МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
«ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»

Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АДМІНІСТРУВАННЯ ГОТЕЛЮ

студента Оліфіренка Кирила Андрійовича _____

Науковий керівник:
к.е.н., професор

_____Сендзюк М.А.

Кваліфікаційний бакалаврський проєкт допущений до захисту в Екзаменаційній комісії з атестації здобувачів вищої освіти
завідувача кафедри _____Тішков Б.О.

Київ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

галузь знань 12 «Інформаційні технології»

спеціальність 122 « Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Тішков Б.О..

«____» _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Студенту Оліфіренку Кирилу Андрійовичу

денна форми навчання

на підготовку кваліфікаційного бакалаврського проєкту

на тему: «*Проєктування інформаційної системи для адміністрування готелю*»

Тему затверджено наказом ректора Університету від **«11.03.2024 р.» №529**

Кваліфікаційний бакалаврський проєкт виконується на матеріалах готелю «Стара Правда»

План кваліфікаційного бакалаврського проєкту

Розділ 1 **виконати до10.02.2024**	**Характеристика та аналіз предметної галузі** (назва розділу)
Розділ 2 **виконати до 20.03.2024**	**Розробка вимог і моделювання інформаційної системи** (назва розділу)
Розділ 3 **виконати до 30.03.2024**	**Проєктування та реалізація компонентів системи** (назва розділу)

Об'єкт дослідження:	це процес або явище, що обрані для дослідження, зокрема методика і технологія готелем, визначивши строго функції адміністрування
Предмет дослідження:	частина об’єкту, що розглядаються в рамках даного проєкту, зокрема інформаційні процеси, комп’ютерні технології з адміністрування готелю
Мета випускного бакалаврського проєкту:	провести дослідження і виконати проєктні рішення з розроблення інформаційної системи адміністрування готелю

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі 1 необхідно дати характеристику галузі – управління готелем, визначити структуру об’єкта управління та функції її складових, особливу увагу слід приділити факторам, що впливають на організацію інформаційної системи, доцільно навести схему організаційної структури. Аналіз літературних джерел проводиться з метою виявлення прогресивних методик автоматизації управління готелем.

У розділі 2 провести аналіз і специфікацію вимог до інформаційної системи готелю. Дотримуючись вимог стандартів виконати постановку та алгоритм розв’язання задач з автоматизації функцій адміністрування готелю. Розробити інформаційну модель підтримки управління готелем у відповідності до вимог, викладених методичних вказівках кафедри.

У розділі 3 виконати проєктування таких компонентів системи: інформаційного, технічного та програмного забезпечення у відповідності з методичними рекомендаціями кафедри, вказати можливості і доцільності впровадження. У додатках обов’язково навести вихідні документи та запити.

Завдання підготував науковий керівник _____ **М.А.Сендзюк**
(підпис) (ініціали, прізвище)
« 28 » січня 2024 р.

Завдання одержав студент _____ **К.А. Оліфіренко**
(підпис) (ініціали, прізвище)
«28 » січня 2024 р.

АНОТАЦІЯ

кваліфікаційного бакалаврського проєкту студента 4 курсу
Навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
Оліфіренка Кирила Андрійовича, виконаного на тему:
«Проєктування інформаційної системи для адміністрування готелю»
Київ: кафедра інформаційних систем в економіці, 2024 р.

Кваліфікаційний бакалаврський проєкт присвячений розробці інформаційної системи для ефективного адміністрування готелю. Метою проєкту є створення комплексної системи, котра не лише спрощує процеси управління готельними ресурсами, але й підвищує якість обслуговування клієнтів та забезпечує гнучкість у прийнятті стратегічних рішень. У процесі розробки проєкту проведено аналіз існуючих інформаційних систем, їхніх сильних та слабких сторін, та визначено основні вектори для вдосконалення.

На основі зібраних даних та сучасних технологічних рішень сформовано концепцію інформаційної системи, яка охоплює всі аспекти діяльності готелю: від бронювання номерів до звітності та аналітики. Система включає модулі для роботи з клієнтськими даними, управління номерним фондом, фінансовим обліком, персоналом, а також інструменти для маркетингу та збору зворотного зв'язку від клієнтів. Значна увага приділяється інтуїтивності інтерфейсів та безпеці даних.

У першому розділі проведено аналіз галузі управління готелем з метою визначення структури об'єкта управління та його функцій. Особлива увага приділена факторам, які впливають на організацію інформаційної системи. В роботі наведено схему організаційної структури готелю та виявлено прогресивні методики автоматизації управління готелем з використанням літературних джерел.

У другому розділі проведено аналіз та специфікацію вимог до інформаційної системи готелю згідно з вимогами стандартів.

Розроблено постановку та алгоритм розв'язання задач з автоматизації функцій адміністрування готелю. Також розроблена інформаційна модель підтримки управління готелем у відповідності до методичних вказівок кафедри.

У третьому розділі виконано проєктування таких компонентів системи як інформаційний, технічний та програмний забезпечення згідно з методичними рекомендаціями кафедри. Проаналізовано можливості та доцільність впровадження запропонованих рішень.

У додатках представлені фрагменти коду для бекенду системи, включаючи контролери, сторінки прототипу системи, розроблені у Figma, тестова документація, підготовлена за допомогою Enterprise Architect та результат перетворення ER-моделі в модель даних СУБД MySQL.

Висновки містять рекомендації щодо доцільності проєктування та впровадження інформаційної системи для адміністрування готелю.

Також, визначено ключові структурні одиниці інформації, розроблено відповідну базу даних та інтерфейси користувача. Кваліфікаційний бакалаврський проєкт включає рекомендації щодо подальшої розробки та впровадження системи в реальних умовах готельного бізнесу.

РЕФЕРАТ

Кваліфікаційний бакалаврський проєкт 117 сторінки, 6 таблиць, 74 рисунків, перелік джерел посилань з 52 найменувань, 4 додатки.

Проєктування інформаційної системи для адміністрування готелю.

Перелік ключових слів: *інформаційна система, адміністрування готелю, управління ресурсами, бронювання номерів, зручний інтерфейс, обробка даних, база даних, MySQL, Java, Spring, Thymleaf, Angular, Bootstrap, JavaScript, прототипування, проєктування, Figma, Об'єктно-орієнтоване програмування, діаграма Use case, діаграма послідовності, діаграма класів, стани, компоненти системи UML, SySML, Enterprise Architect, тест-кейси, функціональні вимоги, нефункціональні вимоги, бізнес-цілі…*

Об’єктом розроблення є створення прототипу інформаційної системи для адміністрування готелю, що спеціалізований на обліку та управлінні готелю.

Мета кваліфікаційного бакалаврського проєкту навчитися проєктувати та впроваджувати інформаційні системи, а саме «адміністрування готелю».

Методами розроблення кваліфікаційного бакалаврського проєкту є створення інформаційної системи в середовищі розробки IntelliJ IDEA і використані його відносно зручного функціоналу в виді додаткових функцій а саме: БД в MySQL.

Апаратура використана при розробленні кваліфікаційного бакалаврського проєкту є персональним комп’ютером:

- Процесор Intel(R) Core(TM) i5-7400 CPU @ 3,00GHz;
- Оперативна пам’ять – 32 Гб;
- Накопичувач – SSD, 1 Тб;
- Екран – IPS, 24”.

Одержані результати можуть бути використані для практичного та теоретичного застосування, наприклад, у практичній діяльності готельного бізнесу, розроблена система надає готелям ефективний інструмент для керування бронюваннями, управління користувачами, обліку фінансів та

поліпшення якості обслуговування клієнтів. Нововведення та підходи, розроблені в проєкті, можуть слугувати базою для подальшої розробки подібних систем, а також надихати на створення нових інноваційних рішень.

Рік виконання кваліфікаційного бакалаврського проєкту – 2024. Рік захисту кваліфікаційного бакалаврського проєкту – 2024.

В і д г у к

про кваліфікаційний бакалаврський проєкт
здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»
спеціальності «Комп'ютерні науки»

Оліфіренка Кирила Андрійовича

на тему

«Проєктування інформаційної системи для адміністрування готелю»

1. Актуальність теми: Соціальна сфера займає все більш вагоме значення в розвитку демократичного суспільства. При цьому готельне господарство є одним з центральних ланок, що характеризують рівень суспільних відносин. Тому автоматизація функцій адміністрування готелю є на сьогодні актуальною проблемою.
2. Позитивні риси кваліфікаційного бакалаврського проєкту: на основі глибокого аналізу літературних джерел і практичного досвіду внесені пропозиції і розроблені проєктні рішення з удосконалення існуючих технологій автоматизації функції адміністрування готелю.
3. Наявність самостійних розробок: автор самостійно розробив ряд програмних модулів, які можуть бути використаними на практиці.
4. Цінність теоретичних висновків та практичних рекомендацій: дослідив зарубіжний та вітчизняний досвід комп'ютеризації адміністрування готелів і вніс пропозиції про запровадження передових технологій у вітчизняну практику.
5. Наявність недоліків: можна відмітити наявність окремих стилістичних погрішностей.
6. Загальна оцінка кваліфікаційного бакалаврського проєкту та його допущення до захисту перед ЕК: Допустити до захисту кваліфікований бакалаврський проєкт Оліфіренка К.А. і рекомендую його на конкурс як кращу роботу, відповідно і оцінка.

Науковий керівник - професор, к.е.н., доцент Сендзюк М.А.

(посада, учене звання, науковий ступінь) *(підпис)* *(прізвище, ініціали)*

«___»_____ 20__ р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
ЗМІ	Засоби масової інформації
ЕОМ	Електронно-обчислювальна машина
КС	Комп’ютерна система
ООП	Об’єктно-орієнтований підхід
ОС	Операційна система
ІС	Інформаційна Система
СКБД	Система керування базами даних
API	Application Programming Interface
BDD	Block definition diagrams
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
J2EE	Java 2 Enterprise Edition
MVC	Model View Controller
RAD	Rapid Application Development
IBD	Internal Block diagram
SQL	Structured Query Language
UML	Unified Modeling Language
SPA	Single-Page Application
SysML	Systems Modeling Language
OPD	Object-Process Diagram

ВСТУП

Актуальність теми кваліфікаційного бакалаврського проєкту «Проєктування інформаційної системи для адміністрування готелю» зумовлена стрімким розвитком технологій оброблення інформації та зростаючими вимогами ринку гостинності. У сучасному світі готельний бізнес зіштовхується з необхідністю автоматизації управління, оптимізації сервісу та підвищення якості обслуговування клієнтів. Це вимагає впровадження комплексної інформаційної системи, яка б забезпечувала цілісний підхід до управління всіма аспектами готельної діяльності — від онлайн-бронювання до аналітики та взаємодії з клієнтами.

Актуалізація теми також обумовлена зростанням конкуренції в готельній індустрії, яка вимагає від готелів бути гнучкими, швидко адаптуючись до змін ринкових умов та вподобань клієнтів. Інформаційні системи дозволяють збирати великі обсяги даних, аналізувати їх для прийняття обґрунтованих рішень та прогнозування майбутніх тенденцій, що є вирішальним для збереження та збільшення клієнтської бази.

Мета кваліфікаційного бакалаврського проєкту, – спроєктувати інформаційну систему, а саме «адміністрування готелю».

Пошуки шляхів досягнення цієї мети обумовили необхідність визначення наступних завдань:

– дослідити предметну область для якої створюється інформаційна система;

– визначити вимоги до інформаційної системи (бізнес-вимоги, функціональні та не функціональні);

– обґрунтувати вибір програмних рішень для розроблення інформаційної системи;

– розробити інформаційну модель системи та описати вхідні і вихідні дані;

- розробити функціональну модель системи;
- розробити структурний алгоритм роботи інформаційної системи;
- спроєктувати архітектуру інформаційної системи;
- спроєктувати базу даних для інформаційної системи;
- спроєктувати користувацький інтерфейс для інформаційної системи;
- скласти тест-кейси для перевірки вимог до інформаційної системи;
- представити прототип інформаційної системи;
- представити спроєктовану систему в IntelliJ IDEA.

Практичне значення результатів, одержаних від проєктування інформаційної системи для адміністрування готелю, полягає в значному підвищенні ефективності управлінських процесів, оптимізації роботи персоналу та покращенні якості обслуговування клієнтів. Система забезпечує автоматизацію ключових операцій, вдосконалення процесу бронювання та реєстрації, а також підтримку прийняття рішень через детальну аналітику та звітність. Це сприяє не лише підвищенню доходів і заповненості готелю, але й гарантує мінімізацію помилок, підвищення безпеки даних та підтримку постійного розвитку та адаптації бізнесу до змінних ринкових умов.

Використані інструментальні засоби: середовище проєктування IntelliJ IDEA Ultimate 2023.2.6, система проєктування баз даних MySQL Workbench 8.0 CE, HTML5, CSS, JavaScript, Angular, BootStrap, WebPack, Draw.IO, Lucidchart, UMLetino 14.3, Enterprise Architect, OPCAT, операційна система Windows 10, текстовий редактор Microsoft Word 2019 для підготовки та оформлення пояснювальної записки до кваліфікаційного бакалаврського проєкту.

Структура роботи зумовлена метою і завданнями та складається зі вступу, чотирьох розділів, висновку, переліку джерел посилання та додатків.

РОЗДІЛ 1

ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

Предметна галузь дослідження охоплює широке коло питань, пов’язаних з функціонуванням сучасних готельних підприємств. Вона включає в себе процеси управління номерним фондом, бронювання, взаємодії з клієнтами, організації роботи користувачів, ведення фінансової діяльності, забезпечення безпеки та комфорту гостей, а також маркетингу та формування позитивного іміджу готелю.

Предметом дослідження є інформаційні процеси та комп'ютерні технології, що використовуються для оптимізації та адміністрування діяльності готелю.

Об'єкт дослідження – це процес або явище, що обраний для дослідження, зокрема методика та технологія адміністрування готелю з використанням спеціалізованої інформаційної системи, яка інтегрує різні аспекти діяльності готелю та забезпечує ефективне управління ними. Увага акцентована на конкретні процеси, а саме онлайн-бронювання, управління номерним фондом та управління користувачами та персоналом, як ключовим елементам сучасного готельного бізнесу.

Сучасний готельний бізнес неможливо уявити без використання інформаційних технологій.[1] Активний розвиток українського ринку IT відкриває нові можливості для підприємництва, особливо в сфері гостинності. Однією з ключових інновацій є онлайн-системи бронювання. Вони спрощують процес резервування готельних номерів, оплати, комунікації з клієнтами, управління номерним фондом та іншими аспектами діяльності готелю. Онлайн-бронювання сприяє розвитку цифрової економіки та підвищенню конкурентоспроможності готельної галузі.[5]

Процес онлайн-бронювання складається з декількох етапів: пошуку готелю клієнтом через спеціалізовані веб-сайти, мобільні додатки або сайт самого готелю, вибору параметрів проживання, таких як дати заїзду та виїзду, кількість гостей, тип номера, додаткові послуги, введення контактної та платіжної інформації, підтвердження бронювання системою та відправлення підтвердження клієнту електронною поштою або через SMS.[3]

Інформаційна система для адміністрування готелю інтегрується з системою онлайн-бронювання та забезпечує автоматизацію процесів. Після завершення онлайн-бронювання інформація миттєво передається в систему готелю, що дозволяє оперативно отримувати актуальні дані про заброньовані номери та ефективно управляти номерним фондом, персоналом, фінансами та іншими аспектами діяльності.[4]

В рамках нашої інформаційної системи для адміністрування готелю процес платежу є важливою частиною онлайн-бронювання. Після вибору номера та введення необхідних даних, система переходить до етапу оплати. На цьому етапі кошти спочатку резервуються на банківській картці користувача, а потім, після перевірки і підтвердження всіх введених даних, здійснюється фактичне списання коштів.

Після успішного завершення процесу оплати, клієнт отримує квитанцію, що підтверджує бронювання, наприклад, електронний ваучер на заселення або електронний документ про бронювання. Цей документ гарантує клієнту надання обраної послуги та забезпечує всі необхідні деталі для заселення.

Системи онлайн-бронювання в нашій моделі характеризуються миттєвим інформуванням готелю про резервацію номера. Це означає, що відразу після того, як клієнт завершує процес бронювання, інформація надходить до готелю, забезпечуючи актуальність даних про наявність вільних номерів. Такий підхід гарантує високу точність управління номерним фондом і дозволяє клієнтам бронювати номери за мінімально можливий період часу до початку бронювання.

Детальніше розглянемо переваги та недоліки адміністрування готелю:

– миттєва оплата замовлення на сайті;

– клієнт одразу отримує гарантію заїзду за цінами готелю. Часто інформаційні системи онлайн-бронювання надають знижку на проживання в номерах за рахунок своєї комісії, щоб залучити більше клієнтів;

– клієнт сам вибирає період проживання, категорію номера, набір додаткових готельних послуг;

– готель немає необхідності зв'язуватися з клієнтом, так як бронювання проходить в автоматичному режимі без участі адміністратора;

– готель сам визначає розмір квоти для бронювання онлайн, ціни, набір додаткових послуг;

– адміністратор має данні користувачів в онлайн доступі;

– адміністратор готелю може бачити час прибуття до їх готелю;

– система працює в автономному режимі цілодобово 24 години 7 днів на тиждень.

Проте існують недоліки:

– ризики кібербезпеки, що вимагають надійного захисту інформаційної системи від кібератак;

– необхідність значних інвестицій у впровадження та підтримку інформаційної системи;

– необхідність навчання персоналу готелю роботі з інформаційною системою.

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

Вибір програмних систем для адміністрування готелю може бути визначеним конкретними потребами та масштабом готельного бізнесу. В області бронювання номерів готелів розглянемо чотири популярні ІС: booking.com, agoda.com, hotelogix, opera by oracle hospitality.

1.2.1 Дослідження існуючих інформаційних систем для обраної предметної галузі

Загальна характеристика системи Booking.com:

Компанія Booking.com, заснована в 1996 році в Амстердамі, пройшла шлях від маленького голландського стартапу до одного зі світових цифрових лідерів у сфері подорожей та бронювання номерів готелів.[7]

Інвестуючи в технології, які допомагають подорожувати без турбот, Booking.com пропонує мільйонам гостей чудові варіанти дозвілля, транспортні послуги та неймовірне житло, починаючи від будинків та закінчуючи готелями та не тільки. Як найбільша у світі туристична платформа як для відомих брендів, так і для підприємців різного рівня, Booking.com допомагає власникам об'єктів розміщення по всьому світу залучати гостей та розширювати бізнес.[7]

На ній доступно для бронювання понад 28 мільйонів заявлених одиниць розміщення, серед яких понад 6,2 мільйона у будинках, апартаментах та інших унікальних об'єктах розміщення.[7]

Booking.com в основному відомий як онлайн-платформа для бронювання готелів та інших помешкань.[8] На відміну від традиційних PMS, Booking.com надає готелям доступ до великої аудиторії клієнтів, проте використання його як єдиної системи для управління готелем може вимагати інтеграції з іншими інструментами для повноцінного управління всіма аспектами готельного бізнесу. Головна сторінка сайту бронювання номерів зображена на рисунку 1.1.

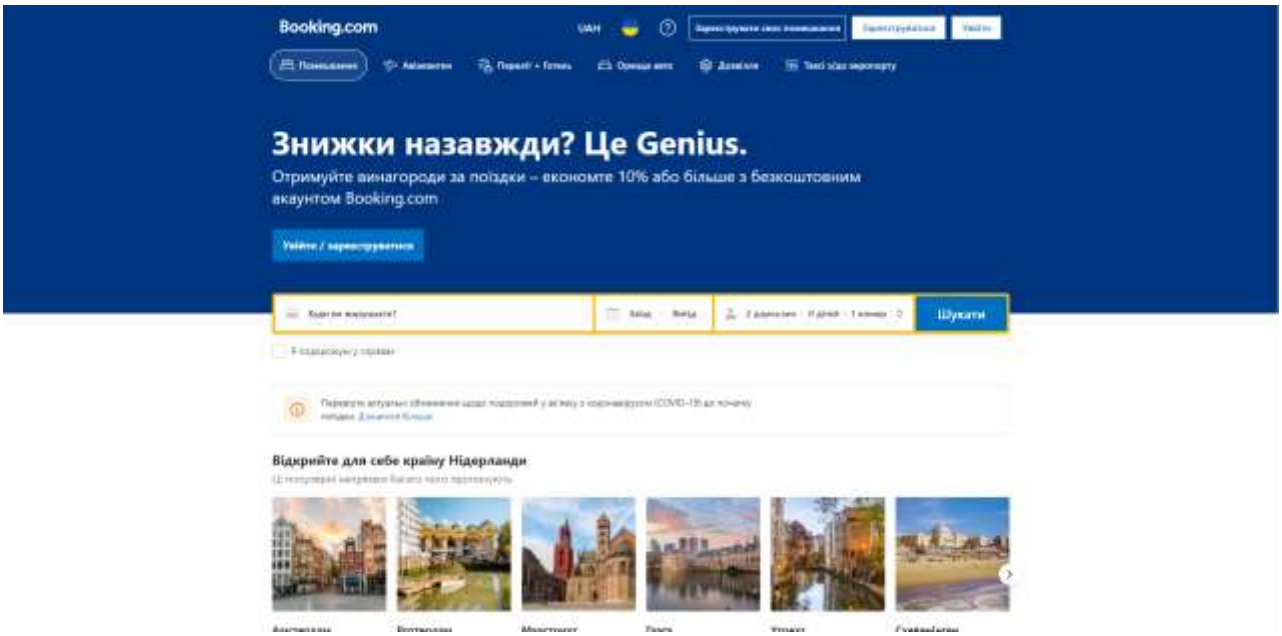


Рисунок 1.1 – Головна сторінка сайту бронювання номерів «Booking.com»

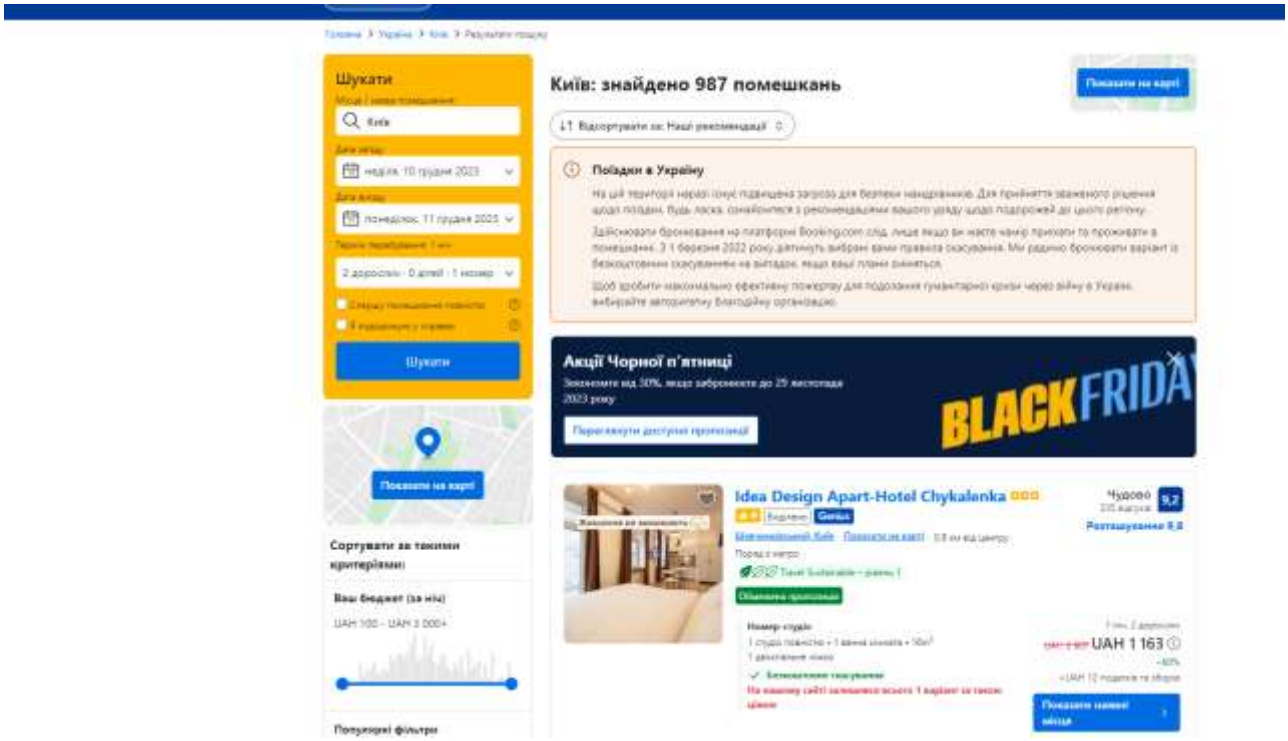


Рисунок 1.2 – Сторінка бронювання «Booking.com»

Загальна характеристика системи agoda.com:

Agoda.com, також як і Booking.com, визначається як онлайн-платформа для бронювання готелів та інших помешкань. В основному використовується як канал для привертання гостей, а не як інтегрована система для управління готелем.[9]

Agoda – це один з найкращих сайтів для бронювання готелів. Його спеціалізація – Південно-Східна Азія, проте останнім часом активно розвиваються інші напрями. Загалом у базі Agoda понад 2 мільйони пропозицій щодо житла для туристів.[9]

Принцип схожий на те, як забронювати готель на Booking, але Agoda несе відповідальність за надані послуги. Цей сервіс приймає оплату і сам контактує з адміністрацією готелю. Букінг передає дані кредитної картки, а гроші отримує безпосередньо готель. Якщо з готелем виникнуть проблеми, Agoda безкоштовно надасть поблизу заміну аналогічного рівня.

Плюси сайту Agoda.com:

– дуже багато пропозицій щодо Азії, у тому числі й ексклюзивних;

– дані карти залишаються на сайті, їх не передають третім особам (готелям);

– сервіс готовий допомогти, якщо на місці виникнуть проблеми із забронюваним готелем.

На рисунку 1.3 відображена головна сторінка сайту бронювання номерів готелів «agoda.com».

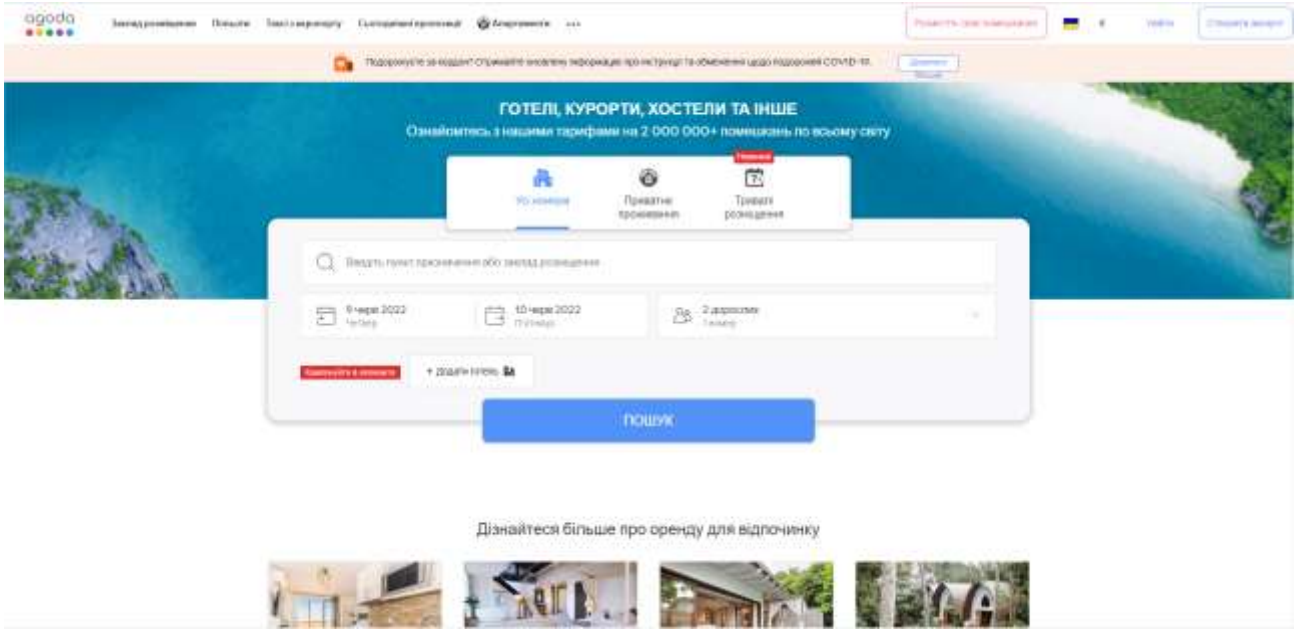


Рисунок 1.3 – Головна сторінка сайту бронювання номерів «agoda.com»

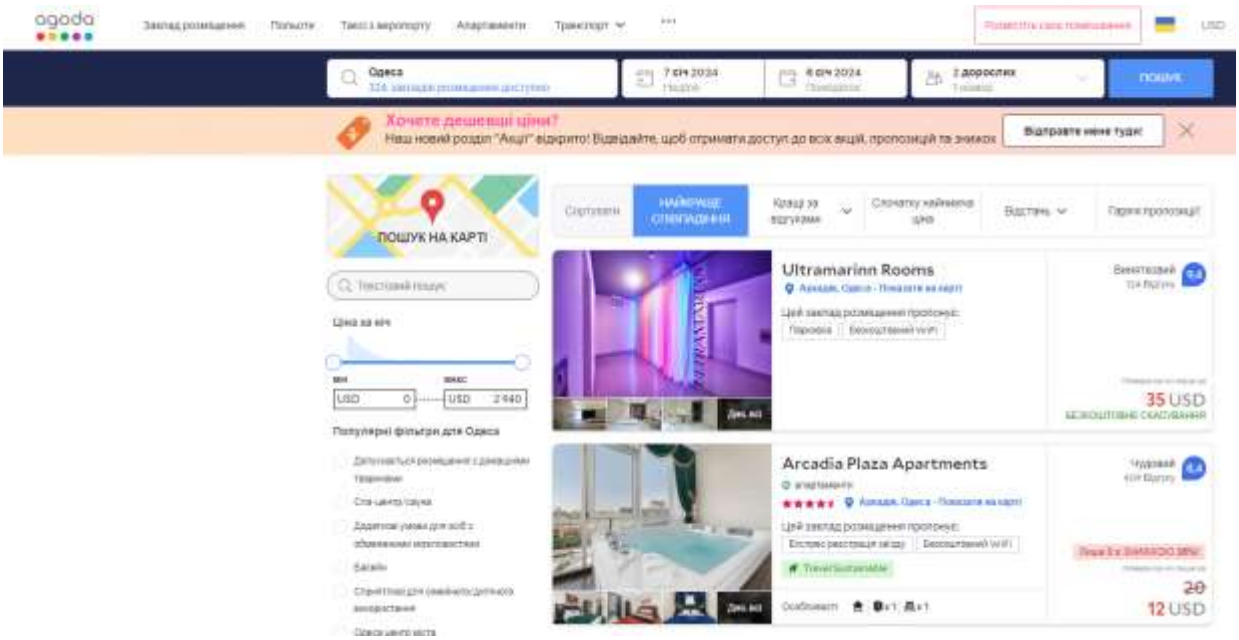


Рисунок 1.4 – Сторінка бронювання «agoda.com»

Загальна характеристика системи Hotelogix:

Hotelogix – це хмарна система управління готелем, що надає широкий спектр функцій, включаючи бронювання, фронт-офіс, звітності та інші. Вона дозволяє готелям ефективно керувати різними аспектами готельного бізнесу, зберігаючи дані в хмарі для легкого доступу та управління.[10]

Бронювання та резервації в Hotelogix дозволяє гостям легко бронювати номери онлайн через різні канали. Інтуїтивний інтерфейс сприяє швидкому і зручному процесу бронювання, а власна система управління інвентарем допомагає уникнути перепланувань та перевантажень.

Фронт-офіс та облік гостей, в Hotelogix система забезпечує повний контроль над процесом реєстрації та обліку гостей. Інтерфейс для фронт-офісу допомагає персоналу ефективно виконувати завдання, забезпечуючи швидку реєстрацію та виписку гостей.

Онлайн-звітність та аналітика, Hotelogix надає детальні звіти та аналітику для керівництва готелю. Можливість моніторингу ключових показників, таких як витрати, прибутки, використання номерів, дозволяє приймати обґрунтовані управлінські рішення.[10]

Інтеграція та мобільний доступ, Hotelogix підтримує інтеграцію з іншими системами готелю та сервісами, такими як POS (Point of Sale). Крім того, інтерфейс оптимізований для мобільних пристроїв, що дозволяє персоналу готелю працювати з системою в будь-якому місці та часі.[10]

Гнучкі інструменти для управління резерваціями та тарифами дозволяють готелям ефективно адаптуватися до змін в попиті та ринкових умовах. Автоматичне оновлення та керування тарифами роблять процес максимально ефективним.

Hotelogix дозволяє готелям будь-якого розміру автоматизувати та оптимізувати свою діяльність, забезпечуючи високий рівень обслуговування для гостей та спрощуючи роботу персоналу. На рисунку 1.5 відображена головна сторінка системи управління готелем «Hotelogix».

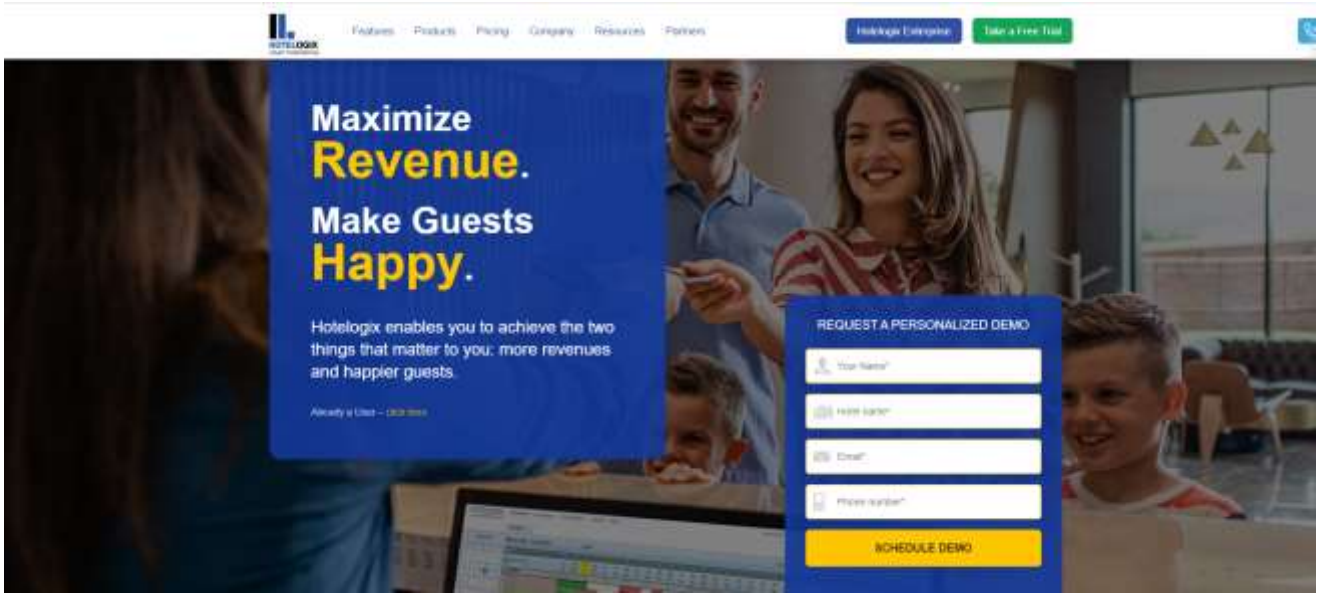


Рисунок 1.5 – Головна сторінка системи управління готелем «Hotelogix»

Загальна характеристика системи Opera by Oracle Hospitality:

Opera – інтегрована програмна система для готелів, розроблена Oracle Hospitality. Вона надає модулі для бронювання, обліку гостей, ресторанного бізнесу та інших функцій. Опера забезпечує глибоку інтеграцію та високий рівень налаштувань, що дозволяє адаптувати систему під конкретні потреби готелю.[11]

Opera включає різні модулі, такі як Property Management System (PMS), Sales and Catering, Customer Relationship Management (CRM), та інші. Кожен модуль спеціалізується на конкретній галузі готельного бізнесу, дозволяючи готелям максимально використовувати різні аспекти своєї діяльності.[11]

Opera відома своєю здатністю інтегруватися з іншими системами та сервісами. Це дозволяє готелям максимально використовувати інструменти, які вони вже мають, і легко впроваджувати нові технології.

Opera надає великий рівень гнучкості та налаштувань. Готелям надається можливість адаптувати систему під свої конкретні потреби та процеси.[11] Це важливо для того, щоб кожен готель міг використовувати Opera відповідно до свого бізнес-моделі та стратегії.

Однією з ключових особливостей є розширені засоби аналітики та звітності. Opera надає готелям доступ до різноманітних звітів, які допомагають аналізувати ефективність, витрати, прибуток та інші ключові показники.

Opera by Oracle Hospitality визначається своєю повнотою та інтегрованістю, роблячи її однією з популярних вибірок для готелів різних розмірів та класів. На рисунку 1.6 відображена головна сторінка системи управління готелем «Opera by Oracle Hospitality».

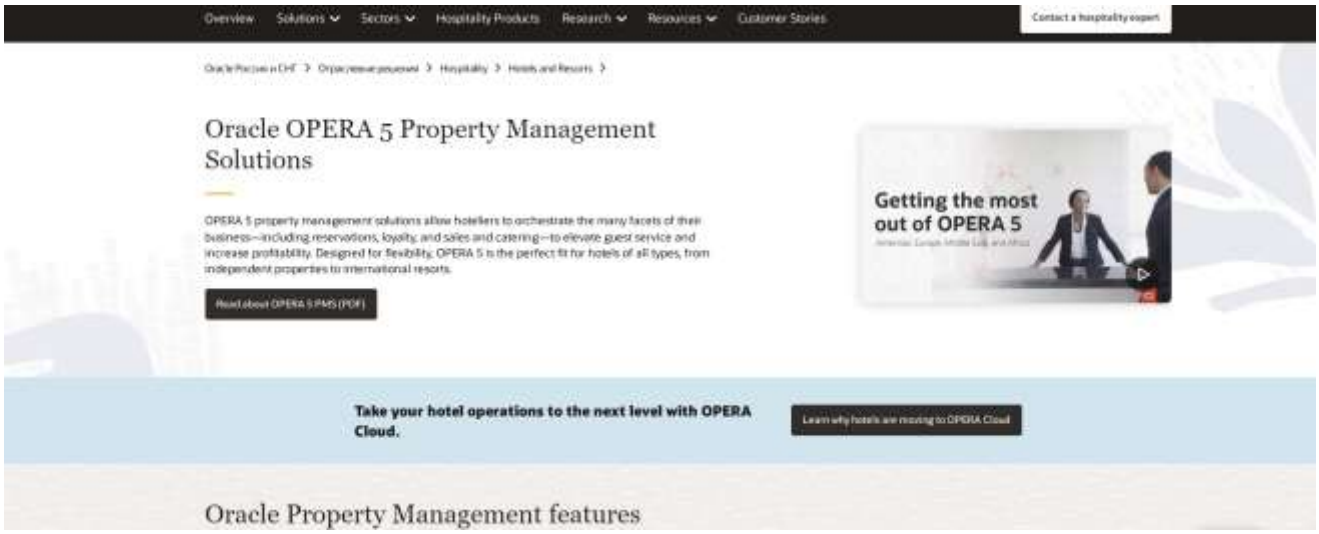


Рисунок 1.6 – Головна сторінка системи управління готелем «Opera by Oracle Hospitality»

Кожна з цих систем має свої переваги та обмеження і вибір між ними повинен бути здійснений з урахуванням конкретних потреб та масштабу готельного бізнесу.

1.2.2 Порівняння інформаційних систем

Результати порівняння програмних систем booking.com, agoda.com, hotelogix та opera by oracle hospitality, наведені в табл. 1.1:

Порівняння програмних систем Таблиця 1.1

№	Характеристика	Hotelogix	Opera by Oracle Hospitality	Booking.com	agoda.com
Зручність використання систем/сайтів					
1	Інтерфейс користувача	Інтуїтивний	Зручний та легко налаштовується	Спрощений та легкий у використанні	Простий та зрозумілий
2	Мобільний доступ	Підтримує мобільний доступ та оптимізований для пристроїв	Забезпечує мобільний доступ, інтерфейс оптимізований для мобільних пристроїв	Мобільний додаток для гостей, зручний інтерфейс	Мобільний додаток для гостей, простий та зрозумілий інтерфейс
3	Навігація системою	Легка та логічна	Зручна та ефективна	Інтуїтивна та швидка	Проста та доступна
4	Керування резерваціями	Просте та швидке бронювання	Забезпечує ефективне управління резерваціями	Легке бронювання та управління бронюванням	Швидке та просте бронювання
5	Візуалізація інформації	Графічне відображення ключових	Візуалізація даних для	Зручна візуалізація результатів пошуку	Графічне відображення інформації

		показникі в	швидкого аналізу		
6	Підтримка мов та локалізація	Підтримка різних мов та локалізаці я	Підтримк а різних мов та локалізаці я	Мультиязи чність та локалізова ні версії	Мовна підтримка для різних регіонів
7	Корисні інструменти та функції	Наявність додаткови х інструмен тів для оптимізаці ї роботи готелю	Розширен і функції для кращого управлінн я готелем	Наявність фільтрів та інструмент ів для гостей	Додаткові функції для комфорту користувачі в
Подання інформації					
8	Деталі бронювання	Так	Так	Так	Так
9	Інформація про готель та послуги	Так	Так	Так	Так
10	Фотографії номерів та об'єкту	Так	Так	Так	Так
11	Рейтинги та відгуки гостей	Так	Так	Так	Так
12	Мапи та розташування об'єкту	Так	Так	Так	Так
13	Інформація про відмінність об'єкту	Так	Ні	Так	Так
14	Додаткові послуги та пропозиції	Так	Так	Так	Так
15	Правила та умови проживання	Так	Ні	Так	Так
16	Наявність онлайн-чату/підтримки	Так	Так	Так	Так
17	Детальність сторінок готелів	Так	Так	Так	Так
Спілкування					
18	Електронна пошта з гостем	Так	Так	Так	Так
19	Онлайн-чат на сайті	Так	Так	Так	Так
20	Сповіщення та підтримка через додаток	Так	Так	Так	Так
21	Гостьові панелі та особистий кабінет	Так	Так	Так	Так
22	Спілкування через соціальні мережі	Так	Ні	Так	Так
23	Система зворотного зв'язку	Так	Так	Так	Так
24	Сповіщення про акції та пропозиції	Так	Так	Так	Так
Функціональні можливості					
25	Управління бронюваннями та резервaціями	Керування бронюван ням, зміна та скасуванн я	Ефективн е управлінн я резерваці ями та номерами	Легке бронюванн я та управління бронюванн ям	Швидке та просте бронювання
26	Інтеграція з іншими системами	Можливіс ть	Широкий спектр	Інтеграція з багатьма	Підтримка інтеграції з

		інтеграції з різними готельними системами та сервісами	інтеграцій для покращення функціоналу	партнерами та сервісами	різними системами
27	Фінансовий контроль та облік доходів	Система обліку витрат та прибутку, фінансовий контроль	Контроль над фінансами готелю, звіти та аналітика	Засоби обліку доходів та фінансовий звіт	Облік доходів та контроль фінансів
28	Маркетинг та просування	Рекламні та маркетингові інструменти, знижки та акції	Системи маркетингу та рекламні кампанії	Знижки, рекламні пропозиції та маркетингові інструменти	Програми лояльності та рекламні акції
29	Аналіз даних та звітність	Статистичний аналіз та звітність, аналітика використання послуг	Засоби для аналізу та звітності по всіх аспектах	Статистика та аналіз даних для прийняття рішень	Система звітності та аналіз результативності

Висновки за результатами аналізу програмних систем:

Аналіз показав, що кожна програмна система має свої переваги та недоліки. Вибір між ними повинен залежати від конкретних потреб готелю, його розміру, бюджету та вимог до функціоналу. При цьому, необхідно враховувати якість технічної підтримки та можливість інтеграції з іншими системами, оскільки це ключові аспекти успішного впровадження.

1.2.3 Позитивні характеристики існуючих систем, що їх слід відтворити у проєктних рішеннях

Позитивні характеристики системи Hotelogix:

– ефективне управління бронюваннями та резерваціями;

– широкі можливості інтеграції з іншими готельними системами;

– функціональність для фінансового контролю та обліку доходів.

Позитивні характеристики системи Opera by Oracle Hospitality:

– ефективне управління резерваціями та номерами;

– широкий спектр інтеграцій для розширення функціоналу;

– контроль над фінансами готелю та засоби обліку доходів.

Позитивні характеристики системи Booking.com:

– легке бронювання та управління бронюванням;

– широка мережа партнерів та інтеграція з різними сервісами.

Позитивні характеристики системи Agoda.com:

– швидке та просте бронювання;

– підтримка інтеграції з різними системами.

Позитивні характеристики існуючих систем, що їх слід відтворити у проєктних рішеннях Таблиця 1.2

№	Характеристика	Hotelogix	Opera by Oracle Hospitality	Booking.com	agoda.com
Позитивні характеристики					
1	Ефективне управління бронюваннями	Забезпечує швидке та ефективне управління бронюваннями	Забезпечує ефективне управління резерваціями та номерами	Легке бронювання та управління бронюванням	Швидке та просте бронювання
2	Інтеграція з іншими системами	Можливість інтеграції з різними готельними системами та сервісами	Широкий спектр інтеграцій для розширення функціоналу	Інтеграція з багатьма партнерами та сервісами	Підтримка інтеграції з різними системами
3	Фінансовий контроль та облік доходів	Система обліку витрат та прибутку, фінансовий контроль	Контроль над фінансами готелю, звіти та аналітика	Засоби обліку доходів та фінансовий звіт	Облік доходів та контроль фінансів
4	Маркетинг та просування	Рекламні та маркетингові інструменти, знижки та акції	Системи маркетингу та рекламні кампанії	Знижки, рекламні пропозиції та маркетингові інструменти	Програми лояльності та рекламні акції

При аналізі позитивних аспектів існуючих інформаційних систем управління готелями, таких як Hotelogix, Opera by Oracle Hospitality, Booking.com та Agoda.com, виявлено ключові характеристики, що повинні бути втілені у нашій системі. Це включає ефективне управління бронюваннями та резерваціями, широкі можливості інтеграції з іншими системами, функціональність для точного фінансового контролю, а також простоту та швидкість процесу бронювання. Інтегруючи ці елементи, можна значно підвищити якість обслуговування, оптимізувати управлінські процеси та забезпечити ефективну взаємодію з клієнтами.

1.2.4 Недоліки існуючих систем та напрями їх удосконалення

Недоліки системи Hotelogix:

– потребує докладного навчання для використання всіх можливостей.

Недоліки системи Opera by Oracle Hospitality:

– висока вартість імплементації та обслуговування.

Недоліки системи Booking.com:

– обмежений функціонал для внутрішнього управління готелем.

Недоліки системи Agoda.com:

– обмежені можливості для управління готелем внутрішньо.

Недоліки існуючих систем та напрями їх удосконалення Таблиця 1.3

№	Характеристика	Hotelogix	Opera by Oracle Hospitality	Booking.com	agoda.com
Недоліки та напрями удосконалення					
1	Висока вартість імплементації	Вимагає докладного навчання та може бути витратним	Високі витрати на впровадження та обслуговування	Потребує обов'язкового користувацького обліку	Зменшення вартості імплементації та обслуговування
2	Обмежені можливості для управління готелем внутрішньо	Вимагає додаткового навчання для використання всіх можливостей	Обмежені можливості для внутрішнього управління готелем	Обмежений функціонал для внутрішнього управління	Розширення функціоналу для повного управління готелем

3	Висока вартість обслуговування	Може бути витратним для деяких готелів	Висока вартість обслуговування та підтримки	Обов'язковий користувацький облік та можливі витрати	Оптимізація вартості обслуговування та підтримки

При аналізі існуючих систем адміністрування готелями, виявлено декілька ключових аспектів, які вимагають удосконалення в рамках розробки нашої інформаційної системи. Одним з основних напрямків для поліпшення є забезпечення більшої гнучкості та кастомізації системи, щоб вона могла відповідати унікальним потребам кожного готелю. Це означає можливість адаптувати інтерфейси, функціонал та інтеграції згідно з конкретними вимогами та управлінськими стилями.

Іншим важливим аспектом є розширення можливостей інтеграції, особливо з місцевими платіжними системам. Необхідно забезпечити підтримку широкого спектра платіжних опцій, включаючи місцеві та міжнародні платіжні шлюзи, для забезпечення зручності та доступності для користувачів з різних країн.

Крім того, важливим аспектом є підвищення зручності користування (юзабіліті) та мультиязичності системи. Це передбачає розробку інтуїтивно зрозумілого інтерфейсу, доступного для користувачів без технічного фону та підтримку різних мов для залучення міжнародних клієнтів. Також, необхідно забезпечити високий рівень безпеки даних, оскільки інформаційні системи управління готелями зберігають конфіденційну інформацію про клієнтів та їх фінансові транзакції.

1.3 Розробка концепції інформаційної системи

Головним архітектурним підходом, обраним для розробки системи, є трьохрівнева архітектура, яка базується на моделі MVC (Model-View-Controller), що дозволяє розбити систему на три основні рівні: рівень представлення (інтерфейс користувача), рівень бізнес-логіки, та рівень доступу до даних. Цей підхід забезпечує чітке відокремлення логіки від інтерфейсу користувача та

даних, сприяючи більшій організованості, гнучкості та легкості управління та розширення системи.[12]

На рівні представлення фокус зосереджений на розробці інтерфейсу користувача, що включає обробку введення користувача та забезпечення зворотного зв'язку. Це включає розробку зручних та інтуїтивно зрозумілих інтерфейсів, що сприяють покращенню користувацького досвіду.

Рівень бізнес-логіки відповідає за основну логіку та правила застосунку, інтегруючи класи, які представляють різні сутності та компоненти бізнес-логіки системи. Це є серцевиною системи, де зосереджено основні функції та процеси, пов'язані з управлінням готелем.

Рівень доступу до даних забезпечує взаємодію з базою даних або іншими джерелами даних. Він містить класи для доступу та маніпуляції даними, такі як репозиторії або DAO (Data Access Objects), гарантуючи ефективне управління даними та їх цілісність.

У межах архітектури MVC також використовуються класи, що представляють компоненти моделі, вигляду та контролера.[13] Це дозволяє розробити баланс між зручністю користувача, ефективністю обробки даних та гнучкістю управління бізнес-процесами. Така структура системи не тільки відповідає сучасним технічним стандартам, але й забезпечує масштабованість та адаптивність до змінних потреб готельного бізнесу.

Також, у проєкті для розробки інформаційної системи адміністрування готелю було обрано мову програмування Java, яка є однією з найбільш розповсюджених та надійних мов у світі розробки програмного забезпечення. Java відома своєю масштабованістю, безпекою та кросплатформеністю, що робить її ідеальним вибором для створення великих, складних додатків, таких як корпоративні інформаційні системи. [15]

Для розробки використовується середовище розробки IntelliJ IDEA Ultimate, яке забезпечує широкий спектр інструментів для розробників та підтримує роботу з Java та іншими мовами програмування. IntelliJ IDEA

відзначається своєю продуктивністю, зручністю інтерфейсу та розширеними можливостями для рефакторингу та відладки коду, що сприяє підвищенню ефективності процесу розробки.[16]

У цілому, в моєму проєкті використовується об'єктно-орієнтований підхід (ООП). Цей підхід дозволяє структурувати програмне забезпечення шляхом створення об'єктів, що представляють реальні сутності та концепції. Об'єктно-орієнтоване програмування забезпечує високу рівень абстракції та інкапсуляції, що дозволяє ефективно управляти складними системами, спрощуючи розробку, тестування та підтримку.

Завдяки ООП, розробка стає більш модульною, що сприяє легшому повторному використанню коду та покращенню його читабельності. Також цей підхід сприяє більш ефективному управлінню пам'яттю та ресурсами, а також надає можливість легшого масштабування та адаптації програмного забезпечення до змінних вимог і потреб користувачів. Використання ООП у комбінації з мовою програмування Java, фреймворком Spring Boot, та створює міцну основу для розробки надійної, ефективної та гнучкої інформаційної системи для готельного менеджменту.

Фреймворк, обраний для розробки проєкту, – це Spring Framework, зокрема, його модуль Spring Boot. Spring Boot дозволяє значно спростити процес налаштування та розгортання веб-додатків на Java, забезпечуючи автоматизацію багатьох процесів, що традиційно вимагали великої кількості ручної роботи. Особливістю Spring Boot є його "опініоністичність", що означає, що фреймворк пропонує стандартні рішення для багатьох задач, спрощуючи процес конфігурації.[19] Це робить Spring Boot чудовим вибором для розробки в умовах, де важливі швидкість розробки та простота управління проєктом.

Spring складається з деяких модулів, таких як:

- data access;
- web;
- core;

- testing;
- spring MVC;
- spring security;
- та інших.

Цей фреймворк пропонує послідовну модель і робить її придатною до більшості типів додатків, які вже створені на основі платформи Java. Spring реалізує модель розробки, засновану на кращих стандартах індустрії, і робить її доступною в багатьох областях Java.

У моєму проєкті інформаційної системи для адміністрування готелю для розробки бази даних було обрано MySQL Workbench. Цей вибір зумовлений низкою переваг цього інструменту. Перш за все, MySQL Workbench відомий своїми візуальними інструментами для проєктування, що дозволяють легко створювати, візуалізувати та редагувати схеми баз даних.[20] Це допомагає як в плануванні структури даних і в розробці зв'язків та обмежень між різними таблицями, що є критично важливим для ефективного управління даними у великих системах.

Другий аспект, який робить MySQL Workbench відповідним для проєкту, полягає в його здатності інтегруватися з іншими технологіями та інструментами розробки, що використовуються в проєкті, включаючи Java, Spring Boot та IntelliJ IDEA.[21] Це забезпечує плавну взаємодію між розробкою бази даних та іншими компонентами системи, що сприяє більш ефективному та координованому процесу розробки. Така інтеграція є ключовою для створення єдиного, злагодженого середовища розробки, яке підтримує всі аспекти інформаційної системи.

MySQL Workbench надає широкі можливості для адміністрування та обслуговування бази даних. Це включає інструменти для виконання запитів SQL, тестування їх продуктивності та оптимізації, а також можливості для резервного копіювання, моніторингу та підтримки бази даних.[21]

У розробці фронтенду ключову роль відіграють Bootstrap, HTML, CSS та JavaScript. Bootstrap використовується для створення респонсивного та зручного дизайну веб-інтерфейсу, що адаптується до різних розмірів екранів і пристроїв.[24] Це гарантує, що веб-сайт буде зручним і привабливим для користувачів. HTML та CSS забезпечують структурування вмісту та візуальну стилізацію сторінок, тоді як JavaScript додає інтерактивність, дозволяючи реалізувати такі функції, як обробка подій користувача, анімації та динамічне оновлення контенту.

Для розвитку більш складних інтерфейсів в проєкті застосовується Angular, сучасний JavaScript-фреймворк, який допомагає створювати ефективні та масштабовані односторінкові додатки (SPA).[24] Angular надає потужні інструменти для розробки, які покращують організацію коду та спрощують управління різними компонентами додатку. Використання цих технологій дозволяє створити інтуїтивно зрозумілий, швидкий та гнучкий інтерфейс, що підвищує загальну задоволеність користувачів та ефективність взаємодії з системою.

Для демонстрації користувацьких інтерфейсів було створено два прототип (один з яких для клієнтської частини, а інший для адміністраторської частини) інформаційної системи у векторному онлайн-сервісі Figma.

РОЗДІЛ 2

РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної системи

Задача аналізу потреб полягає у зборі, розумінні та аналізі детальних вимог і очікувань від системи з боку всіх основних зацікавлених сторін.[26] Це включає в себе адміністрацію готелю, персонал та клієнтів. Кожна з цих груп має свої унікальні потреби та очікування, які повинні бути враховані під час розробки системи.

Основні задачі аналізу інформаційної системи:

1. Адміністрація готелю:

– потреби в ефективному управлінні ресурсами готелю (номерний фонд, персонал, фінанси);

– максимізація прибутковості та оптимізації використання ресурсів;

– вимоги до системи щодо забезпечення аналітичних звітів для прийняття обґрунтованих управлінських рішень.

2. Персонал готелю:

– потреба в зручному та ефективному інструменті для виконання щоденних завдань (управління бронюваннями та обслуговування клієнтів);

– необхідність у швидкому доступі до актуальної інформації про клієнтів, бронювання та доступність номерів;

– потреба мати систему, яка спрощує комунікацію між різними відділами готелю.

3. Клієнти готелю:

– очікування щодо простоти та зручності бронювання номерів онлайн;

– потреба у персоналізованих пропозиціях та високому рівні клієнтського сервісу;

– бажання мати доступ до історії своїх бронювань та можливості управління ними онлайн.

Далі, потрібно скористатися методами для збору інформації, а саме проаналізувати ринок та дослідити сучасні тенденцій та найкращі практики у сфері готельного менеджменту.

На основі цього визначаються наступні види вимог:

– функціональні, які визначають, що система повинна робити;

– нефункціональні, які визначають вимоги до якості та характеристик системи;

– бізнес-вимоги, які враховують стратегічні цілі та потреби бізнесу.[28]

Мета цього аналізу є забезпечення глибокого розуміння вимог і очікувань усіх зацікавлених сторін, щоб розроблена система була максимально ефективною та відповідала потребам сучасного готельного бізнесу. Це допоможе розробити систему, яка не лише вирішує поточні завдання, але й має потенціал для адаптації до майбутніх викликів та розвитку готельного ринку.

2.1.1 Функціональні вимоги до системи

Функціональні вимоги визначають основні можливості та функції, які наша ІС для адміністрування готелю повинна надавати для задоволення потреб користувачів та досягнення бізнес-цілей проєкту. Функціональні вимоги забезпечують, що система матиме необхідний функціонал для управління обліковими записами користувачів, бронювання та управління номерами, обліку фінансових операцій та послуг, а також для генерації звітів та аналітичної звітності.

