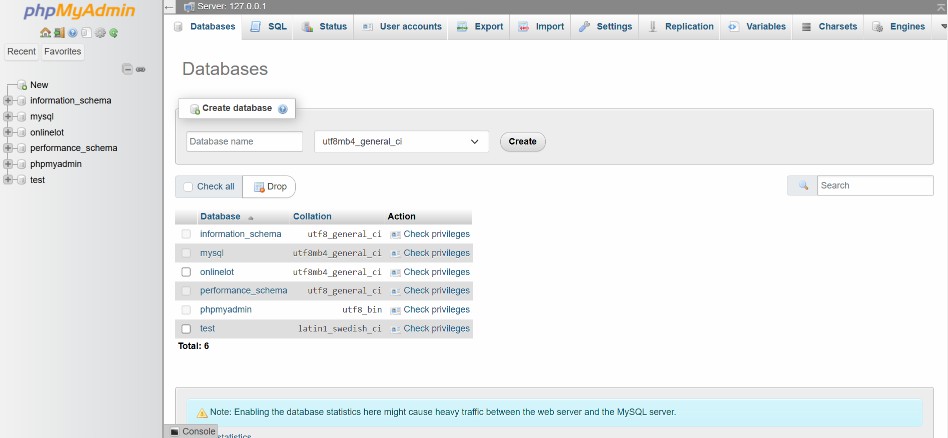


Рис 3.1.2. Інтерфейс phpMyAdmin

## Основний функціонал системи

При відкритті веб-сторінки системи онлайн-бронювання, користувач бачить головну сторінку з короткою інформацією про систему.

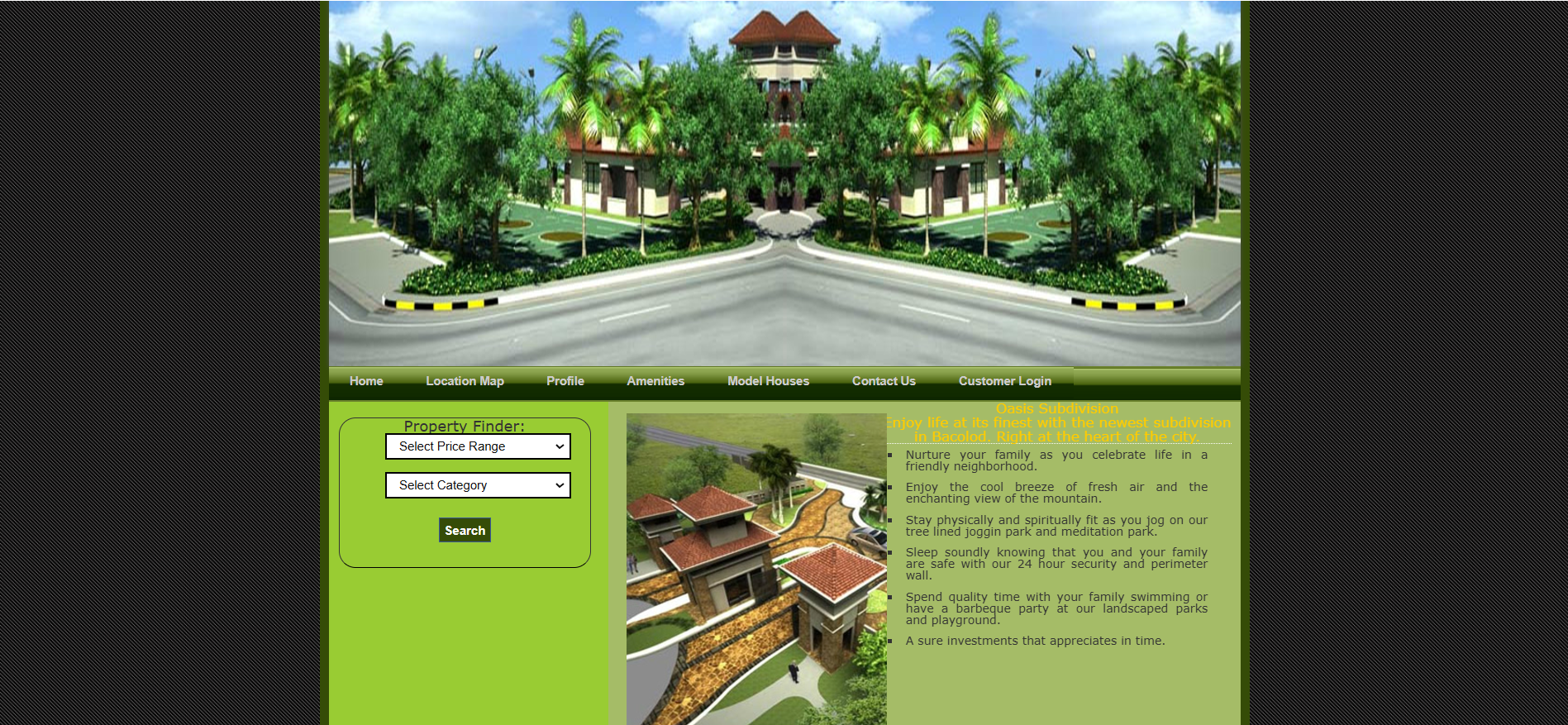


Рис 3.2.1 Початкова сторінка системи онлайн-бронювання Також з головної сторінки користувач може знайти ділянки та будинки,

відфільтрувавши пошук за ціною та типом. Використання компоненту пошуку з елементами фільтрації дозволяє надати користувачу вибір пропозицій за його можливостями та потребами, що знижує ризик виникнення колізій, пов’язаних з перевантаженням системи.

