НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
комп'ютерних наук
(назва кафедри)

_____ / Голуб Б.Л. /
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна система бронювання в готельно-ресторанному бізнесі»

Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор _____ Руденський Р.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
К.п.н, доцент _____ Волошина Тетяна Володимирівна
(науковий ступінь та вчене звання) (підпис) (ПІБ)

асистент _____ Понзель Ярослав Юрійович
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Іванчук Микола Ігорович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Іванчук Микола Ігорович

Спеціальність 122 – «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система бронювання в готельно-ресторанному бізнесі» затверджена наказом ректора НУБіП України від 16.12.2024 № _2246_С_
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані до роботи: надання інформації про вільні номери в готелях, столики в ресторанах, їхню ціну та час бронювання.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "___"_____ 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ / Волошина Т.В. /
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ / Понзель Я.Ю. /
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____ / Іванчук М.І. /
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Готельно-ресторанна індустрія є одним із найбільш швидкозростаючих секторів світової економіки, щодня обслуговує мільйони клієнтів. У зв’язку зі зростанням попиту на послуги з розміщення та харчування підприємства в цьому секторі повинні ефективно керувати бронюванням, щоб забезпечити безперебійну роботу та високу задоволеність клієнтів. Традиційні методи обробки бронювань, такі як телефонне бронювання, рукописні журнали та електронні таблиці, часто забирають багато часу, схильні до помилок і неефективні під час роботи з великою кількістю клієнтів. Ці застарілі підходи можуть призвести до подвійних бронювань, втрачених бронювань і неправильного використання доступних ресурсів, що негативно впливає як на досвід клієнтів, так і на прибутковість бізнесу. В епоху цифрової трансформації впровадження автоматизованих систем бронювання стало необхідним для того, щоб сучасний готельний бізнес залишався конкурентоспроможним.

Інформаційна система для бронювання в готельно-ресторанному бізнесі забезпечує комплексне вирішення цих проблем, пропонуючи керування бронюванням у режимі реального часу, автоматизоване відстеження доступності та централізоване зберігання даних. Така система не

тільки підвищує операційну ефективність, але й покращує точність записів про бронювання, зменшує людські помилки та забезпечує кращий досвід як для клієнтів, так і для персоналу [3].

Актуальність розробки

Зростаюча залежність від цифрових рішень у різних галузях, включаючи гостинність, підкреслює необхідність автоматизованої системи бронювання. Оскільки очікування клієнтів постійно зростають, компанії повинні надавати швидкі, ефективні та зручні варіанти бронювання, щоб залишатися конкурентоспроможними. Сучасні клієнти віддають перевагу онлайн-бронюванню перед традиційним телефонним дзвінком або особистим бронюванням через зручність, гнучкість і прозорість, які він пропонує. Ефективна система бронювання повинна дозволяти користувачам перевіряти наявність, миттєво робити бронювання та отримувати підтвердження без затримок.

Основною **метою** цього дослідження є покращення існуючої індустрії бронювання розробивши інформаційну систему бронювання в готельно-ресторанному бізнесі, яка забезпечує зручне та ефективне рішення для керування бронюваннями. Система дозволить менеджерам готелів і ресторанів реєструвати клієнтів, бронювати номери в готелях і столики в ресторанах, відстежувати наявність місць і керувати бронюванням в організований спосіб. Реалізована за допомогою PHP із фреймворком Symfony і MySQL, система забезпечить безпечну обробку даних, оновлення в реальному часі та автоматизоване ведення записів.

Досягнувши цих цілей, система сприятиме підвищенню ефективності індустрії гостинності, зниженню операційних витрат, підвищенню задоволеності клієнтів і підвищенню загальної ефективності бізнесу. Розроблене рішення стане основою для подальших удосконалень, включаючи інтеграцію з платіжними системами, розробку мобільних додатків і прогнозування попиту на основі штучного інтелекту.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Ефективне управління бронюванням є фундаментальним аспектом готельної та ресторанної індустрії, що забезпечує безперебійну роботу та високу задоволеність клієнтів. Однак багато підприємств все ще покладаються на застарілі методи, такі як ручне ведення записів або базові електронні таблиці, що часто призводить до неефективності та помилок. Відсутність централізованої автоматизованої системи призводить до таких проблем, як подвійне бронювання, втрата бронювань і непорозуміння між співробітниками. Ці виклики не тільки порушують діяльність, але й негативно впливають на репутацію та прибутковість бізнесу в секторі гостинності.

Традиційні методи бронювання мають значні недоліки, насамперед через їхню залежність від людського впливу. Імовірність помилок зростає, коли бронювання записуються вручну, що ускладнює відстеження наявності в реальному часі. Співробітникам часто важко керувати кількома бронюваннями одночасно, що призводить до ситуацій, коли кімнати або столики помилково призначаються кільком клієнтам. Крім того, ручним системам не вистачає масштабованості, що робить їх непридатними для підприємств, які розвиваються або мають справу з великим обсягом резервувань. У галузі, де ефективність і точність є ключовими, обмеження цих застарілих методів перешкоджають загальній продуктивності та якості послуг.