Функціональні вимоги передбачають наявність важливих функцій безпеки, таких як контроль доступу до даних, захист від несанкціонованого доступу та забезпечення конфіденційності інформації про клієнтів готелю. Також, вони

включають можливість автоматизації процесів, таких як генерація рахунків та оповіщення про бронювання.

Характеристика головних функціональних вимог системи:

Аутентифікація в системі, а саме безпечний доступ та авторизація:

1. Система повинна забезпечувати безпечний доступ до своїх функцій шляхом використання надійних методів аутентифікації, таких як паролі, двохфакторна аутентифікація або біометричні дані.

2. Кожен користувач повинен мати відповідні права доступу до функцій системи залежно від його ролі.

Бронювання номерів, а саме автоматизація процесів бронювання:

1. Користувачам повинна бути доступна можливість швидко та зручно бронювати номери в готелі через систему.

2. Адміністраторам готелю потрібна можливість переглядати, редагувати та скасовувати бронювання, а також відстежувати їх статус.

Облік платежів, куди входить звітність та аналітика:

1. Система повинна здійснювати облік фінансових транзакцій, таких як оплата за проживання в готелі, додаткові послуги тощо.

2. Адміністраторам готелю потрібна можливість генерувати звіти та аналітику щодо фінансових операцій, прибутків, витрат та інших фінансових показників для ефективного управління готелем.

Детальніше переглянути функціональні вимоги можна на діаграмі, що зображена на рисунку 2.1 та специфікації вимог зроблені за допомогою менеджера специфікацій (Specification Manager) Enterprise Architect, що зображені на рисунках 2.2-2-4.

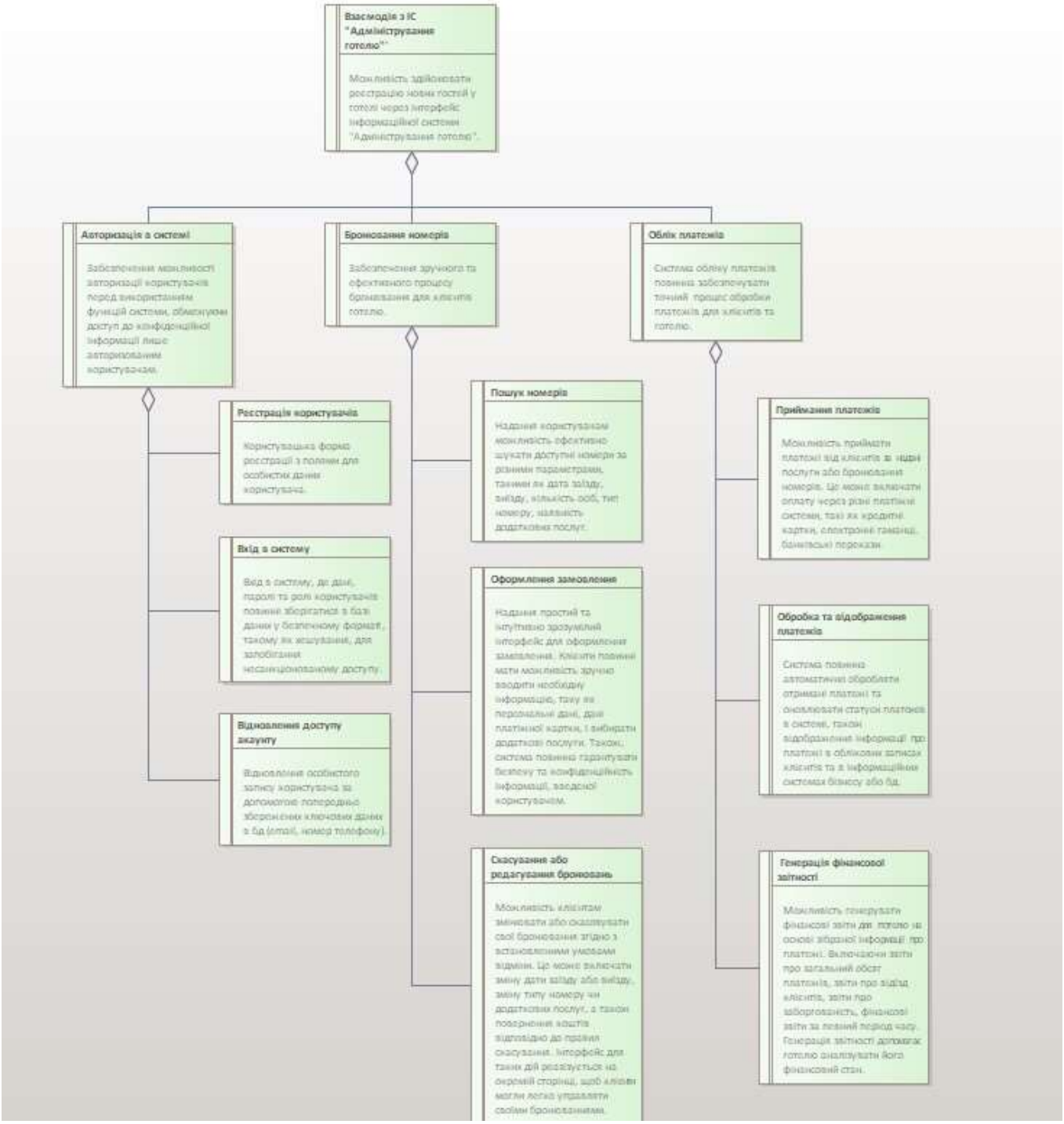


Рисунок 2.1 – Діаграма функціональних вимог

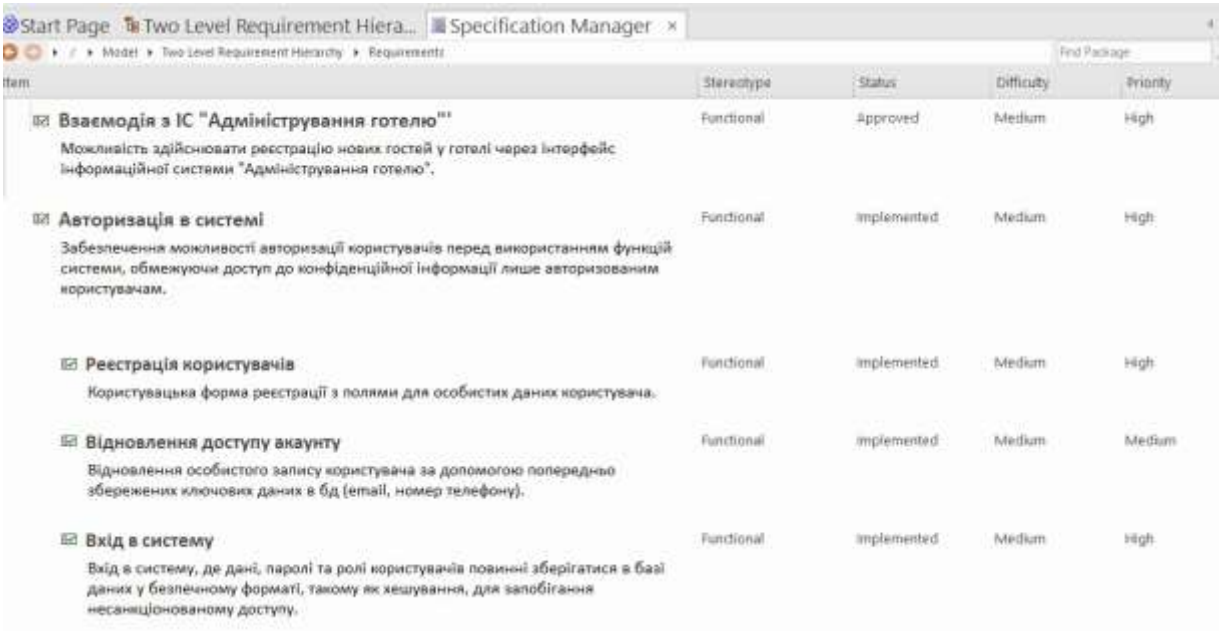


Рисунок 2.2 – Специфікації вимог «Авторизація в системі»

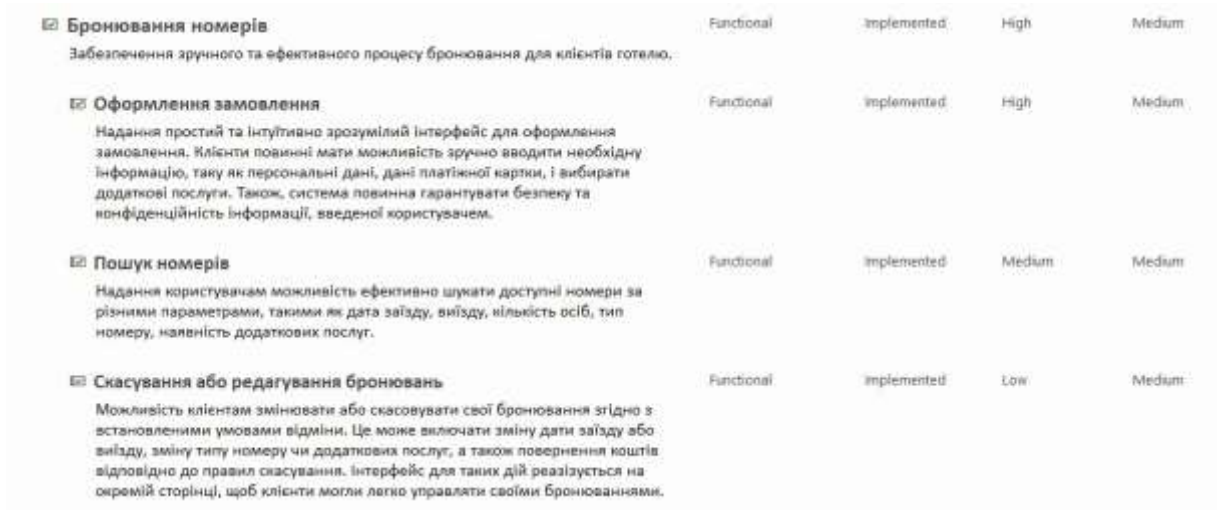


Рисунок 2.3 – Специфікації вимог «Бронювання номерів»

Рисунок 2.4 – Специфікації вимог «Облік платежів»

2.1.2 Нефункціональні вимоги до системи

Нефункціональні вимоги системи адміністрування готелю визначають критерії, які не стосуються безпосередньо основних функцій системи, але мають вирішальне значення для її ефективності та придатності до використання.

Вони охоплюють аспекти, такі як продуктивність, надійність, безпека, зручність використання, масштабованість, резервне копіювання та відновлення даних, швидкодія, сумісність та інтеграція з іншими системами.

Ключовою нефункціональною вимогою є забезпечення швидкості та ефективності роботи системи навіть при великому обсязі даних та великій кількості одночасних користувачів. Вимагається максимальний час відповіді системи на запити користувачів не більше 3 секунд, оптимізація роботи бази даних для швидкого доступу та можливість автоматичного масштабування ресурсів системи при збільшенні навантаження.

Не менш важливою вимогою є забезпечення стабільної та безперебійної роботи системи, а також захист конфіденційності та цілісності даних. Система повинна мати гарантований час безвідмовної роботи протягом місяця не менше 99%, механізми автоматичного відновлення після можливих збоїв, а також захист від несанкціонованого доступу, шифрування даних та моніторингу активності користувачів.

Ще однією нефункціональною вимогою є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів системи. Це включає в себе зручну навігацію, доступність функціоналу та можливість персоналізації інтерфейсу відповідно до потреб користувачів.

Також потрібно забезпечити регулярне резервне копіювання та відновлення даних для запобігання втраті інформації та забезпечення надійності роботи системи в разі непередбачуваних ситуацій.

Загальні нефункціональні вимоги можна оглянути на рисунку 2.5 та специфікації вимог зроблені за допомогою менеджера специфікацій (Specification Manager) Enterprise Architect, що зображені на рисунках 2.6-2-10.

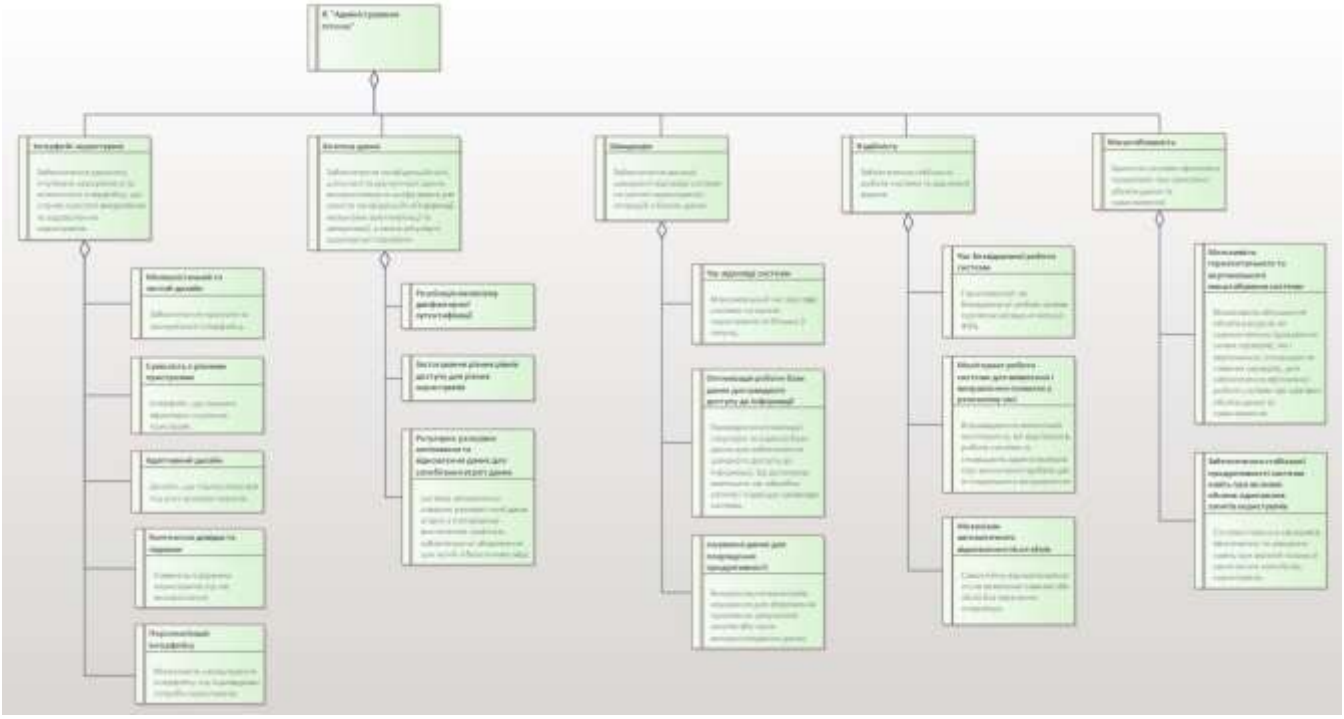


Рисунок 2.5 – Діаграма нефункціональних вимог

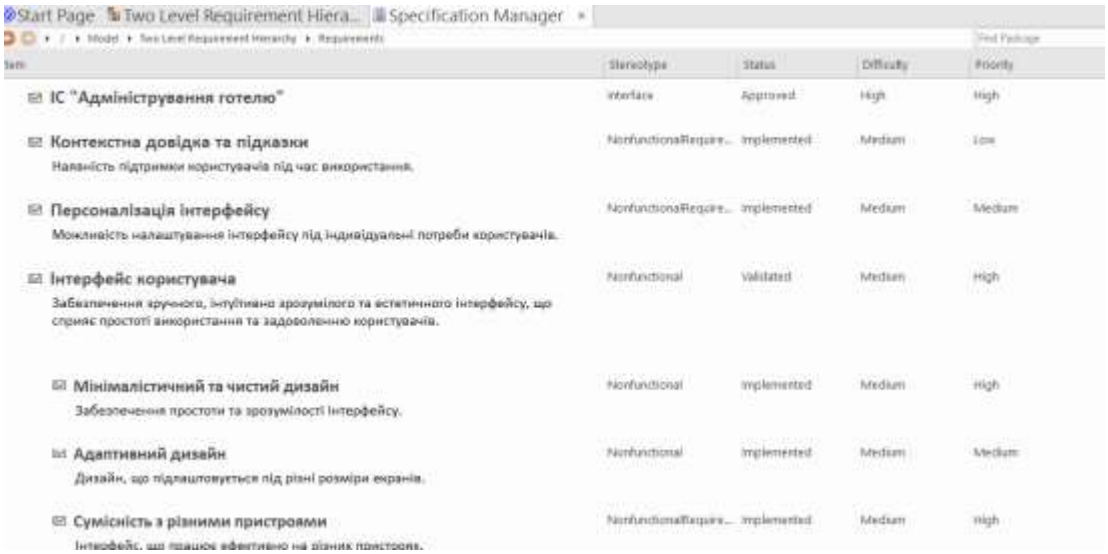


Рисунок 2.6 – Специфікації нефункціональних вимог «Інтерфейс користувача»

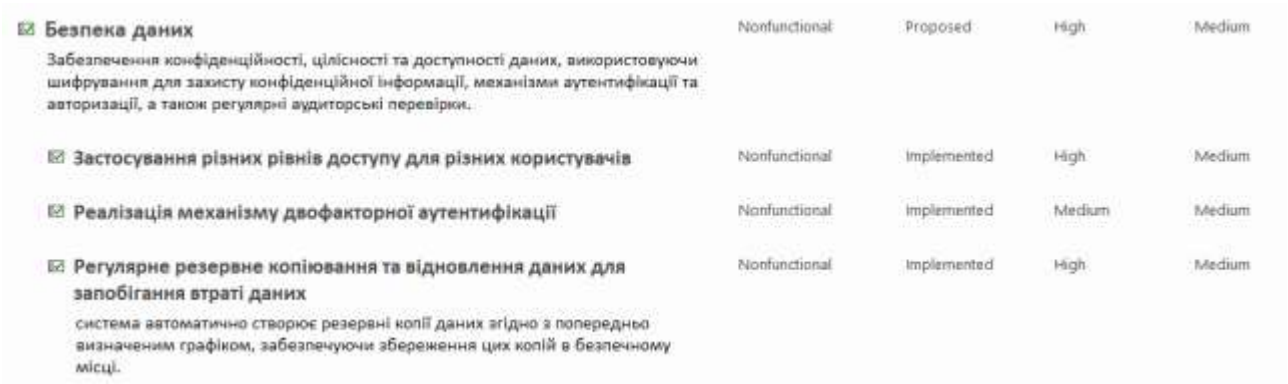


Рисунок 2.7 – Специфікації нефункціональних вимог «Безпека даних»

Рисунок 2.8 – Специфікації нефункціональних вимог «Швидкодія»

Рисунок 2.9 – Специфікації нефункціональних вимог «Надійність»

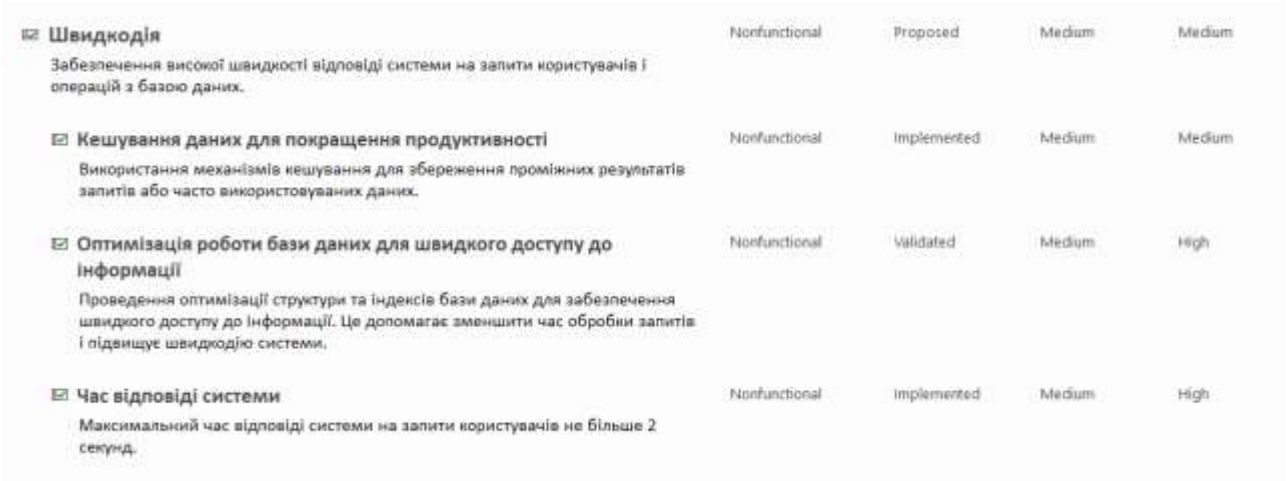


Рисунок 2.10 – Специфікації нефункціональних вимог «Масштабованість»

2.1.3 Бізнес-вимоги до системи

Сформовані бізнес-цілі визначають основну мету створення системи адміністрування готелю та забезпечують відповідність її функціональності потребам користувачів і вимогам готельного ринку.

Однією з цілей є автоматизація та оптимізація управління готельними процесами, такими як бронювання номерів, облік гостьових відвідувань та управління персоналом. Додатковою метою є покращення якості обслуговування гостей, забезпечуючи швидку реакцію на їхні запити та покращення комунікації між різними відділами готелю.

Бізнес-вимоги визначають функціональність та характеристики, які має мати інформаційна система для задоволення потреб стейкхолдерів і досягнення цілей бізнесу.

В ІС для адміністрування готелю стейкхолдери включають власника готелю, менеджера готелю, адміністратора готелю та технічну підтримку. Кожен стейкхолдер має свої унікальні потреби та очікування від системи, які визначаються вимогами до системи. Наприклад, власник готелю може бажати мати звіти про прибуток та витрати готелю, тоді як адміністратор готелю має потребувати можливість управління бронюваннями номерів та іншими аспектами готельного бізнесу. Технічна підтримка потребує систему для

відстеження та вирішення проблем з використанням системи адміністрування готелю.

Вимоги до системи:

Управління бронюваннями – система повинна забезпечувати можливість ефективного прийому, збереження, оновлення, відміни та зміни бронювань номерів готелю, для забезпечення належного контролю за доступністю номерів та ефективного використання готельних ресурсів.

Облік гостьових даних – система повинна збирати, зберігати та обробляти особисті дані гостей, включаючи інформацію про реєстрацію, перевірку особи, платіжні дані та інші важливі деталі, забезпечуючи ефективне управління базою даних гостей та забезпечити персоналу готелю доступ до необхідної інформації.

Фінансове управління – система повинна забезпечувати можливість відстежування фінансових транзакцій, включаючи оплату за проживання, додаткові послуги та інші витрати.

Управління персоналом – система повинна надавати інструменти для управління робочим графіком, відпустками та відсутністю персоналу.

Забезпечення безпеки – гарантування захисту конфіденційної інформації гостей, включаючи персональні дані та фінансові інформації, від несанкціонованого доступу та зловживання. Також, це може включати в себе застосування шифрування даних, механізми аутентифікації та системи регулярних аудитів безпеки.

Далі наведено діаграму бізнес-вимог та бізнес-цілей кожної особи яка взаємодіє з системою (див. рисунках 2.11-2.13). Основними особами, що взаємодіють з системою є власник готелю, менеджери готелю, адміністратори готелю та технічна підтримка.

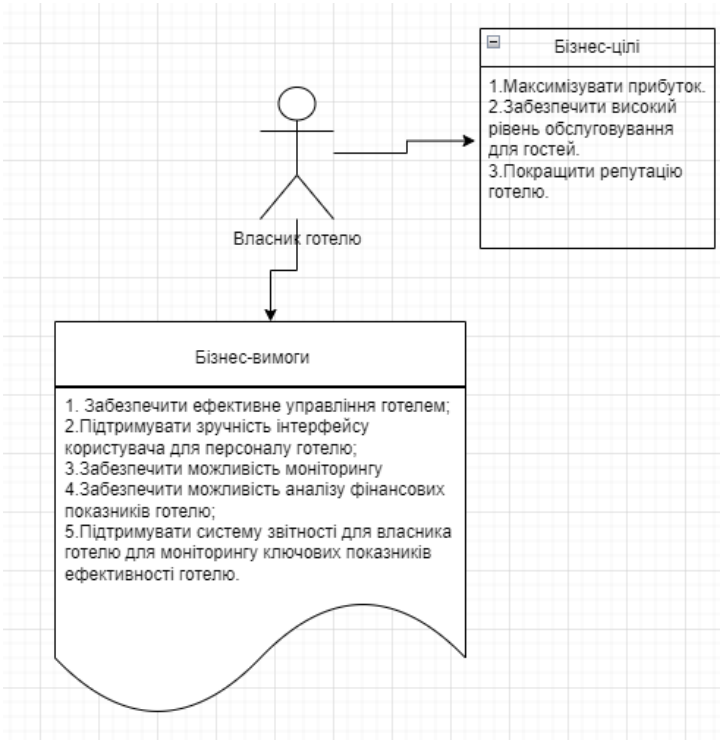


Рисунок 2.11 – Діаграма визначення бізнес-цілей та бізнес-вимог власника готелю

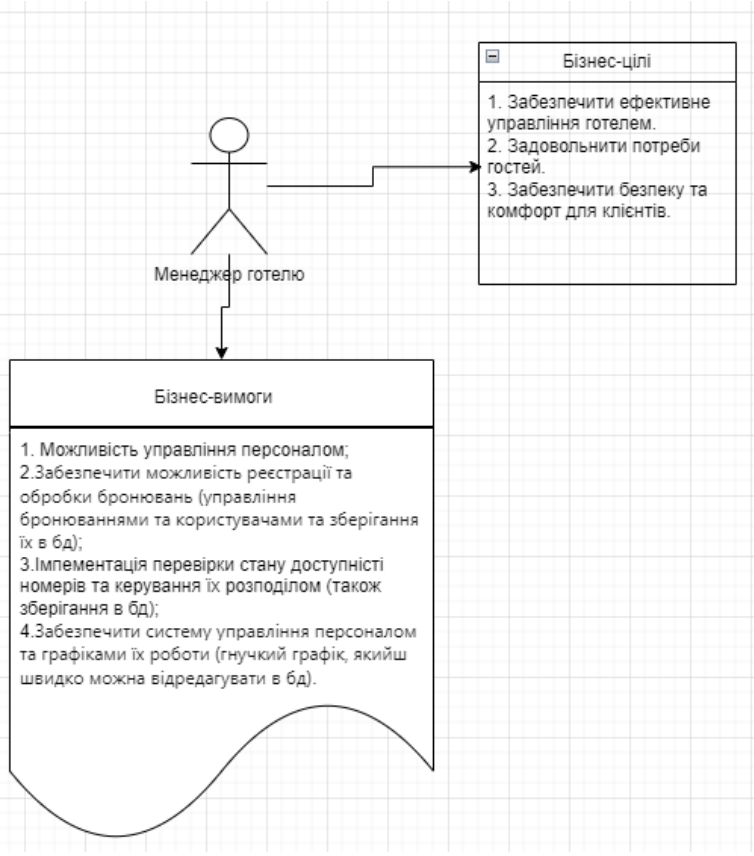


Рисунок 2.12 – Діаграма визначення бізнес-цілей та бізнес-вимог менеджера готелю

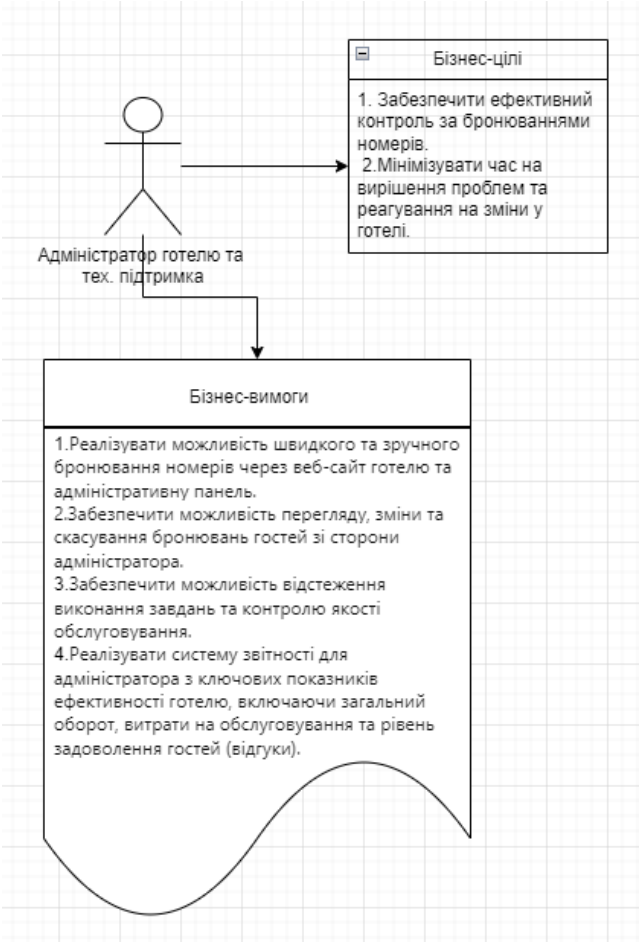


Рисунок 2.13 – Діаграма визначення бізнес-цілей та бізнес-вимог адміністратора готелю та технічної підтримки

2.2 Обґрунтування методології проєктування та функціональна модель задачі

Для кваліфікаційного бакалаврського проєкту на тему проєктування інформаційнійної системи для адміністрування готелю, я обрав методологію розробки – Rapid Application Development (RAD). Використання RAD дозволяє оптимізувати процес розробки за рахунок швидкого створення прототипів і неперервних ітерацій. На відміну від більш традиційних методів, як Waterfall, RAD фокусується на практичному використанні програмного забезпечення та залученні зворотного зв'язку від користувачів.[30] Цей підхід забезпечує гнучкість та адаптивність у процесі розробки, що є особливо важливим у контексті динамічно змінюваних вимог готельної індустрії. Основні переваги

RAD включають підвищення рівня задоволеності клієнтів, поліпшення управління ризиками та прискорення процесу розробки та впровадження системи, що забезпечує ефективне взаємодія з замовником на ранніх стадіях проєкту.[31]

Окрім цього, у проєкті застосовується об'єктно-орієнтований підхід (ООП), який дозволяє ефективно організувати структуру програмного коду. Завдяки використанню принципів інкапсуляції, поліморфізму та спадкування, ООП сприяє створенню модульної, гнучкої та масштабованої архітектури.[32] Це особливо важливо для готельної інформаційної системи, де необхідно ефективно управляти різноманітними бізнес-процесами, від бронювання номерів до управління персоналом та фінансами. Використання ООП дозволяє не тільки поліпшити якість та зрозумілість проєкту, але й забезпечує легкість у підтримці та розширенні системи в майбутньому.

RAD обрано з огляду на такі переваги, які є ключовими для успішної реалізації даного проєкту:

– підвищення рівня задоволеності клієнтів, а саме залучення замовника до процесу розробки через обговорення прототипів та збір зворотного зв'язку дозволить створити систему, яка максимально відповідає його потребам та очікуванням;

– поліпшення управління ризиками, ітеративний підхід RAD дозволяє виявляти та усувати потенційні проблеми на ранніх стадіях розробки, що знижує ризик виникнення критичних помилок та затримок;

– прискорення процесу розробки та впровадження системи, RAD дозволяє скоротити час, необхідний для створення та запуску системи, що є критичним фактором у конкурентному середовищі готельної індустрії.[31]

RAD складається з чотирьох ключових фаз:

– **Вимоги:**

Збір вимог здійснюється шляхом проведення інтерв’ю з власником готелю, менеджерами та адміністраторами, а також шляхом аналізу існуючих систем та

конкурентів. Для документування вимог використовуватиметься менеджер специфікацій Enterprise Architect.

– **Проєктування:**

Для проєктування системи використовуватимуться UML-діаграми (діаграми класів, прецедентів, послідовностей, станів, дій, компонентів, розгортання) для візуалізації структури та поведінки системи.

Будуть створені інтерактивні прототипи ключових елементів інтерфейсу за допомогою Figma для демонстрації роботи системи.

– **Побудова:**

Розробка ведеться з використанням практик Agile Development, таких як Scrum, з короткими спринтами та щоденними scrum-мітингами для ефективного контролю процесу та швидкого реагування на зміни.[29]

– **Впровадження:**

Впровадження системи буде поетапним, починаючи з пілотного запуску для обмеженого кола користувачів.

Перед запуском буде проведено комплексне тестування системи, включаючи функціональне тестування, тестування безпеки та навантажувальне тестування.

Використання ООП в рамках проєкту дозволяє:

– поліпшити якість та зрозумілість проєкту, ООП сприяє створенню більш структурованого, зрозумілого та легкого для супроводу коду;

– забезпечити легкість у підтримці, модульна структура коду, створена за допомогою ООП, дозволяє легко додавати нові функції та модифікувати існуючі без суттєвих змін в інших частинах системи.[32]

Ключові принципи ООП, що будуть використані в проєкті:

– **Інкапсуляція:**

Наприклад, дані про клієнта готелю (ПІБ, контактна інформація, історія бронювань) будуть інкапсульовані в окремий клас "User".

Це дозволить приховати внутрішню структуру даних та забезпечити доступ до них лише через чітко визначені методи, що підвищує безпеку та цілісність даних.

– **Поліморфізм:**

Наприклад, різні типи номерів (стандарт, люкс, бізнес) можуть бути представлені як підкласи класу "Room".

Кожен підклас матиме свої особливості (наприклад, ціна, площа, набір послуг), але всі вони будуть реалізовувати спільний інтерфейс бронювання в класі "RoomReservation".

Це дозволяє створити більш гнучкий код, який легко адаптувати до різних типів номерів.

– **Спадкування:**

Наприклад, клас "Admin" може успадковувати властивості та методи класу "User", додаючи до них специфічні функції управління системою.

Це дозволяє уникнути дублювання коду та покращити його структуру.

Переваги ООП для проєкту:

– модульність – ООП сприяє розбиттю коду на окремі модулі (класи), які легко тестувати та підтримувати;

– гнучкість та масштабованість – ООП дозволяє легко додавати нові функції та модулі до системи без суттєвих змін в її основній структурі;

– підвищення якості коду – ООП сприяє створенню більш зрозумілого, структурованого та легкого для супроводу коду.[32]

Функціональна модель задачі (блоки системи):

1. Блок управління користувачами:

– реєстрація та авторизація користувачів;

– управління правами доступу.

2. Блок управління бронюваннями:

– пошук та бронювання номерів;

– управління статусами бронювань;

– робота з клієнтськими даними.

3. Блок управління номерним фондом:

– додавання, редагування та видалення номерів;

– контроль доступності номерів;

– управління цінами та тарифами.

4. Блок фінансового управління:

– облік оплат та послуг;

– формування рахунків та звітів.

5. Блок звітності та аналітики:

– генерація звітів про завантаженість номерів;

– аналіз фінансових показників.

Потоки даних:

– клієнт відправляє запити на бронювання, отримує інформацію про номери, здійснює оплату;

– адміністратор керує бронюваннями, номерним фондом, персоналом, формує звіти;

– система обробляє запити, зберігає та обробляє дані, формує звіти.

Приклади сценаріїв використання:

Сценарій 1: Бронювання номера клієнтом:

1. Клієнт шукає номер на сайті готелю.

2. Система відображає доступні номери.

3. Клієнт обирає номер та вводить свої дані.

4. Система реєструє бронювання та відправляє підтвердження клієнту.

Сценарій 2: Управління номерним фондом адміністратором:

1. Адміністратор переглядає список номерів та їх доступність.

2. Адміністратор змінює статус номера (наприклад, "зайнятий", "вільний").

3. Адміністратор редагує інформацію про номер (ціна, опис, фото).

Сценарій 3: Генерація звітів власником готелю:

1. Власник готелю обирає тип звіту (наприклад, "завантаженість номерів", "фінансові показники").

2. Система формує звіт за обраний період.

3. Власник готелю аналізує дані звіту.

Поєднання методології RAD та ООП у розробці інформаційної системи забезпечує комплексний підхід до вирішення задачі. Воно дозволяє не тільки оперативно реагувати на зміни вимог та очікувань клієнтів, але й створити високофункціональну, гнучку та легко масштабовану систему, що відповідає потребам сучасного готельного бізнесу.

2.3 Моделювання предметної галузі

Основна мета цієї інформаційної системи для адміністрування готелю полягає у впорядкуванні та оптимізації управління всіма аспектами готельного бізнесу. Ця система має спрощувати та автоматизувати процеси, що пов'язані з обслуговуванням гостей, керуванням персоналом, бронюванням номерів, обліком фінансів, а також забезпечувати безпеку даних та конфіденційність інформації. Головна мета полягає в покращенні якості обслуговування гостей, ефективності управління готелем та забезпеченні задоволення потреб як гостей, так і персоналу готелю.

На системній діаграмі верхнього рівня «Інформаційна система для адміністрування готелю» представлено головний процес системи, який є центральним елементом теми проєктування. Основним об'єктом на цьому рівні є сама інформаційна система, що забезпечує функціонал для ефективного управління готелем. Також, визначаються споживачі результатів – користувачі системи, персонал готелю, адміністратори, які мають доступ до функціоналу, а також учасники процесу, що здійснюють дії в межах системи.

Модель предметної області подано ієрархією об’єктно-процесних діаграм мовою моделювання ОРМ (див. рисунки 2.14-2.20).[33]

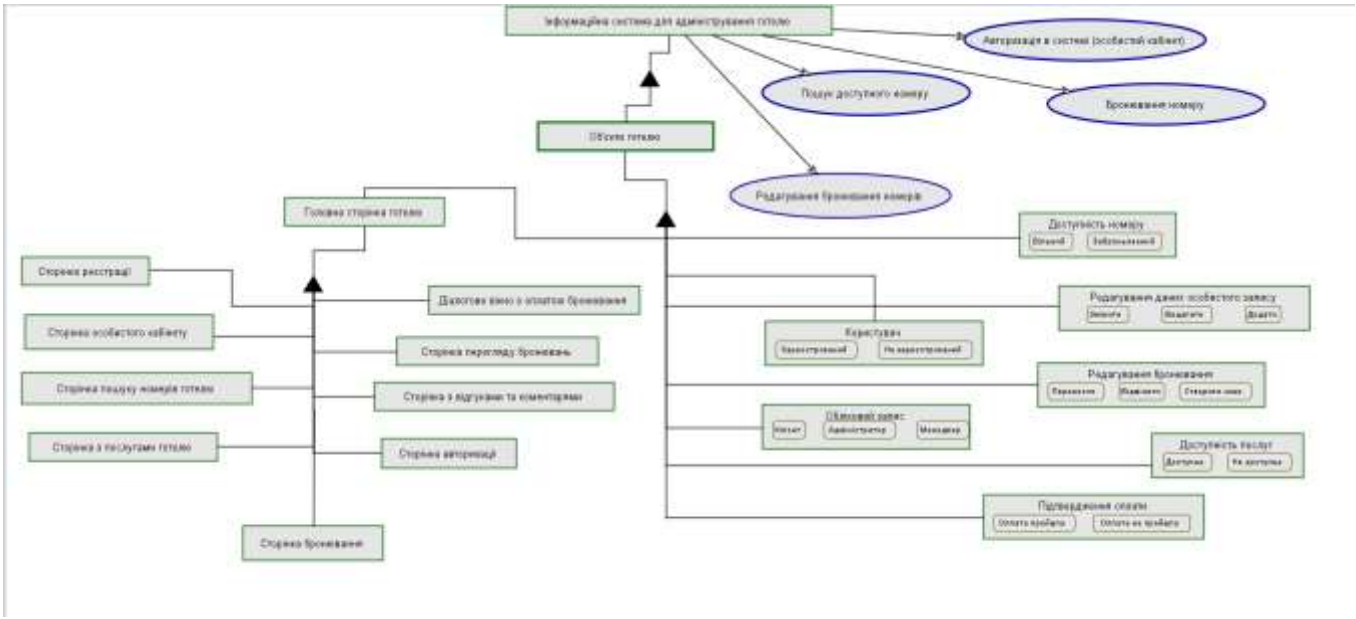


Рисунок 2.14 – Головна діаграма

На діаграмах наступного рівня деталізується функціонал системи через розгортання підпроцесів та асоційованих об'єктів. Перший підпроцес, «Авторизація в системі», визначає послідовність дій, які користувач повинен виконати для входу в систему, включаючи аутентифікацію та перевірку доступу.

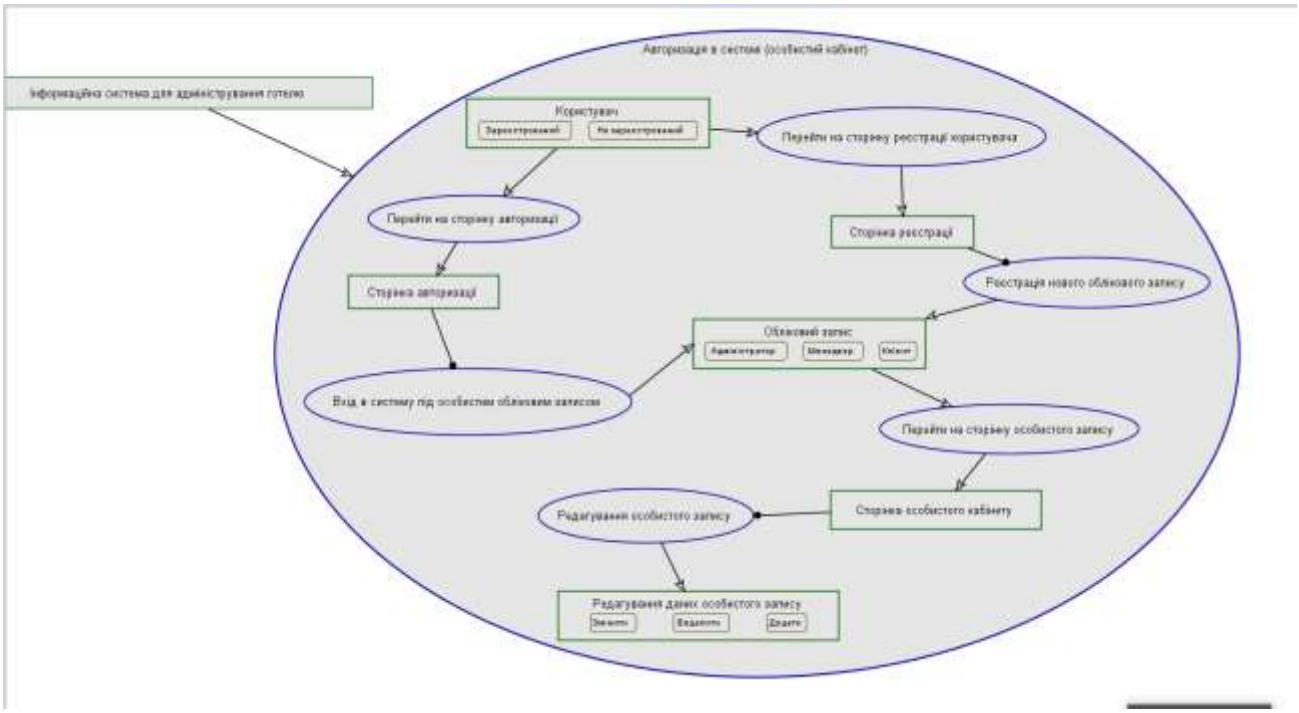


Рисунок 2.15 – Діаграма «Авторизація в системі in-zoomed»

Другий підпроцес, «Пошук доступного номеру», описує алгоритм для пошуку вільного номеру в готелі, враховуючи введені користувачем критерії.

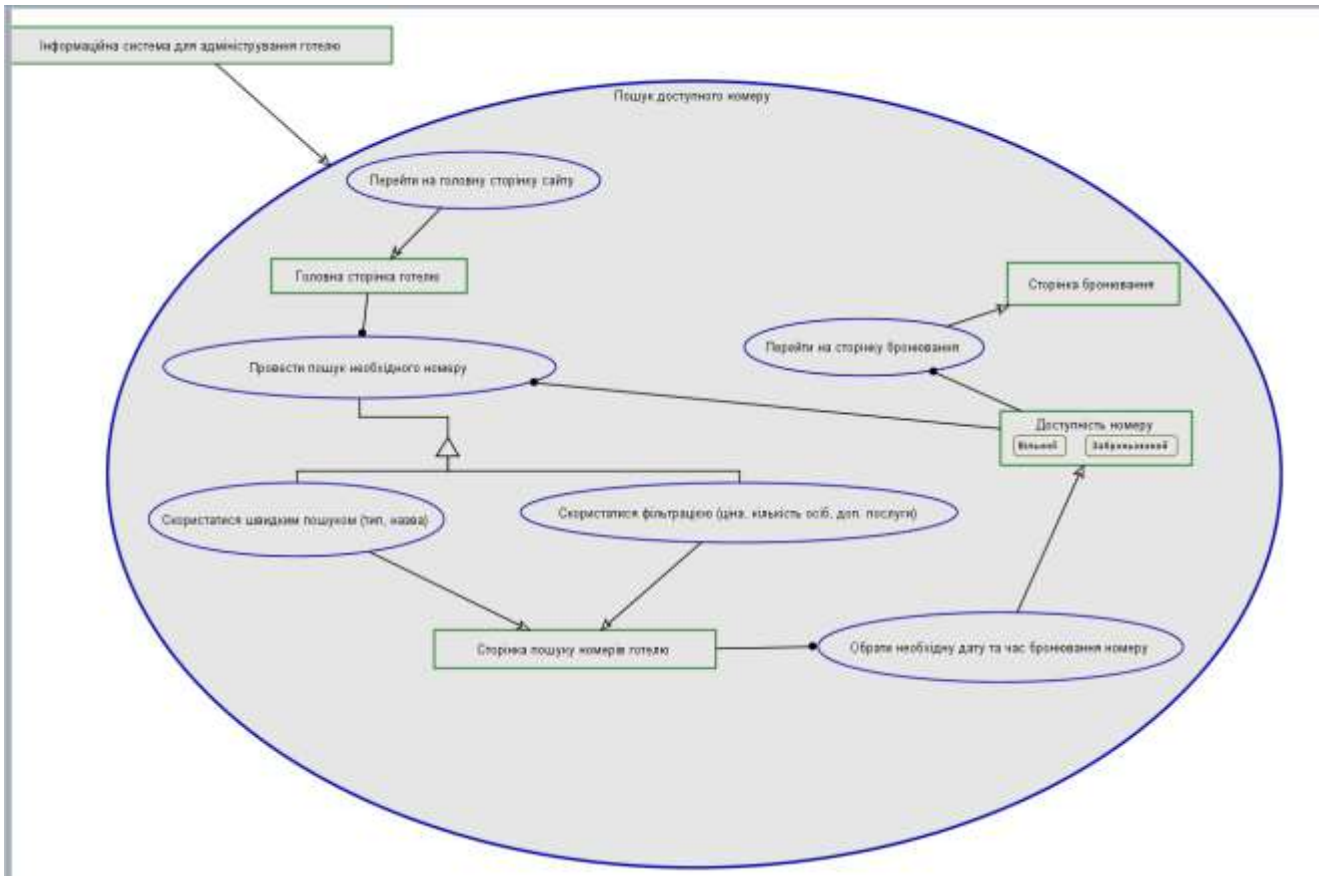


Рисунок 2.16 – Діаграма «Пошук доступного номеру in-zoomed»

Третій підпроцес, «Бронювання номеру», розкриває процедуру бронювання обраного номеру, включаючи введення дат заїзду та виїзду та вибір типу номеру.

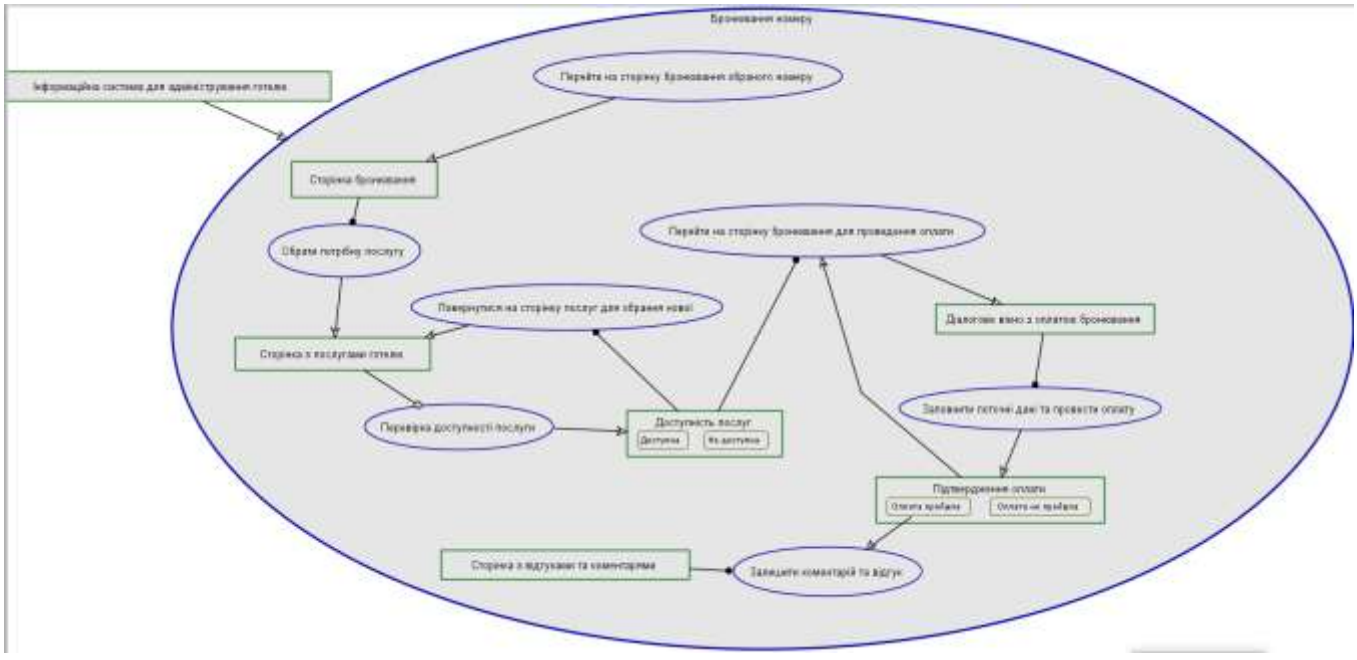


Рисунок 2.17 – Діаграма «Бронювання номеру in-zoomed»

Четвертий підпроцес, «Редагування бронювання номерів», визначає можливі дії адміністратора готелю щодо зміни або скасування існуючих бронювань.

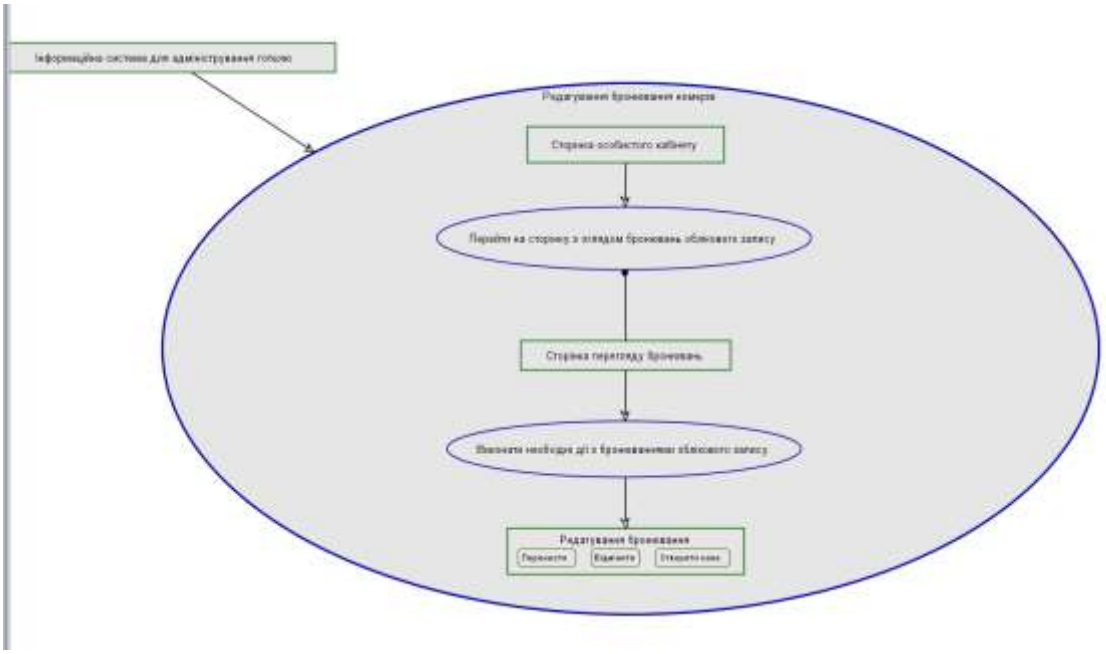


Рисунок 2.18 – Діаграма «Редагування бронювання номерів in-zoomed»

Механізм розгортання описує процес впровадження системи, залучені ресурси та послуги, необхідні для її функціонування, такі як сервери, бази даних та інші зовнішні системи. Кожен з цих підпроцесів дозволяє докладніше розкрити функціонал та процеси, які відбуваються у межах інформаційної системи для адміністрування готелю.

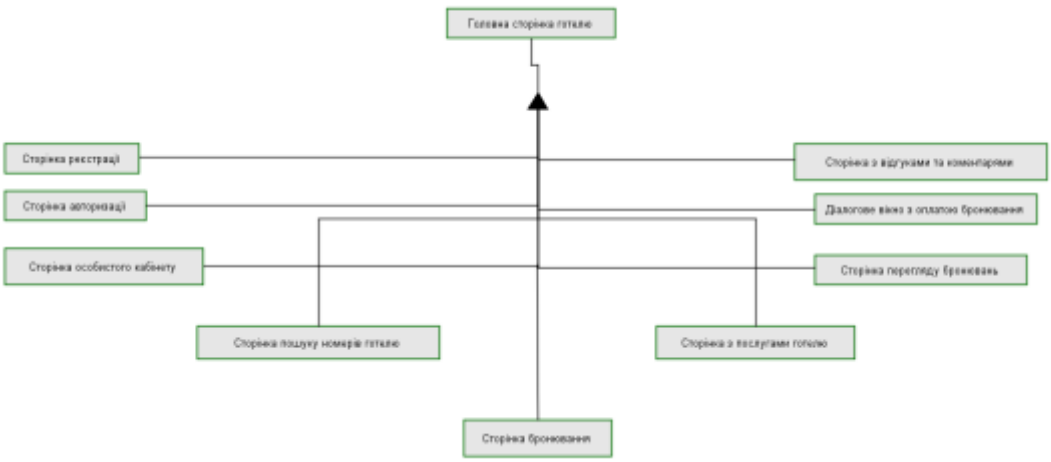


Рисунок 2.19 – Механізм розгортання (огляд всіх сторінок інформаційної системи)

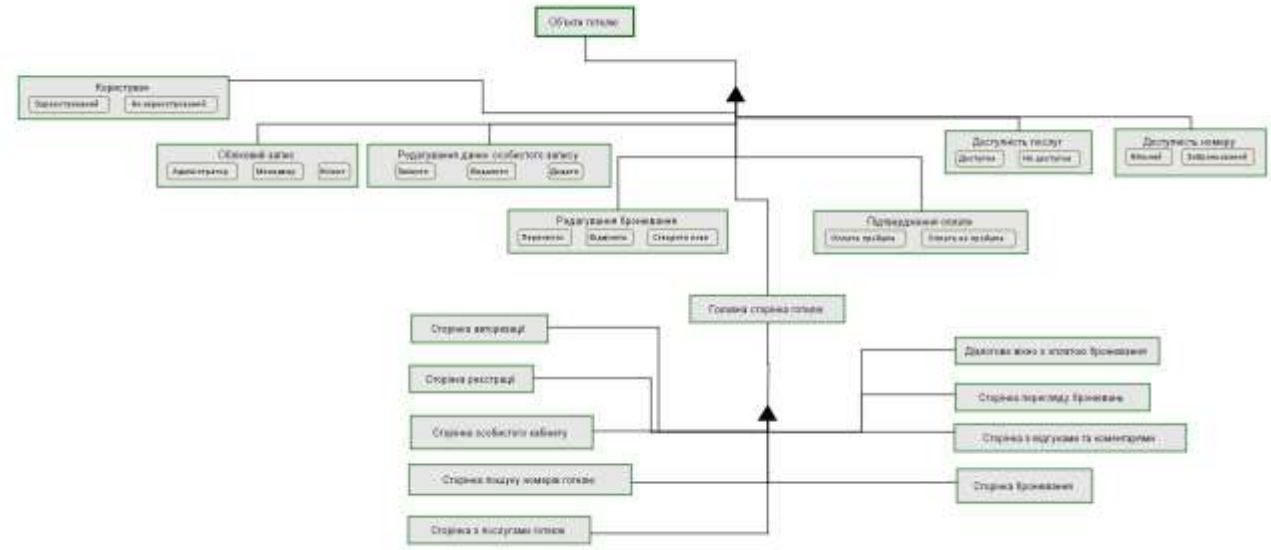


Рисунок 2.20 – Механізм розгортання (огляд всіх об’єктів готелю)

За допомогою середовища OPCAT було створено модель інформаційної системи. У вище наведеній ієрархії OPD-діаграм представлено повний набір процесів і об'єктів разом з їхніми станами для інформаційної системи. Рівень деталізації є достатнім для визначення функціональних вимог до проєктованої інформаційної системи.

2.4 Постановка задачі

Основна мета проєкту полягає в розробці комплексної інформаційної системи, призначеної для автоматизації управлінських та операційних процесів в готельному бізнесі. Система покликана оптимізувати внутрішні процеси, забезпечити ефективне управління ресурсами, покращити обслуговування клієнтів та збільшити загальну продуктивність готелю.[34]

Задача полягає у створенні багатофункціональної, інтуїтивно зрозумілої, безпечної та надійної інформаційної системи, яка зможе задовольняти всі потреби сучасного готельного бізнесу. Вона має забезпечити ефективне управління всіма аспектами готельної діяльності, від бронювання та обслуговування клієнтів до управління персоналом та фінансових операцій. Також важливо, щоб система була здатна адаптуватися до мінливих умов ринку

та зростаючих вимог готелю, забезпечуючи при цьому високий рівень безпеки та конфіденційності даних.