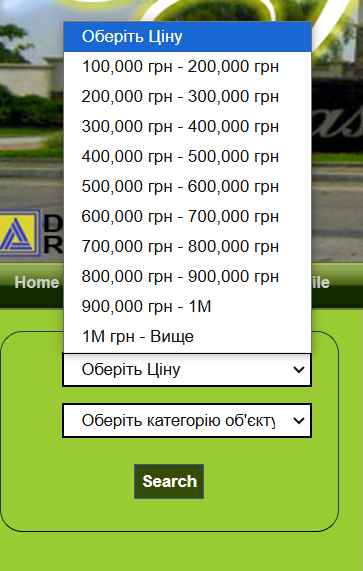


Рис 3.2.2. Пошук за ціною та об’єктом бронювання Пошук для будинків реалізовано наступним шляхом:

<?php

require ("include/dbconnection.php");

$range=$\_POST['range'];

if (isset($\_POST['search'])) {

if ($range=="Оберіть ціновий діапазон") {

echo "<font size=4 color=red>Оберіть ціновий діапазон та об’єкт бронювання!<font>";

} else if ($\_POST['range']=="Оберіть категорію") {

echo "<font size=4 color=red>Оберіть ціновий діапазон та об’єкт бронювання!</font>";

}

}

if ($\_POST['category']=="Будинок" and $range=='1') {

$result = mysqli\_query($conn, "SELECT \* FROM house WHERE costprice<200000 and

costprice>=100000 ORDER BY house\_id");

if (mysqli\_num\_rows($result)>0) {

while ($row=mysqli\_fetch\_array($result))

{

?>

<div class="search">

<img src="<?php echo $row['houseimg']; ?>"width=180 height=150 alt='Помилка відображення' style='margin-left: 17px; margin-top: 27px;' />

<div class="searchresult">

Назва будинку: <?php echo $row['name']; ?><br /> Тип будинку: <?php echo $row['type']; ?><br /> Модель будинку: <?php echo $row['model']; ?><br /> Опис: <?php echo $row['description']; ?><br /> Розташування: <?php echo $row['lotarea']; ?><br />

Кількість поверхів: <?php echo $row['floorarea']; ?><br />

Ціна оренди:&nbsp;<font color="#FF0000"> $</font>&nbsp; <?php echo number\_format($row['price'],2); ?><br />

Ціна (для купівлі):&nbsp;<font color="#FF0000"> $</font>&nbsp; <?php echo number\_format($row['costprice'],2); ?><br />

<a href="qoutation.php?houseid= <?php echo $row['house\_id']; ?>">Детальніше</a>

</div>

</div><br />

Пошук для ділянок реалізовано логікою:

<?php

if ($\_REQUEST['category']=="Ділянка" and $range=="1" ) {

$result = mysqli\_query($conn, "SELECT \* FROM lot WHERE tcp<200000 and tcp>=100000 ORDER BY lot\_id");

if (mysqli\_num\_rows($result)>0){

while($row=mysqli\_fetch\_array($result))

{

$status=$row['lotstatus'];

?>

<div class="search">

<div class="searchresult"><h6>

Назва ділянки: <?php echo $row['subdname']; ?><br />

Зона No.: <?php echo $row['phase']; ?><br /> Блок No.: <?php echo $row['block']; ?><br /> Ділянка No.: <?php echo $row['lotno']; ?><br />

Розташування: <?php echo $row['lotarea']; ?><br />

Тип ділянки: <?php echo $row['class']; ?><br />

Ціна оренди:&nbsp;<font color="#FF0000"> $</font>&nbsp; <?php echo number\_format($row['price'],2); ?><br />

Ціна для купівлі:&nbsp;<font color="#FF0000"> $</font>&nbsp; <?php echo number\_format($row['tcp'],2); ?><br />

Після сортування за обраними фільтрами, користувачу виводиться список доступних до бронювання об’єктів та коротка інформація про ціни, розташування, вигляд тощо. При натисканні на кнопку «Детальніше» відкривається нова сторінка з більш розгорнутою інформацією про об’єкт бронювання. Також навпроти кожного доступного варіанту є кнопка «Забронювати», яка переводить користувача на сторінку бронювання ділянки/будинку.

На сторінці бронювання користувач має ввести свої дані для оформлення бронювання, також при переході на сторінку бронювання автоматично заповнюється інформація щодо обраного об’єкту.