Однією з найактуальніших проблем в управлінні бронюванням є надмірне бронювання, яке виникає, коли підприємство приймає більше бронювань, ніж дозволяє його фактична кількість. Ця проблема виникає через

затримки в оновленні записів бронювання, поганий зв’язок між відділами або відсутність синхронізації в реальному часі між кількома каналами бронювання. Надмірне бронювання не тільки призводить до незадоволення клієнтів, але й змушує компанії пропонувати компенсацію, відшкодування або альтернативне розміщення, що призводить до фінансових втрат і шкоди репутації бренду [3].

Очікування клієнтів також змінилися з технологічним прогресом, і сучасні споживачі віддають перевагу швидким, безпроблемним процесам бронювання. Відсутність ефективної системи бронювання часто призводить до тривалого очікування, плутанини та розчарування. Клієнти, які відчувають труднощі із забезпеченням бронювання, можуть вибрати конкурентів, які пропонують більш плавний і прозорий процес. Крім того, підприємства, які покладаються на ручне відстеження бронювання, ризикують втратити дані клієнтів, що ускладнює надання персоналізованих послуг або побудову довгострокових відносин зі своїми клієнтами. У галузі, яка керується досвідом клієнтів, невиконання цих очікувань може мати тривалі наслідки.

Хоча доступно кілька комерційних платформ бронювання, вони мають власний набір обмежень. Багато існуючих рішень, таких як OpenTable або Booking.com, вимагають від компаній сплачувати високу плату за підписку, що може бути фінансово невигідним для малих і середніх підприємств. Крім того, системам сторонніх виробників часто бракує гнучкості, що не дозволяє підприємствам адаптувати функції до своїх конкретних операційних потреб. Покладання на зовнішні платформи також означає, що підприємства мають обмежений контроль над своїми даними, що викликає занепокоєння щодо безпеки та довгострокової стабільності. Враховуючи ці недоліки, існує очевидна потреба в адаптованій, економічно ефективній альтернативі, яка б обслуговувала саме готелі та ресторани.

Розробка спеціальної інформаційної системи для управління бронюванням вирішує ці проблеми шляхом автоматизації ключових процесів і зниження ризику людської помилки. Добре розроблена система забезпечує

оновлення доступності в реальному часі, гарантуючи мінімізацію подвійних бронювань і конфліктів розкладу. Це також підвищує зручність клієнтів, дозволяючи онлайн-бронювання з миттєвим підтвердженням, усуваючи невизначеність, часто пов’язану з традиційними методами бронювання. Завдяки інтеграції таких функцій, як керування клієнтами, історія бронювань і аналітика, сучасна система бронювання дає можливість компаніям оптимізувати свою діяльність і покращити процес прийняття рішень.

Щоб задовольнити ці вимоги галузі, розробляється надійна та масштабована інформаційна система з використанням PHP із структурою Symfony та MySQL. Це рішення надасть готелям і ресторанам інтуїтивно зрозумілу платформу для ефективного керування бронюванням, покращуючи як задоволеність клієнтів, так і ефективність роботи. Спрощуючи процес бронювання, зменшуючи адміністративне навантаження та забезпечуючи точність даних, запропонована система має на меті змінити те, як компанії обробляють бронювання. У умовах дедалі більшої конкуренції на ринку гостинності впровадження такої системи має важливе значення для довгострокового успіху та зростання.

Готельний і ресторанний бізнес відіграє вирішальну роль у сфері послуг, надаючи послуги розміщення та харчування мільйонам клієнтів у всьому світі. Ефективне керування резервуванням має важливе значення для забезпечення безперебійної роботи, оптимізації використання ресурсів і підвищення якості обслуговування клієнтів. Із зростаючим попитом на цифрову трансформацію в сфері гостинності компанії все більше покладаються на автоматизовані системи для обробки бронювань, оптимізації зв’язку та зменшення адміністративного навантаження [4].

У традиційних умовах бронюваннями часто керують вручну за допомогою телефонних дзвінків, електронних листів або паперових записів. Цей підхід схильний до людських помилок, неправильного спілкування та неефективності, що призводить до надмірного бронювання, втрати бронювань і незадоволених клієнтів. Відсутність централізованої системи ускладнює для

персоналу відстеження наявності, керування вподобаннями клієнтів та ефективну обробку змін або скасувань. У пікові сезони проблеми стають ще більш помітними, оскільки підприємствам важко встигати за великим обсягом бронювань.

Поява цифрових систем бронювання значно покращила роботу готелів і ресторанів. Сучасні платформи бронювання дозволяють компаніям керувати бронюваннями в режимі реального часу, надаючи миттєві оновлення про наявність номерів і столиків. Клієнти отримують вигоду від зручності онлайн-бронювання, яке дає їм змогу перевірити наявність, зробити бронювання та отримати підтвердження без прямої взаємодії з персоналом. Крім того, автоматизовані системи допомагають компаніям вести точні записи, аналізувати тенденції бронювання та персоналізувати послуги на основі історії клієнтів.