Проаналізувавши та використавши функціональні та нефункціональні вимоги дозволять нам створити ефективну, надійну та користувацько-орієнтовану інформаційну систему для адміністрування готелю, яка відповідатиме сучасним вимогам та тенденціям готельної індустрії.

Вихідна інформація зберігатиметься у базі даних. Вихідна інформація передбачає:

1. Структурування даних – база даних повинна мати чітко визначені таблиці для кожної категорії вихідної інформації, таких як бронювання, фінансові звіти, відгуки клієнтів, діяльність персоналу. Використання реляційних зв'язків між таблицями для відображення взаємозв'язків між різними аспектами даних (зв'язок між бронюваннями та клієнтами).

2. Ефективна нормалізація для забезпечення цілісності, зменшення дублювання даних та полегшення управління даними.

3. Безпека даних, а саме використання шифрування та інших механізмів безпеки для захисту конфіденційної інформації. Також, реалізація системи контролю доступу для обмеження доступу до чутливих даних.

4. Індексація для оптимізації запитів – ключових стовпців для підвищення швидкості пошуку та витягування даних.

5. Забезпечення можливості інтеграції з іншими системами та додатками, наприклад, з системою управління відносинами з клієнтами (CRM) або фінансовими системами.

6. Планування архівації старих даних та регулярне створення резервних копій для запобігання втраті даних.

7. Ведення журналів змін для відстеження коректив, оновлень або видалення даних.

Розробка і впровадження інформаційної системи для управління готелем є складним та багатогранним процесом, що включає детальне планування,

глибокий аналіз потреб, розробку високої якості, та ґрунтовне тестування. Цей процес передбачає ефективне розгортання системи в робочому середовищі готелю.

На рисунку 2.21 представлена інформаційна модель задачі, яка відображає взаємодію між різними компонентами інформаційної системи. Вона включає вузли для клієнтів, адміністратора системи, веб-системи та менеджера, а також вхідні дані, такі як реєстрація, замовлення, перегляд та редагування інформації про номери, послуги та бронювання.

Інформаційна модель також охоплює внутрішнє адміністрування, включаючи управління запитами на зміну послуг, редагування інформації про номери, і управління бронюваннями.

Система також включає функції, такі як аналітика та звітність, які дозволяють адміністрації готелю здійснювати фінансовий контроль і стратегічне планування. Можливість аналізувати дані про бронювання, виручку та відгуки клієнтів допомагає керівництву у прийнятті обґрунтованих рішень.

Модель також відображає взаємодію з базою даних, яка включає таблиці для користувачів, їхніх ролей, логів активності, типів номерів, сервісів, бронювань та фінансових документів. Система забезпечує різні форми вхідної та вихідної інформації, що відповідають на запити клієнтів і дозволяють адміністрації готелю ефективно управляти операціями.

Для описаної інформаційної моделі задачі використовуються різні способи подання інформації, зокрема екранні форми, електронні документи та файли.

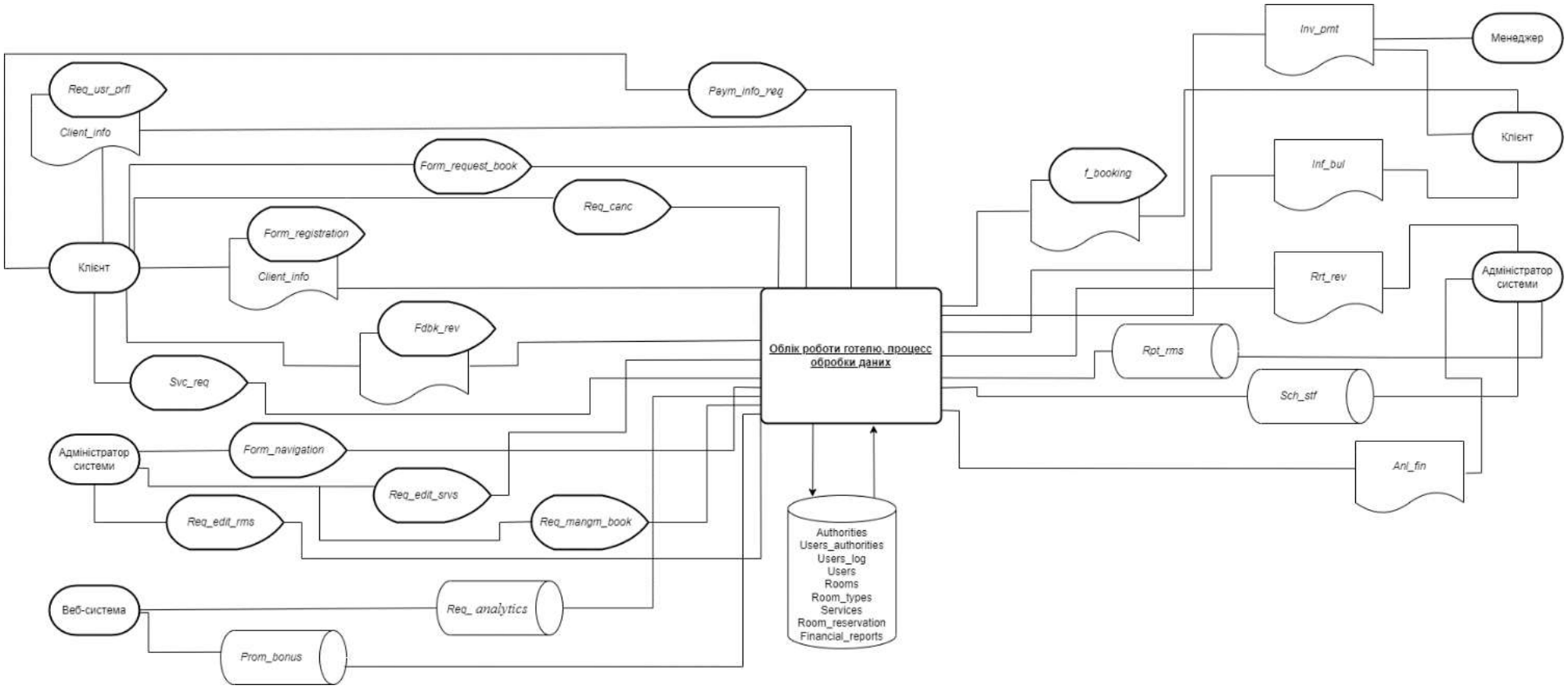


Рисунок 2.21 – Інформаційна модель задач

2.4.1Вихідна інформація

Перелік і опис вихідних повідомлень наведено у табл. 2.1.

Повідомлення вихідної інформації Таблиця 2.1

№ з/п	*Назва вихідного повідомлення*	*Ідентифікатор*	*Форма подання*	*Термін видання і частота надходження*	*Користувач інформації*
1	Підтвердження бронювання	*f_booking*	Екранна форма, документ	Відразу після бронювання	Клієнт
2	Звіт по обігу номерів	*Rpt_rms*	Файл PDF/HTML	Щомісяця	Адміністрація готелю
3	Звіт про відгуки клієнтів	*Rrt_rev*	Документ	По запиту	Управління готелю (Адміністратор)
4	Рахунок-фактура	*Inv_pmt*	Документ	Після оплати	Менеджер/Клієнт
5	Інформаційний бюлетень	*Inf_bul*	Документ	Періодично	Клієнти
6	Графік роботи персоналу	*Sch_stf*	Файл	Щотижня	Адміністрація (для працівників системи)
7	Аналітичні звіти (фінансові) готелю	*Anl_fin*	Документ	Щомісяця	Адміністрація готелю

Таблиця містить повідомлення вихідної інформації, яка генерується інформаційною системою, екранні форми, в яких ця інформація подається, частоту з якою вона надається та осіб, які її отримують.

Кожен вид вихідної інформації має відповідну форму подання та розрахований на конкретного отримувача, забезпечуючи своєчасне та ефективне розподілення інформації для управлінських, операційних та маркетингових потреб готелю.

Результати розв'язання задачі в інформаційній системі адміністрування та управління готелем використовуються для оптимізації ключових бізнес-процесів готелю. Вони включають управління бронюванням, поліпшення обслуговування клієнтів, ефективне фінансове управління, планування та координацію роботи персоналу.

Структурними одиницями інформації в системі є детальні дані про клієнтів, номерний фонд, фінансові операції та операційна діяльність. Інформація про клієнтів допомагає персоналізувати обслуговування та підвищувати лояльність. Дані про номерний фонд забезпечують оптимальне управління наповненістю готелю. Фінансова інформація сприяє точному бухгалтерському обліку та фінансовому плануванню. Операційні дані необхідні для координації робочих процесів та забезпечення якісного сервісу.

2.4.2Вхідна інформація

Вхідні повідомлення для здійснення контролю за основними операціями для інформаційної системи адміністрування готелем. Перелік і опис вхідних повідомлень наведено у табл. 2.2.

Повідомлення вхідної інформації Таблиця 2.2

№ з/п	***Назва вхідного повідомлення***	***Ідентифікатор***	***Форма подання***	***Термін і частота надходження***	***Джерело***
1	Форма реєстрації клієнтів	*Form_registration*	Екранна форма	По мірі реєстрації клієнтів	Дані від клієнта
2	Запит на редагування особистого кабінету	*Req_usr_prfl*	Екранна форма	По мірі запитів на редагування особистого кабінету	Дані від клієнта
3	Форма навігації типів номерів готелю	*Form_navigation*	Екранна форма	По мірі здійснення вибору типу номера готелю	Дані від адміністратора

4	Запит на бронювання номеру	*Form_request_book*	Екранна форма	По мірі запитів на бронювання	Дані від клієнта (веб-система)
5	Запит на відміну бронювання	*Req_canc*	Екранна форма	За потребою	Клієнт (електронна пошта)
6	Інформація про клієнта	*Client_info*	Документ	При реєстрації або при редагуванні в особистому кабінеті	Клієнт (реєстрація або при редагуванні на сторінці особистого кабінету)
7	Відгуки та оцінки	*Fdbk_rev*	Екранна форма, документ	Після від'їзду клієнта	Клієнт (веб-система)
8	Запит на додаткові послуги	*Svc_req*	Екранна форма	За потребою	Клієнт (веб-система)
9	Запит на інформацію про платіж	*Paym_info_req*	Екранна форма	Після бронювання, за потребою клієнта	Клієнт (веб-система)
10	Промоакції та бонусна програма	*Prom_bonus*	Файл	По мірі активності клієнта з системою	Дані від веб-системи
11	Запит на редагування номерів готелю	*Req_edit_rms*	Екранна форма	За потребою управлінського відділу готелю	Адміністратор (веб-система)
12	Запит на редагування послуг готелю	*Req_edit_srvs*	Екранна форма	За потребою управлінського відділу готелю	Адміністратор (веб-система)
13	Запит на управління бронюваннями готелю	*Req_mangm_book*	Екранна форма	За потребою управлінського відділу готелю	Адміністратор (веб-система)

14	Запит на аналітику готелю	*Req_analytics*	Файл	За потребою фінансового відділу готелю	Дані від веб-системи

Таблиця включає різні типи інформації, які система може отримувати від клієнтів або через інші внутрішні процеси, уточнюючи метод подання, частоту отримання та джерело вхідних даних.

2.5 Математичне забезпечення та алгоритм функціонування системи

2.5.1 Математичний опис

Розглянемо систематизований підхід до аналізу та оптимізації роботи готелю через кількісні методи.[35] Використання математичних моделей дозволяє нам перетворити кваліативні характеристики готельної індустрії, такі як задоволеність клієнтів та ефективність персоналу, на кількісні показники, що можуть бути виміряні та оптимізовані. Це допомагає керівництву готелю в прийнятті обґрунтованих управлінських рішень, базуючись на даних та об'єктивних метриках.

Центральним елементом нашої моделі є розробка формул, що відображають взаємозв'язки між різними аспектами діяльності готелю.

Визначення середнього обсягу бронювання номерів клієнтами кожного із типів номерів (Standart, Luxe, Deluxe, President, Cottage):

Формула:

$$C_{сер.о.} = \frac{N_{сер.}}{N_{тип.н.}} \quad (2.1), \text{ де:}$$

$C_{сер.о.}$ – середній обсяг бронювання;

$N_{сер.}$ – кількість зареєстрованих клієнтів, які бронювали певний тип номеру;

$N_{тип.н.}$ – кількість доступних типів номерів.

$C_{сер.о.}$ визначає середню кількість разів, коли певний тип номеру був заброньований, і це розраховується шляхом ділення загальної кількості бронювань даного типу на кількість доступних номерів цього типу.

Обчислення середнього часу перебування клієнтів в готелі:

Формула:

$$C_{сер.ч} = \frac{\sum \boldsymbol{T}_{заг.}}{\boldsymbol{N}_{к}} \text{ (2.2), де:}$$

$C_{сер.ч}$ – середній час перебування;

$\sum T_{заг.}$ – загальна тривалість перебування всіх клієнтів;

$N_{к}$ – кількість клієнтів.

Розрахунок виручки від послуг готелю:

Формула:

$$V^{вих}_{вируч.} = P_{цін.} \times N^{вх}_{к} \text{ (2.3), де:}$$

$V^{вих}_{вируч.}$ – виручка від послуг;

$P_{цін.}$ – ціна послуги;

$N^{вх}_{к}$ – кількість послуг, наданих.

Обчислення коефіцієнта зайнятості готельних номерів:

Формула:

$$KZ^{вих}_{зайн.} = \frac{ZN_{кст}}{ZN_{total}} \times 100 \text{ (2.4), де:}$$

$KZ^{вих}_{зайн.}$ – коефіцієнт зайнятості;

$ZN_{кст.}$ – кількість зайнятих номерів;

ZN_{total} – загальна кількість номерів.

Результатом цієї формули буде коефіцієнт зайнятості, виражений у відсотках, що вказує на те, наскільки ефективно використовуються готельні номери.

Визначення коефіцієнта задоволеності клієнтів за типом номеру:

Формула:

$$K_{зад.н.} = \frac{\sum O_{кл.н.} \times F_{відв.н.}}{N_{відг.н.}} \text{ (2.5), де:}$$

$K_{зад.н.}$ – коефіцієнт задоволеності клієнтів за типом номеру;

$O_{кл.н.}$ – оцінка задоволеності від клієнта для певного типу номеру;

$F_{відв.н.}$ – частота відвідувань клієнтом номеру цього типу;

$N_{відг.н.}$ – загальна кількість відгуків для номеру цього типу.

У даній формулі ми множимо оцінку задоволеності клієнта на частоту відвідувань, щоб отримати ваговий показник задоволеності, а потім сумуємо ці значення для всіх відгуків по номеру і ділимо на кількість відгуків, щоб отримати середній коефіцієнт задоволеності за типом номеру. Це дозволить нам дізнатися наскільки задоволені клієнти певним типом номеру та як часто вони його бронюють.

2.5.2Алгоритм розв’язання задачі

Алгоритм розв’язання задачі подано на рисунку 2.22.

Форми користувацького інтерфейсу, наприклад, як Form_registration для реєстрації нових користувачів, Fdbk_rev для введення відгуків, або Form_request_book для оформлення бронювання, є вхідними точками для даних, що надходять від користувачів. Ці форми забезпечують збір необхідної інформації для подальшої обробки системою.

Логічні переходи в системі функціонують як розгалуження, де на основі даних, введених користувачем, або його дій, визначається напрямок подальшої обробки запитів. Ці вузли прийняття рішень критично важливі для забезпечення коректної логіки роботи системи та правильного вибору функціоналу, який буде використовуватися відповідно до намірів користувача.

Процеси, такі як «Забронювати номер готелю» або «Редагувати дані особистого кабінету», представляють собою послідовність дій, які система виконує для обробки запитів користувача. Ці процеси включають в себе не тільки основні функції, але й додаткові операції, такі як управління послугами готелю

чи адміністрування користувацьких профілів, забезпечуючи широкий спектр можливостей для управління готельним бізнесом.

Бази даних є центральними елементами системи, що заберігають всю критично важливу інформацію, включаючи дані про користувачів, історію бронювань, доступні послуги, а також логи системи. Вони виконують функції зберігання, організації та забезпечення доступу до даних, необхідних для операційних та аналітичних потреб бізнесу.

Дії користувача, такі як введення відгуків або здійснення платежів, є взаємодіями, які ініціюють подальші дії в системі.

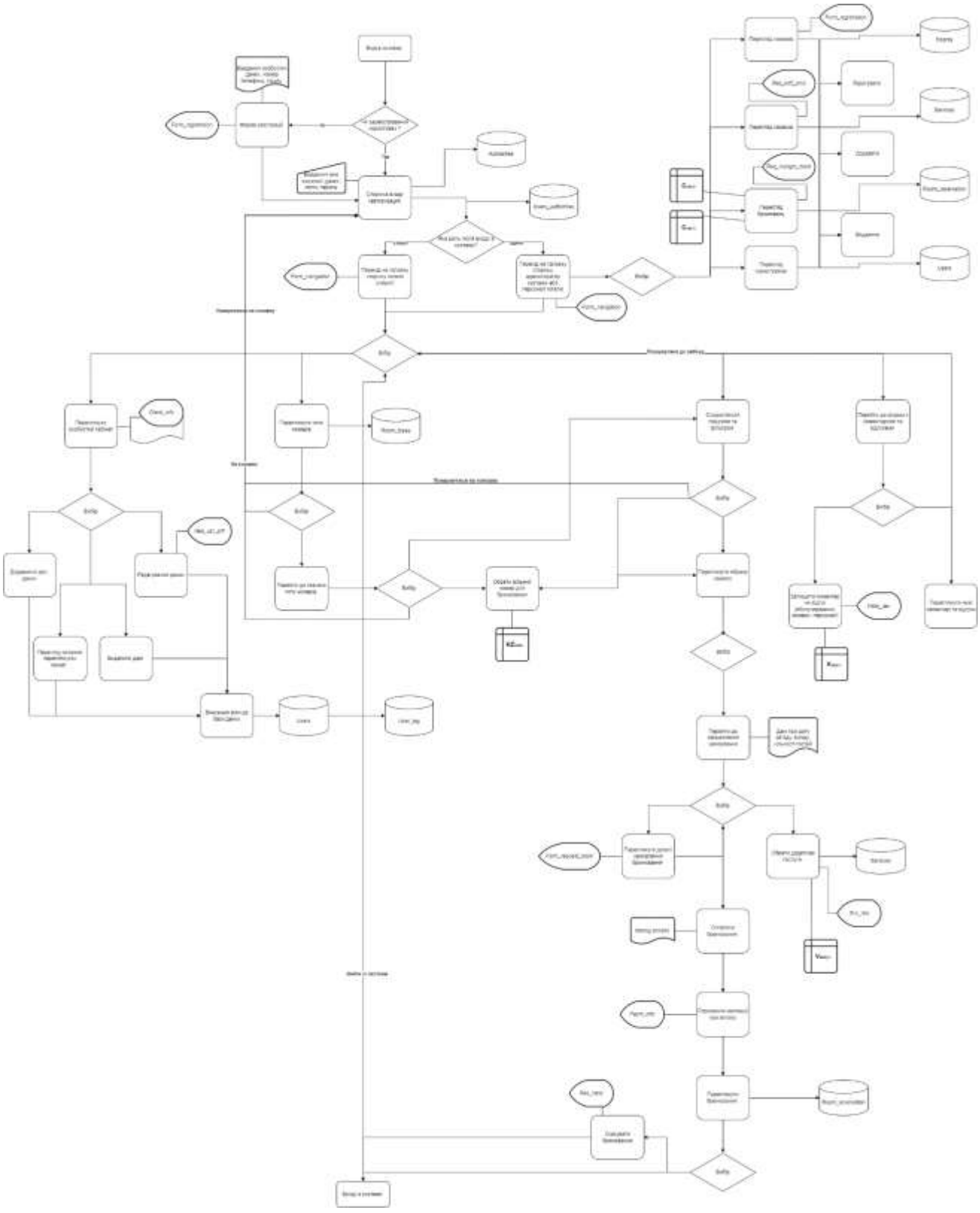


Рисунок 2.22 – Алгоритм розв’язання задачі з обліку роботи готелю, просесу обробки даних

2.6 Моделювання інформаційної системи

Моделювання поведінки інформаційної системи адміністрування готелем включає взаємодію з користувачами, саму систему, обробку даних, бронювання та управління готельними номерами і послугами.[36]

Характеристика моделі поведінки системи для ІС «адміністрування готелю»:

1. Реєстрація користувачів:

– можливість реєструвати нових користувачів, включаючи менеджерів готелів та гостей;

– користувачі можуть ввести свої персональні дані та створити обліковий запис для доступу до системи.

2. Аутентифікація:

– після реєстрації користувачі можуть увійти в систему за допомогою ідентифікаційних даних, таких як ім'я користувача та пароль.

3. Управління бронюванням:

– дозвіл користувачам (гостям) бронювати номери в готелі;

– користувач може вибрати дати заїзду та виїзду, кількість номерів і тип номера, який він хоче забронювати.

4. Управління номерами та послугами:

– адміністратор готелю може додавати, редагувати або видаляти доступні номери та готельні послуги;

– функціональність включає в себе різні методи управління готелем, такі як встановлення цін на номери та послуги, а також оновлення інформації про наявність вільних місць.

5. Обробка платежів:

– можливість обробляти платежі за бронювання та інші готельні послуги;

– прийом платежів від гостей різними методами, такими як кредитна картка або електронний переказ коштів.

6. Управління персоналом:

– модулі для управління персоналом готелю, такі як розподіл завдань, графіки роботи та інформація про заробітну плату.

7. Звітність та аналіз:

– надання звітні та аналітичні дані про завантаження номерів, фінансові показники, рейтинги клієнтів тощо;

– операційна модель системи забезпечує краще розуміння, як різні компоненти взаємодіють один з одним і з користувачами для забезпечення ефективної роботи інформаційної системи управління готелем.

8. Захист даних:

ІС повинна мати надійні механізми захисту конфіденційності, цілісності та доступності даних, а саме:

– шифрування даних;

– контроль доступу до системи та даних;

– аудит доступу;

– резервне копіювання;

– відновлення даних у випадку аварій.

9. Система відгуків та оцінок:

– інтеграція системи для збору відгуків та оцінок дозволяє гостям залишати свої враження про перебування в готелі;

– адміністратори можуть використовувати цю інформацію для покращення якості обслуговування та управління репутацією готелю.

10.Система сповіщень та повідомлень (промоакції):

– інтеграція системи сповіщень дозволяє надсилати автоматичні повідомлення користувачам про підтвердження бронювань, нагадування про дати заїзду та виїзду, спеціальні промоакції та акції готелю;

– користувачі можуть отримувати сповіщення через електронну пошту або SMS за номером телефону.

Для роботи із ІС «Адміністрування готелю» можна виділити дві головні дійові особи:

– користувач (клієнт);

– адміністратор.

Взаємодія дійових осіб та функціональних можливостей системи відображена на діаграмі прецедентів (див. рисунок 2.23).

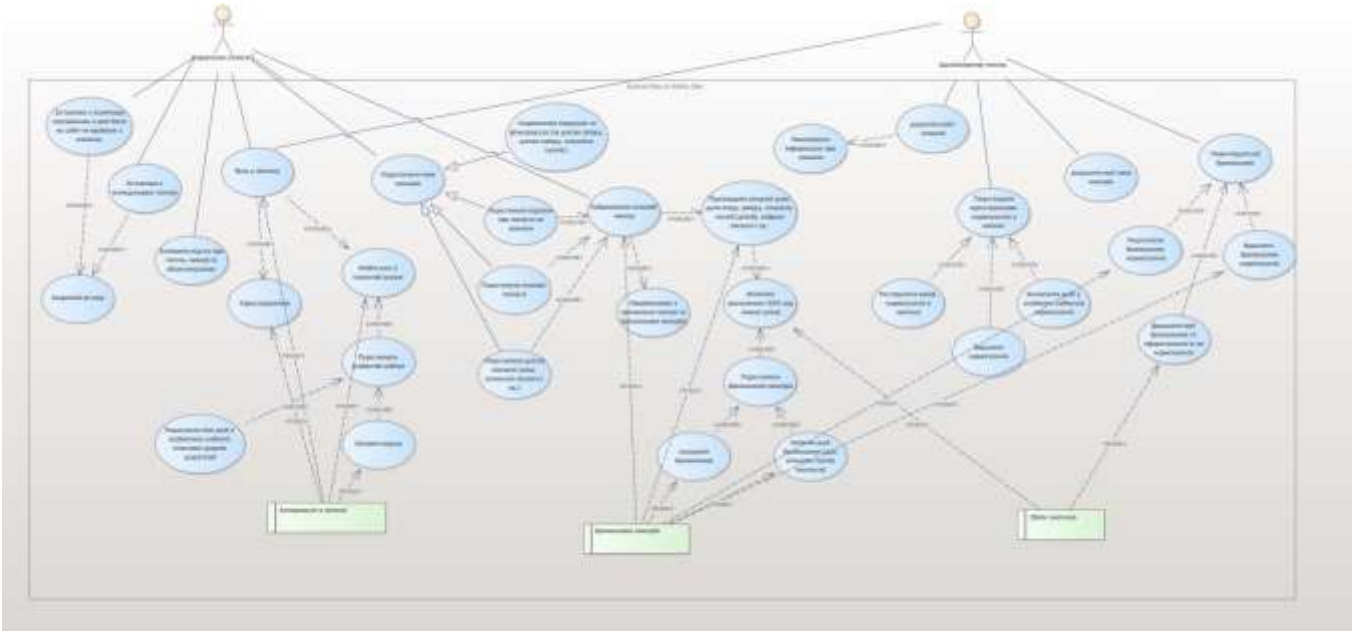


Рисунок 2.23 – Діаграма прецедентів

На діаграмі прецедентів (Use Case) для інформаційної системи адміністрування готелю відображені ключові функціональні можливості системи та їх взаємодія з головними користувачами. Діаграма охоплює різні сценарії, які можуть відбуватися між користувачами (клієнтами, адміністраторами), включаючи бронювання номерів, управління бронюваннями, та обслуговування клієнтів. Виділені також допоміжні процеси, такі як управління відгуками клієнтів та запитами на обслуговування.

Діаграма відображає складність та багатозадачність інформаційної системи, підкреслюючи необхідність її інтеграції з різними аспектами готельного бізнесу. З її допомогою можна зрозуміти, як система буде сприяти

автоматизації бізнес-процесів та підвищенню ефективності управління готельним комплексом, водночас забезпечуючи високий рівень задоволеності клієнтів через оптимізований інтерфейс та зручні функції.

Також зроблено матриці відповідності вимог і прецедентів, які продемонстровані на рисунках 2.24-2.26.

Рисунок 2.24 – Матриця відповідності вимог і прецедентів (частина матриці 1)

Рисунок 2.25 – Матриця відповідності вимог і прецедентів (частина матриці 2)

Рисунок 2.26 – Матриця відповідності вимог і прецедентів (частина матриці 3)

Далі, створені текстові сценарії, які детально описують варіанти використання системи, щоб забезпечити зрозумілість та чіткість у розумінні функціональності. Для цього створюються "нормальні" сценарії, що описують типовий хід подій, а також альтернативні сценарії, які відображають можливі відхилення від "нормального" сценарію. Використали інструмент – розробник сценаріїв (Scenario Builder), для створення деталізованих описів, урахування попередніх та наступних умов, а також обмежень. Кроки кожного сценарію прив'язані до елементів моделі.

На рисунках 2.27-2.28 продемонстровано один з текстового сценарію для прецеденту «Реєстрація користувача», він ілюструє, як виглядає процес реєстрації користувача в інформаційній системі.

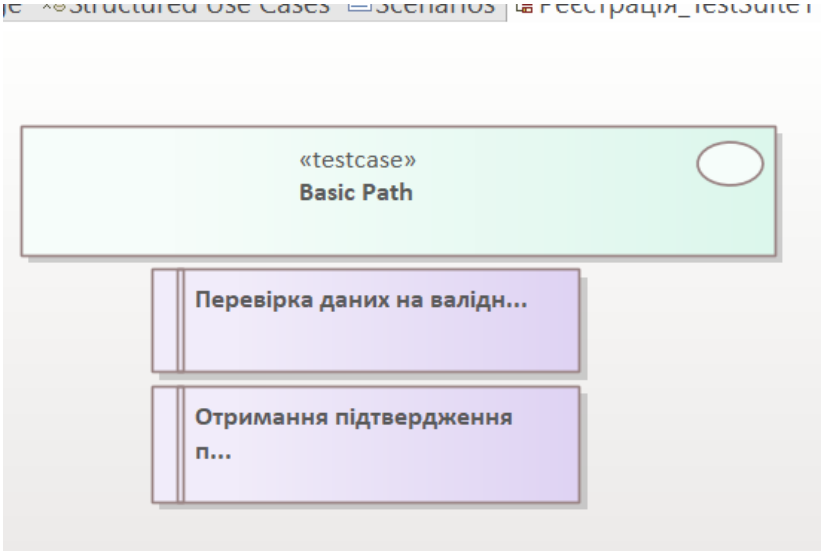


Рисунок 2.27 – Згенерований сценарій з 2 результатами

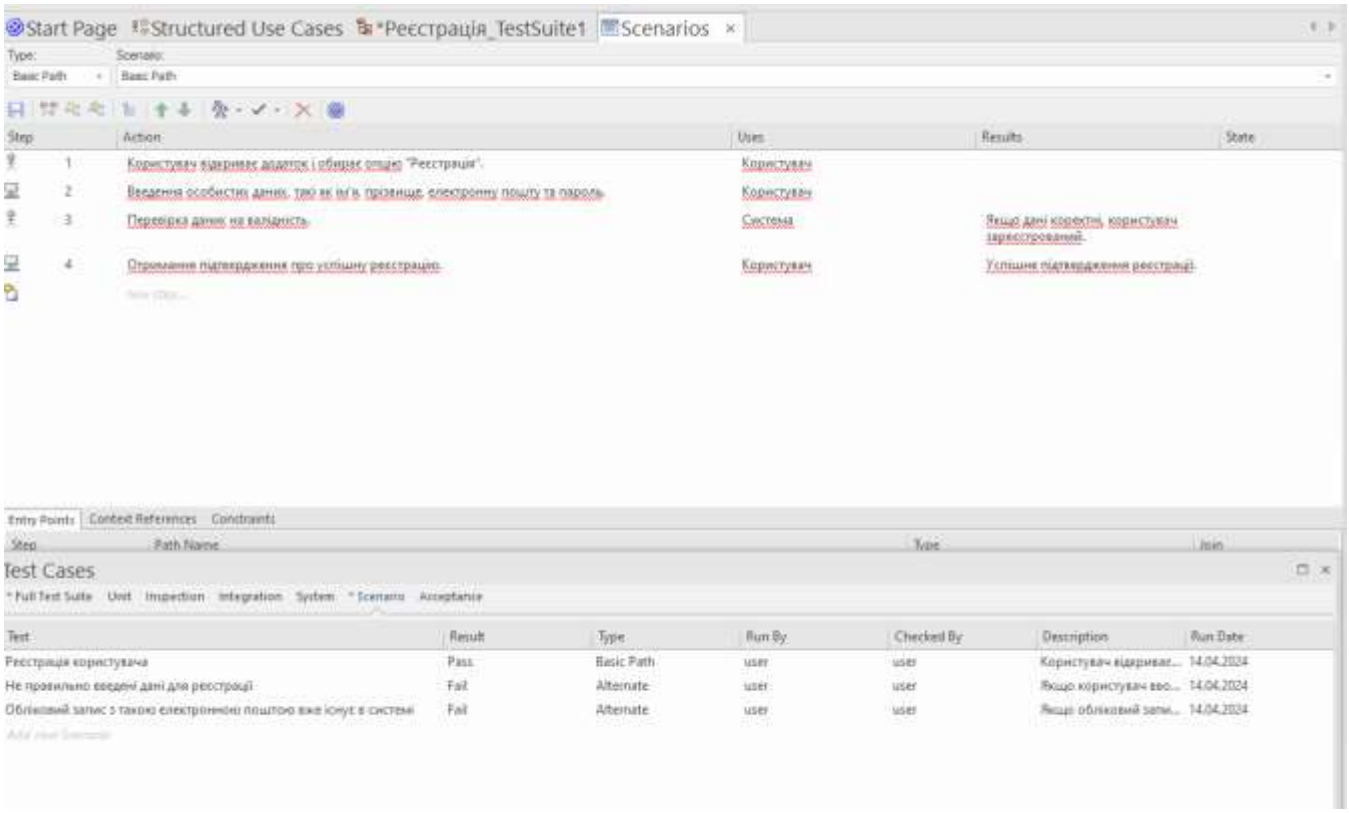


Рисунок 2.28 – Текстовий «нормальний» сценарій

Далі наведено приклади поведінки системи залежних від дій користувача на діаграмах послідовностей (див. рисунки 2.29-2.32).

На діаграмі послідовності для готелю продемонстровано, як об’єкти співпрацюють між собою в системі бронювання номерів готелю(див. рисунок 2.29).

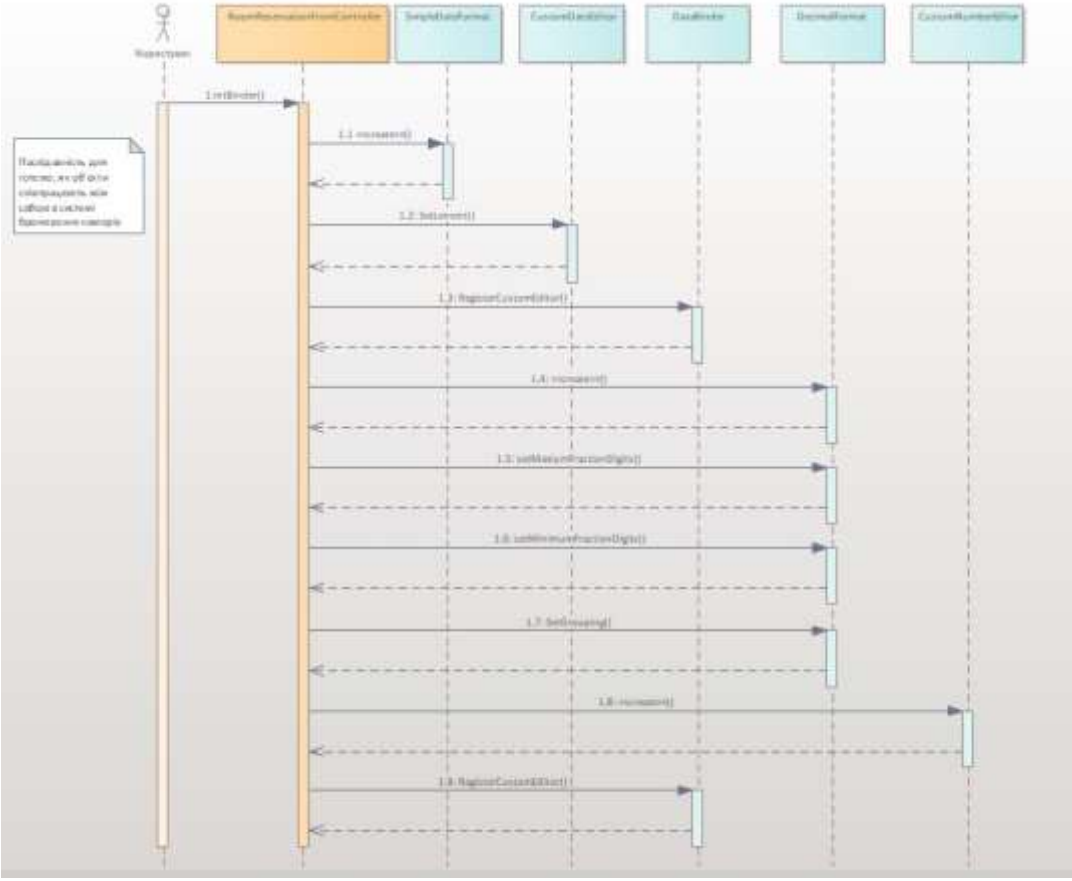


Рисунок 2.29 – Послідовність для готелю, як об’єкти співпрацюють між собою в системі бронювання номерів

Вона включає такі елементи:

По-перше, актори – це зовнішні сутності, які взаємодіють з системою. На діаграмі є два актори: Клієнт та система бронювання.

Кроки діаграми:

1. Клієнт вводить дати заїзду та виїзду та надсилає запит на пошук доступних номерів.
2. Система бронювання створює об’єкт готель та запитує його про доступні номери за вказаними датами.

3. Готель створює об’єкт номер та перевіряє його статус. Якщо номер доступний, він повертає його ціну та інформацію до системи бронювання.
4. Система бронювання повертає список доступних номерів до клієнта.
5. Клієнт обирає номер та надсилає запит на бронювання.
6. Система бронювання створює об’єкт бронювання та запитує його про деталі бронювання, такі як ім’я клієнта, кількість осіб, тип оплати тощо.
7. Бронювання повертає деталі бронювання до системи бронювання.
8. Система бронювання повертає деталі бронювання до клієнта.
9. Клієнт підтверджує бронювання та надсилає запит на оплату.
10. Система бронювання створює об’єкт платіж та запитує його про статус оплати.
11. Платіж обробляє оплату та повертає статус оплати до системи бронювання.
12. Система бронювання повертає статус оплати до клієнта.
13. Клієнт отримує підтвердження оплати та бронювання.

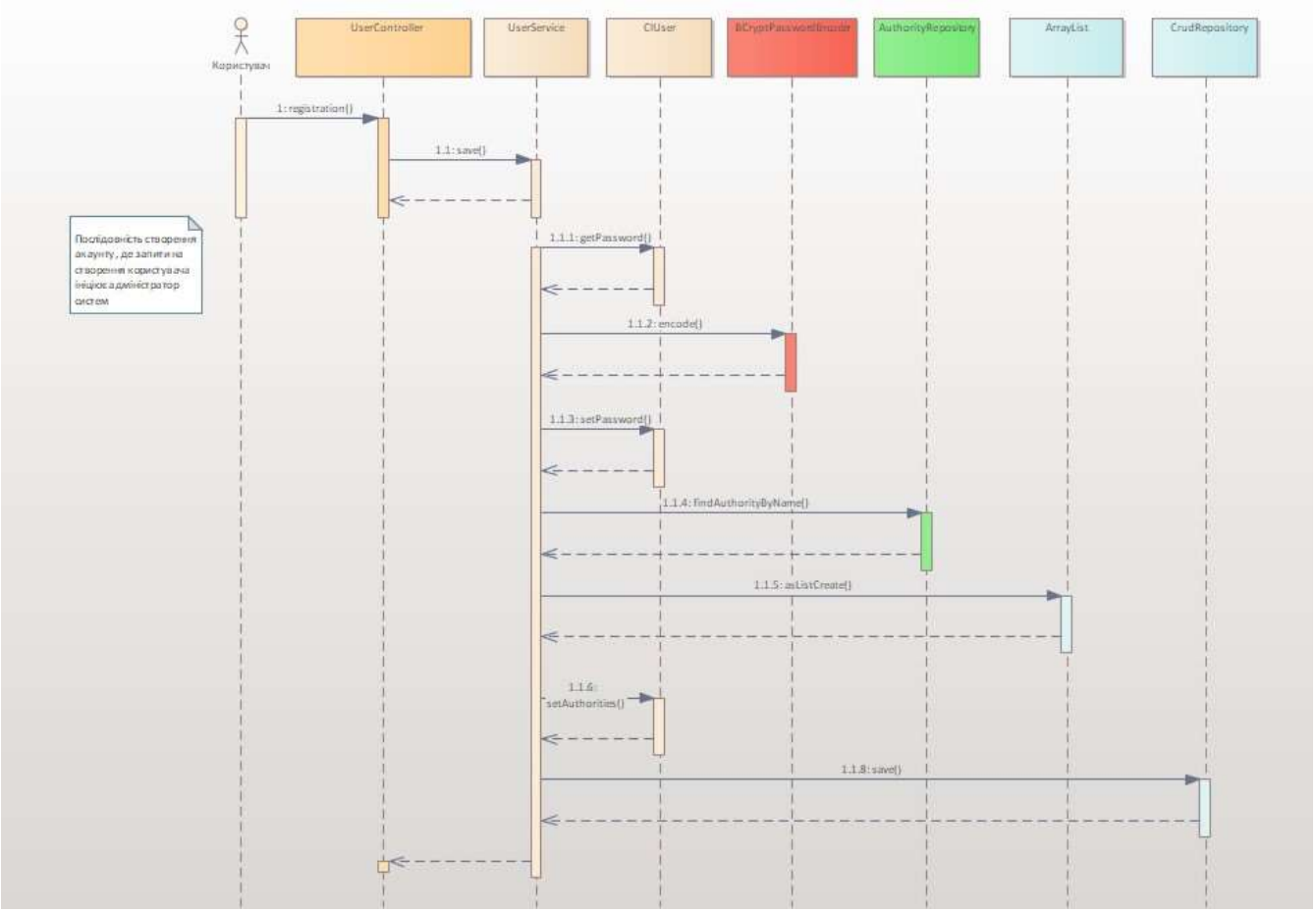
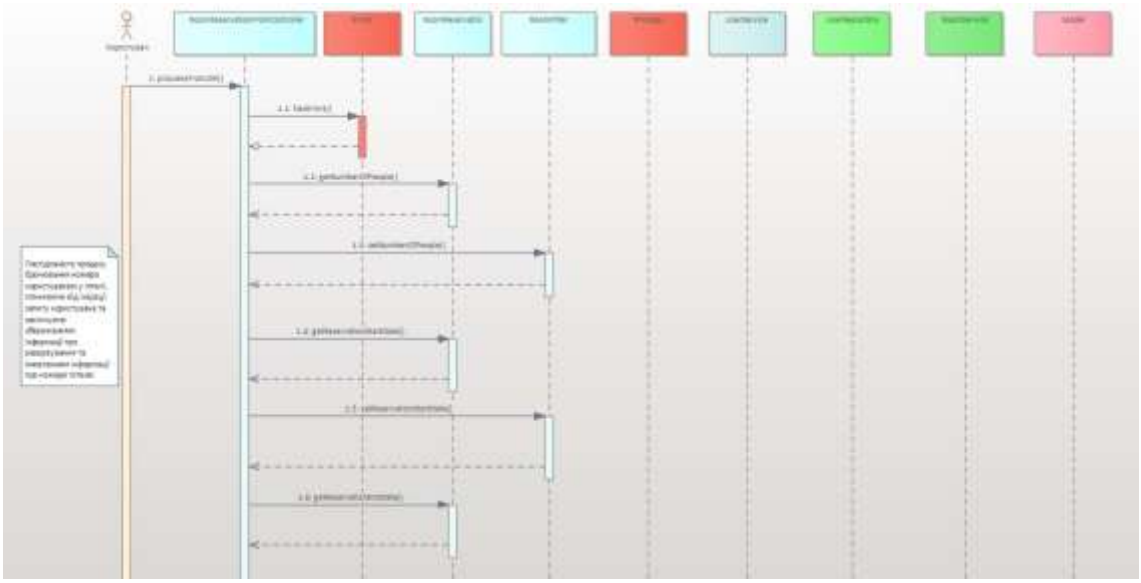
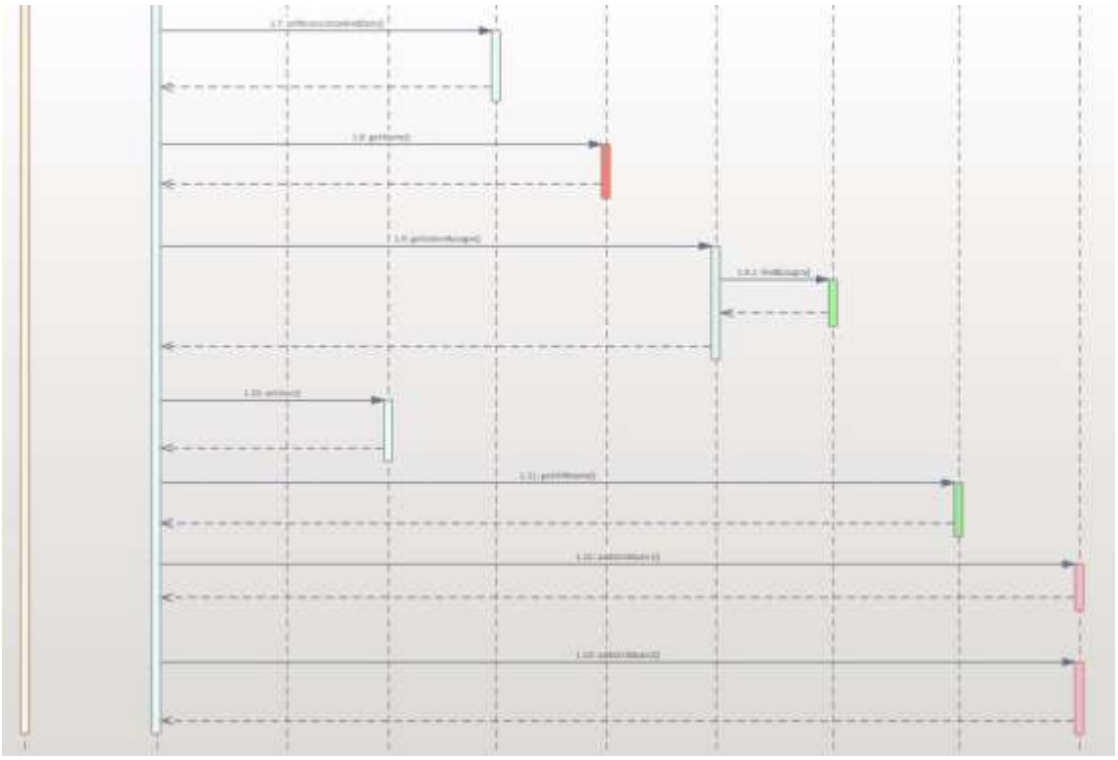


Рисунок 2.30 – Послідовність створення акаунту, де запити на створення користувача ініціює адміністратор систем

На вищенаведеній діаграмі послідовності для готелю продемонстровано взаємодію між адміністратором, контролером користувача, службою користувача, кодувальником (шифруванням) паролів BCrypt, репозиторієм повноважень, масивами, списком масивів та репозиторієм Crud (див. рисунок 2.30).

Наступна діаграма послідовності показує взаємодію між користувачем та різними компонентами системи у послідовному порядку (див. рисунки 2.31-2.32).





Рисунки 2.31-2.32 – Послідовність ілюструє взаємодію користувача з системою при бронюванні номеру в готелі

Основні кроки діаграми послідовності на рисунках 2.31-2.32 включають створення бронювання номеру в готелі.

Основні компоненти та їх взаємодія:

Користувач (User), який хоче забронювати номер в готелі. Наступна дія це викликається RoomReservationFormController для відображення форми бронювання.

RoomReservationFormController – відповідає за взаємодію з користувачем та ініціює подальші кроки бронювання. Отримується повідомлення від користувача (showFormUI).

Викликається RoomReservationService для отримання доступних номерів. Далі, OptionalRoomReservationService компонент надає сервіс для отримання доступних номерів та обробки бронювання. Отримує запит від RoomReservationFormController (getAvailableRooms).

Після цього, викликається RoomReservation для створення бронювання. Цей компонент представляє бронювання номеру та взаємодіє з базою даних. Отримується запит на створення бронювання та взаємодіє з User та Model.

Компоненти User та Model представляють користувача та модель даних в системі, взаємодіють з RoomReservation для збереження інформації про бронювання та користувача.

Наступна діаграма послідовності більш розширена, якщо брати попередню, вона також відображає взаємодію між різними компонентами системи. Основні компоненти включають актора (користувача, клієнта), контролер резервування номерів, резервування номерів, контролер номерів, сервіс користувачів, репозиторій користувачів та модель номерів (див. рисунок 2.33).

Актор – користувач (клієнт), який хоче забронювати номер у готелі. Контролер резервування номерів відповідає за обробку запитів на резервування номерів від користувача. Резервування номерів створює та зберігає об'єкти резервування номерів на основі параметрів, отриманих від контролера резервування номерів. Далі, контролер номерів, який управляє інформацією про номери у готелі, викликаючи методи моделі номерів. Сервіс користувачів, який обробляє дані користувачів, викликаючи методи репозиторію користувачів для зберігання та отримання інформації, а вже репозиторій користувачів зберігає дані користувачів у базі даних. Модель номерів зберігає дані про номери готелю та взаємодіє з контролером номерів.

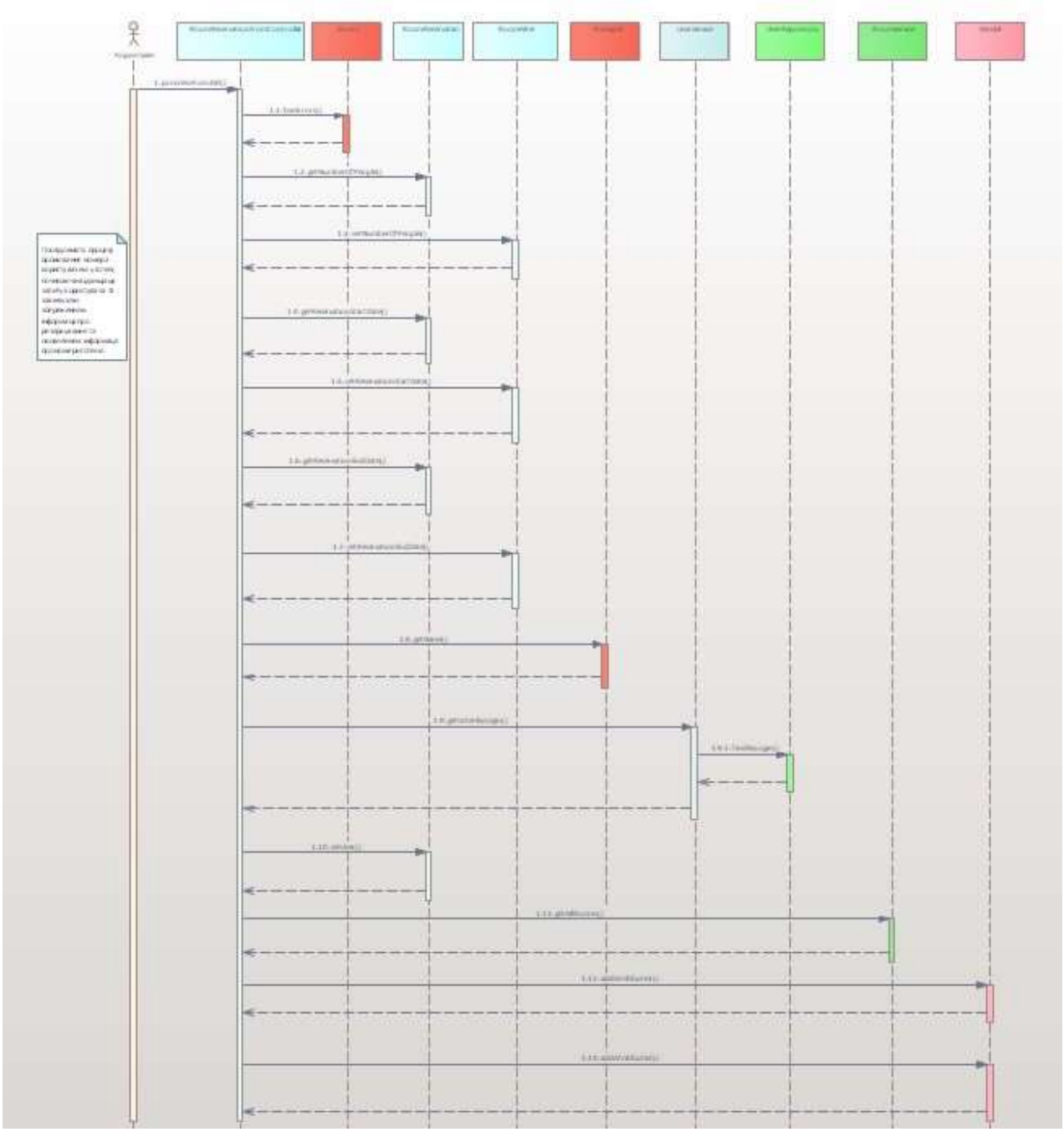


Рисунок 2.33 – Послідовність процесу бронювання номера користувачем у готелі, починаючи від ініціації запиту користувача та закінчуючи збереженням інформації про резервування та оновленням інформації про номери готелю

Вищенаведена діаграма послідовності (див. рисунок 2.33) описує повний процес бронювання номера користувачем у готелі, починаючи від ініціації запиту користувача та закінчуючи збереженням інформації про резервування та оновленням інформації про номери готелю.

Наступна діаграма послідовності описує процес взаємодії користувача з системою для перегляду свого списку бронювань номерів у готелі (див. рисунок 2.34).

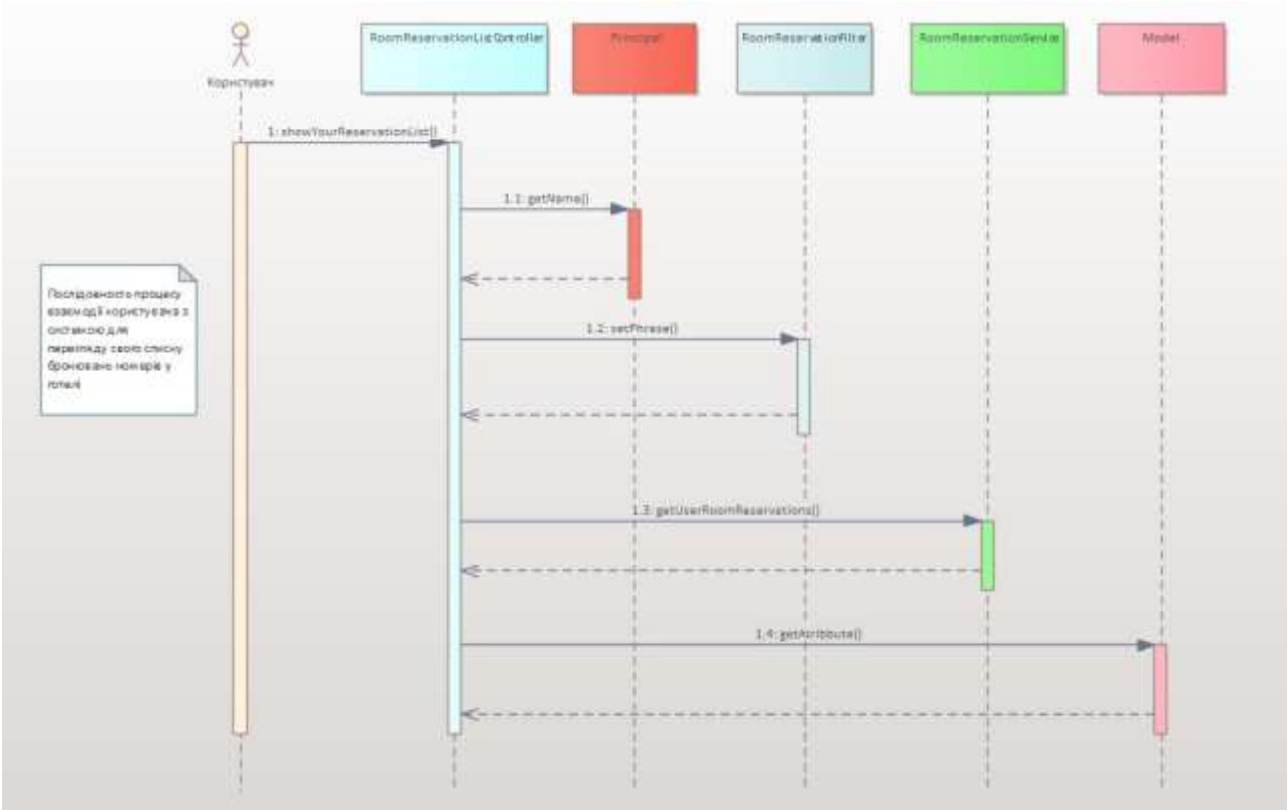


Рисунок 2.34 – Послідовність процесу взаємодії користувача з системою для перегляду свого списку бронювань номерів у готелі

Актор – користувач (клієнт), який хоче переглянути своє бронювання номерів у готелі. Ініціює взаємодію, надсилаючи повідомлення "showYourReservation" до контролера списку бронювання номерів.

Контролер списку бронювання номерів відповідає за обробку запиту користувача та виконання бізнес-логіки. Взаємодіє з принципалом для отримання імені користувача, встановлює фразу для фільтрації бронювань номерів, отримує та обробляє список бронювань через сервіс бронювання номерів та додає атрибут roomReservations до моделі. Принципал зберігає інформацію про поточного користувача. Відповідає на запит контролера списку бронювання номерів, повертаючи ім'я користувача.

Фільтр бронювання номерів фільтрує бронювання номерів за певною фразою. Отримує фразу від контролера списку бронювання номерів. Далі, сервіс бронювання номерів надає доступ до даних про бронювання номерів. Отримує фразу від фільтра бронювання номерів та повертає список бронювань номерів. Після цього модель зберігає дані, які будуть відображені на веб-сторінці. Отримує атрибут roomReservations від контролера списку бронювання номерів.

2.6.1Моделювання структури системи

Головним архітектурним підходом, обраним для розробки системи, є трьохрівнева архітектура, яка базується на моделі MVC (Model-View-Controller), що дозволяє розбити систему на три основні рівні: рівень представлення (інтерфейс користувача), рівень бізнес-логіки, та рівень доступу до даних. Цей підхід забезпечує чітке відокремлення логіки від інтерфейсу користувача та даних, сприяючи більшій організованості, гнучкості та легкості управління та розширення системи.[37]

У межах архітектури MVC також використовуються класи, що представляють компоненти моделі, вигляду та контролера.[37] Це дозволяє розробити баланс між зручністю користувача, ефективністю обробки даних та гнучкістю управління бізнес-процесами. Така структура системи не тільки відповідає сучасним технічним стандартам, але й забезпечує масштабованість та адаптивність до змінних потреб готельного бізнесу.

Також, використовується клієнт-серверна архітектура, де клієнтські додатки взаємодіють з сервером.[38] Розглядаються компоненти, які забезпечують взаємодію з користувачем (клієнтська сторона) та сервер, який обробляє та зберігає дані. На далі, описується структура баз даних, включаючи таблиці для зберігання даних про клієнтів, номери кімнат, бронювання, операції оплати та інші важливі сутності. Визначається взаємозв'язок між цими таблицями та розробляються оптимальні схеми для ефективного зберігання та отримання даних.

