МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 12.00.00.000 ПЗ

Група ІПм-23-3

Петрів Валентин

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Петрів Валентин Миколайович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми для автоматизації бізнес-процесів готельного комплексу

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Петрів В.М.
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Григорчук Любомир Іванович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту
Завідувач кафедри
доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль
доц. Вовк Р.Б.
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
‘04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Петрів Валентин Миколайович

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі, методи та алгоритми для автоматизації бізнес-процесів готельного комплексу **”**
керівник проекту (роботи) Григорчук Любомир Іванович, к.т.н., доцент
затверджені наказом закладу вищої освіти від **“** 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Моделі, методи та алгоритми для автоматизації бізнес-процесів готельного комплексу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)
1. Аналіз предметної області бізнес-процесів готельного комплексу
2. Розробка моделей і методів автоматизації бізнес-процесів
3. Реалізація інформаційної системи для автоматизації бізнес-процесів
4. Розробка моделі бази даних

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
1. Property Information Management System (Рисунок 1.1 ст. 19)
2. Компоненти та інтерфейси програми (Рисунок 2.1, ст. 34)
3. ER-діаграма БД (Рисунок 2.2, ст.39)
4. UML діаграма прецендентів (Рисунок 2.3, ст.42)
5. Інтерфейс головної сторінки системи (Рисунок 3.1, ст.57)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз та вивчення літературних джерел	20.09.2024	виконано
2	Огляд існуючих технологій та підходів до автоматизації	01.10.2024	виконано
3	Розробка концепції інформаційної системи	12.10.2024	виконано
4	Опис архітектури та компонентів інформаційної системи	25.10.20234	виконано
5	Опис моделей та алгоритмів для автоматизації готельного бізнесу	05.11.2024	виконано
6	Програмна реалізація інформаційної системи	22.11.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 86 с., 22 рис., 1 табл., 45 джерел.

Тема: Моделі, методи та алгоритми для автоматизації бізнес-процесів готельного комплексу.

Об’єкт дослідження: моделі, методи та алгоритми для автоматизації процесів управління готельним комплексом.

Мета роботи: створення ефективної програмної системи автоматизації готельного комплексу на основі сучасних інформаційних технологій із використанням мови програмування C# та платформи .NET.

Предмет дослідження: програмні моделі, методи оптимізації бізнес-процесів, алгоритми прогнозування попиту та управління ресурсами готельного комплексу.

Висновок:

У результаті виконання магістерської роботи створено програмну систему автоматизації готельного комплексу, яка забезпечує централізоване управління бізнес-процесами, високу надійність та продуктивність. Запропоноване рішення дозволяє автоматизувати операції бронювання, фінансовий облік, управління персоналом і номерним фондом, а також здійснювати аналітику даних для підвищення ефективності роботи готелю.

АВТОМАТИЗАЦІЯ БІЗНЕС-ПРОЦЕСІВ, УПРАВЛІННЯ ГОТЕЛЬНИМ КОМПЛЕКСОМ, C#, .NET, БАЗА ДАНИХ MYSQL, ОПТИМІЗАЦІЯ РЕСУРСІВ, ПРОГНОЗУВАННЯ ПОПИТУ, ЗВІТНІСТЬ.

ANNOTATION

Master's work: 86 p., 22 fig., 1 tab., 45 sources.

Topic: Models, Methods, and Algorithms for Automating Business Processes of a Hotel Complex.

Object of Research: Models, methods, and algorithms for automating the management processes of a hotel complex.

Purpose of the Study: To develop an efficient software system for automating the business processes of a hotel complex using modern information technologies, leveraging the C# programming language and the .NET platform.

Subject of Research: Software models, business process optimization methods, demand forecasting algorithms, and resource management for hotel complexes.

Conclusion:

As a result of the master's thesis, a software system for automating the business processes of a hotel complex was developed, providing centralized management, high reliability, and productivity. The proposed solution enables the automation of operations such as booking management, financial accounting, personnel management, and room management, while also facilitating data analytics to enhance the hotel's operational efficiency.

BUSINESS PROCESS AUTOMATION, HOTEL COMPLEX MANAGEMENT, C#, .NET, MYSQL DATABASE, RESOURCE OPTIMIZATION, DEMAND FORECASTING, REPORTING.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API - Прикладний програмний інтерфейс

CRM - Система управління взаємовідносинами з клієнтами

ERP - Система планування ресурсів підприємства

BPMN - Нотація для моделювання бізнес-процесів

UML - Уніфікована мова моделювання

ARIMA - Авторегресивна інтегрована модель ковзного середнього

ВСТУП

Актуальність роботи

1. Підвищення конкурентоспроможності та важливість автоматизації у готельному бізнесі обговорюються в огляді розвитку IT-рішень для автоматизації готельних бізнес-процесів, включаючи такі інструменти, як CRM і ERP. Згідно з цим, автоматизація знижує витрати та покращує обслуговування клієнтів [2].

2. Оптимізація бізнес-процесів шляхом автоматизації дозволяє уникати помилок і ефективніше керувати ресурсами, що особливо актуально для великих готельних комплексів [1].

3. Технологічний прогрес у вигляді BPMN, ERP-систем і хмарних платформ забезпечує інтеграцію процесів і підвищення ефективності роботи готелів. Ці аспекти актуальні для сучасної готельної індустрії, яка адаптується до змін на ринку [3].

4. Зростання клієнтських очікувань висвітлено в статтях, які підкреслюють потребу у швидких і персоналізованих послугах для задоволення сучасного клієнта в індустрії гостинності [3].

Порівняння роботи з відомими розв'язаннями проблеми

У сфері автоматизації готельного бізнесу вже існують численні рішення, такі як хмарні платформи, інтегровані ERP-системи та спеціалізовані CRM-інструменти. Однак більшість із них або занадто дорогі для невеликих і середніх готельних комплексів, або не враховують специфіку локальних ринків.

Особливістю розробленої в межах цієї роботи системи є її адаптація до потреб різних масштабів готельного бізнесу, можливість інтеграції з іншими платформами та модульність, що забезпечує розширення функціоналу в майбутньому.

Мета і задачі дослідження

Метою магістерської роботи є розробка моделей, методів та алгоритмів для автоматизації бізнес-процесів готельного комплексу, спрямованих на оптимізацію управління ресурсами, підвищення якості обслуговування клієнтів та ефективності роботи персоналу.

Розроблена система повинна забезпечувати автоматизоване управління основними бізнес-процесами, такими як бронювання номерів, облік клієнтів та фінансових операцій, а також мати інтеграцію з іншими інструментами управління. Для цього використовуватимуться сучасні технології моделювання процесів (BPMN) та об’єктно-орієнтованого проєктування (UML).

Досягнення мети включає вирішення наступних завдань:

1. Аналіз існуючих підходів до автоматизації бізнес-процесів у готельній сфері.
2. Моделювання основних бізнес-процесів готельного комплексу за допомогою BPMN.
3. Створення UML-діаграм для опису компонентів системи та їхньої взаємодії.
4. Розробка алгоритмів автоматизації управління бронюваннями, обслуговуванням клієнтів та персоналом.
5. Програмна реалізація десктопної системи управління готельним комплексом з використанням C# та MySQL.

Об’єкт дослідження: моделі, методи та алгоритми для автоматизації процесів управління готельним комплексом.

Предмет дослідження: програмні моделі, методи оптимізації бізнес-процесів, алгоритми прогнозування попиту та управління ресурсами готельного комплексу.

Методи дослідження

Для досягнення поставлених завдань використовувалися наступні методи:

– Моделювання процесів за допомогою BPMN і UML для опису бізнес-процесів.

– Алгоритми оптимізації для управління ресурсами та прогнозування попиту.

– Методи об'єктно-орієнтованого програмування для реалізації функціоналу системи.

Новизна роботи:

– Вдосконалено і реалізовано алгоритм прогнозування попиту та оптимізації ресурсів, які адаптовані до специфіки вдосконалення готельного розвитку.

– Вдосконалено роботу модулів при бронюванні номерів та якісного обслуговування клієнтів в єдиній програмній системі.

Теоретична значущість роботи:

– Використано методологічні підходи та алгоритми , такі як ARIMA для прогнозування роботи готельних підприємств, які можуть бути адаптовані до інших сфер бізнесу.

Практична значущість роботи:

– Розроблена система автоматизації може бути впроваджена в готельних комплексах різного масштабу для підвищення ефективності роботи.

– Система зменшує витрати на ручну працю, оптимізує управління ресурсами та покращує обслуговування клієнтів.

– Програму можна модифікувати для інтеграції з іншими CRM- та ERP-системами.

– Потенціал для масштабування: додавання модулів мобільного доступу, інтеграція з онлайн-платформами бронювання (наприклад, Booking.com).

– Завдяки автоматизації бізнес-процесів, готель може знизити витрати на управління та обслуговування, збільшуючи рентабельність.

– Готелі, які впроваджують сучасні системи управління, можуть залучати більше клієнтів за рахунок покращеного сервісу та ефективного управління.

Особистий внесок студента

– Аналіз існуючих підходів і вибір оптимальної моделі для автоматизації бізнес-процесів.

– Розробка математичної моделі та алгоритмів автоматизації.

– Програмна реалізація системи управління готельним комплексом.

Структура магістерської роботи.

Магістерська робота викладена на 86 сторінках друкованого тексту, який складається з вступу, трьох розділів, висновків, списку використаних джерел (45 найменувань). Робота містить 22 рисунків і 1 таблиця.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ БІЗНЕС-ПРОЦЕСІВ ГОТЕЛЬНОГО КОМПЛЕКСУ

1.1 Опис функціонування готельного комплексу як об'єкта автоматизації.

Готельний комплекс – це організація, що надає послуги розміщення, харчування, дозвілля, а також додаткові сервіси для своїх клієнтів. Діяльність готелю охоплює широкий спектр процесів, які потребують ефективного управління, координації та контролю.

Загальна характеристика діяльності готелю

Основна мета готелю полягає у забезпеченні комфорту для гостей та максимізації доходів через оптимальне використання своїх ресурсів. Діяльність готельного комплексу поділяється на кілька ключових напрямів:

1. Управління номерним фондом:
   - Моніторинг стану номерів (вільні, зайняті, потребують прибирання).
   - Контроль доступності номерів для бронювання.
   - Оновлення інформації про вартість проживання залежно від сезону, завантаженості чи попиту.
2. Управління персоналом:
   - Розподіл робочих змін серед працівників.
   - Контроль продуктивності та виконання обов’язків.
   - Забезпечення підвищення кваліфікації працівників (рецепція, обслуговуючий персонал).
3. Обслуговування клієнтів:
   - Надання основних послуг проживання та харчування.
   - Організація додаткових сервісів (наприклад, транспорт, екскурсії, конференц-зали).
   - Взаємодія з клієнтами через веб-сайти, мобільні додатки чи кол-центри.
4. Фінансовий облік і звітність:

- Автоматизований розрахунок вартості проживання та послуг.
- Облік платежів (готівкових, безготівкових, через платіжні системи).
- Формування фінансових звітів для керівництва.

Основні завдання готельного комплексу

Автоматизація в готельному комплексі спрямована на вирішення таких завдань:

1. Бронювання:
   - Забезпечення швидкого і точного оброблення заявок на бронювання через різні канали (онлайн-системи, агентства, рецепція).
   - Впровадження динамічного ціноутворення, яке залежить від попиту та сезону.
2. Надання послуг:
   - Спрощення процедур реєстрації та виселення гостей через інтеграцію систем керування.
   - Забезпечення якості послуг (прибирання, доставка їжі тощо).
3. Контроль оплат:
   - Облік оплат у реальному часі, включаючи часткові платежі чи аванси.
   - Зменшення помилок при обліку завдяки автоматизації.

Роль автоматизації

Автоматизація діяльності готелю дозволяє значно підвищити ефективність бізнесу:

- Знижує витрати часу на рутинні операції.
- Покращує якість обслуговування клієнтів.
- Дозволяє проводити аналіз завантаженості номерного фонду та оптимізувати використання ресурсів [10].

1.2 Ідентифікація ключових бізнес-процесів

Функціонування готельного комплексу залежить від ряду ключових бізнес-процесів, які спрямовані на забезпечення високої якості обслуговування клієнтів,

ефективного управління ресурсами та оптимізації операцій. Основні бізнес-процеси готельного комплексу включають:

1. Бронювання

Процес бронювання є першим етапом взаємодії клієнта з готелем та охоплює:

- Прийом запитів через різні канали: веб-сайти, мобільні додатки, агентства чи телефонні дзвінки.
- Перевірку доступності номерів у реальному часі та резервування.
- Застосування алгоритмів динамічного ціноутворення, які враховують сезонність, попит та пропозиції.
- Управління бронюваннями: підтвердження, зміна або скасування заявок.

Автоматизація цього процесу зменшує час обробки заявок, мінімізує ризик помилок та дозволяє використовувати аналітику для прогнозування попиту.

2. Обслуговування клієнтів

Цей процес охоплює весь цикл взаємодії клієнта з готелем:

Реєстрація гостей при заселенні: перевірка бронювання, оформлення документів, надання інформації про послуги.

Обслуговування під час перебування:

- Надання базових послуг (проживання, харчування).
- Реалізація додаткових послуг, таких як трансфер, екскурсії, пральня.
- Обробка запитів та скарг клієнтів.

Виселення клієнтів: формування рахунків, перевірка оплати, збір відгуків.

Автоматизація дозволяє персоналізувати обслуговування, підвищити швидкість вирішення запитів та забезпечити позитивний досвід клієнтів.

3. Управління ресурсами

Ефективне управління ресурсами забезпечує стабільну роботу готелю та мінімізацію витрат. Основні аспекти включають:

- Керування номерним фондом: контроль зайнятості номерів, їхнього стану та відповідності вимогам клієнтів.
- Управління персоналом: планування робочих змін, контроль виконання завдань, забезпечення навчання та мотивації працівників.

– Контроль запасів і матеріалів: моніторинг рівня запасів, управління закупівлями.

– Фінансове управління: планування бюджету, контроль витрат і доходів, оптимізація процесів.

Автоматизовані системи допомагають ефективно розподіляти ресурси, передбачати пікові навантаження та адаптувати операції до поточних умов.

4. Управління даними

Окремим важливим процесом є збір, обробка та аналіз даних:

– Ведення клієнтської бази для повторних продажів.

– Збір статистики щодо бронювань, використання послуг та відгуків.

– Використання аналітики для виявлення трендів, прогнозування попиту та планування майбутніх стратегій [11].

1.3 Аналіз сучасних підходів до автоматизації готельного бізнесу.

Автоматизація готельного бізнесу стає все більш необхідною через зростаючий попит на послуги, підвищення конкуренції та розвиток цифрових технологій. Сучасні підходи до автоматизації охоплюють інтеграцію інноваційних інформаційних систем, алгоритмів оптимізації та інструментів прогнозування. Нижче розглянуті основні підходи, які використовуються в готельному секторі.

1.3.1 Використання спеціалізованих інформаційних систем

Готельні комплекси впроваджують інформаційні системи для управління ключовими аспектами бізнесу:

– Property Management System (PMS) – системи управління готелем для автоматизації бронювання, реєстрації клієнтів, управління номерним фондом.

– Customer Relationship Management (CRM) – системи для управління взаємовідносинами з клієнтами, що забезпечують персоналізацію обслуговування.

– Revenue Management Systems (RMS) – для динамічного ціноутворення та оптимізації доходів.

Ці системи дозволяють інтегрувати всі бізнес-процеси в єдину платформу, забезпечуючи централізоване управління.

1.3.2 Хмарні технології

Хмарні рішення надають можливість доступу до даних у реальному часі з будь-якого пристрою, що є критично важливим для:

- Забезпечення безперебійної роботи готелю.
- Синхронізації даних між різними відділами та підрозділами.
- Зниження витрат на інфраструктуру та обслуговування серверів.

Хмарні платформи також забезпечують масштабованість і легку інтеграцію з іншими системами.

1.3.3 Автоматизація операційних процесів

Інтеграція спеціалізованого програмного забезпечення дозволяє автоматизувати такі процеси:

- Бронювання: онлайн-сервіси, інтегровані з глобальними системами дистрибуції (GDS), дозволяють клієнтам здійснювати бронювання в декілька кліків.
- Оплата та білінг: впровадження платіжних шлюзів для швидких і безпечних транзакцій.
- Управління персоналом: автоматизоване планування робочих змін і контроль завдань працівників.

1.3.4 Використання алгоритмів та аналітики даних

Аналітика великих даних (Big Data) та алгоритми штучного інтелекту (AI) відіграють важливу роль у прогнозуванні попиту та прийнятті рішень.

- Алгоритми прогнозування (ARIMA, машинне навчання) дозволяють передбачати заповнюваність номерного фонду, пік попиту, обсяги бронювань.
- Аналітика поведінки клієнтів забезпечує розробку персоналізованих пропозицій.

– Оптимізація ресурсів на основі моделей розподілу, що враховують сезонність та зміну потреб.

1.3.5 Інтеграція IoT і мобільних технологій

Розумні пристрої та мобільні додатки трансформують обслуговування клієнтів:

– Smart Room Systems дозволяють керувати освітленням, кліматом та іншими послугами в номері через додаток.

– Мобільні додатки для клієнтів забезпечують можливість онлайн-реєстрації, замовлення послуг та комунікації з персоналом.

– IoT-рішення для управління ресурсами дозволяють моніторити стан обладнання, енергоспоживання та технічний стан номерів.

1.3.6 Автоматизація маркетингу та продажів

– Інтеграція з OTA (Online Travel Agencies) для залучення клієнтів через популярні платформи бронювання (Booking.com, Expedia тощо).

– Автоматизовані системи email-маркетингу для залучення клієнтів, повідомлень про акції та утримання клієнтської бази.

– Соціальні мережі та онлайн-реклама автоматизуються через спеціальні платформи для таргетингу та аналітики [12].

1.4 Огляд інформаційних систем і технологій, які використовуються в готельному секторі

Сучасний готельний сектор активно використовує інформаційні системи та технології для автоматизації бізнес-процесів, підвищення ефективності роботи і забезпечення високого рівня обслуговування клієнтів. У цьому розділі розглянуто основні типи систем і технологій, які знаходять застосування в готельному бізнесі.

1. Системи управління готелем (Property Management Systems – PMS)

PMS – це базові інформаційні системи, які використовуються для управління операційною діяльністю готелю. Їх основні функції:

– Управління бронюванням номерів.

– Реєстрація клієнтів та управління статусом номерного фонду.

– Контроль фінансових операцій та облік доходів і витрат. Популярні системи: Opera PMS, Cloudbeds, RoomRaccoon, Mews.

