

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

**ПОГОДЖЕНО
Декан факультету**

_____ **Ігор БОЛБОТ**
(підпис)

« ____ » _____ 2026 р.

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних наук**

_____ **Белла ГОЛУБ**
(підпис)

« ____ » _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система обліку апартаментів у готелі»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор

_____ **Роман РУДЕНСЬКИЙ**
(підпис)

**Керівник бакалаврської
кваліфікаційної роботи**
асистент

_____ **Ярослав ГОРДІЙ**
(підпис)

Консультант
к.т.н., доцент

_____ **Белла ГОЛУБ**
(підпис)

Виконав

_____ **Дмитро ПРОЦЕНКО**
(підпис)

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

**ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук**

К.Т.Н., доцент _____ Белла ГОЛУБ
(підпис)

« 06 » січня 2026 р.

**ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Проценко Дмитру Вадимовичу**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Інформаційна система обліку апартаментів у готелі»

затверджена наказом від « 06 » січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру « 25 » травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Відомості про автоматизацію процесів бронювання та реєстрації гостей, організацію взаємодії адміністратора з базою даних для формування фінансової звітності.

Перелік питань, що підлягають дослідженню:

- Аналіз предметної області та порівняльне дослідження існуючих систем управління готелями.
- Проєктування архітектури та інтерфейсу керування номерним фондом.
- Програмна реалізація бізнес-логіки бронювання, реєстрації гостей та генерації фінансових документів.
- Тестування функціональних можливостей системи та організація взаємодії компонентів із локальною базою даних.

Дата видачі завдання « 06 » січня 2026 р.

**Керівник бакалаврської
кваліфікаційної роботи**

(підпис)

Ярослав ГОРДІЙ

Консультант

(підпис)

Белла ГОЛУБ

**Завдання взяв до
виконання**

(підпис)

Дмитро ПРОЦЕНКО

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП.....	6
1 СИСТЕМНИЙ АНАЛІЗ ОБЛАСТІ ОБСЛУГОВУВАННЯ КЛІЄНТІВ У ГОТЕЛІ.....	9
1.1 Опис предметної області.....	9
1.2 Моделювання предметної області.....	11
1.3 Огляд інформаційних джерел та існуючих рішень.....	14
1.4 Постановка завдання.....	16
2 ІНФОРМАЦІЙНЕ ЗАБЕСПЕЧЕННЯ СИСТЕМИ.....	19
2.1 Логічна модель даних.....	19
2.2 Вибір системи управління інформаційною базою.....	21
2.3 Створення бази даних.....	23
3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ.....	31
3.1 Організаційна структура програмного забезпечення.....	31
3.2 Вибір інструментарію для створення ППЗ.....	33
3.3 Реалізація інтерфейсної частини.....	35
3.4 Проектування і реалізація сервісів.....	38
3.5 Проектування і реалізація взаємодії з базою даних.....	43
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ.....	47
4.1 Діаграма розгортання.....	47
4.2 Апаратні та програмні вимоги до системи.....	49
4.3 Склад інсталяційного пакету.....	50
4.4 Тестування системи.....	53
ВИСНОВОК.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

ІС - інформаційна система;

ПК - персональний комп'ютер;

ППЗ - прикладне програмне забезпечення;

СУБД - система управління базами даних;

ACID (Atomicity, Consistency, Isolation, Durability) - набір транзакційних властивостей реляційних СУБД: атомарність, узгодженість, ізолюваність, довговічність;

ADO.NET - бібліотека та технологія безпосереднього доступу до баз даних у платформі .NET;

ER-діаграма (Entity-Relationship diagram) - діаграма «сутність-зв'язок», призначена для побудови логічних моделей даних;

IDE (Integrated Development Environment) - інтегроване середовище розробки програмного забезпечення;

PMS (Property Management System) - автоматизована система управління готелем та операційного менеджменту номерного фонду;

SaaS (Software as a Service) - модель програмного забезпечення, за якої хмарні сервіси надаються користувачам через інтернет-браузер;

SQL (Structured Query Language) - декларативна мова структурованих запитів, що використовується для опису та управління реляційними структурами;

TCP/IP (Transmission Control Protocol / Internet Protocol) - базовий стек мережеских протоколів для забезпечення зв'язку між клієнтськими пристроями та сервером баз даних;

T-SQL (Transact-SQL) - процедурне розширення мови SQL від компанії Microsoft, що використовується для написання тригерів та збережених процедур;

UML (Unified Modeling Language) - уніфікована мова об'єктно-орієнтованого моделювання для графічного опису бізнес-процесів та архітектури систем;

WPF (Windows Presentation Foundation) - програмна технологія екосистеми .NET для проектування настільних графічних інтерфейсів користувача;

XAML (Extensible Application Markup Language) - декларативна мова розмітки візуальних компонентів, що застосовується для створення форм.

ВСТУП

Актуальність розробки. У сучасних умовах розвиток готельного бізнесу нерозривно пов'язаний із впровадженням нових цифрових технологій. Звичайні методи управління готельними активами, засновані на паперових журналах або розрізнених електронних таблицях стають малоефективними через високий ризик виникнення помилок, таких як, втрата даних, дублювання записів та виникнення ситуацій подвійного бронювання. Стійкість функціонування готельного бізнесу сьогодні безпосередньо залежить від якості автоматизації внутрішніх бізнес-процесів.

Практична цінність системи полягає в тому, що вона звільняє персонал від рутинної «паперової» роботи. В умовах реального готелю, коли на стойці реєстрації одночасно з'являється кілька гостей, на вагу кожна секунда. Програма має бути побудована так, щоб адміністратор міг оформити заїзд у кілька кліків, не відволікаючись на складні налаштування.

Впровадження ІС дозволить централізувати управління номерним фондом та автоматизувати рутинні завдання, тим самим значно прискорити операційну роботу персоналу готелю. Розробка системи, здатної працювати в межах локальної мережі без обов'язкового доступу до Інтернету є критично важливою для забезпечення безперебійності сервісу в умовах нестабільного енергопостачання та мережевих обмежень.

Мета розробки. Метою розробки є проєктування та програмна реалізація інформаційної системи обліку апартаментів, що дозволить автоматизувати функції операційного менеджменту номерного фонду та фінансової звітності.

Методи та технології. Для виконання поставленої мети було використано такий перелік технологій:

- Об'єктно-орієнтована мова C# та платформа .NET, яка забезпечує типізацію та відповідає за логіку додатку.
- Технологія інтерфейсу WPF для створення інтерфейсу користувача.
- Microsoft SQL Server – СУБД для зберігання даних що гарантує виконання принципів ACID.
- Доступ до даних реалізовано через низькорівневу бібліотеку ADO.NET для максимального контролю продуктивності запитів та використання збережених процедур на стороні сервера.
- Для підвищення зручності і привабливості інтерфейсу використано бібліотеку MahApps.Metro.
- Застосовано бібліотеку LiveCharts для графічного оформлення аналітичної звітності.

Апробація програмного додатку. Було опубліковано результати дипломної роботи у тезах виступу на VII Всеукраїнській науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем '2026»: Проценко Д.В., науковий керівник Голуб Б.Л. Інформаційна система обліку апартаментів у готелі.

Структура записки. Пояснювальна записка складається з 64 сторінок, 20 використаних джерел та семи додатків. Основна частина розділена на такі 4 розділи:

1. Системний аналіз предметної області та актуальність автоматизації обліку апартаментів. На основі порівняння з комерційними аналогами Opera PMS та Cloudbeds було сформульовано функціональні й нефункціональні вимоги до системи. Розроблено комплекс моделей UML, включаючи діаграми прецедентів, діяльності та послідовності.
2. Проєктування реляційної бази даних та обґрунтування вибору СУБД. Описано побудову логічної структури ER-діаграми та проведення нормалізації таблиць до третьої нормальної форми. На стороні сервера мовою T-SQL створено збережені процедури і тригери.
3. Архітектура та програмна реалізація настільного програмного додатку мовою C#. Описано створення графічного інтерфейсу користувача за допомогою WPF та XAML. Наведено принципи взаємодії з базою даних через ADO.NET та інтеграцію LiveCharts для візуалізації звітності.
4. Практичні рекомендації щодо фізичного розгортання, супроводу та оцінювання якості створеного програмного продукту. Побудовано діаграму розгортання, визначено апаратні вимоги до обладнання та описано склад інсталяційного пакету. Наведено результати ручного тестування всіх модулів системи.

1 СИСТЕМНИЙ АНАЛІЗ ОБЛАСТІ ОБСЛУГОВУВАННЯ КЛІЄНТІВ У ГОТЕЛІ

1.1 Опис предметної області

Функціонування сучасного готелю - це не просто надання спального місця, а комплексний сервіс, що складається з багатьох дрібних, але важливих операційних процесів. Предметною областю даної роботи є діяльність готельного підприємства, яка зосереджена навколо двох головних завдань: ефективного використання житлової площі та забезпечення якісного обслуговування клієнтів.

Для того, щоб зрозуміти архітектуру майбутньої інформаційної системи, необхідно детально розібрати існуючий порядок роботи закладу та визначити основні сутності, що взаємодіють між собою.

1.1.1 Основні поняття та термінологія. Для чіткого аналізу предметної області введемо основні визначення, що описують діяльність готелю:

Апартаменти (номер) — це базова одиниця житлового фонду готелю, яка має свій номер, категорію наприклад, Економ чи Люкс, та визначену вартість проживання за одну добу.

Клієнт (гість) — це фізична особа, яка звертається до працівника готелю з метою отримання послуг проживання та надає свої персональні дані для реєстрації.

Запит на бронювання — процес, під час якого клієнт подає попередню заявку на закріплення за ним номера на певні дати, використовуючи засоби зв'язку (телефон, пошта) або особистий візит.

Реєстрація заїзду — процедура підтвердження особи клієнта працівником готелю, під час якої фіксується точний час прибуття та передаються ключі від номера.

Рахунок гостя — документ або запис, у якому накопичується інформація про всі фінансові витрати клієнта: вартість основних днів проживання та додаткові сервіси.

Статус номера — поточний фізичний стан апартаментів (вільний, зайнятий гостем, потребує прибирання або перебуває на ремонті).

1.1.2. Опис традиційного життєвого циклу обслуговування. Зазвичай, без використання спеціалізованого програмного забезпечення, процес обслуговування базується на ручному веденні журналів обліку та візуальному контролі. Цей процес складається з таких етапів:

Етап 1. Прийом запиту та перевірка доступності. Коли клієнт звертається до готелю, адміністратор повинен вручну перевірити книгу обліку або настільну таблицю-шахматку на відповідні дати. Якщо вільні апартаменти знайдено, у журналі робиться помітка олівцем про бронювання. На цьому етапі висока ймовірність помилки, якщо адміністратор забуде зробити запис або випадково запише двох людей на одне місце.

Етап 2. Зустріч та реєстрація клієнта. Під час приїзду гостя адміністратор перевіряє паспортні дані та вписує їх у паперову анкету або загальну книгу реєстрації. Номер отримує статус «зайнятий», що зазвичай відмічається на спеціальній дошці ключів або в журналі. Весь цей процес займає значний час через необхідність ручного заповнення форм.

Етап 3. Період проживання та облік послуг. Протягом перебування гостя в готелі можуть виникати ситуації, що потребують фіксації: замовлення сніданків, використання міні-бару чи послуг пральні. У традиційній схемі адміністратор записує ці послуги на окремих листках або в картці гостя, що зберігається на стійці реєстрації. Це створює ризик того, що під час виїзду деякі послуги можуть бути не враховані в підсумковому рахунку.

Етап 4. Розрахунок та підготовка номера до нового циклу. При виїзді адміністратор підсумовує всі дні проживання та додаткові витрати за допомогою калькулятора. Після оплати номер вважається звільненим, про що робиться запис у журналі. Далі береться технічна пауза на прибирання, після якої працівник

повідомляє адміністратора про готовність номера до наступного клієнта, і статус знову змінюється на «вільний».

1.1.3. Виклики та проблеми ручного управління. Системний аналіз існуючого ручного процесу виявляє такі слабкі місця:

- 1) проблема людського фактора: високий ризик втрати інформації, помилок у розрахунках або виникнення ситуацій подвійного бронювання;
- 2) низька швидкість обслуговування: клієнти змушені чекати, поки адміністратор знайде потрібну інформацію в паперових записах;
- 3) відсутність швидкої аналітики: для того, щоб дізнатися прибуток за місяць або завантаженість готелю, керівнику потрібно вручну переглядати всі журнали та робити складні підрахунки.

Таким чином, аналіз підтверджує, що звичайне управління готелем за допомогою паперових носіїв стикається з високими ризиками втрати даних та помилками. Описані процеси є занадто проблематичними для ручного контролю що створює затримки в обслуговуванні та унеможливорює швидке отримання фінансової аналітики. Отже, розробка автоматизованої інформаційної системи є об'єктивною необхідністю для підвищення точності даних та переходу до сучасних стандартів сервісу.

1.2 Моделювання предметної області

Результатом системного аналізу стала розробка моделей UML-діаграм, які перетворюють розмиті вимоги до обслуговування клієнтів на чіткі алгоритми роботи.

1.2.1 Діаграма прецедентів. Діаграма прецедентів демонструє функціональні межі системи та визначає, які саме сервіси готелю будуть автоматизовані для кожного типу користувачів. Вона фіксує результати розподілу прав доступу та обов'язків персоналу.

Відповідно до розробленої моделі, у програмі взаємодіють три головні актори:

- адміністратор - це основний працівник, який користується програмою щодня. Він шукає вільні номери, оформлює проживання гостей, змінює статус кімнат та друкує звіти
- гість - людина, яка звертається із запитом на проживання та надає свої дані для реєстрації.
- власник готелю - користувач, якому важливо бачити загальну картину: наскільки заповнений готель та який прибуток він приносить.

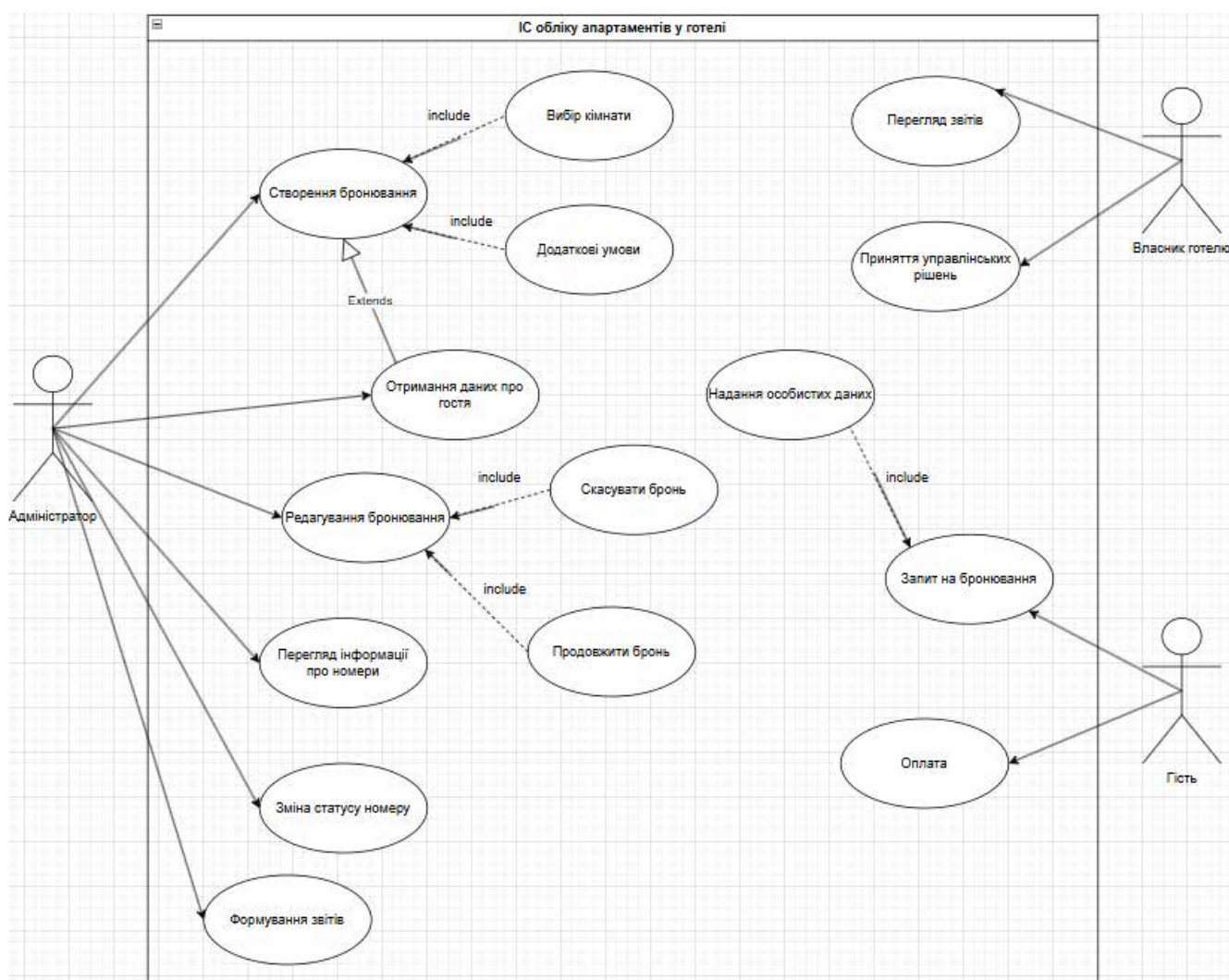


Рис. 1 Діаграма прецедентів предметної області

1.2.2 Діаграма діяльності. Діаграма діяльності представлена у додатку А і демонструє логічну послідовність кроків процесу від запиту на бронювання до фінального підтвердження. Вона візуалізує результати аналізу робочого циклу адміністратора.

Основні результати моделювання логіки процесів наведено у табл. 1.

Таблиця 1

Назва дії	Що демонструє
Запит на бронювання	Початок процесу бронювання за вимогою гостя.
Перегляд переліку номерів	Доступ до актуального стану всіх номерів.
Перевірка статусу	Умова розгалуження: якщо номер зайнятий, треба шукати вільний.
Вибір умов та дат	Вибір можливих термінів перебування гостя та додаткових побажань щодо обслуговування.
Заповнення даних	Процес надання гостем його даних та внесення їх до реєстру.
Підтвердження та оплата	Завершення процесу бронювання та закріплення номера за клієнтом.

1.2.3 Діаграма послідовності. Діаграма послідовності представлена у додатку Б і наочно показує послідовність дій між учасниками процесу: гостем, адміністратором та електронним журналом готелю. Вона показує як саме передаються повідомлення та запити під час реального обслуговування у часовому порядку, дозволяючи зрозуміти шлях проходження інформації від звернення клієнта до успішного створеного бронювання.

1.3 Огляд інформаційних джерел та існуючих рішень

На сьогоднішній день ринок програмного забезпечення пропонує велику кількість автоматизованих систем управління готелями PMS, які дозволяють автоматизувати практично всі аспекти готельного бізнесу. Кожне з таких рішень має свою специфіку, орієнтуючись на різні масштаби підприємств: від невеликих приватних апартаментів до міжнародних готельних мереж. Для того, щоб обґрунтувати розробку власної системи, необхідно розглянути функціональні можливості та обмеження найбільш популярних сучасних програмних продуктів.

Для детального огляду було обрано два найбільш показові рішення: систему корпоративного рівня Opera PMS та сучасну хмарну платформу Cloudbeds.