Розглядаються окремі функціональні модулі, такі як модуль бронювання, модуль обліку оплат, модуль управління номерами, модуль адміністрування тощо. Для кожного модуля визначаються функції та взаємодія з іншими компонентами системи. Визначаються механізми безпеки для захисту конфіденційності даних, контролю доступу користувачів до різних функціональних модулів та застосування шифрування для зберігання конфіденційної інформації.

ІС повинна взаємодіяти з іншими сервісами чи системами (платіжними шлюзами, системами бронювання), розробляється стратегія інтеграції та описується протокол комунікації.

Інтерфейси користувача відіграють важливу роль у взаємодії між користувачами та інформаційною системою.[39] Під час розробки цих інтерфейсів враховується широкий спектр платформ, на яких вони будуть використовуватися, включаючи ІС та адміністративні панелі. Основні моменти, які визначаються під час розробки інтерфейсів, включають дизайн, функціональність та зручність використання. Дизайн інтерфейсу має забезпечувати візуальну привабливість та зручність використання для користувачів. Зручність використання інтерфейсу полягає в його легкості та інтуїтивній зрозумілості, що дозволяє користувачам ефективно взаємодіяти з системою без зайвих зусиль.

В моєму проєкті використовується трьохрівнева архітектура (MVC-архітектура). У контексті діаграми класів, можна очікувати класи, пов'язані з цими рівнями.

На рисунку 2.35 продемонстровано діаграму класів, яка відображає взаємозв'язки та залежності між різними елементами системи.

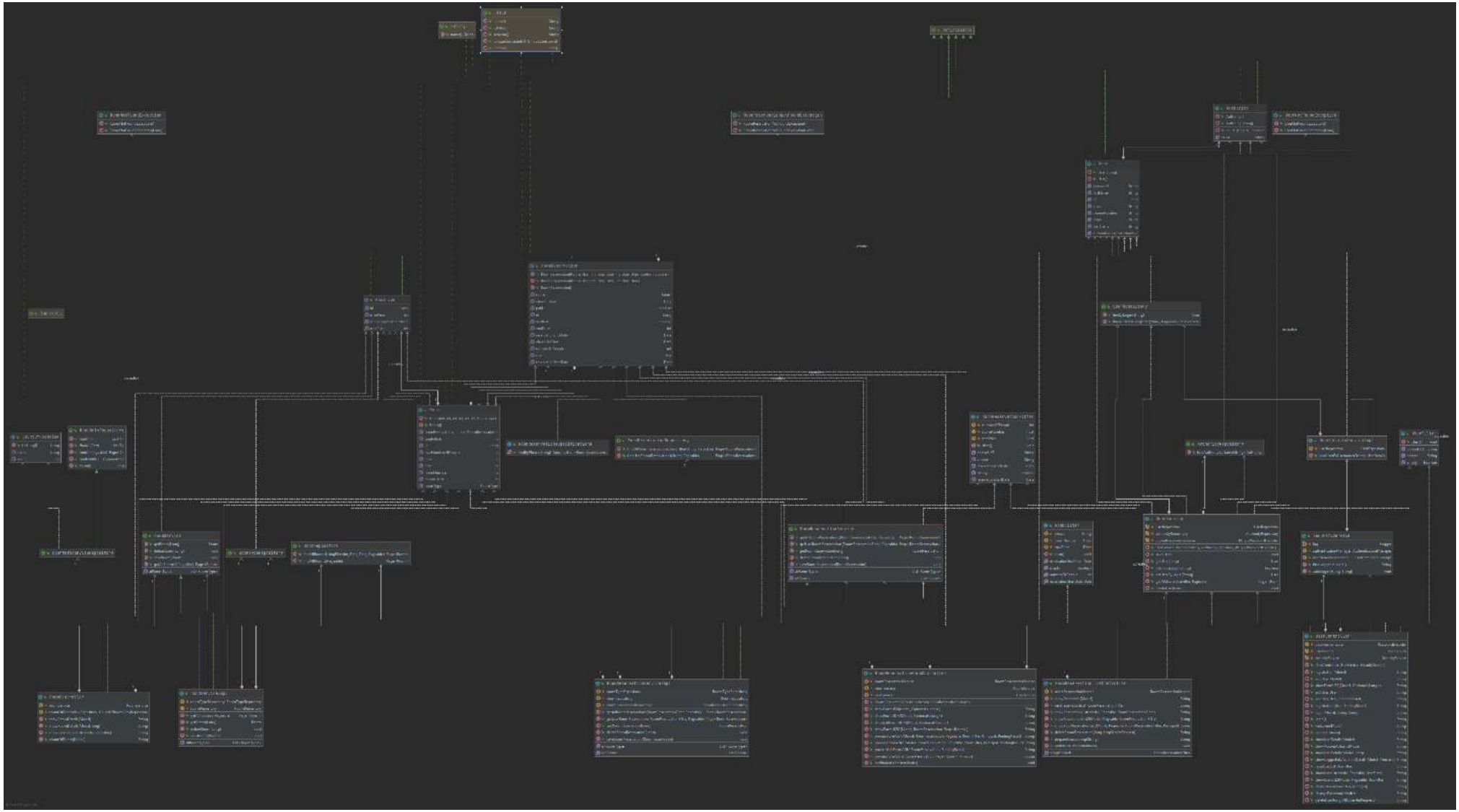


Рисунок 2.35 – Повна діаграма класів згенерована в intellij idea (MVC архітектура)

Фрагменти коду, що було створено для реалізації бекенду системи, наведені у «Додатку Б».

Аналізуючи класи за архітектурним підходом, можна виділити:

Сутності (Model):

Authority – клас, що визначає рівень доступу користувача. Він пов'язаний із системою прав доступу;

CountOfRoomView – представляє вид на кількість доступних номерів. Пов'язаний з відображенням статистики готельних номерів;

Room – представлення об'єкта номеру готелю з основними характеристиками. Відображає структуру номеру;

RoomReservation – відображає бронювання готельного номера. Містить інформацію про дати, користувача та інші деталі;

RoomType – представляє тип номеру готелю. Містить інформацію про категорію номеру;

User – представляє користувача системи. Містить основні дані про користувача.

Класи доступу до даних (Data Access):

ViewRooms – пов'язаний з класом, який відображає статистику кількості номерів;

CountOfRoomViewRepository – клас для доступу до даних про перегляд кількості номерів;

ReadOnlyRepository – загальний клас для доступу до даних у режимі лише читання;

AuthorityRepository – клас для доступу до даних про рівні доступу користувачів;

RoomRepository – клас для доступу до даних про готельні номери;

RoomReservationRepository – клас для доступу до даних про бронювання готельних номерів;

RoomTypeRepository – клас для доступу до даних про типи готельних номерів;

UserRepository – клас для доступу до даних про користувачів.

Класи доступу до даних сервіси:

RoomReservationService – сервіс для обробки операцій з бронюванням готельних номерів;

RoomReservationServiceImpl – реалізація сервісу для бронювання готельних номерів;

RoomService – сервіс для обробки операцій з готельними номерами;

RoomServiceImpl – реалізація сервісу для роботи з готельними номерами;

SecurityService – сервіс для роботи з системою безпеки, із системою автентифікації та авторизації;

UserDetailsServiceImpl – реалізація сервісу для отримання деталей користувача;

UserService – сервіс для операцій з користувачами.

Специфікації:

RoomReservationsSpecifications – клас для визначення специфікацій щодо бронювань готельних номерів.

Контролери:

RoomController – контролер для взаємодії з готельними номерами;

RoomReservationFormController – контролер для обробки форми бронювання готельного номера;

RoomReservationListController – контролер для отримання списку бронювань готельних номерів;

UserController – контролер для взаємодії з користувачами.

Фільтри:

RoomFilter – фільтр для готельних номерів;

RoomReservationFilter – фільтр для бронювань готельних номерів;

UserFilter – фільтр для користувачів.

Exceptions:

RoomNotFoundException – виняток для ситуації, коли номер готелю не знайдено;

RoomReservationNotFoundException – виняток для ситуації, коли бронювання готельного номера не знайдено;

UserNotFoundException – виняток для ситуації, коли користувача не знайдено.

Config:

DatabaseConfiguration – конфігурація для налаштування бази даних;

RepositoriesInitializer – ініціалізатор для налаштування репозиторіїв;

SecurityConfiguration – конфігурація системи безпеки, налаштування автентифікації та авторизації.

Також, наведено діаграму класів ІС «адміністрування готелю» спроєктовану в Enterprise Architec (див. рисунок 2.36). В системі змодельовано повну структуру об'єктів, які в ній присутні, їхні взаємовідносини, властивості та операції.

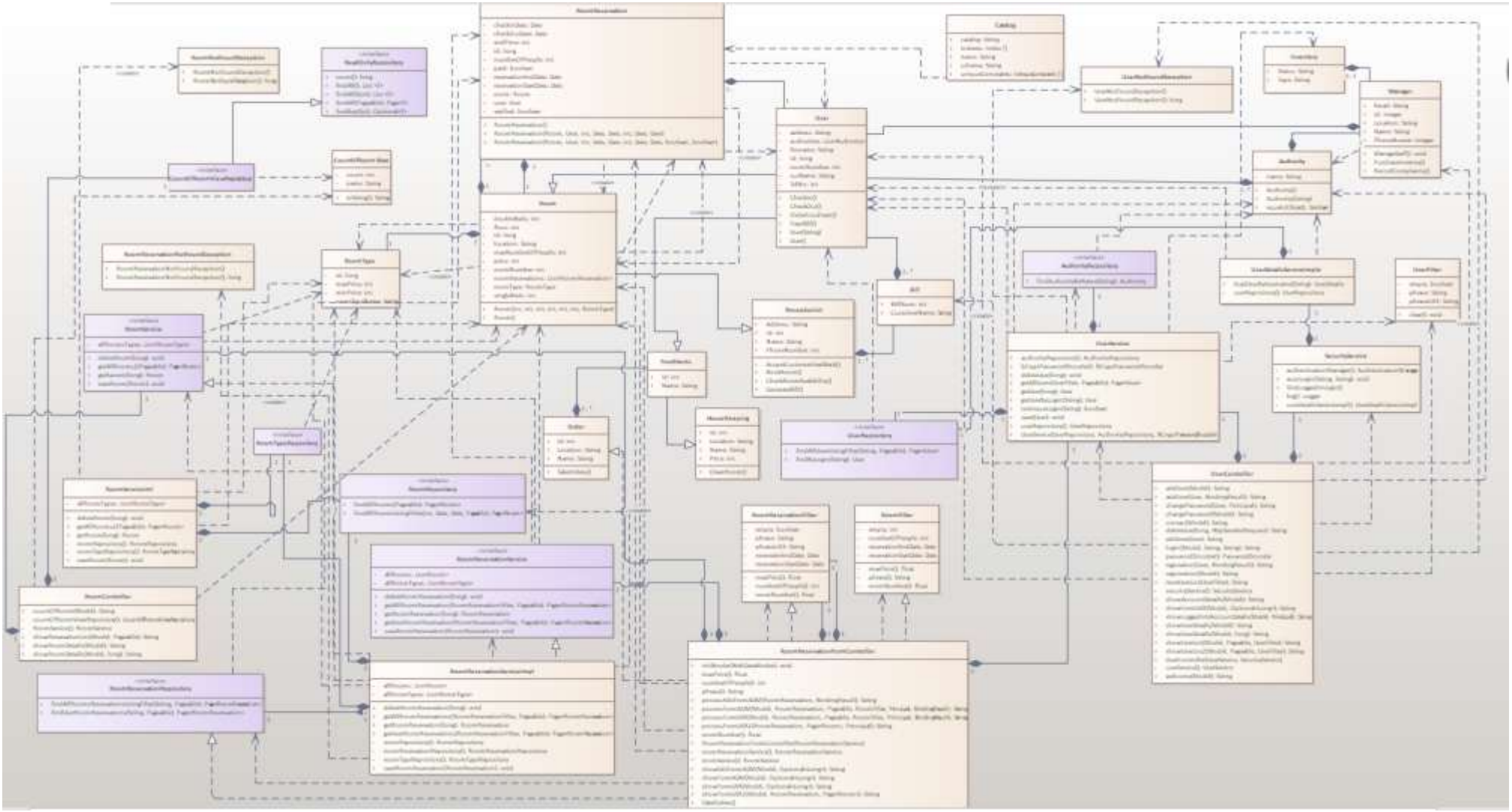


Рисунок 2.36 – Діаграма класів системи

Діаграма визначення блоків для подання внутрішньої структури системи у вигляді ієрархії блоків різних типів відображає компоненти, які складають систему та їхні взаємозв'язки. Для нашої інформаційної системи для адміністрування готелю така діаграма відображатиме різні рівні абстракції та компоненти, що складають систему. На верхньому рівні ієрархії блоків можуть бути такі компоненти, як «Модуль бронювання та обліку номерів», «Модуль обліку клієнтів», «Фінансовий модуль, аналізу та звітності», «Модуль управління персоналом». Кожен з цих блоків має свої власні підкомпоненти та залежності, що відображають структуру та взаємозв'язки між різними частинами системи (див. рисунок 2.37).

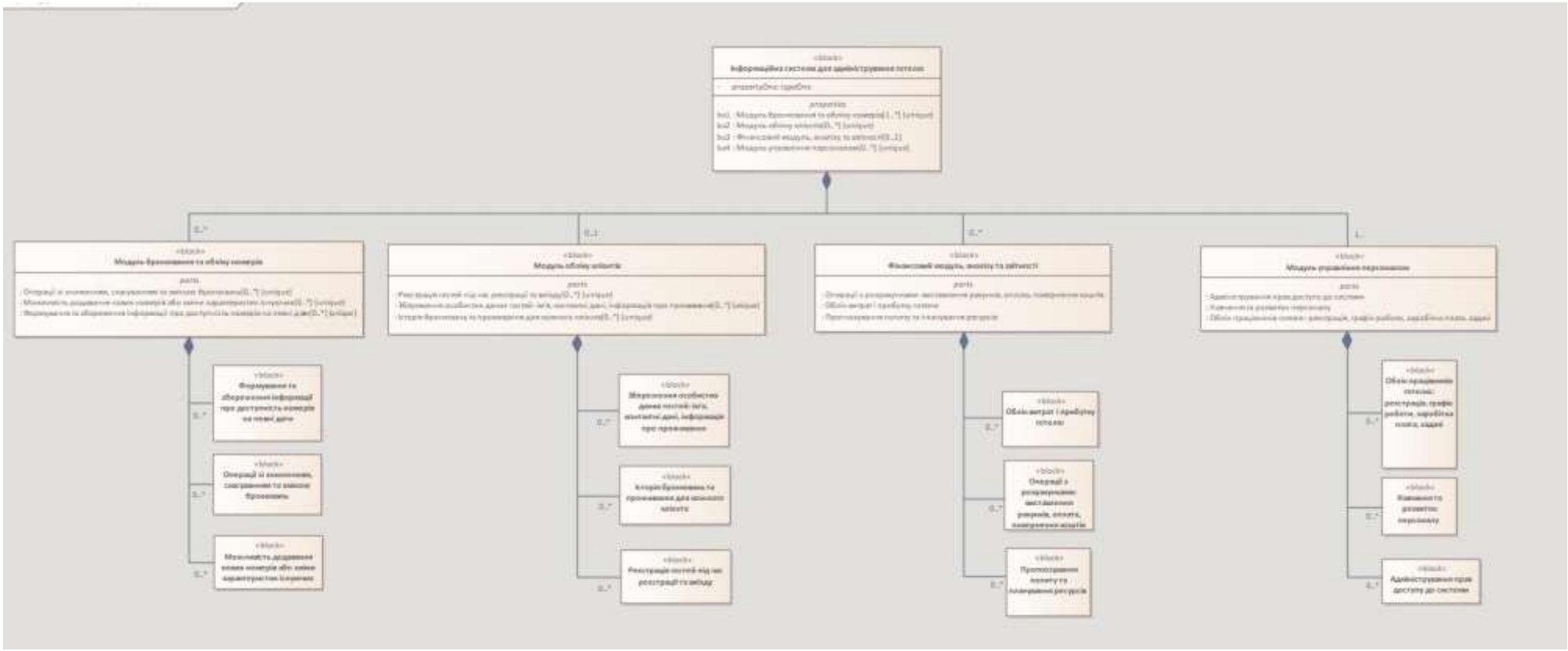


Рисунок 2.37 – Діаграма визначення блоків для подання внутрішньої структури системи (BDD)

На рисунку 2.38 зображено приклад роботи API компонент ІС. На цій діаграмі показано фрагмент залежності компонентів системи, для оформлення бронювання та редагування даних в бд. API приймає запит з вказаними даними і це все надсилається в базу та виводиться на користувальницькому інтерфейсі. Так само під час бронювання. Кожен з цих блоків може мати внутрішні компоненти, які відображають конкретні функціональні частини. Зв'язки між цими блоками можуть бути визначені через порти, які відповідають за обмін даними та виклики функцій між різними компонентами.

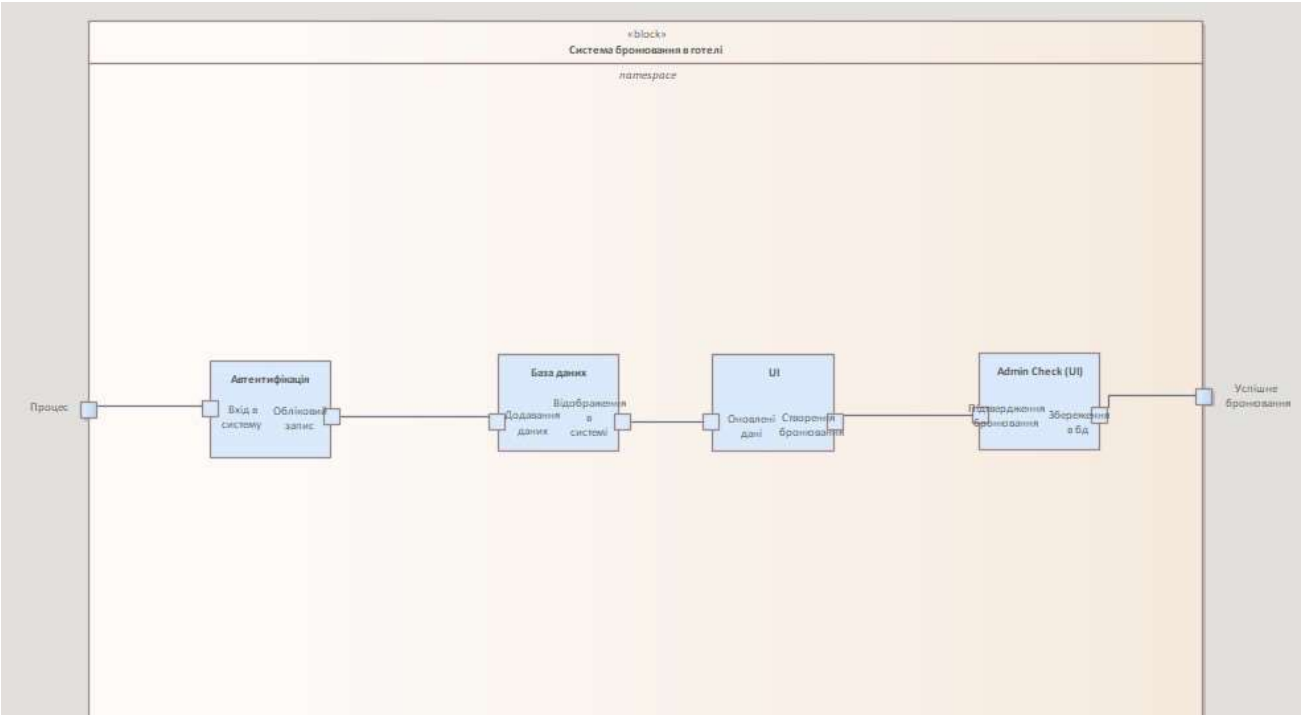


Рисунок 2.38 – Процес оформлення замовлення (IBD)

2.6.2Моделювання користувацького інтерфейсу

На рисунках 2.39-2.41 зображено приклади користувацького інтерфейсу ІС, створені в Enterprise Architect. Наведено можливості використання адмін панелі, процес бронювання номеру та фільтрація і (швидкий) пошук номерів.[40] Ці приклади ілюструють різні аспекти використання системи:

1. Адмін панель:

На рисунку 2.39 зображено інтерфейс адміністративної панелі, який призначений для управління різними параметрами та функціями системи з боку адміністратора. Цей інтерфейс містить можливості налаштування системи, керування користувачами, перегляд статистики та управління бронюваннями.

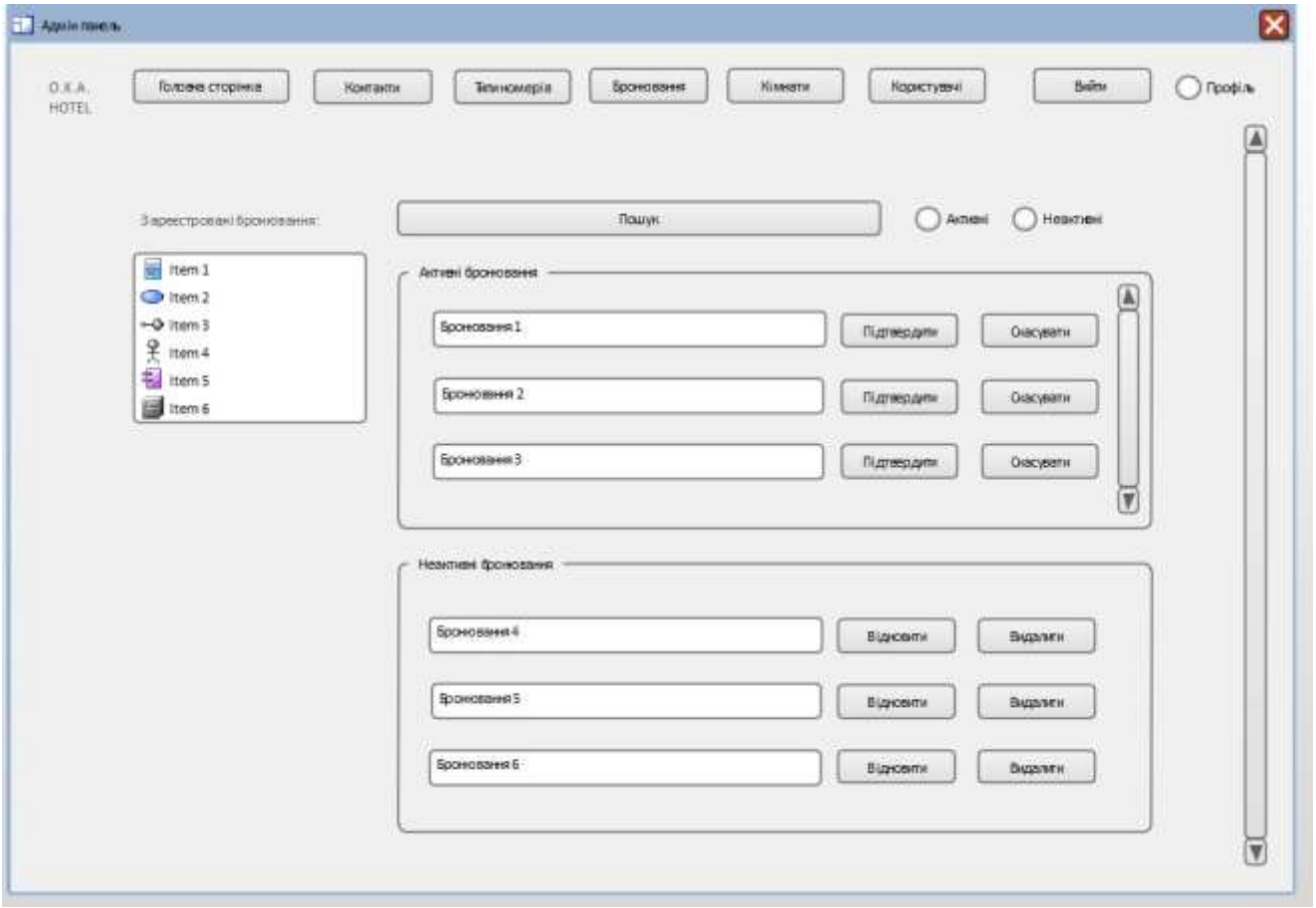


Рисунок 2.39 – Інтерфейс сторінки адмін панелі

2. Процес бронювання номеру:

На рисунку 2.40 продемонстровано інтерфейс процесу бронювання номеру готелю. Це включає форму для заповнення даних про бронювання, вибір дат та кількості номерів, обробку оплати та ін.

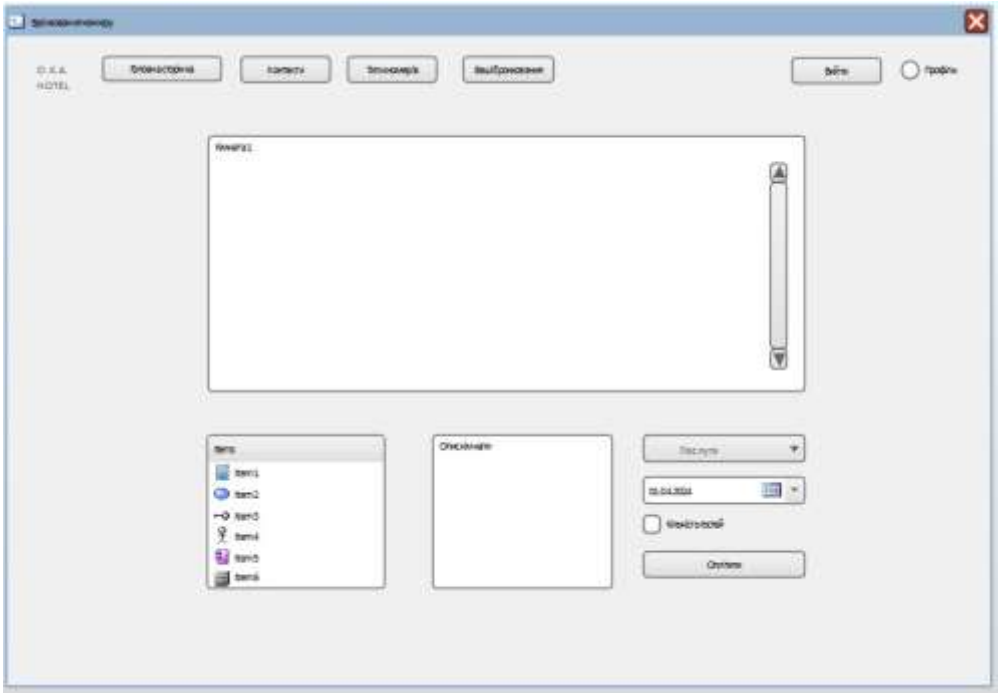


Рисунок 2.40 – Інтерфейс сторінки бронювання номеру готелю

3. Фільтрація та пошук номерів:

На рисунку 2.41 продемонстровано функціонал фільтрації та пошуку номерів готелю. Цей інтерфейс дозволяє користувачам швидко знаходити потрібні номери за різними критеріями, такими як ціна, тип номеру, доступність та ін.

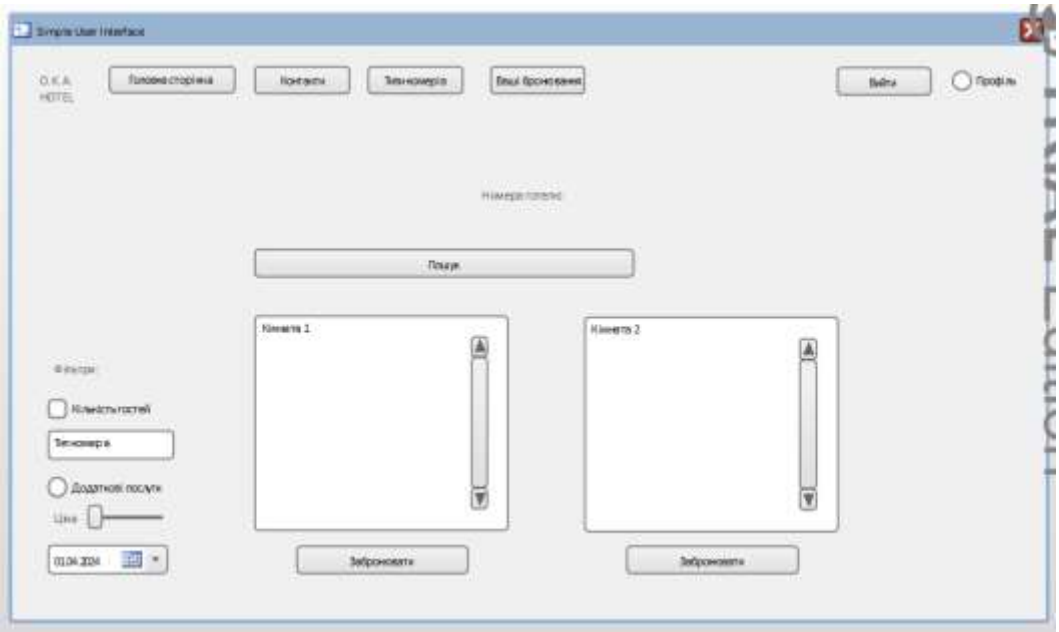


Рисунок 2.41 – Інтерфейс сторінки з функціоналом фільтрації та пошуку номерів готелю

2.6.3Прототипування користувацьких інтерфейсів

Для демонстрації користувацьких інтерфейсів було створено прототип інформаційної системи у векторному онлайн-сервісі Figma.[41] На рисунку 2.42 представлено клікабельний прототип. Було сворено 2 прототипи, один з яких для клієнтської частини, а інший для адміністраторської частини.

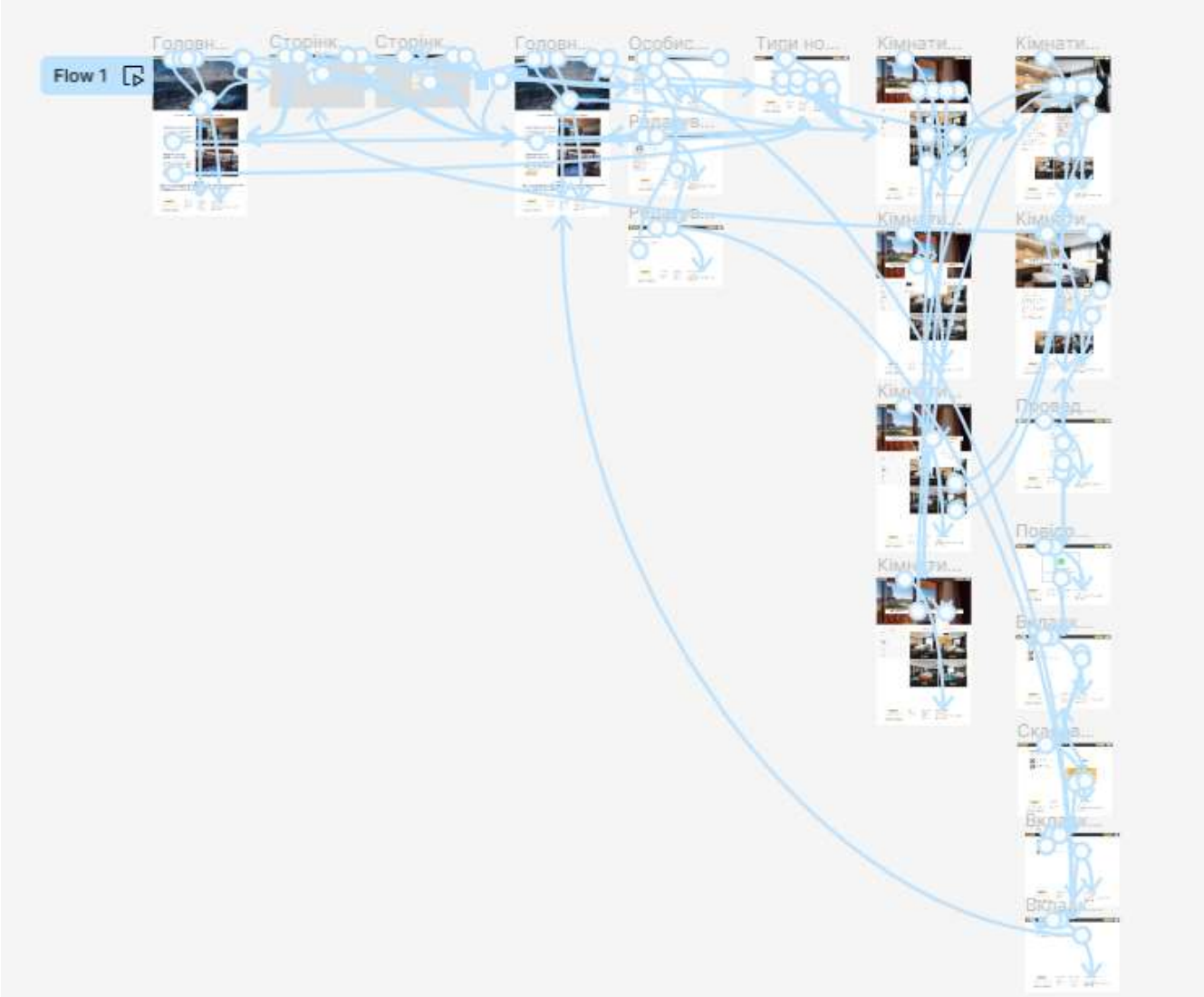


Рисунок 2.42 – Клікабельний прототип інформаційної системи «Адміністрування готелю» розроблений у Figma

Сторінки прототипу системи наведено у «Додатку В».

Інформаційна система для адміністрування готелю має широкий функціонал, представлений різноманітними сторінками та опціями, які дозволяють користувачам зручно та ефективно взаємодіяти з системою. Головні

сторінки готелю призначені для різних типів користувачів: від неавторизованих відвідувачів, які отримують базову інформацію про готель та його послуги, до авторизованих клієнтів та адміністраторів системи. Користувачі мають можливість авторизуватися через Google акаунт або зареєструвати новий обліковий запис в системі. Після входу в особистий кабінет користувач може переглянути та змінити свої особисті дані, переглянути історію дій та відгуки, які він залишав. З іншого боку, адміністратор системи має розширені можливості управління користувачами та номерами готелю через відповідний особистий кабінет.

Після вибору необхідних номерів користувач переходить на сторінку пошуку кімнат, де він може використовувати різноманітні фільтри та параметри для точного підбору пропозицій. На цій сторінці доступні випадаючі списки фільтрації, де користувач може вибрати параметри в'їзду, виїзду та кількості гостей, щоб знайти найбільш відповідні номери. Після обрання параметрів і натискання кнопки "Знайти номер", користувач переходить на сторінку перегляду конкретного номеру. Тут він може побачити всю необхідну інформацію про кімнату та послуги, а також здійснити вибір із доступних опцій або перейти до іншої підходящої кімнати.

Після вибору необхідного номера користувач переходить на сторінку оформлення бронювання. Тут він повинен погодитися з умовами бронювання та підтвердити замовлення, оплативши 50% від загальної вартості. Після натискання кнопки "Оплатити" користувач переходить на сторінку проведення оплати, де йому пропонується вибрати платіжну систему та здійснити оплату.

Після успішної оплати користувач отримує повідомлення про успішну транзакцію, а також квитанцію про оплату на вказану ним електронну адресу. Ці деталі, а також статус та іншу інформацію про бронювання можна переглянути в особистому кабінеті користувача в розділі "Мої бронювання". Також на сторінці доступні опції скасування бронювання та перегляду деталей скасованих замовлень.

Також, був проведений аналіз функціонала системи та узгодження з вимогами (див. табл. 2.3).

Аналіз функціонала та узгодження з вимогами Таблиця 2.3

Priority	User Story	Результат
1	Користувач системи (клієнт), повинен мати можливість ввійти в систему, використовуючи свій обліковий запис.	Дизайн екрану входу та реєстрації. Клікабельні елементи для введення логіну та паролю та можливість входу використовуючи Google акаунт.
2	Користувач системи (клієнт), повинен мати можливість зареєструвати новий обліковий запис.	Дизайн екрану реєстрації з полями для введення обов'язкових даних, які будуть потім відображатися в особистому кабінеті користувача. Також, реєстрація за допомогою акаунту Google.
3	Користувач системи (клієнт) повинен бачити несильно навантажену, привабливу та інформативну головну сторінку, яка відображає основні аспекти та послуги системи.	Дизайн головної сторінки, який забезпечує легкий доступ до ключових розділів сайту, можливість переглянути особистий кабінет і передає основну ідею проєкту.
4	Адміністратор готелю, повинен мати можливість додавати, видаляти та редагувати інформацію про користувачів системи.	Дизайн екрану для керування користувачами та їх особистими кабінетами. Додається, видаляється та редагується інформація.
5	Адміністратор готелю, повинен мати можливість переглядати та керувати бронюваннями номерів (можливість редагувати).	Дизайн екрану для перегляду та керування бронюваннями. Клікабельні елементи для перегляду деталей.
6	Користувач системи (клієнт) та адміністратор повинені мати особистий кабінет для зручного керування своїм профілем та операціями.	Дизайн сторінок особистих кабінетів з можливістю перегляду профіля, історії переглядів номерів, та інших основних операцій.
7	Користувач системи (клієнт), повинен мати можливість швидко знаходити необхідні кімнати за допомогою пошуку та фільтрів, мати пошук за основними параметрами (дата в’їзду, дата виїзду, кількість гостей).	Дизайн екранів зі зручним пошуком за параметрами, за типами номерів та використання фільтрів для вибору кімнат.
8	Надати користувачам можливість перегляду контактної інформацію, правил бронювання та іншої інформації.	Дизайн та швидкий доступ до контактів, зв’язку з менеджером через меню опцій та швидкий перехід до правил бронювань.
9	Підвищити візуальну привабливість сайту за допомогою анімації.	Інтеграція анімаційних ефектів на різних сторінках та елементах меню для поліпшення користувацького досвіду.
10	Надати користувачам системи можливість залишати відгуки та оцінювати послуги готелю.	У функціональному особистому кабінеті можна переглядати та додавати відгуки та виставлення рейтингів. Також, створено додаткові вкладки для зворотного зв’язку та бажаним

		способом (гаряча лінія телефону чи електронна пошта).
11	Користувач системи (клієнт), повинен мати можливість легко оплатити замовлення та отримувати квитанції про платежі на пошту.	Створено сторінку з платіжною системою, оплатою замовлення, де можна теж використовувати Google Pay, після успішної оплати буде повідомлення від менеджера на пошту з квитанцією про оплати. Ці деталі, статуси також можна дивитися в особистому кабінеті ваших бронюваннях.
12	Менеджер готелю, повинен мати можливість отримувати платежі та надавати квитанції про оплати на пошту клієнта.	Дизайн екрану для обробки платежів та видачі рахунків. Менеджер може переглядати статуси бронювань, їх оплату і тд., після чого висилає квитанції на пошти клієнтів.

Користувач із сервером зв’язуються за допомогою HTTP-запитів. Серед них є:

- запит на реєстрацію нового користувача;
- запит на авторизацію (отримання ключа авторизації);
- запит на авторизацію за певною роллю;
- запит на отримання особистої інформації користувача (облікові дані);
- запит на редагування облікових даних користувача в особистому кабінеті;
- запит на пошук типів номерів;
- запит на пошук номерів;
- запит на перегляд послуг;
- запит на створення бронювання;
- запит на редагування бронювання;
- запит на видалення старих бронювань;
- запит на скасування бронювання;
- запит на створення нового коментаря;
- запит на створення нового відгуку;
- запит на видалення відгуку;
- запит на додавання вподобайки номеру готелю;

- запит на створення нових типів номерів;
- запит на редагування типів номерів;
- запит на видалення типів номерів;
- запит на створення нових номерів;
- запит на редагування номерів;
- запит на видалення номерів;
- запит на створення нового користувача адміном;
- запит на редагування видалення користувача адміном;
- запит на додавання нової послуги;
- запит на редагування послуги;
- запит на видалення послуги;
- запит для аналітики даних користувача;
- запит для аналітики відгуків номерів;
- запит для аналітики доданих користувачем послуг до бронювання;
- запит для аналітики найпопулярнішого типу номерів;
- запит для аналітики найпопулярніших номерів.

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

Для створення бази даних системи було обрано MySQL.[21]

MySQL є однією з найпопулярніших систем управління базами даних (СУБД) завдяки високій продуктивності, надійності та легкості використання. Порівняно з іншими СУБД, такими як Oracle або Microsoft SQL Server, MySQL пропонує ряд переваг, особливо для проєктів, які вимагають швидкої розробки та гнучкості, що є характерним для багатьох веб-орієнтованих систем.

Перш за все, MySQL оптимізована для високої швидкості роботи та надійного зберігання даних, що робить її ідеальною для інформаційних систем з великою кількістю транзакцій, як системи адміністрування готелю. Її відкритий вихідний код дозволяє налаштовувати і оптимізувати систему під конкретні потреби проєкту.[20]

Коли MySQL використовується у зв'язці з Java та середовищем розробки IntelliJ IDEA, вона надає розробникам могутній інструментарій для створення, тестування та управління базами даних. Інтеграція з Java є безшовною завдяки JDBC API, який дозволяє Java-додаткам взаємодіяти з базою даних для виконання запитів, оновлень та отримання даних.[42]

У рамках інформаційної системи адміністрування готелю приділено велику увагу розробці користувацького інтерфейсу з метою забезпечення ефективності та зручності користування системою. Вхідні форми, як форма реєстрації, форма бронювання номера та форма введення послуг, розроблені таким чином, щоб мінімізувати необхідність введення даних вручну та забезпечити швидке та безпомилкове внесення інформації. Наприклад, форма бронювання може включати датапікери (фільтри) для вибору дат, дропдауни для вибору типу номера та кількості гостей, а також автоматичні перевірки на доступність номерів у вказаний період.

Для реєстрації нових користувачів використовуються форми, що містять поля для введення персональних даних, контактної інформації, а також безпекові запитання та вибір паролю. Це спрощує процес інтеграції користувачів у систему та дозволяє легко відслідковувати їх активність і управління бронюваннями.

Вихідні форми забезпечують адміністраторам та користувачам доступ до важливої інформації через звіти, які можуть бути автоматично генеровані системою. Це включає звітність про статус номерів, аналіз завантаженості готелю, фінансову звітність, а також звіти про задоволеність клієнтів. Завдяки цьому адміністрування готелю стає прозорішим та контрольованим, що сприяє прийняттю обґрунтованих управлінських рішень.

Продумані вхідні та вихідні форми є ключовими для створення ефективної інформаційної системи, яка оптимізує управління готелем та покращує досвід користувачів, забезпечуючи високий рівень обслуговування та управління.

Отриману базу даних наведено на рисунку 3.1.

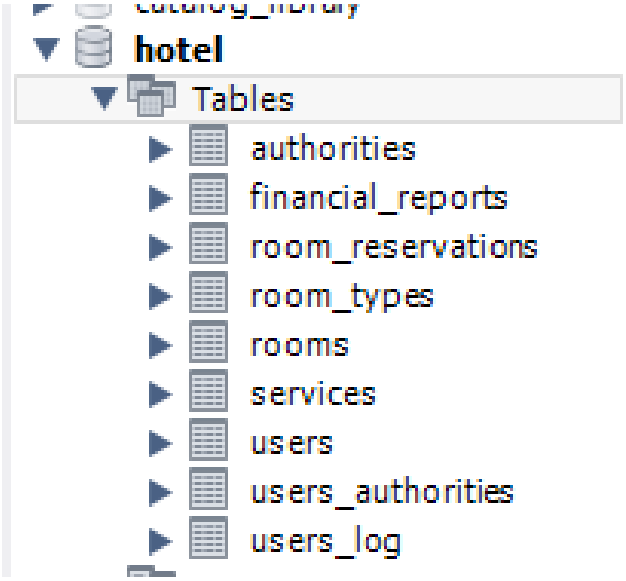


Рисунок 3.1 – База даних побудована в MySQL Workbench

Модель бази даних продемонстрована на рисунку 3.2 в середовищі MySQL Workbench.

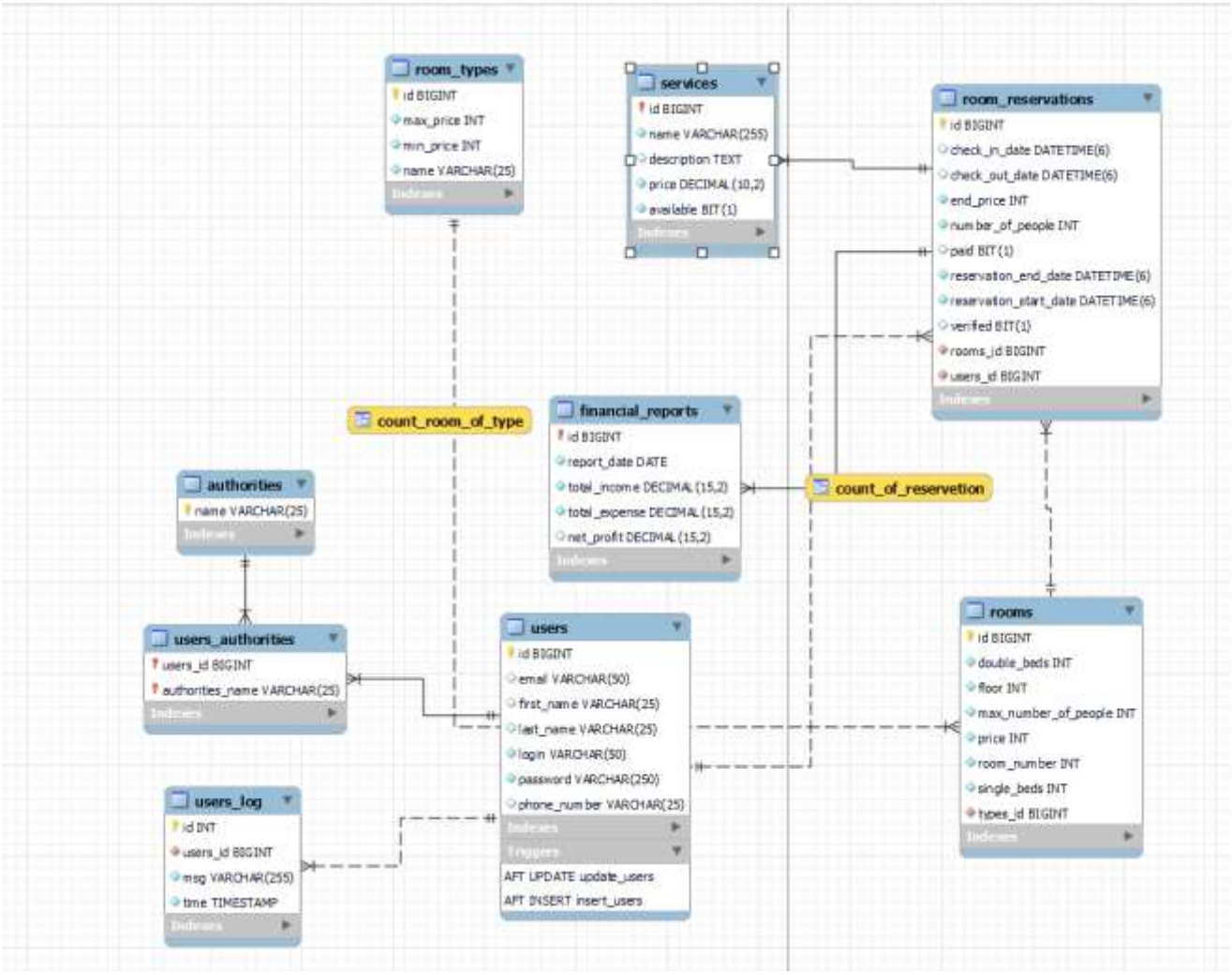


Рисунок 3.2 – EER-діаграма побудована в MySQL Workbench

Деякі з наведених колекцій є вкладеними тобто є частиною інших.

Результат перетворення ER-моделі в конкретну схему бази даних на основі СУБД MySQL та вміст файлу згенерованої моделі даних представлено у «Додатку Г».

Специфікація вхідних та вихідних даних інформаційної системи «Адміністрування готелю» представлена на рисунках 3.3-3.11.

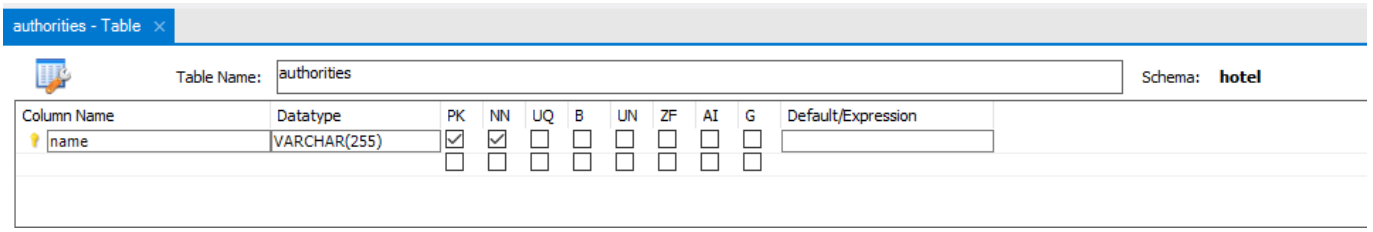


Рисунок 3.3 – Специфікація таблиці «Authorities»

room_reservations - Table

Table Name: room_reservations Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
check_in_date	DATETIME(6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
check_out_date	DATETIME(6)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
end_price	INT	☐	☑	☐	☐	☐	☐	☐	☐	

Рисунок 3.4 – Специфікація таблиці «Room_reservations»

room_types - Table

Table Name: room_types Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
max_price	INT	☐	☑	☐	☐	☐	☐	☐	☐	
min_price	INT	☐	☑	☐	☐	☐	☐	☐	☐	
name	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	

Рисунок 3.5 – Специфікація таблиці «Room_types»

rooms - Table

Table Name: rooms Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
double_beds	INT	☐	☑	☐	☐	☐	☐	☐	☐	
floor	INT	☐	☑	☐	☐	☐	☐	☐	☐	
max_number_of_people	INT	☐	☑	☐	☐	☐	☐	☐	☐	

Рисунок 3.6 – Специфікація таблиці «Rooms»

users - Table

Table Name: users Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	☑	☑	☐	☐	☐	☐	☑	☐	
email	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
first_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
last_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 3.7 – Специфікація таблиці «Users»

users_authorities - Table

Table Name: users_authorities Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
users_id	BIGINT	☑	☑	☐	☐	☐	☐	☐	☐	
authorities_name	VARCHAR(255)	☑	☑	☐	☐	☐	☐	☐	☐	

Рисунок 3.8 – Специфікація таблиці «Users_authorities»

users_log - Table

Table Name: users_log Schema: **hotel**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	☐	☐	☑	☐	☑	☐	
users_id	BIGINT	☐	☑	☐	☐	☐	☐	☐	☐	
msg	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
time	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP

Рисунок 3.9 – Специфікація таблиці «Users_log»

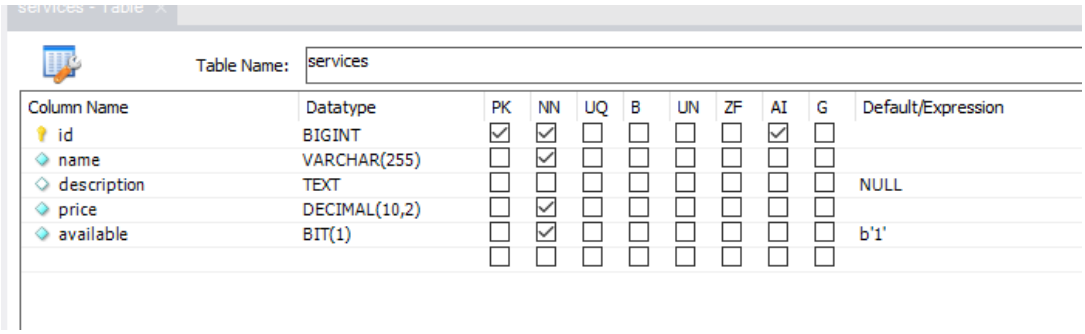


Рисунок 3.10 – Специфікація таблиці «Services»

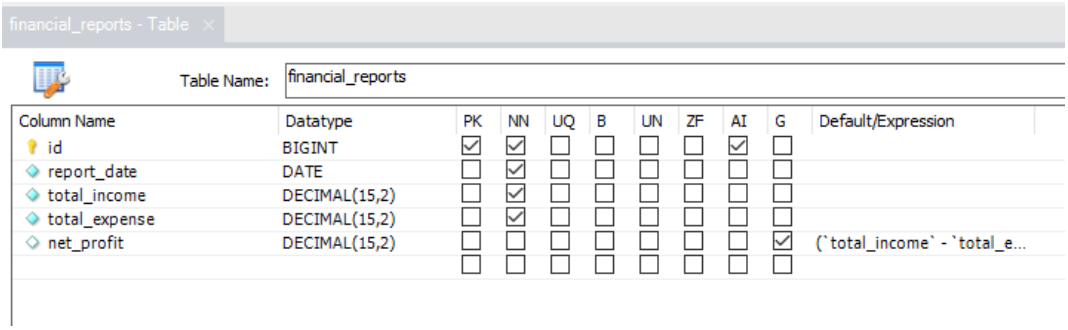


Рисунок 3.11 – Специфікація таблиці «Financial_reports»

Використання MySQL, тригерів, процедур та виглядів забезпечує високу продуктивність, важливу для інформаційної системи з великою кількістю транзакцій, також надає надійність та цілісність даних, зручність використання, спрощуючи доступ до інформації та автоматизуючи деякі процеси.

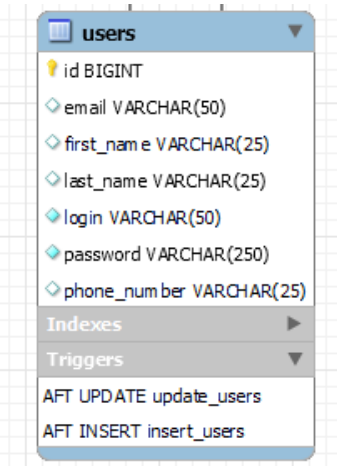


Рисунок 3.12 – Тригери insert_users та update_users в таблиці «Users»

Тригери insert_users та update_users автоматизують процес ведення журналу змін в таблиці users.insert_users спрацьовує після додавання нового користувача, записуючи в таблицю users_log повідомлення "insert" разом з ID нового користувача. update_users діє аналогічно, але запускається при оновленні

даних про користувача, записуючи повідомлення "update" та ID зміненого рядка. Ці тригери спрощують відстеження змін в таблиці users, забезпечуючи хронологічний журнал операцій. Це корисно для аудиту, аналізу активності користувачів та виявлення потенційних проблем.

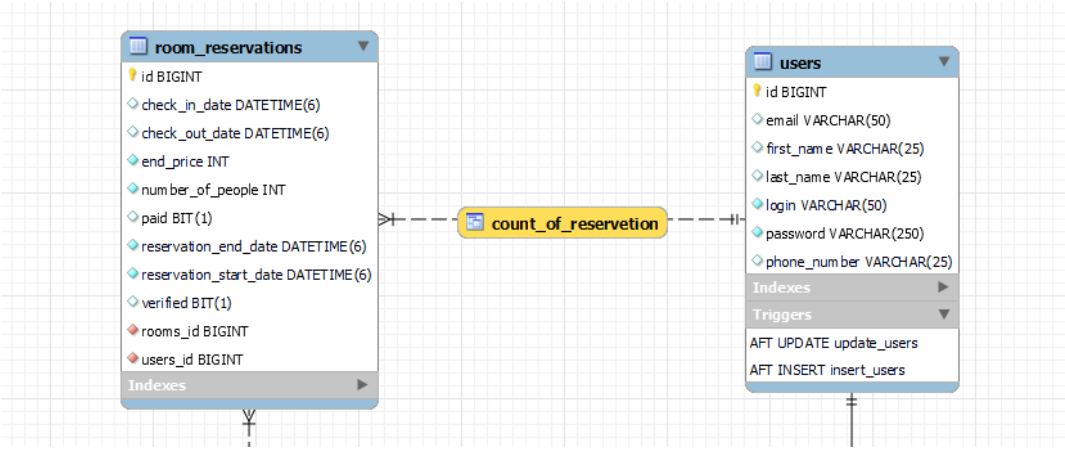


Рисунок 3.13 – View (вигляд) count_of_reservetion, який агрегує дані з таблиць users та room_reservations

Вигляд count_of_reservetion агрегує дані з таблиць users та room_reservations, щоб показати кількість бронювань, зроблених кожним користувачем. Це дозволяє швидко отримати інформацію про активність користувачів та їх популярність.

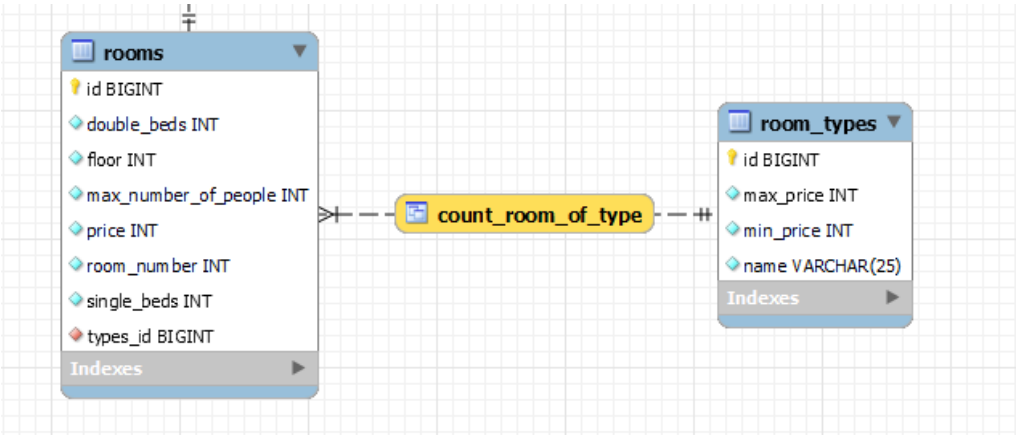


Рисунок 3.14 – View (вигляд) count_room_of_type поєднує дані з room_types та rooms для підрахунку кількості кімнат кожного типу

Вигляд count_room_of_type поєднує дані з room_types та rooms, щоб підрахувати кількість кімнат кожного типу. Це спрощує аналіз номерного фонду готелю та його структури.