Добре розроблена система бронювання об’єднує різні функціональні можливості, включаючи реєстрацію клієнтів, керування бронюванням, обробку платежів та інструменти звітності. Для готелів система повинна враховувати різні типи номерів, структури ціноутворення та сезонні тарифи, а також обробляти процеси реєстрації та виїзду. У ресторанах функції керування столиками повинні дозволяти персоналу ефективно розподіляти місця для сидіння, мінімізувати час очікування та оптимізувати використання простору. Здатність безпечно керувати клієнтськими даними також має вирішальне значення, оскільки компаніям необхідно зберігати контактні дані, спеціальні запити та інформацію про програму лояльності, дотримуючись правил захисту даних.

Ця система, розроблена з використанням PHP із структурою Symfony та MySQL, запропонує комплексну платформу для ефективного керування бронюваннями. Завдяки інтеграції оновлень бронювання в режимі реального часу, автоматизованих підтверджень і зручного інтерфейсу система оптимізує процес бронювання як для компаній, так і для клієнтів. Впровадження такої системи дозволить модернізувати традиційні практики бронювання,

забезпечивши більшу точність, зручність і надійність управління готелями та ресторанами.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Клієнт	Ідентифікатор	Унікальний номер клієнта
	Ім'я	ПІБ клієнта
	Телефон	Контактний номер телефону
	Електронна пошта	Адреса електронної пошти клієнта
Готельний номер	Ідентифікатор	Унікальний номер кімнати
	Номер кімнати	Позначення кімнати в готелі
	Назва	Тип кімнати (наприклад, "Люкс", "Стандарт")
	Опис	Деталі та особливості кімнати
	Кількість місць	Скільки людей може розміститися в номері
Бронювання готельного номера	Ідентифікатор	Унікальний номер бронювання
	Клієнт	Хто забронював номер
	Номер кімнати	Яку саме кімнату заброньовано
	Дата заїзду	День, коли клієнт заселяється
	Дата виїзду	День, коли клієнт виїжджає
	Вартість	Вартість бронювання за добу

1.2 Огляд існуючих рішень

Індустрія гостинності стала свідком значних трансформацій із появою технологій, що призвело до розвитку різноманітних систем бронювання, спрямованих на підвищення ефективності та задоволення клієнтів. У цьому розділі розглядаються деякі з найвідоміших існуючих рішень для бронювання в готельному та ресторанному бізнесі, висвітлюються їхні особливості, переваги та обмеження.

Booking.com є одним із провідних онлайн-туристичних агентств (OTA) у всьому світі, зосереджуючись на полегшенні пошуку та бронюванні готелів, апартаментів, курортів тощо. Заснована в 1996 році в Амстердамі,

Нідерланди, Booking.com значно розширилася, щоб стати видатним гравцем у туристичній індустрії, пропонуючи широкий вибір об’єктів у багатьох напрямках по всьому світу [1].

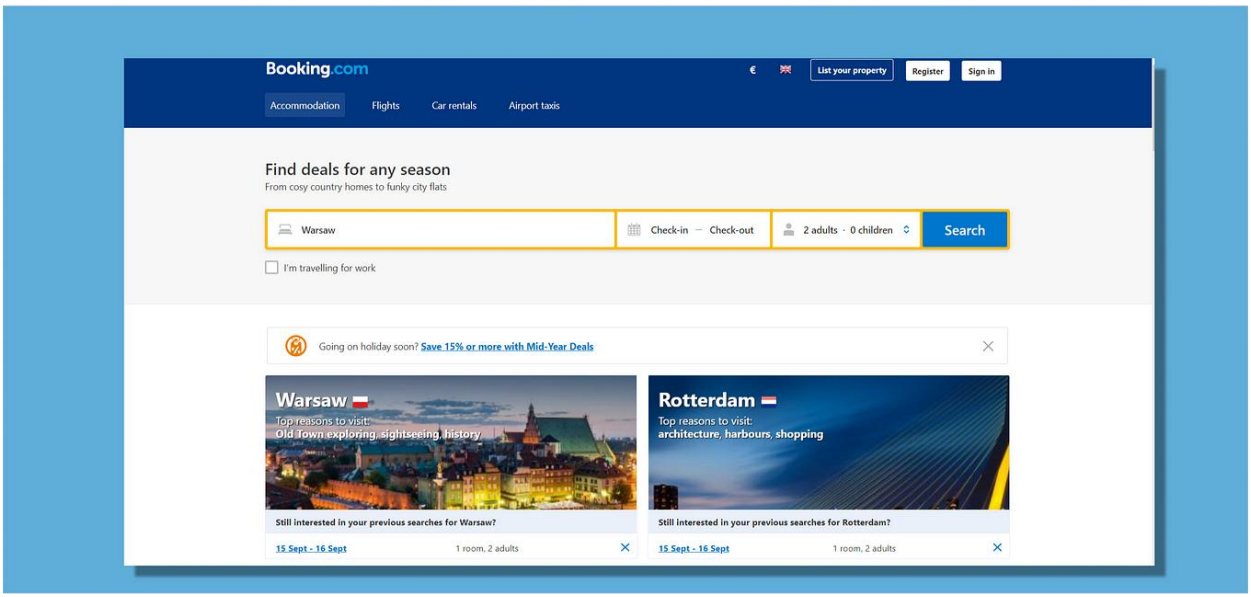

Рис. 1 Інформаційна система “Booking”

Платформа надає повний набір функцій, призначених для покращення взаємодії з користувачем під час пошуку та бронювання житла. Однією з його ключових сильних сторін є величезна база даних, яка включає мільйони списків, які задовольняють широкий спектр уподобань і бюджетів мандрівників. Ця різноманітність дозволяє користувачам знаходити варіанти, які відповідають їхнім потребам, незалежно від того, чи шукають вони бюджетне житло чи розкішний відпочинок.