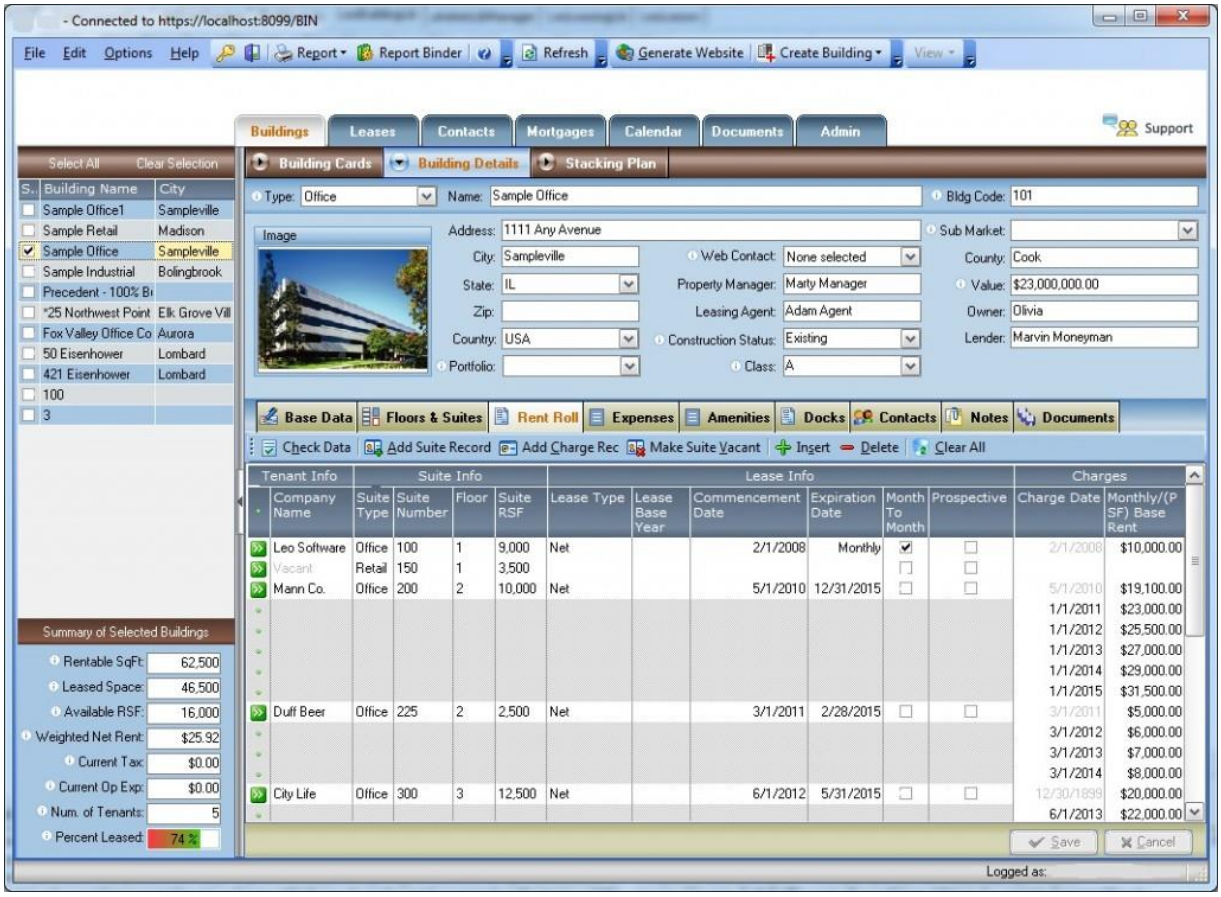

Рис. 1.1. Property Information Management System

2. Системи управління взаємовідносинами з клієнтами (Customer Relationship Management – CRM)

CRM-системи дозволяють автоматизувати процеси управління клієнтською базою.

– Збір і аналіз даних про клієнтів.

– Відстеження історії взаємодії з клієнтами.

– Розробка персоналізованих пропозицій і програм лояльності. Популярні системи: Salesforce CRM, HubSpot, Zoho CRM.

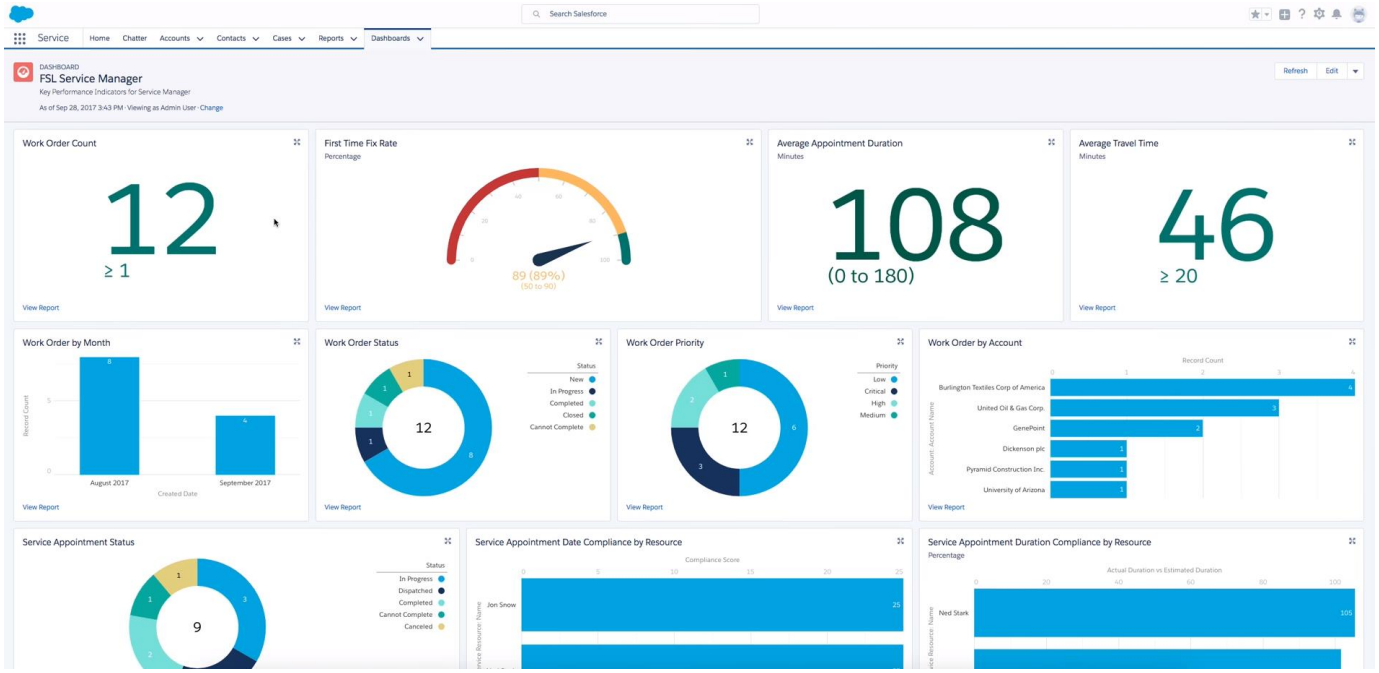

Рис. 1.2. Salesforce CRM

3. Системи динамічного ціноутворення (Revenue Management Systems – RMS)

RMS допомагають готелям оптимізувати ціни на послуги, враховуючи:

– Сезонність.

– Пік та спад попиту.

– Дані конкурентів.

Такі системи використовують алгоритми прогнозування та аналітику великих даних для максимізації прибутку.

Популярні рішення: IDeaS, Duetto, RateGain.

4. Глобальні дистрибутивні системи (Global Distribution Systems – GDS)

GDS – це платформи для інтеграції готелів із туристичними агентствами та онлайн-туроператорами. Вони дозволяють автоматично оновлювати доступність номерів і ціни на різних платформах.

Популярні GDS: Amadeus, Sabre, Travelport.

5. Хмарні технології

Хмарні системи забезпечують доступ до даних у реальному часі, спрощують інтеграцію з іншими інструментами та забезпечують масштабованість.

– Хмарні PMS: наприклад, Cloudbeds, Hostelworld.

- Хмарні сховища для централізованого зберігання та обробки даних.

6. Технології Big Data та аналітика даних

Системи аналітики дозволяють готелям використовувати великі обсяги даних для прогнозування попиту, аналізу клієнтської поведінки та оптимізації бізнес-процесів.

- Інструменти для аналізу даних: Power BI, Tableau, Google Analytics.
- Використання алгоритмів штучного інтелекту для сегментації клієнтів і розробки стратегій ціноутворення.

7. Інтеграція з онлайн-платформами та мобільними додатками

- OTA (Online Travel Agencies): готелі інтегрують свої системи з платформами типу Booking.com, Expedia, Airbnb для збільшення охоплення клієнтів.
- Мобільні додатки для клієнтів: надають доступ до послуг готелю, дозволяють здійснювати бронювання, онлайн-реєстрацію та спілкування з персоналом.

8. Рішення для автоматизації обслуговування клієнтів

- Системи самообслуговування: кіоски для самостійної реєстрації та виїзду клієнтів.
- Чат-боти: забезпечують відповіді на запити клієнтів у реальному часі.
- IoT (Internet of Things): дозволяють клієнтам управляти номером (освітлення, клімат) через мобільні додатки.

9. Системи управління персоналом (Human Resource Management Systems – HRMS)

HRMS допомагають автоматизувати такі процеси:

- Планування графіків роботи персоналу.
- Контроль виконання завдань.
- Оцінка продуктивності.

Популярні системи: BambooHR, SAP SuccessFactors, Workday [13].

1.5 Аналіз проблем та викликів впровадження автоматизації в готельному бізнесі

Автоматизація бізнес-процесів у готельному секторі відкриває великі можливості для підвищення ефективності, зменшення витрат і поліпшення обслуговування клієнтів. Однак впровадження сучасних технологій супроводжується низкою проблем та викликів.

1. Технічні проблеми

Несумісність систем

Багато готелів використовують старі інформаційні системи, які важко інтегрувати з сучасними інструментами, такими як CRM, GDS або мобільні додатки. Це може призводити до дублювання даних і помилок в управлінні.

Обмеження IT-інфраструктури

Не всі готелі мають достатньо розвинену технічну інфраструктуру (сервери, мережі, інтернет-з'єднання), щоб забезпечити стабільну роботу складних програмних рішень.

Безпека даних

Автоматизація передбачає обробку великої кількості конфіденційної інформації (особисті дані клієнтів, фінансові транзакції). Готелі стикаються з ризиками кіберзлочинності, витоку даних і шахрайства.

2. Економічні виклики

Висока вартість впровадження

Сучасні автоматизовані системи, як-от PMS, RMS або аналітичні інструменти, потребують значних фінансових інвестицій. Малі та середні готелі можуть не мати достатніх коштів для впровадження нових технологій.

Довгострокова окупність

Результати автоматизації (наприклад, збільшення прибутку чи скорочення витрат) можуть бути помітні лише через кілька років, що знижує мотивацію для її впровадження.

3. Людський фактор

Опір персоналу

Співробітники готелю можуть боятися змін через можливу втрату роботи або складність у засвоєнні нових технологій.

Недостатня кваліфікація

Для ефективного використання автоматизованих систем потрібні спеціальні знання й навички. У багатьох випадках персонал не має необхідної підготовки.

4. Організаційні проблеми

Складність управління змінами

Автоматизація вимагає перебудови існуючих бізнес-процесів і зміни корпоративної культури, що може спричинити додаткові витрати часу і ресурсів.

Недостатнє розуміння потреб

Часто готелі впроваджують технології без чіткого розуміння власних потреб, що призводить до вибору невідповідних рішень.

5. Виклики в аналітиці та прогнозуванні

Якість даних

Автоматизовані системи залежать від точності й повноти даних. Помилки в даних або їх відсутність можуть викликати помилки в прогнозах і управлінських рішеннях.

Складність аналітичних алгоритмів

Деякі методи, наприклад, ARIMA чи машинне навчання, потребують високого рівня технічних знань і ресурсів для обчислення.

6. Зміни у споживацькій поведінці

Динамічність ринку

Швидкі зміни у попиті, спричинені зовнішніми факторами (економічна ситуація, пандемії), ускладнюють прогнозування та управління ресурсами.

Високі очікування клієнтів

Клієнти очікують швидкого та персоналізованого обслуговування. Впровадження технологій повинно враховувати ці очікування.

7. Правові та етичні проблеми

Регуляторні обмеження

Законодавство щодо захисту даних, наприклад, GDPR у Європі, вимагає від готелів дотримання жорстких стандартів у роботі з персональними даними.

Етичні аспекти

Використання алгоритмів прогнозування та аналітики може викликати питання щодо приватності клієнтів і прозорості прийняття рішень [14].

1.6 Методологія розробки програмного забезпечення для готельного комплексу

Сучасна розробка програмного забезпечення (ПЗ) для автоматизації готельних комплексів базується на використанні передових методологій, таких як Agile, DevOps, Scrum, та інших. Ці підходи дозволяють створювати гнучкі, масштабовані, та ефективні рішення, які відповідають вимогам готельного бізнесу.

Agile (гнучка розробка)

Agile – це підхід, який передбачає ітеративну розробку ПЗ, де основний акцент робиться на швидку адаптацію до змін вимог і тісну співпрацю з замовником. Переваги для автоматизації готельного комплексу:

1. Гнучкість: дозволяє враховувати нові функціональні вимоги під час розробки, такі як зміни в процесах бронювання чи управління запасами.
2. Ітеративний підхід: розробка проходить через невеликі етапи (спринти), що дозволяє регулярно перевіряти робочі версії продукту.
3. Постійний зворотний зв’язок: співпраця з користувачами (адміністрація готелю, персонал) забезпечує розуміння ключових потреб.

DevOps

DevOps – це методологія, яка інтегрує розробку ПЗ (Development) та операційні процеси (Operations). Вона спрямована на автоматизацію та покращення доставки програмного забезпечення.

Особливості використання в готельному бізнесі:

1. Автоматизація розгортання: дає змогу швидко впроваджувати оновлення, наприклад, для інтеграції нових модулів бронювання.

2. Контроль якості: постійна інтеграція (Continuous Integration, CI) та постійна доставка (Continuous Delivery, CD) забезпечують стабільність і якість системи.

3. Масштабованість: DevOps дозволяє масштабувати систему відповідно до потреб готельного комплексу, наприклад, додавати нові точки доступу для персоналу.

Scrum

Scrum – це популярна реалізація Agile, яка використовує спринти (короткі цикли розробки) та фокус на командну роботу.

Чому Scrum підходить для автоматизації готельних систем?

1. Швидке створення MVP (Minimum Viable Product): створення базового продукту, який можна протестувати в реальних умовах готелю.

2. Роль Product Owner: менеджер готелю або замовник проєкту може бути Product Owner, визначаючи пріоритети завдань.

3. Постійне вдосконалення: кожен спринт дозволяє додавати новий функціонал, наприклад, інтеграцію систем лояльності чи аналітики [15].

Таблиця 1.1

Порівняння методологій

Методологія	**Особливості**	**Переваги для автоматизації готелів**
Agile	Гнучкість, ітеративний підхід	Адаптація до змін вимог, залучення користувачів
DevOps	Автоматизація розгортання, CI/CD	Забезпечення стабільності, масштабованість
Scrum	Командна робота, короткі цикли	Швидке створення MVP, чітке планування

1.7 Висновки до розділу

У цьому розділі було проведено детальний аналіз предметної області бізнес-процесів готельного комплексу. Описано функціонування готелю як об'єкта автоматизації, визначено ключові бізнес-процеси, зокрема бронювання, обслуговування клієнтів та управління ресурсами. Проведено огляд сучасних підходів до автоматизації готельного бізнесу, а також розглянуто інформаційні системи й

технології, що використовуються в галузі. Було проаналізовано основні проблеми та виклики автоматизації, включаючи технічні, організаційні та економічні аспекти, які впливають на впровадження сучасних систем. Окремо визначено методологічні підходи до розробки програмного забезпечення для автоматизації, що стануть основою для подальшого проектування ефективної інформаційної системи готельного комплексу.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛЕЙ І МЕТОДІВ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ

2.1 Загальна інформація про десктопні програми

Десктопні програми є програмами, які встановлюються на комп'ютер користувача і працюють без підключення до Інтернету. Вони надають ряд переваг для бізнесу, забезпечуючи широкий функціонал, безпеку, мобільність, швидкодію та інші можливості.

Основні переваги десктопних програм включають:

– Функціональність: Десктопні програми можуть бути розроблені з урахуванням унікальних потреб вашого бізнесу, надаючи розширені функціональні можливості. Ви можете реалізувати практично будь-яку ідею і створити програму, яка повністю задовольняє ваші бізнес-потреби.

– Безпека та надійність: Десктопні програми надають вищий рівень безпеки і захисту даних, оскільки вони працюють локально на комп'ютері користувача. Це дозволяє уникнути потенційних загроз безпеки, пов'язаних з підключенням до Інтернету, і зберегти ваші дані в безпеці.

– Мобільність: Десктопні програми можуть бути розроблені для різних операційних систем і навіть використовуватись на мобільних пристроях за допомогою нативних версій. Це дозволяє вам мати доступ до програми та керувати бізнес-процесами незалежно від місця знаходження.

– Швидкодія: Десктопні програми працюють на комп'ютері, використовуючи його ресурси, що забезпечує високу швидкодію виконання завдань. З правильною оптимізацією, такі програми можуть бути дуже продуктивними і ефективно виконувати роботу.

– Інтерфейс користувача: Десктопні програми надають можливість налаштування інтерфейсу користувача з урахуванням потреб користувачів. Ви можете створити зручний і інтуїтивно зрозумілий інтерфейс, який підходить для бізнесу.

– Використання периферійних пристроїв: Десктопні програми мають доступ до всіх пристроїв, підключених до комп'ютера, таких як принтери, сканери, фіскальні апарати тощо. Це дозволяє легко інтегрувати їх з програмою і використовувати в потрібних бізнес-процесах.

Однією з переваг десктопних програм є їхня висока продуктивність. Оскільки вони встановлюються на комп'ютер користувача і працюють локально, вони можуть ефективно використовувати ресурси системи. Це дозволяє запускати великі обчислювальні завдання, обробляти великі обсяги даних і швидко відгукуватись на дії користувача.

Десктопні програми також забезпечують більший контроль над програмою та даними. Оскільки вони працюють локально, ви маєте повний доступ до своїх даних і можете керувати ними безпосередньо. Це дозволяє зберігати конфіденційну інформацію на власному комп'ютері і уникнути потенційних загроз безпеки, пов'язаних з передачею даних через Інтернет.

Іншою перевагою десктопних програм є їх мобільність. Завдяки розробці нативних версій для різних операційних систем, десктопні програми можуть використовуватись на різних пристроях, включаючи персональні комп'ютери, ноутбуки, планшети та смартфони. Це дає змогу мати доступ до програми та керувати бізнес-процесами незалежно від місця знаходження.

Десктопні програми забезпечують більш гнучку настройку інтерфейсу користувача. Ви можете створити інтуїтивно зрозумілий інтерфейс, який відповідає потребам вашого бізнесу і забезпечує зручну роботу користувачів. Можна використовувати різні елементи керування, налаштовувати вигляд і розташування елементів, щоб забезпечити оптимальний досвід користувача.

Необхідно також згадати про можливість інтеграції десктопних програм з периферійними пристроями. Вони мають доступ до всіх пристроїв, підключених до комп'ютера, таких як принтери, сканери, фіскальні апарати тощо. Це дозволяє легко інтегрувати їх з програмою і використовувати в потрібних бізнес-процесах. Наприклад, ви можете друкувати документи безпосередньо з програми або сканувати документи для подальшої обробки.