Рисунок 3.15 – Процедура get_user_roles_by_first_name

Процедура get_user_roles_by_first_name приймає ім'я користувача як вхідний параметр та повертає список його ролей. Вона збирає дані з таблиць users, users_authorities та authorities, використовуючи ім'я користувача для фільтрації результатів. Ця процедура спрощує отримання інформації про ролі користувачів, уникаючи необхідності писати складні SQL-запити.

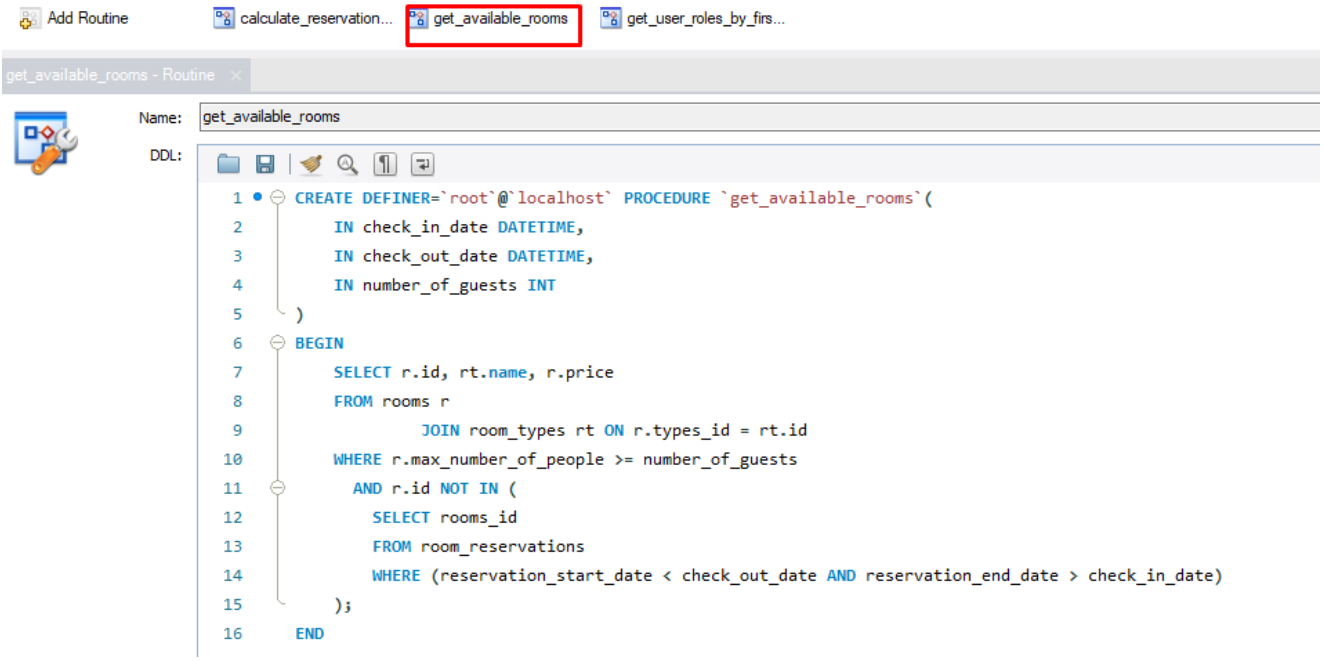


Рисунок 3.16 – Процедура get_available_rooms

Процедура get_available_rooms повертає список доступних номерів на заданий період часу, з урахуванням кількості гостей. Ця процедура спрощує пошук доступних номерів для клієнтів. Вона враховує не тільки дати заїзду та виїзду, але й кількість гостей, забезпечуючи точний результат.

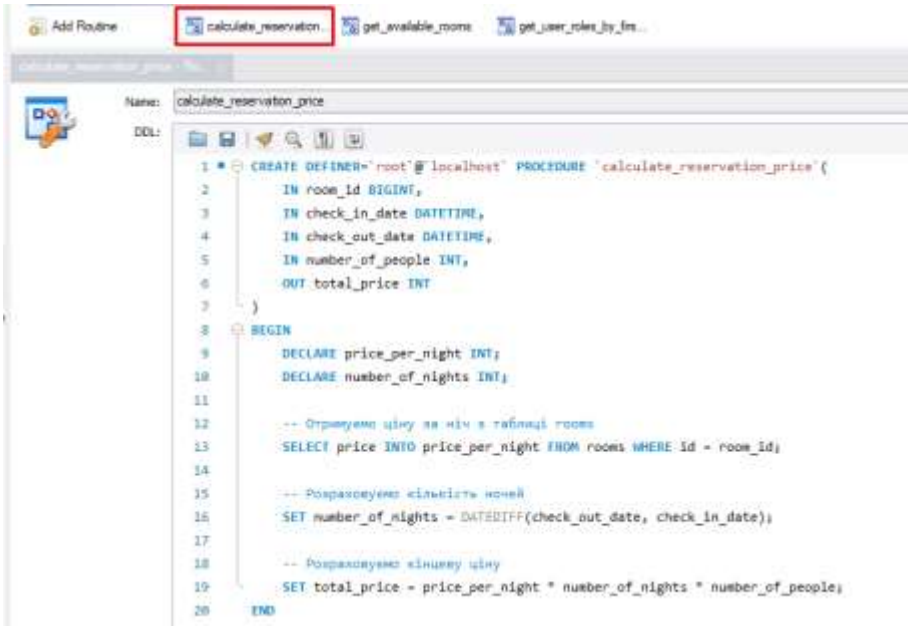


Рисунок 3.17 – Процедура calculate_reservation_price

Процедура calculate_reservation_price розраховує кінцеву вартість бронювання, враховуючи ціну за ніч номера, кількість ночей та кількість людей. Ця процедура автоматизує розрахунок вартості бронювання, що спрощує процес оформлення бронювань. Вона також забезпечує точність розрахунків, уникаючи помилок, які можуть виникнути при ручному розрахунку.

Розроблена база даних надає всі необхідні інструменти для ефективного управління готелем. Вона дозволяє зберігати інформацію про користувачів, типи номерів, наявність кімнат та бронювання. Завдяки тригерам, зміни в базі даних автоматично фіксуються, забезпечуючи цілісність інформації. Вигляди спрощують отримання ключових даних, таких як кількість бронювань на користувача або кількість кімнат кожного типу. Процедури автоматизують складні операції, наприклад, розрахунок вартості бронювання або пошук доступних номерів на заданий період.

3.2 Організаційне забезпечення

Далі на рисунку 3.18 продемонстрована діаграма станів.[43] Діаграма станів допомагає визначити логіку системи і відслідковувати її реакцію на різні події.

Спочатку система перебуває в стані idle (бездіяльності), поки не отримає запит на доступ від користувача. Після отримання запиту, система переходить в стан перевірки (верифікація), де вона перевіряє, чи доступне НД. Якщо НД доступне, система переходить в стан «Активний», де вона здійснює запит користувача. Якщо НД не доступне, система повертається в стан «Відмова» і відхиляє запит користувача. У стані «Активний», система може отримати дві події: скасування запиту або підтвердження. У випадку скасування запиту, система повертається в стан idle. У випадку підтвердження, система переходить в стан виконання запиту, де вона завершує запит користувача і повертається в стан idle.

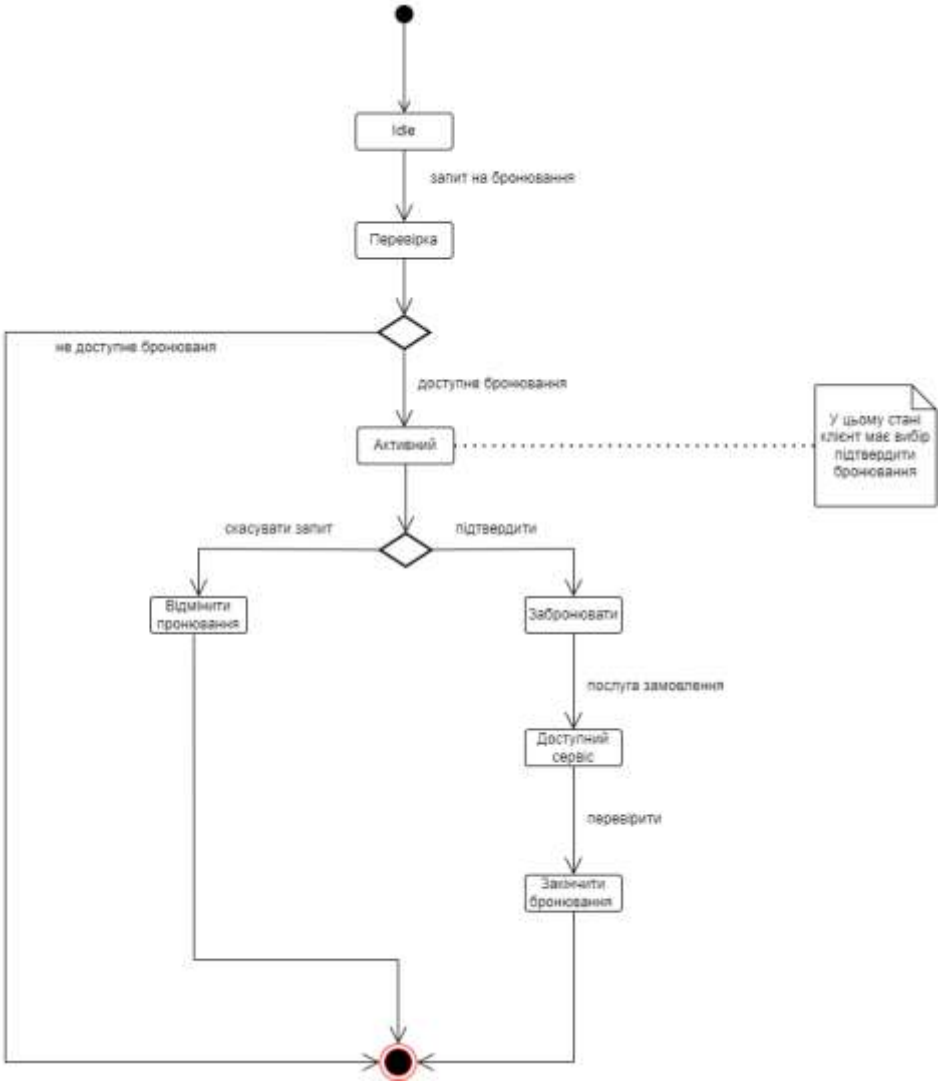


Рисунок 3.18 – Діаграма станів процесу взаємодії системи з користувачем при отриманні запиту на доступ

На рисунку 3.19 продемонстровано діаграму активності або дії, вона відображає процес відправлення відгуку або перегляд чужих відгуків.[44]

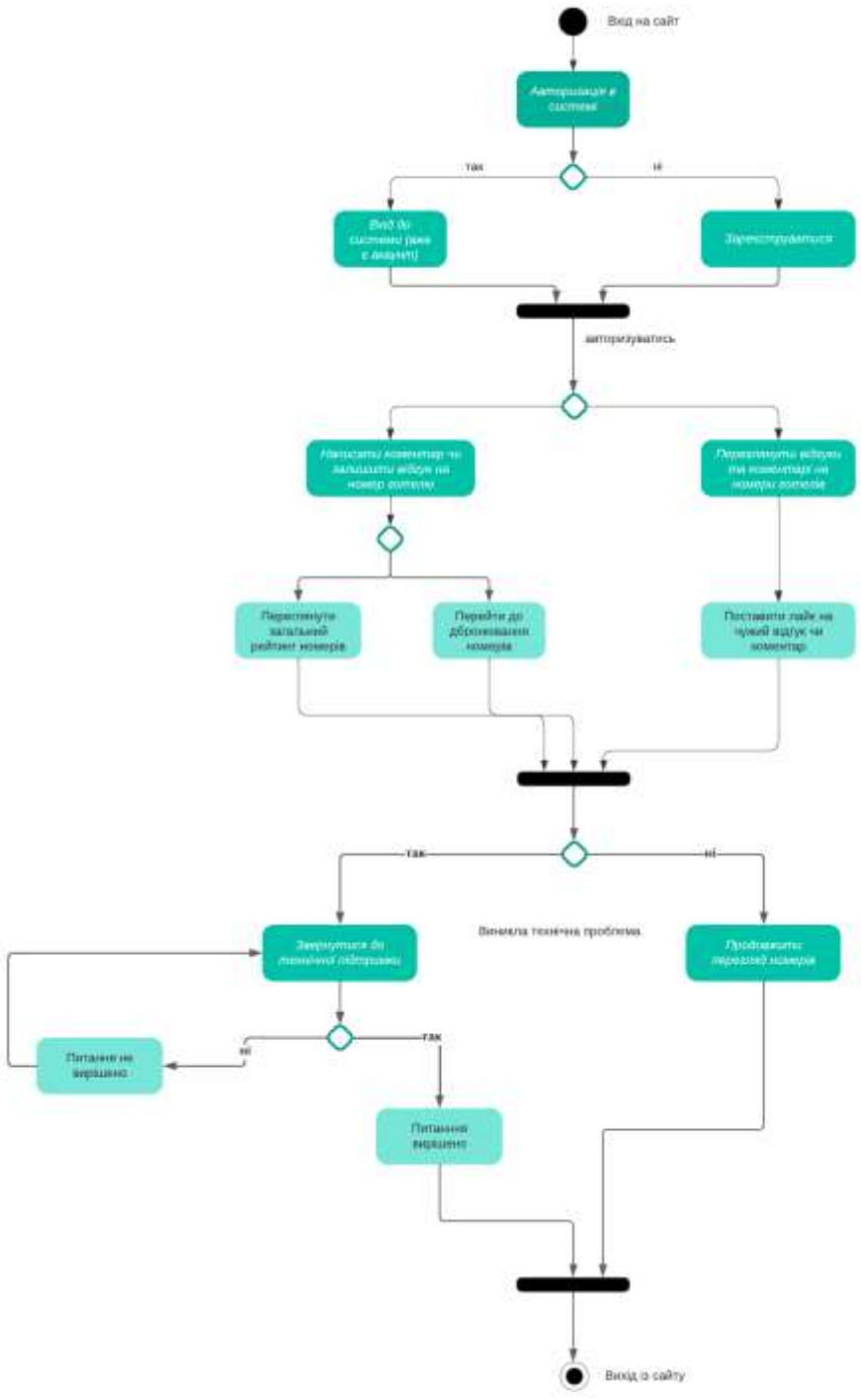


Рисунок 3.19 – Діаграма активності (дії)

Користувач входить на сайт, вибирає спосіб авторизації до системи, він може як зареєструватися, так і увійти, після авторизації користувач може або залишити відгук чи коментар до номеру готелю або переглянути чужі відгуки та

коментарі, також користувач може залишити вподобайки чужому коментарю та відгуку. Після цього користувач має вибір, переглянути загальний рейтинг номерів чи продовжити бронювання номеру. Далі є вибір чи продовжити бронювання та перегляд номерів чи звернутися до технічної підтримки в разі знаходження багів системи або виникнення технічної проблеми, якщо не допомогло, то повернутися назад, якщо допомогло, то можна вже вийти з сайту.

3.3 Програмне забезпечення

Детальніше розглянемо програмне забезпечення, яке використовувалося для розробки інформаційної системи «адміністрування готелю».

Системне програмне забезпечення:

Була використана операційна система Windows 10 для розробки і тестування інформаційної системи. ОС забезпечує стабільну роботу і надає зручний інтерфейс для програмування, налагодження і тестування.

До системних утиліт, які підтримують роботу розробницького середовища, належать утиліти для керування дисками, моніторингу системи, резервного копіювання та відновлення даних. Вони забезпечують належне функціонування як сервера, так і клієнтських машин, що використовуються в готельному бізнесі.

Прикладне програмне забезпечення:

Основна мова програмування, яка використовується для розробки серверної частини системи. Java забезпечує кросплатформену сумісність та надійність, що є критичним для систем адміністрування готелю.

Spring Framework – який використовується для створення веб-застосунків і включає в себе кілька важливих модулів:

Spring Boot дозволяє швидко налаштовувати і розгортати веб-застосунки. Автоматизує більшість конфігураційних завдань, що значно спрощує початок розробки. Наприклад, для реалізації функцій реєстрації та аутентифікації користувачів було використано Spring Boot.

Hibernate використовується для об’єктно-реляційного відображення (ORM). Це забезпечує зручну роботу з базами даних, перетворюючи дані з таблиць у об’єкти Java. Для управління даними про клієнтів і бронювання в готелі застосовано Hibernate.

JDBC (Java Database Connectivity) – API для роботи з базами даних, що надає підключення до бази даних MySQL і виконання SQL-запитів. Приклад використання в проєкті: для забезпечення доступу до даних про номери та послуги готелю використовується JDBC.

Spring Security забезпечує безпеку додатку, включаючи автентифікацію та авторизацію користувачів. Цей модуль використовується для контролю доступу до адміністративних функцій системи.

На рисунку 3.20 продемонстрована структура проєкту в середовищі Intelij Idea.

Структура проекту IC для адміністрування готелю, розробленого за допомогою Spring Framework, розподілена на декілька ключових пакетів, кожен з яких виконує конкретні функції. Основні пакети включають component, config, domain, exceptions, repository та service.

Пакет component містить контролери та фільтри, які обробляють HTTP-запити та здійснюють фільтрацію даних. Класи контролерів, такі як RoomController та UserController, відповідають за управління запитами, що надходять від користувачів, і взаємодію з сервісами для виконання бізнес-логіки. У пакеті config знаходяться конфігураційні класи, зокрема DatabaseConfiguration та SecurityConfiguration, які налаштовують підключення до бази даних та забезпечують безпеку додатку.

Пакет domain включає сутності доменної моделі, такі як Room, User та RoomReservation, які представляють основні об'єкти системи та використовуються для зберігання і передачі даних. Пакет repository містить інтерфейси репозиторіїв (RoomRepository, UserRepository), що забезпечують

доступ до бази даних і виконують CRUD операції. Пакет exceptions містить класи винятків, які обробляють помилки, пов'язані з відсутністю даних у базі.

Пакет service, реалізує бізнес-логіку додатку та взаємодіє з репозиторіями для обробки даних. Головний клас проекту, HotelApplication, відповідає за запуск Spring Boot додатку. Фрагменти коду контролерів демонструють прикладний аспект розробки, реалізуючи логіку обробки запитів та взаємодію з базою даних представлено в «Додаток Б».

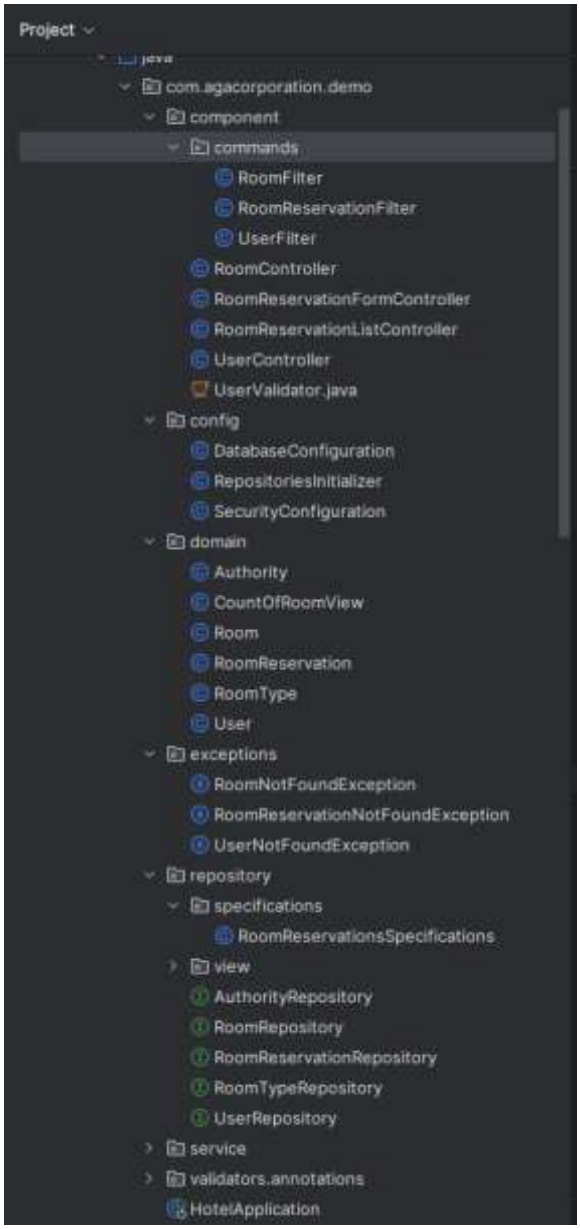


Рисунок 3.20 – Структура проєкту в середовищі Intelij Idea

MySQL система керування базами даних, яка використовується для зберігання всієї необхідної інформації про готель, клієнтів, бронювання та інші

дані. Структура бази даних розроблена з урахуванням вимог проекту і описана у «Додаток Г».

HTML, CSS, JavaScript (Thymleaf) – технології, що використовуються для розробки клієнтської частини веб-застосунку. HTML забезпечує структуру сторінок, CSS – їх стильове оформлення, а JavaScript додає інтерактивність. Наприклад, динамічні елементи та функції для управління бронюванням реалізовані за допомогою JavaScript. На рисунку 3.21 представлена структура проекту інформаційної системи адміністрування готелю, зокрема папки resources, яка містить всі необхідні ресурси для функціонування веб-додатку.

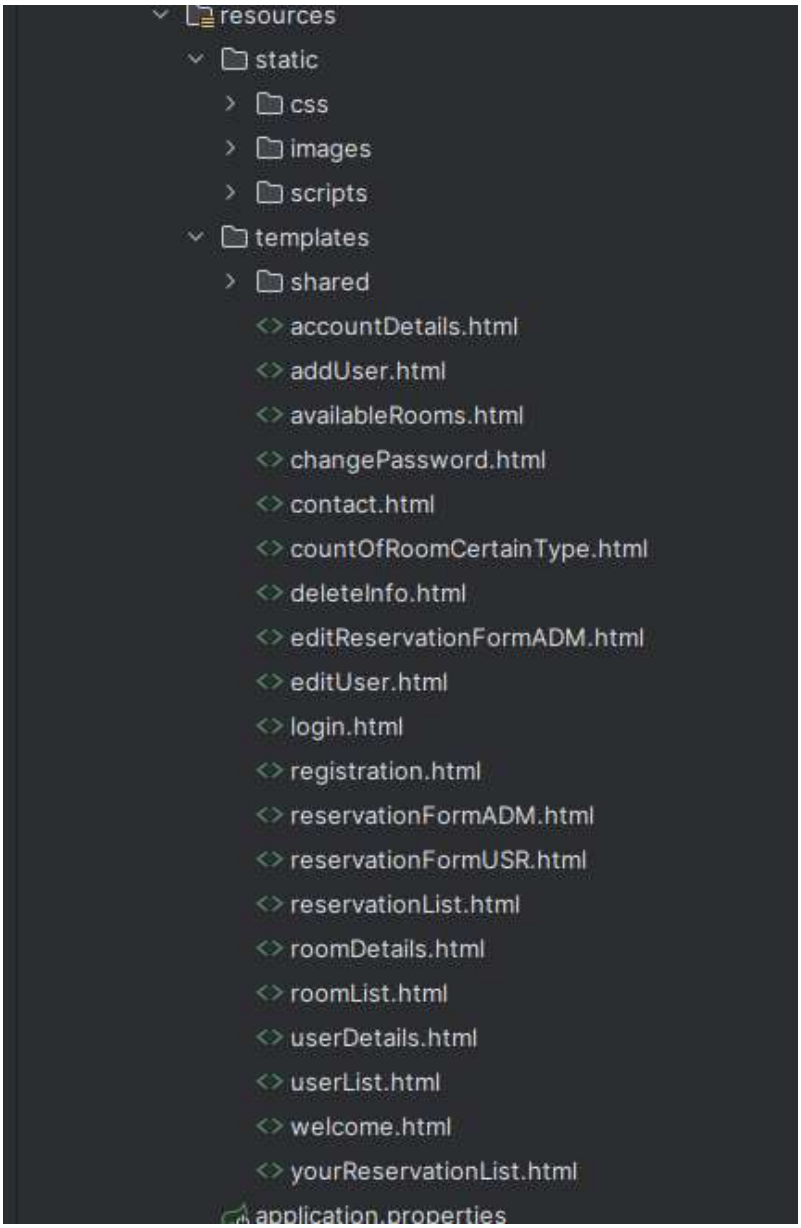


Рисунок 3.21 – Структура проєкту в середовищі Intelij Idea

Папка resources складається з двох основних підпапок: static та templates, а також файлу конфігурації application.properties.

Підпапка static містить три підпапки: css, images та scripts. Папка css містить стилі, які забезпечують візуальну привабливість та узгодженість веб-сторінок. Папка images зберігає всі зображення, такі як логотипи та іконки, що використовуються на веб-сторінках для покращення користувацького досвіду. Папка scripts містить JavaScript файли, які додають динамічну функціональність до веб-сторінок, забезпечуючи інтерактивність та покращену взаємодію з користувачем.

Підпапка templates містить підпапку shared, яка включає HTML-шаблони для рендерингу різних сторінок веб-додатку. Ці шаблони забезпечують відображення інтерфейсу користувача та інтерактивність. Наприклад, шаблон accountDetails.html відображає деталі облікового запису користувача, addUser.html – форму для додавання нового користувача, availableRooms.html – доступні кімнати для бронювання, login.html – сторінку входу, а registration.html – сторінку реєстрації. Крім того, є шаблони для редагування бронювань адміністратором, відображення списку кімнат та привітальної сторінки після входу користувача. Фрагменти HTML-коду головної сторінки інформаційної системи , фрагменти CSS коду та фрагменти JS-коду, що відповідає за створення динамічних елементів та функцій в інформаційній системі наведено в «Додаток Б».

Файл application.properties містить налаштування конфігурації додатку. Тут зберігаються параметри бази даних, налаштування безпеки, параметри серверу та інші важливі конфігурації, необхідні для коректного функціонування веб-додатку.

На рисунку 3.22 зображена діаграма компонентів системи адміністрування готелем виділено два основних блоки взаємодії: сторона адміністратора та сторона клієнта.[45]

Сторона адміністратора містить інструменти для управління користувачами, номерами та бронюваннями. Це дає адміністратору можливість здійснювати повний контроль над робочими процесами готелю, від ведення реєстру клієнтів до розподілу номерів і обробки бронювань. Крім того, функція моніторингу та оновлення забезпечує можливість стежити за поточним статусом задач та оперативно вносити зміни у систему.

Сторона клієнта оснащена веб-інтерфейсом, який надає клієнтам інтуїтивно зрозумілий доступ до різноманітних послуг готелю. Веб-інтерфейс спрощує процес бронювання та надання послуг, забезпечуючи високий рівень задоволеності клієнтів. Компонент "Перелік послуг" дає можливість користувачам переглядати та вибирати послуги, що їх пропонує готель, а також робити бронювання в режимі реального часу.

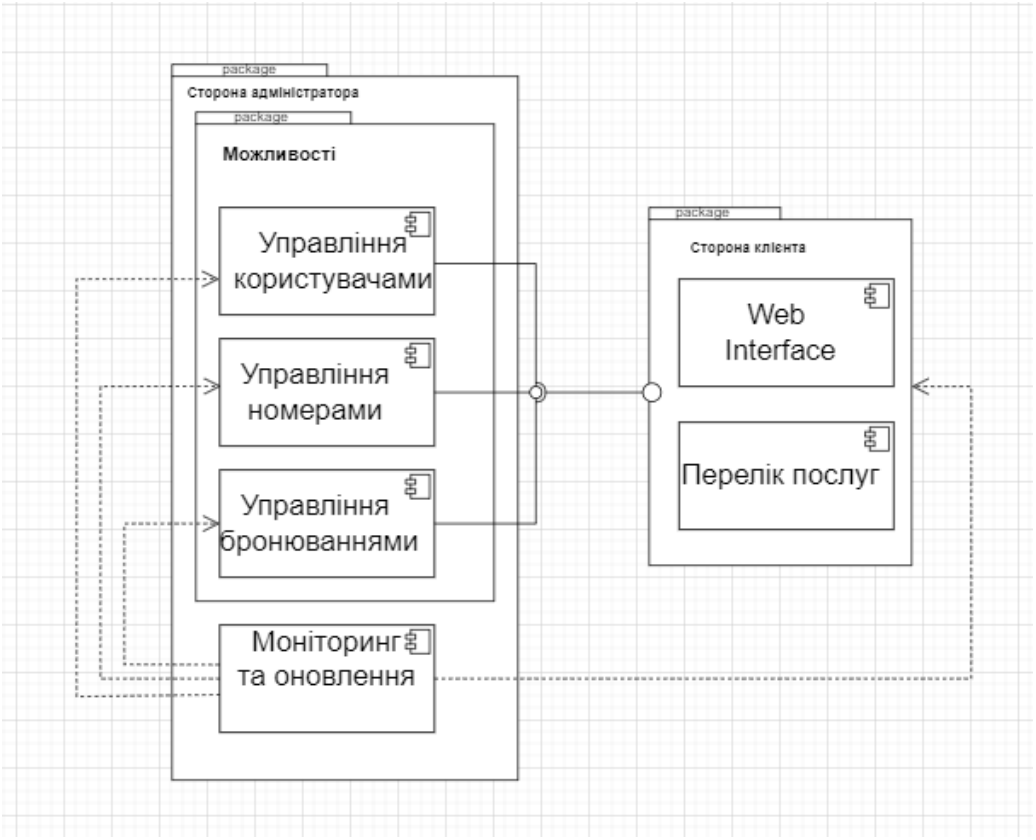


Рисунок 3.22 – Діаграма компонентів системи управління готелем – ілюстрацією компоненти апаратного та програмного вузла

Діаграма відображає взаємозв'язок та інтеграцію між різними компонентами системи, підкреслюючи її модульну структуру та легкість

навігації для різних типів користувачів. Чітке розмежування функціоналу між адміністративною та клієнтською частинами забезпечує ефективність управління готелем та високу якість обслуговування клієнтів.

Таким чином, у цьому розділі розглянуто основні компоненти програмного забезпечення, використані в проекті забезпечуючи повне розуміння використаних технологій і підходів для створення ефективної інформаційної системи адміністрування готелю.

3.4 Технічне забезпечення

На представленій діаграмі розгортання Angular Spring Boot Full-Stack для адміністрації готелю (див. рисунок 3.23), видно, що вона складається з двох основних компонентів: Angular Frontend HTTP Library і Spring Boot Backend App, яке взаємодіє з MySQL сервером для зберігання даних.[46]

Фронтенд-частина, розроблена з використанням Angular, включає в себе різноманітні компоненти та сервіси, які забезпечують інтерфейс користувача. Використання Axios HTTP бібліотеки дозволяє здійснювати асинхронні запити до сервера, а Temlate8 забезпечує використання шаблонів для створення динамічних HTML-сторінок. Компоненти Angular відповідають за відображення інтерфейсу та взаємодію користувача з системою, тоді як сервіси Angular обробляють бізнес-логіку на стороні клієнта та комунікацію з бекендом.

Бекенд-частина, побудована на Spring Boot, містить Spring REST контролери, які управляють HTTP запитами та відповідями. Модель Hotel описує бізнес-об'єкти системи, що використовуються для представлення даних.

DAO (Data Access Object) – шаблон проектування, що забезпечує ізоляцію бізнес-логіки додатку від деталей зберігання даних. DAO інкапсулює всі операції з базою даних, забезпечуючи зручне управління даними про користувачів та бронювання.

DAO (Repository) компоненти відповідають за взаємодію з базою даних, а сервіси бекенда виконують логіку обробки даних, отриманих від фронтенду та інших джерел.

MySQL сервер, в свою чергу, поділений на дві бази даних: основну базу даних для зберігання основних даних системи та базу даних аналітики, яка може використовуватися для збору та аналізу даних, з метою покращення бізнес-процесів та прийняття рішень. Така структура забезпечує міцну основу для створення масштабованої, ефективної та надійної інформаційної системи адміністрування готелю.

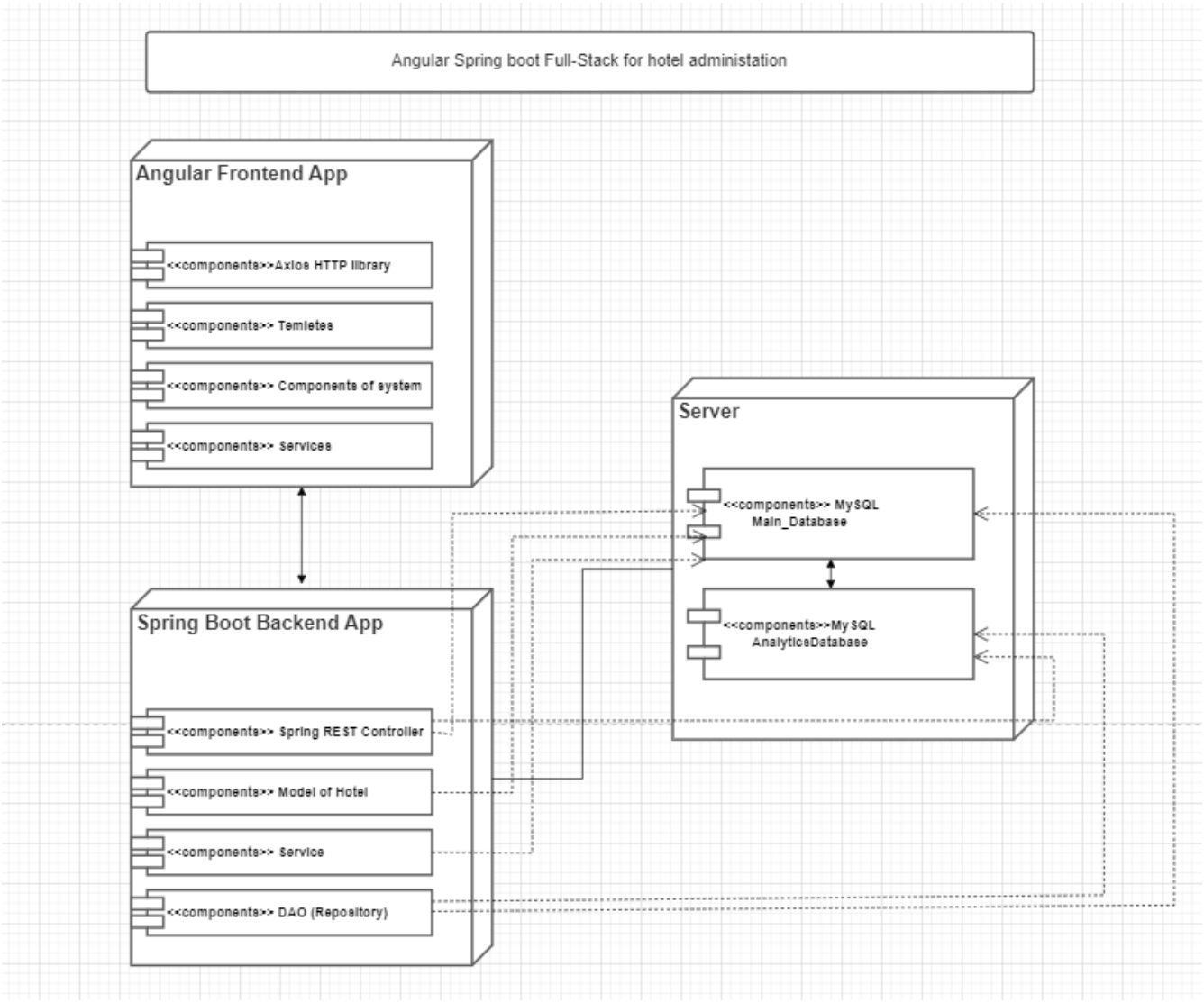


Рисунок 3.23 – Діаграма розгортання Angular Spring Boot Full-Stack для адміністрування готелю

3.5 Результати реалізації інформаційної системи

В процесі виконання кваліфікаційної бакалаврської роботи було розроблено інформаційуну систему, що призначена для автоматизації та оптимізації процесів адміністрування в готельному бізнесі. ІС демонструє працездатність ключових функціональних модулів, що є основою для створення повноцінної системи управління готелем.[49]

В результаті реалізації ІС було досягнуто наступних результатів:

1. Функціональний модуль управління користувачами:

Система дозволяє реєструвати нових користувачів, авторизовуватися в системі, редагувати особисті дані та керувати правами доступу. Реалізовано розмежування ролей користувачів: адміністратор має розширені права для управління системою, клієнти – обмежені права для бронювання та перегляду інформації.

2. Модуль управління номерним фондом:

Адміністратор має можливість додавати, редагувати та видаляти типи номерів та інформацію про окремі номери. Користувачі можуть переглядати інформацію про кожен номер, включаючи кількість ліжок, поверх, ціну, послуги та максимальну кількість гостей.

3. Модуль бронювання номерів:

Ключовий модуль системи дозволяє користувачам шукати доступні номери на заданий період, використовуючи фільтри за кількістю гостей, датами заїзду та виїзду. Після вибору відповідного номера користувач може оформити бронювання, вказавши необхідні дані. Система фіксує деталі бронювання та дозволяє адміністратору керувати статусами бронювань, такими як «підтверджено», «скасовано» або «оплачено».

4. Модуль обліку платежів:

Реалізовано облік платежів, хоча на даному етапі не інтегрований з платіжними системами. Система фіксує факт оплати за бронювання та генерує

квитанції. В подальшому можливо розширити цей модуль для підтримки онлайн-оплати з використанням платіжних шлюзів.

5. Додатковий функціонал:

ІС включає можливість переглядати та залишати відгуки про номери, що дозволяє збирати зворотний зв'язок від клієнтів. Також реалізовано перегляд статистики бронювань та завантаженості номерів, що допомагає адміністратору аналізувати дані та приймати рішення.

Для перевірки працездатності системи було проведено тестування з використанням різних типів тестів, включаючи модульні, інтеграційні, системні, приймальні та тести-сценарії.[50]

Тестування було проведено в середовищі IntelliJ IDEA, що дозволило перевірити коректність роботи окремих модулів та системи в цілому. Результати тестування підтвердили відповідність системи поставленим вимогам та коректність реалізації основного функціоналу.[51]

ІС демонструє потенціал для створення повноцінної системи управління готелем. ІС реалізує ключові функції, необхідні для ефективного управління бронюваннями, номерним фондом, користувачами та обліком платежів.

Для подальшого розвитку системи виділено такі рекомендації:

Інтегрувати систему з платіжними шлюзами для забезпечення можливості онлайн-оплати бронювань, підвищуючи зручність для клієнтів та автоматизуючи процес обробки платежів.

Розширити функціонал системи, а саме додати модулі для управління персоналом, обліку фінансів, ведення складу, аналітики та звітності. Це дозволить створити комплексне рішення для управління всіма аспектами готельного бізнесу.

Розробити мобільний додаток для забезпечення доступу до системи з мобільних пристроїв, підвищуючи зручність для клієнтів та персоналу, що дозволить керувати бронюваннями, переглядати інформацію про номери та здійснювати інші операції з будь-якого місця.

Впровадити розширену систему безпеки для захисту конфіденційної інформації від несанкціонованого доступу. Це може включати двофакторну автентифікацію та інші механізми захисту.

Покращити інтерфейс користувача для забезпечення ще більш зручного та інтуїтивного використання системи, вдосконалення дизайну, додавання нових функцій та оптимізацію навігації.

Підводячи підсумок, розроблена ІС є перспективним рішенням для автоматизації та оптимізації процесів в готельному бізнесі. ІС здатна значно покращити якість обслуговування клієнтів, підвищити ефективність управління та забезпечити безпеку даних.

РОЗДІЛ 4

ТЕСТУВАННЯ І ВЕРИФІКАЦІЯ ПРОЄКТНИХ РІШЕНЬ

4.1 Розробка тест-кейсів

Розробка тест-кейсів є важливою складовою процесу тестування програмного забезпечення. Кожен вид тест-кейсів має свою функціональну спрямованість та сприяє виявленню певних категорій помилок та недоліків у програмі. [51]

Характеристика кожного виду тест-кейсів та їх використання у контексті проєктування інформаційної системи для адміністрування готелю:

1. Модульні тест-кейси:

Назва: Перевірка функції додавання нового користувача.

Опис: Перевіряє, чи працює функція додавання нового користувача до системи.

Призначення: Перевірка правильності роботи окремих модулів програми або їх функціональних складових.

Використання: Допомагають ізолювати та тестувати окремі частини програми, щоб виявити та виправити помилки на ранніх етапах розробки.

2. Інтеграційні тест-кейси:

Назва: Перевірка взаємодії між модулем бронювання і модулем оплати.

Опис: Перевіряє, чи правильно обробляються дані про бронювання в процесі оплати.

Призначення: Перевірка взаємодії між різними модулями або компонентами програми.

Використання: Виявлення помилок у взаємодії між компонентами системи та підтвердження їх коректної роботи в комплексі.

3. Системні тест-кейси:

Назва: Перевірка функціональності пошуку номерів.

Опис: Перевіряє, чи працює функція пошуку номерів за різними критеріями.

Призначення: Перевірка функціональності системи як цілісної єдності відповідно до вимог та очікувань користувачів.

Використання: Встановлення того, чи відповідає програмне забезпечення вимогам та специфікаціям, а також чи забезпечує воно потрібну продуктивність та надійність.

4. Приймальні тест-кейси:

Назва: Перевірка готовності системи до випуску.

Опис: Перевіряє, чи готова система до використання користувачами та чи відповідає вона поставленим вимогам.

Призначення: Перевірка готовності системи до використання користувачами та її відповідність поставленим вимогам.

Використання: Визначення, чи готова система до випуску, та перевірка, чи задовольняє вона потреби та очікування кінцевих користувачів.

5. Тести-сценарії:

Назва: Сценарій бронювання номеру з використанням пошуку та фільтрації.

Опис: Симулює реальний процес бронювання номеру з використанням пошуку та фільтрації за різними критеріями.

Альтернативний сценарій:

Назва: Сценарій бронювання номеру з помилкою оплати.

Опис: Симулює випадок, коли оплата бронювання не вдалася з технічних причин.

Призначення: Тестування поведінки системи в конкретних сценаріях використання.

Використання: Перевірка роботи системи в реальних ситуаціях, включаючи різноманітні дії та взаємодію з користувачами та іншими системами.

У «Додатку А» наведено тестова документація, підготовлена засобами Enterprise Architect.

4.2 Трасування вимог до системи

Трасування вимог є важливим кроком у процесі розробки програмного забезпечення. Потрібно для визначення взаємозв'язків між вимогами, прецедентами, тестовими кейсами та елементами користувацького інтерфейсу і допомагає гарантувати, що реалізація функціональності програмного забезпечення є повною і відповідає вимогам користувачів та бізнес-цілям.[52]

Трасування вимог – це процес відстеження вимог від початку до реалізації та тестування. В інформаційній системі адміністрування готелю трасування системних вимог допомагає забезпеченню відповідності функціональності системи потребам користувачів і бізнес-цілям проєкту.[52]

Характеристика використання діаграм відстеження та матриць взаємозв'язків для нашої ІС:

Діаграма трасування дозволяє візуалізувати, як кожна вимога впливає на функціональність системи і взаємозв'язок між кожною вимогою. Наприклад, вона може показати, як прецедент "Забронювати вільний номер" виконує вимогу "бронювання номерів" і які тести необхідно виконати для перевірки цієї функціональності. Також, взаємодія відбувається між вимогами для спрощення читабельності діаграми.

Матриці зв'язків надають додаткову інформацію про взаємозв'язки між різними елементами системи. Вони використовуються, для демонстрації відповідності між вимогами і прецедентами, вимогами і тестовими кейсами, а також вимогами і елементами інтерфейсу. Ці матриці можна використовувати для кращого розуміння того, як кожна вимога відповідає і взаємодіє з різними аспектами системи.

Трасування вимог системи допомагає забезпечити високу якість інформаційної системи адміністрування готелю та відповідність її функціональності потребам користувачів і бізнесу.

На діаграмі трасування вимог для інформаційної системи адміністрування готелю можуть використані такі зв'язки:

Зв'язок «Trace» для трасування вимог до системи (функціональні або нефункціональні вимоги) до відповідних прецедентів або сценаріїв використання.

Зв'язок «Realization» – компоненти, модулі програмного забезпечення пов'язані з абстрактними прецедентами або функціональними вимогами, які вони реалізують. Класи системи пов'язані з відповідними прецедентами, функціональними вимогами, які вони втілюють.

Зв'язок «Abstraction» – абстрактні класи та інтерфейси пов'язані з конкретними компонентами системи, які їх реалізують.

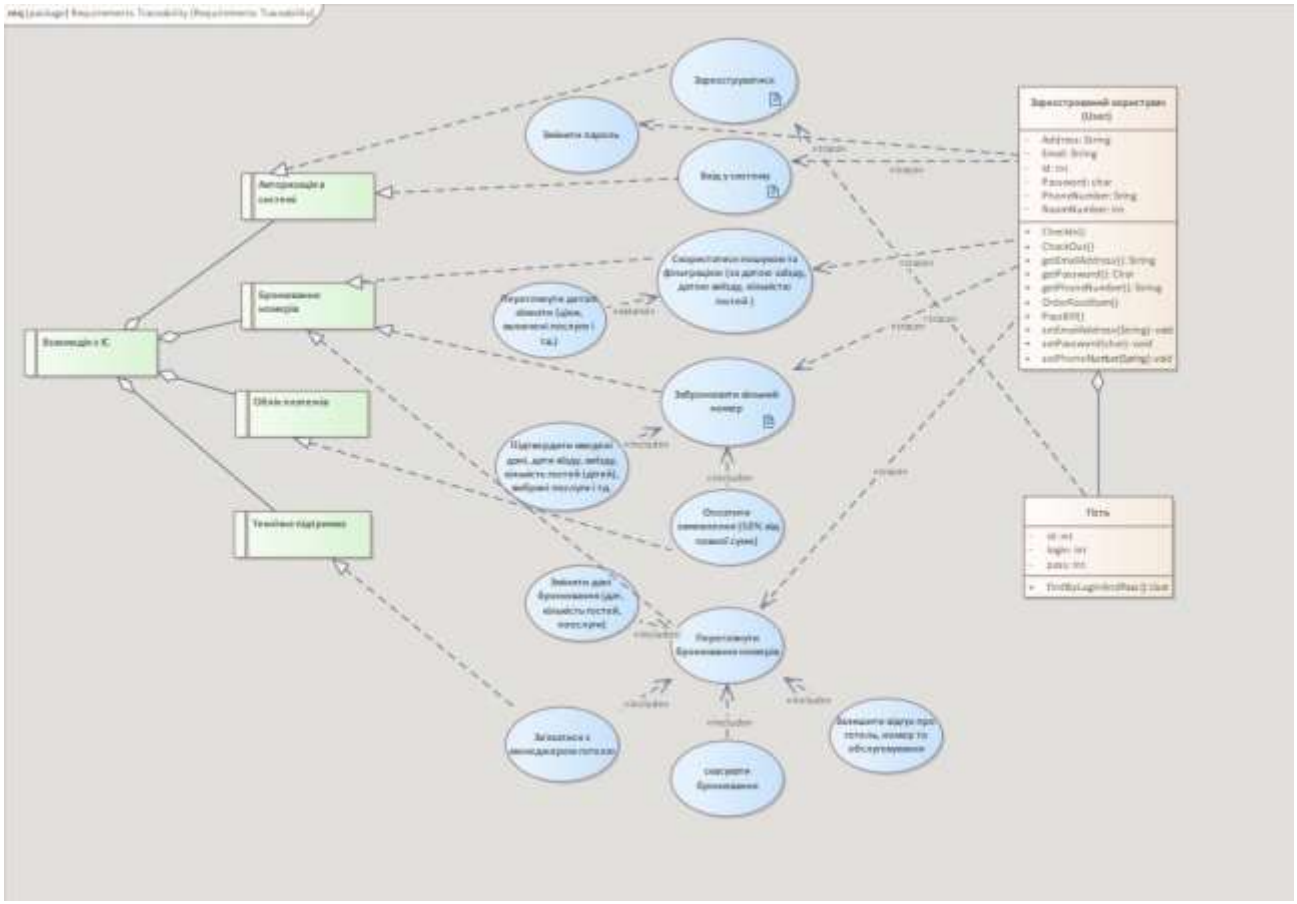


Рисунок 4.1 – Діаграма трасування в системі

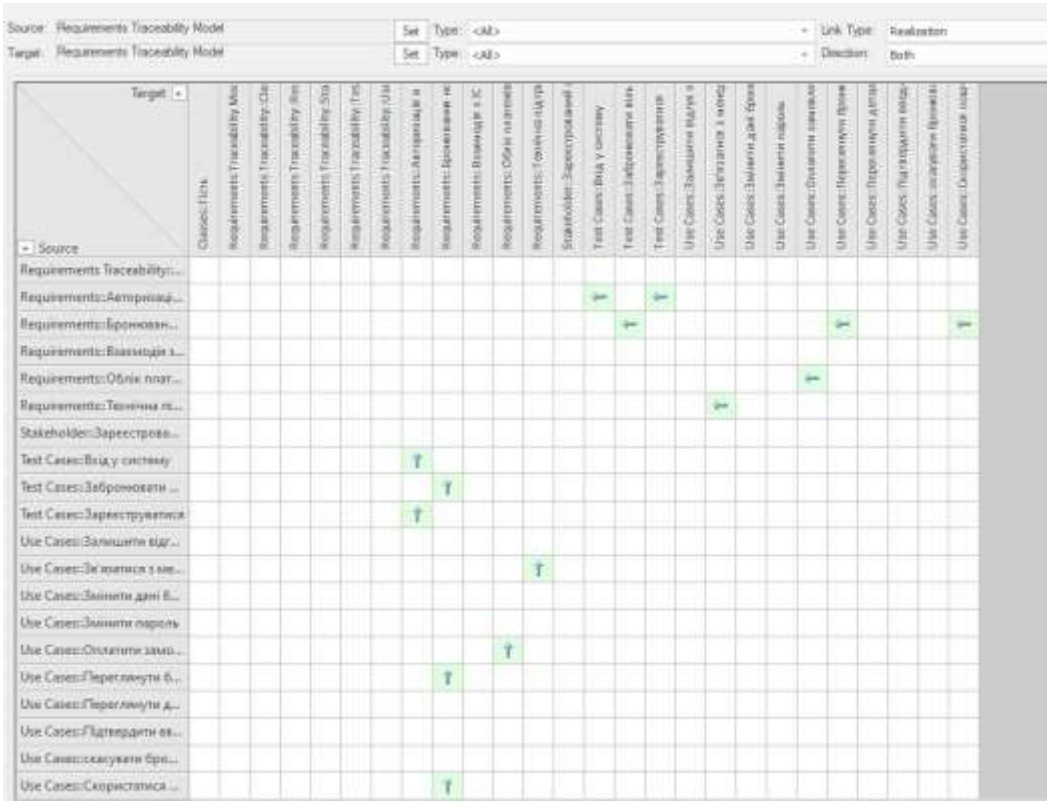


Рисунок 4.2 – Матриця взаємозв’язків в системі «Realization»

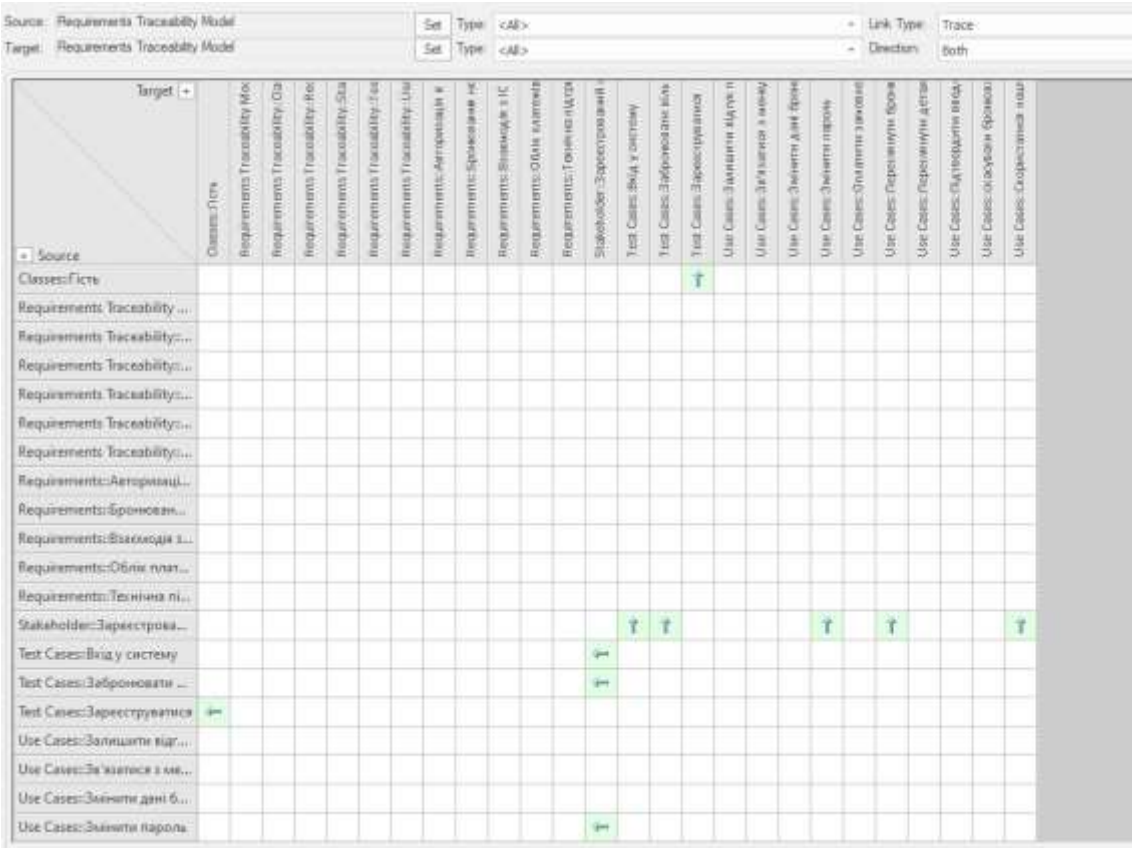


Рисунок 4.3 – Матриця взаємозв’язків в системі «Trace»

ВИСНОВКИ

Кваліфікаційна бакалаврська робота присвячена актуальній проблемі сучасного готельного бізнесу – автоматизації процесів управління та покращенню якості обслуговування клієнтів. В роботі проведено комплексне дослідження предметної області, детальний аналіз існуючих рішень та розроблено концепцію інформаційної системи, що відповідає сучасним вимогам ринку гостинності.

У першому розділі детально розглянуто особливості функціонування сучасних готельних підприємств, включаючи процеси управління номерним фондом, бронювання, взаємодії з клієнтами, організацію роботи персоналу, ведення фінансової діяльності, а також маркетингу та формування позитивного іміджу готелю. Проведено порівняльний аналіз популярних ІС, таких як Booking.com, Agoda.com, Hotelogix, та Opera by Oracle Hospitality. Виявлено їх сильні та слабкі сторони, що дозволило сформувати вимоги до власної розробки, фокусуючись на ефективності управління, гнучкості, безпеці даних та інтеграції з іншими системами. На основі проведеного аналізу сформульовано ключові принципи розробки ІС, що включають використання трьохрівневої архітектури (MVC), об’єктно-орієнтованого підходу (ООП), мови програмування Java, фреймворку Spring Boot, бази даних MySQL та сучасних фронтенд-технологій (Bootstrap, HTML, CSS, JavaScript, Thymleaf, Angular).

В другому розділі детально описуються функціональні, нефункціональні та бізнес-вимоги до системи. Також, увагу приділено аспектам безпеки, зручності використання, масштабованості, інтеграції з іншими системами та можливостям звітності та аналітики. Для реалізації проєкту обрано методологію Rapid Application Development (RAD) у поєднанні з об’єктно-орієнтованим підходом, що забезпечує гнучкість та швидкість розробки. За допомогою середовища OPCAT було створено модель ІС, а саме ієрархії OPD-діаграм – повний набір процесів і об'єктів разом з їхніми станами для ІС. Проаналізувавши

всі вимоги та бізнес-цілі, створено інформаційну модель задачі, що відображає структуру даних, у вигляді таблиць вхідні та вихідні повідомлення, а також взаємодію між різними компонентами системи. Наступним кроком було розроблено математичне забезпечення та спроєктова алгоритм функціонування системи, на якому продемонстровано кожний крок клієнта, адміністратора (персоналу готелю). Розроблено діаграму прецедентів (Use case), представлено матриці відповідності вимог і прецедентів, діаграми послідовності (Sequence), класів (Class), визначення блоків для подання внутрішньої структури системи (BDD), процес оформлення замовлення (IBD), які детально описують функціональні можливості системи, взаємодію між користувачами та системою, а також потоки даних.

Змодельовано приклади користувацького інтерфейсу ІС, які створені в Enterprise Architect та клікабельний прототип інтерфейсу користувача у Figma, що дозволяє наглядно продемонструвати основні функціональні можливості. Сторінки прототипу ІС продемонстровано в Додатку В.

У третьому розділі продемонстровано спроєктовану базу даних, а саме структуру бази даних, ключові таблиці, зв’язки між ними, тригери та процедури. Використання MySQL Workbench забезпечує зручний інструмент для проєктування та управління базою даних. Розроблено діаграми станів (State) та активності (Activity), що деталізують процеси взаємодії системи з користувачами та послідовність виконання операцій. Створено діаграму компонентів, що ілюструє взаємодію між модулями системи, включаючи фронтенд та бекенд компоненти. Також, розроблено діаграму розгортання, що відображає фізичну архітектуру системи, включаючи веб-сервер, сервер додатків та базу даних. Представлено скріншоти інтерфейсу користувача та результати роботи основних функцій системи.

В четвертому розділі розроблено різні типи тест-кейсів (модульні, інтеграційні, системні, приймальні, тести-сценарії), що охоплюють основні функціональні можливості системи. Продемонстровано діаграми трасування та

матриці зв'язків для перевірки відповідності реалізованої функціональності початковим вимогам.

В ході виконання кваліфікаційної бакалаврської роботи було розроблено комплексну інформаційну систему для адміністрування готелю, яка відповідає сучасним вимогам до функціональності, безпеки та зручності використання. Система надає можливості для ефективного управління номерним фондом, бронюваннями, обслуговуванням клієнтів, персоналом та фінансовими операціями. Використання сучасних технологій та методологій проєктування забезпечує масштабованість, гнучкість та надійність системи.