Booking.com відомий своїм зручним інтерфейсом, який спрощує процес пошуку житла. Клієнти можуть легко фільтрувати результати за різними критеріями, включаючи ціну, розташування, зручності та оцінки гостей. Ця інтуїтивно зрозуміла навігація допомагає мандрівникам швидко визначати відповідні варіанти, роблячи процес бронювання ефективним і простим. Крім того, платформа пропонує оновлення доступності в реальному часі, гарантуючи, що користувачі можуть бачити, які об’єкти доступні на бажані дати подорожі. Ця функція мінімізує ризик подвійних бронювань і підвищує ймовірність забезпечення бронювання.

Включення відгуків і рейтингів клієнтів є ще одним важливим аспектом Booking.com. Платформа дозволяє перевіреним гостям ділитися своїм досвідом, надаючи цінну інформацію потенційним клієнтам. Цей елемент соціального доказу допомагає зміцнити довіру та надійність, заохочуючи користувачів приймати обґрунтовані рішення при виборі житла. Крім того, багато оголошень на Booking.com мають гнучку політику скасування, що дозволяє мандрівникам змінювати свої плани без значних комісій. Ця гнучкість особливо приваблива для тих, кому може знадобитися скорегувати свої маршрути через непередбачені обставини.

Booking.com також пропонує мобільний додаток, який дозволяє користувачам шукати житло, робити бронювання та керувати своїми бронюваннями навіть у дорозі. Ця додаткова зручність покращує загальну взаємодію з користувачем, полегшуючи мандрівникам захистити своє житло. Крім того, програма лояльності Genius нагороджує постійних користувачів знижками та перевагами, сприяючи утриманню клієнтів і заохочуючи повторні бронювання [1].

Попри численні переваги, Booking.com не позбавлений обмежень. Одним із помітних недоліків є високі комісії, що стягуються з власників нерухомості, що може суттєво вплинути на прибутки готелів та інших закладів розміщення. Ці витрати можуть змусити деяких власників нерухомості підняти ціни, що зрештою вплине на споживачів. Крім того, платформа забезпечує обмежений контроль для постачальників розміщення щодо відносин із клієнтами та даних, що може перешкоджати прямим маркетинговим зусиллям та персоналізованому взаємодії з гостями.

Інша потенційна проблема полягає в мінливості політики скасування. У той час як багато списків пропонують гнучкі варіанти, інші можуть мати суворіші умови, які можуть заплутати мандрівників. Тому важливо, щоб клієнти уважно ознайомилися з положеннями та умовами, пов’язаними з кожним бронюванням, перш ніж підтверджувати своє бронювання.

Підсумовуючи, Booking.com змінив спосіб, у який мандрівники шукають і бронюють житло, пропонуючи широкий спектр варіантів, зручний інтерфейс і функції, які задовольняють потреби мільйонів користувачів у всьому світі. Однак залежність від комісійних зборів і обмежений контроль, який відчувають власники нерухомості, підкреслюють деякі проблеми в платформі. У міру розвитку туристичної індустрії Booking.com продовжує адаптуватися до мінливих уподобань споживачів і технологічного прогресу, зберігаючи свої позиції ключового гравця на ринку.

Розглянемо інше програмне рішення на ринку.

OpenTable — це відома платформа онлайн-бронювання, спеціально розроблена для ресторанів, що дозволяє клієнтам легко та ефективно бронювати столики. Сервіс, запущений у 1998 році, став значним гравцем у сфері харчування, з’єднавши мільйони відвідувачів із ресторанами по всьому світу. Завдяки зручному інтерфейсу та надійним функціям OpenTable революціонізував спосіб бронювання місць у столових [2].

Однією з ключових переваг OpenTable є розгалужена мережа ресторанів-учасників. Платформа дозволяє користувачам шукати заклади харчування за різними критеріями, включаючи розташування, тип кухні, ціновий діапазон і оцінки клієнтів. Ця обширна база даних дозволяє відвідувачам легко знаходити відповідні ресторани, які відповідають їхнім уподобанням і вимогам. Користувачі можуть переглядати список доступних столиків у режимі реального часу, гарантуючи, що вони можуть забезпечити бронювання відповідно до їхнього розкладу.