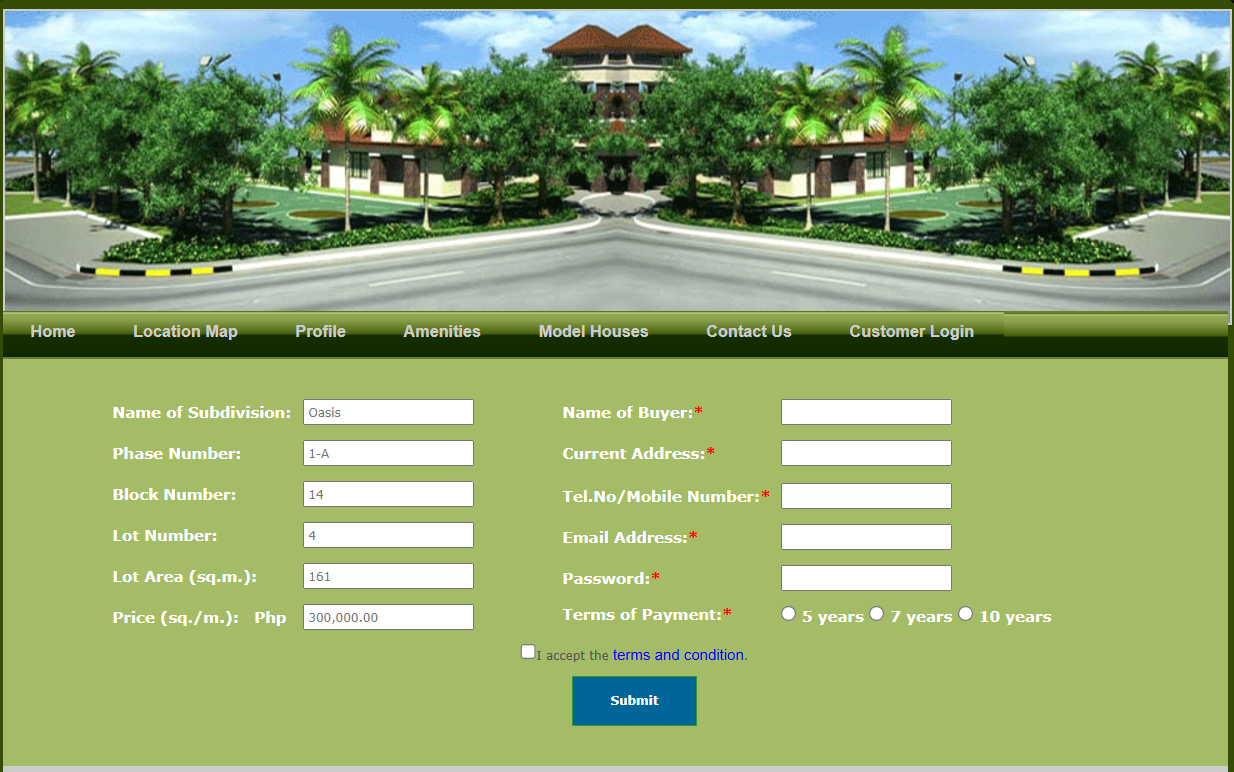


Рис 3.2.3. Сторінка бронювання земельної ділянки або заміського будинку При натисканні кнопки «Завершити бронювання» з’являється повідомлення

про успішне бронювання та запит відправляється до бази даних. Якщо якісь поля не заповнені, біля незаповнених полів з’являється попередження, що вони обов’язкові до заповнення.