В ході проєктування особлива увага була приділена забезпеченню безпеки даних. Впроваджено систему авторизації та контролю доступу, яка дозволяє розмежовувати права користувачів та захищати конфіденційну інформацію. Використання шифрування даних забезпечує їх цілісність та захист від несанкціонованого доступу. Розроблений користувацький інтерфейс є інтуїтивно зрозумілим та зручним у використанні. Його дизайн базується на принципах юзабіліті, що забезпечує легкість навігації та швидке оволодіння системою навіть для користувачів без технічного досвіду. Важливим аспектом проєкту є його масштабованість. Завдяки використанню модульної архітектури, систему можна легко розширювати та додавати новий функціонал.

Результати кваліфікаційної бакалаврської роботи можуть бути використані для впровадження в реальних готельних підприємствах, система може бути адаптована під конкретні потреби будь-якого готелю, незалежно від його розміру та спеціалізації. Розроблені моделі та алгоритми можуть слугувати базою для створення більш складних та спеціалізованих ІС. Також, робота демонструє поєднання теоретичних знань та практичних навичок у розробці інформаційної системи для адміністрування готелю.

Рекомендації щодо подальшого розвитку системи включають інтеграцію з системами управління відносинами з клієнтами (CRM), що дозволить персоналізувати обслуговування та підвищити лояльність клієнтів; інтеграцію з

платіжними системами та сервісами онлайн-бронювання, а саме розширення функціоналу для управління персоналом та заробітною платою; впровадження модуля аналітики та звітності для оптимізації бізнес-процесів; розробка мобільного додатку для клієнтів та персоналу; впровадження модуля управління лояльністю клієнтів з можливістю нарахування бонусів та знижок; розширення аналітичних можливостей системи для прогнозування попиту та оптимізації ціноутворення; розробка інтерфейсів на додаткових мовах, що сприятиме залученню міжнародних клієнтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) The Role of Information Technology in the Hospitality Industry [Електронний ресурс]. URL: https://www.researchgate.net/publication/344012960_The_Role_of_Information_Technology_in_the_Hospitality_Industry (дата звернення: 20.01.2024).
2) Застосування новітніх інформаційних систем управління готелем [Електронний ресурс]. URL: https://tourlib.net/statti_ukr/ryabenka.htm (дата звернення: 20.01.2024).
3) Менеджмент туристичної індустрії. Автоматизація процесу управління підприємствами готельного господарства [Електронний ресурс]. URL: https://buklib.net/books/32532/ (дата звернення: 20.01.2024).
4) Фрієва Я. В. Системи резервування та бронювання послуг [Електронний ресурс]. URL: https://vseosvita.ua/lesson/11-systemy-rezervuvannia-ta-broniuvannia-posluh-434980.html (дата звернення: 20.01.2024).
5) Kasavana, M. L., & Brooks, R. M. (2001). Managing Front Office Operations. Hospitality Management. John Wiley & Sons, Inc. [Книга]. (дата звернення: 21.01.2024).
6) O'Connor, P., & Frew, A. J. (2004). Information technology adoption in the hotel sector. International Journal of Hospitality Management, 23(1), 1-18. [Стаття]. URL: https://www.tandfonline.com/doi/full/10.1080/19368623.2016.1173297 (дата звернення: 21.01.2024).
7) Booking.com [Електронний ресурс]. URL: https://www.booking.com/index.uk.html?aid=309654;label=hotels-ukrainian-uk-qiIpaEVbamMmcLZNTvGUgAS506720657925:pl:ta:p1:p2:ac:ap:neg:fi:tiaud-297601666475:kwd-965735174632:lp9061012:li:dec:dm;ws=&gclid=Cj0KCQjw1tGUBhDXARIsAIJx01lodDYF1Gllcuu8WWMk_RLwQZGW9vQzb96QdI3lQupevec2ziTOi-8aAhS7EALw_wcB (дата звернення: 21.01.2024).

8) Про Booking.com TM [Електронний ресурс]. URL: https://www.booking.com/content/about.uk.html (дата звернення: 21.01.2024).
9) Agoda.com [Електронний ресурс]. URL: https://www.agoda.com/uk-ua/?cid=1844104 (дата звернення: 21.01.2024).
10) Hotelogix: Cloud-based Hotel Management System [Електронний ресурс]. URL: https://www.hotelogix.com/ (дата звернення: 22.01.2024).
11) Opera Cloud: Hotel Property Management System [Електронний ресурс]. URL: https://www.oracle.com/industries/hospitality/opera-cloud/ (дата звернення: 22.01.2024).
12) MVC: Model, View, Controller [Електронний ресурс]. URL: https://www.codecademy.com/article/mvc (дата звернення: 26.01.2024).
13) Hernandez R. D. The Model View Controller Pattern – MVC Architecture and Frameworks Explained (2021) [Стаття]. URL: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/ (дата звернення: 27.01.2024).
14) Java [Електронний ресурс]. URL: https://www.java.com/. (дата звернення: 27.01.2024).
15) Переваги і недоліки мови програмування Java [Електронний ресурс]. URL: https://qagroup.com.ua/publications/benefits-of-java/ (дата звернення: 27.01.2024).
16) IntelliJ IDEA [Електронний ресурс]. URL: https://www.jetbrains.com/idea/ (дата звернення: 28.01.2024).
17) Spring Boot [Електронний ресурс]. URL: https://spring.io/projects/spring-boot (дата звернення: 29.01.2024).
18) Spring Framework: Why Spring? [Електронний ресурс]. URL: https://spring.io/why-spring (дата звернення: 29.01.2024).
19) Spring Tutorial [Електронний ресурс]. URL: https://www.geeksforgeeks.org/spring/# (дата звернення: 29.01.2024).
20) База даних MySQL [Електронний ресурс]. URL: https://uk.photo-555.com/2686691-what-is-mysql-database. (дата звернення: 02.02.2024).

21) MySQL [Електронний ресурс]. URL: https://www.mysql.com/. (дата звернення: 02.02.2024).
22) Introduction to HTML [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML (дата звернення: 05.02.2024).
23) Introduction to CSS [Електронний ресурс]. URL: layouthttps://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction (дата звернення: 05.02.2024).
24) BootStrap [[Електронний ресурс]. URL: http://savelink.org.ua/shho-take-bootstrap/ (дата звернення: 09.02.2024).
25) Огляд популярних JavaScript фреймворків [Електронний ресурс]. URL: https://foxminded.ua/freimvorky-javascript/ (дата звернення: 09.02.2024).
26) Sommerville, I. (2010). Software engineering. Addison-Wesley. [Книга]. URL: https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf/ (дата звернення: 10.02.2024).
27) IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/document/720577 (дата звернення: 11.02.2024).
28) Wiegers, K. E., & Beatty, J. (2013). Software requirements. Pearson Education. [Книга]. URL: https://ptgmedia.pearsoncmg.com/images/9780735679665/samplepages/9780735679665.pdf (дата звернення: 12.02.2024).
29) Martin, R. C. (2002). Agile software development: principles, patterns, and practices. Pearson Education. [Книга]. URL: https://dl.ebooksworld.ir/motoman/Pearson.Agile.Software.Development.Principles.Patterns.and.Practices.www.EBooksWorld.ir.pdf (дата звернення: 14.02.2024).
30) Rapid Application Development (RAD): A Complete Guide [Електронний ресурс]. URL: https://www.guru99.com/rad-model.html (дата звернення: 15.02.2024).

31) Rapid Application Development [Електронний ресурс] // INFORMICUS. – URL: http://www.informicus.ua/default.aspx?SECTION=6&id=93 (дата звернення: 15.02.2024).
32) Object-Oriented Programming (OOP) Concepts in Java [Електронний ресурс]. URL: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/ (дата звернення: 20.02.2024).
33) OPM – Object-Process Methodology [Електронний ресурс]. URL: http://www.opcat.com/opm.htm (дата звернення: 25.02.2024).
34) Hotel Management Systems: Features and Benefits [Електронний ресурс]. URL: https://www.softwareadvice.com/hotel-management/ (дата звернення: 27.02.2024).
35) Quantitative Methods in Hospitality and Tourism Research [Електронний ресурс]. URL: https://journals.sagepub.com/doi/full/10.1177/1096348018805530 (дата звернення: 28.02.2024).
36) System Behavior Modeling [Електронний ресурс]. URL: https://www.uml-diagrams.org/behavior-diagrams.html (дата звернення: 01.03.2024).
37) Three-Tier Architecture [Електронний ресурс]. URL: https://www.guru99.com/three-tier-architecture.html (дата звернення: 01.03.2024).
38) Client-Server Architecture [Електронний ресурс]. URL: https://www.tutorialspoint.com/software_engineering/client_server_architecture.htm (дата звернення: 21.05.2024).
39) User Interface Design [Електронний ресурс]. URL: https://www.usability.gov/what-and-why/user-interface-design.html (дата звернення: 01.03.2024).
40) User Interface Prototyping [Електронний ресурс]. URL: https://www.interaction-design.org/literature/topics/ui-prototyping (дата звернення: 04.03.2024).
41) Векторний онлайн-сервіс розробки інтерфейсів [Електронний ресурс] // Figma. URL: https://www.figma.com/ (дата звернення: 04.03.2024).

42) JDBC API [Електронний ресурс]. URL: https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html (дата звернення: 09.03.2024).
43) UML State Machine Diagrams [Електронний ресурс]. URL: https://www.uml-diagrams.org/state-machine-diagrams.html (дата звернення: 15.03.2024).
44) UML Activity Diagrams [Електронний ресурс]. URL: https://www.uml-diagrams.org/activity-diagrams.html (дата звернення: 15.03.2024).
45) UML Component Diagrams [Електронний ресурс]. URL: https://www.uml-diagrams.org/component-diagrams.html (дата звернення: 22.03.2024).
46) Deployment Diagrams [Електронний ресурс]. URL: https://www.uml-diagrams.org/deployment-diagrams.html (дата звернення: 22.03.2024).
47) Онлайн-програмне забезпечення для створення блок-схем [Електронний ресурс] // Draw.IO. URL: https://app.diagrams.net/ (дата звернення: 25.03.2024).
48) Reynolds, R. (2016). Information Systems for Hospitality and Tourism. [Книга]. URL: https://www.researchgate.net/publication/345761479_Philosophies_of_Hospitality_and_Tourism_Giving_and_Receiving (дата звернення: 08.04.2024).
49) MVC Architecture [Електронний ресурс]. URL: https://www.tutorialspoint.com/mvc_framework/mvc_architecture.htm (дата звернення: 15.04.2024).
50) Software Testing [Електронний ресурс]. URL: https://www.tutorialspoint.com/software_testing/index.htm (дата звернення: 21.04.2024).
51) Types of Software Testing [Електронний ресурс]. URL: https://www.guru99.com/types-of-software-testing.html (дата звернення: 21.04.2024).
52) Example SysML diagrams [Електронний ресурс]. URL: https://sparxsystems.com/enterprise_architect_user_guide/16.1/the_application_desktop/create_traceability_diagrams.html (дата звернення: 21.04.2024).

ДОДАТКИ

ДОДАТОК А

Тестова документація, підготовлена засобами Enterprise Architect

Документація містить у собі описи різних тестових сценаріїв, які покривають різні аспекти системи, включаючи функціональність, продуктивність, безпеку тощо. Це допомагає забезпечити повноту тестування системи та виявити потенційні проблеми або недоліки ще на етапі проєктування.

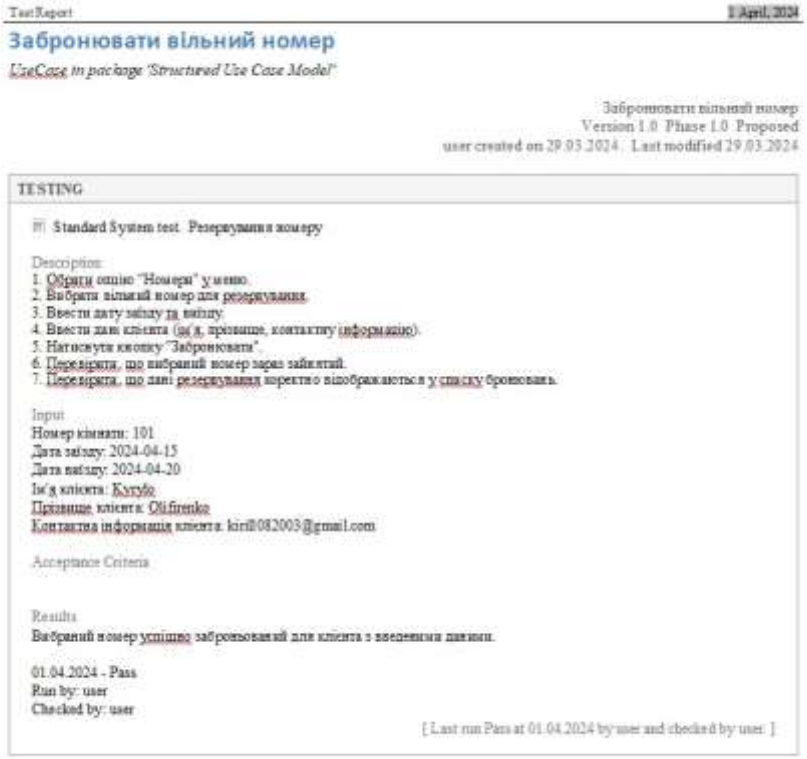
Тестова документація включає в себе плани тестування продуктивності та надійності, що дозволяє оцінювати, наскільки система відповідає вимогам щодо швидкодії та стабільності в реальних умовах експлуатації.

Раннє виявлення помилок або недоліків за допомогою тестування дозволяє уникнути серйозних проблем на пізніших етапах розробки, що допомагає знизити ризики та витрати на виправлення дефектів.

А.1 Системні тести

Розроблені тестові звіти (Test Report):

Рисунок А.1.1 – Системний тест-кейс «Реєструвати нових користувачів в систему»



Test Report 1 April, 2024

Забронювати вільний номер

UseCase in package 'Structured Use Case Model'

Забронювати вільний номер
Version 1.0 Phase 1.0 Proposed
user created on 29.03.2024. Last modified 29.03.2024

TESTING

Standard System test. Резервування номеру

Description
1. Обрати опцію "Номери" у меню.
2. Вибрати вільний номер для резервування.
3. Ввести дату заїзду та виїзду.
4. Ввести дані клієнта (ім'я, прізвище, контактну інформацію).
5. Натиснути кнопку "Забронювати".
6. Перевірити, що вибраний номер зараз зайнятий.
7. Перевірити, що дані резервування коректно відображаються у списку бронювань.

Input
Номер кімнати: 101
Дата заїзду: 2024-04-15
Дата виїзду: 2024-04-20
Ім'я клієнта: Кутуб
Прізвище клієнта: Olifirenko
Контактна інформація клієнта: kirill082003@gmail.com

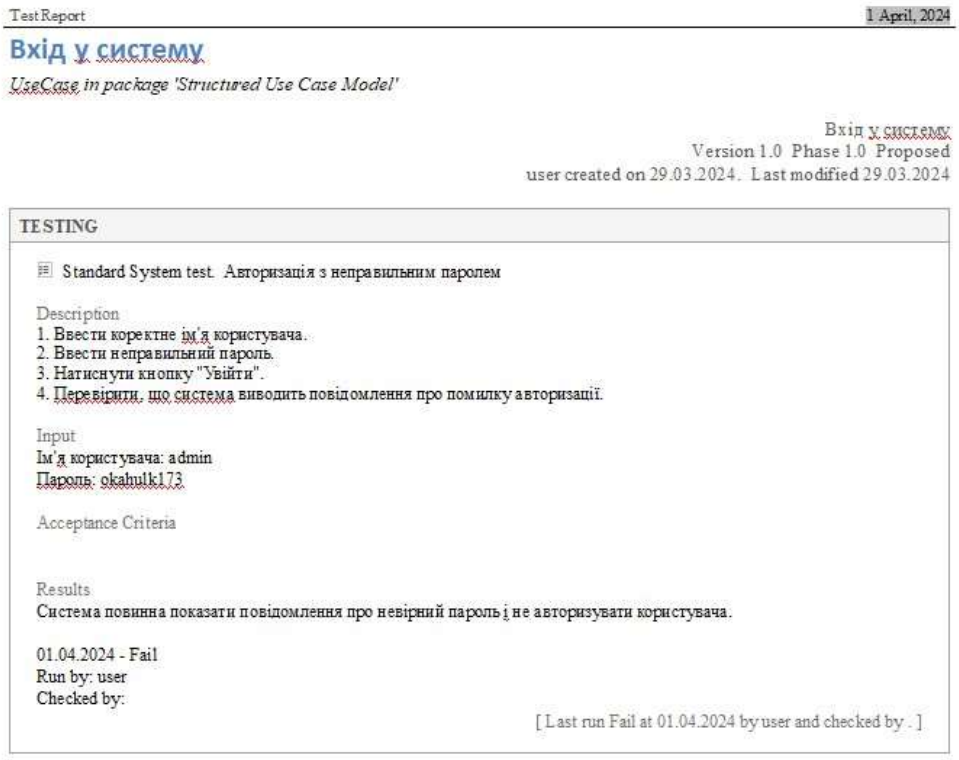
Acceptance Criteria

Results
Вибраний номер успішно заброньований для клієнта з введеними даними.

01.04.2024 - Pass
Run by: user
Checked by: user

[Last run Pass at 01.04.2024 by user and checked by user.]

Рисунок А.1.2 – Системний тест-кейс «Забронювати вільний номер»



Test Report 1 April, 2024

Вхід у систему

UseCase in package 'Structured Use Case Model'

Вхід у систему
Version 1.0 Phase 1.0 Proposed
user created on 29.03.2024. Last modified 29.03.2024

TESTING

Standard System test. Авторизація з неправильним паролем

Description
1. Ввести коректне ім'я користувача.
2. Ввести неправильний пароль.
3. Натиснути кнопку "Увійти".
4. Перевірити, що система виводить повідомлення про помилку авторизації.

Input
Ім'я користувача: admin
Пароль: okahulk173

Acceptance Criteria

Results
Система повинна показати повідомлення про невірний пароль і не авторизувати користувача.

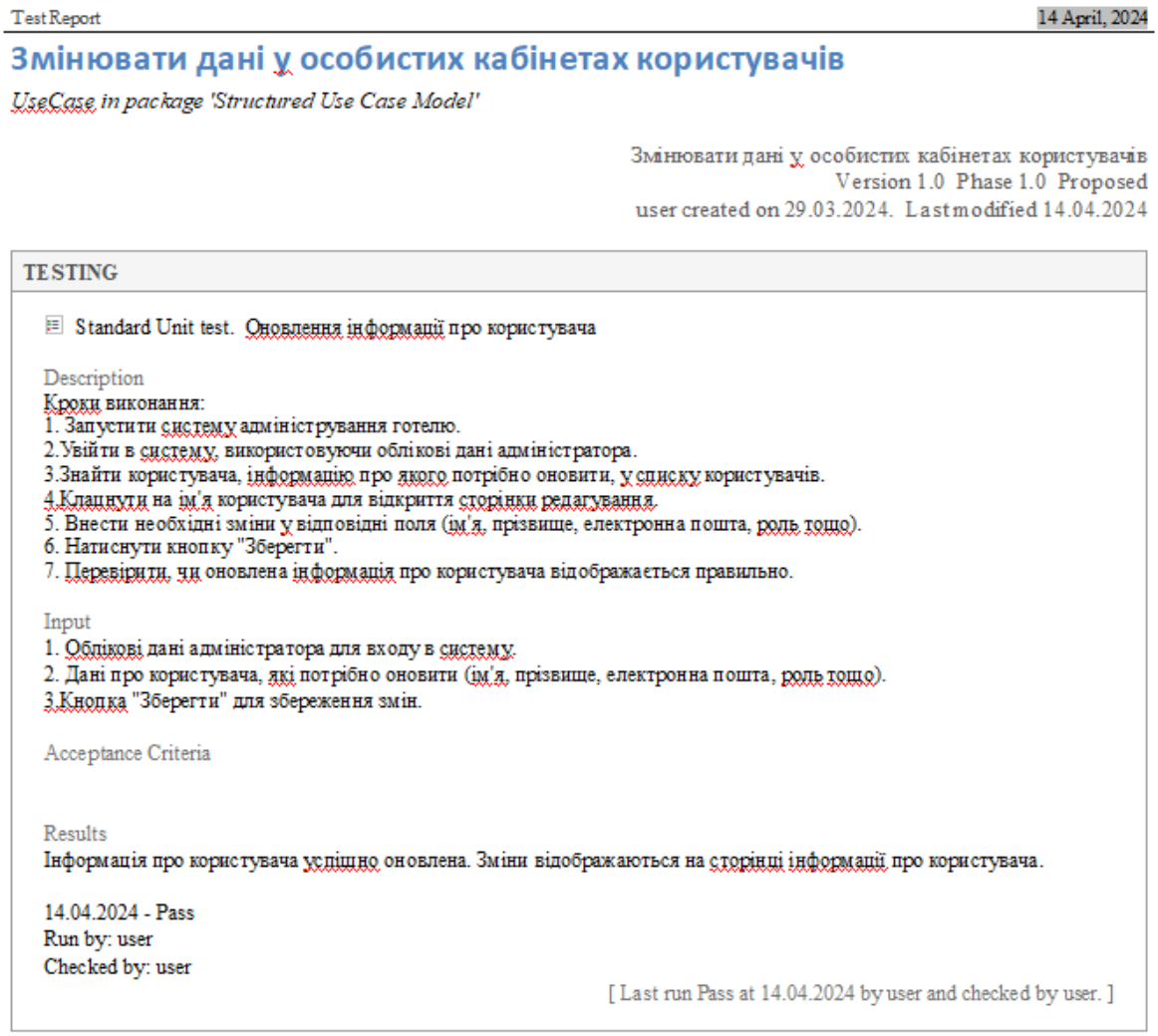
01.04.2024 - Fail
Run by: user
Checked by:

[Last run Fail at 01.04.2024 by user and checked by .]

Рисунок А.1.3 – Системний тест-кейс «Вхід в систему»

А.2 Модульні тести

Розроблені тестові звіти (Test Report):



Test Report 14 April, 2024

Змінювати дані у особистих кабінетах користувачів

UseCase in package 'Structured Use Case Model'

Змінювати дані у особистих кабінетах користувачів
Version 1.0 Phase 1.0 Proposed
user created on 29.03.2024. Last modified 14.04.2024

TESTING

Standard Unit test. Оновлення інформації про користувача

Description
Кроки виконання:
1. Запустити систему адміністрування готелю.
2.Увійти в систему, використовуючи облікові дані адміністратора.
3.Знайти користувача, інформацію про якого потрібно оновити, у списку користувачів.
4.Клацнути на ім'я користувача для відкриття сторінки редагування.
5. Внести необхідні зміни у відповідні поля (ім'я, прізвище, електронна пошта, роль тощо).
6. Натиснути кнопку "Зберегти".
7. Перевірити, чи оновлена інформація про користувача відображається правильно.

Input
1. Облікові дані адміністратора для входу в систему.
2. Дані про користувача, які потрібно оновити (ім'я, прізвище, електронна пошта, роль тощо).
3.Кнопка "Зберегти" для збереження змін.

Acceptance Criteria

Results
Інформація про користувача успішно оновлена. Зміни відображаються на сторінці інформації про користувача.

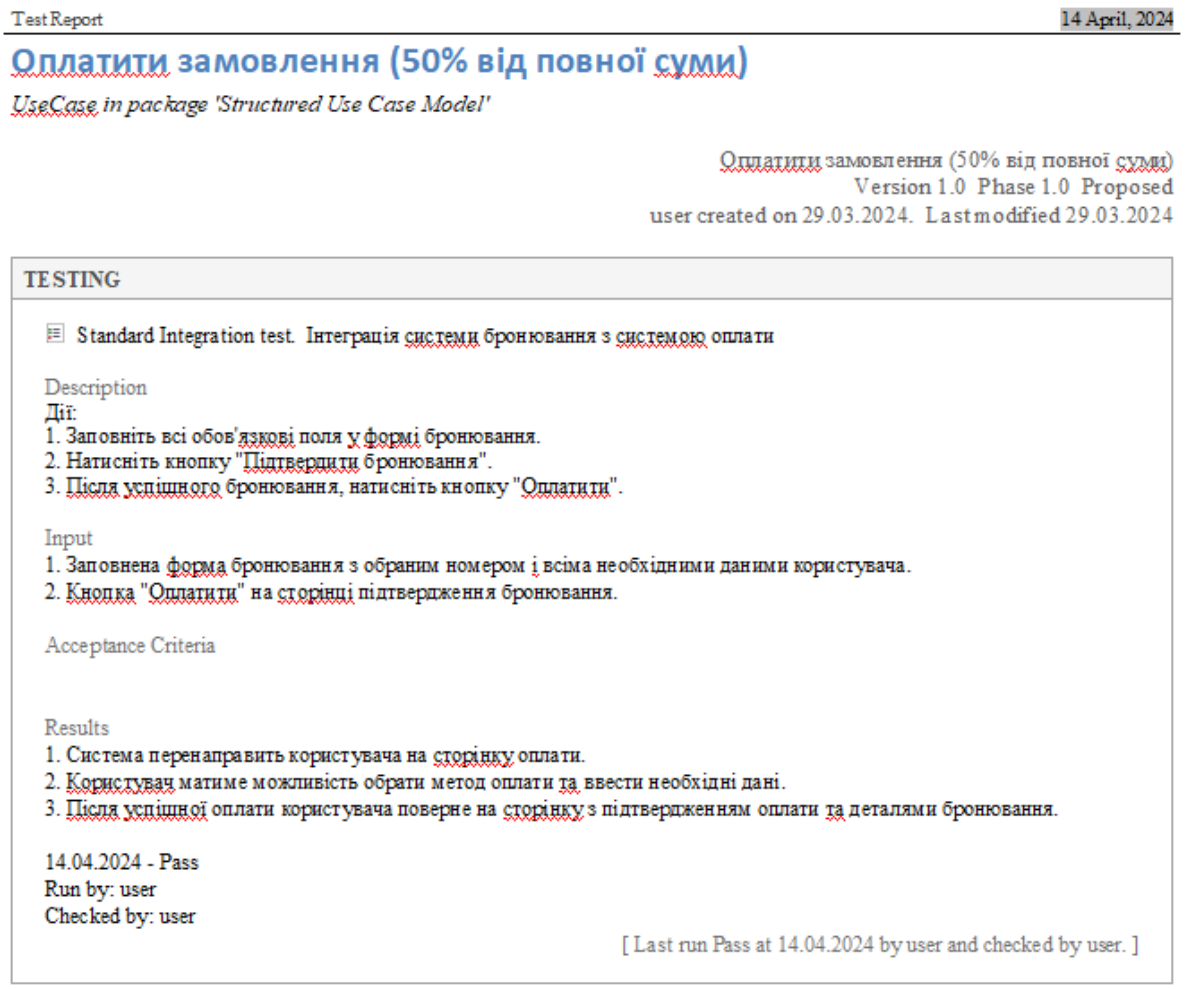
14.04.2024 - Pass
Run by: user
Checked by: user

[Last run Pass at 14.04.2024 by user and checked by user.]

Рисунок А.2.1 – Модульний тест-кейс «Оновлення інформації про користувача за допомогою адмін панелі»

А.3 Інтеграційні тести

Розроблені тестові звіти (Test Report):



Test Report 14 April, 2024

Оплатити замовлення (50% від повної суми)

UseCase in package 'Structured Use Case Model'

Оплатити замовлення (50% від повної суми)
Version 1.0 Phase 1.0 Proposed
user created on 29.03.2024. Last modified 29.03.2024

TESTING

Standard Integration test. Інтеграція системи бронювання з системою оплати

Description
Дії:
1. Заповніть всі обов'язкові поля у формі бронювання.
2. Натисніть кнопку "Підтвердити бронювання".
3. Після успішного бронювання, натисніть кнопку "Оплатити".

Input
1. Заповнена форма бронювання з обраним номером і всіма необхідними даними користувача.
2. Кнопка "Оплатити" на сторінці підтвердження бронювання.

Acceptance Criteria

Results
1. Система перенаправить користувача на сторінку оплати.
2. Користувач матиме можливість обрати метод оплати та ввести необхідні дані.
3. Після успішної оплати користувача поверне на сторінку з підтвердженням оплати та деталями бронювання.

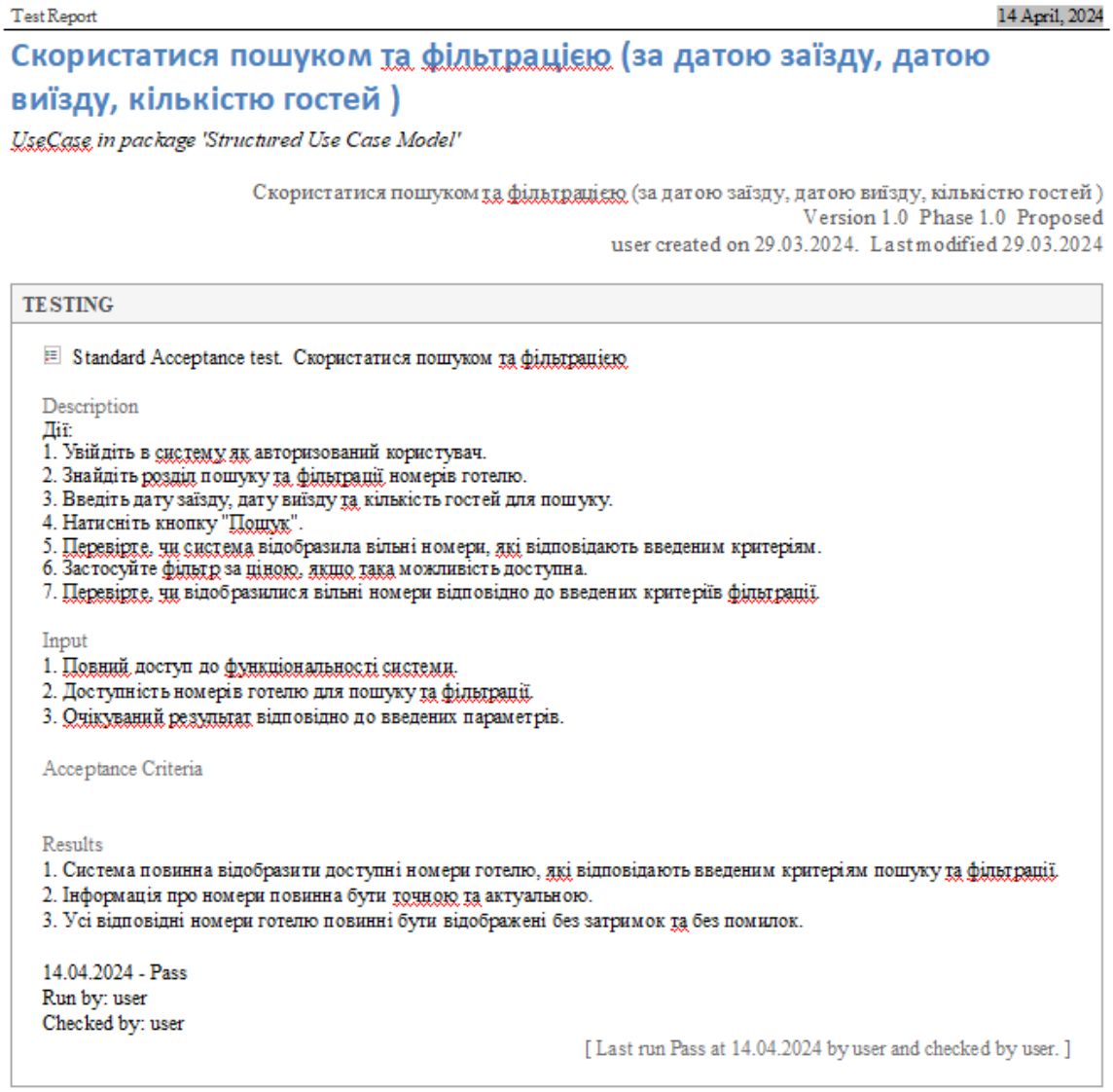
14.04.2024 - Pass
Run by: user
Checked by: user

[Last run Pass at 14.04.2024 by user and checked by user.]

Рисунок А.3.1 – Інтеграційні тест-кейс «Інтеграція системи бронювання з системою оплати»

А.4 Приймальні тести

Розроблені тестові звіти (Test Report):



Test Report 14 April, 2024

Скористатися пошуком та фільтрацією (за датою заїзду, датою виїзду, кількістю гостей)

UseCase in package 'Structured Use Case Model'

Скористатися пошуком та фільтрацією (за датою заїзду, датою виїзду, кількістю гостей)
Version 1.0 Phase 1.0 Proposed
user created on 29.03.2024. Last modified 29.03.2024

TESTING

Standard Acceptance test. Скористатися пошуком та фільтрацією

Description
Дії:
1. Увійдіть в систему як авторизований користувач.
2. Знайдіть розділ пошуку та фільтрації номерів готелю.
3. Введіть дату заїзду, дату виїзду та кількість гостей для пошуку.
4. Натисніть кнопку "Пошук".
5. Перевірте, чи система відобразила вільні номери, які відповідають введеним критеріям.
6. Застосуйте фільтр за ціною, якщо така можливість доступна.
7. Перевірте, чи відобразилися вільні номери відповідно до введених критеріїв фільтрації.

Input
1. Повний доступ до функціональності системи.
2. Доступність номерів готелю для пошуку та фільтрації.
3. Очікуваний результат відповідно до введених параметрів.

Acceptance Criteria

Results
1. Система повинна відобразити доступні номери готелю, які відповідають введеним критеріям пошуку та фільтрації.
2. Інформація про номери повинна бути точною та актуальною.
3. Усі відповідні номери готелю повинні бути відображені без затримок та без помилок.

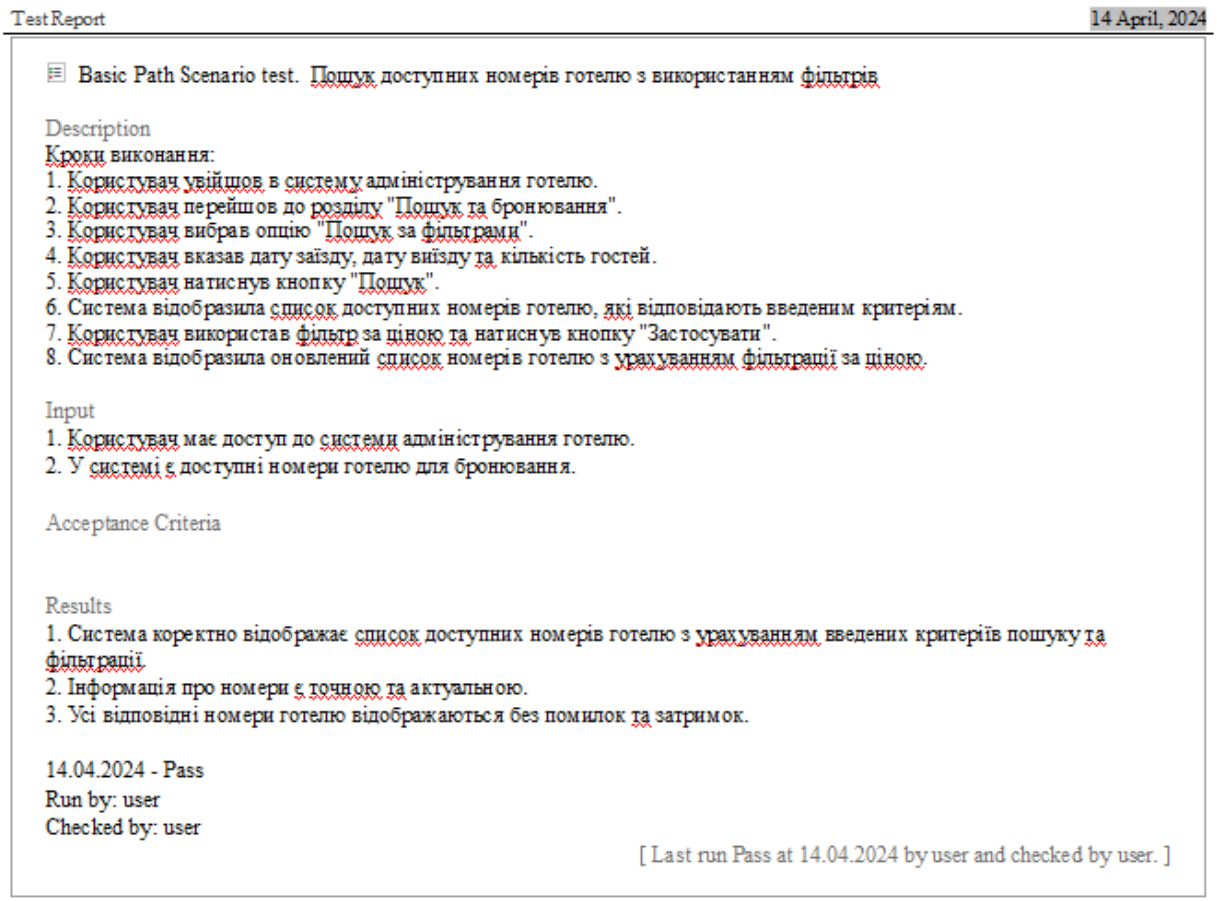
14.04.2024 - Pass
Run by: user
Checked by: user

[Last run Pass at 14.04.2024 by user and checked by user.]

Рисунок А.4.1 – Приймальний тест-кейс «Скористатися пошуком та фільтрацією»

А.5 Тести-сценарії

Розроблені тестові звіти (Test Report):



Test Report 14 April, 2024

Basic Path Scenario test. Пошук доступних номерів готелю з використанням фільтрів

Description
Кроки виконання:
1. Користувач увійшов в систему адміністрування готелю.
2. Користувач перейшов до розділу "Пошук та бронювання".
3. Користувач вибрав опцію "Пошук за фільтрами".
4. Користувач вказав дату заїзду, дату виїзду та кількість гостей.
5. Користувач натиснув кнопку "Пошук".
6. Система відобразила список доступних номерів готелю, які відповідають введеним критеріям.
7. Користувач використав фільтр за ціною та натиснув кнопку "Застосувати".
8. Система відобразила оновлений список номерів готелю з урахуванням фільтрації за ціною.

Input
1. Користувач має доступ до системи адміністрування готелю.
2. У системі є доступні номери готелю для бронювання.

Acceptance Criteria

Results
1. Система коректно відображає список доступних номерів готелю з урахуванням введених критеріїв пошуку та фільтрації.
2. Інформація про номери є точною та актуальною.
3. Усі відповідні номери готелю відображаються без помилок та затримок.

14.04.2024 - Pass
Run by: user
Checked by: user

[Last run Pass at 14.04.2024 by user and checked by user.]

Рисунок А.5.1 – Тести-сценарії «Пошук доступних номерів готелю з використанням фільтрів»

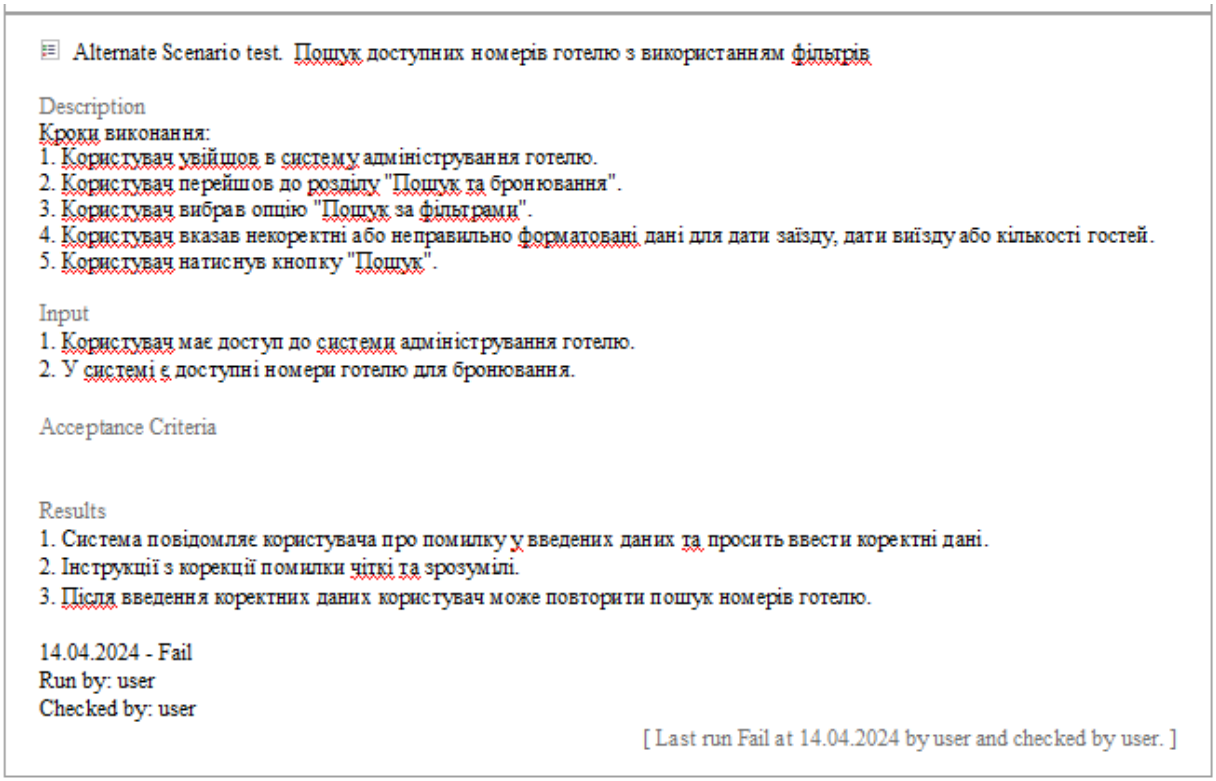


Рисунок А.5.2 – Тести-сценарії (Альтернативний сценарій) «Пошук доступних номерів готелю з використанням фільтрів (негативний випадок)»

ДОДАТОК Б

Фрагменти коду, що було створено для реалізації бекенду системи (контролери)

//Room Controller

```
@Controller
@RequestMapping
public class RoomController {
    @Autowired
    private RoomService roomService;
    @Autowired
    private CountOfRoomViewRepository countOfRoomViewRepository;
    @GetMapping({"/roomDetails"})
    public String showRoomDetails(Model model) {
        return "roomDetails.html";
    }
    @RequestMapping(value="/reservationList.html", params = {"rid"}, method = RequestMethod.GET)
    public String showRoomDetails(Model model, long rid){
        Room r = roomService.getRoom(rid);
        model.addAttribute("room", r);
        return "roomDetails";
    }
    @RequestMapping(value="/roomList.html", method = {RequestMethod.GET})
    public String showReservationList(Model model,Pageable pageable){
        model.addAttribute("roomListPage", roomService.getAllRooms2(pageable));
        return "roomList.html";
    }
    @GetMapping({"/countOfRoom"})
    public String countOfRoom(Model model) {
        List<CountOfRoomView> countOfRoomViews = countOfRoomViewRepository.findAll();
        model.addAttribute("countOfRoomViews", countOfRoomViews);
        System.out.println(countOfRoomViews);
        return "countOfRoomCertainType.html";
    }
}
```

//RoomReservationFromController

```
@Controller
@RequestMapping
@SessionAttributes({"roomReservation","roomListPage"})
```

```
public class RoomReservationFormController {
    private RoomReservationService roomReservationService;
    @Autowired
    RoomService roomService;
    @Autowired
    UserService userService;
    @Autowired
    public RoomReservationFormController(RoomReservationService roomReservationService)
    {
        this.roomReservationService = roomReservationService;
    }
    @RequestMapping(value="/reservationFormUSR.html", method= RequestMethod.GET)
    public String showFormUSR(Model model, Optional<Long> id){
        model.addAttribute("roomReservation",
                id.isPresent()?
                        roomReservationService.getRoomReservation(id.get()):
                        new RoomReservation());
        return "reservationFormUSR.html";
    }
    @RequestMapping(value="/reservationFormADM.html", method= RequestMethod.GET)
    public String showFormADM(Model model, Optional<Long> id){
        model.addAttribute("roomReservation",
                id.isPresent()?
                        roomReservationService.getRoomReservation(id.get()):
                        new RoomReservation());
        return "reservationFormADM.html";
    }
    @RequestMapping(value="/editReservationFormADM.html", method= RequestMethod.GET)
    public String showEditFormADM(Model model, Optional<Long> id){
        model.addAttribute("roomReservation",
                id.isPresent()?
                        roomReservationService.getRoomReservation(id.get()):
                        new RoomReservation());
        return "editReservationFormADM.html";
    }
    @RequestMapping(value="/availableRooms.html", method= RequestMethod.GET)
    public String showFormUSR2(Model model, @ModelAttribute("roomReservation") RoomReservation v, @ModelAttribute("roomListPage")
Page<Room> r){
        model.addAttribute("roomListPage", r);
        model.addAttribute("roomReservation", v);
        return "availableRooms.html";
```

```
    }


    @RequestMapping(value="/reservationFormUSR.html", method= RequestMethod.POST)
    public String processFormUSR(Model model, @Valid @ModelAttribute("roomReservation") RoomReservation v,  Pageable pageable, RoomFilter search, Principal principal, BindingResult errors){
        if(errors.hasErrors()){
            return "reservationFormUSR";
        }
        search.setNumberOfPeople(v.getNumberOfPeople());
        search.setReservationStartDate(v.getReservationStartDate());
        search.setReservationEndDate(v.getReservationEndDate());

        v.setUser(userService.getUserByLogin(principal.getName()));
        model.addAttribute("roomListPage", roomService.getAllRooms2(pageable));
        model.addAttribute("roomReservation", v);
        return "redirect:/availableRooms.html";
    }
    @RequestMapping(value="/reservationFormADM.html", method= RequestMethod.POST)
    public String processFormADM(Model model, @Valid @ModelAttribute("roomReservation") RoomReservation v,  Pageable pageable, RoomFilter search, Principal principal, BindingResult errors){
        if(errors.hasErrors()){
            return "reservationFormADM";
        }
User usertemp=userService.getUserByLogin(v.getUser().getLogin());
        v.setUser(usertemp);
        search.setNumberOfPeople(v.getNumberOfPeople());
        search.setReservationStartDate(v.getReservationStartDate());
        search.setReservationEndDate(v.getReservationEndDate());
        model.addAttribute("roomListPage", roomService.getAllRooms2(pageable));
        model.addAttribute("roomReservation", v);
        return "redirect:/availableRooms.html";
    }
    @RequestMapping(value="/editReservationFormADM.html", method= RequestMethod.POST)
    public String processEditFormADM(@Valid @ModelAttribute("roomReservation") RoomReservation v, BindingResult errors){
        roomReservationService.saveRoomReservation(v);
        return "redirect:/reservationList.html";
    }
    @RequestMapping(value="/availableRooms.html", method= RequestMethod.POST)
    public String processFormUSR2(@Valid @ModelAttribute("roomReservation") RoomReservation v, @ModelAttribute("roomListPage") Page<Room> r, Principal principal){
        v.setPaid(false);
        v.setVerified(false);
```

```
        roomReservationService.saveRoomReservation(v);
        Long idRoom=v.getRoom().getId();
       Room rr=roomService.getRoom(idRoom);
        List<RoomReservation> roomReservations=rr.getRoomReservations();
        roomReservations.add(v);
        rr.setRoomReservations(roomReservations);
        if(principal.getName().equals("admin")){
            return "redirect:reservationList.html";
        }
        else {
            return "redirect:yourReservationList.html";
        }
    }
    @InitBinder
    public void initBinder(WebDataBinder binder) {//Ми реєструємо редакторів власності
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        dateFormat.setLenient(false);
        CustomDateEditor dateEditor = new CustomDateEditor(dateFormat, false);
        binder.registerCustomEditor(Date.class, "reservationStartDate", dateEditor);
        binder.registerCustomEditor(Date.class, "reservationEndDate", dateEditor);
        binder.registerCustomEditor(Date.class, "checkInDate", dateEditor);
        binder.registerCustomEditor(Date.class, "checkOutDate", dateEditor);
        DecimalFormat numberFormat = new DecimalFormat("#0.00");
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);
        numberFormat.setGroupingUsed(false);
        binder.registerCustomEditor(Float.class, "price", new CustomNumberEditor(Float.class, numberFormat, false));
    }
}
```

//RoomReservationFromController

```
@Controller
@RequestMapping
@SessionAttributes({"roomReservation","roomListPage"})
public class RoomReservationFormController {
    private RoomReservationService roomReservationService;
    @Autowired
    RoomService roomService;
    @Autowired
    UserService userService;
    @Autowired
```

```
public RoomReservationFormController(RoomReservationService roomReservationService)
{
    this.roomReservationService = roomReservationService;
}
@RequestMapping(value="/reservationFormUSR.html", method= RequestMethod.GET)
public String showFormUSR(Model model, Optional<Long> id){
    model.addAttribute("roomReservation",
            id.isPresent()?
                    roomReservationService.getRoomReservation(id.get()):
                    new RoomReservation());
    return "reservationFormUSR.html";
}
@RequestMapping(value="/reservationFormADM.html", method= RequestMethod.GET)
public String showFormADM(Model model, Optional<Long> id){

    model.addAttribute("roomReservation",
            id.isPresent()?
                    roomReservationService.getRoomReservation(id.get()):
                    new RoomReservation());
    return "reservationFormADM.html";
}
@RequestMapping(value="/editReservationFormADM.html", method= RequestMethod.GET)
public String showEditFormADM(Model model, Optional<Long> id){
    model.addAttribute("roomReservation",
            id.isPresent()?
                    roomReservationService.getRoomReservation(id.get()):
                    new RoomReservation());
    return "editReservationFormADM.html";
}
@RequestMapping(value="/availableRooms.html", method= RequestMethod.GET)
public String showFormUSR2(Model model, @ModelAttribute("roomReservation") RoomReservation v, @ModelAttribute("roomListPage")
Page<Room> r){
    model.addAttribute("roomListPage", r);
    model.addAttribute("roomReservation", v);
    return "availableRooms.html";
}
@RequestMapping(value="/reservationFormUSR.html", method= RequestMethod.POST)
public String processFormUSR(Model model, @Valid @ModelAttribute("roomReservation") RoomReservation v,  Pageable pageable,
RoomFilter search, Principal principal, BindingResult errors)
   if(errors.hasErrors()){
       return "reservationFormUSR";
```

```
        }
        search.setNumberOfPeople(v.getNumberOfPeople());
        search.setReservationStartDate(v.getReservationStartDate());
        search.setReservationEndDate(v.getReservationEndDate());
        v.setUser(userService.getUserByLogin(principal.getName()));
        model.addAttribute("roomListPage", roomService.getAllRooms2(pageable));
        model.addAttribute("roomReservation", v);
        return "redirect:/availableRooms.html";
    }
    @RequestMapping(value="/reservationFormADM.html", method= RequestMethod.POST)
    public String processFormADM(Model model, @Valid @ModelAttribute("roomReservation") RoomReservation v,  Pageable pageable,
RoomFilter search, Principal principal, BindingResult errors){
        if(errors.hasErrors()){
            return "reservationFormADM";
        }
User usertemp=userService.getUserByLogin(v.getUser().getLogin());
        v.setUser(usertemp);
        search.setNumberOfPeople(v.getNumberOfPeople());
        search.setReservationStartDate(v.getReservationStartDate());
        search.setReservationEndDate(v.getReservationEndDate());
        model.addAttribute("roomListPage", roomService.getAllRooms2(pageable));
        model.addAttribute("roomReservation", v);
        return "redirect:/availableRooms.html";
    }
    @RequestMapping(value="/editReservationFormADM.html", method= RequestMethod.POST)
    public String processEditFormADM(@Valid @ModelAttribute("roomReservation") RoomReservation v, BindingResult errors){
        roomReservationService.saveRoomReservation(v);
        return "redirect:/reservationList.html";
    }
    @RequestMapping(value="/availableRooms.html", method= RequestMethod.POST)
    public String processFormUSR2(@Valid @ModelAttribute("roomReservation") RoomReservation v, @ModelAttribute("roomListPage")
Page<Room> r, Principal principal){
        v.setPaid(false);
        v.setVerified(false);
        roomReservationService.saveRoomReservation(v);
        Long idRoom=v.getRoom().getId();
       Room rr=roomService.getRoom(idRoom);
        List<RoomReservation> roomReservations=rr.getRoomReservations();
        roomReservations.add(v);
        rr.setRoomReservations(roomReservations);
        if(principal.getName().equals("admin")){
```

```
            return "redirect:reservationList.html";
        }
        else {
            return "redirect:yourReservationList.html";
        }
    }
    @InitBinder
    public void initBinder(WebDataBinder binder) {//Ми реєструємо редакторів власності
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        dateFormat.setLenient(false);
        CustomDateEditor dateEditor = new CustomDateEditor(dateFormat, false);
        binder.registerCustomEditor(Date.class, "reservationStartDate", dateEditor);
        binder.registerCustomEditor(Date.class, "reservationEndDate", dateEditor);
        binder.registerCustomEditor(Date.class, "checkInDate", dateEditor);
        binder.registerCustomEditor(Date.class, "checkOutDate", dateEditor);
        DecimalFormat numberFormat = new DecimalFormat("#0.00");
        numberFormat.setMaximumFractionDigits(2);
        numberFormat.setMinimumFractionDigits(2);
        numberFormat.setGroupingUsed(false);
        binder.registerCustomEditor(Float.class, "price", new CustomNumberEditor(Float.class, numberFormat, false));
    }
}
// User Controller
Controller
@RequestMapping
@SessionAttributes("userForm")
public class UserController {
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
  private EmailService emailService;

    private final UserService userService;
    private final SecurityService securityService;
    public UserController(UserService userService, SecurityService securityService) {
        this.userService = userService;
        this.securityService = securityService;
    }
    @GetMapping("/registration")
    public String registration(Model model) {
```

```
    model.addAttribute("userForm", new User());
    return "registration.html";
}
@Secured("ROLE_ADMIN")
@GetMapping("/addUser")
public String addUser(Model model) {

    model.addAttribute("userForm", new User());
    return "addUser.html";
}
@Secured("ROLE_ADMIN")
@RequestMapping(value="/editUser.html", method= RequestMethod.GET)
public String showFormUSR(Model model, Optional<Long> id){
    model.addAttribute("userForm",
            id.isPresent()?
                    userService.getUser(id.get()):
                    new User());
    System.out.println("login get: "+userService.getUser(id.get()).getLogin());
    return "editUser.html";
}
@Secured("ROLE_ADMIN")
@PostMapping("/editUser.html")
public String editUser(@ModelAttribute("userForm") User userForm){
    System.out.println("login post: "+userForm.getLogin());
    userService.save(userForm);
    return "redirect:userList.html";
}
@Secured("ROLE_ADMIN")
@PostMapping("/addUser")
public String addUser(@ModelAttribute("userForm") User userForm, BindingResult bindingResult){
    userForm.setPassword(passwordEncoder.encode("test"));
    userService.save(userForm);
    return "welcome.html";
}
@PostMapping("/registration")
public String registration(@ModelAttribute("userForm") User userForm, BindingResult bindingResult) {
    userService.save(userForm);
    return "welcome.html";
}
@PostMapping("/login")
```

```
public String login(Model model, String error, String logout) {
    if (error != null)
        model.addAttribute("error", "Your username and password is invalid.");
    if (logout != null)
        model.addAttribute("message", "You have been logged out successfully.");
   return "welcome.html";
   return "login.html";
}
@GetMapping("/login")
public String login() {
    return "login.html";
}
@GetMapping({"/", "/welcome"})
public String welcome(Model model) {
    return "welcome.html";
}
@GetMapping({"/contact"})
public String contact(Model model) {
    return "contact.html";
}
@GetMapping({"/userDetails"})
public String showUserDetails(Model model) {
    return "userDetails.html";
}
@GetMapping({"/accountDetails"})
    public String showAccountDetails(Model model) {
    return "accountDetails.html";
}
@RequestMapping(value="/reservationList.html", params = {"uid"}, method = RequestMethod.GET)
public String showUserDetails(Model model, long uid){
    User u = userService.getUser(uid);
    model.addAttribute("user", u);
    return "userDetails";
}
//info about logged in user
@RequestMapping(value="/accountDetails.html", method = RequestMethod.GET)
public String showLoggedInUAccountDetails(Model model, Principal principal){
    model.addAttribute("user", userService.getUserByLogin(principal.getName()));
    System.out.println(userService.getUserByLogin(principal.getName()).getLogin());
    return "accountDetails";
```

```
    }
    @GetMapping(value="/userList.html", params = {"all"})
    public String resetUserList(@ModelAttribute("searchCommand") UserFilter search){
        search.clear();
        return "redirect:userList.html";
    }
    @RequestMapping(value="/userList.html", method = {RequestMethod.GET})
    public String showUserList(Model model, Pageable pageable, @Valid @ModelAttribute("searchCommand") UserFilter search){
        model.addAttribute("userListPage", userService.getAllUsers(search, pageable));
        return "userList";
    }
    @RequestMapping(value="/userList.html", method = {RequestMethod.POST})
    public String showUserList2(Model model, Pageable pageable, @Valid @ModelAttribute("searchCommand") UserFilter search){
        model.addAttribute("userListPage", userService.getAllUsers(search, pageable));
        System.out.println("fraza "+search.getPhrase());
        return "userList";
        // return "redirect:reservationList";
    }
    @PostMapping("/changePassword.html")
    public String changePassword(@ModelAttribute("userForm") User userForm, Principal principal) {
        return "accountDetails.html";
    }
    @GetMapping("/changePassword.html")
    public String changePassword(Model model) {
        model.addAttribute("userForm", new User());
        return "redirect:/accountDetails.html";
    }
    @RequestMapping(value="/userList.html", params = "id", method = RequestMethod.GET)
    public String deleteUser(long id, HttpServletRequest request){
        userService.deleteUser(id);
        return "deleteInfo";
    }
}
```