Враховуючи всі переваги десктопних програм, важливо зазначити, що вони також мають деякі недоліки. Одним з них є потреба у встановленні програмного забезпечення на кожен окремий комп'ютер, де вони будуть використовуватись. Це може бути часомоємним і складним процесом, особливо якщо необхідно встановити програму на велику кількість комп'ютерів.

Також слід враховувати, що десктопні програми можуть бути обмежені сумісністю з різними операційними системами. Якщо ви розробляєте програму для Windows, вона може не працювати на комп'ютерах з операційною системою macOS або Linux без додаткових зусиль для портування програми.

Незважаючи на ці недоліки, десктопні програми залишаються потужним інструментом для бізнесу. Вони надають широкий функціонал, безпеку, мобільність, швидкодію та можливість інтеграції з периферійними пристроями. Правильно розроблена десктопна програма може підвищити ефективність роботи вашої компанії та сприяти досягненню бізнес-цілей.

Отже розробка десктопних програм дозволяє вам впроваджувати цифрові продукти у вашу компанію, що сприяє підвищенню ефективності, автоматизації процесів та оптимізації роботи. Завдяки можливостям десктопних програм, ви можете мати повний контроль над програмою, надійну безпеку даних та забезпечити високу продуктивність [16].

2.2 Дослідження математичної моделі для управління бізнес-процесами

У цьому підрозділі було запропоновано математичну модель, яка враховує специфіку роботи готельного комплексу, та спрямована на підвищення ефективності управління бізнес-процесами.

Основні компоненти моделі:

1. Динамічний розподіл номерів залежно від запиту клієнтів

Враховано такі параметри:

- Кількість доступних номерів кожного типу.
- Час перебування клієнта в готелі.

- Особливі побажання клієнтів (тип номера, вид із вікна тощо).

Модель формалізується як задача оптимального розподілу ресурсів, де:

$$max \sum_{i=n}^{n} P_i x_i, \quad (2.1)$$

де P_i — прибуток від розміщення клієнта у номері і, x_i — бінарна змінна, що відображає факт бронювання номера. Обмеження забезпечують урахування доступності номерів та конфліктів у бронюванні.

2. Оптимізація робочого часу персоналу

Модель спрямована на розподіл завдань серед персоналу з урахуванням:

- Робочого графіка співробітників.
- Навантаження на кожного працівника.
- Пріоритетності завдань (прибирання, обслуговування клієнтів, ремонт тощо).

Оптимізація здійснюється за допомогою задачі лінійного програмування:

$$min \sum_{j=1}^{m} C_j y_j, \quad (2.2)$$

де C_j — час виконання завдання j, y_j — змінна, що показує виконання завдання працівником.

3. Прогнозування на основі часових рядів (ARIMA)

Для прогнозування пікових періодів використано модель ARIMA (AutoRegressive Integrated Moving Average).
Вхідні дані: історичні дані про заповненість номерів, сезонні фактори, святкові періоди.
Формула ARIMA:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t, \quad (2.3)$$

Де:

y_t — прогнозоване значення.

c — константа.

ϕ, θ — параметри моделі.

ϵ_t — випадкова помилка.

Це дозволяє точно визначити періоди високого та низького попиту, щоб ефективно планувати ресурси.

Переваги запропонованої моделі:

- Інтеграція кількох компонентів дозволяє створити універсальне рішення для управління бізнес-процесами.
- Гнучкість моделі дає можливість адаптувати її під конкретні умови роботи готелю.
- Використання прогнозування дозволяє зменшити втрати через нераціональне використання ресурсів.

Ця модель є основою для подальшої реалізації алгоритмів автоматизації в інформаційній системі [17].

2.3 Технічні вимоги до апаратного забезпечення, на якому буде працювати система

Для забезпечення стабільної роботи системи автоматизації готельного комплексу, розробленої у вигляді десктопної програми, необхідно врахувати мінімальні та рекомендовані вимоги до апаратного забезпечення. Ці вимоги залежать від обсягу даних, функціональних можливостей програми та архітектури системи.

Мінімальні вимоги:

1. Процесор (CPU):

Двоядерний процесор із тактовою частотою не менше ніж 2 ГГц.

2. Оперативна пам'ять (RAM):

4 ГБ (для базових функцій і невеликої кількості даних).

3. Жорсткий диск (HDD/SSD):

Вільне місце: 10 ГБ (включаючи місце для операційної системи, програми та бази даних).

4. Графічна підсистема:

Інтегрована відеокарта з підтримкою роздільної здатності 1024x768.

5. Операційна система (OS):

Windows 10 (64-bit), macOS 10.15 або Linux (Ubuntu 20.04 і вище).

6. Підключення до мережі:

Ethernet або Wi-Fi для роботи з хмарними сервісами або базами даних.

Рекомендовані вимоги:

1. Процесор (CPU):

Чотириядерний процесор із тактовою частотою 3 ГГц і підтримкою багатопоточності (наприклад, Intel Core i5 або AMD Ryzen 5).

2. Оперативна пам'ять (RAM):

8-16 ГБ (для одночасної роботи з великими обсягами даних і запуску кількох процесів).

3. Жорсткий диск (HDD/SSD):

SSD на 256 ГБ і більше для забезпечення швидкості роботи системи.

4. Графічна підсистема:

Дискретна відеокарта з підтримкою DirectX 12 або OpenGL 4.5 (наприклад, NVIDIA GeForce GTX 1050).

5. Операційна система (OS):

Windows 11 Pro (64-bit), macOS Ventura або Linux (RHEL 9).

6. Підключення до мережі:

Гігабітний Ethernet для швидкого обміну даними або стабільне Wi-Fi з’єднання.

Додаткові вимоги (залежно від особливостей програми):

1. Сервер баз даних (MySQL):

Для локальної роботи: сервер з мінімум 8 ГБ оперативної пам'яті та SSD-диском.

Для хмарної роботи: підключення до зовнішнього сервера або хмарних платформ (AWS RDS, Google Cloud SQL).

2. Периферійні пристрої:

Принтер для друку чеків чи звітів.

3. Резервне копіювання:

Зовнішній жорсткий диск або хмарне сховище для збереження резервних копій баз даних.

Врахування масштабування

Для систем із більшим навантаженням (наприклад, мережа готелів) рекомендується використання серверного обладнання з вищими характеристиками: багатоядерні сервери, RAID-масиви для збереження даних, апаратні брандмауери для безпеки [18].

2.4 Опис окремих компонентів, методів та інтерфейсів системи

У системі автоматизації готельного комплексу для адміністратора доступні різні компоненти, методи та інтерфейси, що допомагають зручно та ефективно управляти всіма аспектами готельного бізнесу.

Централізована система управління: Центральний сервер та керуюче програмне забезпечення є основною складовою системи автоматизації готельного комплексу для адміністратора. Цей сервер забезпечує централізоване управління всіма компонентами системи, а керуюче програмне забезпечення надає адміністраторам можливість налаштовувати та керувати різними аспектами готельного бізнесу, включаючи номери, бронювання, послуги та персонал. Завдяки централізованій системі управління, адміністраторам та менеджерам легко керувати різними процесами та отримувати необхідну інформацію для ефективної роботи.

Управління гостьовими номерами: Для адміністратора готельного комплексу важливо мати повний контроль над гостьовими номерами. Система автоматизації надає модуль управління номерами, який дозволяє адміністраторам призначати номери гостям, встановлювати ціни, переглядати наявність вільних номерів та здійснювати необхідні зміни в інформації про номери. Інтерфейс модуля управління номерами є зручним та інтуїтивно зрозумілим, що дозволяє адміністраторам легко виконувати рутинні операції та контролювати стан номерів.

Управління бронюваннями: Адміністратор готельного комплексу має можливість керувати процесом бронювання через спеціальний модуль управління бронюваннями. Цей модуль дозволяє адміністраторам переглядати та змінювати інформацію про бронювання, створювати нові бронювання, приймати запити на бронювання та робити необхідні підтвердження. Крім того, система автоматично оновлює календар зайнятості номерів, отримує сповіщення про нові бронювання та забезпечує можливість виконати необхідні дії з бронюваннями.

Управління послугами: Готельний комплекс може пропонувати різноманітні послуги для своїх гостей, такі як ресторани, спа-центри, басейни тощо. Адміністратор системи автоматизації має можливість управляти цими послугами через відповідний модуль управління послугами. Він може налаштовувати ціни на послуги, встановлювати графік роботи, призначати персонал для надання послуг та здійснювати моніторинг популярності послуг серед гостей. Інтерфейс модуля управління послугами дозволяє адміністраторам зручно керувати всіма аспектами послуг готельного комплексу.

Ці компоненти та їх інтерфейси допомагають адміністраторам готельного комплексу ефективно управляти готелем, керувати номерами, бронюванями, послугами. Вони надають зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє адміністраторам легко здійснювати всі необхідні операції та контролювати стан готельного бізнесу. Завдяки автоматизації, адміністраторам випадає менше рутинної роботи, а їхні дії стають більш організованими та продуктивними [19].

На (рис. 2.1) зображено зв'язок компонента і інтервейсів.

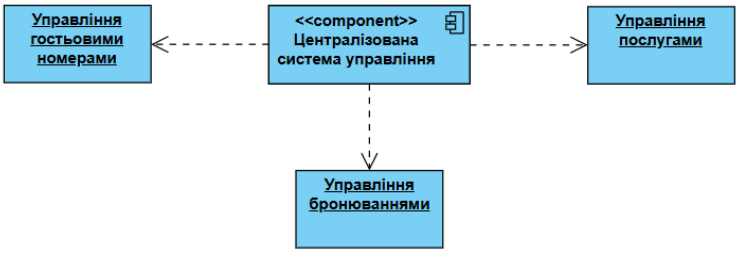

Рис. 2.1. Компоненти та інтерфейси програми

2.5 Представлення алгоритмів автоматизації

Автоматизація бізнес-процесів готельного комплексу неможлива без розробки ефективних алгоритмів, які забезпечують інтеграцію, оптимізацію та автоматизацію ключових функцій. Ці алгоритми спрямовані на вирішення завдань управління ресурсами, покращення взаємодії з клієнтами, планування роботи персоналу та підвищення загальної ефективності роботи готелю.

1. Алгоритми для оптимізації бронювання

Система бронювання є серцем готельного бізнесу, адже саме вона забезпечує швидкий і безпомилковий облік номерного фонду. Основні аспекти:

- Пошук доступних номерів: реалізовано алгоритм, який аналізує доступність номерів у реальному часі, враховуючи такі параметри, як тип номера, дата заїзду та виїзду, особливі запити клієнтів (наприклад, додаткове ліжко чи вид на море).
- Динамічне ціноутворення: використовуючи методи прогнозування (наприклад, ARIMA), алгоритм дозволяє коригувати ціни залежно від попиту, сезону та інших факторів. Це сприяє максимізації доходів і ефективному використанню номерного фонду.
- Скасування та зміни бронювань: алгоритми забезпечують автоматичне оновлення статусу номерів після змін, внесених клієнтом, мінімізуючи вплив людського фактора.

2. Алгоритми для управління персоналом

Ефективне управління персоналом є важливим фактором у підвищенні якості послуг і зменшенні витрат. Основні алгоритми включають:

- Планування змін: алгоритм автоматично розподіляє робочі години персоналу, враховуючи такі параметри, як рівень завантаженості готелю, кількість гостей, кваліфікація співробітників і особливості їхніх змін.
- Розподіл завдань: розроблено механізм, що дозволяє оперативно розподіляти завдання серед персоналу залежно від їхньої ролі. Наприклад, прибирання номерів, обслуговування клієнтів у ресторані чи організація додаткових послуг.

- Моніторинг продуктивності: алгоритм відстежує ефективність роботи кожного співробітника, що дозволяє вчасно виявляти проблемні зони та коригувати процеси.

3. Алгоритми управління ресурсами

Оптимальне використання ресурсів — одна з основних умов для ефективного функціонування готелю. Розроблені алгоритми включають:

- Управління матеріальними ресурсами: створено систему, яка автоматично відслідковує запаси (наприклад, постільної білизни, миючих засобів) та генерує замовлення на поповнення, якщо кількість ресурсів досягає критичного мінімуму.
- Розподіл номерного фонду: алгоритм оптимізує використання номерів, пропонуючи найбільш доцільні варіанти розміщення гостей, залежно від їхніх запитів і доступності ресурсів.
- Планування інвентаризації: автоматизація цього процесу дозволяє своєчасно оцінювати стан обладнання та необхідність його заміни.

4. Алгоритми прогнозування попиту

Для підготовки до пікових періодів і уникнення недовантаження в низький сезон використовуються алгоритми прогнозування:

- Прогнозування рівня завантаженості: застосування статистичних моделей (ARIMA, Holt-Winters) дозволяє визначити очікуваний попит на номери у конкретні періоди.
- Сезонний аналіз: аналіз історичних даних дає змогу виявити закономірності та адаптувати пропозицію під очікувані зміни в поведінці клієнтів.

5. Алгоритми інтеграції та обробки даних

- Інтеграція даних: алгоритми забезпечують взаємодію з іншими системами (CRM, ERP), дозволяючи обмінюватися даними про клієнтів, бронювання та фінансові операції.
- Обробка винятків: розроблено алгоритми для роботи з нестандартними ситуаціями, такими як помилкові дані, раптові зміни в бронюваннях чи конфлікти в ресурсах.

Результати та переваги

Застосування цих алгоритмів дозволяє:

1. Автоматизувати рутинні операції, зменшуючи навантаження на персонал.
2. Оптимізувати використання ресурсів, мінімізуючи витрати.
3. Підвищити рівень обслуговування клієнтів завдяки швидкому реагуванню на їхні запити.
4. Зменшити кількість помилок, пов’язаних із людським фактором [19].

2.6 Побудова ER-діаграми бази даних

Моделювання предметної області в моделі «сутність - зв'язок» (ER - entity - relationship) базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів. Основними поняттями ER-моделі є сутність, зв'язок і атрибут.

Сутністю називається реальний або представлений об'єкт, інформація про який повинна зберігатися в базі даних [20-22].

При аналізі предметної області і розробленні ER–діаграми бази даних використано типи зв’язку «багато–до–одного» між таблицями.

Взаємозв'язки між сутностями бази даних готельного комплексу можна описати наступними словесними зв'язками:

1. client ↔ registrationclient
   - Зв’язок: 1 до багатьох
   - Один клієнт може мати багато реєстрацій (наприклад, відвідування в різний час).
   - Зовнішній ключ: ClientId в таблиці registrationclient посилається на ClientId в таблиці client.
2. employee ↔ registrationclient
   - Зв’язок: 1 до багатьох
   - Один співробітник може обслуговувати багато реєстрацій.
   - Зовнішній ключ: EmployeeId в таблиці registrationclient посилається на EmployeeId в таблиці employee.

3. room ↔ registrationclient

- Зв’язок: 1 до багатьох
- Один номер може бути зарезервований у різний час для клієнтів.
- Зовнішній ключ: RoomId в таблиці registrationclient посилається на RoomId в таблиці room.

4. client ↔ serviceprovided

- Зв’язок: 1 до багатьох
- Один клієнт може отримувати багато послуг.
- Зовнішній ключ: ClientId в таблиці serviceprovided посилається на ClientId в таблиці client.

5. service ↔ serviceprovided

- Зв’язок: 1 до багатьох
- Одна послуга може надаватися кілька разів для різних клієнтів.
- Зовнішній ключ: ServiceId в таблиці serviceprovided посилається на ServiceId в таблиці service.

6. employee ↔ tasks

- Зв’язок: 1 до багатьох
- Один співробітник може виконувати багато завдань.
- Зовнішній ключ: EmployeeId в таблиці tasks посилається на EmployeeId в таблиці employee.

7. users ↔ logs

- Зв’язок: 1 до багатьох
- Один користувач може генерувати багато логів.
- Зовнішній ключ: UsersId в таблиці logs посилається на UsersId в таблиці users.

ER-діаграму бази даних представлено на (рис. 2.2)

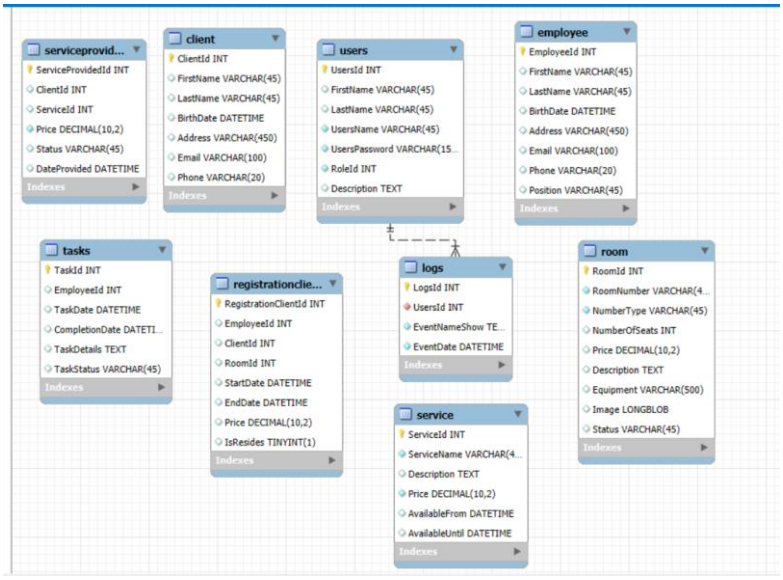

Рис. 2.2. ER-діаграма БД

2.7 Побудова моделі бізнес-процесів готельного комплексу (UML)

UML, скорочено для Unified Modeling Language, - це стандартизована мова моделювання, що складається з інтегрованого набору діаграм, розроблених для допомоги розробникам системи та програмного забезпечення для конкретизації, візуалізації, побудови та документування артефактів програмних систем, а також для бізнес-моделювання та інших непрограмних систем. UML являє собою сукупність найкращих інженерних практик, які виявилися успішними в моделюванні великих і складних систем. UML є дуже важливою частиною розроблення об'єктно-орієнтованого програмного забезпечення та процесу розроблення програмного забезпечення. UML використовує переважно графічні познаки для вираження дизайну програмних проєктів. Використання UML допомагає командам проектів спілкуватися, досліджувати потенційні проєкти та підтверджувати архітектурний дизайн програмного забезпечення [40-42].

2.7.1 Побудова діаграми варіантів використання

Діаграми варіантів використання (прецедентів) застосовуються для моделювання виду системи з точки зору зовнішнього спостерігача. На діаграмі

прецедентів графічно показана сукупність прецедентів та суб'єктів, а також відносини між ними.

Актор "Адміністратор" представляє особу, яка має повний доступ та контроль над системою автоматизації готельного комплексу. Він відповідає за управління та надання послуг для клієнтів, а також за підтримку та адміністрування системи. Описуємо окремі прецеденти та їх опис для актора "Адміністратор":

1. Управління клієнтами:

– Додати клієнта: Адміністратор має можливість додавати нових клієнтів, вводячи їх особисті дані, такі як ім'я, прізвище, контактний телефон, електронна пошта, а також будь-які інші важливі відомості.

– Редагувати клієнта: Інформація про існуючих клієнтів може бути змінена, включаючи оновлення контактних даних, особистих переваг або приміток про попередні візити.

– Видалити клієнта: У випадках, коли клієнт більше не користується послугами або його дані стають застарілими, адміністратор може видалити його профіль із системи.