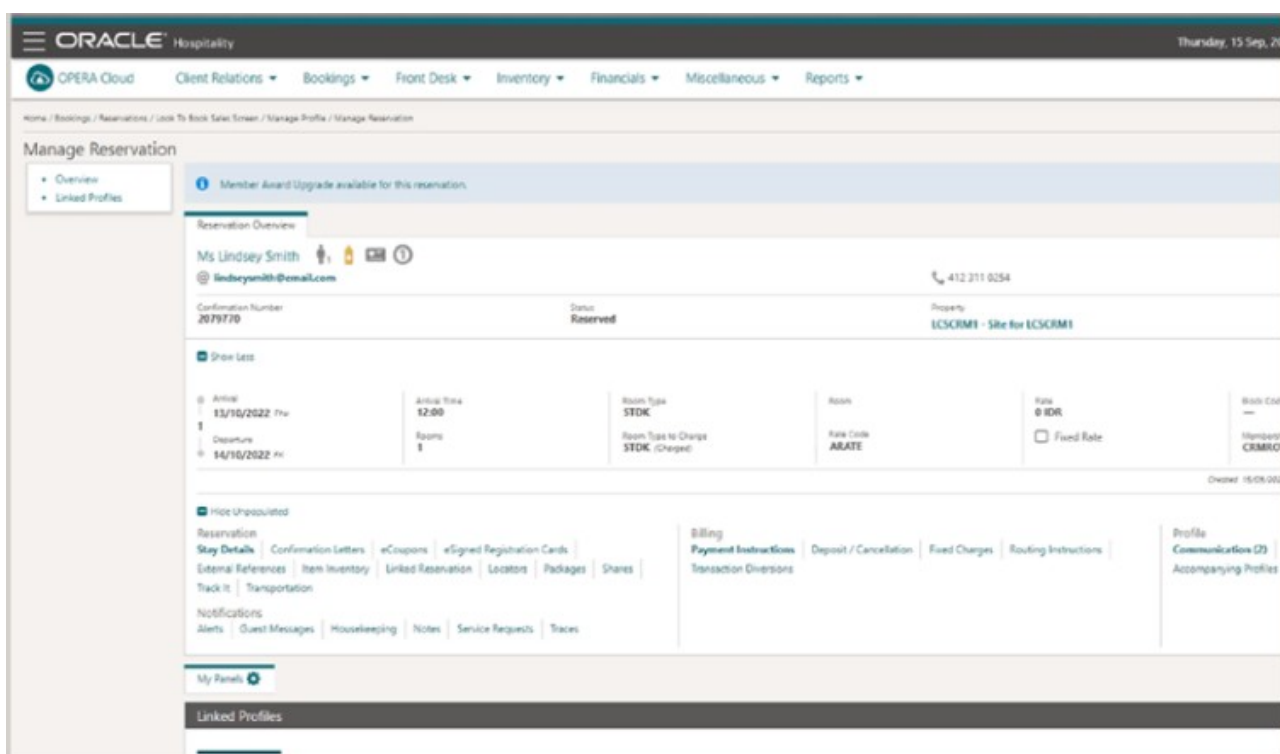


Рис. 2 OPERA PMS від Oracle

1) Opera PMS від Oracle.

Ця система є визнаним світовим стандартом і використовується найбільшими готельними мережами світу. Вона розроблена для комплексного управління великими об'єктами.

Функціонал: Дозволяє керувати профілями гостей, бронюванням, розподілом номерів, фінансовими розрахунками та інтегруватися з глобальними системами продажів номерів.

Переваги: Максимальний набір функцій, дуже високий рівень захисту даних та можливість налаштування під складні бізнес-процеси великих підприємств.

Недоліки: Система є дуже дорогою у впровадженні та підтримці. Її інтерфейс складний для швидкого засвоєння, а більшість функцій залишаються незадіяними у малих готелях, що робить її використання економічно недоцільним для малого бізнесу.

2) Cloudbeds

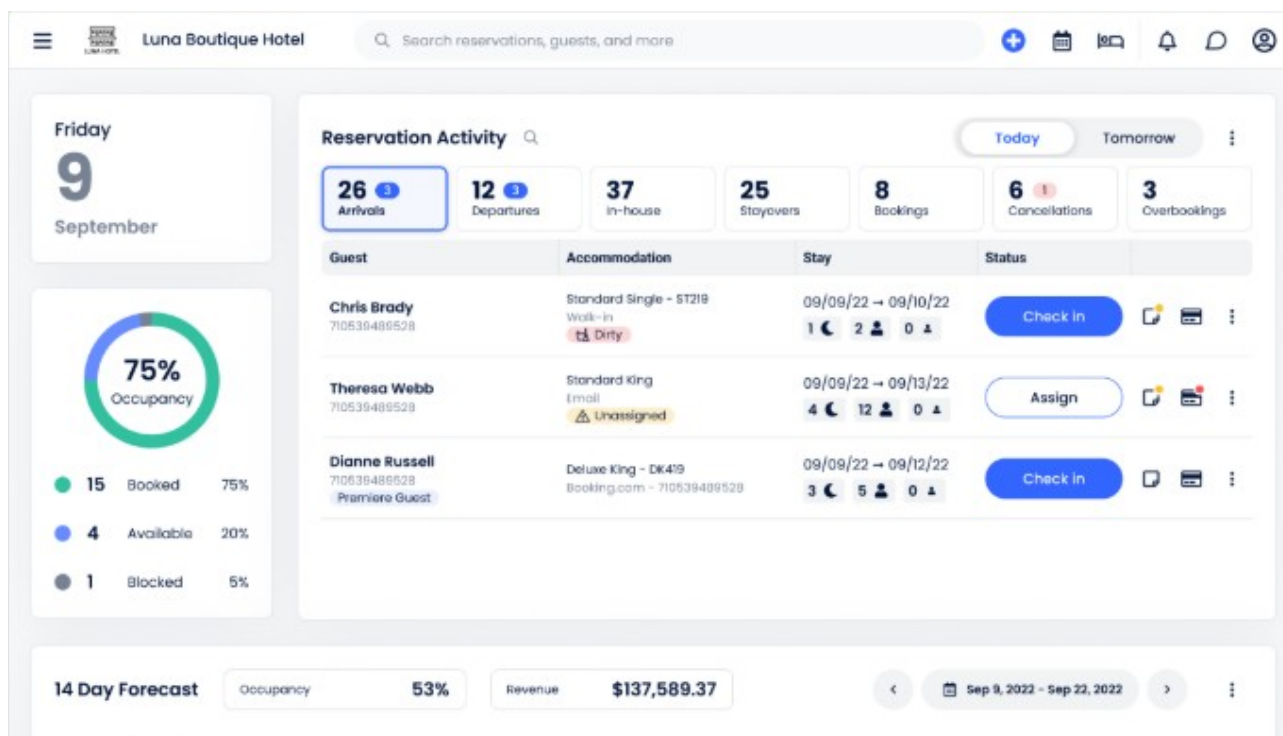


Рис. 3 Cloudbeds

Це одна з найбільш прогресивних хмарних систем SaaS, яка працює через інтернет-браузер і орієнтована на готелі середнього та малого сегментів.

Функціонал. Включає зручний календар для управління номерами, модуль для бронювання прямо з сайту готелю та систему синхронізації з майданчиками типу Booking.com чи Airbnb.

Переваги. Не потребує встановлення на комп'ютер або купівлі власних серверів. Програма має сучасний зрозумілий інтерфейс, що дозволяє персоналу швидко почати роботу без тривалого навчання.

Недоліки. Головним мінусом є повна залежність від стабільності інтернету, якщо зв'язок зникне, робота адміністратора буде паралізована. Також дані клієнтів зберігаються на сторонніх серверах розробника, а щомісячна абонплата створює постійне фінансове навантаження.

Огляд існуючих систем показав, що професійні рішення типу Opera PMS є занадто складними та дорогими для невеликих готельних комплексів, а хмарні сервіси, такі як Cloudbeds, мають ризики через залежність від зовнішньої мережі та сторонніх серверів тим самим підтверджуючи доцільність розробки локальної інформаційної системи. Впровадження такої системи дозволить поєднати простоту управління з повною автономністю та безпекою збереження даних безпосередньо на локальному обладнанні готелю.

1.4 Постановка завдання

Основна мета розробки: створення автоматизованої інформаційної системи, яка буде забезпечувати централізований облік апартаментів, реєстрацію клієнтів та управління базою номерів. Розроблювана система повинна максимально мінімізувати кількість помилок, пов'язаних із людським фактором, та підвищити швидкість обслуговування гостей.

1.4.1 Функціональні вимоги. Перелік функціональних вимог представлений нижче.

1. Управління номерним фондом

- Система повинна забезпечувати ведення повного переліку номерів готелю.
- Для кожного номера необхідно зберігати детальні характеристики: унікальний номер кімнати, тип, кількість місць, ціну за добу та опис зручностей.
- Програма має підтримувати динамічне оновлення статусів номерів: «вільний», «зайнятий», «заброньований».

2. Реєстрація та обробка бронювань

- Адміністратор повинен мати змогу здійснювати швидкий пошук та перевірку доступності номерів за заданими датами, типом та бюджетом.
- Система має забезпечувати реєстрацію нових бронювань із прив'язкою до конкретного клієнта та обраного періоду проживання.
- Обов'язковою є функція автоматичного контролю перевантажень для запобігання подвійному бронюванню одного номера на ті самі дати.
- Користувач повинен мати можливість редагувати або скасовувати існуючі бронювання.

3. Обслуговування заїздів та виїздів

- При заїзді система має забезпечувати реєстрацію особистих даних гостя (ПІБ, паспортні дані, контактна інформація) та зміну статусу номера на «зайнятий».
- Під час оформлення виїзду програма повинна автоматично розраховувати підсумкову суму до оплати на основі тривалості проживання та вартості додаткових послуг.
- Після виїзду статус номера має змінюватися відповідно до внутрішніх процедур.

4. Фінансовий облік та звітність

- Формування рахунків, електронних квитанцій та фіскальних чеків для гостей.
- Облік різних типів оплати: готівкових та безготівкових.
- Генерація аналітичних звітів за обраний період щодо завантаженості номерів, отриманих доходів та статистики бронювань.

1.4.2 Нефункціональні вимоги. Перелік нефункціональних вимог представлений нижче.

- Авторизація користувачів у системі за допомогою індивідуального логіна та пароля.
- Впровадження ролей, що визначає доступний функціонал для кожного працівника.
- Захист персональних даних клієнтів відповідно до законодавства України та міжнародних стандартів.
- Забезпечення цілісності даних у БД та запобігання помилкам, що призводять до втрати інформації.
- Забезпечення зручного та зрозумілого середовища користувача для швидкої обробки запитів.

2 ІНФОРМАЦІЙНЕ ЗАБЕСПЕЧЕННЯ СИСТЕМИ

2.1 Логічна модель даних

Логічна модель даних описує поняття предметної області, їх взаємозв'язок, а також обмеження по даних, що накладаються предметною областю. Такі моделі існують для того, щоб визначити елементи й функціональність системи, які необхідно реалізувати.

Для реалізації логічної моделі даних було створено ER-діаграму. Вона дозволяє систематизувати всю інформацію, яка є необхідною для функціонування готелю, від технічних характеристик апартаментів до персональних даних клієнтів та фінансової звітності.

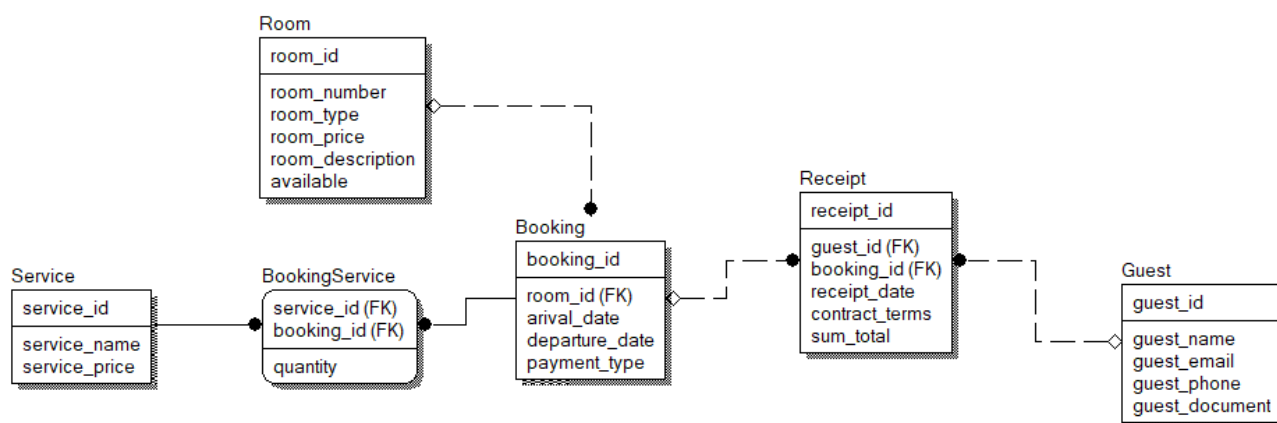


Рис. 4 Логічна модель даних

Відповідно до розробленої схеми, логічна модель складається з шести взаємопов'язаних сутностей які представлені нижче.

1. **Room (Номери):** зберігає дані про житловий фонд готелю. Кожен запис містить унікальний ідентифікатор (`room_id`), фізичний номер кімнати (`room_number`), категорію (`room_type`), вартість проживання за добу (`room_price`), текстовий опис (`room_description`) та поточну доступність (`available`).
2. **Guest (Клієнти):** сутність для обліку осіб, які користуються послугами готелю. Вона включає персональні дані: ПІБ (`guest_name`), електронну пошту

(guest_email), номер телефону (guest_phone) та дані документа, що посвідчує особу (guest_document).

3. **Booking (Бронювання)**: центральна сутність, що пов'язує клієнта з номером на певний період. Вона фіксує дати заїзду (arrival_date) та виїзду (departure_date), а також обраний тип оплати (payment_type).
4. **Service (Додаткові послуги)**: містить довідник послуг, які готель надає крім проживання такі як, сніданок, прання тощо. Кожна послуга має назву (service_name) та встановлену ціну (service_price).
5. **BookingService (Послуги замовлення)**: проміжна таблиця, яка дозволяє додавати до одного бронювання кілька різних послуг у необхідній кількості (quantity), реалізуючи зв'язок багато до багатьох між бронюваннями та послугами;
6. **Receipt (Квитанція/Рахунок)**: сутність для фіксації фінансових розрахунків. Вона пов'язана з конкретним гостем та бронюванням, містить дату формування (receipt_date), умови контракту (contract_terms) та підсумкову суму до оплати (sum_total).

Зв'язки між сутностями. Зв'язок між сутностями Room та Booking має тип один до багатьох (1:N), тому що один номер може бути заброньований багато разів у різні періоди часу. Зв'язок між Guest та Receipt також є 1:N, що дозволяє зберігати історію всіх рахунків, виставлених конкретному клієнту за час його візитів. Зв'язок між таблицями Receipt та Booking реалізований за допомогою зовнішнього ключа booking_id, який вказує на конкретний запис про проживання. Це дозволяє системі автоматично узагальнювати дані про термін перебування та вартість наданих послуг для формування коректного підсумкового рахунку. Сутності Booking та Service взаємодіють через проміжну таблицю BookingService, яка спеціалізується на наданні додаткового сервісу кожному гостю.

2.2 Вибір системи управління інформаційною базою

Процес проектування інформаційної системи обліку апартаментів готелю вимагає приділення особливої уваги вибору СУБД, оскільки саме вона є основою для надійного збереження всієї важливої інформації. Ефективність роботи готелю та його прибутковість безпосередньо залежить від швидкості доступу до даних, їхньої цілісності та здатності системи обробляти запити без помилок. При виборі програмного рішення для керування базою даних було враховано комплекс факторів, серед яких ключовими стали продуктивність, масштабованість, рівень безпеки та, що не менш важливо, легкість інтеграції з обраним інструментарієм програмування.

На початковому етапі аналізу розглядалися різні варіанти реляційних систем, результати якого представлені в табл. 2.

Таблиця 2

Параметр	SQLite	PostgreSQL	MS SQL Server
Архітектура	Файлова	Клієнт-сервер	Клієнт-сервер
Макс. обсяг БД	До 140 ТБ	Необмежено	До 10 ГБ у Express версії
Тип ліцензії	Безкоштовна	Безкоштовна	Безкоштовна
Інтеграція	Через сторонні провайдери	Через драйвер Npgsql	Вбудована
Підтримка процедур	Обмежена	Повна (PL/pgSQL)	Повна (T-SQL)

Наприклад, такі популярні СУБД з відкритим кодом, як MySQL або PostgreSQL, демонструють високу стабільність у веб-орієнтованих проектах, проте їхнє впровадження у настільні додатки на платформі .NET часто вимагає

Особлива увага при виборі SQL Server приділялася підтримці принципів ACID, що гарантує надійність виконання транзакцій. SQL Server надає потужні механізми блокування та ізоляції транзакцій, що повністю виключає виникнення ситуацій подвійного бронювання. Окрім того, функціональні можливості системи дозволяють перенести частину складної логіки перевірок безпосередньо на рівень бази даних за допомогою збережених процедур та тригерів, що підвищує загальну швидкість роботи додатка та зменшує навантаження на програмний код клієнтської частини.

Також важливим аргументом на користь даної СУБД стала її гнучка модель ліцензування та безпеки. Безкоштовна версія SQL Server Express цілком задовольняє потреби малого готелю, надаючи при цьому професійні засоби захисту інформації та можливість розмежування прав доступу користувачів. У разі масштабування бізнесу архітектура бази даних не потребуватиме змін, система зможе легко перейти на більш потужну редакцію сервера.

Отже, вибір Microsoft SQL Server дозволить створити стійке, безпечне та високопродуктивне інформаційне середовище, яке відповідає всім сучасним вимогам до автоматизації готельного бізнесу.

2.3 Створення бази даних

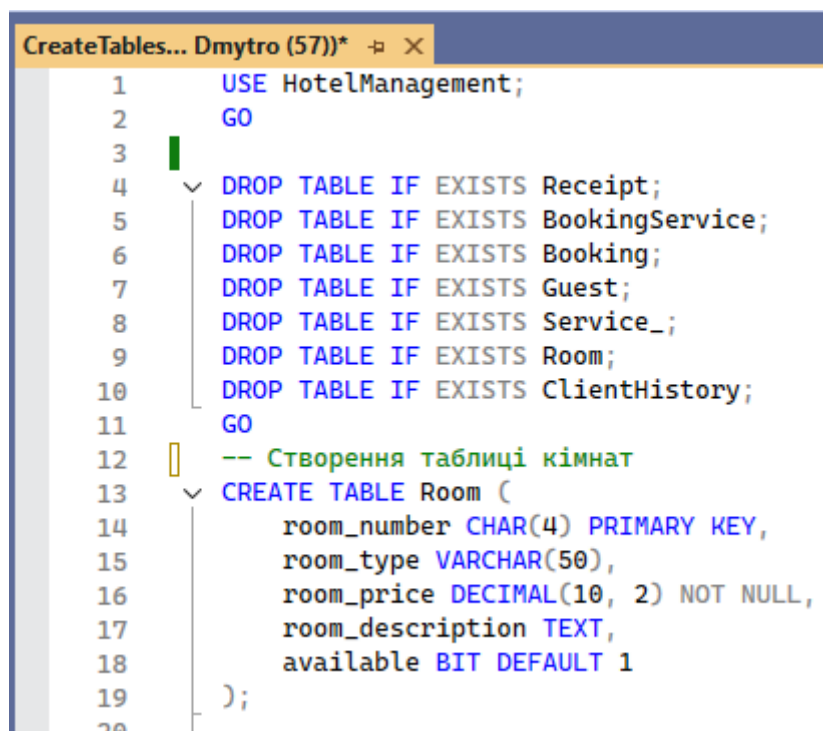
Процес створення інформаційної бази є етапом фізичного проектування, на якому логічна модель даних трансформується у конкретні структури обраної СУБД. На основі розробленої раніше ER-моделі, у середовищі Microsoft SQL Server було створено базу даних «HotelManagement», що містить набір таблиць, процедур та тригерів. Усі створені таблиці пройшли процедуру нормалізації до третьої нормальної форми.