HTML код сторінки для бронювання номеру

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Hotel</title>
    <div th:replace="shared/header :: header-css"/>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
</head>
<body>
<div th:replace="shared/header :: header('reservationFormUSR')"/>
<div class="container-fluid text-center">
    <div class="row content">
        <div class="col-sm-2 sidenav">
            <div sec:authorize="isAuthenticated()">
                <p>Бронювання номерів</p>
            </div>
        </div>
        <div class="col-sm-8 text-left">
            <form th:method="POST" th:object="${roomReservation}" th:action="@{/reservationFormUSR.html}">
                <fieldset>
                    <div class="form-group">
                        <label for="reservationStartDate" class="bmd-label-floating">Початок перебування:</label>
                        <input id="reservationStartDate" type="date" th:field="*{reservationStartDate}"
                               class="form-control" th:classappend="${#fields.hasErrors('reservationStartDate')}?'is-invalid'"
                               required="true"/>
                        <div class="error text-danger" th:if="${#fields.hasErrors('reservationStartDate')}">
                            <p th:each="err : ${#fields.errors('reservationStartDate')}" th:text="${err}"></p>
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="reservationEndDate" class="bmd-label-floating">Кінець перебування:</label>
                        <input id="reservationEndDate" type="date" th:field="*{reservationEndDate}"
                               class="form-control" th:classappend="${#fields.hasErrors('reservationEndDate')}?'is-invalid'"
                               required="true"/>
                        <div class="error text-danger" th:if="${#fields.hasErrors('reservationEndDate')}">
                            <p th:each="err : ${#fields.errors('reservationEndDate')}" th:text="${err}"></p>
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="numberOfPeople" class="bmd-label-floating">Кількість людей:</label>
                        <input id="numberOfPeople" type="text" th:field="*{numberOfPeople}"
                               class="form-control" th:classappend="${#fields.hasErrors('numberOfPeople')}?'is-invalid'"
```

```
                    required="true"/>
                <div class="error text-danger" th:if="${#fields.hasErrors('numberOfPeople')}">
                  <p th:each="err : ${#fields.errors('numberOfPeople')}" th:text="${err}"></p>
                </div>
              </div>
              <input type="submit" class="btn btn-lg btn-primary btn-block" value="Далі"/>
            </fieldset>
          </form>
    <br>
          <div sec:authorize="hasRole('ADMIN')">
            <a class="btn btn-lg btn-block" href="reservationList.html">Скасувати</a>
          </div>
          <div sec:authorize="hasRole('USER')">
            <a class="btn btn-lg  btn-block" href="yourReservationList.html">Скасувати</a>
          </div>
        </div>
        <div class="col-sm-2 sidenav">
          <div class="well">
            <label style="color:darkred; margin-top: 20px;" sec:authorize="isAuthenticated()">
              Вітаю <span sec:authentication="name"/>!
            </label>
          </div>
          <div class="well">
            <a sec:authorize="isAuthenticated()" class="btn btn-success" href="accountDetails.html">Ваш рахунок</a>
          </div>
        </div>
      </div>
    </div>
    </body>
    </html>
```

Фрагменти CSS коду

```
section {
  display: block;
}

body {
  line-height: 1;
  font-family: 'Montserrat', sans-serif;
}

ol,
ul {
  list-style: none;
}

blockquote,
q {
  quotes: none;
}

blockquote:before,

  -ms-flex-align: center;
  align-items: center;
}

nav.nav-wrapper ul li {
```

```
    display: inline-block;
    padding: 0 10px;
    margin-bottom: 0;
}

nav.nav-wrapper ul li a {
    font-size: 17px;
    line-height: 22px;
    text-decoration: none;
    color: white;
    cursor: pointer;
    border-bottom: 1px solid transparent;
    padding-bottom: 3px;
    -webkit-transition: 0.3s;
    transition: 0.3s;
}

nav.nav-wrapper ul li a:hover {
    border-bottom: 1px solid #ffffff;
}

header .logo img {
    max-width: 100px;
}

/* PARALLAX PAGE STYLES START */
.cover-main {
    background: linear-gradient(rgba(0, 0, 0, 0.75),
rgba(0, 0, 0, 0.75)),
        url('../../images/header/wallpaper.jpg') no-repeat
center;
    background-size: cover;
    height: 75vh;
    width: 100%;
}

ul#scene li.name {
    text-align: center;
    font-size: 40px;
    line-height: 45px;
    text-transform: uppercase;
    color: white;
    width: 100%;
}

ul#scene li.title {
    text-align: center;
    font-size: 30px;
    line-height: 35px;
    color: white;
    width: 100%;
    margin-top: 60px;
}

ul#scene {
    margin-top: 100px;
}

.btn {
    width: 100%;
    text-align: center;
    margin-top: 150px;
}

.reservation-btn,
.logout-btn,
.reservations-btn {
    margin: 0 auto;
    display: inline-block;
    padding: 10px 20px;
    border: 1px solid white;
    background: transparent;
    color: white;
    text-decoration: none;
    -webkit-transition: 0.3s;
    transition: 0.3s;
}

.reservations-btn:hover {
    background: rgb(221, 219, 219);
    color: rgb(0, 0, 0);
}

.reservation-btn:hover {
    background: rgb(7, 49, 7);
    color: white;
}

.logout-btn:hover {
    background: rgb(129, 5, 5);
    color: white;
}

.reservation-btn {
    margin-right: 30px;
}

.reservations-btn {
    margin-left: 30px;
}

.logout-btn {
    margin-top: 30px;
}

/* PARALLAX PAGE STYLES END */

/* Rooms START */
.section-padding {
    position: relative;
}
```

```
h2.sub-title {
    position: relative;
    display: block;
    margin: 70px auto;
    text-align: center;
    font-size: 35px;
    line-height: 40px;
}

h2.sub-title::after {
    display: block;
    height: 1px;
    background-color: #000000;
    content: "";
    width: 7%;
    margin: 30px auto 0 auto;
}

.img-btm {
    margin-bottom: 20px;
}

.img-btm>div>img {
    margin-left: 45px;
}

.card {
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.6), 0 1px 2px
rgba(0, 0, 0, 0.6);
    transition: all 0.3s cubic-bezier(.25, .8, .25, 1);
}

.card:hover {
    box-shadow: 0 22px 30px rgba(0, 0, 0, 0.25), 0 15px
15px rgba(0, 0, 0, 0.22);
}

.slide-item {
    background: #ffffff;
    overflow: hidden;
    width: 32%;
    height: 62vh;
    margin-bottom: 100px;
    vertical-align: top;
    float: left;
    -webkit-margin-start: 200px;
}

.slide-item h3 {
    text-align: justify;
    font-size: 22px;
    padding-top: 10px;
    padding-left: 10px;
    color: rgb(0, 0, 0);
}

.slide-item:hover {
    background-color: rgb(0, 0, 0);
}

.slide-item P {
    text-align: center;
    font-size: 20px;
    margin-top: 3px;
    color: white;
}

.slide-item img {
    position: center;
    padding: 20px;
    height: 340px;
    color: white;
    filter: grayscale(70%);
}

.slide-item img:hover {
    filter: grayscale(0%);
}

/* Rooms END */

/* OUR FACILITIES AND SERVICES START */
section.services-sec {
    background: linear-gradient(rgba(0, 0, 0, 0.75),
rgba(0, 0, 0, 0.75)),
        url('../../images/service/Menu.jpg') no-repeat right
bottom/cover;
    text-align: center;
    clear: both;
    height: 100vh;
    padding: 50px 10px;
    color: white;
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-pack: center;
    -ms-flex-pack: center;
    justify-content: center;
    -webkit-box-align: center;
    -ms-flex-align: center;
    align-items: center;
    -ms-flex-item-align: center;
    align-self: center;
}

section.services-sec span {
    font-size: 100px;
    line-height: 105px;
}

section.services-sec h2 {
```

```
    margin-bottom: 30px;
}

section.services-sec h2::after {

    background: white;
}

.tm-service {
    max-width: 30%;
    width: 30%;
    margin: 1.5% 12% 1.5% 10%;
}

.tm-service-section {
    display: flex;
}

.tm-ser-image {
    display: inline-block;
    max-width: 25%;
    width: 20%;
    height: auto;
    padding-bottom: 10%;
}

.tm-service div {
    display: inline-block;
    max-width: 75%;
    padding: 5%;
}

.tm-service p {
    font-size: 1.5vmax;
}

.tm-service .b {
    padding-bottom: 2%;
    font-weight: 700;
}

/* OUR FACILITIES AND SERVICES END */

/* SLIDER START */
section.slider-sec h2 {
    margin-bottom: 0;
}

.slider {
    margin: 20px auto;
    text-align: center;
    padding: 5px;
    color: white;
    margin-bottom: 70px;
}

.slide {
    height: 330px;
    width: 500px;
    padding: 35px;
}

.slick-prev,
.slick-next {
    background: black;
    width: 40px;
    height: 50px;
    z-index: 1;
}

.slick-prev {
    left: 0;
}

.slick-next {
    right: 0;
}

.slick-center {
    -webkit-transform: scale(1);
    -moz-transform: scale(1);
    transform: scale(1.3);
}

.slick-prev:hover,
.slick-next:hover {
    background: black;
}

/* SLIDER END */

/* FOOTER START */
#footer.main-footer {
    position: relative;
    z-index: 2;
    background: black;
    color: white;
    text-align: center;
}

.container-body {
    max-width: 100%;
    height: 37vh;
    margin: auto;
    display: flex;
    padding-top: 30px;
    justify-content: space-evenly;
}

.colum1 {
    max-width: 400%;
}
```

```
.colum1 img {
    width: 150px;
    height: 150px;
    display: block;
}

.colum1 h1 {
    font-size: 25px;

}

.colum1 p {
    font-size: 15px;
    color: #C7C7C7;
    margin-top: 20px;
}

.colum2 {
    max-width: 400px;
    padding-left: 16px;
}

.colum2 h1 {
    font-size: 25px;
    padding-bottom: 23px;
}

.row {
    margin-top: 20px;
    display: flex;
    padding-left: 25px;
}

.row img {
    width: 30px;
    height: 30px;
    display: block;
}

.row p {
    font-size: 16.5px;
    padding-left: 15px;
    color: #C7C7C7;
}

.row label {
    margin-top: 10px;
    margin-left: 20px;
    color: #C7C7C7;
}

.colum3 {
    max-width: 400px;
    padding-left: 16px;
}

.colum3 h1 {
    font-size: 25px;
    padding-bottom: 23px;
    padding-right: 50px;
}

.row2 {
    margin-top: 18px;
    display: flex;
    padding-left: 25px;
}

.row2 img {
    width: 32px;
    height: 32px;
    display: block;
}

.row2 p {
    font-size: 16.5px;
    padding-left: 15px;
    color: #C7C7C7;
}

.row2 label {
    margin-top: 10px;
    margin-left: 20px;
    color: #C7C7C7;
}

.container-footer {
    width: 100%;
    background: #101010;
}

.footer {
    max-width: 65%;
    height: 45px;
    margin: auto;
    justify-content: space-between;
    display: flex;
    padding: 17px;
    font-size: 14px;
}

.copyright {
    color: #C7C7C7;
    padding-bottom: 5px;
}

.copyright a {
    text-decoration: none;
    color: white;
    font-weight: bold;
}
```

```
.information a {
    text-decoration: none;
    color: #C7C7C7;
    padding-bottom: 10px;
}

@media screen and (max-width: 1100px) {
    .container-body {
      flex-wrap: wrap;
    }

    .colum1 {
      max-width: 100%;

}

.colum1 p {
    font-size: 15px;
    color: #C7C7C7;
    margin-top: 20px;
}

.colum2 {
    max-width: 400px;
    padding-left: 16px;
}

.colum2 h1 {
    font-size: 25px;
    padding-bottom: 23px;
}

.row {
    margin-top: 20px;
    display: flex;
    padding-left: 25px;
}

.row img {
    width: 30px;
    height: 30px;
    display: block;
}

.row p {
    font-size: 16.5px;
    padding-left: 15px;
    color: #C7C7C7;
}

.row label {
    margin-top: 10px;
    margin-left: 20px;
    color: #C7C7C7;
}

.colum3 {
    max-width: 400px;
    padding-left: 16px;
}

.colum3 h1 {
    font-size: 25px;
    padding-bottom: 23px;
    padding-right: 50px;
}

.row2 {
    margin-top: 18px;
    display: flex;
    padding-left: 25px;
}

.row2 img {
    width: 32px;
    height: 32px;
    display: block;
}

.row2 p {
    font-size: 16.5px;
    padding-left: 15px;
    color: #C7C7C7;
}

.row2 label {
    margin-top: 10px;
    margin-left: 20px;
    color: #C7C7C7;
}

.container-footer {
    width: 100%;
    background: #101010;
}

.footer {
    max-width: 65%;
    height: 45px;
    margin: auto;
    justify-content: space-between;
    display: flex;
    padding: 17px;
    font-size: 14px;
}

.copyright {
    color: #C7C7C7;
    padding-bottom: 5px;
}
```

```
.copyright a {
    text-decoration: none;
    color: white;
    font-weight: bold;
}

.information a {
    text-decoration: none;
    color: #C7C7C7;
    padding-bottom: 10px;
}

@media screen and (max-width: 1100px) {
    .container-body {
        flex-wrap: wrap;
    }

    .colum1 {
        max-width: 100%;

    }

    .colum2,
    .colum3 {
        margin-top: 40px;
    }
}

/* FOOTER END */

/* SCROLL BAR START */

::-webkit-scrollbar {
    width: 8px;
    height: 8px;
}

::-webkit-scrollbar-track {
    background-color: rgba(255, 255, 255, 0.4);
    border-radius: 10px;
}

::-webkit-scrollbar-thumb {
    background-color: #181717;
    border-radius: 10px;
}

/* SCROLL BAR END */

/* RESPONSIVE MENU START */

@media only screen and (max-width: 1160px) {

    .slide-item {
        background: #ffffff;
        width: 80%;
        height: 65vh;
        -webkit-margin-start: 10px;
    }

    .main-footer {
        height: 130vh;
    }

    .container-body {
        height: 130vh;
    }

    .container-footer {
        height: 12vh;
    }

    .footer {
        height: 30vh;
        display: grid;
        font-size: 13px;
    }

}

@media only screen and (min-width: 768px) and
(max-width: 1024px) {

    .nav-wrapper {
        width: 100%;
    }

    .cover-main {
        width: 105%;
        height: 50vh;
    }

    .slide-item {
        background: #ffffff;
        width: 85%;
        height: 45vh;
    }

    .main-footer {
        height: 25vh;
    }

    .container-body {
        height: 25vh;
    }

    .colum2,
    .colum3 {
        margin-top: 7px;
    }

    .footer {
```

```
    max-width: 100%;
    height: 15vh;
    margin: auto;
    font-size: 15px;
  }

}

@media only screen and (max-width: 768px) {

  .nav-wrapper {
    width: 100%;
  }

  .menu-trigger {
    background: url('../../images/menu-icon.svg') no-
repeat center/cover;
    width: 30px;
    height: 35px;
    display: block;
    z-index: 10;
    position: absolute;
    right: 20px;
    top: 15px;

  }

  .main-overlay {
    position: fixed;
    top: 0;
    right: 0;
    left: 0;
    bottom: 0;
    background: rgba(0, 0, 0, 0.72);
    z-index: 6;
    display: none;
  }

  .main-overlay:after {
    background: url('../../images/close-icon.svg') no-
repeat center/cover;
    width: 35px;
    height: 35px;
    display: block;
    content: "";
    position: fixed;
    top: 10px;
    right: 20px;
    z-index: 20;
  }

  .menu {
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    background: #131313;
    width: 100%;
    max-width: px;
    z-index: 30;
    -webkit-transform: translate3d(-100%, 0, 0);
    transform: translate3d(-100%, 0, 0);
    -webkit-transition: 0.3s;
    transition: 0.3s;
  }

  nav.nav-wrapper ul li {
    display: block;
    margin-top: 20px;
    padding: 0;
    padding-bottom: 20px;
  }

  nav.nav-wrapper ul li a {
    font-size: 18px;
    line-height: 23px;
  }

  nav.nav-wrapper ul li a:hover {
    border-color: transparent;
  }

  body.menu-is-active .menu {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
    display: block;
    padding: 0 15px;
  }

  body.menu-is-active .main-overlay {
    display: block;
  }

  body.menu-is-active .menu-trigger {
    display: none;
  }

}

@media only screen and (max-width: 650px) {

  .nav-wrapper {
    width: 100%;
  }

  ul#scene li.name {
    font-size: 22px;
    line-height: 27px;
  }

  ul#scene li.title {
    font-size: 20px;
```

```
  line-height: 25px;
  margin-top: 40px;
}

div.grid {
  width: initial !important;
}

.btn {
  margin-top: 80px;
}

ul#scene {
  margin-top: 50px;
}

.reservation-btn,
.reservations-btn {
  display: block;
  margin: 20px auto;
  max-width: 120px;
  font-size: 15px;
  line-height: 20px;
  padding: 10px 0;
}

h2.sub-title {
  margin: 30px auto;
  font-size: 23px;
  line-height: 28px;
}

section.beating-heart {
  height: initial;
}

.grid-item {
  width: initial;
}

.tall-img {
  height: initial;
}

.menu {
  max-width: 150px;
}

.slide-item {
  margin-left: 50px;
  position: relative;
  padding: 7px 0;
  vertical-align: top;
  background-color: rgb(19, 18, 18);
  color: #ffffff;
  }

  .slide-item h3 {
    color: white;
  }

  .slide-item img {
    filter: grayscale(0);
  }

  .main-footer {
    height: 130vh;
  }

  .container-body {
    padding: 15%;

  }

  .container-footer {
    height: 15vh;
  }

  .footer {
    max-width: 100%;
    height: 10vh;
    margin: auto;
    font-size: 15px;
  }
}

@media only screen and (max-width: 400px) {
  .nav-wrapper {
    width: 100%;
  }

  .cover-main {
    width: 105%;
    height: 110vh;
  }

  .slide-item {
    background: #ffffff;
    width: 85%;
    height: 90vh;
    -webkit-margin-start: 30px;
  }

  .slide-item h3 {
    color: #000000;
  }

  .slide-item img {
    position: center;
    padding: 20px;
    height: 300px;
```

```
    color: white;
    filter: grayscale(70%);
  }

  .main-footer {
    height: 140vh;
  }

  .container-body {
    height: 140vh;
  }

  .footer {
    height: 15vh;
    display: grid;
    font-size: 13px;
  }

}

@media only screen and (min-width: 300px) and
(max-width:350px) {

  .nav-wrapper {
    width: 100%;
  }

  .main-footer {
    height: 175vh;
  }

  .slide-item {
    background: #ffffff;
    width: 85%;
    height: 110vh;
    -webkit-margin-start: 30px;

  }

  .container-body {
    height: 175vh;
  }

  .footer {
    height: 16vh;
  }
}


/* RESPONSIVE MENU END *
```

Фрагмент JS коду, що відповідає за створення динамічних елементів та функцій в інформаційній системі

```
$(document).ready(function () {

 // define parallax library
 var scene = document.getElementById('scene');
 var parallax = new Parallax(scene);

 // define sticky header
 var $header = $('header');
 var $sticky = $header.before($header.clone().addClass("sticky"));

 $(window).on("scroll", function () {
  var scrollFromTop = $(window).scrollTop();
  $("body").toggleClass("scroll", (scrollFromTop > 350));
 });

 // define smooth scroll
 $('.menu li a[href^="#"]').on('click', function (e) {
  e.preventDefault();

  var target = $(this.hash);

  if (target.length) {
   $('html, body').stop().animate({
    scrollTop: target.offset().top - 70
   }, 1000);
  }

 });
```

```
// define slick slider
$('.slider').slick({
  autoplay: true,
  autoplaySpeed: 2000,
  arrows: true,
  dots: false,
  centerMode: true,
  slidesToShow: 3,
  fade: false,
  prevArrow: '<button type="button" class="slick-prev">Previous</button>',
  nextArrow: '<button type="button" class="slick-next">Previous</button>',

  responsive: [{
      breakpoint: 990,
      settings: {
        slidesToShow: 2
      }
    },
    {
      breakpoint: 768,
      settings: {
        slidesToShow: 1
      }
    }
  ]

});


// define responsive menu
var body = $('body');
var menuTrigger = $('.js-menu-trigger');
var mainOverlay = $('.js-main-overlay');

menuTrigger.on('click', function () {
  body.addClass('menu-is-active');
});

mainOverlay.on('click', function () {
  body.removeClass('menu-is-active');
});