2. Управління бронюваннями:

– Додати бронювання: Адміністратор створює нове бронювання, вказуючи дати заїзду та виїзду, обраний тип номера, спеціальні запити клієнта, а також статус платежу.

– Редагувати бронювання: Можливість змінювати деталі існуючих бронювань, наприклад, дати перебування, номер кімнати або додаткові послуги.

– Скасувати бронювання: У разі потреби адміністратор може видалити бронювання, оновлюючи статус доступності номеру.

3. Управління номерами:

– Додати новий номер: Адміністратор може додати до системи нові кімнати, вказуючи їх тип (стандарт, люкс тощо), кількість спальних місць, ціну та опис обладнання.

– Редагувати інформацію про номер: Можливість оновлювати дані про номери, зокрема їх вартість, опис або фотографії.

4. Облік персоналу:

– Реєстрація співробітників: Адміністратор додає профілі співробітників із зазначенням їхніх посад, робочих графіків і контактних даних.

– Контроль виконання задач: Відстеження завершення призначених завдань та оновлення їх статусу.

5. Фінансовий облік:

– Генерація рахунків: Система автоматично створює рахунки для клієнтів, враховуючи оплату номерів і додаткових послуг.

– Облік платежів: Ведення реєстру сплачених, скасованих або заборгованих платежів.

– Фінансова звітність: Автоматичне формування звітів про доходи та витрати за вибраний період.

6. Обслуговування гостей:

– Реєстрація запитів: Гості можуть замовляти додаткові послуги, такі як трансфер, харчування або організація екскурсій.

– Контроль виконання запитів: Адміністратор відстежує статус виконання кожного запиту, оновлюючи його в системі.

7. Аналітика та звітність:

– Аналіз заповнюваності: Система генерує звіти про завантаженість номерів за періоди.

– Прогнозування: Алгоритми аналізують історичні дані, прогнозуючи попит на номери або послуги.

– Популярність послуг: Генерація звітів про найбільш затребувані послуги для подальшого покращення роботи комплексу.

Діаграму варіантів використання наведено на (рис. 2.3)

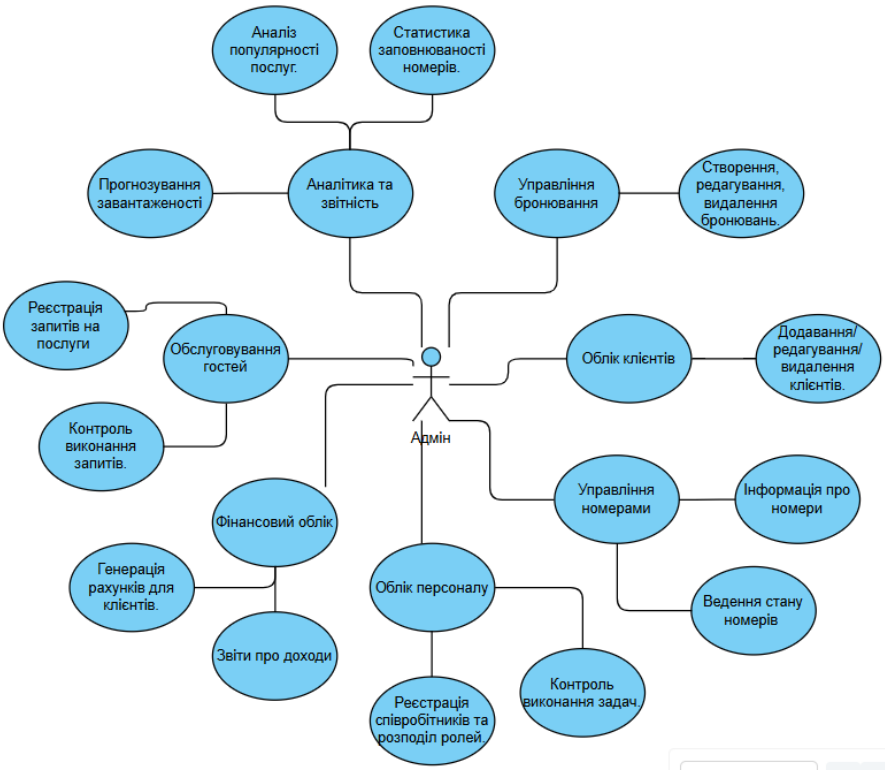

Рис. 2.3. Діаграма прецендентів

2.8 Вибір методів для автоматизації процесів

Автоматизація бізнес-процесів готельного комплексу вимагає застосування сучасних методів і підходів, які дозволяють ефективно вирішувати завдання планування, обробки даних і управління ресурсами. Вибір методів базується на аналізі потреб готелю, доступних технологій і алгоритмічних рішень, спрямованих на підвищення продуктивності, мінімізацію витрат і покращення якості обслуговування клієнтів.

1. Методи для планування

Планування є основою для координації дій персоналу, управління номерним фондом та оптимізації використання ресурсів. Основні методи:

- Лінійне програмування: застосовується для розв’язання задач оптимального розподілу номерів і робочих змін персоналу. Наприклад, метод дозволяє мінімізувати витрати часу та коштів, забезпечуючи рівномірне завантаження персоналу.

- Методи календарного планування: забезпечують автоматизацію розкладу роботи співробітників і управління бронюваннями. Ці методи дозволяють враховувати часові обмеження, сезонні коливання попиту та спеціальні запити клієнтів.
- Пріоритетне планування: алгоритми визначають черговість обробки задач на основі їхньої важливості та строків виконання, що дозволяє оптимізувати роботу з великою кількістю замовлень.

2. Методи для обробки даних

Сучасні готелі працюють із великим обсягом інформації: дані про клієнтів, бронювання, фінансові операції тощо. Обробка цих даних є критично важливою для прийняття обґрунтованих рішень. Методи, що використовуються:

- Аналіз часових рядів: застосовується для прогнозування попиту та завантаженості номерів на основі історичних даних. Найбільш популярними є моделі ARIMA та Holt-Winters, які дозволяють виявляти тренди та сезонні коливання.
- Методи кластеризації: використовуються для сегментації клієнтів за параметрами, такими як частота бронювань, середній чек та особливі вимоги. Це дозволяє налаштувати послуги під конкретні групи клієнтів.
- Методи обробки великих даних (Big Data): у великих готельних мережах можуть застосовуватися методи для аналізу значних обсягів інформації в реальному часі. Наприклад, технології Hadoop або Spark для виявлення закономірностей і прийняття рішень.

3. Методи для розподілу ресурсів

Оптимальне використання номерного фонду, персоналу та матеріальних ресурсів є ключовим фактором для ефективного управління готельним комплексом. Методи, що застосовуються:

- Жадібні алгоритми: використовуються для розподілу номерів між клієнтами з урахуванням їхніх запитів, типів номерів та наявності спеціальних пропозицій.
- Методи динамічного програмування: дозволяють розв’язувати складні задачі розподілу, враховуючи взаємозв’язки між ресурсами, наприклад, розподіл номерів із урахуванням їхньої доступності та доходності.

- Алгоритми Маркова: застосовуються для моделювання процесів обслуговування клієнтів і оптимізації використання ресурсів у довгостроковій перспективі.

4. Методи для управління бізнес-процесами

Для інтеграції всіх елементів готельного комплексу в єдину автоматизовану систему використовуються наступні методи:

- Моделювання бізнес-процесів (BPMN): дає змогу формалізувати процеси готельного комплексу, ідентифікувати їхні вузькі місця та оптимізувати робочі потоки.
- Методи машинного навчання: використовуються для аналізу поведінки клієнтів, прогнозування попиту та налаштування персоналізованих пропозицій.
- Методи багатокритеріального аналізу: дозволяють оцінити ефективність різних варіантів управління, враховуючи кілька критеріїв, наприклад, витрати, завантаженість та рівень задоволеності клієнтів.

5. Результати та переваги вибору методів

Застосування цих методів для автоматизації процесів забезпечує:

1. Оптимізацію використання ресурсів, мінімізуючи витрати та підвищуючи прибутковість.
2. Підвищення рівня обслуговування клієнтів завдяки персоналізації пропозицій і швидкому реагуванню на їхні потреби.
3. Зменшення впливу людського фактора на прийняття рішень та обробку даних.
4. Забезпечення прозорості та ефективності бізнес-процесів у готельному комплексі.
5. Масштабованість системи для інтеграції нових технологій і розширення функціоналу.

Таким чином, обрані методи автоматизації створюють основу для впровадження сучасної інформаційної системи, яка дозволяє ефективно керувати всіма аспектами діяльності готельного комплексу.

2.9 Висновки до розділу

У другому розділі роботи проведено детальний аналіз та розробку моделей і методів, які забезпечують автоматизацію бізнес-процесів готельного комплексу. Було визначено ключові аспекти побудови ефективної системи управління, що включає як математичні моделі, так і алгоритми оптимізації для автоматизації діяльності готелю.

Зокрема, було описано загальні принципи функціонування десктопних програм, визначено технічні вимоги до апаратного забезпечення та сформовано логічну архітектуру системи. Розробка математичної моделі дозволила врахувати такі важливі аспекти, як динамічний розподіл номерного фонду, оптимізація робочого часу персоналу та прогнозування попиту з використанням методів аналізу часових рядів.

Особливу увагу приділено алгоритмам автоматизації, що забезпечують оптимізацію бронювання, управління ресурсами та персоналом, а також зменшення впливу людського фактора на бізнес-процеси. Було побудовано ER-діаграму для моделювання бази даних, яка забезпечує ефективне зберігання й обробку даних, необхідних для роботи системи.

Побудова UML-діаграм та моделювання бізнес-процесів дозволили структурувати й формалізувати взаємодію між компонентами системи, що є основою для її подальшої реалізації. Вибір сучасних методів автоматизації, таких як алгоритми планування, обробки даних і управління ресурсами, забезпечує масштабованість і функціональність розробленої системи.

Таким чином, результати, отримані в цьому розділі, формують фундамент для практичної реалізації програмного продукту, спрямованого на оптимізацію бізнес-процесів готельного комплексу та підвищення його конкурентоспроможності на ринку послуг.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ

Система автоматизації готельного комплексу надає адміністратору можливість ефективного керування клієнтами та всіма аспектами бронювання, послуг і скарг. Адміністратор може додавати, редагувати та видаляти інформацію про клієнтів, зокрема особисті дані, контактну інформацію тощо. Крім того, адміністратор може створювати та змінювати бронювання клієнтів, вказуючи дати заселення та виселення, номери кімнати та іншу важливу інформацію.

Система також надає можливість керувати послугами та стравами, що надаються готельним комплексом. Адміністратор може надавати нові послуги та страви, вказуючи їх назви, описи та ціни. Він також може редагувати існуючі послуги та страви, змінюючи їх описи, ціни або назви за обслуговуванням.

Управління скаргами також є важливою функцією системи. Адміністратор може переглядати всі скарги, які надійшли від клієнтів, і приймати запити на їх вирішення. Він також може відповідати на скарги клієнтів, надавати пояснення або пропонувати розв'язання проблеми.

Загалом, система надає адміністратору повний контроль над усіма аспектами контролю клієнтів, бронюванням, послугами та скаргами. Вона спрощує рутинні операції та полегшує процес прийняття рішень, що дозволяють підвищити ефективність та якість обслуговування клієнтів у готельному комплексі.

3.1 Вимоги до інформаційної системи для готельного комплексу

Для успішної реалізації інформаційної системи, що автоматизує бізнес-процеси готельного комплексу, важливо визначити функціональні та

нефункціональні вимоги до системи:

1. Функціональні вимоги:

– Управління процесами бронювання: створення, редагування, скасування та перегляд бронювань.

– Ведення бази даних клієнтів із доступом до історії бронювань та інших записів.

– Управління номерним фондом, включаючи актуальну інформацію про доступність номерів.

– Оптимізація розподілу ресурсів: робочих змін персоналу, матеріальних і фінансових ресурсів.

– Інтеграція з аналітичними модулями для створення звітів і прогнозів попиту.

2. Нефункціональні вимоги:

– Продуктивність: система повинна забезпечувати швидкий відгук на дії користувачів навіть при великій кількості даних.

– Надійність: безперервна робота без збоїв та втрачених даних.

– Масштабованість: можливість додавання нових модулів або інтеграції зі сторонніми сервісами.

– Безпека: захист персональних даних клієнтів та запобігання несанкціонованому доступу.

– Дружній інтерфейс: зрозумілий та інтуїтивний дизайн, який спрощує взаємодію користувача із системою.

3.2 Архітектура та компоненти інформаційної системи

1. Архітектура системи:

– Клієнт-серверна модель:

Клієнтська частина забезпечує доступ користувача до функціоналу через зручний інтерфейс, тоді як серверна частина обробляє бізнес-логіку та взаємодіє з базою даних [44].

– Модульна структура: система складається з незалежних модулів, таких як модуль бронювання, управління клієнтами, управління ресурсами, фінансовий та аналітичний модулі.

2. Основні компоненти:

– База даних: MySQL, яка зберігає всю інформацію про клієнтів, бронювання, фінансові операції та звіти.

– Клієнтський інтерфейс: побудований з використанням Windows Forms (C#), який забезпечує доступ до функціоналу для різних користувачів (адміністратор, менеджер, клієнт).

– Серверний модуль: реалізує бізнес-логіку, взаємодію з базою даних та обробку запитів клієнтів.

3. Схема взаємодії:

– Користувачі через інтерфейс клієнтської частини взаємодіють із системою.

– Сервер обробляє запити клієнтів, перевіряє дані у базі даних та виконує відповідні операції.

– Дані передаються клієнту у зручному вигляді через інтерфейс.

3.2.1. Опис інстументів та технологій

Для створення інформаційної системи обрано такі інструменти та технології:

1. Мова програмування:

C#: забезпечує високу продуктивність та гнучкість у розробці десктопних додатків.

2. СУБД:

MySQL: як основний інструмент для управління даними. Використання MySQL Workbench дозволяє створювати ER-діаграми та оптимізувати структуру бази даних.

3. Середовище розробки:

Visual Studio: потужний інструмент для написання, тестування та налагодження коду.

4. Фреймворки:

.NET Framework: для розробки програмного забезпечення з широкими можливостями інтеграції.

3.3 Створення бази даних

База даних MySQL була створена у середовищі MySQL Workbench. Це середовище забезпечує зручну роботу з базами даних, дозволяючи графічно моделювати структуру, включаючи встановлення зв’язків між таблицями, таких як «один до багатьох», за допомогою інструментів для створення ER-діаграм.

Крім того, MySQL Workbench пропонує потужний редактор SQL-запитів, який дозволяє легко виконувати запити, відправляти їх на сервер та отримувати результати у вигляді таблиць. Такі можливості роблять це середовище ефективним для проєктування, аналізу та управління базою даних.

Частина SQL коду по створення таблиць наведено в лістингу 3.1:

Лістинг 3.1.

```
-- Таблиця клієнтів
DROP TABLE IF EXISTS `client`;
CREATE TABLE `client` (
  `ClientId` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) DEFAULT NULL,
  `LastName` VARCHAR(45) DEFAULT NULL,
  `BirthDate` DATETIME DEFAULT NULL,
  `Address` VARCHAR(450) DEFAULT NULL,
  `Email` VARCHAR(100) DEFAULT NULL,
  `Phone` VARCHAR(20) DEFAULT NULL,
  PRIMARY KEY (`ClientId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця співробітників
DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `EmployeeId` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) DEFAULT NULL,
  `LastName` VARCHAR(45) DEFAULT NULL,
  `BirthDate` DATETIME DEFAULT NULL,
  `Address` VARCHAR(450) DEFAULT NULL,
  `Email` VARCHAR(100) DEFAULT NULL,
  `Phone` VARCHAR(20) DEFAULT NULL,
  `Position` VARCHAR(45) DEFAULT NULL,
```

```
  PRIMARY KEY (`EmployeeId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця логів
DROP TABLE IF EXISTS `logs`;
CREATE TABLE `logs` (
  `LogsId` INT NOT NULL AUTO_INCREMENT,
  `UsersId` INT NOT NULL,
  `EventNameShow` TEXT NOT NULL,
  `EventDate` DATETIME NOT NULL,
  PRIMARY KEY (`LogsId`),
  KEY `UsersId_fk_idx` (`UsersId`),
  CONSTRAINT `UsersId_fk` FOREIGN KEY (`UsersId`) REFERENCES `users`
(`UsersId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця реєстрації клієнтів
DROP TABLE IF EXISTS `registrationclient`;
CREATE TABLE `registrationclient` (
  `RegistrationClientId` INT NOT NULL AUTO_INCREMENT,
  `EmployeeId` INT DEFAULT NULL,
  `ClientId` INT DEFAULT NULL,
  `RoomId` INT DEFAULT NULL,
  `StartDate` DATETIME DEFAULT NULL,
  `EndDate` DATETIME DEFAULT NULL,
  `Price` DECIMAL(10,2) DEFAULT NULL,
  `IsResides` TINYINT(1) DEFAULT NULL,
  PRIMARY KEY (`RegistrationClientId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
```

Повний код у додатку А

3.4 Реалізація функціоналу

Розробка імплементації: здійснюється програмування функціоналу на основі встановлених вимог. Використовуються певні мови програмування, фреймворки та інструменти розробки для створення програмного коду.

Вданій роботі використувувалась мова C# для розробки інтерфейса.

При розробленні інтерфейсу було дотримано кількох правил:

- розуміння цілей і поведінкових звичок користувачів;
- позбавлення інтерфейсу сторонніх завдань;
- проста логіка;
- єдиний стиль.