2.3.1 Створення таблиць. Нижче наведено опис та програмний код створення деяких основних таблиць системи. Повний код створення таблиць розміщений в додатку В.

1. Таблиця **Room**: Ця таблиця містить усю інформацію про житлові одиниці. Її основними атрибутами є:

- **room_number** - первинний ключ, фізичний номер кімнати в готелі,

- **room_type** - категорія апартаментів,
- **room_price** - встановлена вартість проживання за добу,
- **room_description** - текстові нотатки про особливості номера,
- **available** - вказує на поточну доступність номера для заселення.



```

1  USE HotelManagement;
2  GO
3
4  DROP TABLE IF EXISTS Receipt;
5  DROP TABLE IF EXISTS BookingService;
6  DROP TABLE IF EXISTS Booking;
7  DROP TABLE IF EXISTS Guest;
8  DROP TABLE IF EXISTS Service_;
9  DROP TABLE IF EXISTS Room;
10 DROP TABLE IF EXISTS ClientHistory;
11 GO
12 -- Створення таблиці кімнат
13 CREATE TABLE Room (
14     room_number CHAR(4) PRIMARY KEY,
15     room_type VARCHAR(50),
16     room_price DECIMAL(10, 2) NOT NULL,
17     room_description TEXT,
18     available BIT DEFAULT 1
19 );
20

```

Рис. 6 Приклад створення таблиць

2. Таблиця **Guest**: Елементи цієї таблиці призначені для ідентифікації осіб, що користуються послугами готелю:

- **guest_id** - унікальний номер гостя в системі;
- **guest_name** - повне прізвище, ім'я та по батькові клієнта;
- **guest_email** та **guest_phone** - контактні дані для зв'язку та розсилки сповіщень;
- **guest_document** - дані паспорта або іншого документа для офіційної реєстрації.

3. Таблиця **Booking**: Це ключова таблиця, яка фіксує сам факт оренди апартаментів.

- **booking_id** - ідентифікатор операції бронювання;
- **room_id** - посилання на конкретну кімнату з таблиці номерів;

- **arrival_date** та **departure_date** - дати заїзду та запланованого виїзду гостя;
- **payment_type** - обраний клієнтом спосіб розрахунку: готівка або картка.

4. Таблиці послуг **Service** та **BookingService**: для реалізації додаткового сервісу використовуються дві таблиці. Таблиця **Service** містить назву та ціну послуги. Проміжна таблиця **BookingService** включає такі елементи як **service_id** та **booking_id** - посилання на відповідну послугу та бронювання, і атрибут **quantity** - кількість наданих одиниць послуги для можливості бронювання одразу кількох однакових послуг клієнтом.

5. Таблиця **Receipt**: Ця сутність генерує фінальні дані для оплати. Її елементами є:

- **receipt_id** - унікальний номер квитанції;
- **guest_id** та **booking_id** - посилання на клієнта та його бронювання для автоматизації виписки рахунку;
- **receipt_date** - дата і час створення фінансового документа;
- **contract_terms** - умови надання послуг або додаткові зауваження до договору;
- **sum_total** - підсумкова сума, розрахована системою.

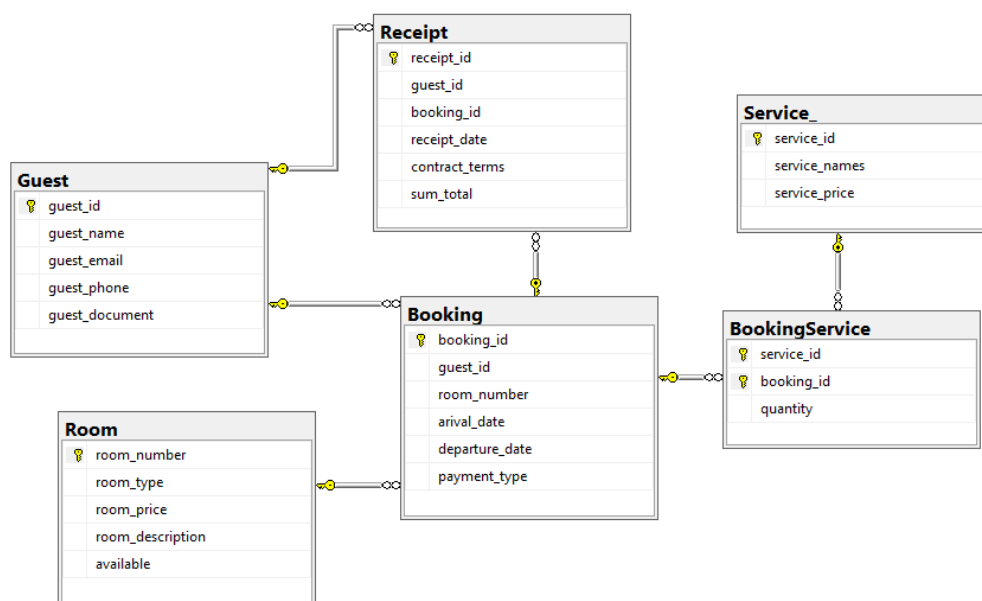


Рис. 7 Фізична модель даних

2.3.2 Створення процедур. Важливим етапом розробки інформаційного забезпечення є винесення частини бізнес-логіки на рівень бази даних за допомогою збережених процедур. Це дозволяє не лише підвищити безпеку, а також прискорити виконання типових запитів, оскільки всі обчислення відбуваються на стороні сервера.

В межах розроблюваної системи обліку апартаментів було розроблено ряд процедур для автоматизації аналітики, розрахунків та коректного видалення даних. Повний код всіх розроблених процедур розміщений в додатку Д.

1) Процедура формування фінансової звітності (**pr_GetRevenueReport**): призначена для надання власнику готелю повної інформації про прибутковість закладу за обраний період. Процедура виконує наступні функції:

- підраховує загальну кількість виписаних чеків та сумарний дохід;
- обчислює середній чек за одну операцію;
- демонструє рейтинг типів номерів, які принесли найбільший прибуток.

2) Процедура автоматичної генерації рахунку (**pr_GenerateFinalReceipt**): представлена на рис. 8. Алгоритм роботи процедури включає:

- автоматичне визначення кількості днів проживання на основі дат заїзду та виїзду;
- підсумовування вартості проживання та всіх додатково замовлених послуг із таблиці BookingService;
- перевірку на дублювання, щоб виключити створення кількох квитанцій для одного бронювання;
- створення фінального запису в таблиці Receipt із присвоєнням статусу «Оплачено повністю».

```

1  USE HotelManagement;
2  GO
3  CREATE OR ALTER PROCEDURE pr_GenerateFinalReceipt
4      @booking_id CHAR(10),
5      @guest_id CHAR(10)
6  AS
7  BEGIN
8      DECLARE @room_sum DECIMAL(10,2);
9      DECLARE @services_sum DECIMAL(10,2);
10     DECLARE @days INT;
11     DECLARE @total DECIMAL(10,2);
12     DECLARE @new_receipt_id CHAR(10);
13
14     -- Генеруємо випадковий ID для чека
15     SET @new_receipt_id = 'R' + LEFT(CAST(ABS(CHECKSUM(NEWID())) AS VARCHAR), 9);
16
17     -- Кількість днів
18     SELECT @days = DATEDIFF(day, arrival_date, departure_date)
19     FROM Booking WHERE booking_id = @booking_id;
20
21     IF @days <= 0 SET @days = 1;
22
23     -- Вартість проживання
24     SELECT @room_sum = r.room_price * @days
25     FROM Booking b JOIN Room r ON b.room_number = r.room_number
26     WHERE b.booking_id = @booking_id;
27
28     -- Вартість послуг
29     SELECT @services_sum = ISNULL(SUM(s.service_price * bs.quantity), 0)
30     FROM BookingService bs
31     JOIN Service_ s ON bs.service_id = s.service_id
32     WHERE bs.booking_id = @booking_id;
33
34     SET @total = @room_sum + @services_sum;
35
36     -- Перевірка на дублювання чеку
37     IF NOT EXISTS (SELECT 1 FROM Receipt WHERE booking_id = @booking_id)
38     BEGIN
39         INSERT INTO Receipt (receipt_id, guest_id, booking_id, receipt_date, contract_terms, sum_total)
40         VALUES (@new_receipt_id, @guest_id, @booking_id, GETDATE(), 'Оплачено повністю', @total);
41
42         SELECT * FROM Receipt WHERE receipt_id = @new_receipt_id;
43     END
44     ELSE
45     BEGIN
46         SELECT * FROM Receipt WHERE booking_id = @booking_id;
47     END
48 END;

```

Рис. 8 Процедура автоматичної генерації рахунку

3) Процедура скасування бронювання (**pr_CancelBooking**): виконує механізм безпечного видалення записів. Перед виконанням команди DELETE процедура перевіряє наявність бронювання в базі. Якщо запис не знайдено, система генерує відповідну помилку, запобігаючи некоректному виконанню операцій адміністратором.

4) Процедура пошуку вільних апартаментів (**pr_SearchAvailableRooms**): допомагає адміністратору для швидкої перевірки стану номерного фонду.

Алгоритм роботи процедури передбачає:

- автоматичну фільтрацію записів таблиці Room;

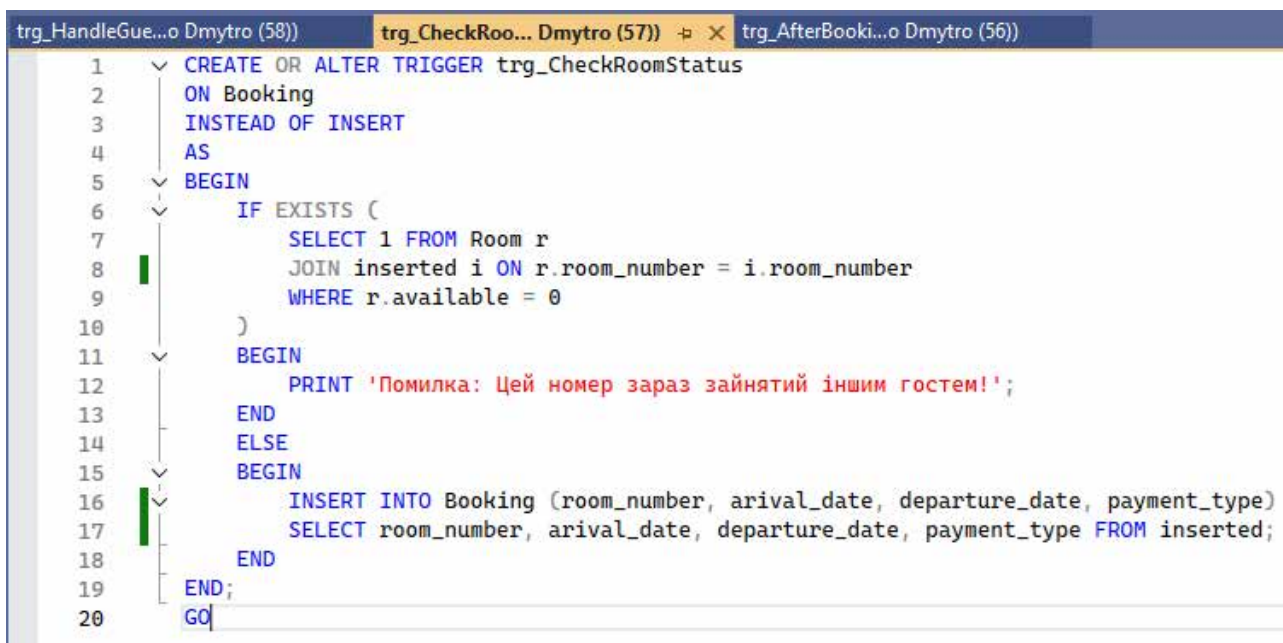
- можливість використання необов'язкового параметра @room_type, завдяки чому пошук можна звузити до конкретної категорії;
- виведення повної інформації про номер, включаючи його ціну та опис зручностей;
- реалізацію механізму зворотного зв'язку: якщо за обраними критеріями в базі даних не знайдено жодного вільного номера, процедура виводить інформаційне повідомлення про відсутність результатів.

2.3.3 Створення тригерів. Використання тригерів у системі обліку апартаментів готелю є елементом для забезпечення цілісності даних та автоматизації рутинних операцій, які раніше вимагали ручного втручання адміністратора.

Тригери - це спеціальні об'єкти бази даних, які автоматично спрацьовують у відповідь на певні дії (вставлення, зміна або видалення записів). У даному проекті вони реалізують поведінку інформаційної бази гарантуючи, що стан номерного фонду завжди відповідає реальній ситуації з бронюваннями. Повний код всіх розроблених тригерів розміщений в додатку Е.

1) Тригер контролю зайнятості номера (**trg_CheckRoomStatus**):

Найбільш важливий тригер типу INSTEAD OF INSERT, який діє як захист від подвійного бронювання і представлений на рис. 9. Перед тим, як дозволити запис нового бронювання в таблицю Booking, алгоритм перевіряє поточне значення поля available у таблиці Room для відповідного номера. Якщо номер уже має статус «зайнято» (available = 0), тригер блокує операцію та виводить інформаційне повідомлення про неможливість створення броні. Тільки у випадку, якщо перевірка пройшла успішно, дані фізично заносяться до бази.



```

1  CREATE OR ALTER TRIGGER trg_CheckRoomStatus
2  ON Booking
3  INSTEAD OF INSERT
4  AS
5  BEGIN
6      IF EXISTS (
7          SELECT 1 FROM Room r
8          JOIN inserted i ON r.room_number = i.room_number
9          WHERE r.available = 0
10     )
11     BEGIN
12         PRINT 'Помилка: Цей номер зараз зайнятий іншим гостем!';
13     END
14     ELSE
15     BEGIN
16         INSERT INTO Booking (room_number, arrival_date, departure_date, payment_type)
17         SELECT room_number, arrival_date, departure_date, payment_type FROM inserted;
18     END
19 END;
20 GO

```

Рис. 9 Тригер контролю зайнятості номеру

2) Тригер відновлення доступності номера (**trg_AfterBookingDelete**): механізм який спрацьовує при скасуванні бронювання або виїзді гостя. Тригер типу AFTER DELETE відстежує видалення записів із таблиці Booking. Як тільки бронювання зникає з активного переліку, тригер автоматично повертає номеру статус «вільний» (available = 1). Це дозволяє номерному фонду готелю уникнути ризику того, що адміністратор забуде оновити дані вручну. Кімнати стають доступними для нових клієнтів одразу після звільнення без часових затримок.

2.3.4 Створення користувачів. Процес налаштування безпеки в БД включає два основні етапи: створення логінів на рівні сервера для автентифікації та створення користувачів на рівні бази даних для визначення конкретних прав доступу. Для системи було виділено дві основні ролі: власник готелю та менеджер. Повний код створених ролей для користувачів розміщений в додатку Ж.

Роль власника готелю (hotel_owner): Обліковий запис власника призначений для здійснення контролю та аналізу діяльності підприємства. Згідно з налаштованими правами доступу, цей користувач має право на виконання команди SELECT для всіх основних таблиць системи, включаючи реєстр номерів, дані гостей, архів бронювань та фінансові квитанції. Це дозволяє власнику переглядати будь-яку інформацію та формувати аналітичні звіти, але

водночас обмежує можливість випадкової зміни або видалення записів, що забезпечує цілісність даних готелю.

Роль менеджера готелю (`hotel_manager`): Цей обліковий запис орієнтований на щоденну роботу з клієнтами та управління житловим фондом. Права доступу для менеджера є більш розширеними в частині редагування даних, але суворо обмеженими в частині системних налаштувань:

- Управління клієнтами та бронюваннями: менеджер має повні права на читання, додавання, оновлення та видалення записів у таблицях `Guest`, `Booking` та `BookingService`, що необхідно для реєстрації нових гостей та обробки замовлень.
- Контроль номерного фонду: для таблиці `Room` менеджеру надано право на перегляд інформації, проте можливість редагування обмежена.
- Фінансова звітність: менеджер може створювати нові квитанції та переглядати існуючі рахунки, а також має виключне право на виконання збереженої процедури `pr_GenerateFinalReceipt` для автоматичного розрахунку вартості послуг.

3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення представлена у вигляді діаграми пакетів, яка зображена на рис.10. Ця діаграма ілюструє розподіл програмного коду на логічні модулі та архітектурні залежності між ними.

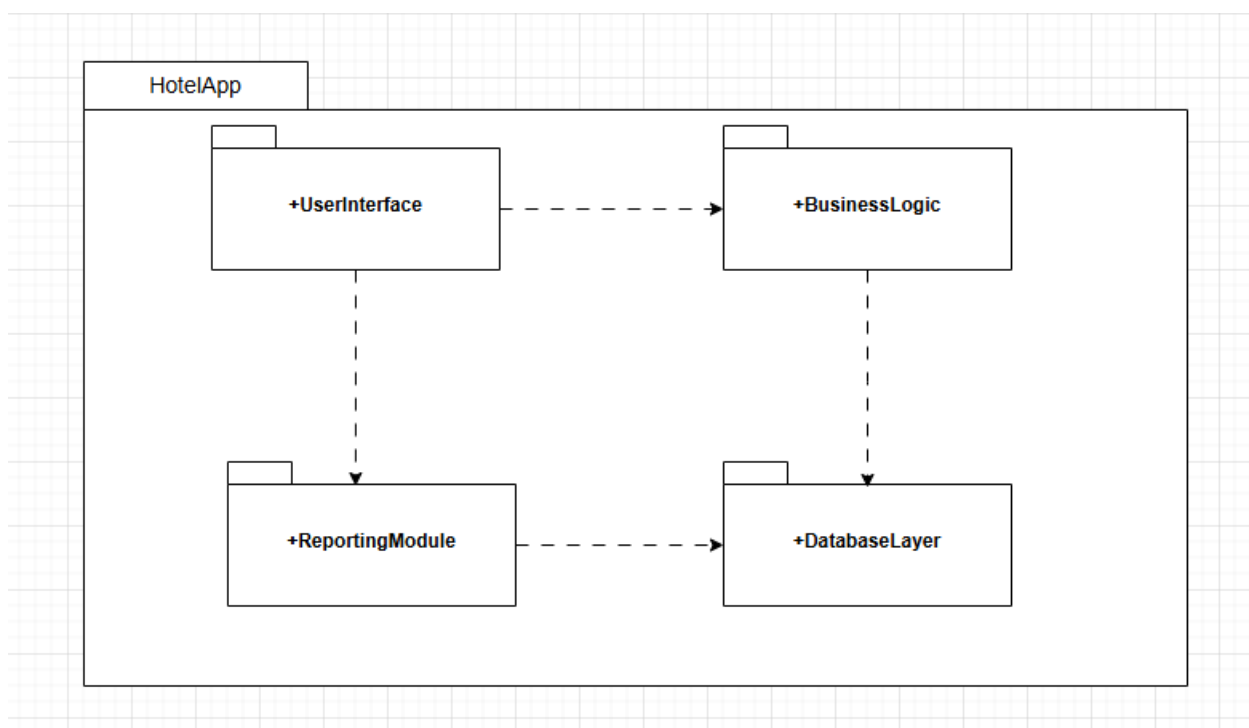


Рис. 10 Діаграма пакетів

Для простоти підтримки та масштабованості програмного продукту було обрано багаторівневу архітектуру розробки. Згідно з побудованою моделлю, архітектура складається з чотирьох основних взаємозв'язних пакетів:

1) **UserInterface** (Рівень представлення): відповідає за безпосередню взаємодію з кінцевим користувачем системи. Він містить візуальні форми, вікна авторизації, карти номерів та елементи введення даних. Пакет є точкою входу в систему і не має прямого доступу до бази даних.