$('.menu li a').on('click', function () {
  $('body').removeClass("menu-is-active");
});
```

ДОДАТОК В

Сторінки прототипу системи зроблені у Figma

Розглянемо прототипи сторінок інформаційної системи «Адміністрування готелю» що представлені на рисунках В.1 – В.29.

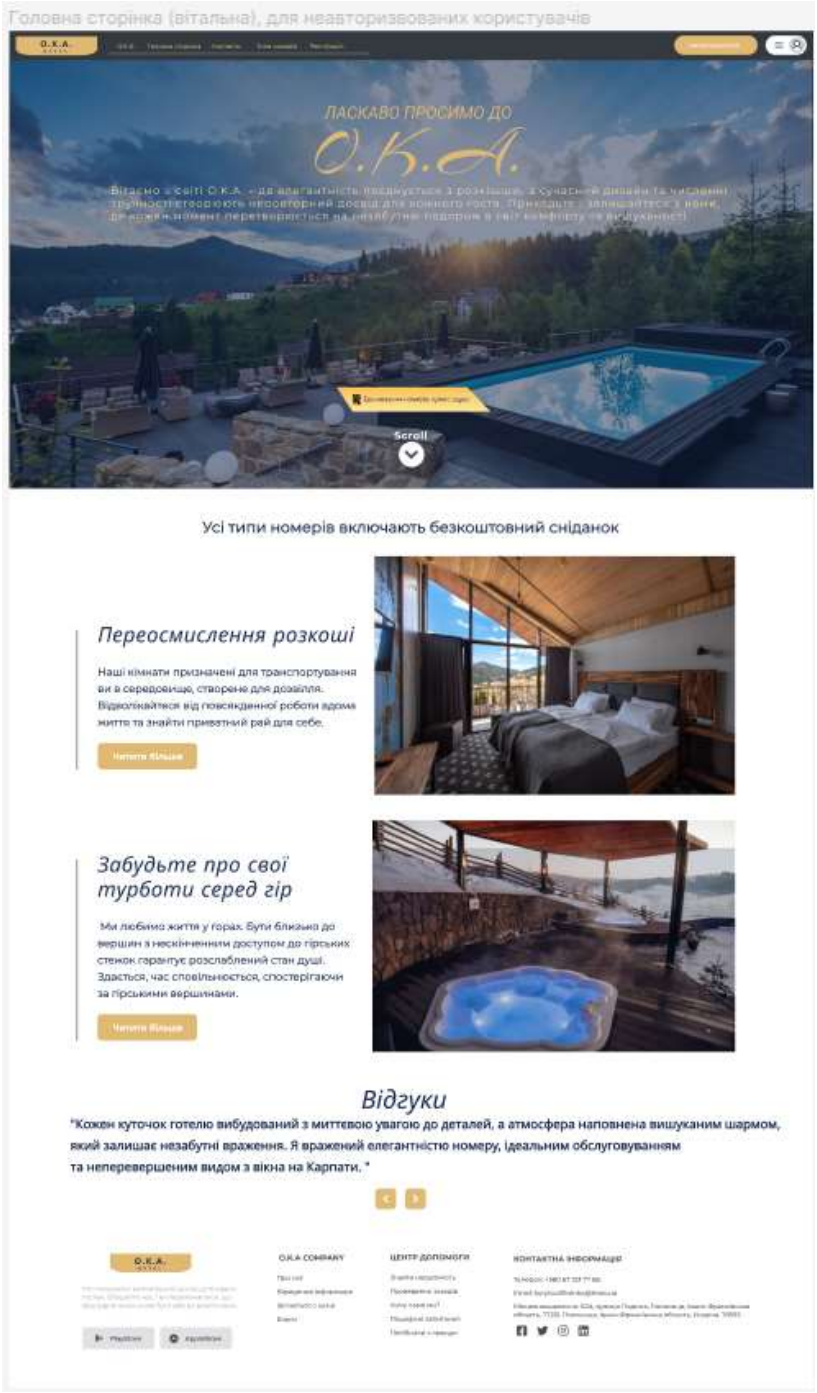


Рисунок В.1 – Головна сторінка (вітальна), для неавторизованих користувачів

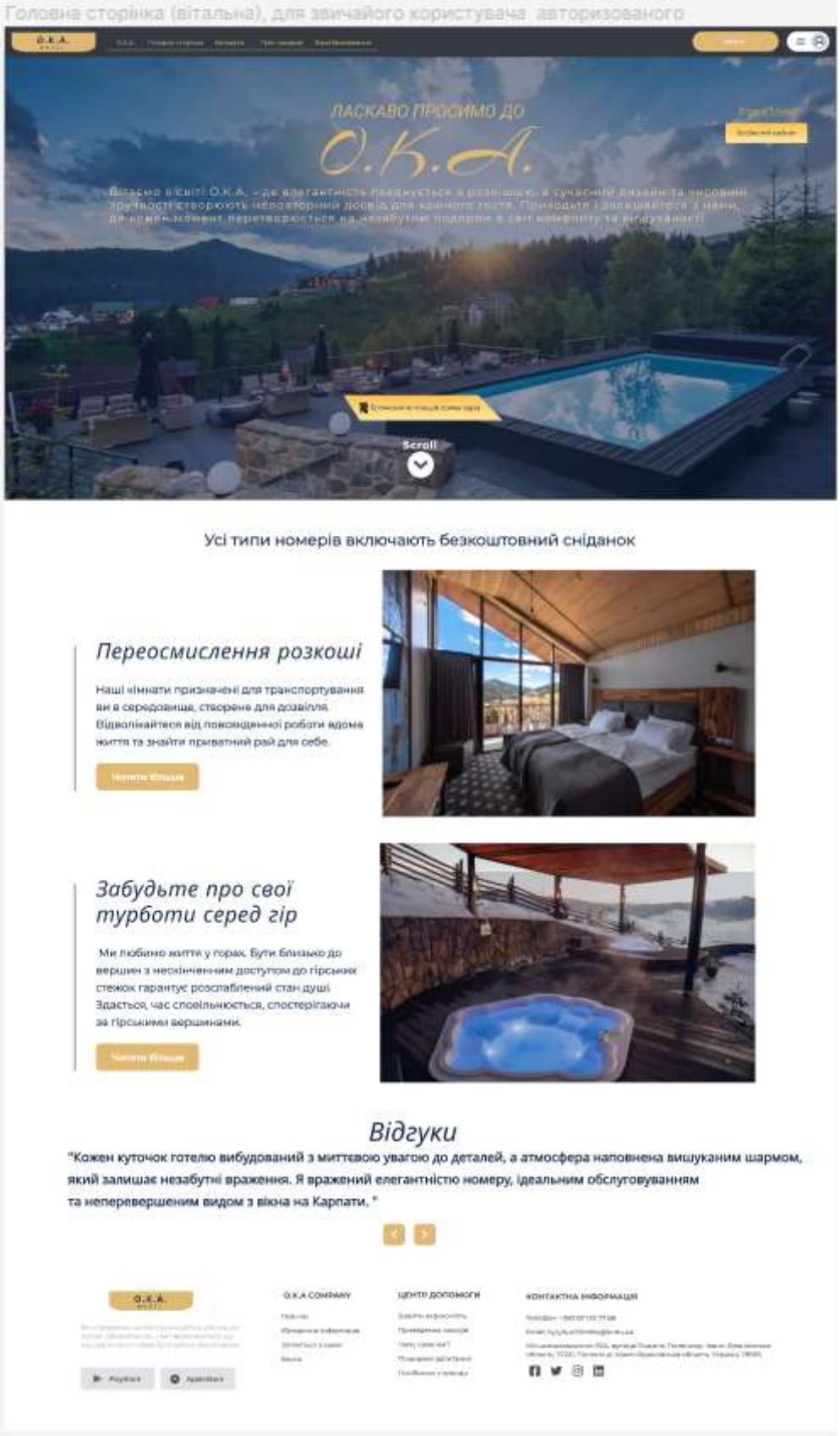


Рисунок В.2 – Головна сторінка (вітальна), для звичайного користувача авторизованого

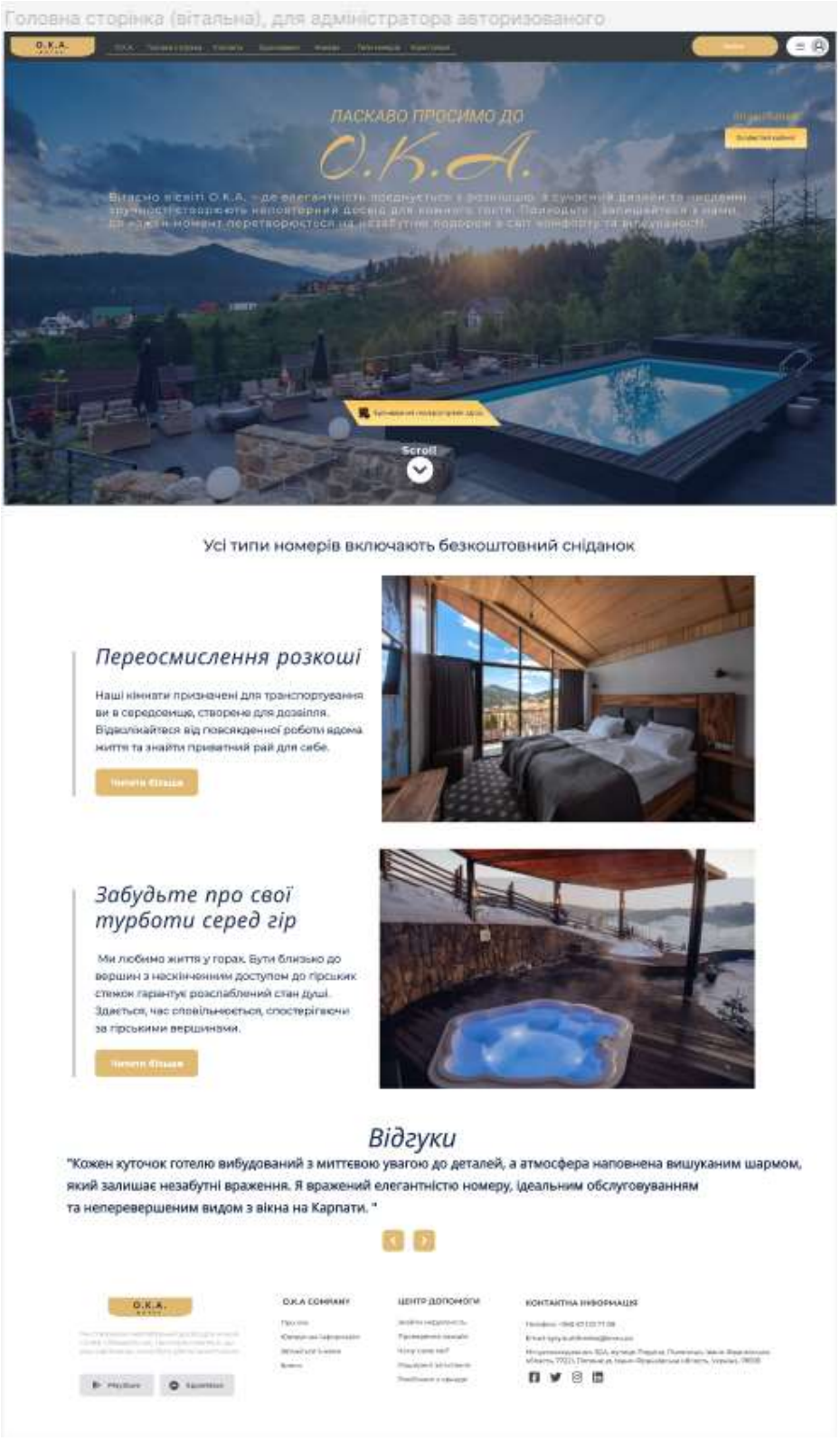


Рисунок В.3 – Головна сторінка (вітальна), для адміністратора авторизованого

Також, реалізована авторизація, можливий варіант через Google акаунт та реєстрація в систему.

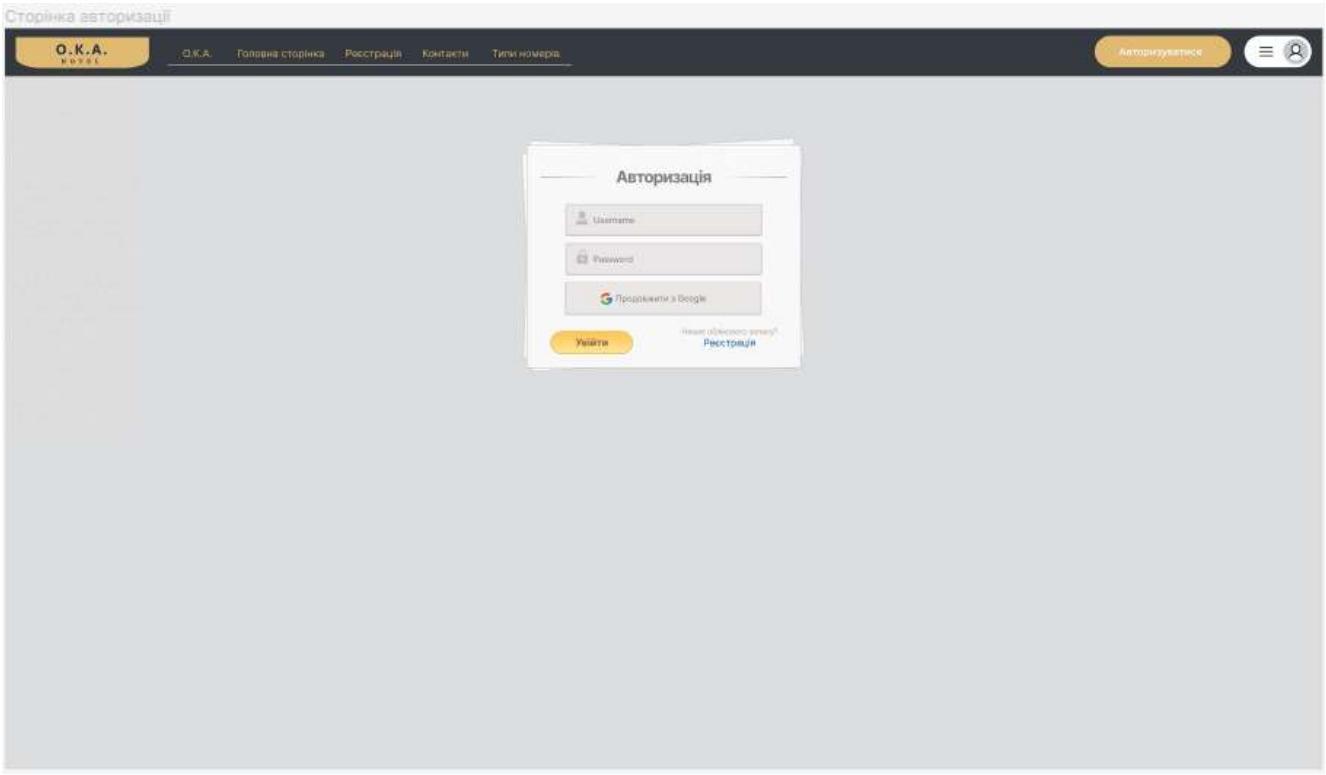


Рисунок В.4 – Сторінка авторизації в системі

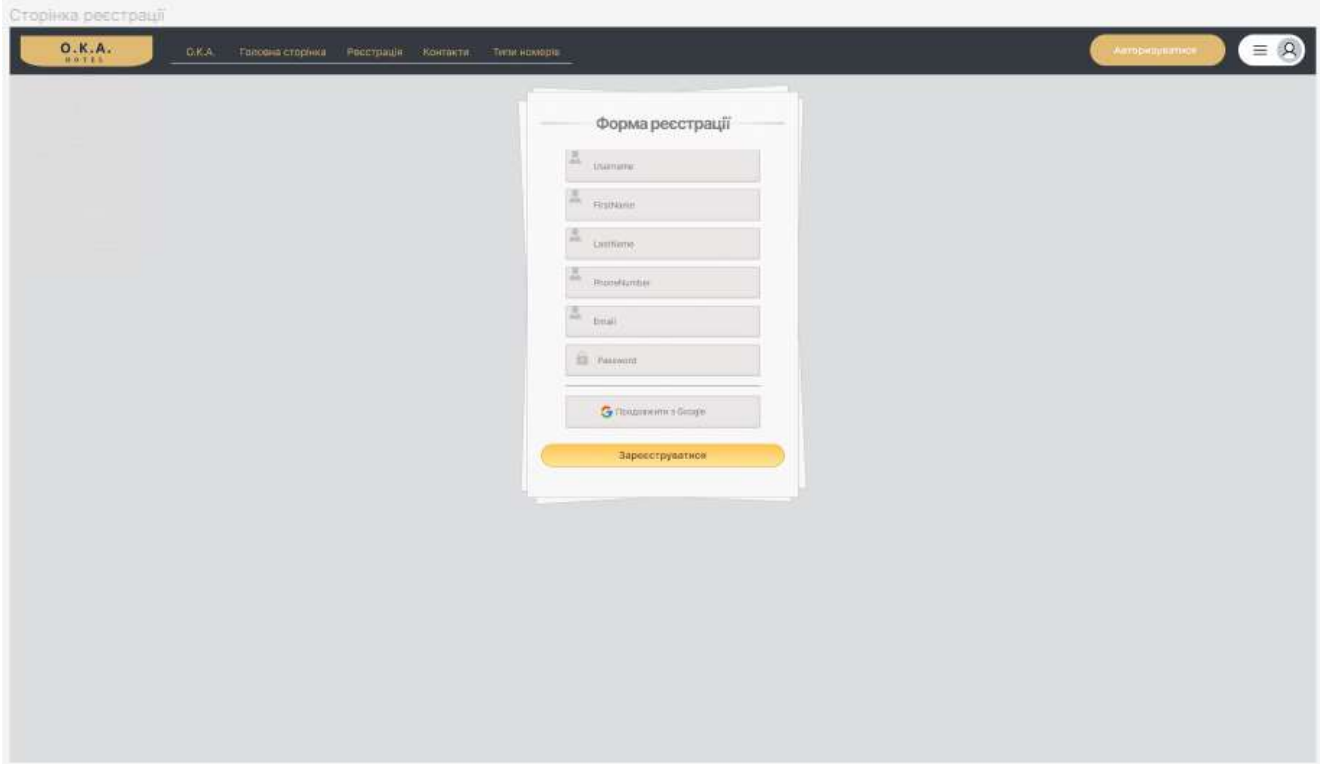


Рисунок В.5 – Сторінка реєстрації в системі

Далі, для зручності реалізоване випадаючий список, меню опцій.

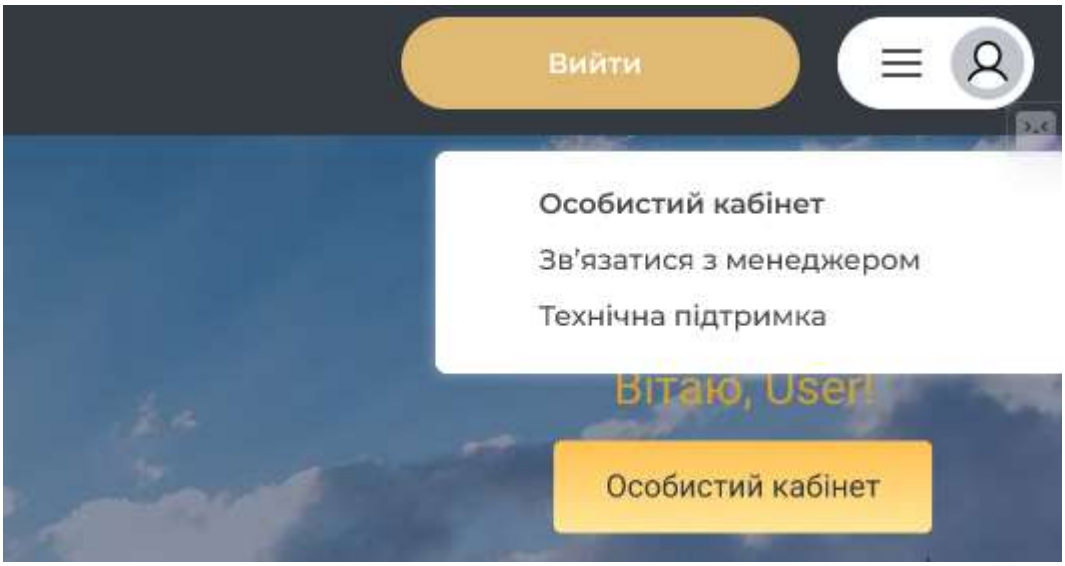


Рисунок В.6 – Випадаючий список «Меню опцій»

Після входу в аккаунт або реєстрації нового облікового запису, ми можемо переглянути особистий кабінет, змінити дані, додати додаткові, змінити пароль, переглянути відгуки, які користувач залишав та подальші операції які він виконував в системі (наприклад, останні перегляди номерів для бронювання).

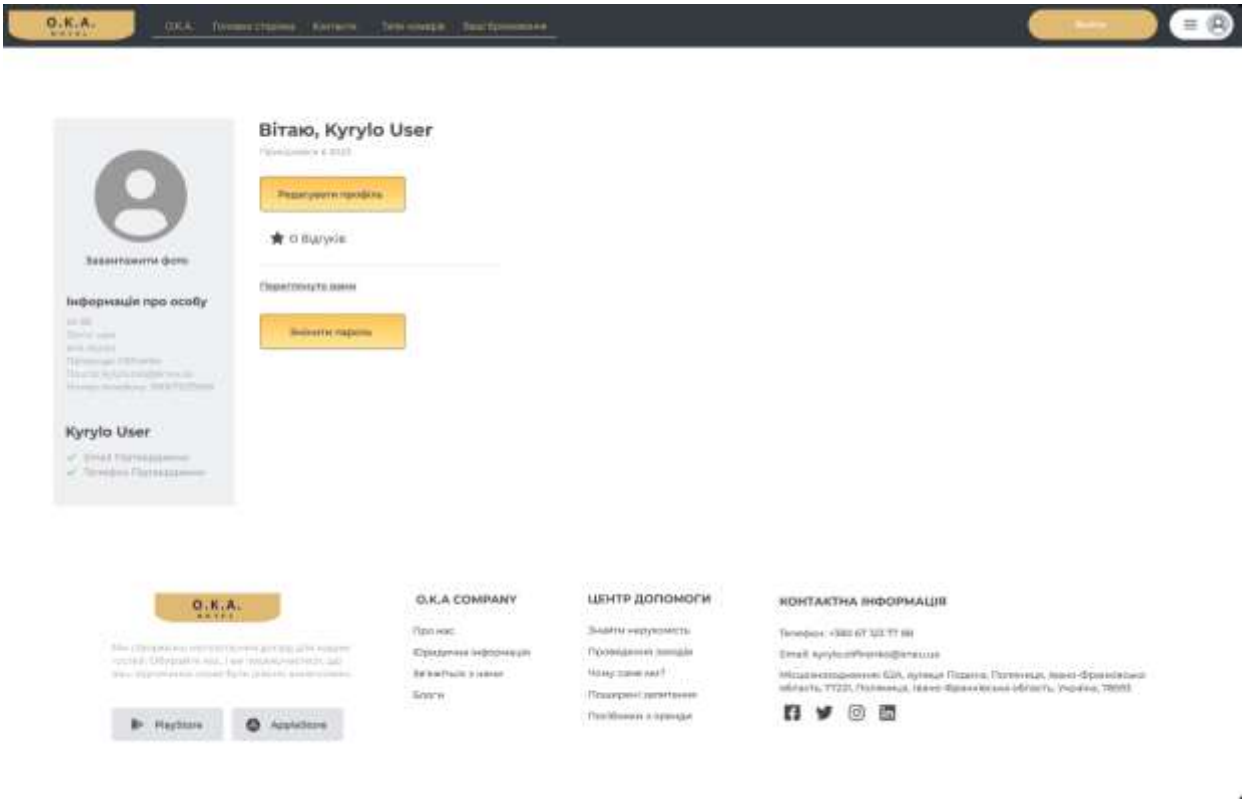


Рисунок В.7 – Сторінка «Особистий кабінет користувача»

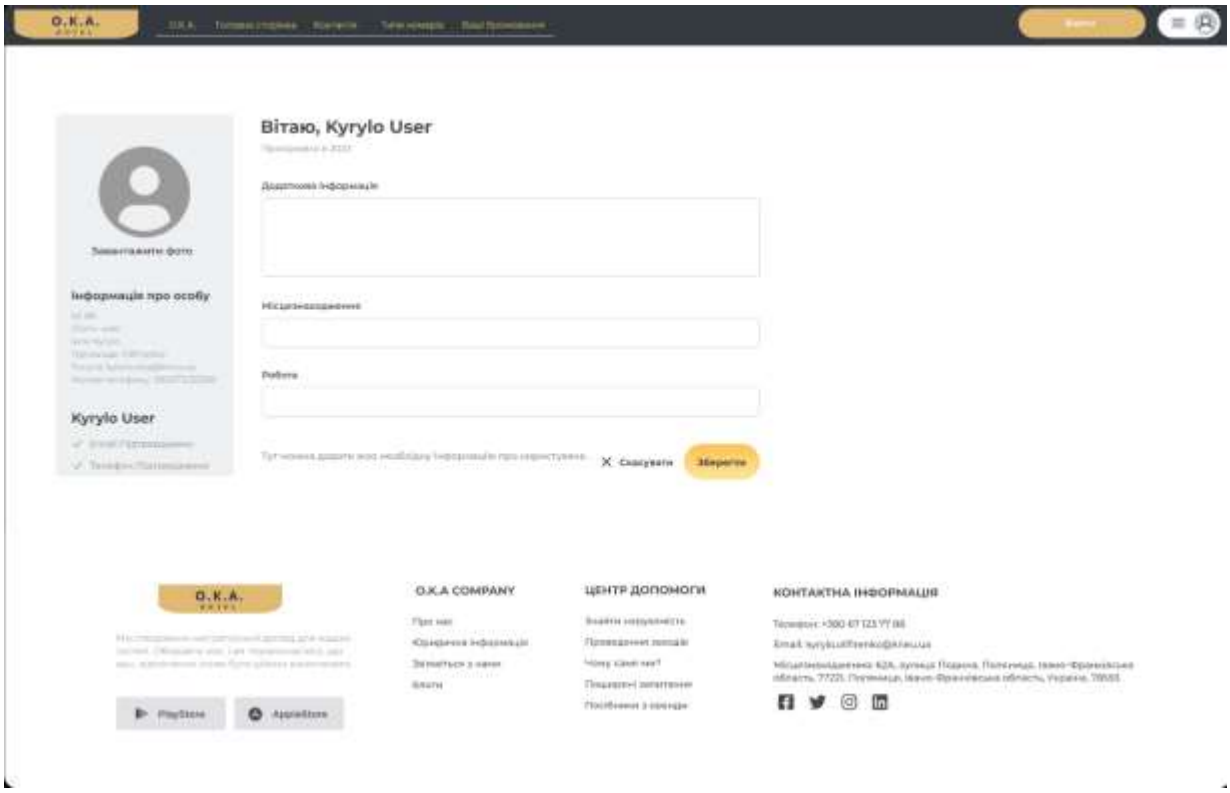


Рисунок В.8 – Сторінка «Редагування інформації особистого кабінету користувача, додавання доп. інфо»

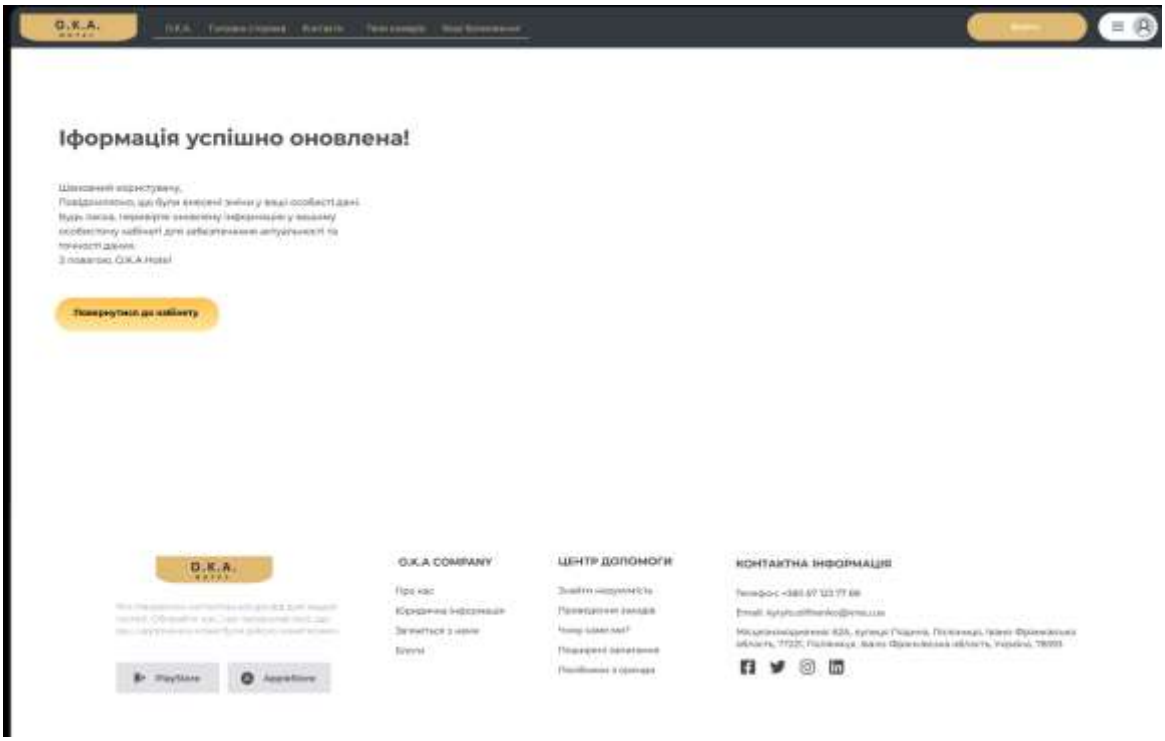


Рисунок В.9 – Повідомлення про успішне додавання нової інформації «особистого кабінету користувача»

Далі, взаємодія адміністратора з особистим кабінетом та його можливості редагування даних, додавання користувачів в систему і їх видалення, так само як із кімнатами готелю.

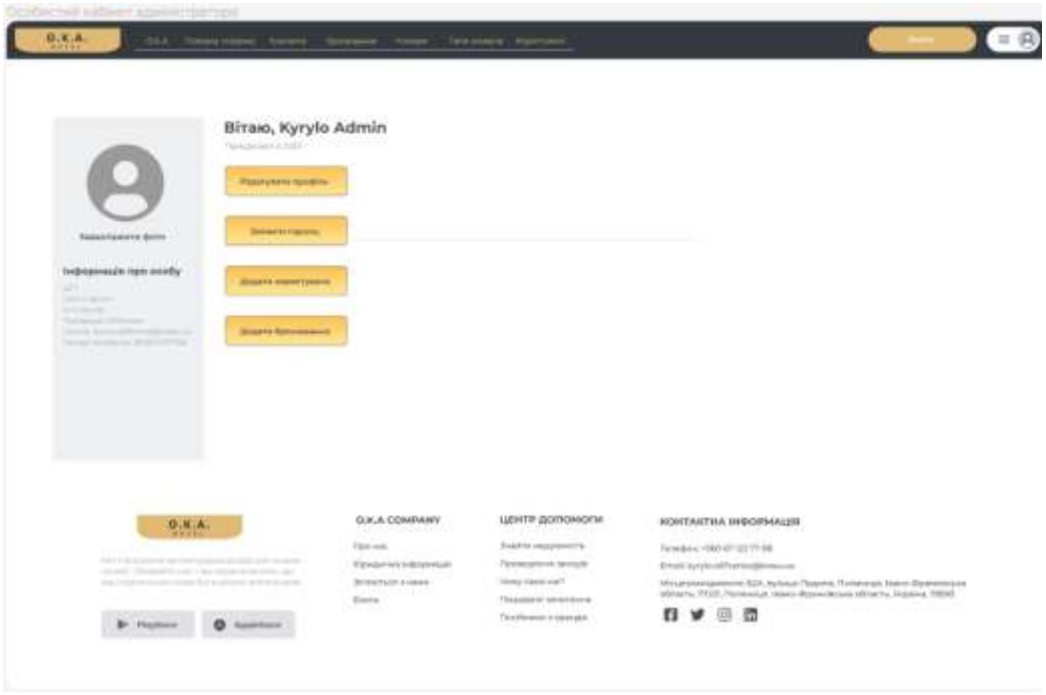


Рисунок В.10 – Сторінка «Особистий кабінет адміністратора»

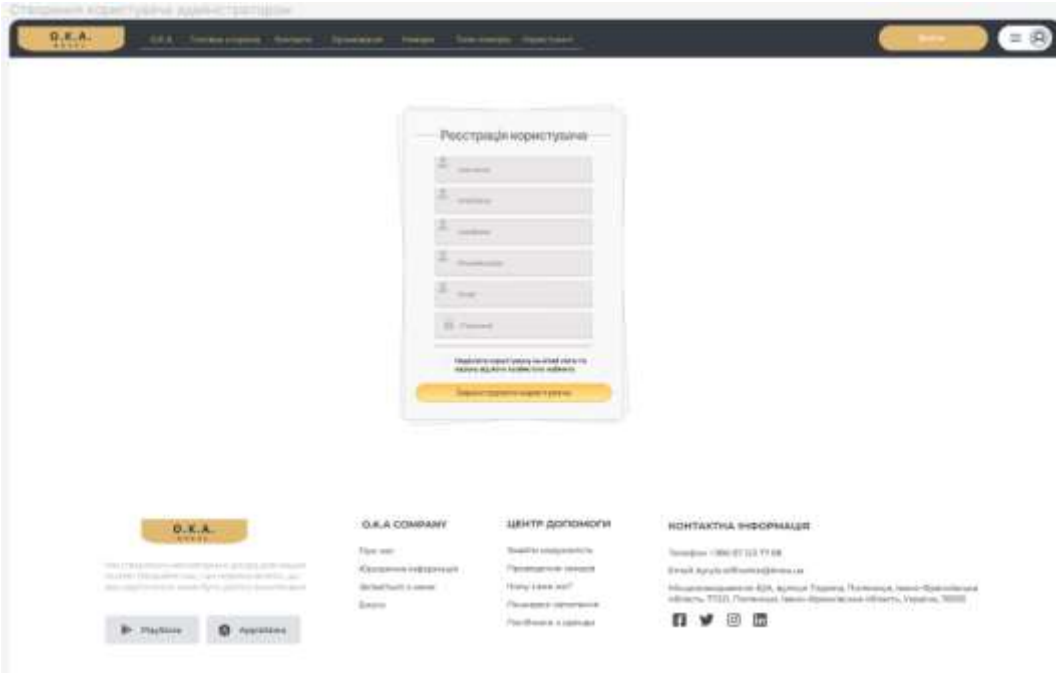


Рисунок В.11 – Створення нового облікового запису клієнта зі сторони адміністратора

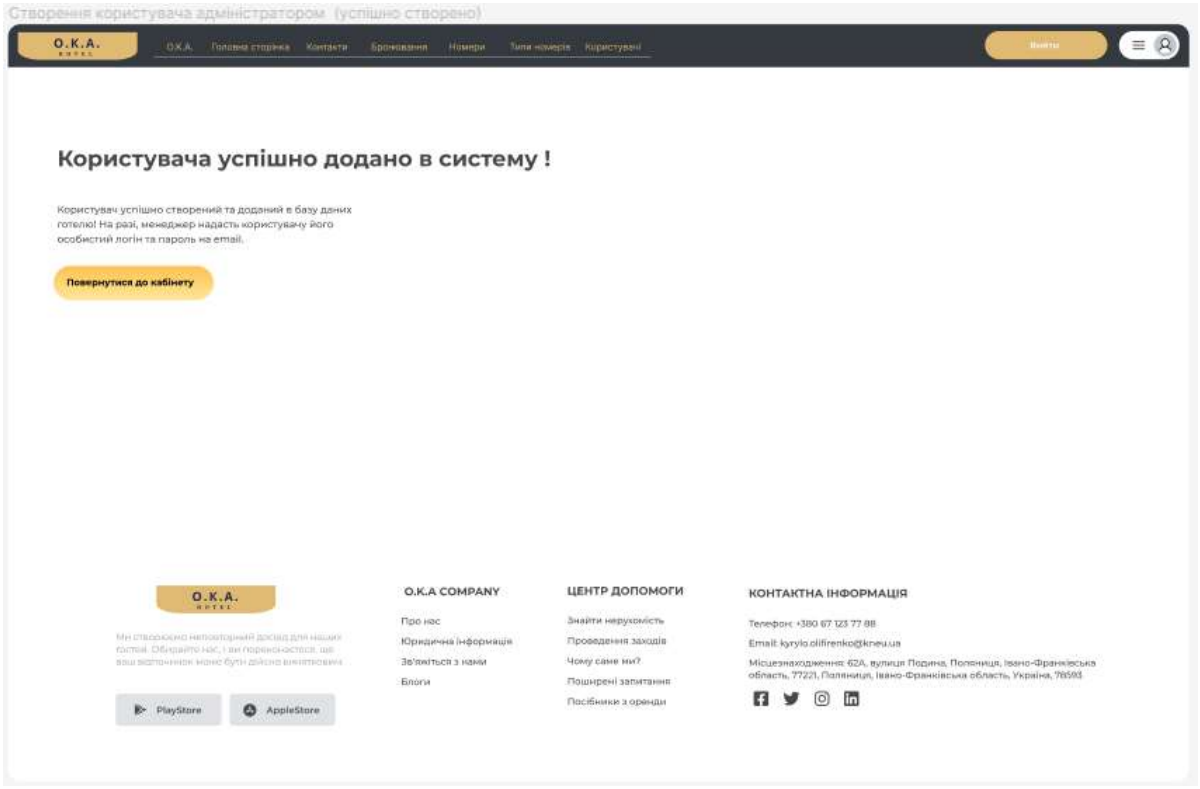


Рисунок В.12 – Повідомлення про успішне створення нового облікового запису клієнта зі сторони адміністратора

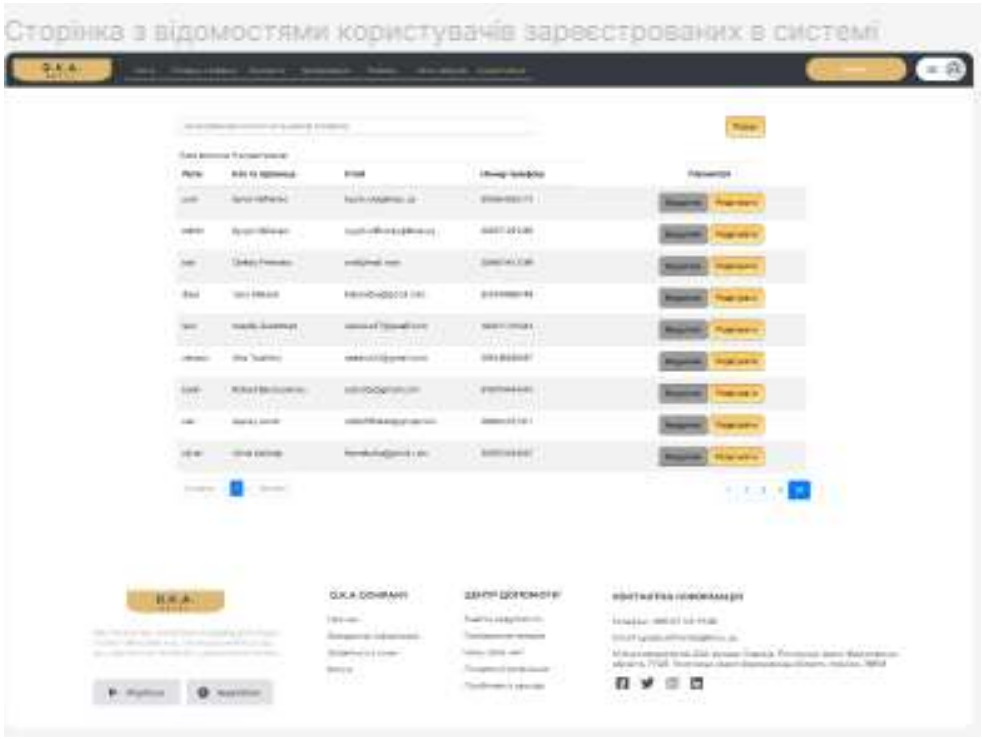


Рисунок В.13 – Сторінка з відомостями користувачів зареєстрованих в системі, можна редагувати дані, додавати доп. обмеження, видаляти облікові записи

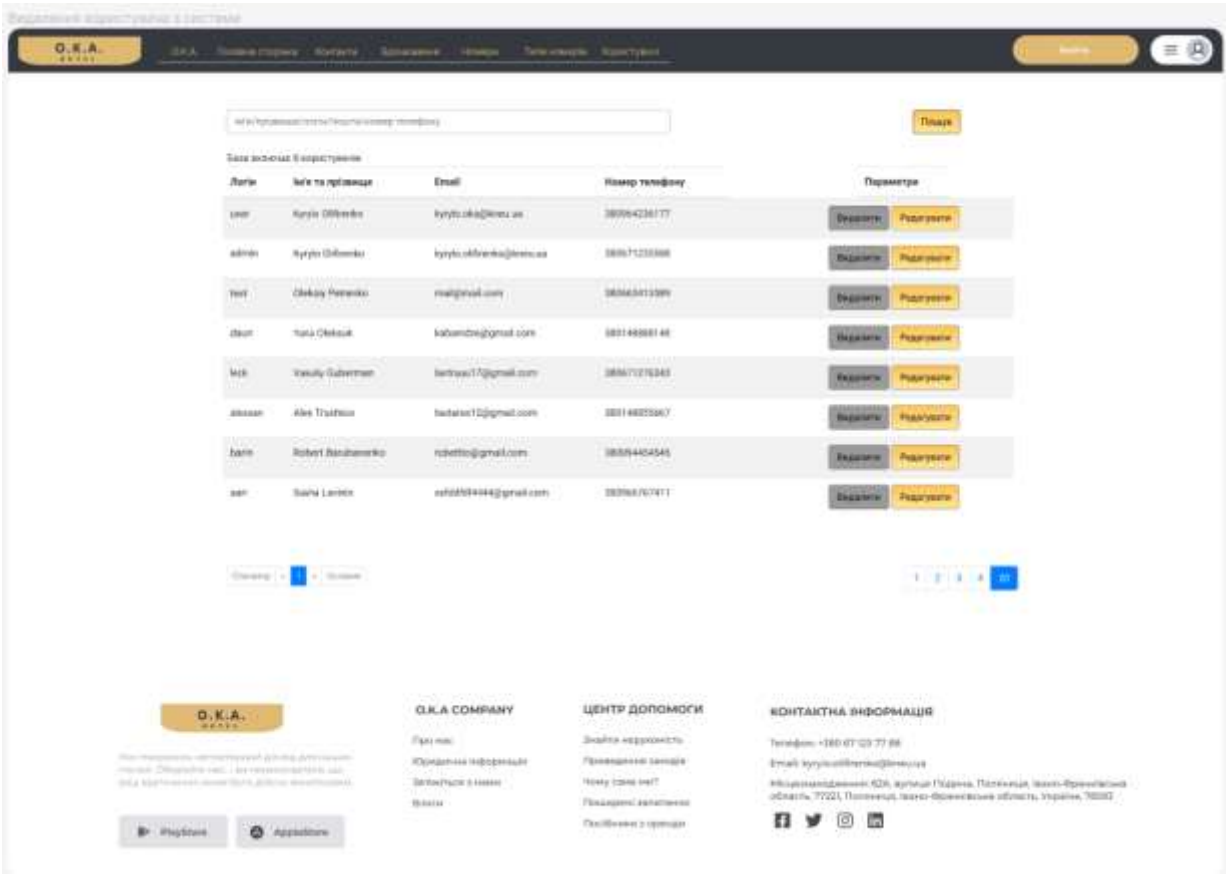
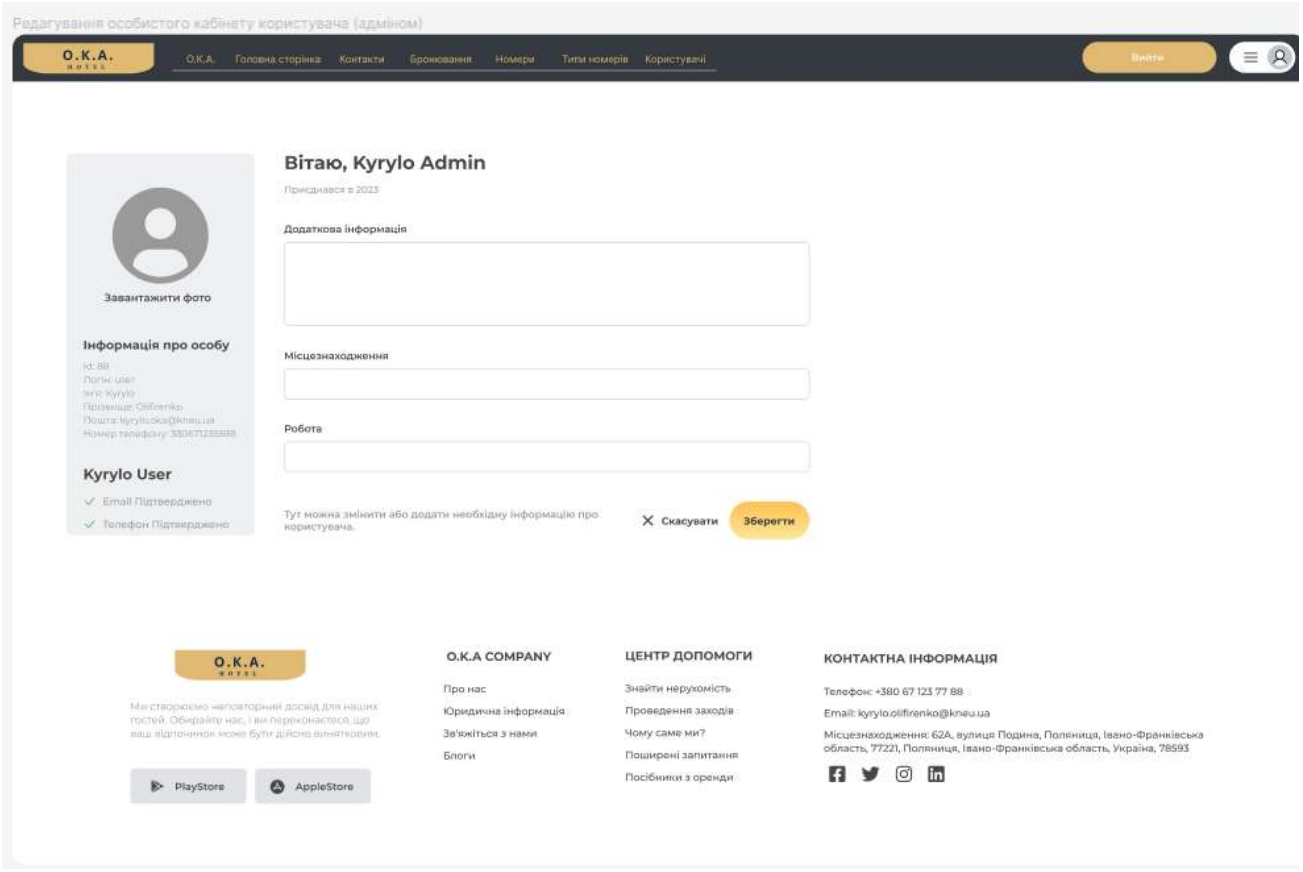
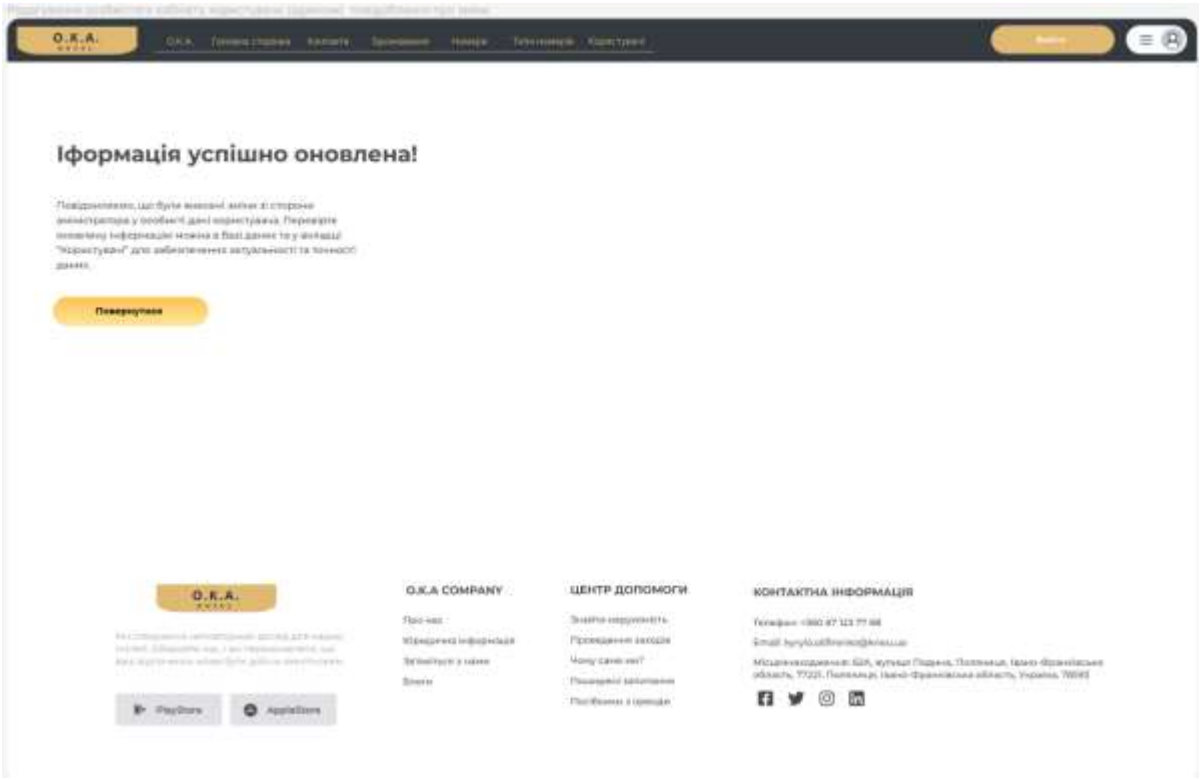


Рисунок В.14 – Видалення облікового запису користувача системи





Рисуноки В.15-В.16 – Повідомлення про успішне додавання нової інформації особистого кабінету користувача зі сторони адміністратора

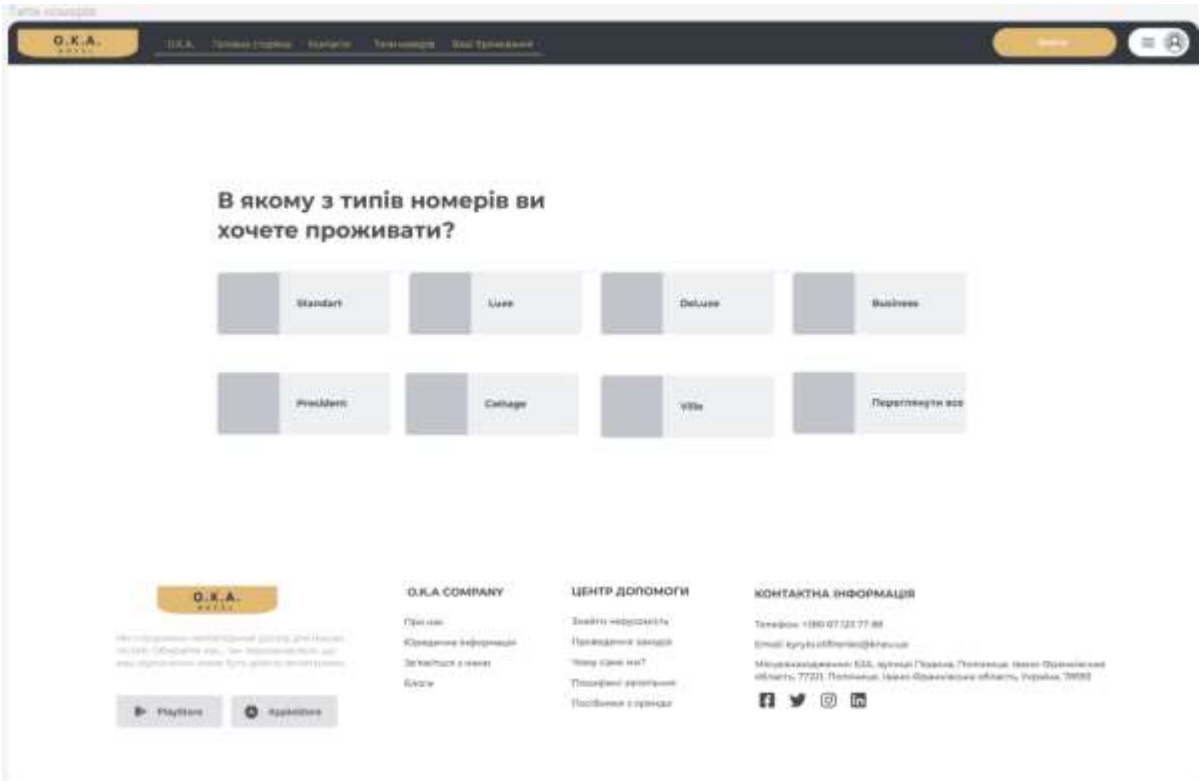


Рисунок В.17 – Всі типи номерів для бронювання

Процес бронювання номерів, фільтрація, пошук, інформація про номери і тд.

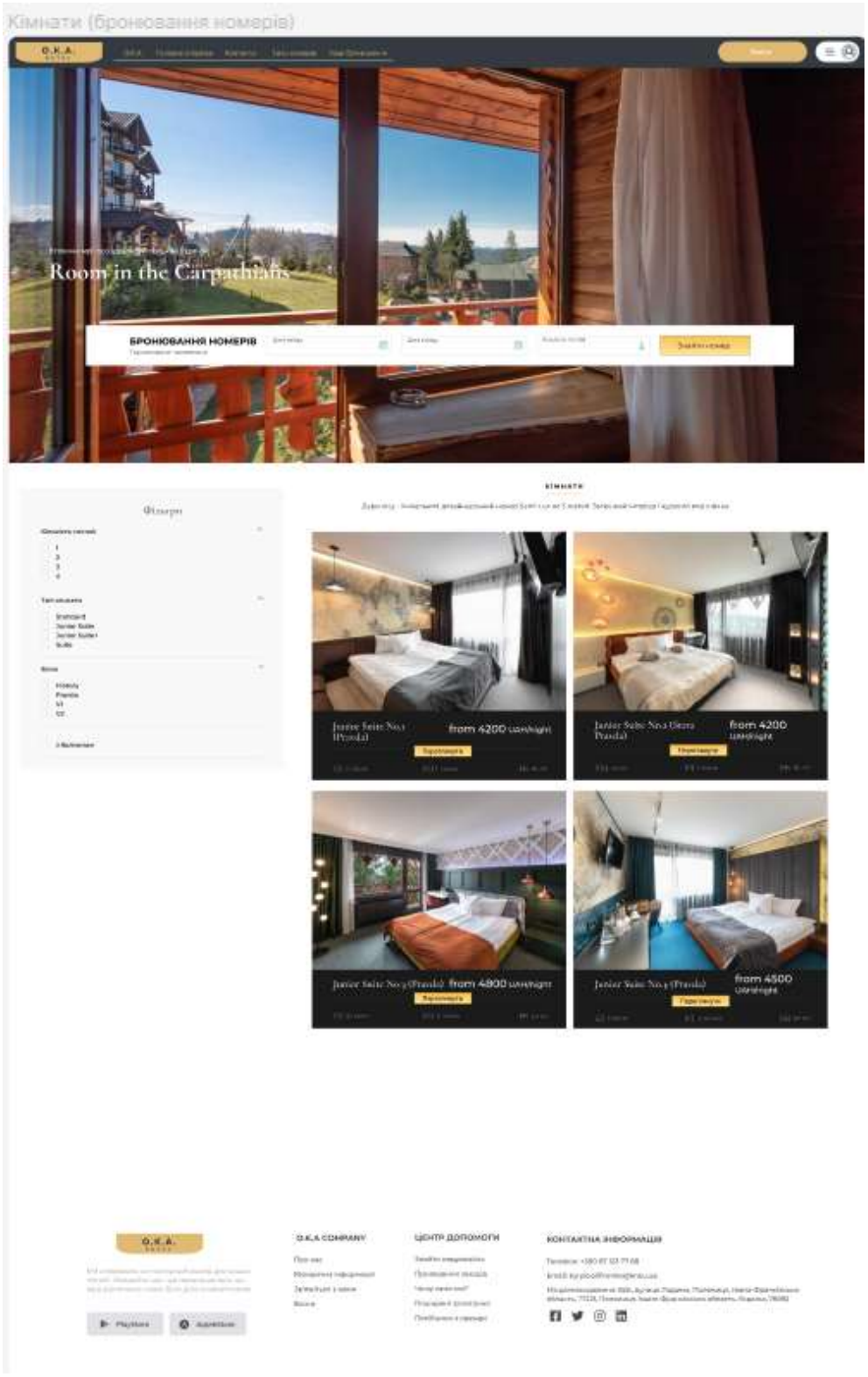
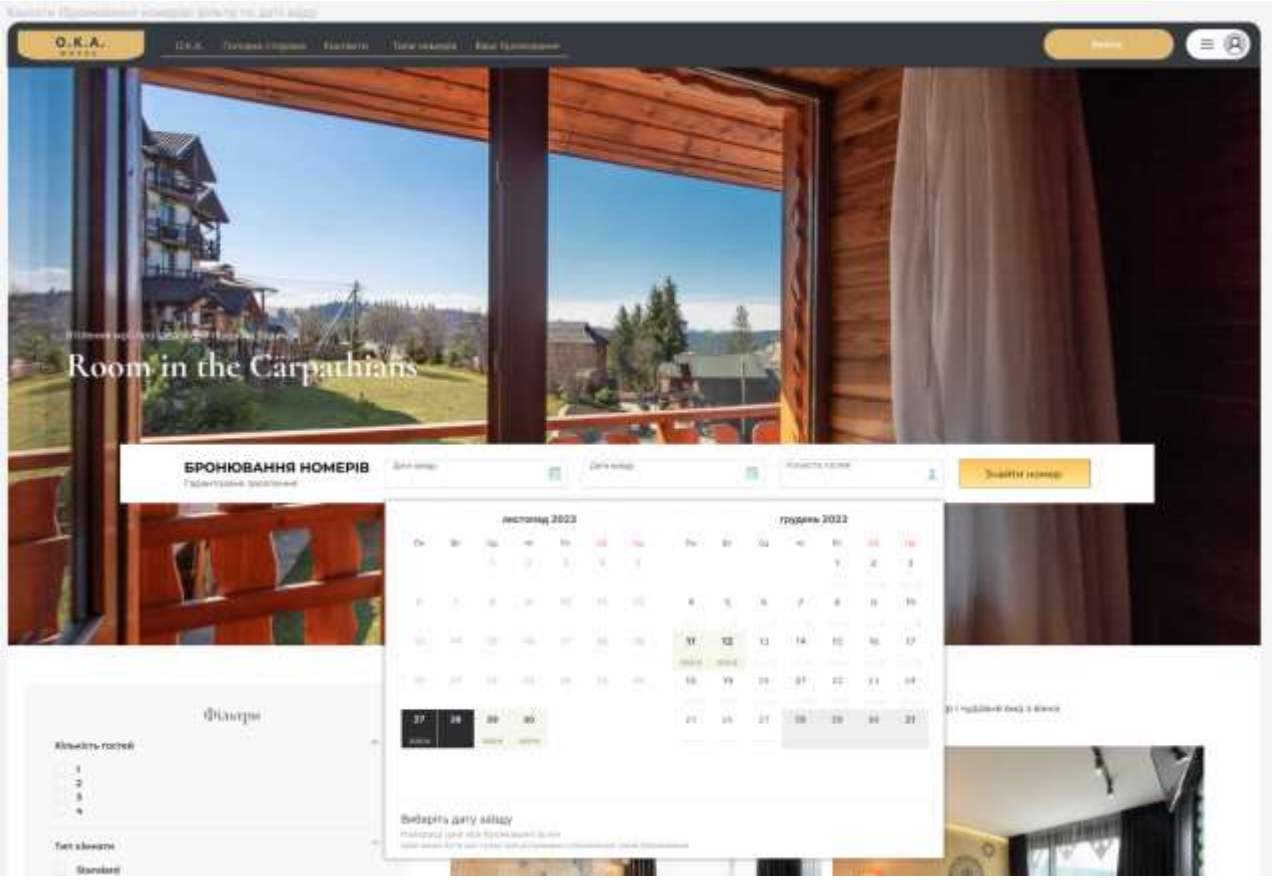
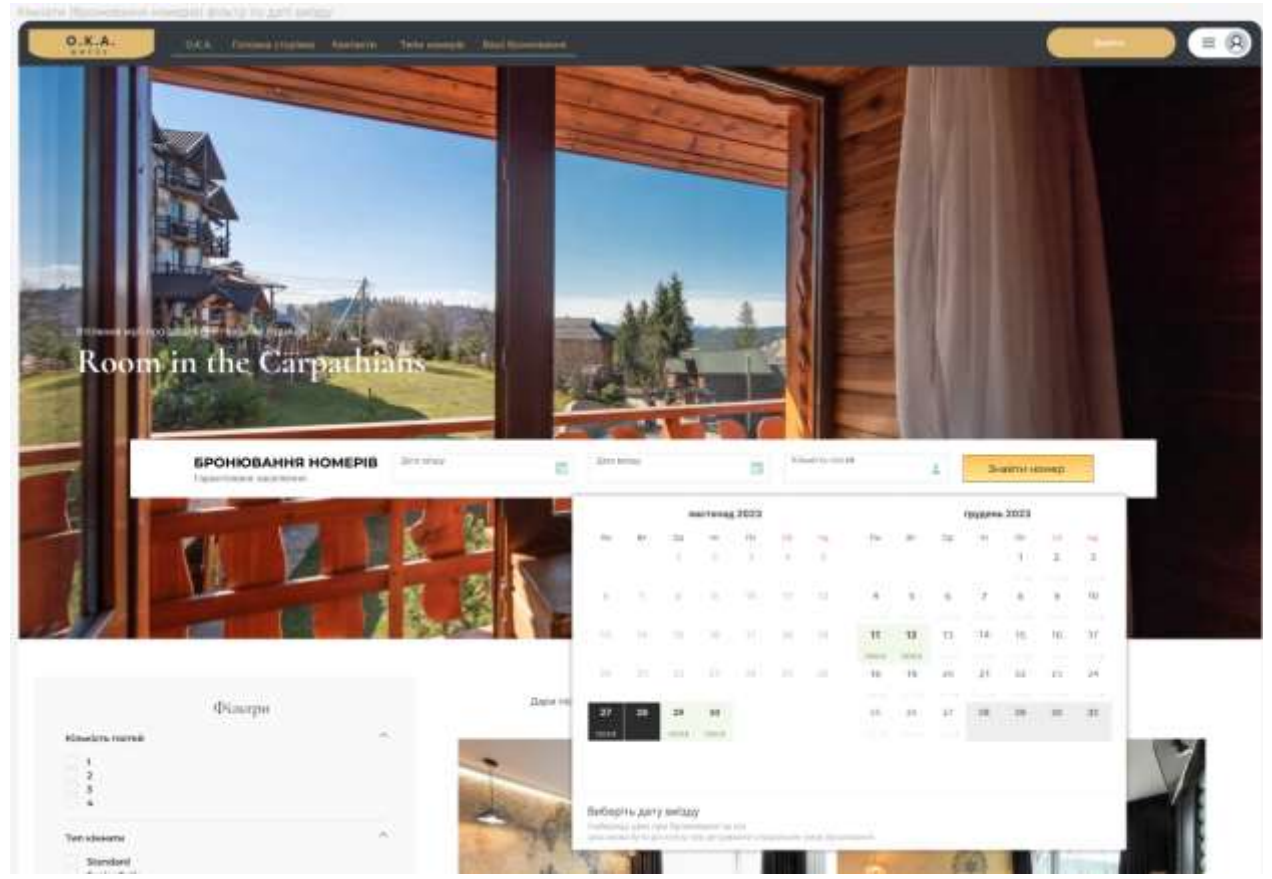
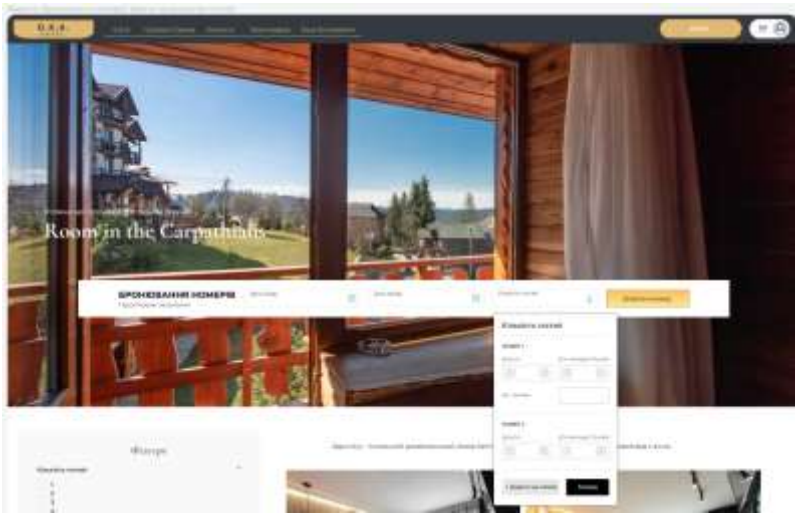


Рисунок В.18 – Після вибору номерів сторінка з пошуком кімнат, де є фільтри та параметри в’їзду, виїзду та кількості гостей


O.K.A.
Room in the Carpathians
БРОНЮВАННЯ НОМЕРІВ
листопад 2023
грудень 2023
Фільтри
Виберіть дату виїзду


O.K.A.
Room in the Carpathians
БРОНЮВАННЯ НОМЕРІВ
листопад 2023
грудень 2023
Фільтри
Виберіть дату виїзду



Рисуноки В.19-В.21 – Випадаючі списки фільтрації, де можна обрати параметри в’їзду, виїзду та кількості гостей для пошуку номеру

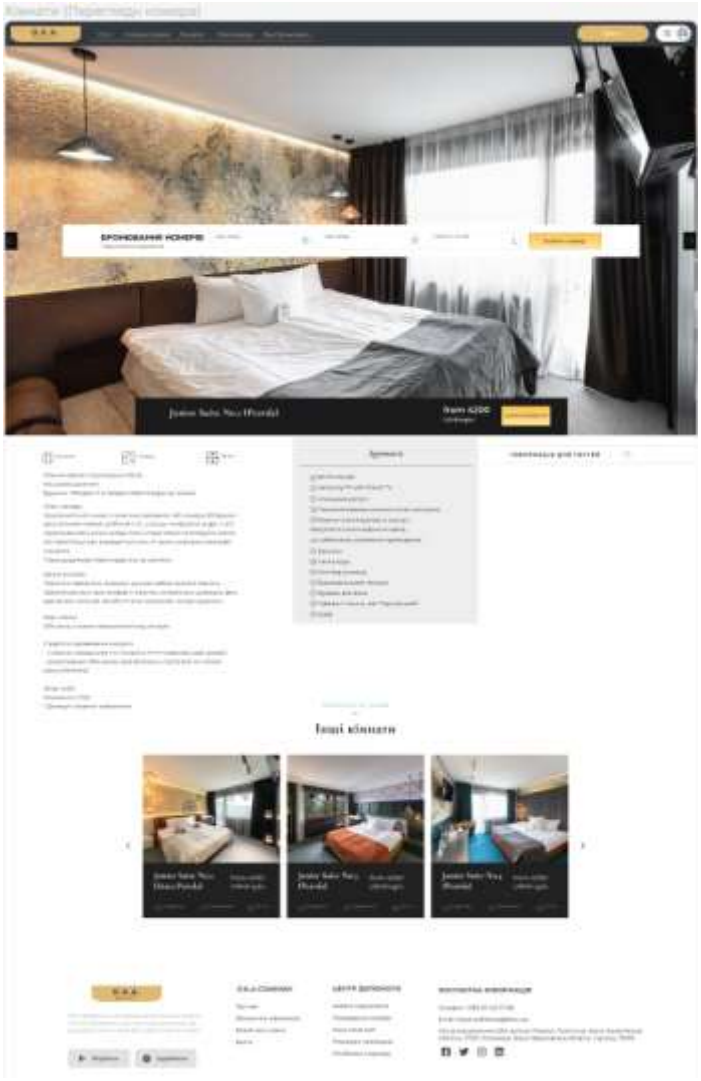


Рисунок В.22 – Сторінка перегляду номеру

Після натискання кнопки «Знайти номер», нас переводить на сторінку перегляду номеру, де можна передивитися всю інформацію, послуги та кімнату або перейти на іншу підходящу кімнату.

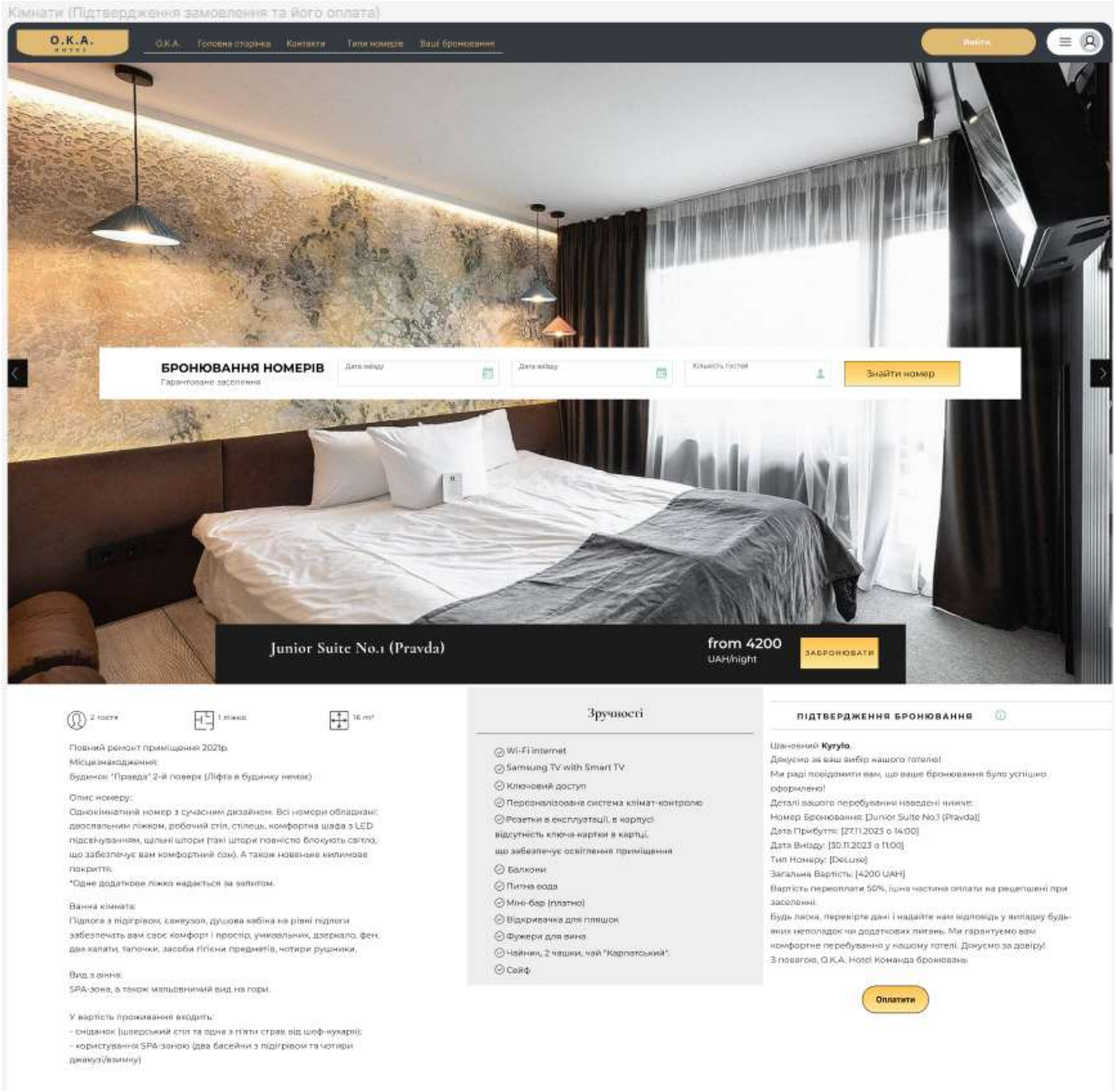


Рисунок В.23 – Сторінка оформлення бронювання

Після натискання на кнопку забронювати, ми повинні погодитися, прочитати умови бронювання та підтвердити замовлення, оплативши 50% від замовлення натиснувши кнопку «Оплатити».

Проведення оплати, вибір платіжної системи продемонстровано на наступному рисунку (див. рисунок В.24).

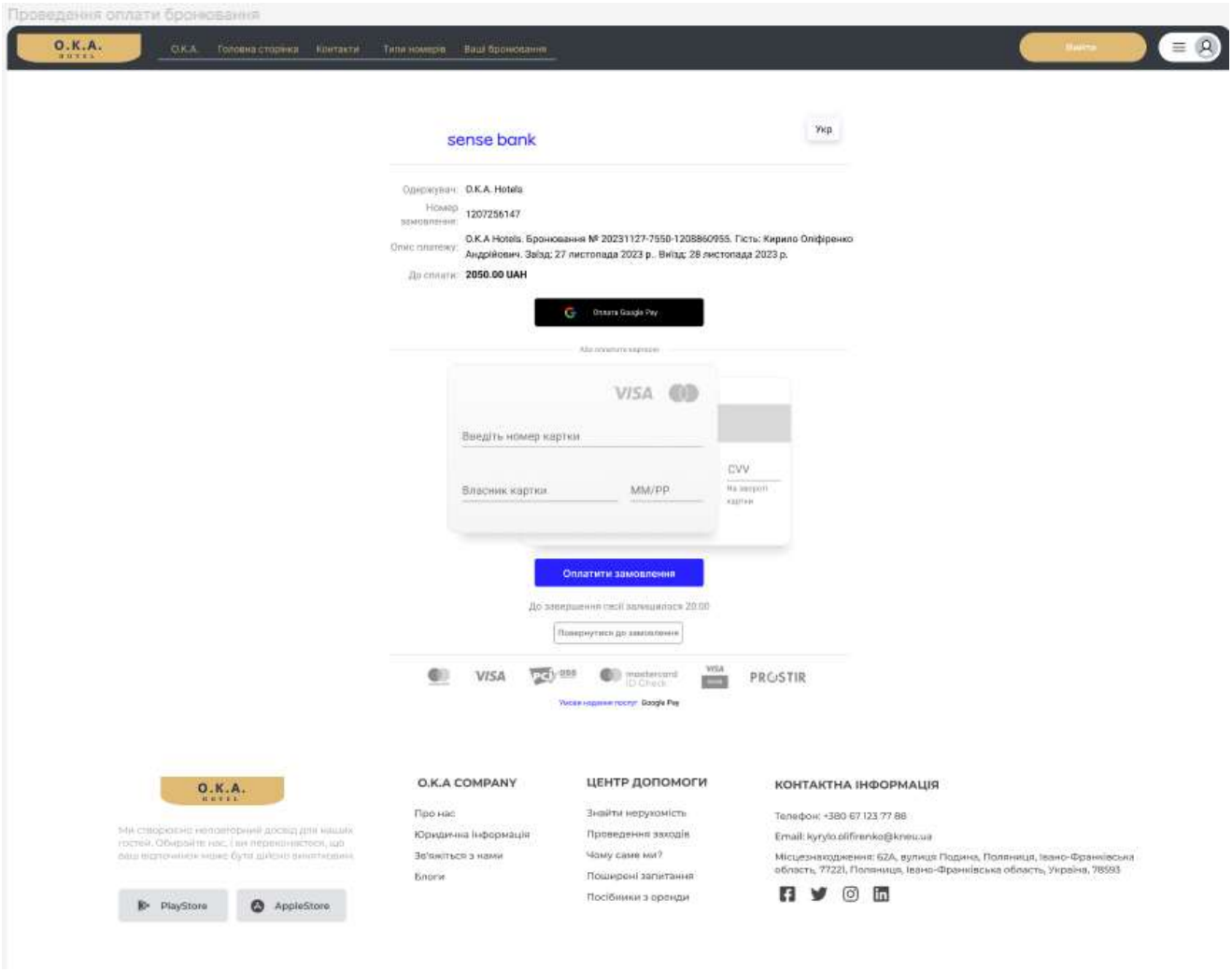


Рисунок В.24 – Сторінка з проведенням оплати

Створено сторінку з платіжною системою, оплатою замовлення, де можна теж використовувати Google Pay, після успішної оплати буде повідомлення від менеджера на пошту з квитанцією про оплати. Ці деталі, статуси також можна дивитися в особистому кабінеті ваших бронюваннях.

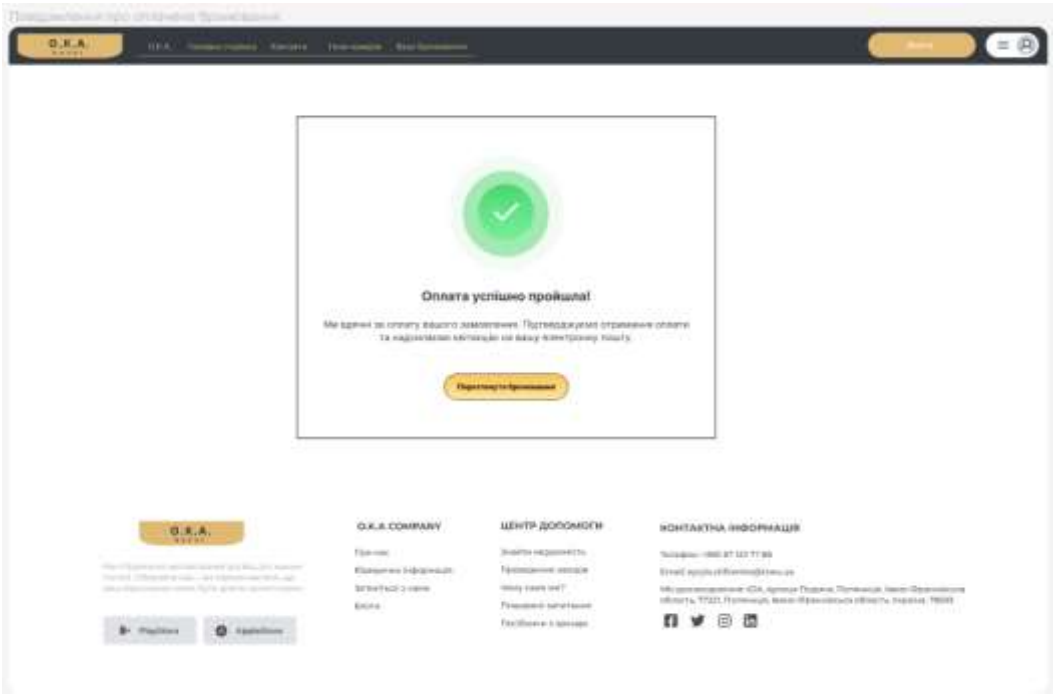


Рисунок В.25 – Повідомлення про успішну оплату

На наступному рисунку (див. рисунок В.26) перегляд своїх бронювань, минулих чи майбутніх, деталі скасування бронювань, повернення коштів та інформації про майбутні бронювання, які зареєстровані менеджером та оплачені згідно умов готелю.

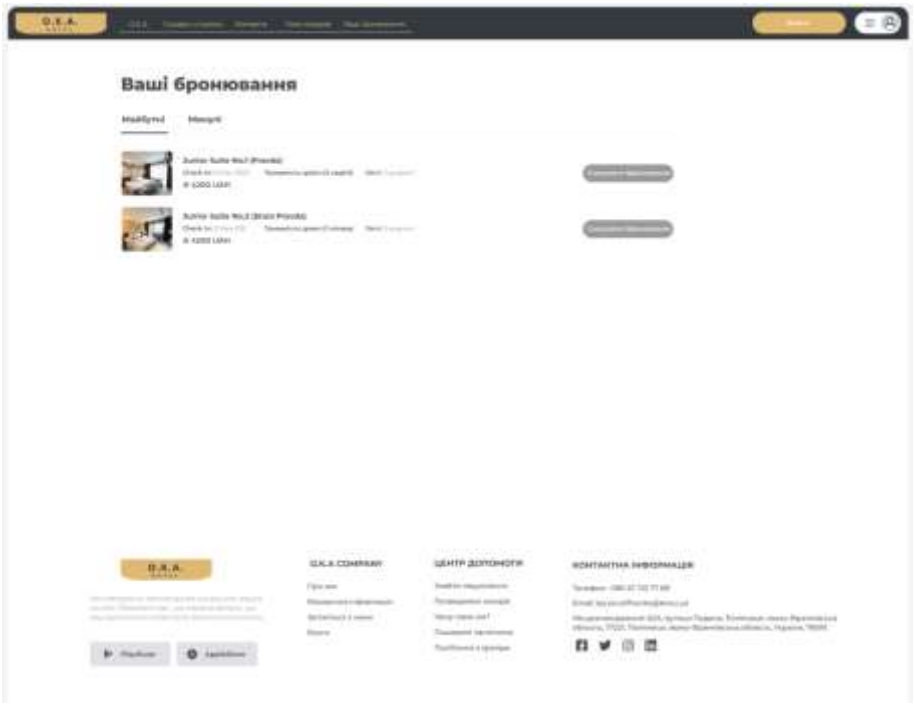


Рисунок В.26 – Сторінка з бронюванням клієнта, майбутні та старі бронювання

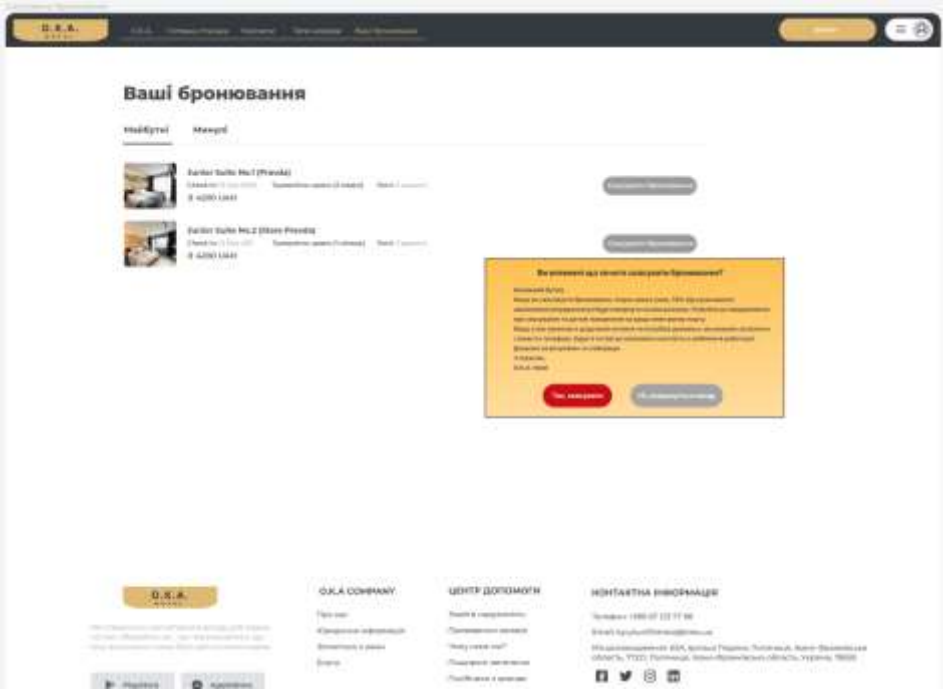


Рисунок В.27 – Повідомлення про скасування бронювання номеру

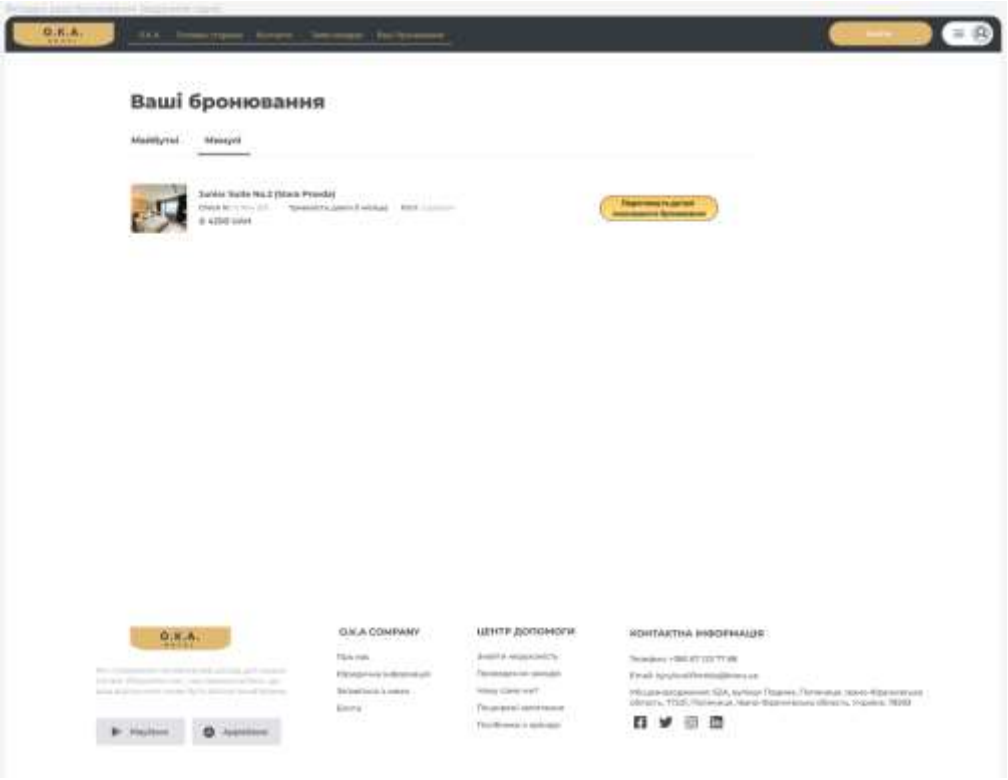


Рисунок В.28 – Сторінка бронювань клієнта зі старими бронюваннями

Скасоване бронювання буде відображатися в минулих та можна передивитися деталі скасованого бронювання.

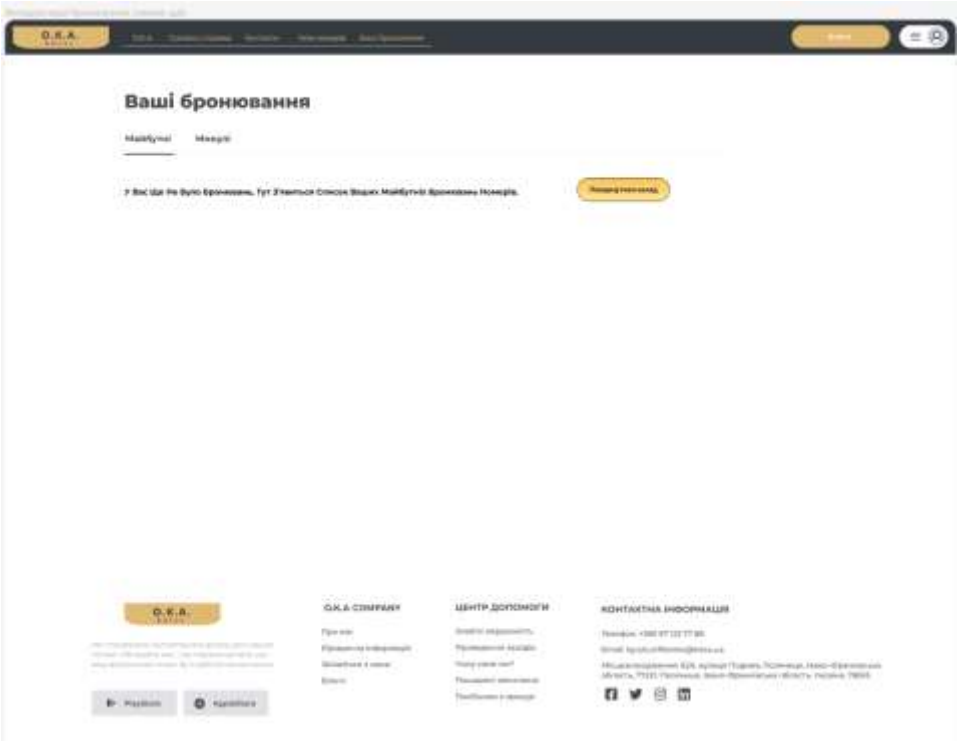


Рисунок В.29 – Вигляд сторінки бронювань користувача, коли в нього ще їх не було

ДОДАТОК Г

Результат перетворення ER-моделі в модель даних СУБД MySQL

Згенерований код:

```
CREATE SCHEMA IF NOT EXISTS `hotel` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `hotel` ;
-- -----------------------------------------------------
-- Table `hotel`.`authorities`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`authorities` (
  `name` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`room_types`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`room_types` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `max_price` INT NOT NULL,
  `min_price` INT NOT NULL,
  `name` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 6
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`rooms`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`rooms` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `double_beds` INT NOT NULL,
  `floor` INT NOT NULL,
  `max_number_of_people` INT NOT NULL,
  `price` INT NOT NULL,
  `room_number` INT NOT NULL,
  `single_beds` INT NOT NULL,
  `types_id` BIGINT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `FK_rooms_room_types_id` (`types_id` ASC) VISIBLE,
  CONSTRAINT `FK_rooms_room_types_id`
    FOREIGN KEY (`types_id`)
    REFERENCES `hotel`.`room_types` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`users`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`users` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(50) NULL DEFAULT NULL,
  `first_name` VARCHAR(25) NULL DEFAULT NULL,
  `last_name` VARCHAR(25) NULL DEFAULT NULL,
  `login` VARCHAR(50) NOT NULL,
  `password` VARCHAR(250) NOT NULL,
  `phone_number` VARCHAR(25) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 12
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`room_reservations`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`room_reservations` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `check_in_date` DATETIME(6) NULL DEFAULT NULL,
  `check_out_date` DATETIME(6) NULL DEFAULT NULL,
  `end_price` INT NOT NULL,
  `number_of_people` INT NOT NULL,
  `paid` BIT(1) NULL DEFAULT NULL,
  `reservation_end_date` DATETIME(6) NOT NULL,
  `reservation_start_date` DATETIME(6) NOT NULL,
  `verified` BIT(1) NULL DEFAULT NULL,
  `rooms_id` BIGINT NOT NULL,
  `users_id` BIGINT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `FK_room_reservations_users_id` (`users_id` ASC) VISIBLE,
  INDEX `FK_room_reservations_rooms_id` (`rooms_id` ASC) VISIBLE,
```

```
  CONSTRAINT `FK_room_reservations_rooms_id`
    FOREIGN KEY (`rooms_id`)
    REFERENCES `hotel`.`rooms` (`id`),
  CONSTRAINT `FK_room_reservations_users_id`
    FOREIGN KEY (`users_id`)
    REFERENCES `hotel`.`users` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 13
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`financial_reports`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`financial_reports` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `report_date` DATE NOT NULL,
  `total_income` DECIMAL(15,2) NOT NULL,
  `total_expense` DECIMAL(15,2) NOT NULL,
  `net_profit` DECIMAL(15,2) GENERATED ALWAYS AS ((`total_income` - `total_expense`)) STORED,
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_financial_reports_room_reservations1`
    FOREIGN KEY (`id`)
    REFERENCES `hotel`.`room_reservations` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`services`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`services` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `description` TEXT NULL DEFAULT NULL,
  `price` DECIMAL(10,2) NOT NULL,
  `available` BIT(1) NOT NULL DEFAULT b'1',
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_services_room_reservations1`
    FOREIGN KEY (`id`)
    REFERENCES `hotel`.`room_reservations` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`users_authorities`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`users_authorities` (
  `users_id` BIGINT NOT NULL,
  `authorities_name` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`users_id`, `authorities_name`),
  INDEX `FK_users_authorities_authorities_name` (`authorities_name` ASC) VISIBLE,
  CONSTRAINT `FK_users_authorities_authorities_name`
    FOREIGN KEY (`authorities_name`)
    REFERENCES `hotel`.`authorities` (`name`),
  CONSTRAINT `FK_users_authorities_users_id`
    FOREIGN KEY (`users_id`)
    REFERENCES `hotel`.`users` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
-- -----------------------------------------------------
-- Table `hotel`.`users_log`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`users_log` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `users_id` BIGINT NOT NULL,
  `msg` VARCHAR(255) NOT NULL,
  `time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  INDEX `users_id` (`users_id` ASC) VISIBLE,
  CONSTRAINT `users_log_ibfk_1`
    FOREIGN KEY (`users_id`)
    REFERENCES `hotel`.`users` (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
USE `hotel`
-- -----------------------------------------------------
-- Placeholder table for view `hotel`.`count_of_reservetion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`count_of_reservetion` (`first_name` INT, `COUNT(*)` INT);
-- -----------------------------------------------------
-- Placeholder table for view `hotel`.`count_room_of_type`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `hotel`.`count_room_of_type` (`name` INT, `count` INT);
-- -----------------------------------------------------
-- procedure count_of_reservetion
```

```
DELIMITER $$
USE `hotel`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `count_of_reservetion`()
BEGIN
  DECLARE p_type VARCHAR(40);
  DECLARE p_count INT(10) UNSIGNED;
  DECLARE done INT DEFAULT 0;
  DECLARE product CURSOR FOR
    SELECT first_name, COUNT(*) FROM users JOIN room_reservations ON (users.id = room_reservations.users_id) GROUP BY first_name;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'  SET done = 1; done = 1
  CREATE TABLE IF NOT EXISTS count_of_reservetion (
                                    user VARCHAR(40),
                                    count INT(10) UNSIGNED
  );
  TRUNCATE count_of_reservetion;
  OPEN product;
  REPEAT FETCH product INTO p_type, p_count;
  IF NOT done THEN
    INSERT INTO count_of_reservetion SET user = p_type, count = p_count;
  END IF;
  UNTIL done END REPEAT;
  CLOSE product;

  SELECT * FROM count_of_reservetion;
END$$
DELIMITER ;
-- ----------------------------------------------------
-- procedure get_user_roles_by_first_name
-- ----------------------------------------------------
DELIMITER $$
USE `hotel`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_user_roles_by_first_name`(IN in_name VARCHAR(50))
BEGIN
  SELECT authorities.name FROM users
                JOIN users_authorities ON(users_authorities.users_id = users.id)
                JOIN authorities ON(users_authorities.authorities_name = authorities.name)
  WHERE first_name = in_name;
END$$
DELIMITER ;
-- ----------------------------------------------------
-- View `hotel`.`count_of_reservetion`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `hotel`.`count_of_reservetion`;
USE `hotel`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `hotel`.`count_of_reservetion` AS select `hotel`.`users`.`first_name` AS `first_name`,count(0) AS `COUNT(*)` from (`hotel`.`users` join `hotel`.`room_reservations` on((`hotel`.`users`.`id` = `hotel`.`room_reservations`.`users_id`))) group by `hotel`.`users`.`first_name`;
-- ----------------------------------------------------
-- View `hotel`.`count_room_of_type`
-- ----------------------------------------------------
DROP TABLE IF EXISTS `hotel`.`count_room_of_type`;
USE `hotel`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `hotel`.`count_room_of_type` AS select `hotel`.`room_types`.`name` AS `name`,count(0) AS `count` from (`hotel`.`room_types` join `hotel`.`rooms` on((`hotel`.`room_types`.`id` = `hotel`.`rooms`.`types_id`))) group by `hotel`.`room_types`.`name`;
USE `hotel`;
DELIMITER $$
USE `hotel`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `hotel`.`insert_users`
AFTER INSERT ON `hotel`.`users`
FOR EACH ROW
BEGIN
  INSERT INTO users_log Set msg = 'insert', row_id = NEW.id;
END$$
USE `hotel`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `hotel`.`update_users`
AFTER UPDATE ON `hotel`.`users`
FOR EACH ROW
BEGIN
  INSERT INTO users_log Set msg = 'update', row_id = NEW.id;
END$$
DELIMITER ;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```