Нижче наведено код реалізації початкової форми тобто меню програми, яка відповідає за перехід до інших форм для роботи з даними БД, а також містить код, який загружає певні форми наведено в лістингу 3.2:

Лістинг 3.2.

```
      private void персоналізаціяToolStripMenuItem_Click(object sender,
EventArgs e) {
     PersonalizationForm personalizationForm = new
PersonalizationForm(LoginForm.CurrentUser.UsersId);
     personalizationForm.MdiParent = this;
     personalizationForm.WindowState = FormWindowState.Maximized;
     personalizationForm.Show();
   }
   private void змінитиКористувачаToolStripMenuItem_Click(object
sender, EventArgs e) {
     // Закриття всіх дочірніх вікон
     CloseAllWindows();
     // Перезапуск програми
     Program.RestartApplication();
   }
   private void персоналToolStripMenuItem_Click(object sender,
EventArgs e) {
     CloseAllWindows();
     EmployeeForm employeeForm = new EmployeeForm();
     employeeForm.MdiParent = this;
     employeeForm.WindowState = FormWindowState.Maximized;
     employeeForm.Show();
   }
private void клієнтиToolStripMenuItem_Click(object sender, EventArgs e)
{
  CloseAllWindows();
  ClientForm clientForm = new ClientForm();
  clientForm.MdiParent = this;
  clientForm.WindowState = FormWindowState.Maximized;
  clientForm.Show();
}
private void номериToolStripMenuItem_Click(object sender, EventArgs e)
{
  CloseAllWindows();
  RoomForm roomForm = new RoomForm();
  roomForm.MdiParent = this;
  roomForm.WindowState = FormWindowState.Maximized;
  roomForm.Show();
}
```

У лістингу 3.3 наведено код реалізації форми для обробки і відображення даних про бронювання кімнат, яка відповідає за додавання, редагування і видалення даних про бронювання з БД:

Лістинг 3.3.

```
    public RegistrationClient
SelectedRegistrationClientByRegistrationClientId(int
RegistrationClientId) {
      string query = "SELECT * FROM RegistrationClient WHERE
RegistrationClientId = @RegistrationClientId";
      RegistrationClient oneRegistrationClient = new
RegistrationClient();
      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);
      connection.Open();
      using (MySqlDataReader reader = command.ExecuteReader()) {
        if (reader.Read()) {
          oneRegistrationClient.RegistrationClientId =
Convert.ToInt32(reader["RegistrationClientId"]);
          oneRegistrationClient.EmployeeId =
Convert.ToInt32(reader["EmployeeId"]);
          oneRegistrationClient.ClientId =
Convert.ToInt32(reader["ClientId"]);
          oneRegistrationClient.RoomId =
Convert.ToInt32(reader["RoomId"]);
          oneRegistrationClient.StartDate =
Convert.ToDateTime(reader["StartDate"]);
          oneRegistrationClient.EndDate =
Convert.ToDateTime(reader["EndDate"]);
          oneRegistrationClient.Price =
Convert.ToDouble(reader["Price"]);
          oneRegistrationClient.IsResides =
Convert.ToBoolean(reader["IsResides"]);
        }
      }
      connection.Close();
      return oneRegistrationClient;
    }
    public void UpdateRegistrationClient(int EmployeeId, int ClientId,
int RoomId, DateTime StartDate, DateTime EndDate, double Price, int
RegistrationClientId) {
      string query = "UPDATE RegistrationClient SET EmployeeId =
@EmployeeId, ClientId = @ClientId, RoomId = @RoomId, " +
                     "StartDate = @StartDate, EndDate = @EndDate, Price
= @Price WHERE RegistrationClientId = @RegistrationClientId";
      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@EmployeeId", EmployeeId);
      command.Parameters.AddWithValue("@ClientId", ClientId);
      command.Parameters.AddWithValue("@RoomId", RoomId);
      command.Parameters.AddWithValue("@StartDate",
StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
```

```
      command.Parameters.AddWithValue("@EndDate",
EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@Price", Price);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);
      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }
    public void UpdateIsResidesStatus(int RegistrationClientId) {
      string query = "UPDATE RegistrationClient SET IsResides = 1 WHERE
RegistrationClientId = @RegistrationClientId";
      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);
      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }
    public void DeleteRegistrationClientByRegistrationClientId(int
RegistrationClientId) {
      string query = "DELETE FROM RegistrationClient WHERE
RegistrationClientId = @RegistrationClientId";
      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);

      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }
  }
}
```

Повний код реалізації функціоналу наведено в додатку Б.

Метод GetRoomOccupancyForecast, який прогнозує завантаженість номерів готелю на основі історичних даних. Метод працює з базою даних через SQL-запити і використовує статистичні обчислення для формування прогнозу. У лістингу 3.4 наведено код метода:

Лістинг 3.4.

```
    /// </summary>
    /// <param name="historyStartDate">Початок історичного періоду для
аналізу.</param>
    /// <param name="historyEndDate">Кінець історичного періоду для
аналізу.</param>
```

```
    /// <param name="forecastStartDate">Початок періоду, для якого
здійснюється прогноз.</param>
    /// <param name="forecastEndDate">Кінець періоду, для якого
здійснюється прогноз.</param>
    /// <returns>Список прогнозів для кожного номера у вигляді об'єктів
RoomOccupancyForecast.</returns>
    public List<RoomOccupancyForecast>
GetRoomOccupancyForecast(DateTime historyStartDate, DateTime
historyEndDate, DateTime forecastStartDate, DateTime forecastEndDate) {
      // Список для збереження прогнозів
      List<RoomOccupancyForecast> forecasts = new
List<RoomOccupancyForecast>();

      // Розрахунок кількості днів в історичному та прогнозованому
періодах
      int historyDays = (historyEndDate - historyStartDate).Days + 1;
      int forecastDays = (forecastEndDate - forecastStartDate).Days +
1;

      // SQL-запит для отримання середніх значень, стандартного
відхилення та прогнозованої завантаженості
      string query = @"
        SELECT
            r.RoomId, -- Унікальний ідентифікатор номера
            r.RoomNumber, -- Номер кімнати
            COALESCE(ROUND(AVG(DATEDIFF(rc.EndDate, rc.StartDate) +
1)), 0) AS AvgOccupiedDays, -- Середня кількість днів зайнятості
            COALESCE(ROUND(STDDEV(DATEDIFF(rc.EndDate, rc.StartDate) +
1)), 0) AS StdDevOccupiedDays, -- Стандартне відхилення завантаженості
            COALESCE(ROUND(AVG(DATEDIFF(rc.EndDate, rc.StartDate) + 1)
* @ForecastDays / @HistoryDays), 0) AS ForecastedOccupiedDays --
Прогнозована кількість зайнятих днів
        FROM room r
        LEFT JOIN registrationclient rc ON r.RoomId = rc.RoomId
            AND rc.StartDate BETWEEN @HistoryStartDate AND
@HistoryEndDate -- Вибірка даних лише за обраний історичний період
        GROUP BY r.RoomId, r.RoomNumber;
    ";

      using (MySqlConnection connection = new
MySqlConnection(_ConnString)) {
        // Ініціалізація SQL-команди
        MySqlCommand command = new MySqlCommand(query, connection);

        // Додавання параметрів для безпечної передачі даних
        command.Parameters.AddWithValue("@HistoryStartDate",
historyStartDate);
        command.Parameters.AddWithValue("@HistoryEndDate",
historyEndDate);
        command.Parameters.AddWithValue("@HistoryDays", historyDays);
        command.Parameters.AddWithValue("@ForecastDays", forecastDays);

        // Ініціалізація генератора випадкових чисел для варіацій
```

```
        Random random = new Random();

        // Відкриття з'єднання з базою даних
        connection.Open();

        // Виконання запиту та обробка результатів
        using (MySqlDataReader reader = command.ExecuteReader()) {
          while (reader.Read()) {
            // Створення об'єкта прогнозу для номера
            RoomOccupancyForecast forecast = new RoomOccupancyForecast
{
              RoomId = Convert.ToInt32(reader["RoomId"]), //
Ідентифікатор номера
              RoomNumber = reader["RoomNumber"].ToString(), // Номер
кімнати
              AvgOccupiedDays =
Convert.ToDecimal(reader["AvgOccupiedDays"]), // Середня зайнятість
              ForecastedOccupiedDays =
Convert.ToDecimal(reader["ForecastedOccupiedDays"]) // Початковий
прогноз завантаженості
            };

            // Додавання випадкової варіації до прогнозу (-10% до +10%)
            decimal variation = (decimal)(random.Next(-10, 11) /
100.0); // Розрахунок відсоткового відхилення
            forecast.ForecastedOccupiedDays +=
forecast.ForecastedOccupiedDays * variation;

            // Заокруглення прогнозу до найближчого цілого числа
            forecast.ForecastedOccupiedDays =
Math.Round(forecast.ForecastedOccupiedDays);

            // Додавання об'єкта прогнозу до списку
            forecasts.Add(forecast);
          }
        }
      }

      // Повернення списку прогнозів
      return forecasts;
    }
  }
}
```

Призначення методу

Метод дозволяє прогнозувати кількість зайнятих днів для кожного номера у майбутньому періоді, використовуючи історичні дані про завантаженість. Прогноз враховує середнє значення зайнятості, стандартне відхилення, а також додає випадкові варіації для реалістичності.

Аргументи методу

1. historyStartDate – дата початку історичного періоду для аналізу.
2. historyEndDate – дата закінчення історичного періоду для аналізу.
3. forecastStartDate – дата початку прогнозованого періоду.
4. forecastEndDate – дата завершення прогнозованого періоду.

Логіка роботи

1. Підготовка до прогнозу:

– Обчислюється тривалість історичного періоду (historyDays) і прогнозованого періоду (forecastDays) в днях.

– SQL-запит отримує середню тривалість зайнятості номерів, стандартне відхилення та початковий прогноз завантаженості.

2. SQL-запит:

– Дані отримуються з таблиць room і registrationclient.

– У таблиці room міститься інформація про номери.

– У таблиці registrationclient зберігаються записи про заселення клієнтів.

– Дані фільтруються за вказаним історичним періодом (StartDate BETWEEN @HistoryStartDate AND @HistoryEndDate).

– Для кожного номера обчислюються:

Середня кількість зайнятих днів (записів у періоді).

Стандартне відхилення зайнятості.

Прогнозована кількість зайнятих днів у майбутньому періоді, розрахована за пропорцією тривалості періодів.

3. Реалізація прогнозу:

– Для кожного номера створюється об'єкт RoomOccupancyForecast.

– До прогнозу додається випадкова варіація в межах від -10% до +10%.

– Прогнозоване значення заокруглюється до найближчого цілого числа.

4. Результат:

– Метод повертає список об'єктів RoomOccupancyForecast, де кожен об'єкт містить:

Ідентифікатор номера (RoomId).

Номер кімнати (RoomNumber).

Середню кількість зайнятих днів (AvgOccupiedDays).

Прогнозовану кількість зайнятих днів (ForecastedOccupiedDays).

Важливі моменти

SQL-запит:

– Використовує функції AVG для середнього значення та STDDEV для стандартного відхилення.

– Обчислення виконується у пропорції до тривалості періодів.

Безпека даних:

– Використовуються параметризовані запити (@ParameterName) для захисту від SQL-ін'єкцій.

Реалістичність прогнозу:

– Випадкова варіація додає реалістичність до прогнозу, враховуючи непередбачуваність завантаженості.

Обробка помилок:

– Код відкриває і закриває з'єднання з базою даних коректно.

3.5 Тестування інформаційної системи: перевірка на ефективність і відповідність вимогам.

В розділі буде представлено аналіз практичних результатів тестування програмної системи. Цей розділ є важливим етапом оцінки функціональності, продуктивності та надійності системи перед її впровадженням у реальне середовище.

Метою цього розділу є детальне описання проведеного тестування, представлення отриманих результатів та формулювання висновків на їх основі. Процес тестування включав використання реальних або штучно створених тестових даних, які були відібрані для відображення різноманітних сценаріїв використання системи.

Тестування є важливим етапом у розробці інформаційної системи, оскільки дозволяє виявити недоліки, оцінити ефективність роботи та перевірити відповідність функціональних і нефункціональних характеристик початковим вимогам.

Початкове вікно програми є важливим елементом користувальницького інтерфейсу, оскільки воно надає зручний доступ до основних функцій та можливостей програми. Форма з меню програми містить набір кнопок або пунктів меню, які представляють різноманітні опції для користувача. Це можуть бути такі функції, як управління клієнтами, бронювання кімнат, робота з послугами та інші. Користувач може вибрати потрібний пункт меню або кнопку, щоб перейти до відповідної сторінки або функціоналу програми. Таке початкове вікно дозволяє зробити роботу з програмою зручною та інтуїтивно зрозумілою для користувача.

На (рис 3.1) зображено початкове вікно.

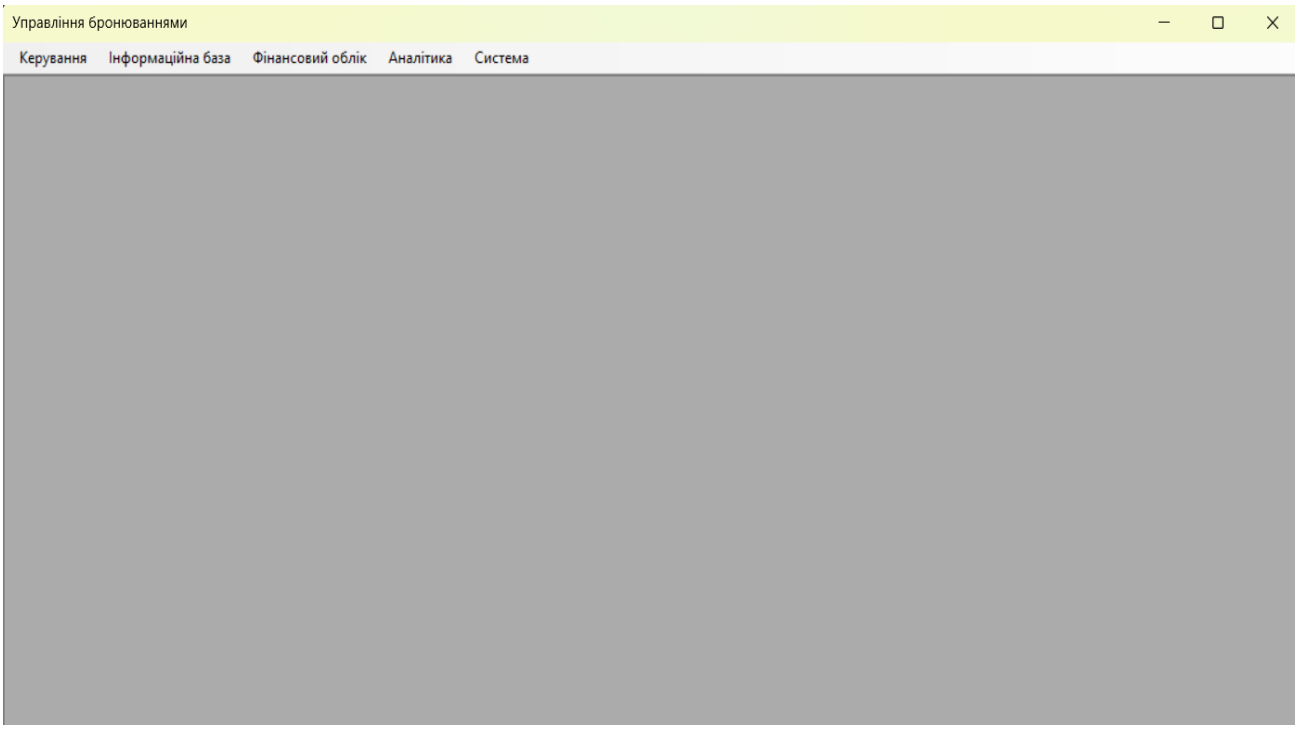

Рис. 3.1. Початкове меню

Після натискання на кнопку із меню Керування появляється підменю яке зображенна на (рис. 3.2) в якому є такі підпункти: Бронювання, Обслуговування гостей, Контроль виконання задач, Вихід.

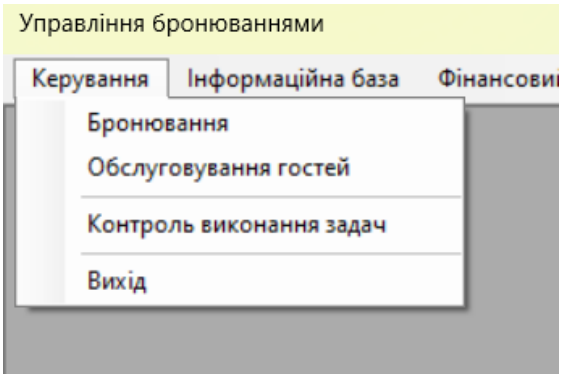

Рис. 3.2. Підменю Керування

Дочірня форма, яка з'являється після натискання кнопки "Бронювання" на початковій формі є важливим елементом функціональності програми. Вона надає зручний спосіб управління даними про бронювання номерів готельного комплексу. На цій формі користувач може додавати, редагувати існуючі дані та видаляти записи про бронювання за необхідністю. (Рис. 3.3) відображає структуру цієї дочірньої форми, де представлені поля для введення основної інформації для броні номеру, а також відображається перелік вже заброньованих номерів. Це дозволяє адміністратору зручно та ефективно керувати даними.

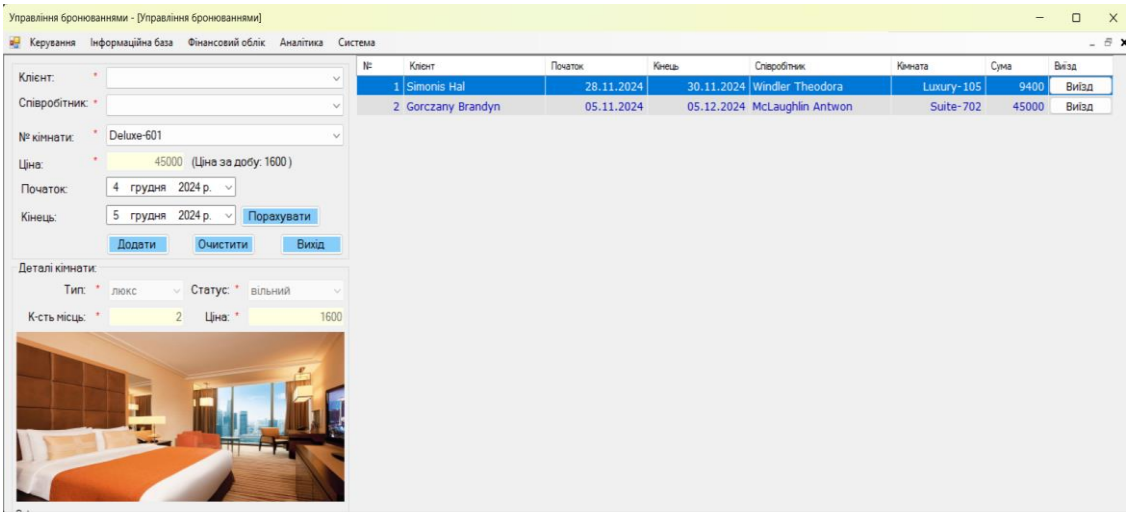

Рис. 3.3 Форма для бронювання

Після натискання на кнопку Обслуговування гостей здійснюється перехід на дочірню форму, яка дасть можливість додавати, редагувати і видаляти дані про додаткові послуги у готелі, яка зображенна на (рис. 3.4).