2) **BusinessLogic** (Рівень бізнес-логіки): є логічним ядром додатка. Пакет містить класи, які описують сутності готелю, та методи обробки основних бізнес-правил. Тут відбувається перевірка коректності введення даних користувачем, ініціація процесів реєстрації заїздів та координація інформації між інтерфейсом та БД.

3) **ReportingModule** (Модуль звітності): виділений в окремий структурний елемент для забезпечення аналітичних можливостей системи. Він відповідає за логіку формування квитанцій, рахунків та генерацію звітів про доходи за обраний період часу. Модуль звертається до рівня даних для збору та узагальнення фінансової інформації.

4) **DatabaseLayer** (Рівень доступу до даних): нижній рівень архітектури, який забезпечує комунікацію додатка із СУБД Microsoft SQL Server. Він містить логіку підключення до бази даних, виконання SQL-запитів та виклики збережених процедур.

Аналіз архітектурних зв'язків та залежностей. Організаційна структура реалізує принцип розділення обов'язків. Напрямки стрілок на діаграмі пакетів відображають існуючі залежності між модулями:

- Пакет **UserInterface** залежить від модулів **BusinessLogic** та **ReportingModule**. Будь-яка дія користувача в інтерфейсі, наприклад, натискання кнопки «Створити бронь» чи «Сформувати звіт», передає керування на рівень обробки бізнес-правил або звітності.
- Пакети **BusinessLogic** та **ReportingModule** для виконання відповідних завдань звертаються до пакета **DatabaseLayer**, від якого вони прямо залежать. Рівень даних повертає відповідні результати виконання запитів у вигляді об'єктів.
- Пакет **DatabaseLayer** є повністю незалежним від інших частин програми. Це дозволяє за потребою змінити внутрішню логіку роботи з базою даних не змінюючи інтерфейс додатку.

3.2 Вибір інструментарію для створення ППЗ

Для вдалої реалізації розробленої архітектури та забезпечення стабільної взаємодії між інтерфейсом користувача, бізнес-логікою та базою даних необхідно було підібрати сучасний та високоефективний стек технологій. Правильний вибір інструментальних засобів безпосередньо впливає на швидкість розробки, надійність функціонування готового додатка, а також на ергономічність графічного середовища, з яким щодня працюватиме персонал готелю.

Основним IDE для створення інформаційної системи було обрано Microsoft Visual Studio. Цей вибір є цілком природним для проектів, орієнтованих на екосистему .NET, оскільки середовище надає повний спектр інструментів для написання коду, його налагодження, рефакторингу та тестування. Потужний вбудований дизайнер інтерфейсів, інтелектуальна система підказок IntelliSense та глибока інтеграція з базами даних Microsoft SQL Server дозволили значно оптимізувати процес створення екранних форм і написання серверних запитів.

Базовою мовою програмування для реалізації логічного ядра системи та модулів обробки даних було обрано C#. Обґрунтування вибору саме цієї мови програмування базується на її архітектурних перевагах та технічних можливостях, які позитивно вплинули на розробку системи готельного обліку:

- Об'єктно-орієнтована парадигма (ООП): дозволяє точно відобразити створені сутності готелю у вигляді програмних класів, що значно спрощує розробку складної бізнес-логіки та забезпечить повторне використання коду.
- Строга типізація і безпека: C# мінімізує ризик виникнення помилок ще на етапі компіляції додатка. Автоматичне керування пам'яттю захищає систему від витоків пам'яті та підвищує загальну стабільність роботи програмного продукту.
- Підтримка `async/await`: є ключовим фактором для створення інтерфейсу користувача. Завдяки асинхронності, виконання тривалих запитів до бази даних відбувається у фоновому режимі, що зменшує шанс на зависання головного вікна програми.

- Інтеграція з екосистемою Microsoft: будучи рідною мовою для платформи .NET, C# забезпечить швидкість та надійність комунікації із СУБД MS SQL Server за допомогою основних технологій доступу до даних.

Для розробки графічного інтерфейсу користувача було обрано технологію WPF (Windows Presentation Foundation), яка повністю замінила минулу технологію Windows Forms. Головною особливістю WPF є використання декларативної мови XAML для проектування візуальної частини інтерфейсу. Це дозволить відокремити дизайн вікон від їхньої програмної логіки, полегшуючи подальшу підтримку коду. Крім того, WPF підтримує потужний механізм прив'язки даних, завдяки якому зміни в моделях бізнес-логіки автоматично відображаються на екрані адміністратора і навпаки. Апаратне прискорення графіки DirectX забезпечує плавне відображення елементів інтерфейсу навіть на менш сучасних пристроях.

Зв'язок між програмним додатком та локальною базою даних MS SQL Server реалізовано за допомогою технології ADO.NET. Цей підхід є ефективним для настільних додатків, тому що він гарантує пряме підключення до сервера даних, мінімізуючи витрати на передачу інформації. Використання даної технології дозволило побудувати безпечну взаємодію з базою через параметризовані запити, захищаючи систему від уразливостей типу SQL-ін'єкцій. Ще одним плюсом використання стало те що цей інструментарій забезпечує нативну підтримку виклику збережених процедур, які виконують основну математичну та фільтраційну роботу безпосередньо на стороні СУБД.

Сформований інструментарій розробки, який об'єднує екосистему .NET та середовище Visual Studio, повністю задовольняє критеріям ефективності, які висуваються до сучасного програмного забезпечення готельного менеджменту. Обраний стек дозволяє реалізувати настільний додаток із високим рівнем автономності, забезпечити мінімальний час відгуку інтерфейсу при роботі зі збереженими процедурами бази даних та створити безпечне середовище для щоденної роботи персоналу готелю.

3.3 Реалізація інтерфейсної частини

Реалізація інтерфейсної частини інформаційної системи є одним із найбільш трудомістких етапів розробки, оскільки саме від ергономіки, інтуїтивності та швидкодії графічного середовища залежить продуктивність роботи. Графічний інтерфейс розроблено на базі технології WPF, яка використовує декларативну мову розмітки XAML для проектування візуальних компонентів.

3.3.1 Архітектура візуального макету та розмітки. При проектуванні екранних форм було повністю відкинуто концепцію абсолютного позиціонування елементів за фіксованими координатами, оскільки це призводить до спотворення інтерфейсу при зміні роздільної здатності монітора або масштабуванні вікон оператором. Замість цього архітектура макету побудована на базі адаптивних контейнерів розмітки WPF:

- Grid - використана як базовий каркас для всіх вікон. Вона дозволяє розділити робочий простір на рядки та стовпці з гнучким керуванням їхніми розмірами, забезпечуючи точне вирівнювання великих блоків керування.
- StackPanel - застосована для послідовного розміщення однотипних компонентів: блоків кнопок навігації або текстових полів введення із відповідними підписами по вертикалі або горизонталі.
- DockPanel - впроваджена у головному вікні для фіксації верхнього навігаційного меню та нижнього інформаційного рядка стану, залишаючи всю центральну область для динамічного контенту.

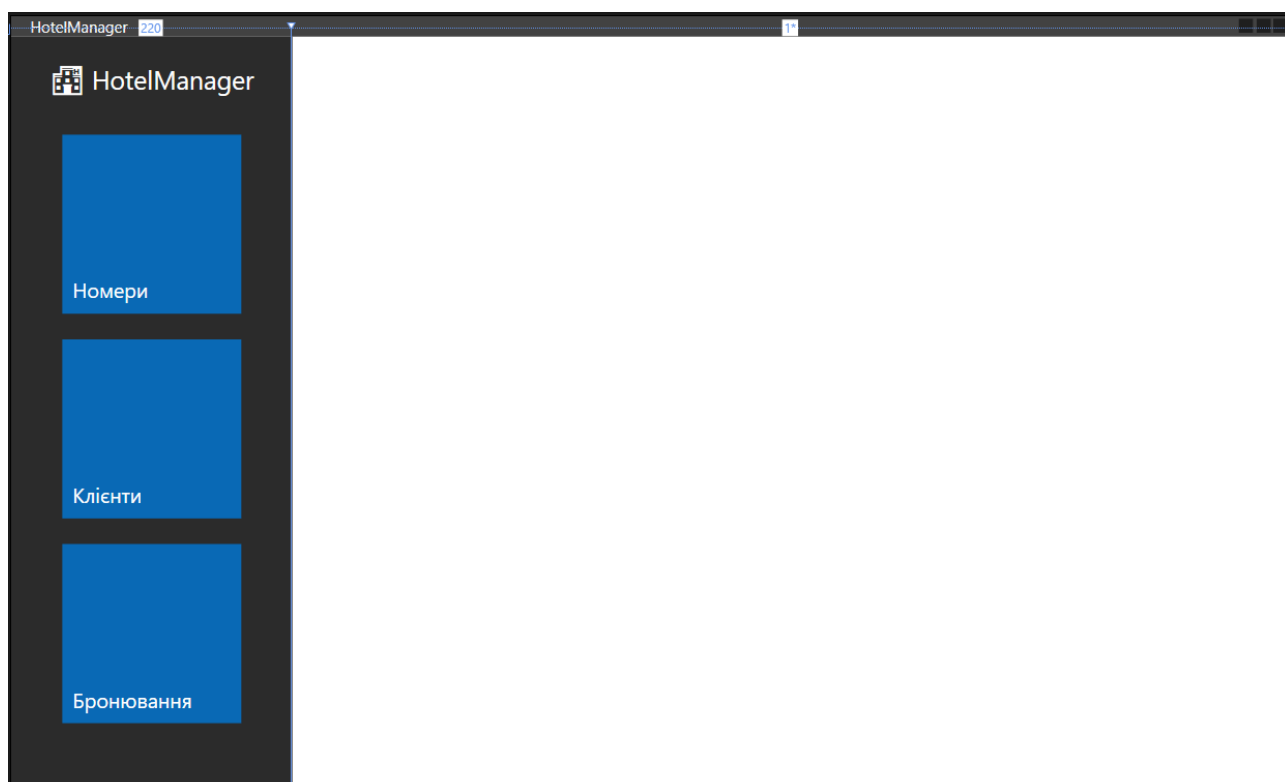


Рис. 11 Проектування базового макету Grid головного вікна

3.3.2 Опис функціональних компонентів інтерфейсу. Візуальний комплекс системи HotelApp складається з чотирьох основних екранних форм, кожна з яких оптимізована під конкретні сценарії діяльності працівників готелю:

1) Вікно автентифікації користувачів: стартова форма додатку, яка виконує функцію первинного фільтра безпеки. Візуально вона спроектована у мінімалістичному стилі, щоб зосередити увагу користувача на процесі авторизації. Форма містить текстове поле введення логіна TextBox та спеціалізоване поле для прихованого введення пароля PasswordBox. Кнопка «Увійти» оснащена тригером стану, який блокує її повторне натискання під час виконання запиту до бази даних для запобігання дублювання підключень.

2) Головне вікно менеджера системи: являє собою центральний дашборд для роботи адміністратора готелю. Робочий простір розділено на дві функціональні зони згідно з макетом сітки Grid. Ліва панель містить кнопки перемикавання розділів, а права зона адаптивно відображає обрані елементи.

Верхня панель містить елементи фільтрації: випадаючий список ComboBox для вибору категорії номера та кнопки швидкого доступу до функцій виклику процедури пошуку. Центральну частину займає табличний компонент DataGrid, який відображає списки кімнат на основі таблиці Room. Особливістю інтерфейсного рішення є динамічне колірне кодування рядків таблиці за допомогою механізму DataTrigger. Система аналізує логічне значення поля available: якщо номер вільний, рядок отримує зелене підсвічування, якщо зайнятий або заброньований - червоне. Це дозволить адміністратору швидко візуально оцінити завантаженість готелю.

```
<DataGridTextColumn.CellStyle>
  <Style TargetType="DataGridCell">
    <Setter Property="FontWeight" Value="SemiBold"/>
    <Style.Triggers>
      <DataTrigger Binding="{Binding Status}" Value="Вільний">
        <Setter Property="Foreground" Value="Green"/>
      </DataTrigger>
      <DataTrigger Binding="{Binding Status}" Value="Зайнятий">
        <Setter Property="Foreground" Value="Red"/>
      </DataTrigger>
    </Style.Triggers>
  </Style>
</DataGridTextColumn.CellStyle>
```

Рис. 12 Реалізація механізму DataTrigger

3) Головне вікно власника готелю: окрема екранна форма, яка завантажується виключно для користувачів із роллю hotel_owner. Оскільки власник володіє правами доступу тільки для читання, його інтерфейс повністю позбавлений будь-яких елементів додавання або редагування даних. Навігаційна панель власника орієнтована на запуск аналітичних сервісів: формування звітів про доходи, моніторинг фінансової успішності категорій номерного фонду та перегляд архіву рахунків із таблиці Receipt.

4) Діалогове вікно реєстрації та обробки бронювань: екранна форма, яка активується при створенні нового запису. Для сприйняття великої кількості інформації, вікно розділено за допомогою візуальних контейнерів GroupBox на два блоки. Блок «Персональні дані гостя» містить поля для введення ПІБ, телефону, електронної пошти та серії/номера документа. Блок «Параметри проживання» містить календар DatePicker, який обмежує вибір дат та випадаючий список для вибору типу оплати.

3.4 Проектування і реалізація сервісів

Сервіси - це ізольовані функціональні частини програми, які містять у собі ключові алгоритми процесів обліку номерного фонду, реєстрації клієнтів та фінансового моніторингу готелю. Вони реалізують прикладні можливості системи, взаємодіючи з графічною WPF-оболонкою для безпечного зчитування вхідних параметрів та передачі оброблених результатів на екран користувача. Сервісна логіка представлена у вигляді спеціалізованих методів, подій та обробників, які інтегровані у кодові файли відповідних вікон додатка. Кожен із цих модулів виконує операції з БД за допомогою штатних підключень SqlConnection, параметризованих запитів та збережених процедур СУБД. Повний код всіх перелічених сервісів буде представлений у додатку И.

Перелік сервісів, реалізованих у програмі:

1) Авторизація та рольова маршрутизація (форма LoginWindow.xaml.cs). Головна ціль сервісу - забезпечити захищений вхід до системи обліку, верифікувати права користувача на рівні СУБД та ініціалізувати відповідний операційний інтерфейс.

Основні функції сервісу:

- Автентифікація на рівні сервера (подія LoginButton_Click): здійснює зчитування логіна та пароля працівника. Особливістю є повна відмова від зберігання, порівняння або передачі відкритих паролів у програмному коді. Додаток передає введені дані у метод DbConfig.GenerateConnectionString(user, pass), формуючи динамічний рядок підключення, та виконує спробу відкриття сесії через

conn.Open(). Якщо СУБД Microsoft SQL Server відхиляє запит, блок catch перехоплює виключення SqlException та виводить вікно попередження про відмову в доступі.

- Перемикання інтерфейсних форм: у разі успішного підключення рядок connStr фіксується у глобальній конфігурації додатка. Після цього алгоритм аналізує логічне ім'я користувача, якщо входить власник готелю, ініціалізується та відображається вікно OwnerWindow, у всіх інших випадках - відкривається головна форма адміністратора MainWindow. Код обробника події входу зображено на рис. 13.

```
private void LoginButton_Click(object sender, RoutedEventArgs e)
{
    string user = LoginField.Text.Trim();
    string pass = PasswordField.Password;

    if (string.IsNullOrEmpty(user) || string.IsNullOrEmpty(pass))
    {
        MessageBox.Show("Будь ласка, введіть логін та пароль!");
        return;
    }

    // підключення
    string connStr = DbConfig.GenerateConnectionString(user, pass);

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        try
        {
            conn.Open();
            DbConfig.ConnectionString = connStr;

            if (user.ToLower() == "hotel_owner")
            {
                OwnerWindow ownerWin = new OwnerWindow();
                ownerWin.Show();
            }
            else
            {
                MainWindow managerWin = new MainWindow();
                managerWin.Show();
            }

            this.Close();
        }
        catch (SqlException)
        {
            MessageBox.Show("Помилка авторизації: Невірний логін або пароль.",
                "Відмовлено в доступі", MessageBoxButton.OK,
                MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Критична помилка підключення: " + ex.Message);
        }
    }
}
```

Рис. 13 Код методу обробки входу

2) Додавання інформації про клієнта (сторінка ClientAddPage.xaml.cs).

Основною функцією сервісу є реєстрація нового клієнта (подія SaveClient_Click): координує процес збору, перевірки та відправки персональних даних гостя до таблиці Guest.

- Сервіс зчитує інформацію з текстових полів, очищаючи її від випадкових пробілів на кінцях за допомогою методу Trim().
- Виконується обов'язкова первинна валідація обов'язкових атрибутів через логічну умову string.IsNullOrEmpty.
- У разі виявлення порожніх полів сервіс перериває транзакцію та виводить попереджувальне вікно.
- Змінні прізвища та імені об'єднуються в один рядок для дотримання структури поля guest_name в СУБД.
- Автоматична генерація унікального ID.

3) Створення бронювання (сторінка BookingCreatePage.xaml.cs). Головна мета цього модуля - дозволити адміністратору готелю швидко та без помилок зареєструвати заселення гостя, обрати для нього потрібні додаткові послуги, автоматично сформувати підсумковий фінансовий рахунок та миттєво оновити статус номерів в одному робочому вікні.

Основні функції програми в цьому модулі:

1. Автоматичне завантаження даних (метод LoadFormData): цей метод запускається самостійно при відкритті сторінки. Цей метод дістає інформацію з бази даних про актуальні списки клієнтів, послуг та кімнат, щоб адміністратор міг обрати їх. Метод робить три запити до бази через одне підключення:

- З таблиці Guest зчитує ПІБ та ID клієнтів для списку GuestComboBox.
- З таблиці Room вибирає лише ті номери, які зараз є вільними (WHERE available = 1), щоб виключити помилку повторного заселення в одну й ту саму кімнату.
- З таблиці Service завантажує назви та ціни додаткових послуг у список ServicesListBox.

2. Збереження бронювання та розрахунок чека (подія `ConfirmBooking_Click`): це головний обробник, який виконує всю основну роботу із заселення клієнта. Його алгоритм працює за такими кроками:

1. Спочатку програма перевіряє, чи вибрав адміністратор гостя, номер і дати проживання. Якщо якесь поле порожнє, виводиться попередження і робота зупиняється.
2. Програма автоматично створює новий код бронювання на основі поточного часу комп'ютера, що гарантує його абсолютну унікальність.
3. Для безпеки даних усі зміни вносяться через транзакцію. Програма виконує 4 послідовні дії як одну єдину команду:
 - Вносить основний запис про терміни проживання та тип оплати в таблицю `Booking`.
 - Перевіряє через цикл, які додаткові послуги вибрав користувач у списку, і записує їх у таблицю `BookingService`.
 - Змінює статус кімнати на «зайнята» за допомогою команди `UPDATE Room SET available = 0`.
 - Викликає на сервері готову процедуру `pr_GenerateFinalReceipt`, яка сама рахує кількість днів проживання, додає вартість послуг і виписує підсумковий фінансовий чек у таблицю рахунків.
4. Якщо всі кроки пройшли успішно, транзакція фіксується командою `Commit`, показується вікно успіху, і сторінка закривається. Якщо ж на якомусь етапі стався збій, спрацьовує захисний блок `catch`, який миттєво скасовує всі внесені зміни, повертаючи базу даних до початкового стану.

Покроковий алгоритм роботи цього модуля у вигляді блок-схеми показано на рис. 14.



Рис. 14 Блок-схема алгоритму створення бронювання

3. Керування навігацією (методи AddNewGuest_Click та Cancel_Click): ці функції відповідають за зручність роботи. Якщо потрібного клієнта немає у

списку, кнопка `AddNewGuest_Click` перенаправляє користувача на сторінку створення картки `ClientAddPage`. Кнопка скасування (`Cancel_Click`) просто повертає користувача на попередній екран додатка.