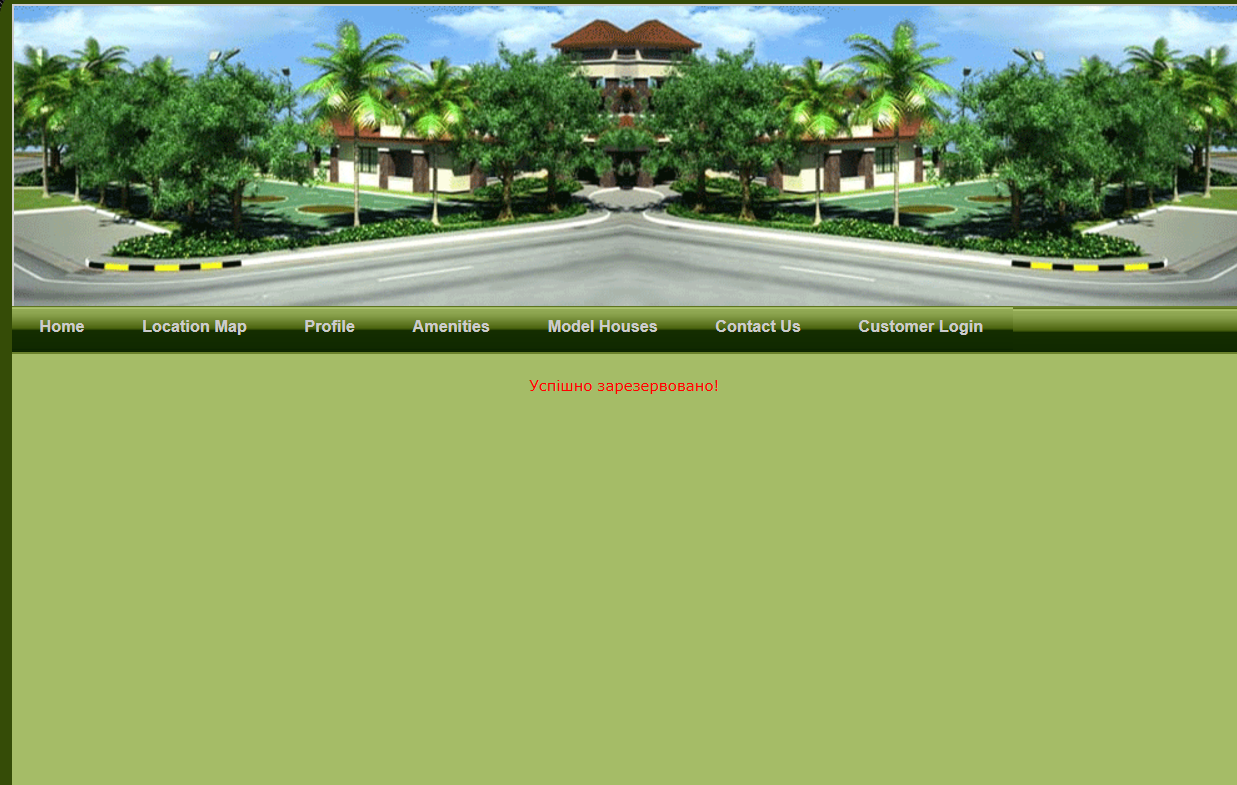


Рис 3.2.3. Успішне створення бронювання

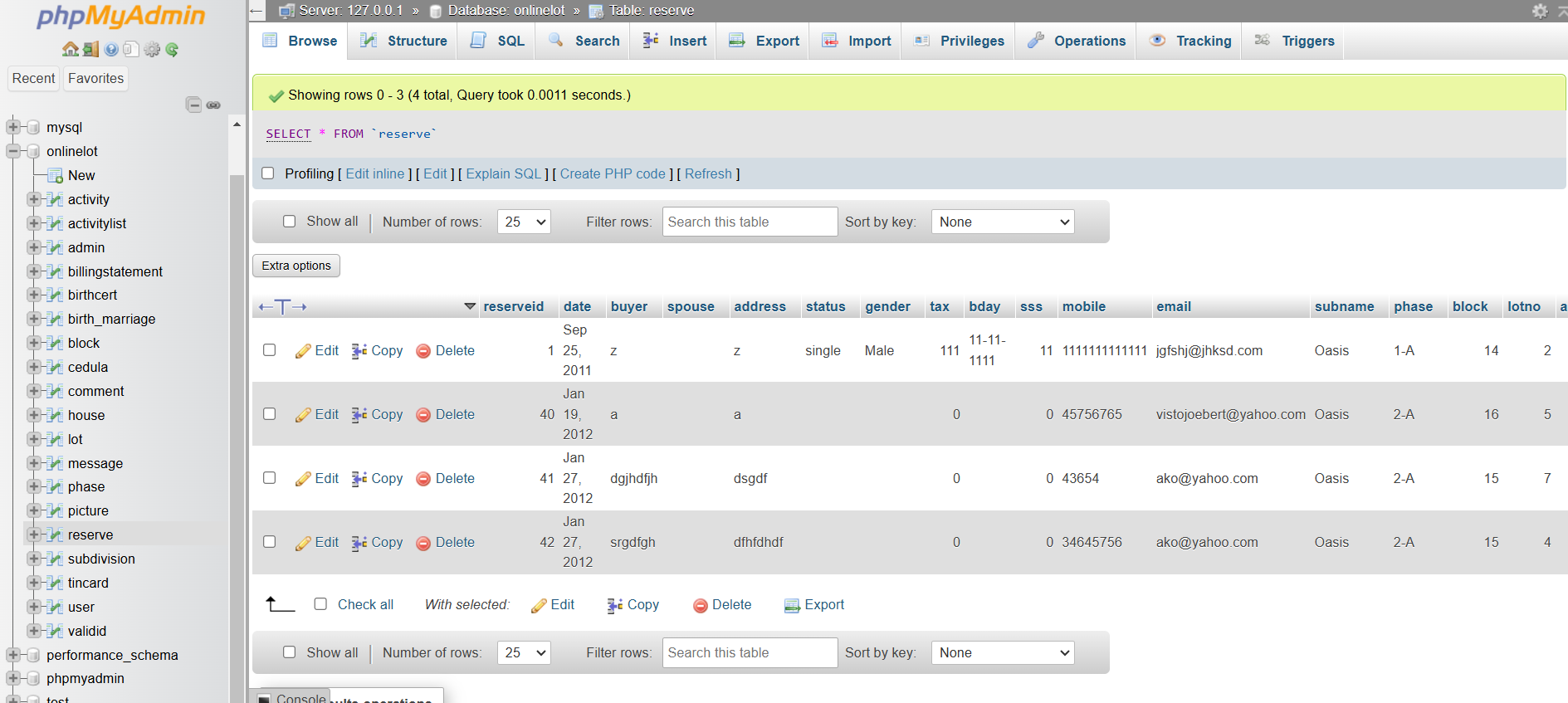


Рис 3.2.4 Створення запиту в базі даних

Для забезпечення доступності наявних ресурсів та уникнення колізій пов’язаних з дублюванням бронювань, після бронювання об’єкт блокується. Також користувачі мають можливість переглянути доступні до бронювання об’єкти в розділі «Розташування», де знаходиться карта місцевості. При кліці на певну область відкривається карта вільних слотів, які можна забронювати натиснувши на вільний слот.