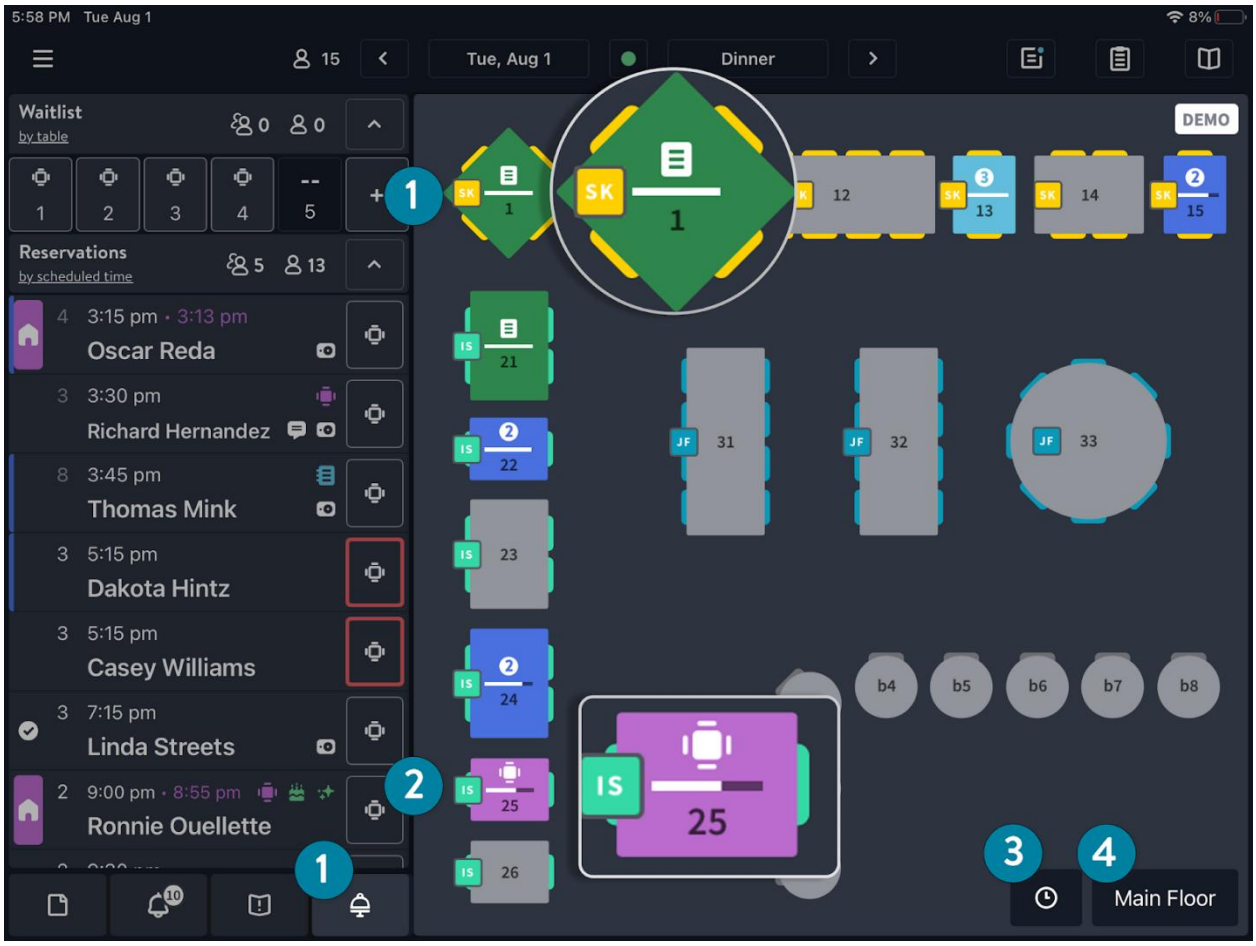

Рис. 2 Інформаційна система “OpenTable”

Взаємодія з користувачем OpenTable розроблена так, щоб бути бездоганною та інтуїтивно зрозумілою. Гості можуть швидко вибрати бажаний ресторан, вибрати дату та час бронювання та вказати кількість гостей. Після підтвердження бронювання клієнти негайно отримують підтвердження електронною поштою, що забезпечує спокій і знижує ймовірність помилок бронювання. Крім того, платформа дозволяє користувачам легко змінювати або скасовувати свої бронювання, підвищуючи зручність для клієнтів, чиї плани можуть змінитися.

Для ресторанів OpenTable пропонує комплексний набір інструментів керування для оптимізації операцій. Платформа містить функції для керування бронюванням, оптимізації оборотності столиків і відстеження вподобань клієнтів. Персонал ресторану має доступ до докладних звітів і аналітики, що дає їм змогу приймати керовані даними рішення, які покращують якість обслуговування та покращують загальний досвід обіду. Система також надає

можливість керувати списками очікування та спеціальними запитами, гарантуючи, що ресторани можуть ефективно задовольняти потреби своїх гостей [2].

Мобільний додаток OpenTable ще більше покращує враження від обіду, дозволяючи користувачам робити замовлення на ходу. Гості можуть шукати ресторани, переглядати вільні столики та керувати своїми бронюваннями безпосередньо зі своїх смартфонів. Ця доступність задовольняє зростаючий попит на зручність у сучасному ресторанному середовищі.