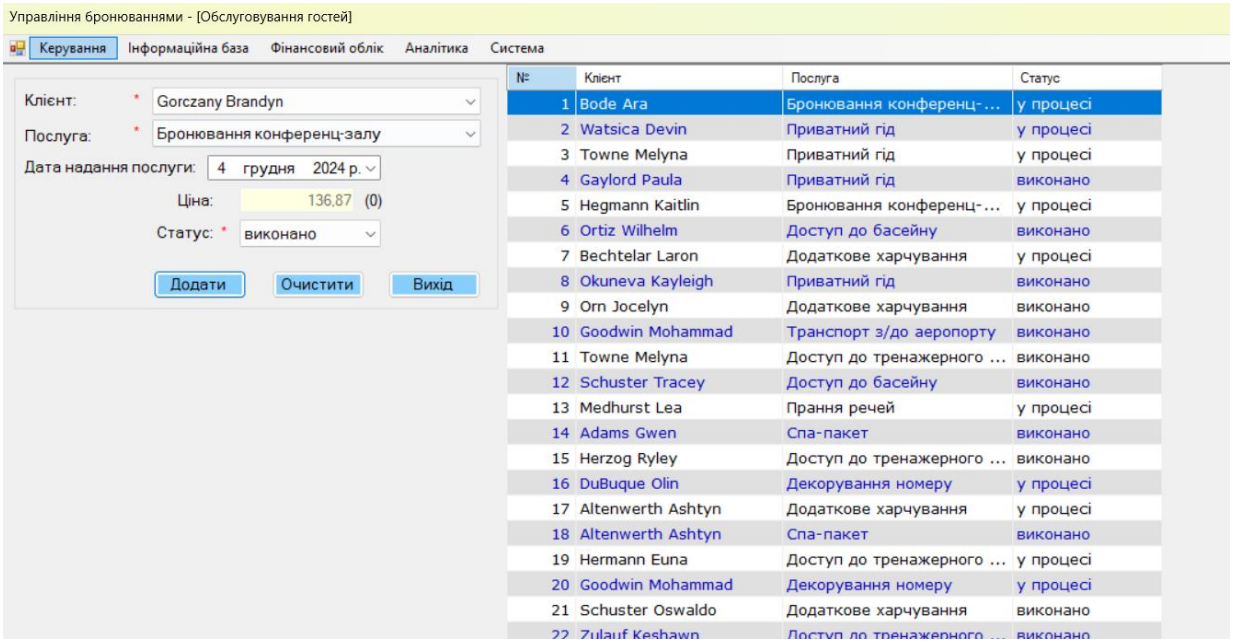

Рис. 3.4. Форма для обслуговування гостей

Після натискання на кнопку Контроль виконання задач в меню на формі здійснюється перехід на дочірню форму, яка дасть можливість додавати, редагувати і видаляти дані про задачі у готелі, яка зображенна на (рис. 3.5).

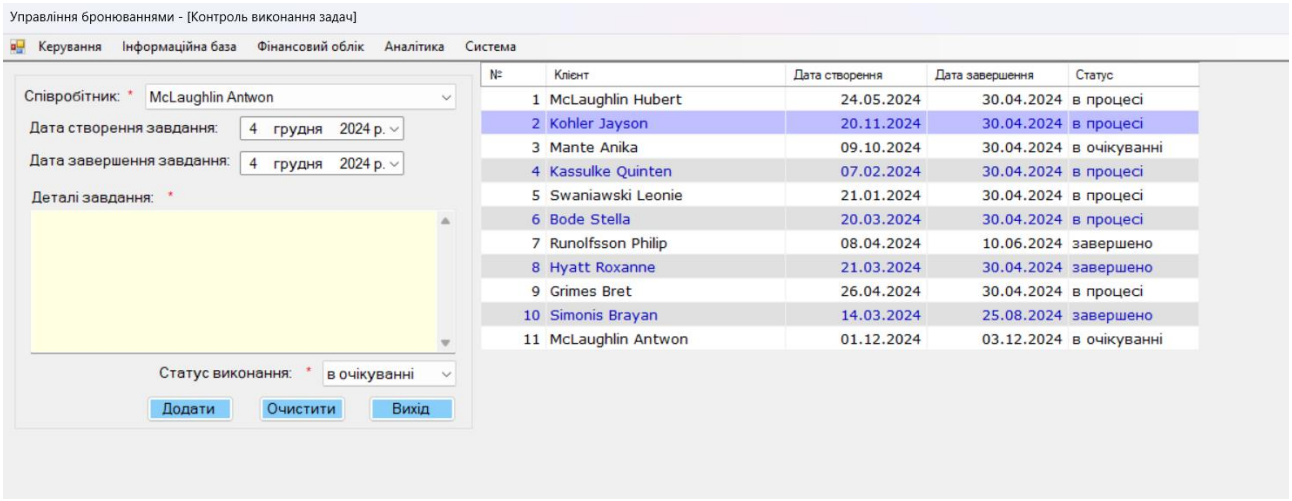

Рис. 3.5. Форма для контролю виконання задач

Після натискання на кнопку із меню Інформаційна база появляється підменю яке зображенна на (рис. 3.6).

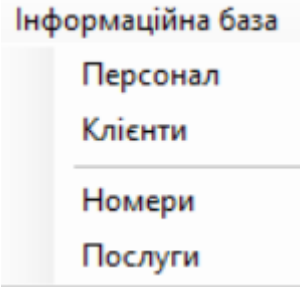

Рис. 3.6. Підменю Інформаційна база

Після натискання на кнопку Персонал в меню на формі здійснюється перехід на дочірню форму Співробітники, яка дасть можливість додавати, редагувати і видаляти дані про Співробітників готелю і не тільки, яка зображенна на (рис. 3.7).

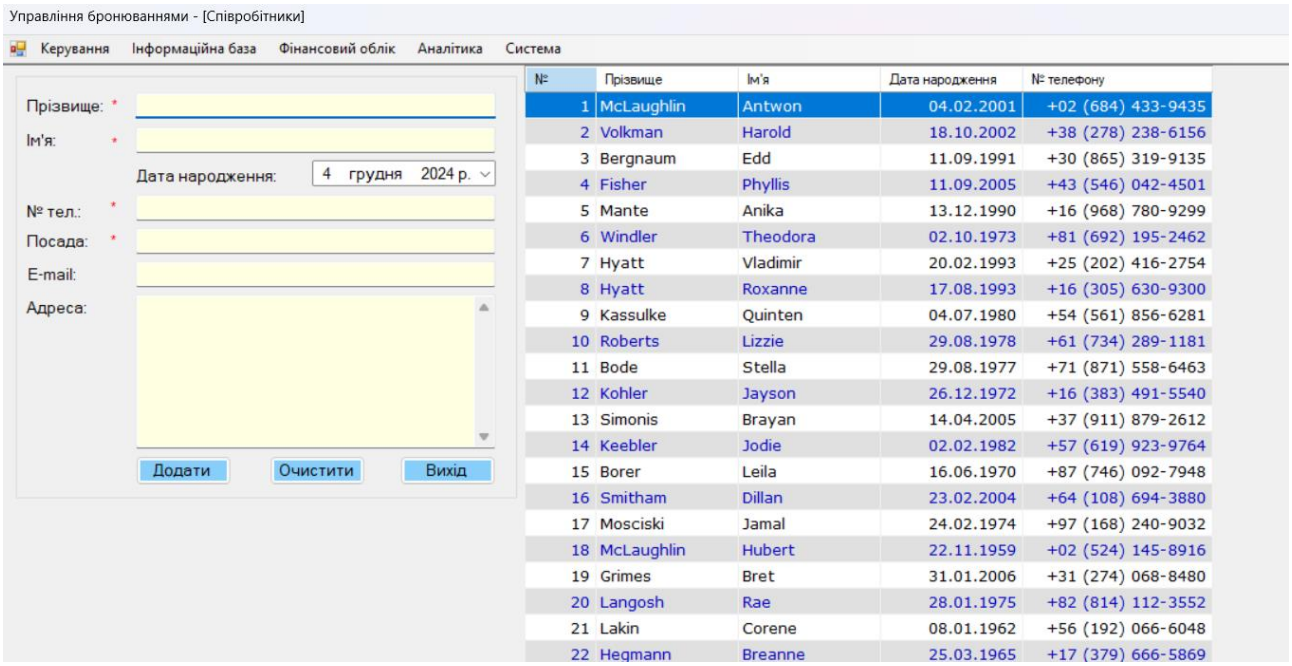

Рис. 3.7. Форма Співробітники

Аналогічно розроблено для форм Клієнти, Номери та Послуги.

Після натискання на кнопку Фінансовий облік обравши Генерація рахунків для клієнтів здійснюється перехід на дочірню форму, яка дасть можливість сформувати рахунок для клієнта і надрукувати його, яка зображенна на (рис. 3.8).

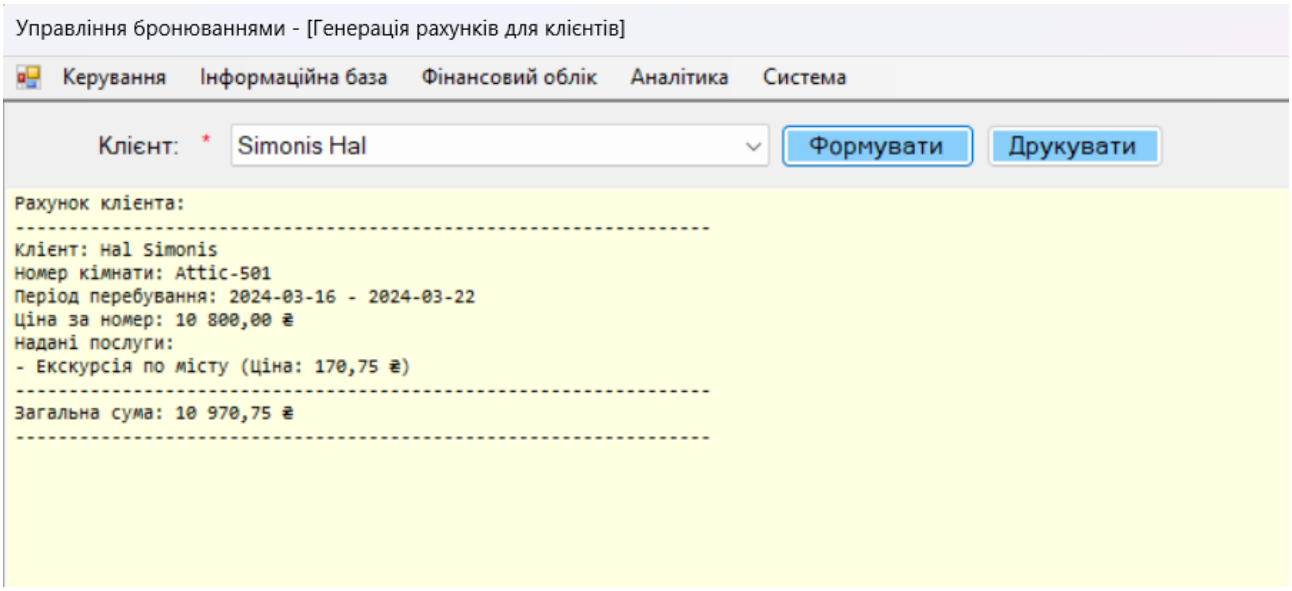

Рис. 3.8. Форма для формування рахунка для клієнта

Після натискання на кнопку Фінансовий облік обравши Доходи за період здійснюється перехід на дочірню форму, яка дасть можливість формувати фінансовий звіт за фказаний період, яка зображенна на (рис. 3.9).

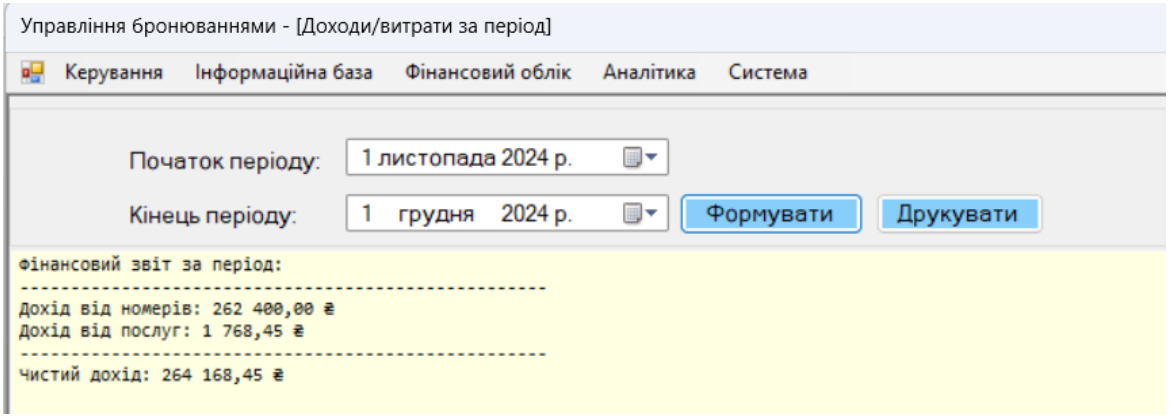

Рис. 3.9. Форма Доходи за місяць

Після натискання на кнопку Аналітика обравши Статистика заповнюваності номерів здійснюється перехід на дочірню форму, яка дасть можливість формувати звіт за вказаний період в якому будем бачити статистику заповнюваності номерів, яка зображенна на (рис. 3.10).

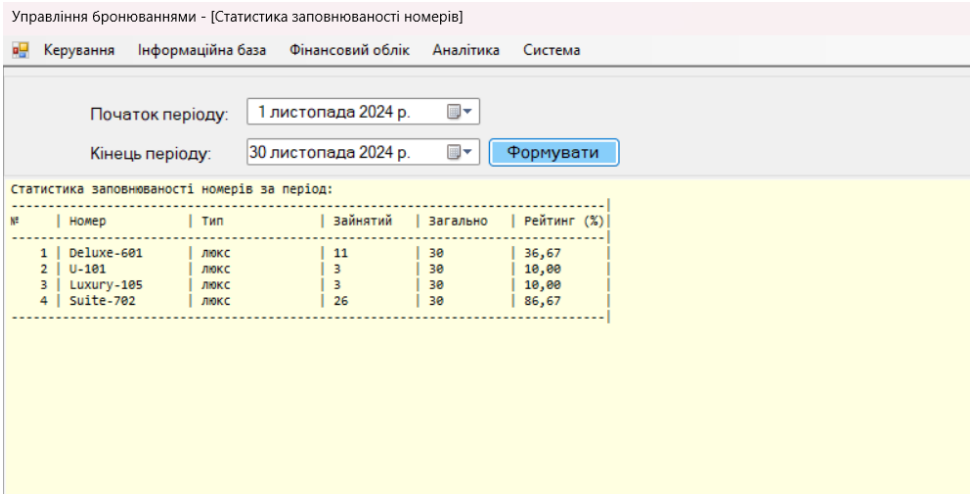

Рис. 3.10. Статистика номерів

Після натискання на кнопку Аналітика обравши Аналіз популярності послуг здійснюється перехід на дочірню форму, яка дасть можливість формувати звіт за вказаний період в якому будем бачити статистику Аналізу послуг в якій будем бачити назва послуги, кількість використання і дохід, яка зображенна на (рис. 3.11).

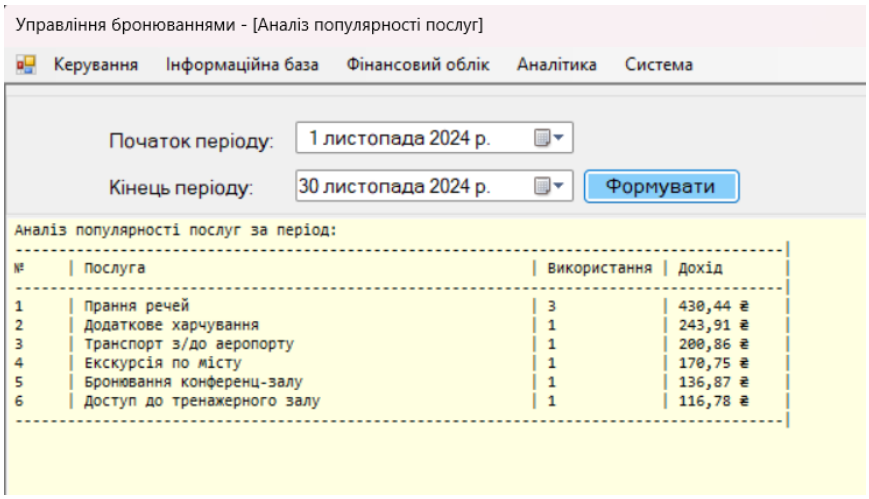

Рис. 3.11. Аналіз популярності послуг

Після натискання на кнопку Аналітика обравши Прогнозування завантаженості здійснюється перехід на дочірню форму, яка дасть можливість формувати прогноз за вказаний період в якому будем бачити прогноз навантаженості номерів, як це зображено на (рис. 3.12).

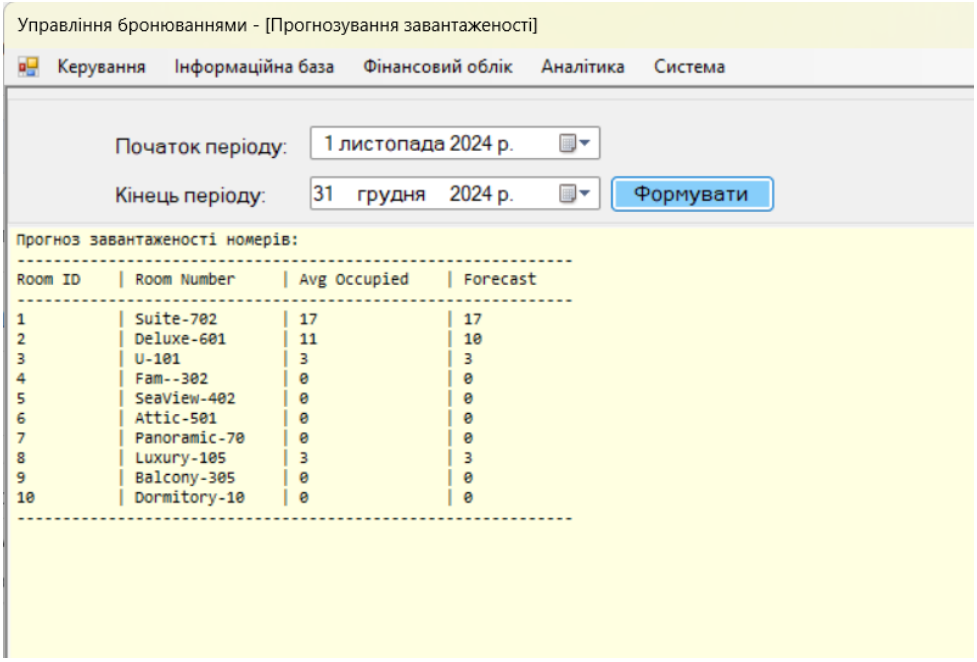

Рис. 3.12. Прогнозування завантаженості

Після натискання на кнопку Система обравши Користувачі системи здійснюється перехід на дочірню форму, яка дасть можливість редагувати додаватии і видаляти користувачів системи, як це зображено на (рис. 3.13).

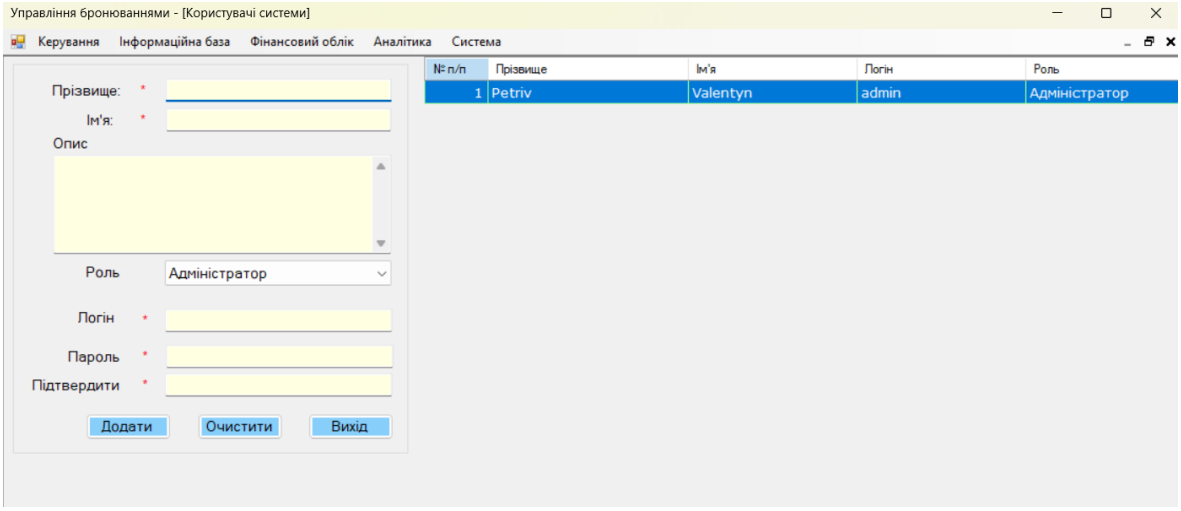

Рис. 3.13. Користувачі системи

Після натискання на кнопку Система обравши Події системи здійснюється перехід на дочірню форму, яка дасть можливість переглядати події про користувачів системи, як це зображено на (рис. 3.14).