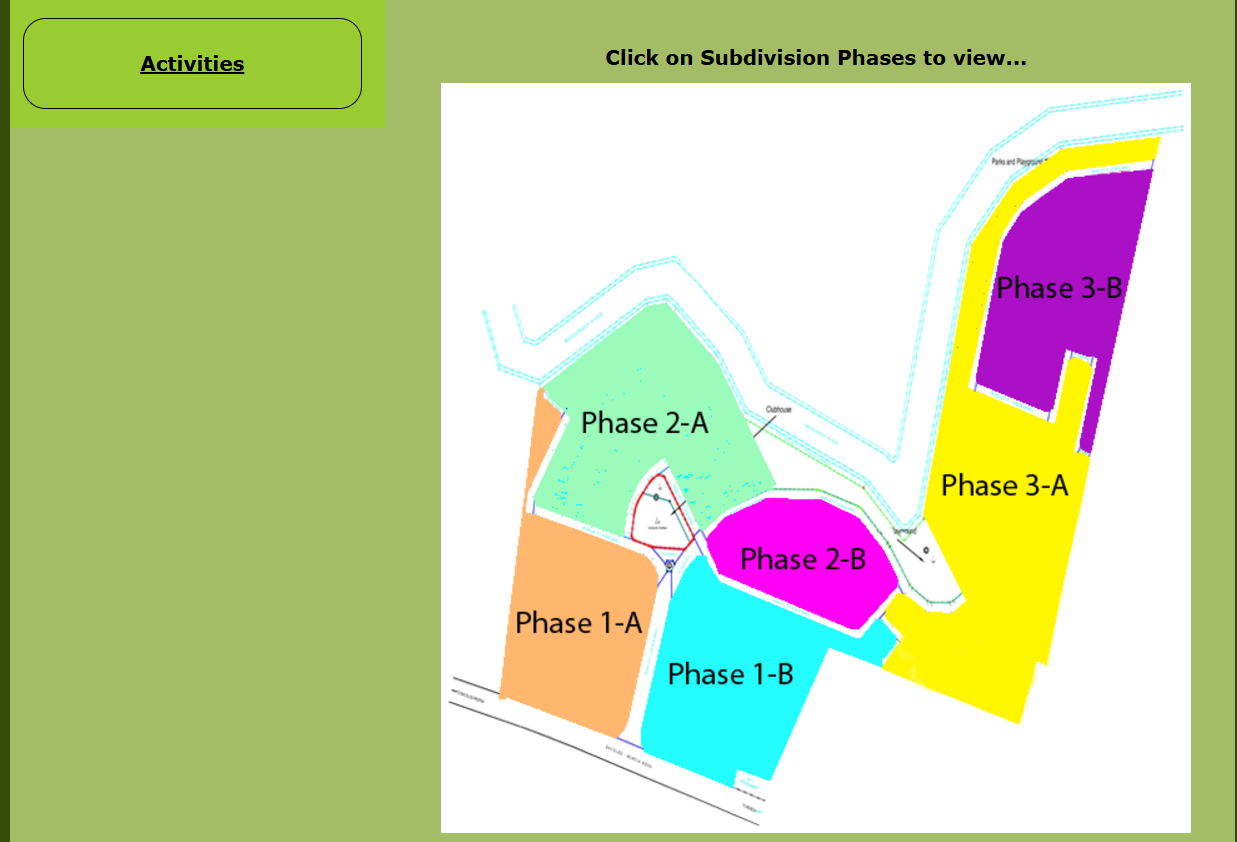


Рис 3.2.5 Карта місцевості з доступними регіонами бронювання

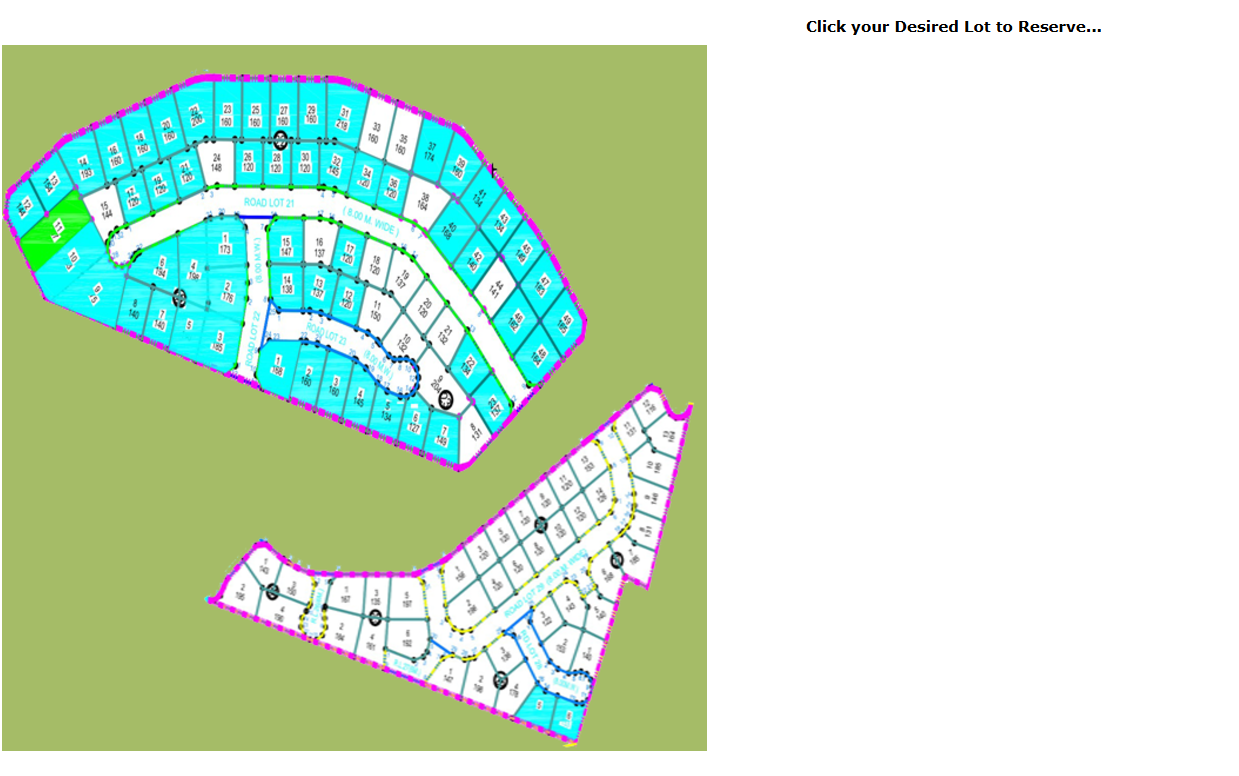


Рис 3.2.6 Карта слотів для бронювання

Програмна реалізація процесу оформлення бронювання виглядає наступним чином:

<?php

session\_start();

include("include/dbconnection.php");

$phase=$\_REQUEST['phase'];

$block=$\_REQUEST['block'];

$lotno=$\_REQUEST['lotno'];

$name=$\_REQUEST['name'];