3.5 Проектування і реалізація взаємодії з базою даних

Взаємодія системи із базою даних реалізована на нижньому архітектурному рівні додатка за допомогою низькорівневих компонентів технології ADO.NET. Усі операції зчитування, запису та модифікації інформації виконуються через пряме керування підключеннями та командами, що дозволить досягти максимальної швидкодії системи та контролю над транзакціями.

Для створення та відкриття сесії зв'язку із сервером у коді програми використовується клас `SqlConnection`. Рядок підключення `Connection String` формується динамічно під час авторизації користувача та зберігається у глобальній статичній змінній `DbConfig.ConnectionString` для використання всіма модулями системи.

Всі запити до таблиць представлені у вигляді текстових констант, які передаються в об'єкт команди `SqlCommand`. Замість прямого виконання коду, у системі реалізовано чіткий перелік логічних операцій:

1. Операція додавання нового гостя: виконує команду вставки нового рядка до таблиці клієнтів. Вона передає у відповідні колонки згенерований додатком текстовий ідентифікатор, сформоване ПІБ, а також введені телефони, email та реквізити паспорта або ID-картки.
2. Операція створення бронювання: здійснює вставку даних у головну таблицю бронювань. Вона фіксує унікальний номер броні, зовнішній ключ клієнта, обраний номер кімнати, точні дати заїзду й виїзду, а також обраний тип фінансового розрахунку.
3. Операція реєстрації додаткових послуг: виконує покрокове додавання записів у проміжну таблицю зв'язку `BookingService`. Вона пов'язує унікальний код поточного бронювання з кодами та кількістю обраних сервісів, що дозволяє деталізувати витрати клієнта.

4. Операція зміни стану номеру: слугує для зміни статусу для таблиці кімнат.

Для вказаного номера апартаментів оновлюється логічний прапорець доступності, встановлюючи його в значення «зайнято», що автоматично виключає номер із переліку доступних для наступних заселень.

Для захисту інформаційного забезпечення готелю від атак типу SQL-ін'єкцій у системі повністю заборонено використання склеювання текстових рядків у тілі запитів. Усі команди виконуються виключно через колекцію параметрів об'єкта `SqlCommand`. Прив'язування значень із графічних полів сторінок додатка до SQL-параметрів здійснюється через метод `Parameters.AddWithValue`. При виконанні інструкцій бібліотека `ADO.NET` передає серверу окремо структуру запиту й окремо масив даних. СУБД сприймає параметри виключно як текстові чи числові літерали, що повністю ліквідує можливість підміни логіки виконання SQL-коду користувачем.

Для забезпечення атомарності даних при комплексному бронюванні, на рівні коду ініціалізується об'єкт локальної транзакції `SqlTransaction trans = conn.BeginTransaction()`. Кожен об'єкт `SqlCommand`, який бере участь у процесі, отримує посилання на поточну транзакцію у своєму конструкторі. Це змушує СУБД виконувати інструкції в ізольованому середовищі. Лише після безпомилкового виконання процедури код викликає метод `trans.Commit()`, який остаточно закріплює зміни у таблицях сервера. Якщо будь-яка команда викликає виключення, керування миттєво переходить у блок `catch`, де інструкція `trans.Rollback()` повністю анулює всі проміжні зміни, повертаючи базу даних до початкового стану.

Окремим важливим процесом взаємодії з СУБД є безпечний та швидкий відбір фінансово-статистичної інформації для кабінету власника готелю. Для цього на рівні коду використовуються об'єкти `SqlDataReader`, які забезпечують потокове читання результатів запитів у режимі тільки для читання.

Програмний алгоритм відбору аналітичних даних працює за чіткими кроками:

1. При виборі звітнього періоду додаток спочатку ініціює метод очищення оперативної пам'яті інтерфейсу.
2. Далі ініціюється параметризований запит або запуск аналітичної збереженої процедури сервера `pr_GetRevenueReport`.
3. Об'єкт `SqlDataReader` зчитує отримані рядки СУБД, а логічне ядро автоматично робить мапінг у колекції об'єктів мови C#.
4. Підготовлені списки через механізм `ItemsSource` передаються компонентам відображення: графічній бібліотеці `LiveCharts` для малювання діаграм прибутку за місяцями та таблицям `DataGrid` для аналізу рентабельності категорій номерного фонду.

Графічне представлення покрокового алгоритму відбору інформації та взаємодії компонентів із базою даних у вигляді блок-схеми наведено на рис. 15.

Таким чином, спроектований рівень взаємодії з базою даних визначає високу життєздатність та експлуатаційну надійність інформаційної системи. Замість перевантаження локальних комп'ютерів персоналу складними обчисленнями, обрана архітектура розподіляє навантаження: сервер бере на себе аналітичну роботу, тим самим додаток залишається максимально швидким.

Це створює можливість для подальшого масштабування - навіть при значному розширенні номерного фонду готелю чи зростанні кількості одночасних клієнтських запитів програма працюватиме без втрати продуктивності. Такий підхід дозволяє використовувати базу даних як динамічне середовище, яке з'єднує роботу адміністратора на рецепції і контролем з боку власника підприємства.

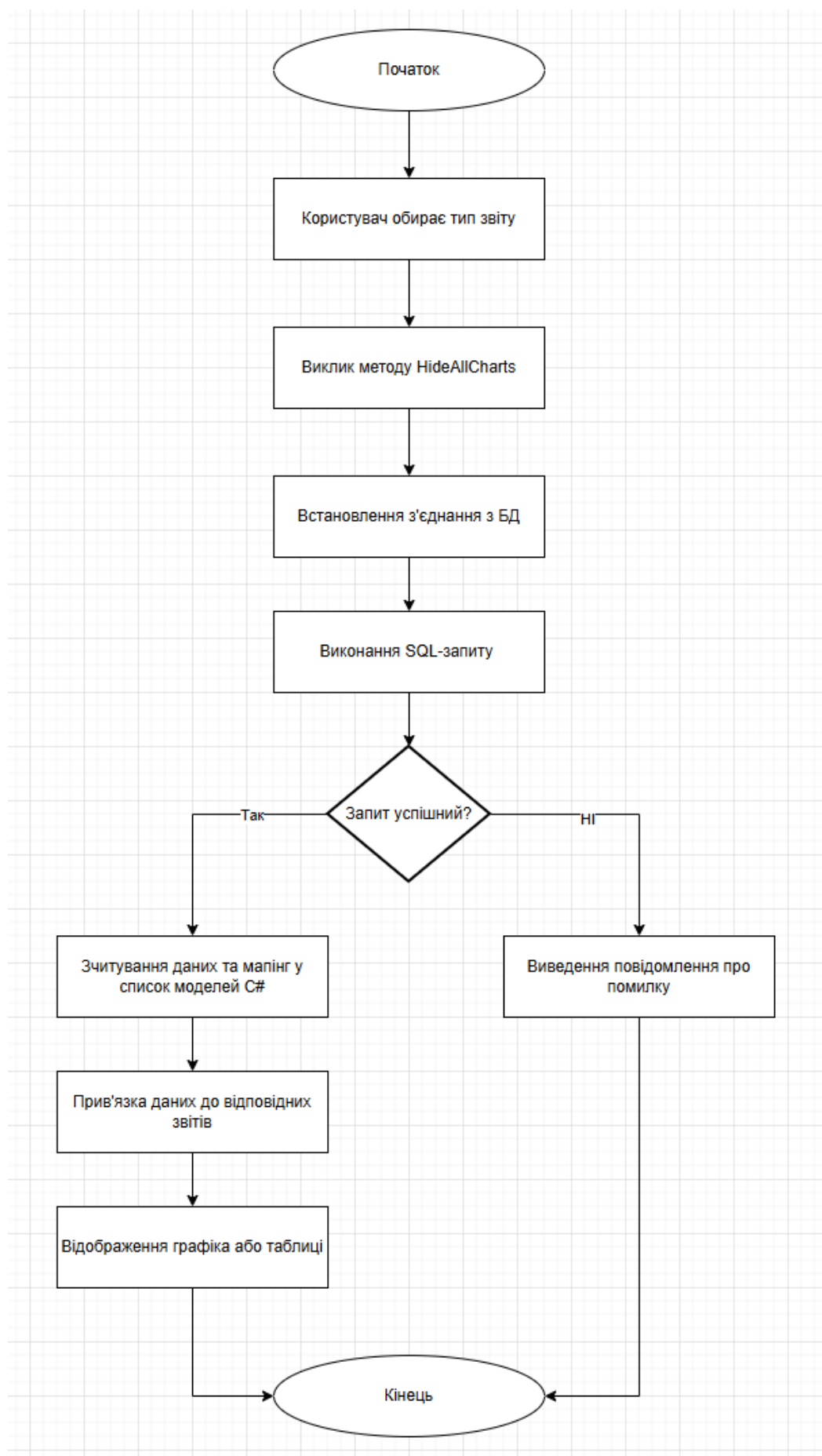


Рис. 15 Блок-схема алгоритму відбору даних для формування звітності

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Діаграма розгортання

Для відображення зв'язку між технічними пристроями та фізичного програмного коду було розроблено діаграму розгортання яка зображена на рис. 16.

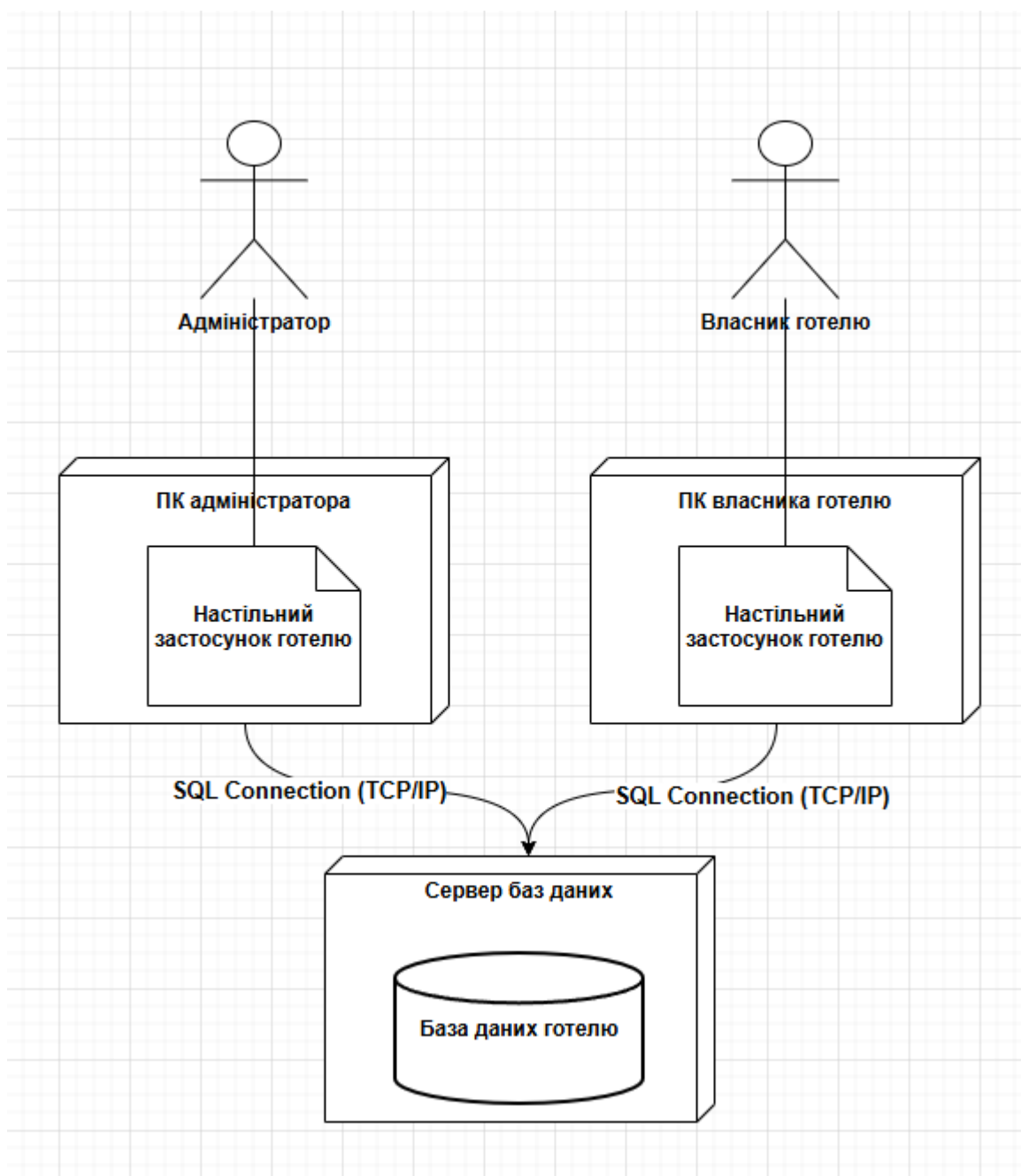


Рис. 16 Діаграма розгортання системи

Ця діаграма дозволяє детально описати структуру системи на фізичному рівні абстракції за допомогою базових понять:

- Вузол - матеріальний обчислювальний ресурс, який володіє власною оперативною пам'яттю та процесором для виконання програм. Саме всередині вузлів створюються та запускаються програмні елементи.
- Компонент та Артефакт - фізичний файл наприклад, виконувана програма чи файл бази даних, який є матеріальним і фізичним втіленням логічного модуля.

Розроблювана інформаційна система обліку апартаментів у готелі за своєю фізичною структурою побудована на базі класичної архітектури клієнт-сервер. Відповідно до побудованої моделі автоматизації, комплекс готелю складається з таких елементів:

1. Апаратні вузли системи

- ПК адміністратора – клієнтський комп'ютер, розташований безпосередньо на рецепції готелю для ведення операційної діяльності. З цим апаратним вузлом взаємодіє актор «Адміністратор» для щоденного швидкого обслуговування відвідувачів.
- ПК власника готелю - клієнтський комп'ютер, призначений для проведення фінансової аналітики та перегляду статистики. З цим вузлом взаємодіє актор «Власник готелю».
- Сервер баз даних - центральний виділений обчислювальний сервер системи, що існує забезпечення роботи серверної частини під управлінням Microsoft SQL Server.

2. Фізичні програмні артефакти

- Настільний застосунок готелю - програмний артефакт, файл який є фізичним елементом розробленого графічного інтерфейсу та шару логіки. Артефакт автономно запускається та працює в оперативній пам'яті обох робочих станцій персоналу.

- База даних готелю - артефакт сховища даних, розгорнутий всередині серверного вузла. Він працює під управлінням СУБД та об'єднує в собі всі таблиці, тригери, і збережені процедури.

3. Зв'язки та протоколи

- Взаємодія між технічними пристроями відображається за допомогою суцільних ліній зв'язку.
- З'єднання клієнтських застосунків із центральним сервером баз даних здійснюється через мережевий протокол TCP/IP за допомогою технології SQL Connection.

4.2 Апаратні та програмні вимоги до системи

Для функціонування інформаційної системи обліку апартаментів у готелі, у умовах експлуатації розроблені технічні вимоги до програмного та апаратного забезпечення. Оскільки додаток побудований за дворівневою архітектурою, специфікації розподілені на дві категорії: вимоги до робочого місця користувача (клієнтська частина) та вимоги до сервера баз даних (серверна частина).

4.2.1 Програмні вимоги.

Вимоги до клієнтського пристрою:

- Операційна система: Microsoft Windows 10 або Windows 11 (64-bit).
- Системна платформа: .NET Desktop Runtime версії 6.0 або 8.0.
- Мережеві компоненти: драйвери локальної мережі з підтримкою протоколів TCP/IP.

Вимоги до сервера баз даних:

- Операційна система: Windows Server 2019 / 2022 Standard чи Datacenter або Windows 10 / 11 Pro.
- Система управління базами даних: Microsoft SQL Server 2019 або Microsoft SQL Server 2022 у редакціях Express, Standard або Developer.
- Додаткові інструменти: інстальована та розгорнута база даних HotelManagement із набором збережених процедур і тригерів.

4.2.2 Апаратні вимоги

Конфігурація робочого місця користувача:

- Процесор (CPU):
 - *Мінімальний*: 2-ядерний процесор з тактовою частотою від 2.0 ГГц.
 - *Рекомендований*: 4-ядерний процесор Intel Core i3 / AMD Ryzen 3 або вище.
- Оперативна пам'ять (RAM):
 - *Мінімальна*: 4 ГБ.
 - *Рекомендована*: 8 ГБ.
- Диск: від 200 МБ вільного місця на накопичувачі HDD або SSD.
- Монітор: роздільна здатність екрана від 1280×720 пікселів (рекомендовано 1920×1080 для коректного відображення).

Конфігурація сервера баз даних:

- Процесор (CPU): 4-ядерний процесор із частотою від 2.5 ГГц Intel Core i5 / AMD Ryzen 5 або серверні аналоги.
- Оперативна пам'ять (RAM):
 - *Мінімальна*: 8 ГБ.
 - *Рекомендована*: 16 ГБ.
- Диск: накопичувач зі швидкістю читання/запису від 500 МБ/с. Вільне місце - мінімум 5 ГБ.

4.3 Склад інсталяційного пакету

Інсталяційний пакет - набір скомпільованих програмних та конфігураційних файлів, зовнішніх залежностей та скриптів бази даних, який здійснює автоматизоване або напівавтоматизоване розгортання системи в вибраному операційному середовищі. На рис.17 зображено склад розробленого інсталяційного пакета інформаційної системи HotelApp.

HotelAppInstaller					Поиск в: HotelAppInstaller	
Имя		Дата изменения	Тип	Размер		
SQL		14.05.2026 19:03	Папка с файлами			
HotelApp_setup.exe		20.05.2026 22:39	Приложение	5 849 КБ		

Рис. 17 Склад інсталяційного пакету системи

Формування цього комплексу поставки відбувалося у два етапи. На першому етапі в середовищі розробки Microsoft Visual Studio було виконано публікацію проекту в конфігурації Release. Це дозволило зібрати в один робочий каталог файли програми, конфігураційні маніфести платформи .NET та всі зовнішні бібліотеки зв'язку, включаючи драйвери для роботи з базою даних інші компоненти.

На другому етапі отримані файли були скомпільовані за допомогою утиліти Inno Setup у єдиний файл встановлення HotelApp_setup.exe. Головний файл HotelApp_setup.exe є інсталятором клієнтського застосунку. При його запуску на комп'ютері адміністратора чи власника готелю запускається автоматичний процес встановлення, який самостійно копіює всі файли програми у системні директорії та створює ярлики для швидкого доступу на робочому столі. А супутні SQL-сценарії для створення структури сховища даних на сервері були виділені в окрему папку SQL. Папка SQL містить SQL-запити для налаштування сервера. Ці запити дозволять адміністратору швидко розгорнути структуру бази даних: таблиці, базову інформацію, збережені процедури та тригери безпосередньо на комп'ютері готелю через стандартну консоль керування СУБД. На рис. 18 зображено склад папки запитів SQL.

CreateDatabase.sql	23.03.2026 22:51	Microsoft SQL Ser...	1 КБ
CreateTables.sql	14.05.2026 18:23	Microsoft SQL Ser...	3 КБ
CreateUsers.sql	14.05.2026 19:03	Microsoft SQL Ser...	2 КБ
InsertInfo.sql	13.04.2026 1:23	Microsoft SQL Ser...	5 КБ
pr_CancelBooking.sql	31.03.2026 0:59	Microsoft SQL Ser...	1 КБ
pr_GenerateFinalReceipt.sql	13.04.2026 2:04	Microsoft SQL Ser...	2 КБ
pr_GetRevenueReport.sql	14.05.2026 18:26	Microsoft SQL Ser...	1 КБ
pr_SearchAvailableRooms.sql	14.05.2026 18:26	Microsoft SQL Ser...	1 КБ
trg_AfterBookingInsert.sql	14.05.2026 18:40	Microsoft SQL Ser...	1 КБ
trg_CheckRoomStatus.sql	14.05.2026 18:40	Microsoft SQL Ser...	1 КБ
trg_HandleGuestCheckOut.sql	12.04.2026 19:55	Microsoft SQL Ser...	1 КБ

Рис. 18 Склад інсталяційної папки створення БД

На рис. 19 зображено склад папки публікації програми інформаційної системи, що утворилася після збирання проекту.