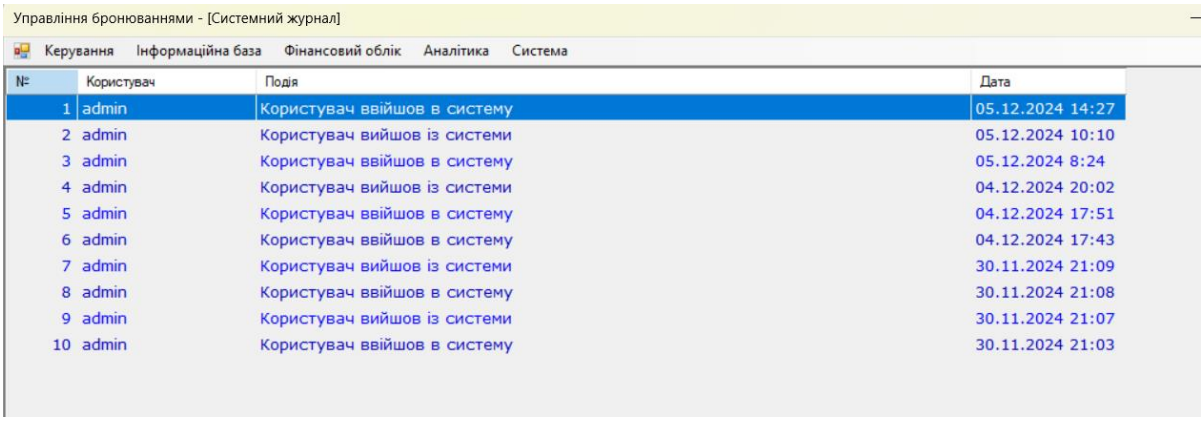

Рис. 3.14. Системний журнал

Після натискання на кнопку Система обравши Персоналізація здійснюється перехід на дочірню форму, яка дасть можливість переглядати дані про користувача який в системі, як це зображено на (рис. 3.15).

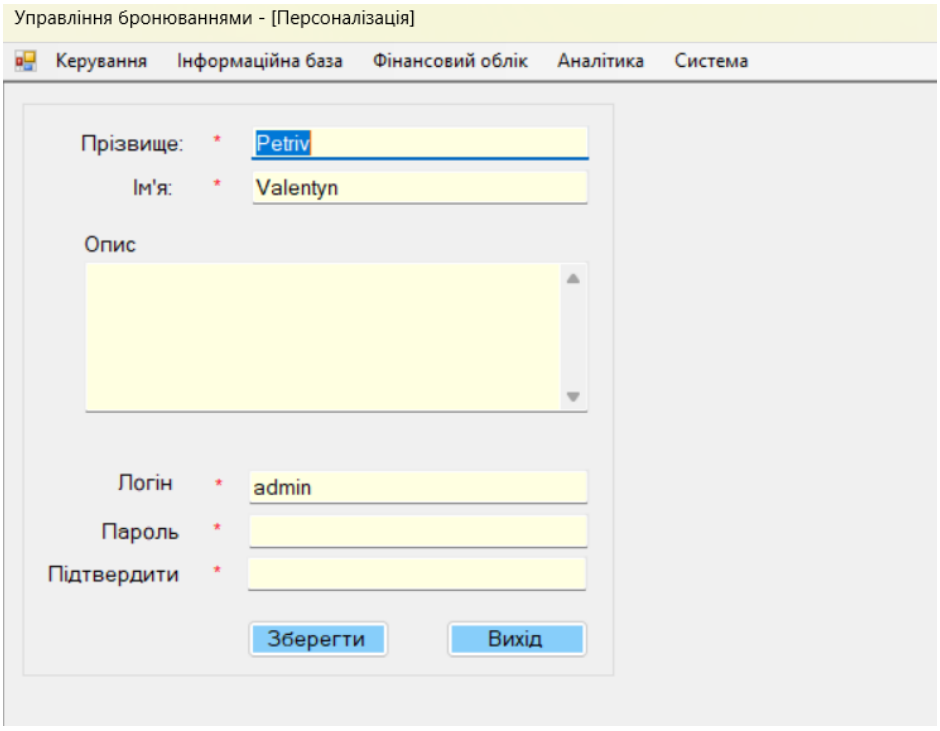

Рис. 3.15. Персоналізація

На (рис 3.16) відображається Авторизація нового користувача

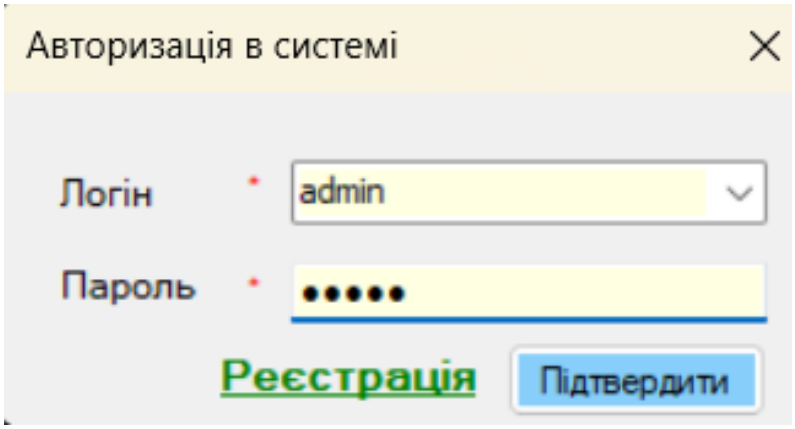

Рис. 3.16. Зміна користувача

На (Рис. 3.17) ілюструє Реєстрацію нового користувача

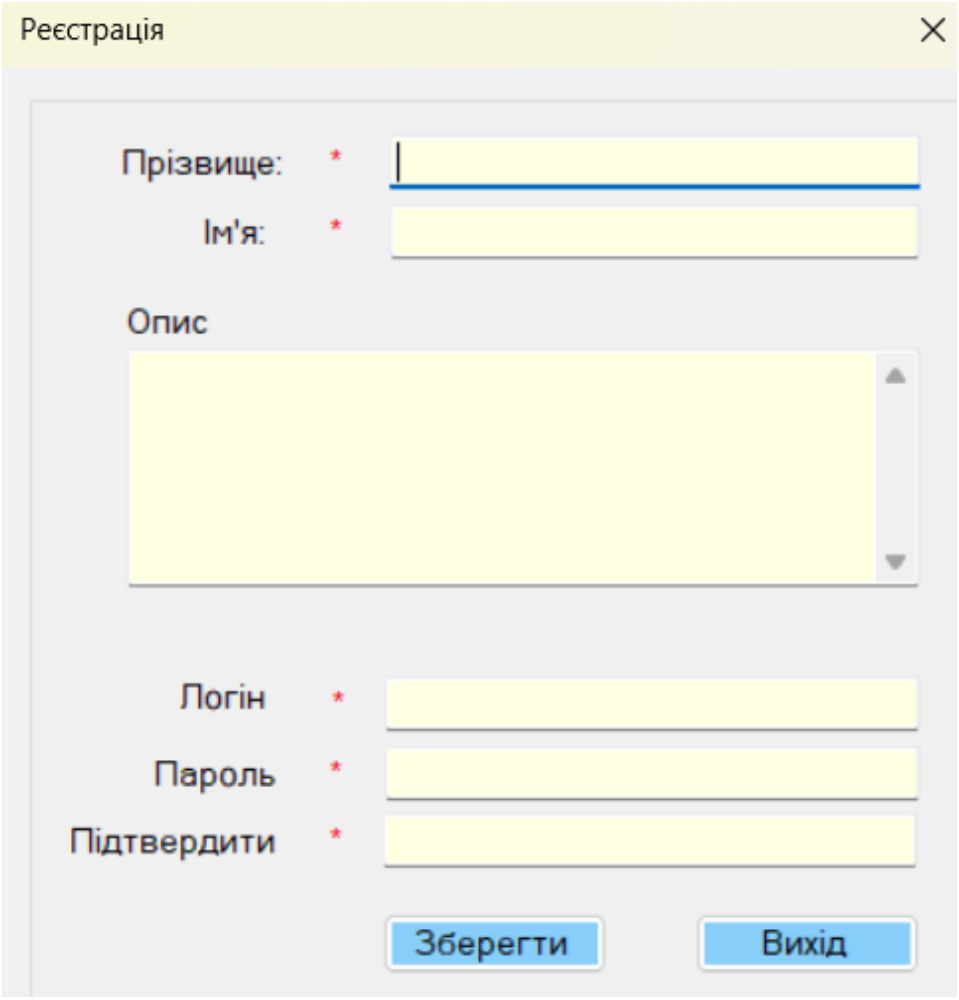

Рис. 3.17. Реєстрація користувача

Після завершення етапу тестування програмного забезпечення, проводиться детальний аналіз отриманих результатів з метою оцінки функціональності, надійності та відповідності системи поставленим вимогам.

1. Оцінка загальної ефективності системи.

Аналіз роботи системи на основі тестових даних дозволяє визначити, наскільки вона відповідає заданим критеріям ефективності. Зокрема, оцінюються такі параметри, як швидкість виконання основних операцій, стабільність роботи, а також обсяг системних ресурсів, необхідних для функціонування. У разі виявлення недоліків у продуктивності, проводиться їх детальний розгляд для можливого усунення.

2. Виявлення слабких місць та рекомендації щодо вдосконалення. У ході тестування можуть бути виявлені окремі аспекти роботи системи, які потребують доопрацювання. Це можуть бути як технічні помилки (наприклад, некоректна обробка даних або невідповідність інтерфейсу вимогам зручності), так і загальні проблеми архітектури. На основі виявлених недоліків розробляються рекомендації щодо їх усунення, включаючи пропозиції щодо оптимізації алгоритмів, покращення інтерфейсу або додавання нових функцій.

3. Порівняння результатів тестування з очікуваннями проекту. Особлива увага приділяється зіставленню отриманих результатів з початковими вимогами, визначеними на етапі планування проекту. Це допомагає оцінити, чи відповідає розроблена система функціональним, технічним і бізнес-вимогам замовника. У разі розходжень між очікуваними та фактичними результатами аналізується причина та формулюються рекомендації для подальшого вдосконалення.

3.6 Висновки до розділу

У третьому розділі було розглянуто практичні аспекти створення інформаційної системи для автоматизації бізнес-процесів готельного комплексу.

На основі аналізу вимог до системи визначено основні функціональні та нефункціональні характеристики, які забезпечують ефективну роботу програми. Було запропоновано архітектуру системи, що базується на клієнт-серверній моделі та

модульному підході. Така структура забезпечує гнучкість і масштабованість системи, дозволяючи легко додавати нові компоненти або інтегруватися з іншими сервісами.

Розробка програмного забезпечення здійснювалася із застосуванням сучасних інструментів, таких як мова програмування C#, середовище Visual Studio, база даних MySQL та фреймворки .NET Framework. Особливу увагу було приділено створенню бази даних, побудованої на основі ER-діаграми, яка ефективно організовує зберігання інформації про клієнтів, бронювання, фінансові операції тощо.

Реалізований функціонал охоплює ключові бізнес-процеси готелю, такі як управління бронюванням, клієнтами, фінансами та ресурсами. Завдяки впровадженню інтуїтивно зрозумілого інтерфейсу та алгоритмів автоматизації, система забезпечує зручність використання, зменшення рутинної роботи та підвищення продуктивності персоналу.

На етапі тестування інформаційної системи було проведено перевірку її ефективності, працездатності всіх компонентів та відповідності початковим вимогам. Це гарантує надійну роботу системи в умовах реального використання.

Таким чином, створена інформаційна система повністю відповідає поставленим цілям і завданням, забезпечуючи оптимізацію ключових бізнес-процесів готельного комплексу, покращення якості обслуговування клієнтів та ефективне управління ресурсами.

ВИСНОВКИ

У цій магістерській роботі було розглянуто питання автоматизації бізнес-процесів готельного комплексу за допомогою технологій .NET, що дозволяє ефективно управляти основними операціями, такими як облік клієнтів, управління бронюваннями, номерним фондом, фінансовими операціями та звітністю. Розроблена система забезпечує гнучке і надійне рішення для автоматизації роботи адміністративного персоналу, що значно підвищує ефективність та якість обслуговування клієнтів.

У ході виконання проєкту було удосконалено програму з інтуїтивно зрозумілим інтерфейсом, яка дозволяє виконувати такі функції:

– Управління клієнтами: додавання, редагування та видалення інформації про клієнтів, що зберігається у базі даних.

– Бронювання номерів: автоматизація процесів бронювання, ведення записів про наявність і стан номерів.

– Фінансовий облік: генерація рахунків для клієнтів, контроль фінансових операцій і створення звітів про доходи.

– Аналітика та прогнозування: розрахунок завантаженості номерного фонду на основі історичних даних та прогнозування майбутньої завантаженості.

– Управління персоналом: реєстрація співробітників, розподіл ролей, а також контроль виконання поставлених задач.

Особливістю розробки стало використання мови програмування C# у поєднанні з MySQL для створення бази даних. Такий підхід дозволив забезпечити високу надійність системи, швидкість обробки даних і масштабованість. У MySQL було створено реляційну базу даних із налаштуванням зв’язків між таблицями, що дозволяє оптимально структурувати дані та ефективно здійснювати їх обробку.

Розроблений функціонал включає в себе підтримку обліку клієнтів, зв’язок із процесами управління номерами, а також автоматизоване створення звітів. Це допомагає адміністраторам швидко реагувати на запити клієнтів, забезпечувати високий рівень обслуговування та здійснювати аналіз ефективності роботи готелю.

Переваги впровадженої системи:

1. Оптимізація рутинних процесів: зменшення часу, необхідного для виконання щоденних задач, таких як реєстрація клієнтів, управління бронюваннями та контроль фінансових операцій.

2. Автоматизація аналітики: забезпечення швидкого доступу до статистичних даних, таких як середня завантаженість номерів, популярність послуг і прогнозування попиту.

3. Вдосконалено і реалізовано алгоритм прогнозування попиту та оптимізації ресурсів, які адаптовані до специфіки вдосконалення готельного розвитку.

4. Вдосконалено роботу модулів при бронюванні номерів та якісного обслуговування клієнтів в єдиній програмній системі.

Функціонал може бути вдосконалено за рахунок:

– Додавання модуля мобільного доступу, що дозволить користувачам виконувати операції з будь-якого пристрою.

– Реалізації інтеграції з зовнішніми платформами, наприклад, сервісами онлайн-бронювання або платіжними системами.

– Запровадження алгоритмів машинного навчання для прогнозування попиту на послуги, управління цінами та оптимізації ресурсів.

Таким чином, розроблена система удосконалена для автоматизації готельного бізнесу. Вона допомагає оптимізувати роботу персоналу, підвищити якість обслуговування клієнтів і забезпечити управління ресурсами. Завдяки використанню сучасних технологій, система відповідає вимогам безпеки, продуктивності та зручності. Дану роботу будем рекомендувати для впровадження в готельних комплексах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. О'Коннор, П. (2010). "Управління доходами готелів: Принципи та практика."
2. Зенгул, Ф. Д., & Макфарлейн, Д. (2020). "Системи управління готельними комплексами: Комплексне керівництво."
3. Amadeus Hospitality (2024). "Оптимізація операцій готелю через розумне управління даними."
4. Opera Property Management System Documentation.
5. Пера, Р., & Ліан, З. (2021). "Виклики масштабованості алгоритмів у системах управління готельними комплексами."
6. CRM і автоматизація в готельному бізнесі [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://er.chdtu.edu.ua/bitstream/ChSTU/3112/1/4.pdf
7. Оптимізація бізнес-процесів у сфері гостинності [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://repo.btu.kharkov.ua/
8. Роль BPMN і новітніх технологій [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://ribashotelsgroup.ua/
9. Готельний бізнес – Мальська М. П – 3.1. Сутність процесу управління готельними підприємством
10. Walker, J. R. (2016). Introduction to Hospitality Management. Pearson.
11. BPMN Method and Style by Bruce Silver
12. Tourism Management та International Journal of Hospitality Management.
13. O’Fallon, M. J., & Rutherford, D. G. (2010). Hotel Management and Operations. Wiley
14. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach.
15. Booch, G., Jacobson, I., & Rumbaugh, J. (2005). Unified Modeling Language User Guide.
16. Desktop Applications Programming with C# and .NET Core by Alessandro Del Sole.
17. Business Process Model and Notation (BPMN)

18. Algorithms in automation: a review"

19. subj.ukr-lit.com [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://subj.ukr-lit.com

20. Офіційний сайт MySQL [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://www.mysql.com/

21. Документація MySQL [Електронний ресурс]:[Веб-сайт] – Режим доступу: https://dev.mysql.com/doc

22. «Дизайн бази даних для простих смертних: практичний посібник із проектування реляційної бази даних» – Майкл Дж. Ернандес.

23. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. − Київ. –2017. –110 с.

24. StudFile [Електронний ресурс]:[Веб-сайт] – Файловий архів для студентів. – Режим доступу: https://studfile.net

25. C# [Електронний ресурс]:[Веб-сайт] – Документація C#. – URL: https://learn.microsoft.com/

26. Metanit [Електронний ресурс]:[Веб-сайт] – Веб-сайт в форматі системи тематичних колективних блогів. – Режим доступу: https://metanit.com

27. CoderLessons [Електронний ресурс]:[Веб-сайт] – Уроки по програмування – Режим доступу: https://coderlessons.com

28. AboutSoft [Електронний ресурс]:[Веб-сайт] – Веб-сайт про програмне забезпечення. – Режим доступу: https://uk.aboutsoft.top

29. Visual-Paradigm [Електронний ресурс]:[Веб-сайт] – документація UML. – Режим доступу: https://www.visualparadigm.com

30. «Системи баз даних: повна книга» - Гектор Гарсіа-Моліна, Джеффрі Д. Уллман, Дженніфер Відом.

31. «Мистецтво SQL» - Стефан Фару, Пітер Робсон.

32. «C# 9 і .NET 5 – сучасна кросплатформна розробка» - Марк Дж. Прайс.

33. "Поглиблено C#" - Джон Скіт.

34. «Head First C#» - Дженніфер Грін, Ендрю Стелман.

35. "Дизайн інтерфейсу" - Володимир Онофрійчук.