Незважаючи на численні переваги, OpenTable має деякі обмеження. Плата за підписку та послуги, пов’язані з використанням платформи, можуть викликати занепокоєння для невеликих ресторанів, оскільки ці витрати можуть вплинути на їх загальну прибутковість. Крім того, хоча OpenTable надає цінні дані про клієнтів, ресторани можуть залежати від платформи для керування бронюваннями, що потенційно обмежує їхню здатність будувати прямі відносини зі своїми відвідувачами.

Іншим потенційним недоліком є мінливість якості обслуговування в ресторанах-учасниках. Незважаючи на те, що OpenTable прагне об’єднати ресторани з авторитетними закладами, досвід клієнтів може значно відрізнятися від одного ресторану до іншого. Ця невідповідність може вплинути на сприйняття відвідувачами платформи в цілому.

Підсумовуючи, OpenTable суттєво змінив процес бронювання ресторанів, запропонувавши зручну платформу, яка з’єднує відвідувачів із різноманітними ресторанами. Його широкі можливості приносять користь як клієнтам, так і операторам ресторанів, спрощуючи процес бронювання та покращуючи загальні враження від обіду. Однак пов’язані з платформою витрати та різноманітність якості послуг підкреслюють проблеми, з якими стикаються невеликі установи. Оскільки індустрія харчування продовжує розвиватися, OpenTable залишається ключовим гравцем, адаптуючись до мінливих очікувань споживачів і технологічного прогресу.

1.3 Постановка завдання

Керівник відділу бронювання відповідає за ведення точного обліку ключових показників, пов’язаних із бронюваннями та взаємодією з клієнтами, таких як кількість зроблених бронювань, відсоток скасувань, відгуки гостей та рівень заповнюваності. Щоб ефективно керувати цією інформацією, програма вимагає створення повної бази даних, яка фіксує всі важливі дані, необхідні для створення персоналізованих сповіщень і інформації про ефективність системи бронювання.

Основна програмна система, що розробляється, дозволить менеджерам готелів і ресторанів швидко отримувати доступ до важливої інформації щодо:

- ефективність стратегій бронювання та рекламних кампаній;
- точність прогнозів заповнюваності та доступності таблиць;
- загальна кількість бронювань, здійснених протягом певного періоду часу.

Програма працюватиме в діалоговому режимі на основі зручного інтерфейсу меню. Це налаштування дозволяє користувачам легко переміщатися між різними параметрами. Крім того, система дозволить менеджерам переглядати та змінювати існуючі бронювання, забезпечуючи гнучкість для скасування або коригування зареєстрованих бронювань, якщо це необхідно.

Завдяки впровадженню цієї інформаційної системи відділ бронювання підвищить свою операційну ефективність, покращить задоволеність клієнтів завдяки кращому управлінню бронюваннями та, зрештою, підвищить заповнюваність і дохід для готельного та ресторанного бізнесу.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. система повинна дозволяти користувачам (клієнтам і персоналу) створювати облікові записи, входити в систему та безпечно виходити з системи. Ця функція повинна містити параметри відновлення пароля;
2. система повинна дозволяти персоналу додавати, оновлювати та видаляти інформацію про клієнта, включаючи контактні дані, уподобання та історію бронювань;
3. користувачі повинні мати можливість створювати, переглядати, змінювати та скасовувати бронювання готельних номерів і столиків у ресторанах. Система також повинна обробляти групові бронювання та спеціальні запити;
4. система повинна забезпечувати вільні номери в готелі та столики в ресторані в режимі реального часу на основі обраної дати та часу. Користувачів слід негайно повідомити, якщо бажане бронювання не може бути виконано;
5. система повинна інтегруватися з безпечними платіжними шлюзами, щоб полегшити онлайн-платежі за бронювання. Він повинен працювати з різними способами оплати, включаючи кредитні/дебетові картки та цифрові гаманці;
6. користувачі повинні отримувати автоматичні сповіщення (електронною поштою або SMS) щодо підтвердження бронювання, нагадування, зміни та скасування;
7. система повинна генерувати звіти про статистику бронювань, рівень заповнюваності та відгуки клієнтів, що дозволяє керівництву аналізувати ефективність і приймати рішення на основі даних;
8. необхідно надати адміністративний інтерфейс для персоналу, щоб керувати бронюваннями, контролювати продуктивність системи та отримувати доступ до аналітики.

Нефункціональні вимоги:

1. система повинна відповідати на запити користувачів протягом двох секунд для виконання стандартних операцій (наприклад, пошуку наявності, підтвердження бронювання), щоб забезпечити безперебійну роботу користувача;
2. система повинна бути розроблена таким чином, щоб обслуговувати зростаючу кількість користувачів і бронювань без шкоди для продуктивності. Він має сприяти зростанню в міру розширення готелю чи ресторану;
3. система повинна впроваджувати надійні заходи безпеки для захисту конфіденційних даних користувача, включаючи шифрування паролів і платіжної інформації. Дотримання норм захисту даних (наприклад, GDPR) має важливе значення;
4. інтерфейс користувача має бути інтуїтивно зрозумілим і простим у навігації для всіх користувачів, у тому числі тих, хто має мінімальні технічні знання. Повинні бути доступні вичерпні посібники користувача та підтримка;
5. система має працювати 24/7 з цільовим часом безвідмовної роботи 99,5%. Про планове технічне обслуговування необхідно повідомляти користувачів заздалегідь, щоб мінімізувати збої;
6. система повинна бути сумісна з різними пристроями та операційними системами, включаючи настільні ПК, планшети та смартфони, щоб забезпечити доступність для всіх користувачів;
7. система має бути розроблена для легкого обслуговування та оновлення, дозволяючи розробникам вносити зміни та виправляти помилки з мінімальними простоями;
8. необхідно регулярно створювати резервні копії даних, щоб запобігти втраті даних, і має бути розроблений план відновлення, щоб відновити функціональність системи у разі збою.