Имя	Дата изменения	Тип	Размер
ko	20.05.2026 22:15	Папка с файлами	
pl	20.05.2026 22:15	Папка с файлами	
pt-BR	20.05.2026 22:15	Папка с файлами	
ru	20.05.2026 22:15	Папка с файлами	
runtimes	20.05.2026 22:15	Папка с файлами	
tr	20.05.2026 22:15	Папка с файлами	
zh-Hans	20.05.2026 22:15	Папка с файлами	
zh-Hant	20.05.2026 22:15	Папка с файлами	
ControlzEx.dll	01.11.2020 0:09	Расширение при...	246 КБ
HotelApp.deps.json	20.05.2026 22:15	Файл "JSON"	28 КБ
HotelApp.dll	20.05.2026 22:15	Расширение при...	68 КБ
HotelApp.exe	20.05.2026 22:15	Приложение	148 КБ
HotelApp.pdb	20.05.2026 22:15	Program Debug D...	26 КБ
HotelApp.runtimeconfig.json	20.05.2026 22:08	Файл "JSON"	1 КБ
LiveCharts.dll	20.06.2017 3:26	Расширение при...	149 КБ
LiveCharts.Wpf.dll	20.06.2017 3:26	Расширение при...	213 КБ
MahApps.Metro.dll	06.07.2023 13:26	Расширение при...	3 489 КБ
Microsoft.Bcl.Cryptography.dll	31.10.2023 16:58	Расширение при...	17 КБ
Microsoft.Data.SqlClient.dll	17.03.2026 18:10	Расширение при...	629 КБ
Microsoft.Data.SqlClient.Extensions.Abst...	17.03.2026 17:36	Расширение при...	24 КБ
Microsoft.Data.SqlClient.Internal.Login...	17.03.2026 17:11	Расширение при...	53 КБ
Microsoft.Extensions.Caching.Abstractions...	31.10.2023 16:58	Расширение при...	31 КБ
Microsoft.Extensions.Caching.Memory.dll	17.09.2024 2:10	Расширение при...	45 КБ
Microsoft.Extensions.DependencyInjecti...	17.09.2024 2:09	Расширение при...	63 КБ
Microsoft.Extensions.Logging.Abstractions...	17.09.2024 2:09	Расширение при...	64 КБ
Microsoft.Extensions.Options.dll	18.01.2024 5:05	Расширение при...	64 КБ
Microsoft.Extensions.Primitives.dll	31.10.2023 16:58	Расширение при...	43 КБ
Microsoft.IdentityModel.Abstractions.dll	12.02.2026 5:32	Расширение при...	20 КБ
Microsoft.IdentityModel.JsonWebTokens....	12.02.2026 5:32	Расширение при...	169 КБ

Рис. 19 Склад програми після інсталяції

Усі створені файли можна поділити на чотири групи:

- **Основні виконувані файли.** Ця частина компонентів відповідає за безпосередній запуск головного вікна програми користувачем, збереження базового коду системи та визначення параметрів робочого середовища платформи .NET. Характерними елементами цієї категорії є файл запуску HotelApp.exe, бібліотека класів HotelApp.dll та файл конфігурації HotelApp.runtimeconfig.json.
- **Компоненти графічного інтерфейсу.** Головним призначенням цих файлів є створення візуального стилю вікон додатка та побудова діаграм та аналітичних графіків. Прикладами цієї групи є бібліотека стилізації MahApps.Metro.dll, графічні модулі LiveCharts.Wpf.dll і ControlzEx.dll.

- **Засоби взаємодії з базою даних.** Призначені для забезпечення обміну інформацією із сервером БД, шифрування каналів зв'язку та перевірки прав доступу користувачів під час авторизації. Основними прикладами є драйвер підключення `Microsoft.Data.SqlClient.dll`, модулі безпеки `Microsoft.Bcl.Cryptography.dll` та системні розширення `Microsoft.Extensions.Logging.dll`.

- **Службові каталоги локалізації та адаптації середовища.** Мовні пакети для правильного відображення стандартних повідомлень системи, також забезпечують стабільну роботу додатка на комп'ютерах із різними технічними характеристиками та типами процесорів. До цієї групи належать папки мовних локалізацій, такі як `de`, `pl`, `tr` та папка системної адаптації `runtimes`.

4.4 Тестування системи

Для перевірки працездатності та правильності виконання всіх функцій інформаційної системи було проведено ручне функціональне тестування. Цей метод дозволяє імітувати реальні дії персоналу готелю адміністратора та власника для перевірки відповідності програми заявленим вимогам.

На рис. 20 зображено стартове вікно входу до системи під час введення облікових даних працівника готелю. Символи у полі «Пароль» автоматично маскуються крапками для збереження конфіденційності. При натисканні кнопки «УВІЙТИ» додаток успішно виконує запит до бази даних для перевірки ролі та відкриває головне робоче вікно системи.

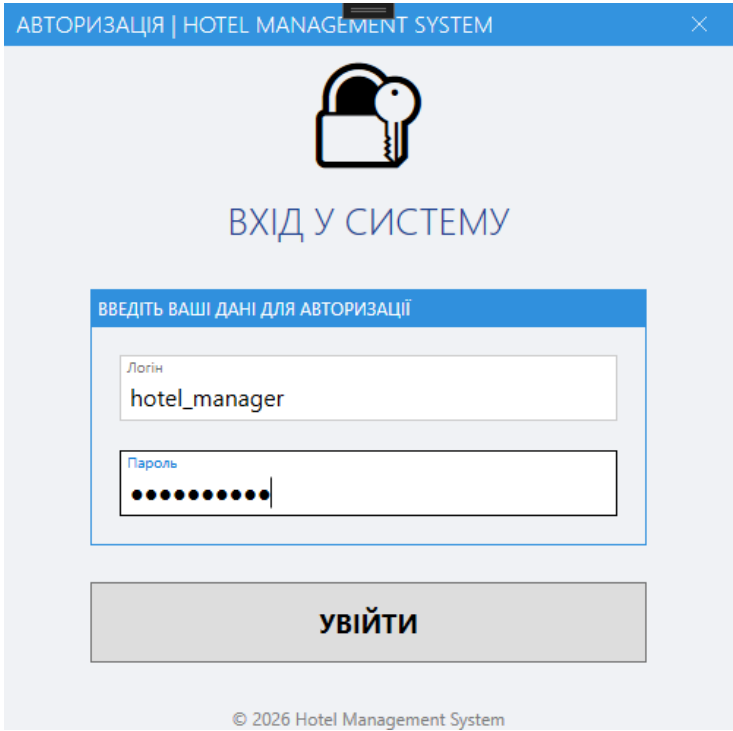


Рис. 20 Авторизація користувача

На рис. 21 зображено головне робоче вікно програми з активованою вкладкою «Номери». Система коректно виводить повний список апартаментів готелю у вигляді таблиці. Поточний стан кімнат має кольорове маркування, де доступні для заселення номери виділяються зеленим кольором а вже заброньовані чи зайняті – червоним.

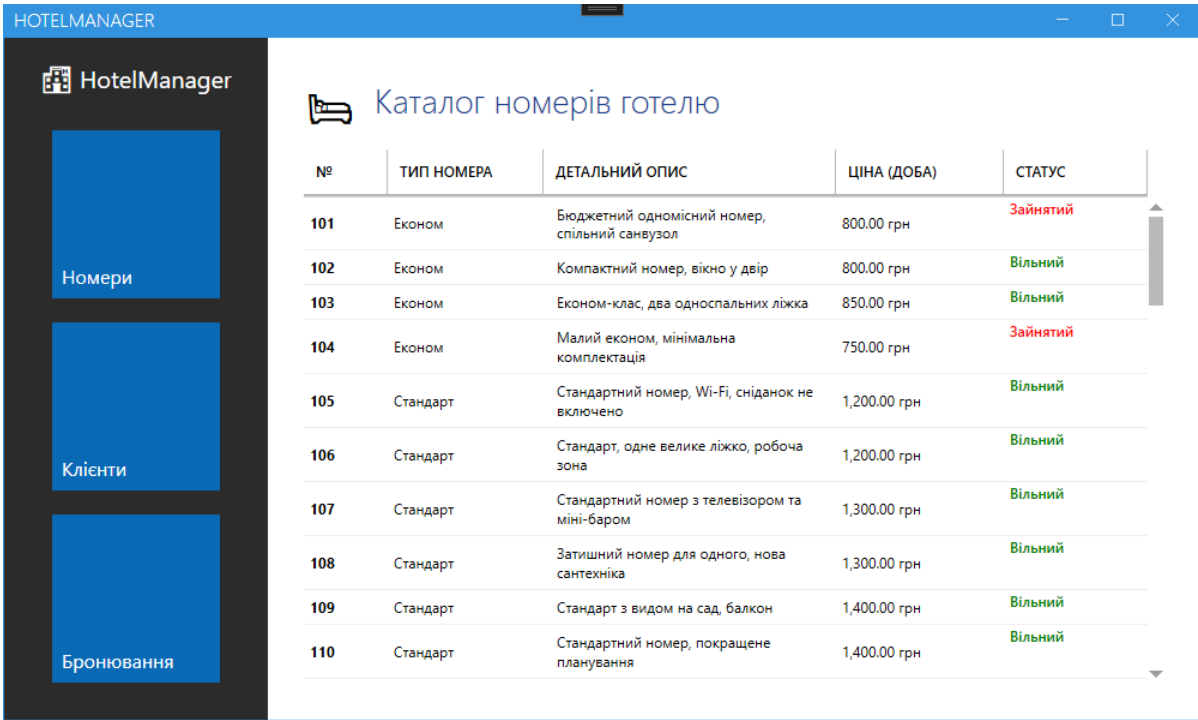


Рис. 21 Каталог номерів готелю

На рис. 22 зображено робочу сторінку «Клієнти», яка призначена для ведення обліку постояльців готелю. Система виводить актуальну інформацію про гостей.

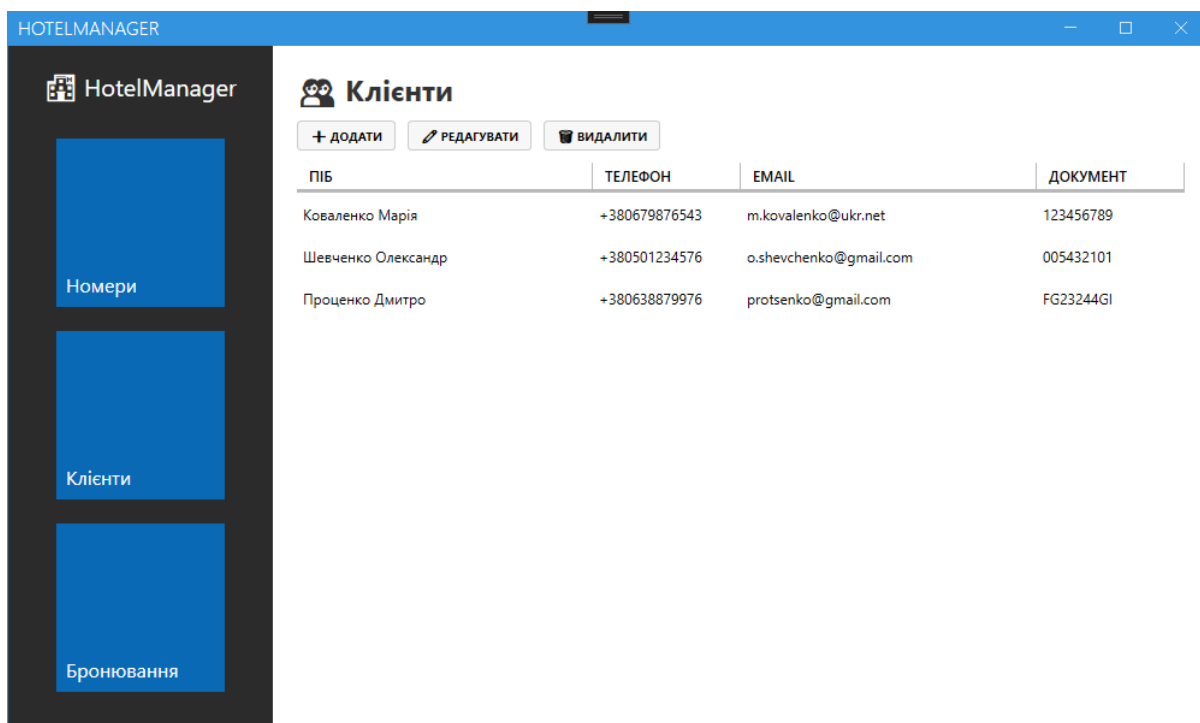


Рис. 22 Вікно управління клієнтською інформацією

На рис. 23 зображено інтерфейс форми, призначеної для реєстрації та внесення даних про нового гостя до бази даних. Кнопка «ЗБЕРЕГТИ» успішно створює новий запис у БД.

Додати нового клієнта

ОСНОВНА ІНФОРМАЦІЯ

Введіть ім'я: Тарас

Введіть прізвище: Шевченко

КОНТАКТНІ ДАНІ

Номер телефону (+380...): +380993462349

Електронна пошта (Email): tarasshe@gmail.com

ІДЕНТИФІКАЦІЯ

Номер документа (Паспорт / ID-картка): KV43404

СКАСУВАТИ ЗБЕРЕГТИ

Рис. 23 Вікно додавання клієнта до бази даних

На рис. 24 зображено інтерфейс форми, призначеної для зміни інформації про вже зареєстрованого в системі гостя. Редагування працює через вибір потрібного гостя у загальній таблиці. Користувач може вносити виправлення у будь-яке з текстових полів, а натискання кнопки «ЗБЕРЕГТИ» успішно оновлює інформацію в базі даних.

The screenshot shows a web application window titled "HOTELMANAGER". On the left is a dark sidebar with the "HotelManager" logo and three blue buttons labeled "Номери", "Клієнти", and "Бронювання". The main content area is titled "Редагувати дані про клієнта" with a user icon. It contains three form sections: "ОСНОВНА ІНФОРМАЦІЯ" with fields for "Введіть ім'я" (filled with "Павло") and "Введіть прізвище" (filled with "Шевченко"); "КОНТАКТНІ ДАНІ" with fields for "Номер телефону (+380...)" (filled with "+380993462349") and "Електронна пошта (Email)" (filled with "tarasshe@gmail.com"); and "ІДЕНТИФІКАЦІЯ" with a field for "Номер документа (Паспорт / ID-картка)" (filled with "KV43404"). At the bottom are two buttons: "СКАСУВАТИ" and "ЗБЕРЕГТИ".

Рис. 24 Вікно редагування даних клієнта

На рис. 25 зображено роботу програми під час спроби видалення інформації про гостя. Процес видалення реалізований через попередній вибір потрібного рядка у загальній таблиці. Для запобігання випадковому стиранню даних система автоматично викликає вікно підтвердження із запитом.

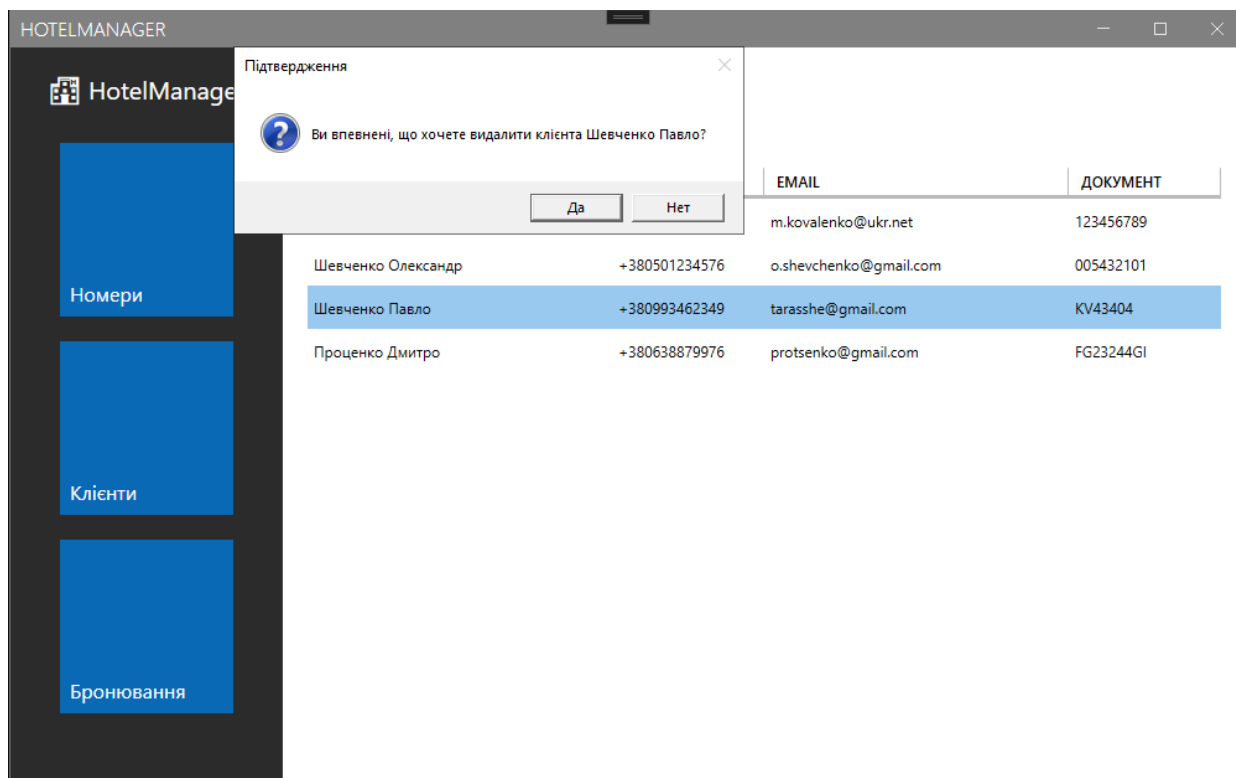


Рис. 25 Процес видалення клієнта

На рис. 26 зображено інтерфейс робочої сторінки «Бронювання». У таблиці відображаються діючі записи, включаючи ім'я клієнта, закріплений номер кімнати, календарні дати заїзду та виїзду та обраний тип оплати.