36. "UI/UX-дизайн: теорія та практика" - Олег Яковлев.

37. "Інтерфейс користувача: принципи і підходи" - Юрій Морозов.

38. "Дизайн інтерфейсу: практичні рекомендації" - Ігор Дудка.

39.https://www.amadeus-hospitality.com/

40. "UML. Опис моделей програмного забезпечення" - Олександр Лисенков

41. "Проектування програмного забезпечення за допомогою UML" - Юрій Полищук ISBN:

42. "UML та RUP. Проектування програмного забезпечення" - Олег Гавриш

43. "Проектування десктопних програм з використанням C#" - Андрій Даниленко

44. "Проектування десктопних додатків на платформі .NET" - Андрій Коневський

45. ".NET: підручник C# 8.0" - Олексій Блинков, Андрій Коржов

ДОДАТКИ

Додаток А

Код створення БД

```
-- Таблиця клієнтів
DROP TABLE IF EXISTS `client`;
CREATE TABLE `client` (
  `ClientId` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) DEFAULT NULL,
  `LastName` VARCHAR(45) DEFAULT NULL,
  `BirthDate` DATETIME DEFAULT NULL,
  `Address` VARCHAR(450) DEFAULT NULL,
  `Email` VARCHAR(100) DEFAULT NULL,
  `Phone` VARCHAR(20) DEFAULT NULL,
  PRIMARY KEY (`ClientId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця співробітників
DROP TABLE IF EXISTS `employee`;
CREATE TABLE `employee` (
  `EmployeeId` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) DEFAULT NULL,
  `LastName` VARCHAR(45) DEFAULT NULL,
  `BirthDate` DATETIME DEFAULT NULL,
  `Address` VARCHAR(450) DEFAULT NULL,
  `Email` VARCHAR(100) DEFAULT NULL,
  `Phone` VARCHAR(20) DEFAULT NULL,
  `Position` VARCHAR(45) DEFAULT NULL,
  PRIMARY KEY (`EmployeeId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця логів
DROP TABLE IF EXISTS `logs`;
CREATE TABLE `logs` (
  `LogsId` INT NOT NULL AUTO_INCREMENT,
  `UsersId` INT NOT NULL,
  `EventNameShow` TEXT NOT NULL,
  `EventDate` DATETIME NOT NULL,
  PRIMARY KEY (`LogsId`),
  KEY `UsersId_fk_idx` (`UsersId`),
  CONSTRAINT `UsersId_fk` FOREIGN KEY (`UsersId`) REFERENCES `users`
(`UsersId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця реєстрації клієнтів
DROP TABLE IF EXISTS `registrationclient`;
CREATE TABLE `registrationclient` (
  `RegistrationClientId` INT NOT NULL AUTO_INCREMENT,
  `EmployeeId` INT DEFAULT NULL,
  `ClientId` INT DEFAULT NULL,
```

```
  `RoomId` INT DEFAULT NULL,
  `StartDate` DATETIME DEFAULT NULL,
  `EndDate` DATETIME DEFAULT NULL,
  `Price` DECIMAL(10,2) DEFAULT NULL,
  `IsResides` TINYINT(1) DEFAULT NULL,
  PRIMARY KEY (`RegistrationClientId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця номерів
DROP TABLE IF EXISTS `room`;
CREATE TABLE `room` (
  `RoomId` INT NOT NULL AUTO_INCREMENT,
  `RoomNumber` VARCHAR(45) NOT NULL,
  `NumberType` VARCHAR(45) NOT NULL,
  `NumberOfSeats` INT DEFAULT NULL,
  `Price` DECIMAL(10,2) DEFAULT NULL,
  `Description` TEXT,
  `Equipment` VARCHAR(500) DEFAULT NULL,
  `Image` LONGBLOB,
  `Status` VARCHAR(45) DEFAULT NULL,
  PRIMARY KEY (`RoomId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця послуг
DROP TABLE IF EXISTS `service`;
CREATE TABLE `service` (
  `ServiceId` INT NOT NULL AUTO_INCREMENT,
  `ServiceName` VARCHAR(45) NOT NULL,
  `Description` TEXT,
  `Price` DECIMAL(10,2) NOT NULL,
  `AvailableFrom` DATETIME DEFAULT NULL,
  `AvailableUntil` DATETIME DEFAULT NULL,
  PRIMARY KEY (`ServiceId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця наданих послуг
DROP TABLE IF EXISTS `serviceprovided`;
CREATE TABLE `serviceprovided` (
  `ServiceProvidedId` INT NOT NULL AUTO_INCREMENT,
  `ClientId` INT DEFAULT NULL,
  `ServiceId` INT DEFAULT NULL,
  `Price` DECIMAL(10,2) NOT NULL,
  `Status` VARCHAR(45) DEFAULT NULL,
  `DateProvided` DATETIME DEFAULT NULL,
  PRIMARY KEY (`ServiceProvidedId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця задач
DROP TABLE IF EXISTS `tasks`;
CREATE TABLE `tasks` (
  `TaskId` INT NOT NULL AUTO_INCREMENT,
  `EmployeeId` INT DEFAULT NULL,
  `TaskDate` DATETIME DEFAULT NULL,
```

```
  `CompletionDate` DATETIME DEFAULT NULL,
  `TaskDetails` TEXT,
  `TaskStatus` VARCHAR(45) DEFAULT NULL,
  PRIMARY KEY (`TaskId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

-- Таблиця користувачів
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `UsersId` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) DEFAULT NULL,
  `LastName` VARCHAR(45) DEFAULT NULL,
  `UsersName` VARCHAR(45) NOT NULL,
  `UsersPassword` VARCHAR(150) NOT NULL,
  `RoleId` INT NOT NULL,
  `Description` TEXT,
  PRIMARY KEY (`UsersId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;
```

Додаток Б

Код створення функціоналу

```
using BookingsApp.AppCode;
using MySqlConnector;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BookingsApp.Providers {
  internal class RegistrationClientProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    ClientProvider _ClientProvider = new ClientProvider();
    List<Client> _ClientList = new List<Client>();
    RoomProvider _RoomProvider = new RoomProvider();
    List<Room> _RoomList = new List<Room>();
    EmployeeProvider _EmployeeProvider = new EmployeeProvider();
    List<Employee> _EmployeeList = new List<Employee>();


    public void InsertRegistrationClient(int EmployeeId, int ClientId,
int RoomId, DateTime StartDate, DateTime EndDate, double Price) {
      MySqlConnection connection = new MySqlConnection(_ConnString);
      string query = "INSERT INTO RegistrationClient (EmployeeId,
ClientId, RoomId, StartDate, EndDate, Price, IsResides) " +
                     "VALUES (@EmployeeId, @ClientId, @RoomId,
@StartDate, @EndDate, @Price, @IsResides)";
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@EmployeeId", EmployeeId);
      command.Parameters.AddWithValue("@ClientId", ClientId);
      command.Parameters.AddWithValue("@RoomId", RoomId);
      command.Parameters.AddWithValue("@StartDate",
StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@EndDate",
EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@Price", Price);
      command.Parameters.AddWithValue("@IsResides", 0);
      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }


    public List<RegistrationClient> GetAllRegistrationClient() {
      _ClientList = _ClientProvider.GetAllClient();
```

```
      _RoomList = _RoomProvider.GetAllRoom();
      _EmployeeList = _EmployeeProvider.GetAllEmployee();

      string query = "SELECT * FROM RegistrationClient";
      List<RegistrationClient> listAllRegistrationClient = new
List<RegistrationClient>();

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);

      connection.Open();
      using (MySqlDataReader reader = command.ExecuteReader()) {
        int i = 0;
        while (reader.Read()) {
          RegistrationClient oneRegistrationClient = new
RegistrationClient {
            Number = ++i,
            RegistrationClientId =
Convert.ToInt32(reader["RegistrationClientId"]),
            EmployeeId = Convert.ToInt32(reader["EmployeeId"]),
            ClientId = Convert.ToInt32(reader["ClientId"]),
            RoomId = Convert.ToInt32(reader["RoomId"]),
            StartDate = Convert.ToDateTime(reader["StartDate"]),
            EndDate = Convert.ToDateTime(reader["EndDate"]),
            Price = Convert.ToDouble(reader["Price"]),
            IsResides = Convert.ToBoolean(reader["IsResides"])
          };
          listAllRegistrationClient.Add(oneRegistrationClient);
        }
      }
      connection.Close();

      if (listAllRegistrationClient.Count == 0) {
        RegistrationClient noRegistrationClient = new
RegistrationClient {
          RegistrationClientId = 0,
          Message = NamesMy.NoDataNames.NoDataInRegistrationClient
        };
        listAllRegistrationClient.Add(noRegistrationClient);
      } else {
        for (int j = 0; j < listAllRegistrationClient.Count; j++) {
          listAllRegistrationClient[j].ClientFIO =
GetClientFIO(listAllRegistrationClient[j].ClientId, _ClientList);
          listAllRegistrationClient[j].EmployeeFIO =
GetEmployeeFIO(listAllRegistrationClient[j].EmployeeId, _EmployeeList);
          listAllRegistrationClient[j].RoomName =
GetRoomNumber(listAllRegistrationClient[j].RoomId, _RoomList);
        }
      }

      return listAllRegistrationClient;
    }
```

```
    public List<RegistrationClient> GetAllRegistrationClientResidents()
{
      _ClientList = _ClientProvider.GetAllClient();
      _EmployeeList = _EmployeeProvider.GetAllEmployee();
      _RoomList = _RoomProvider.GetAllRoom();

      string query = "SELECT * FROM RegistrationClient WHERE IsResides
= 0";
      List<RegistrationClient> listAllRegistrationClient = new
List<RegistrationClient>();

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);

      connection.Open();
      using (MySqlDataReader reader = command.ExecuteReader()) {
        int i = 0;
        while (reader.Read()) {
          RegistrationClient oneRegistrationClient = new
RegistrationClient {
            Number = ++i,
            RegistrationClientId =
Convert.ToInt32(reader["RegistrationClientId"]),
            EmployeeId = Convert.ToInt32(reader["EmployeeId"]),
            ClientId = Convert.ToInt32(reader["ClientId"]),
            RoomId = Convert.ToInt32(reader["RoomId"]),
            StartDate = Convert.ToDateTime(reader["StartDate"]),
            EndDate = Convert.ToDateTime(reader["EndDate"]),
            Price = Convert.ToDouble(reader["Price"]),
            IsResides = Convert.ToBoolean(reader["IsResides"])
          };
          listAllRegistrationClient.Add(oneRegistrationClient);
        }
      }
      connection.Close();

      if (listAllRegistrationClient.Count == 0) {
        RegistrationClient noRegistrationClient = new
RegistrationClient {
          RegistrationClientId = 0,
          Message = NamesMy.NoDataNames.NoDataInRegistrationClient
        };
        listAllRegistrationClient.Add(noRegistrationClient);
      } else {
        for (int j = 0; j < listAllRegistrationClient.Count; j++) {
          listAllRegistrationClient[j].ClientFIO =
GetClientFIO(listAllRegistrationClient[j].ClientId, _ClientList);
          listAllRegistrationClient[j].EmployeeFIO =
GetEmployeeFIO(listAllRegistrationClient[j].EmployeeId, _EmployeeList);
          listAllRegistrationClient[j].RoomName =
GetRoomNumber(listAllRegistrationClient[j].RoomId, _RoomList);
        }
```

```
      }

      return listAllRegistrationClient;
    }


    public List<RegistrationClient>
GetAllRegistrationClientNotResidents() {
      _ClientList = _ClientProvider.GetAllClient();
      _EmployeeList = _EmployeeProvider.GetAllEmployee();
      _RoomList = _RoomProvider.GetAllRoom();

      string query = "SELECT * FROM RegistrationClient WHERE IsResides
= 1";
      List<RegistrationClient> listAllRegistrationClient = new
List<RegistrationClient>();

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);

      connection.Open();
      using (MySqlDataReader reader = command.ExecuteReader()) {
        int i = 0;
        while (reader.Read()) {
          RegistrationClient oneRegistrationClient = new
RegistrationClient {
            Number = ++i,
            RegistrationClientId =
Convert.ToInt32(reader["RegistrationClientId"]),
            EmployeeId = Convert.ToInt32(reader["EmployeeId"]),
            ClientId = Convert.ToInt32(reader["ClientId"]),
            RoomId = Convert.ToInt32(reader["RoomId"]),
            StartDate = Convert.ToDateTime(reader["StartDate"]),
            EndDate = Convert.ToDateTime(reader["EndDate"]),
            Price = Convert.ToDouble(reader["Price"]),
            IsResides = Convert.ToBoolean(reader["IsResides"])
          };
          listAllRegistrationClient.Add(oneRegistrationClient);
        }
      }
      connection.Close();

      if (listAllRegistrationClient.Count == 0) {
        RegistrationClient noRegistrationClient = new
RegistrationClient {
          RegistrationClientId = 0,
          Message = NamesMy.NoDataNames.NoDataInRegistrationClient
        };
        listAllRegistrationClient.Add(noRegistrationClient);
      } else {
        for (int j = 0; j < listAllRegistrationClient.Count; j++) {
          listAllRegistrationClient[j].ClientFIO =
GetClientFIO(listAllRegistrationClient[j].ClientId, _ClientList);
```

```
          listAllRegistrationClient[j].EmployeeFIO =
GetEmployeeFIO(listAllRegistrationClient[j].EmployeeId, _EmployeeList);
          listAllRegistrationClient[j].RoomName =
GetRoomNumber(listAllRegistrationClient[j].RoomId, _RoomList);
        }
      }

      return listAllRegistrationClient;
    }



    private string GetRoomNumber(int RoomId, List<Room> RoomList) {
      for (int i = 0; i < RoomList.Count; i++) {
        if (RoomId == RoomList[i].RoomId) {
          return RoomList[i].RoomNumber.ToString();
        }
      }
      return "";
    }

    private string GetEmployeeFIO(int EmployeeId, List<Employee>
EmployeeList) {
      for (int i = 0; i < EmployeeList.Count; i++) {
        if (EmployeeId == EmployeeList[i].EmployeeId) {
          return EmployeeList[i].FIO;
        }
      }
      return "";
    }

    private string GetClientFIO(int ClientId, List<Client> ClientList)
{
      for (int i = 0; i < ClientList.Count; i++) {
        if (ClientId == ClientList[i].ClientId) {
          return ClientList[i].FIO;
        }
      }
      return "";
    }




    public RegistrationClient
SelectedRegistrationClientByRegistrationClientId(int
RegistrationClientId) {
      string query = "SELECT * FROM RegistrationClient WHERE
RegistrationClientId = @RegistrationClientId";

      RegistrationClient oneRegistrationClient = new
RegistrationClient();
      MySqlConnection connection = new MySqlConnection(_ConnString);
```

```
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);

      connection.Open();
      using (MySqlDataReader reader = command.ExecuteReader()) {
        if (reader.Read()) {
          oneRegistrationClient.RegistrationClientId =
Convert.ToInt32(reader["RegistrationClientId"]);
          oneRegistrationClient.EmployeeId =
Convert.ToInt32(reader["EmployeeId"]);
          oneRegistrationClient.ClientId =
Convert.ToInt32(reader["ClientId"]);
          oneRegistrationClient.RoomId =
Convert.ToInt32(reader["RoomId"]);
          oneRegistrationClient.StartDate =
Convert.ToDateTime(reader["StartDate"]);
          oneRegistrationClient.EndDate =
Convert.ToDateTime(reader["EndDate"]);
          oneRegistrationClient.Price =
Convert.ToDouble(reader["Price"]);
          oneRegistrationClient.IsResides =
Convert.ToBoolean(reader["IsResides"]);
        }
      }
      connection.Close();

      return oneRegistrationClient;
    }


    public void UpdateRegistrationClient(int EmployeeId, int ClientId,
int RoomId, DateTime StartDate, DateTime EndDate, double Price, int
RegistrationClientId) {
      string query = "UPDATE RegistrationClient SET EmployeeId =
@EmployeeId, ClientId = @ClientId, RoomId = @RoomId, " +
                     "StartDate = @StartDate, EndDate = @EndDate, Price
= @Price WHERE RegistrationClientId = @RegistrationClientId";

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@EmployeeId", EmployeeId);
      command.Parameters.AddWithValue("@ClientId", ClientId);
      command.Parameters.AddWithValue("@RoomId", RoomId);
      command.Parameters.AddWithValue("@StartDate",
StartDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@EndDate",
EndDate.ToString("yyyy-MM-dd HH:mm:ss"));
      command.Parameters.AddWithValue("@Price", Price);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);

      connection.Open();
```

```
      command.ExecuteNonQuery();
      connection.Close();
    }


    public void UpdateIsResidesStatus(int RegistrationClientId) {
      string query = "UPDATE RegistrationClient SET IsResides = 1 WHERE
RegistrationClientId = @RegistrationClientId";

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);

      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }


    public void DeleteRegistrationClientByRegistrationClientId(int
RegistrationClientId) {
      string query = "DELETE FROM RegistrationClient WHERE
RegistrationClientId = @RegistrationClientId";

      MySqlConnection connection = new MySqlConnection(_ConnString);
      MySqlCommand command = new MySqlCommand(query, connection);
      command.Parameters.AddWithValue("@RegistrationClientId",
RegistrationClientId);

      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
    }

  }
}



public class RegistrationClient {
  private int _Number;
  private int _RegistrationClientId;
  private int _EmployeeId;
  private string _EmployeeFIO;
  private int _ClientId;
  private string _ClientFIO;
  private int _RoomId;
  private string _RoomName;
  private DateTime _StartDate;
  private DateTime _EndDate;
  private double _Price;
  private bool _IsResides;
```

```
private string _Message;

public RegistrationClient() {
  _Number = 0;
  _RegistrationClientId = 0;
  _EmployeeId = 0;
  _EmployeeFIO = String.Empty;
  _ClientId = 0;
  _ClientFIO = String.Empty;
  _RoomId = 0;
  _RoomName = String.Empty;
  _StartDate = new DateTime();
  _EndDate = new DateTime();
  _Price = 0.0;
  _IsResides = false;
  _Message = String.Empty;
}

public int Number {
  set { _Number = value; }
  get { return _Number; }
}
public int RegistrationClientId {
  set { _RegistrationClientId = value; }
  get { return _RegistrationClientId; }
}
public int EmployeeId {
  set { _EmployeeId = value; }
  get { return _EmployeeId; }
}
public string EmployeeFIO {
  set { _EmployeeFIO = value; }
  get { return _EmployeeFIO; }
}
public int ClientId {
  set { _ClientId = value; }
  get { return _ClientId; }
}
public string ClientFIO {
  set { _ClientFIO = value; }
  get { return _ClientFIO; }
}
public int RoomId {
  set { _RoomId = value; }
  get { return _RoomId; }
}
public string RoomName {
  set { _RoomName = value; }
  get { return _RoomName; }
}
public DateTime StartDate {
  set { _StartDate = value; }
  get { return _StartDate; }
```

```
  }
  public DateTime EndDate {
    set { _EndDate = value; }
    get { return _EndDate; }
  }
  public double Price {
    set { _Price = value; }
    get { return _Price; }
  }
  public bool IsResides {
    set { _IsResides = value; }
    get { return _IsResides; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```