Завдяки визначенню цих функціональних і нефункціональних вимог інформаційна система бронювання в готельно-ресторанному бізнесі буде краще обладнана для задоволення потреб користувачів, підвищення ефективності роботи та забезпечення позитивного досвіду як для клієнтів, так і для персоналу.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) інформаційної системи для бронювання в готельно-ресторанному бізнесі життєво важливий для забезпечення позитивного досвіду користувача. Добре розроблений інтерфейс має бути інтуїтивно зрозумілим, візуально привабливим і функціональним, дозволяючи користувачам легко орієнтуватися в системі. Щоб досягти цього, під час розробки необхідно враховувати кілька ключових аспектів.

Перш за все, дизайн має бути зручним для користувача. Важлива інтуїтивно зрозуміла структура навігації, що дозволяє користувачам легко переміщатися між різними розділами, такими як бронювання, профілі користувачів і звіти. Підтримка узгодженого макета на всіх сторінках покращує загальний досвід, оскільки користувачі знайомляться з елементами дизайну. Крім того, впровадження надійної функції пошуку дозволить користувачам швидко знаходити вільні готельні номери та столики в ресторанах за допомогою простих фільтрів, таких як дата, час і кількість гостей [5].

Доступність є ще одним важливим моментом. Інтерфейс має бути чуйним, плавно адаптуватися до різних розмірів екрана, включаючи настільні комп’ютери, планшети та смартфони, щоб забезпечити безпроблемний доступ до системи для всіх користувачів. Підтримка клавіатурної навігації є важливою для користувачів з обмеженими можливостями, дозволяючи їм отримувати доступ до функцій, не покладаючись лише на мишу. Крім того, інтерфейс повинен забезпечувати достатній колірний контраст і регульований

розмір шрифту для покращення читабельності, особливо для людей з вадами зору.

Візуальні елементи відіграють значну роль в ефективності інтерфейсу користувача. Чіткі кнопки із закликом до дії, такі як «Забронювати зараз» або «Скасувати бронювання», мають бути помітні з чіткими мітками, щоб ефективно орієнтувати користувачів. Продумане використання інтуїтивно зрозумілих піктограм і графіки може ще більше покращити розуміння та взаємодію, допомагаючи користувачам швидко визначати функції. Механізми негайного зворотного зв’язку також важливі, оскільки вони інформують користувачів про успішність чи невдачу їхніх дій, наприклад підтвердження бронювання чи сповіщення про помилку.

Відображення інформації є ще одним важливим аспектом інтерфейсу користувача. Добре організована інформаційна панель повинна надавати користувачам швидкий огляд ключової інформації, такої як майбутні бронювання, рівень заповнюваності та відгуки клієнтів. Переглядаючи деталі бронювання, користувачі повинні мати доступ до вичерпної інформації, включаючи імена гостей, дати, час, спеціальні запити та статуси платежів. Крім того, система повинна візуально відображати в режимі реального часу доступність готельних номерів і столиків у ресторанах, дозволяючи користувачам одразу бачити їхні варіанти.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) — це стандартизована мова моделювання, яка використовується для візуалізації, визначення, конструювання та документування артефактів програмної системи. Він надає набір методів графічної нотації для створення абстрактних моделей конкретних систем, полегшуючи спілкування між зацікавленими сторонами, включаючи розробників, дизайнерів і бізнес-аналітиків. UML широко поширений у розробці програмного забезпечення завдяки своїй здатності представляти складні системи в чіткій і зрозумілій формі [6].

UML охоплює різні типи діаграм, які можна загалом класифікувати на дві категорії: структурні діаграми та поведінкові діаграми. Структурні діаграми зосереджені на статичних аспектах системи, що представляють компоненти та їхні взаємозв’язки. Навпаки, діаграми поведінки відображають динамічні аспекти, ілюструючи, як система поводиться у відповідь на різні події.

Серед ключових структурних діаграм діаграма класів є, мабуть, найвидатнішою. Він представляє класи системи, їхні атрибути, методи та зв’язки між ними. Діаграми класів відіграють важливу роль в об’єктно-орієнтованому проєктуванні, оскільки вони допомагають визначити архітектуру системи та визначити, як об’єкти будуть взаємодіяти.

Іншою важливою структурною схемою є діаграма компонентів, яка зображує організацію та залежності між компонентами програмного забезпечення. Ця діаграма допомагає візуалізувати структуру високого рівня системи, полегшуючи розуміння того, як різні компоненти поєднуються між собою.