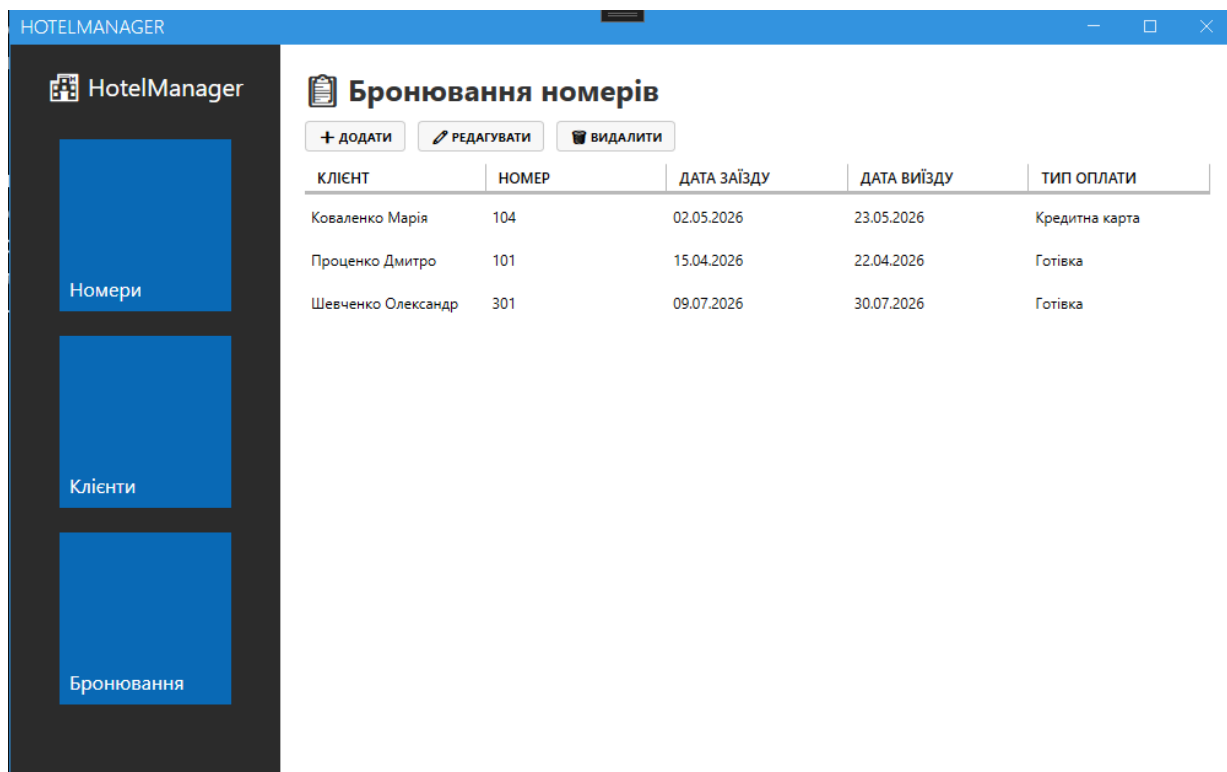


Рис. 26 Вікно журналу активних бронювань

На рис. 27 зображено роботу функції реєстрації нового бронювання в готелі. Кнопка «ПІДТВЕРДИТИ» зберігає сформоване бронювання в БД та запускає тригер для оновлення статусу номера.

HOTELMANAGER

HotelManager

Номери

Клієнти

Бронювання

Оформлення нового бронювання

ВИБІР КЛІЄНТА

Оберіть клієнта...
Шевченко Павло

ДЕТАЛІ ПРОЖИВАННЯ

Оберіть номер (тільки вільні)
№501 - Пентхаус

01.08.2026 14 01.11.2026 14

ДОДАТКОВІ ПОСЛУГИ (ОБЕРІТЬ ТА ВКАЖІТЬ КІЛЬКІСТЬ)

☐	Доставка квітів та компліменту від готелю	1	2,000.00 грн
☐	Охоронювана парковка (доба)	1	300.00 грн
☒	Пізній виїзд (до 18:00)	1	1,200.00 грн
☒	Проживання з домашніми тваринами	1	1,000.00 грн
☒	Денний абонемент у SPA та басейн	14	1,800.00 грн
☐	Послуги няні (за 1 годину)	1	600.00 грн

ОПЛАТА

Оберіть тип оплати
Кредитна карта

СКАСУВАТИ ПІДТВЕРДИТИ

Рис. 27 Форма створення нового бронювання

На рис. 28 зображено інтерфейс аналітичного модуля програми, а саме вкладку «Звітність», яка розроблена для оцінювання фінансових результатів готелю керівником або власником готелю. Графік розподіляє прибуток по місяцях з відображенням підсумкових сум у гривнях над кожним стовпчиком.

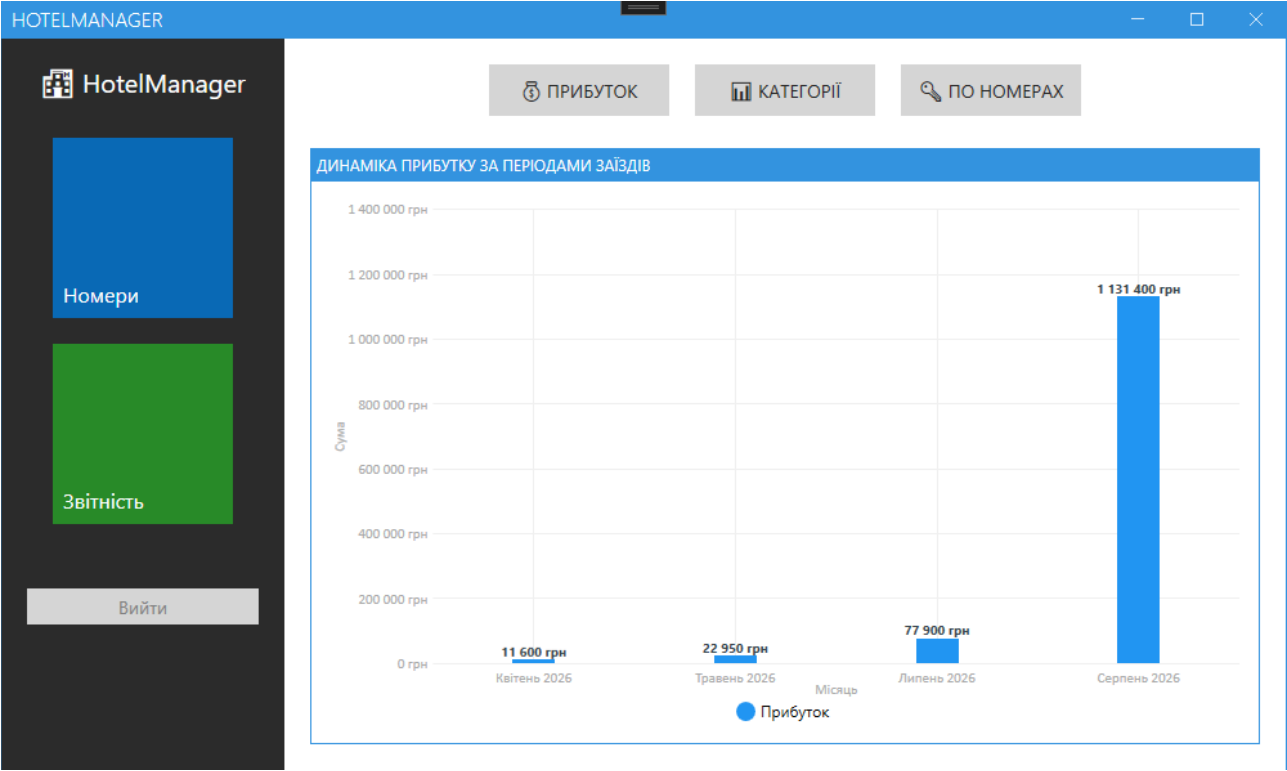


Рис. 28 Графік динаміки прибутку за кожен місяць

На рис. 29 зображено таблицю яка призначена для аналізу доходів у розрізі категорій номерного фонду, яка активується після натискання кнопки «КАТЕГОРІЇ» на верхній панелі.

HotelManager

ПРИБУТОК КАТЕГОРІЇ ПО НОМЕРАХ

Номери

Звітність

Вийти

ФІНАНСОВА ЕФЕКТИВНІСТЬ КАТЕГОРІЙ

КАТЕГОРІЯ	БРОНЮВАНЬ	ДОХІД (МІСЯЦЬ)	ДОХІД (ЗАГАЛЬНИЙ)
Двомісний	0	0 грн	0 грн
Делюкс	1	0 грн	77,900 грн
Економ	2	22,950 грн	34,550 грн
Пентхаус	1	0 грн	1,131,400 грн
Сімейний	0	0 грн	0 грн
Стандарт	0	0 грн	0 грн
Студія	0	0 грн	0 грн
Сюїт	0	0 грн	0 грн

На рис. 30 зображено таблицю детального аналізу прибутковості кожного номеру, яка викликається натисканням кнопки «ПО НОМЕРАХ».

HOTELMANAGER

HotelManager

Номери

Звітність

Вийти

ПРИБУТОК

КАТЕГОРІЇ

ПО НОМЕРАХ

ПРИБУТКОВІСТЬ НОМЕРІВ

№	ТИП	БРОНЮВАНЬ	ДОХІД (МІСЯЦЬ)	ЗАГАЛЬНИЙ ДОХІД
501	Пентхаус	1	0 грн	1,131,400 грн
301	Делюкс	1	0 грн	77,900 грн
104	Економ	1	22,950 грн	22,950 грн
101	Економ	1	0 грн	11,600 грн
102	Економ	0	0 грн	0 грн
103	Економ	0	0 грн	0 грн
502	Пентхаус	0	0 грн	0 грн
503	Пентхаус	0	0 грн	0 грн
504	Пентхаус	0	0 грн	0 грн
505	Пентхаус	0	0 грн	0 грн
506	Сюїт	0	0 грн	0 грн
507	Сюїт	0	0 грн	0 грн
508	Сюїт	0	0 грн	0 грн
509	Сюїт	0	0 грн	0 грн
510	Сюїт	0	0 грн	0 грн

ВИСНОВОК

У ході виконання дипломної роботи було реалізовано програмний комплекс інформаційної системи обліку апартаментів у готелі, призначений для цифровізації процесів обліку та координації номерного фонду в готельному бізнесі. Створена інформаційна система дозволяє повністю перевести в автоматичний режим операції, пов'язані з реєстрацією клієнтів, бронюванням апартаментів та розподілом житлової площі. Завдяки впровадженню цього рішення суттєво полегшується контроль за категоріями кімнат, зводиться до мінімуму ймовірність помилок при оформленні замовлень, а також підвищується загальна продуктивність персоналу служби прийому та розміщення.

Отримані результати підтверджують повне досягнення поставленої мети та успішне вирішення всіх практичних завдань. Розроблений продукт відповідає актуальним критеріям до сучасного програмного забезпечення, виділяючись ергономікою візуального інтерфейсу на базі технології WPF та високою швидкістю обробки інформаційних запитів за допомогою технології ADO.NET.

Спроектowana система детально враховує специфіку готельних комплексів, забезпечуючи відстеження зайнятості приміщень та фіксацію наданих додаткових сервісів. Застосування сучасних інструментів СУБД Microsoft SQL Server дозволило перенести ключові алгоритми перевірки даних безпосередньо на сторону сервера. Використання збережених T-SQL процедур та тригерів автоматично підтримує актуальність статусів кімнат та забезпечує логічну цілісність інформаційного середовища під час виїзду чи заселення гостей готелю.

Окремим технологічним здобутком є інтеграція засобів графічної візуалізації, що дозволяють трансформувати табличні звіти в наочні інтерактивні діаграми прибутковості. Аналіз фінансової ефективності за категоріями та окремими номерами здійснюється у реальному часі, що надає керівництву можливість швидкого оцінювання рентабельності підприємства. Таке рішення

значно зменшує витрати часу на ведення паперового обліку, мінімізує адміністративне навантаження та безпосередньо впливає на покращення якості сервісу.

Розроблена система може бути впроваджена у приватних готелях, апартаментах, хостелах та відпочинкових комплексах малого й середнього бізнесу. На основі отриманого під час проектування досвіду доцільно виділити напрямки для подальшої модернізації системи. Перспективним розвитком проекту є створення додаткового клієнтського веб-інтерфейсу для віддаленого онлайн-бронювання номерів через браузер, розширення функцій миттєвого оповіщення клієнтів через сучасні канали зв'язку, а також розробка інструментів для нарахування бонусів постійним гостям.

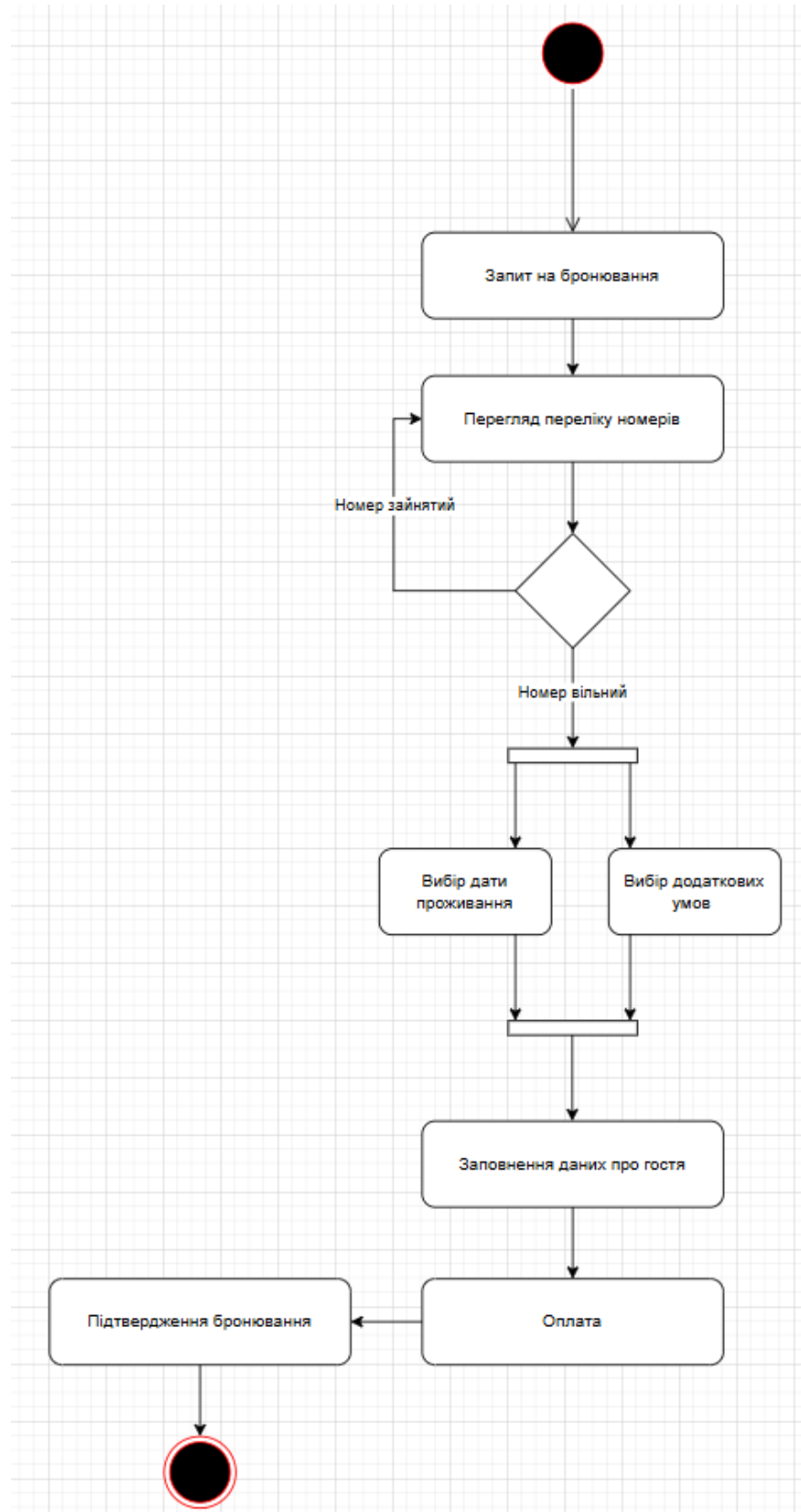
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клієнт-серверна архітектура. *QATestLab*.
URL: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture/> (дата звернення: 20.05.2026).
2. Мікула М., Коцюк Ю., Мікула О. Організація баз даних та знань. Вид-во Нац. ун-ту «Острозька акад.», 2021. URL: <https://doi.org/10.25264/978-617-8041-08-3> (дата звернення: 20.05.2026).
3. Платформа .NET та мова програмування C# 8.0. Тернопіль : © ФОП Паляниця В. А., 2020 © І. В. Коноваленко, П.О Марущак, 2020, 2020. 320 с. URL: <http://elartu.tntu.edu.ua/handle/lib/32825> (дата звернення: 21.05.2026).
4. Сукач М., Скопень М. Інформаційні системи і технології в готельно-ресторанному та туристичному бізнесі: підручник. Ліра-К, 2017.
URL: <https://doi.org/10.26884/mks.t1627> (дата звернення: 21.05.2026).
5. Як будувати uml-діаграми. *Dou.ua*.
URL: <https://dou.ua/forums/topic/40575/> (дата звернення: 21.05.2026).
6. Azure application architecture fundamentals - azure architecture center. *Microsoft Learn: Build with answers in reach*.
URL: <https://learn.microsoft.com/en-us/azure/architecture/guide/> (дата звернення: 20.05.2026).
7. Bai Y. Introduction to ADO.NET. *SQL server database programming with C#*. Boca Raton, 2023. URL: <https://doi.org/10.1201/9781003319832-3> (дата звернення: 21.05.2026).
8. Cloudbeds: #1 award-winning hospitality management system. <https://www.cloudbeds.com/>. URL: <https://www.cloudbeds.com/> (дата звернення: 21.05.2026).
9. Date C. J. Introduction to database systems. Pearson Education, Limited, 2003. 1024 p.

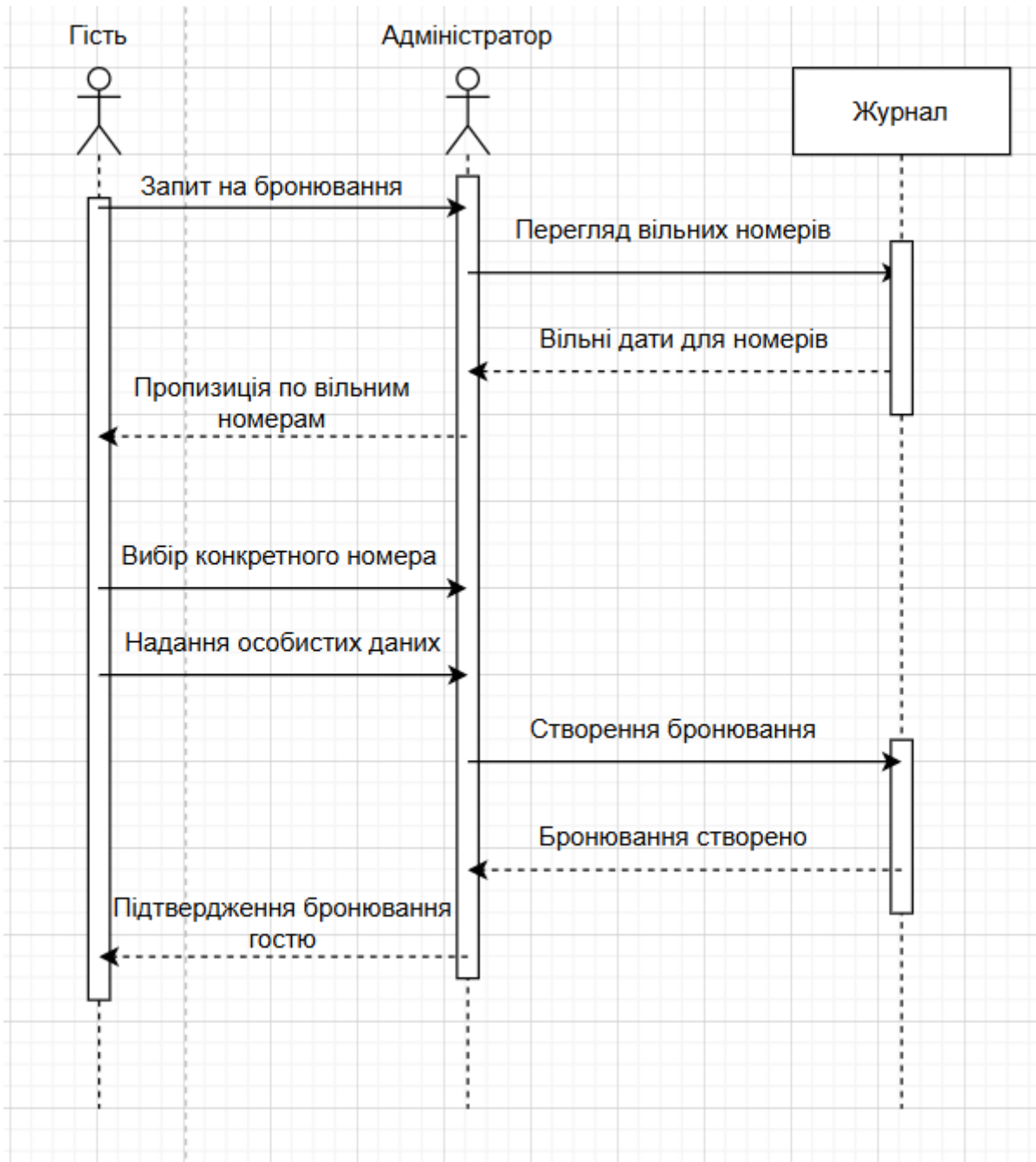
10. Inno setup help. *jrsoftware.org* // *Jordan Russell's Software*.
URL: <https://jrsoftware.org/ishelp/> (date of access: 20.05.2026).
11. ISTQB standard glossary of terms used in software testing / international software testing qualifications board. *ISTQB Glossary*.
URL: https://glossary.istqb.org/en_US/search?term=&exact_matches_first=true (дата звернення: 20.05.2026).
12. Lindemulder G., Kosinski M. What Is Role-Based Access Control (RBAC)? | IBM. *IBM*. URL: <https://www.ibm.com/think/topics/rbac> (дата звернення: 20.05.2026).
13. LiveCharts2. *LiveCharts2*. URL: <https://livecharts.dev/> (дата звернення: 20.05.2026).
14. MahApps.Metro documentation. *MahApps.Metro*.
URL: <https://mahapps.com/> (дата звернення: 20.05.2026).
15. Opera cloud PMS | oracle hospitality.
URL: <https://www.oracle.com/ua/hospitality/hotel-property-management/hotel-pms-software/> (дата звернення: 20.05.2026).
16. Transact-SQL reference (database engine) - SQL server. *Microsoft Learn: Build with answers in reach*.
URL: <https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver17> (дата звернення: 20.05.2026).
17. Troelsen A., Japikse P. Pro C# 9 with .NET 5: foundational principles and practices in programming. Apress L. P., 2021.
18. T-SQL fundamentals. Microsoft Press, 2023.
19. UML® - unified modeling language | object management group. *Object Management Group*. URL: <https://www.omg.org/uml/> (дата звернення: 20.05.2026).
20. Windows Presentation Foundation for .NET documentation. *Microsoft Learn: Build with answers in reach*.
URL: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/> (дата звернення: 20.05.2026).