$lotarea=$\_REQUEST['lotarea'];

$price=$\_REQUEST['price'];

$errors = '';

if(isset($\_POST['submit']))

{

if(empty($errors))

{

$terms=$\_POST['terms'];

$date=$\_POST['date'];

$buyer=$\_POST['buyer'];

$address=$\_POST['address'];

$password=$\_POST['password'];

$mobile=$\_POST['mobile'];

$subname=$\_POST['subname'];

$phase=$\_POST['phase'];

$block=$\_POST['block'];

$lotno=$\_POST['lotno'];

$lotarea=$\_POST['lotarea'];

$price=$\_POST['price'];

$totalprice=$\_POST['costprice'];

$discount=$\_POST['discount'];

$downbal=$\_POST['downbal'];

$amortbal=$\_POST['amortbal'];

$amort=$\_POST['amort'];

if($email!='') {

$result = mysqli\_query($conn, "SELECT \* FROM `user` WHERE email='$email' ");

$numberOfRows = mysqli\_num\_rows($result);

if (!$result) {

die("Помилка, спробуйте пізніше!");

}

if ($numberOfRows > 0) {

$qry="INSERT INTO reserve (date, buyer, spouse, address, status, gender, mobile, email, name, phase, block, lotno, area, price, amortbal, amort,ccode) VALUES ('$date','$buyer','$spouse','$address','$status','$gender','$mobile','$email','$name'

, '$phase', '$block', '$lotno', '$lotarea', '$price', '$terms', '$discount', '$discount','$amortbal', '$amort','$ccode')";

$result = @mysqli\_query($connection, $qry) ;

if($result) {

header("location: reservationsuccess.php");

exit();

}else {

die("Query failed");

}

}

} else if($numberOfRows == 0) {

$sql="INSERT INTO

user(name,address,gender,bday,email,password)VALUES('$buyer','$address','$gender','$b day','$email','$password')";

$qry="INSERT INTO reserve (date, buyer, spouse, address, status, gender, mobile, email, name, phase, block,lotno, area, price, amortbal, amort,ccode) VALUES ('$date','$buyer','$spouse','$address','$status','$gender','$mobile','$email','$name'

, '$phase', '$block', '$lotno', '$lotarea', '$price','$amortbal', '$amort','$ccode')";

$result = @mysqli\_query($connection, $qry) and @mysqli\_query($connection,

$sql) ;

if($result) {

header("location: reservationsuccess.php");

exit();

} else {

die("Помилка");

}

}

}

}

?>

Для оптимізації ресурсів на сторінці веб-системи онлайн-бронювання є розділ з переліком та описом наявних в системі типів земельних ділянок та заміських будинків, де користувачі можуть ознайомитись з доступними варіантами бронювання.

Також в системі наявна система авторизації. Користувач може перейти у вкладку «Вхід» ввести свій пароль та логін і попасти на сторінку користувача, де буде інформація щодо попередніх бронювань, персональних знижок тощо.

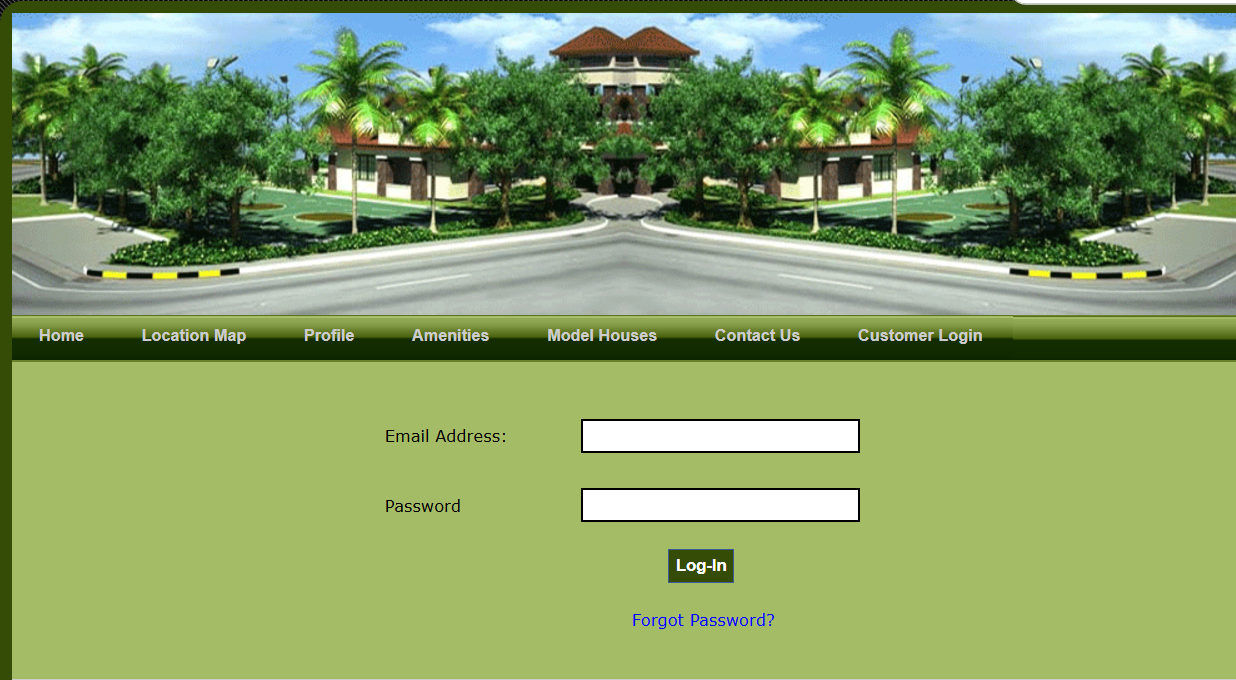


Рис 3.2.7. Вхід в обліковий запис користувача

## Перспективи адаптації розробленої системи онлайн-бронювання під використання в інших галузях.

Завдяки використанню динамічних інструментів розробки система онлайн- бронювання земельних ділянок та заміських будинків може бути адаптована під вимоги різних сфер:

1. Туристична галузь:
   * бронювання отельних номерів: можливе використання основ вже створеної системи з розширенням функціоналу для бронювання номерів, встановлення сезонних знижок та адаптації цінових категорій;
   * бронювання екскурсій: можливе використання прописаної бази з інтеграцією календарів для планування екскурсій та забезпечення великої кількості бронювання одного ресурсу;
   * оренда транспорту: можливе видозмінення системи шляхом додавання до бази даних транспортних засобів та розділів для управління автопарком;
2. Медицина:
   * запис на прийом до лікарів: можливе додавання функціоналу, який дозволить користувачам керувати часом запису та уникати перетинів прийомів;
   * організація ресурсів клініки: можливе видозмінення систем шляхом додавання баз даних постачальників медичного обладнання;
3. Спортивна сфера:
   * бронювання залів та тренажерів: можливе розширення системи з додаванням функціоналу вдосконаленого відстеження доступності ресурсів в реальному часі;
   * запис на тренування: можлива інтеграція системи з сучасними технологіями керування календарями і часом для забезпечення точної синхронізації системи з базами даних;
4. Роздрібна та оптова торгівля: можлива адаптація системи під каталоги товарів продавців;
5. Логістична сфера:
   * резервування місць на вантажівках чи контейнерах: можливе забезпечення модулів для перевірки доступності ресурсів на певні періоди часу;
   * оптимізація місць на складах: можлива адаптація системи під бронювання місць на складах із урахуванням характеристик товарів, що будуть зберігатися на складах;

## Висновки до розділу 3

В даному розділі було розглянуто реалізовану систему онлайн-бронювання земельних ділянок та заміських будинків, з урахуванням забезпечення механізмів уникнення різних типів колізій, оптимізації ресурсів та управління попитом. Завдяки впровадженій клієнт-серверній архітектурі система продемонструвала можливості масштабування та гнучкість використання функціоналу.

Розроблена система онлайн-бронювання може здійснювати перевірку доступності ресурсів для бронювання, усувати конфлікти під час процесів бронювання за допомогою методів обробки запитів та має механізм автоматизованих процесів управління ресурсами та попитом.

Універсальність використаних методів, технологій та інструментів дозволяє адаптувати систему під різні галузі, такі як медицина, туризм, логістика, торгівля, спорт тощо.

Можна зробити висновок, що система онлайн-бронювання дотримується принципів ефективного усунення колізій та демонструє потенціал для розширення функціоналу з можливістю інтеграції більш складних бізнес-логік в різних сферах бронювання.

# ВИСНОВОК

У рамках виконаної роботи було проведено дослідження теоретичних аспектів, технологій, методів та інструментів, що забезпечують ефективну, оптимізовану, продуктивну та надійну основу для створення систем онлайн- бронювання.

Було проаналізовано історичність виникнення систем онлайн-бронювання, починаючи від започаткувань до сьогоднішніх сучасних технологій, які допомагають якісно використати підходи до оптимізації ресурсів та управління попитом. Протягом дослідження було з’ясовано природу виникнення колізій у системах онлайн-бронювання, визначено типи колізій та можливі механізми їх вирішення. Аналіз існуючих систем онлайн-бронювання дозволив визначити можливі проблеми та слабкі сторони при розробці система, а також сформувати позитивні практики та надійні механізми проєктування і розробки систем онлайн- бронювання.

Особлива увага була присвячена огляду існуючих інструментів, технологій та методологій для створення надійної системи онлайн-бронювання та уникнення конфліктів різної природи виникнення. Було детально розглянуто методологію клієнт-серверної архітектури та визначено найбільш ефективні інструменти та технології для забезпечення уникнення колізій.

На основі всіх проаналізованих даних було розроблено систему онлайн- бронювання, яка містить в собі механізми попередження колізій, адаптації ресурсів до змін попиту та механізмів управління базами даних. Зокрема було реалізовано основний функціонал управління ресурсами, забезпечення процесу бронювання та перевірки доступності. Створена система продемонструвала інтеграційну здатність, що розширює можливість її запровадження у різних галузях.

Таким чином, протягом роботи було вирішено поставлені завдання та створено систему, яка відповідає сучасним вимогам до онлайн-бронювання,

мінімізує ризики виникнення колізій, забезпечує ефективний розподіл ресурсів та має потенціал для практичного застосування та подальшого розвитку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* + 1. Eevert Kuusamo. The development of new and improved online booking system for small-scale IT maintenance company: A Case of Tietsikkari. URL: https://[www.theseus.fi/bitstream/handle/10024/744151/Development%20of%20new](http://www.theseus.fi/bitstream/handle/10024/744151/Development%20of%20new)

%20and%20improved%20online%20booking%20system%20for%20a%20small- scale%20IT%20maintenance%20company%20A%20case%20of%20Tietsikkari%20 Eevert%20Kuusamo.pdf?sequence=2 (дата звернення: 21.10.2024).