ДОДАТОК А

Діаграма діяльності предметної області



Діаграма послідовності предметної області



Код створення таблиць

```
USE HotelManagement;
GO
DROP TABLE IF EXISTS Receipt;
DROP TABLE IF EXISTS BookingService;
DROP TABLE IF EXISTS Booking;
DROP TABLE IF EXISTS Guest;
DROP TABLE IF EXISTS Service_;
DROP TABLE IF EXISTS Room;
GO
CREATE TABLE Room (
    room_number CHAR(4) PRIMARY KEY,
    room_type VARCHAR(50),
    room_price DECIMAL(10, 2) NOT NULL,
    room_description TEXT,
    available BIT DEFAULT 1
);
CREATE TABLE Service_ (
    service_id CHAR(2) PRIMARY KEY,
    service_names VARCHAR(100) NOT NULL,
    service_price DECIMAL(10, 2) NOT NULL
);
CREATE TABLE Guest (
    guest_id CHAR(10) PRIMARY KEY,
    guest_name VARCHAR(100) NOT NULL,
    guest_email VARCHAR(100),
    guest_phone VARCHAR(20),
    guest_document VARCHAR(50)
);
CREATE TABLE Booking (
    booking_id CHAR(10) PRIMARY KEY,
    guest_id CHAR(10) NOT NULL,
    room_number CHAR(4),
    arrival_date DATE NOT NULL,
    departure_date DATE NOT NULL,
    payment_type VARCHAR(50),
    FOREIGN KEY (guest_id) REFERENCES Guest(guest_id) ON DELETE CASCADE,
    FOREIGN KEY (room_number) REFERENCES Room(room_number) ON DELETE SET NULL
);
CREATE TABLE BookingService (
    service_id CHAR(2),
    booking_id CHAR(10),
    quantity DECIMAL(5,2) DEFAULT 1,
    PRIMARY KEY (service_id, booking_id),
    FOREIGN KEY (service_id) REFERENCES Service_(service_id) ON DELETE CASCADE,
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id) ON DELETE CASCADE
);
CREATE TABLE Receipt (
    receipt_id CHAR(10) PRIMARY KEY,
    guest_id CHAR(10),
    booking_id CHAR(10),
    receipt_date DATE NOT NULL,
    contract_terms TEXT,
    sum_total DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (guest_id) REFERENCES Guest(guest_id) ON DELETE NO ACTION,
    FOREIGN KEY (booking_id) REFERENCES Booking(booking_id) ON DELETE CASCADE
);
```

Коди створення процедур

Процедура формування фінансової звітності

```

USE HotelManagement;
GO
CREATE OR ALTER PROCEDURE pr_GetRevenueReport
    @start_date DATE,
    @end_date DATE
AS
BEGIN
    SELECT
        COUNT(receipt_id) AS TotalReceipts,
        SUM(sum_total) AS TotalRevenue,
        AVG(sum_total) AS AverageCheck
    FROM Receipt
    WHERE receipt_date BETWEEN @start_date AND @end_date;

    -- Типи номерів, що принесли найбільше грошей
    SELECT
        r.room_type,
        SUM(rec.sum_total) AS RevenueByType
    FROM Receipt rec
    JOIN Booking b ON rec.booking_id = b.booking_id
    JOIN Room r ON b.room_number = r.room_number
    WHERE rec.receipt_date BETWEEN @start_date AND @end_date
    GROUP BY r.room_type
    ORDER BY RevenueByType DESC;
END;
GO

```

Процедура автоматичної генерації рахунку

```

USE HotelManagement;
GO
CREATE OR ALTER PROCEDURE pr_GenerateFinalReceipt
    @booking_id CHAR(10),
    @guest_id CHAR(10)
AS
BEGIN
    DECLARE @room_sum DECIMAL(10,2);
    DECLARE @services_sum DECIMAL(10,2);
    DECLARE @days INT;
    DECLARE @total DECIMAL(10,2);
    DECLARE @new_receipt_id CHAR(10);

    -- Генеруємо випадковий ID для чека
    SET @new_receipt_id = 'R' + LEFT(CAST(ABS(CHECKSUM(NEWID()))) AS VARCHAR), 9);

    -- Кількість днів
    SELECT @days = DATEDIFF(day, arrival_date, departure_date)
    FROM Booking WHERE booking_id = @booking_id;

    IF @days <= 0 SET @days = 1;

    -- Вартість проживання
    SELECT @room_sum = r.room_price * @days
    FROM Booking b JOIN Room r ON b.room_number = r.room_number
    WHERE b.booking_id = @booking_id;

    -- Вартість послуг
    SELECT @services_sum = ISNULL(SUM(s.service_price * bs.quantity), 0)
    FROM BookingService bs
    JOIN Service s ON bs.service_id = s.service_id
    WHERE bs.booking_id = @booking_id;

```

```

SET @total = @room_sum + @services_sum;

-- Перевірка на дублювання чеку
IF NOT EXISTS (SELECT 1 FROM Receipt WHERE booking_id = @booking_id)
BEGIN
    INSERT INTO Receipt (receipt_id, guest_id, booking_id, receipt_date,
contract_terms, sum_total)
VALUES (@new_receipt_id, @guest_id, @booking_id, GETDATE(), 'Оплачено
повністю', @total);

    SELECT * FROM Receipt WHERE receipt_id = @new_receipt_id;
END
ELSE
BEGIN
    SELECT * FROM Receipt WHERE booking_id = @booking_id;
END
END;

```

Процедура скасування бронювання

```

USE HotelManagement;
GO
CREATE OR ALTER PROCEDURE pr_CancelBooking
    @booking_id INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Booking WHERE booking_id = @booking_id)
    BEGIN
        DELETE FROM Booking WHERE booking_id = @booking_id;
        PRINT 'Бронювання успішно скасовано.';
    END
    ELSE
    BEGIN
        RAISERROR('Помилка: Бронювання з таким номером не знайдено.', 16, 1);
    END
END;
GO

```

Процедура пошуку вільних апартаментів

```

USE HotelManagement;
GO
CREATE PROCEDURE pr_SearchAvailableRooms
    @room_type VARCHAR(50) = NULL
AS
BEGIN
    SELECT room_number, room_type, room_price, room_description
    FROM Room
    WHERE available = 1
    AND (@room_type IS NULL OR room_type = @room_type);

    IF @@ROWCOUNT = 0
        PRINT 'Вільних номерів за вашим запитом не знайдено.';
END;
GO

```

Коди створення тригерів

Тригер контролю зайнятості номера

```
USE HotelManagement;
GO
CREATE OR ALTER TRIGGER trg_CheckRoomStatus
ON Booking
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM Room r
        JOIN inserted i ON r.room_number = i.room_number
        WHERE r.available = 0
    )
    BEGIN
        PRINT 'Помилка: Цей номер зараз зайнятий іншим гостем!';
    END
    ELSE
    BEGIN
        INSERT INTO Booking (room_number, arrival_date, departure_date, payment_type)
        SELECT room_number, arrival_date, departure_date, payment_type FROM inserted;
    END
END;
GO
```

Тригер відновлення доступності номера

```
USE HotelManagement;
GO
CREATE OR ALTER TRIGGER trg_AfterBookingInsert
ON Booking
AFTER INSERT
AS
BEGIN
    UPDATE Room
    SET available = 0
    FROM Room
    INNER JOIN inserted i ON Room.room_number = i.room_number;
END;
GO
```

Код створення користувачів

```
USE HotelManagement;
GO

DROP USER IF EXISTS hotel_owner;
DROP USER IF EXISTS hotel_manager;
GO
DROP LOGIN hotel_owner;
DROP LOGIN hotel_manager;
GO

CREATE LOGIN hotel_owner WITH PASSWORD = 'owner123';
CREATE LOGIN hotel_manager WITH PASSWORD = 'manager123';
GO
CREATE USER hotel_owner FOR LOGIN hotel_owner;
CREATE USER hotel_manager FOR LOGIN hotel_manager;
GO

GRANT SELECT ON Room TO hotel_owner;
GRANT SELECT ON Guest TO hotel_owner;
GRANT SELECT ON Booking TO hotel_owner;
GRANT SELECT ON Service_ TO hotel_owner;
GRANT SELECT ON BookingService TO hotel_owner;
GRANT SELECT ON Receipt TO hotel_owner;
GO

GRANT SELECT, INSERT, UPDATE, DELETE ON Guest TO hotel_manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON Booking TO hotel_manager;
GRANT SELECT ON Service_ TO hotel_manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON BookingService TO hotel_manager;
GRANT SELECT ON Room TO hotel_manager;
GRANT UPDATE(available) ON Room TO hotel_manager;
GRANT SELECT, INSERT ON Receipt TO hotel_manager;
GRANT EXECUTE ON pr_GenerateFinalReceipt TO hotel_manager;
GO
```

Коди реалізованих сервісів

Код методу авторизації

```
private void LoginButton_Click(object sender, RoutedEventArgs e)
{
    string user = LoginField.Text.Trim();
    string pass = PasswordField.Password;

    if (string.IsNullOrEmpty(user) || string.IsNullOrEmpty(pass))
    {
        MessageBox.Show("Будь ласка, введіть логін та пароль!");
        return;
    }

    // підключення
    string connStr = DbConfig.GenerateConnectionString(user, pass);

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        try
        {
            conn.Open();
            DbConfig.ConnectionString = connStr;

            if (user.ToLower() == "hotel_owner")
            {
                OwnerWindow ownerWin = new OwnerWindow();
                ownerWin.Show();
            }
            else
            {
                MainWindow managerWin = new MainWindow();
                managerWin.Show();
            }

            this.Close();
        }
        catch (SqlException)
        {
            MessageBox.Show("Помилка авторизації: Невірний логін або пароль.",
                "Відмовлено в доступі", MessageBoxButton.OK,
                MessageBoxImage.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Критична помилка підключення: " + ex.Message);
        }
    }
}
```

Код методу додавання інформації про клієнта

```
private void SaveClient_Click(object sender, RoutedEventArgs e)
{
    // 1. Збір даних
    string firstName = FirstNameTextBox.Text.Trim();
    string lastName = LastNameTextBox.Text.Trim();
    string phone = PhoneTextBox.Text.Trim();
    string email = EmailTextBox.Text.Trim();
    string doc = DocumentNumberTextBox.Text.Trim();

    // 2. Валідація
```

```

        if (string.IsNullOrEmpty(firstName) || string.IsNullOrEmpty(lastName) ||
            string.IsNullOrEmpty(doc))
        {
            MessageBox.Show("Будь ласка, заповніть обов'язкові поля: Ім'я, Прізвище та  
Номер документа.",
                            "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        // 3. Формуємо повне ім'я для БД (guest_name)
        string fullGuestName = $"{lastName} {firstName}";

        // 4. Генеруємо унікальний ID
        string guestId = Guid.NewGuid().ToString().Substring(0, 10).ToUpper();

        // 5. Запит у БД
        string sql = @"INSERT INTO Guest (guest_id, guest_name, guest_email,
            guest_phone, guest_document)
            VALUES (@id, @name, @email, @phone, @doc)";

        try
        {
            using (SqlConnection connection = new
                SqlConnection(DbConfig.ConnectionString))
            {
                SqlCommand command = new SqlCommand(sql, connection);

                command.Parameters.AddWithValue("@id", guestId);
                command.Parameters.AddWithValue("@name", fullGuestName);
                command.Parameters.AddWithValue("@email", email);
                command.Parameters.AddWithValue("@phone", phone);
                command.Parameters.AddWithValue("@doc", doc);
                connection.Open();
                command.ExecuteNonQuery();

                MessageBox.Show("Клієнта успішно зареєстровано в базі!", "Успіх",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);

                if (NavigationService.CanGoBack)
                    NavigationService.GoBack();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Помилка при збереженні: {ex.Message}", "Помилка БД",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

Код методу створення бронювання

```

private void LoadFormData()
{
    try
    {
        using (SqlConnection conn = new SqlConnection(DbConfig.ConnectionString))
        {
            conn.Open();

            //Клієнти
            var guests = new List<object>();
            SqlCommand cmdG = new SqlCommand("SELECT guest_id, guest_name FROM
                Guest", conn);

```

```

        using (var r = cmdG.ExecuteReader())
            while (r.Read()) guests.Add(new { ID = r["guest_id"].ToString(),
Name = r["guest_name"].ToString() });
        GuestComboBox.ItemsSource = guests;
        GuestComboBox.DisplayMemberPath = "Name";
        GuestComboBox.SelectedValuePath = "ID";

        //Вільні номери
        var rooms = new List<object>();
        SqlCommand cmdR = new SqlCommand("SELECT room_number, room_type FROM
Room WHERE available = 1", conn);
        using (var r = cmdR.ExecuteReader())
            while (r.Read()) rooms.Add(new
            {
                Number = r["room_number"].ToString().Trim(),
                Display = $"{r["room_number"].ToString().Trim()} -
{r["room_type"]}"
            });
        RoomComboBox.ItemsSource = rooms;
        RoomComboBox.DisplayMemberPath = "Display";
        RoomComboBox.SelectedValuePath = "Number";

        // Послуги
        var services = new List<ServiceItem>();
        SqlCommand cmdS = new SqlCommand("SELECT service_id, service_names,
service_price FROM Service_", conn);
        using (var r = cmdS.ExecuteReader())
            while (r.Read()) services.Add(new ServiceItem
            {
                Id = r["service_id"].ToString().Trim(),
                Name = r["service_names"].ToString(),
                Price = (decimal)r["service_price"],
                Quantity = 1
            });
        ServicesListBox.ItemsSource = services;
    }
}
catch (Exception ex) { MessageBox.Show("Помилка БД: " + ex.Message); }
}

private void ConfirmBooking_Click(object sender, RoutedEventArgs e)
{
    if (GuestComboBox.SelectedValue == null || RoomComboBox.SelectedValue == null ||
ArrivalDatePicker.SelectedDate == null || DepartureDatePicker.SelectedDate
== null)
    {
        MessageBox.Show("Заповніть усі поля проживання!"); return;
    }

    string bId = "B" + DateTime.Now.Ticks.ToString().Substring(10, 8);
    string gId = GuestComboBox.SelectedValue.ToString();
    string rNum = RoomComboBox.SelectedValue.ToString();
    string payType = (PaymentTypeComboBox.SelectedItem as
ComboBoxItem)?.Content.ToString() ?? "Готівка";

    using (SqlConnection conn = new SqlConnection(DbConfig.ConnectionString))
    {
        conn.Open();
        SqlTransaction trans = conn.BeginTransaction();
        try
        {
            // 1. Вставка в Booking

```

```

        string sqlB = @"INSERT INTO Booking (booking_id, guest_id, room_number,
arival_date, departure_date, payment_type)
        VALUES (@bid, @gid, @rnum, @start, @end, @pay)";
        SqlCommand cmdB = new SqlCommand(sqlB, conn, trans);
        cmdB.Parameters.AddWithValue("@bid", bId);
        cmdB.Parameters.AddWithValue("@gid", gId);
        cmdB.Parameters.AddWithValue("@rnum", rNum);
        cmdB.Parameters.AddWithValue("@start", ArrivalDatePicker.SelectedDate);
        cmdB.Parameters.AddWithValue("@end", DepartureDatePicker.SelectedDate);
        cmdB.Parameters.AddWithValue("@pay", payType);
        cmdB.ExecuteNonQuery();

        // 2. Вставка послуг у BookingService
        var selectedServices = (ServicesListBox.ItemsSource as
List<ServiceItem>).Where(s => s.IsSelected);
        foreach (var s in selectedServices)
        {
            string sqlBS = "INSERT INTO BookingService (service_id, booking_id,
quantity) VALUES (@sid, @bid, @qty)";
            SqlCommand cmdBS = new SqlCommand(sqlBS, conn, trans);
            cmdBS.Parameters.AddWithValue("@sid", s.Id);
            cmdBS.Parameters.AddWithValue("@bid", bId);
            cmdBS.Parameters.AddWithValue("@qty", s.Quantity > 0 ? s.Quantity :
1);
            cmdBS.ExecuteNonQuery();
        }

        // 3. Оновлення статусу номера
        string sqlR = "UPDATE Room SET available = 0 WHERE room_number = @rnum";
        SqlCommand cmdR = new SqlCommand(sqlR, conn, trans);
        cmdR.Parameters.AddWithValue("@rnum", rNum);
        cmdR.ExecuteNonQuery();

        // 4. Генерація чеку
        SqlCommand cmdProc = new SqlCommand("pr_GenerateFinalReceipt", conn,
trans);
        cmdProc.CommandType = CommandType.StoredProcedure;
        cmdProc.Parameters.AddWithValue("@booking_id", bId);
        cmdProc.Parameters.AddWithValue("@guest_id", gId);
        cmdProc.ExecuteNonQuery();

        trans.Commit();
        MessageBox.Show($"Бронювання {bId} та чек оформлені успішно!");
        NavigationService.GoBack();
    }
    catch (Exception ex)
    {
        trans.Rollback();
        MessageBox.Show("Помилка збереження: " + ex.Message);
    }
}
}

```

ДОДАТОК К

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Проценко Дмитро Вадимович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Інформаційна система обліку апартаментів у готелі» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - ☐ інструменти генеративного ШІ не використовувалися.
 - ☐ інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - ☐ перекладу іншомовних джерел;
 - ☐ стилістичного редагування та перевірки граматики;
 - ☐ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

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

«__» _____ 2026 р. _____ Дмитро ПРОЦЕНКО
(підпис)