* + 1. MDN Web Docs. MDN Web Docs. URL: https://developer.mozilla.org/ru/ (дата звернення: 24.10.2024).
    2. 19 HTML5 Advantages and Disadvantages. BrandonGaille.com. URL: https://brandongaille.com/19-html5-advantages-and-disadvantages/ (дата звернення: 21.10.2024).
    3. Using HTML5, CSS3, and JavaScript to their full potential. C# Corner - Community of Software and Data Developers. URL: https://www.c- sharpcorner.com/article/using-html5-css3-and-javascript-to-their-full-potential/ (дата звернення: 21.10.2024).
    4. Web Development Frameworks: A Guide | Built In. Built In. URL: https://builtin.com/articles/web-development-frameworks (дата звернення: 24.10.2024).
    5. Web Development Frameworks Comparison: [Updated List]. PixelCrayons. URL: https://[www.pixelcrayons.com/blog/software-development/best-web-](http://www.pixelcrayons.com/blog/software-development/best-web-) development-frameworks-comparison/ (дата звернення: 24.10.2024).
    6. Most Popular JavaScript Frameworks and Libraries Today. Full Scale. URL: https://fullscale.io/blog/most-popular-javascript-frameworks-and-libraries/ (дата звернення: 25.10.2024).
    7. 12 Best Web Development Frameworks (Front-end & Back-end). WsCube Tech Blog. URL: https://[www.wscubetech.com/blog/web-development-frameworks/](http://www.wscubetech.com/blog/web-development-frameworks/) (дата звернення: 01.11.2024).
    8. Vidjikant S. Why Use PHP in 2024? Advantages and Disadvantages - Softjourn. Softjourn Inc. URL: https://softjourn.com/insights/pros-and-cons-of-php- programming-language (дата звернення: 02.11.2024).
    9. Microsoft. Why Visual Studio Code?. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/Docs/editor/whyvscode (дата звернення: 10.11.2024).
    10. FreeCodeCamp. How to Set Up VS Code for Web Development in A Few Simple Steps. FreeCodeCamp.org. URL: https://[www.freecodecamp.org/news/how-to-set-up-vs-code-for-web-](http://www.freecodecamp.org/news/how-to-set-up-vs-code-for-web-) development/ (дата звернення: 10.11.2024).
    11. PHP Web Development - The Ultimate Guide - Riseup Labs. Riseup Labs. URL: https://riseuplabs.com/php-web-development-ultimate-guide/ (дата звернення: 24.11.2024).
    12. Бурячок В. Л. Технології забезпечення безпеки мережевої інфраструктури. / В. Л. Бурячок, А. О. Аносов, В. В. Семко, В. Ю. Соколов, П. М. Складанний. – К.: КУБГ, 2019. – 218 с.
    13. Методи виявлення і вирішення колізії. StudFiles. URL: https://studfile.net/preview/5207047/page:2/ (дата звернення: 22.10.2024).
    14. Базові поняття мережевих технологій [Інтернет-ресурс] / Web-сайт: mobiz.com.ua; URL: https://mobiz.com.ua/bazovi-poniattiamerezhevykh- tekhnolohij.html. (дата звернення: 22.10.2024).
    15. Попов О. С. (ред.). Управління витратами підприємства: навч. посібник. Київ: КПІ ім. Ігоря Сікорського, 2020
    16. Ma, J., Li, L., & Zhuang, H. Optimization Techniques for Demand Management in the Supply Chain. Academic Journals, 2023\
    17. Lombardi, M., Cesarotti, V., & Benedetti, C. Demand Forecasting and Production Planning for the Smart Grid. Springer, 2021
    18. Exploring the Power of PHP: Use Cases and Best Practices - Imenso Software. *Imenso Software*.

URL: https://[www.imensosoftware.com/blog/exploring-the-power-of-php-use-](http://www.imensosoftware.com/blog/exploring-the-power-of-php-use-) cases-and-best-practices/ (дата звернення: 25.10.2024).

* + 1. Docker: Accelerated Container Application Development. Docker. URL: https://[www.docker.com/](http://www.docker.com/) (дата звернення: 05.11.2024).
    2. 24 Top Frontend Technologies to Use in 2025. Software Development Company

| Netguru. URL: https:/[/www.netguru.com/blog/front-end-technologies](http://www.netguru.com/blog/front-end-technologies) (дата звернення: 18.11.2024).

* + 1. Explore PHP Web Development Basics & Best Practices. Full Scale. URL: https://fullscale.io/blog/php-web-development-trends/ (дата звернення: 20.11.2024).
    2. Top Server-side Scripting Languages & Frameworks - Orient Software. Top Software Outsourcing Company in Vietnam - Orient Software. URL: https://[www.orientsoftware.com/blog/server-side-scripting-languages/](http://www.orientsoftware.com/blog/server-side-scripting-languages/) (дата звернення: 22.11.2024).
    3. Top 10 Server Side Technologies For Creating or Next App. Back4App Blog. URL: https://blog.back4app.com/server-side-technologies/ (дата звернення: 20.11.2024).
    4. Low-code backend to build modern apps. Back4App Blog. URL: https://blog.back4app.com/ (дата звернення: 20.11.2024).
    5. WPShout - Your Website, Sorted. WPShout. URL: https://wpshout.com/ (дата звернення: 21.11.2024).
    6. Production-Grade Container Orchestration. Kubernetes. URL: https://kubernetes.io/ (дата звернення: 21.11.2024).
    7. DigiCert, Inc. TLS/SSL Certificate Authority | Leader in Digital Trust | DigiCert. URL: https:[//www.digicert.com/](http://www.digicert.com/) (дата звернення: 28.10.2024).
    8. OAuth Community Site. OAuth Community Site. URL: https://oauth.net/ (дата звернення: 24.11.2024).
    9. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 10.11.2024).