З боку поведінки діаграма варіантів використання широко використовується для представлення функціональних вимог системи. Він

визначає різних суб’єктів (користувачів чи інших систем), які взаємодіють із системою, і описує варіанти використання (конкретні функції), які вони можуть виконувати. Діаграми варіантів використання є цінними для розуміння взаємодії користувачів і забезпечення врахування всіх необхідних функцій у проєкті.

Діаграма послідовності — це ще одна важлива діаграма поведінки, яка ілюструє, як об’єкти взаємодіють у певному сценарії з часом. Він показує послідовність повідомлень, якими обмінюються об’єкти, надаючи розуміння потоку керування та даних у системі. Діаграми послідовності особливо корисні для деталізації логіки складних операцій і випадків використання [7].

Діаграми діяльності також заслуговують на увагу, оскільки вони представляють робочий процес системи, ілюструючи послідовність дій і рішень, залучених до процесу. Ці діаграми корисні для моделювання бізнес-процесів і розуміння потоку контролю в системі.

UML не прив’язаний до жодної конкретної мови програмування чи методології розробки, що робить його універсальним і застосовним у різних контекстах розробки програмного забезпечення. Його стандартизована нотація дозволяє чітко спілкуватися між членами команди та зацікавленими сторонами, зменшуючи непорозуміння та полегшуючи співпрацю.

Таким чином, UML служить потужним інструментом для розробки програмного забезпечення, забезпечуючи комплексну структуру для моделювання систем за допомогою різноманітних діаграм. Пропонуючи візуальне представлення як структурних, так і поведінкових аспектів, UML допомагає командам ефективніше розробляти та впроваджувати програмні рішення. Його широке визнання в галузі підкреслює його важливість у сфері розробки програмного забезпечення.

Діаграма — це візуальне зображення, призначене для чіткої та спрощеної передачі інформації, концепцій або процесів. За допомогою різноманітних графічних елементів, таких як форми, лінії, символи та кольори, діаграми ілюструють зв’язки, ієрархії та робочі процеси, полегшуючи

розуміння складних ідей. Вони є цінними інструментами для спілкування, аналізу та вирішення проблем у багатьох галузях, включаючи математику, інженерію, бізнес і розробку програмного забезпечення.

У розробці програмного забезпечення діаграми відіграють вирішальну роль у моделюванні систем, дозволяючи зацікавленим сторонам візуалізувати архітектуру, компоненти та поведінку програм. Ці візуальні представлення допомагають роз'яснити вимоги, полегшити обговорення та документувати дизайнерські рішення, сприяючи спільному розумінню серед членів команди.

Діаграми можна розділити на кілька категорій залежно від їхнього призначення та типу інформації, яку вони передають. Структурні діаграми представляють статичні аспекти системи, зосереджуючись на її компонентах та їхніх взаємозв'язках. Наприклад, діаграми класів відображають класи в об'єктно-орієнтованій системі, деталізуючи їхні атрибути, методи та зв'язки. Діаграми компонентів ілюструють організацію та залежності між програмними компонентами, тоді як діаграми розгортання показують фізичне розгортання артефактів на вузлах, таких як сервери чи пристрої [8].

2.2 Об'єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це тип діаграми поведінки в уніфікованій мові моделювання (UML), яка візуально представляє взаємодію між користувачами (або «акторами») і системою. Він забезпечує загальний огляд функціональних вимог системи, ілюструючи різні випадки використання, які є конкретними сценаріями, за якими користувачі взаємодіють із системою для досягнення певних цілей. Діаграми варіантів використання є важливим інструментом для розуміння потреб користувачів і забезпечення того, що система розроблена для ефективного задоволення цих потреб [9].

За своєю суттю діаграма варіантів використання складається з кількох ключових компонентів, включаючи акторів, варіанти використання

та зв’язки між ними. Актори представляють зовнішні сутності, які взаємодіють із системою, до яких можуть входити користувачі, інші системи або пристрої. Кожен актор на схемі зображений у вигляді фігурки або значка.

Варіанти використання, представлені овалами, описують конкретні функції або дії, які система виконує у відповідь на запит актора. Кожен варіант використання фіксує окрему мету, яку прагне досягти учасник, наприклад «Забронювати номер у готелі», «Зробити бронювання» або «Провести платіж». Окреслюючи ці випадки використання, діаграма допомагає з’ясувати, що має робити система та як вона має реагувати на різні взаємодії користувача.

Відносини між акторами та варіантами використання показані лініями, що їх з’єднують. Суцільна лінія вказує на те, що актор бере участь у конкретному випадку використання, ілюструючи пряму взаємодію між ними. Крім того, включайте різні типи зв’язків, наприклад включення та розширення. Відношення включення вказує на те, що варіант використання є обов’язковою частиною іншого варіанту використання, тоді як зв’язок розширення показує, що варіант використання може додати необов’язкову поведінку до іншого варіанту використання за певних умов.

Однією з головних переваг діаграм варіантів використання є їх здатність надавати чіткий і стислий огляд функціональності системи. Вони служать цінним інструментом спілкування між зацікавленими сторонами, включаючи розробників, бізнес-аналітиків і кінцевих користувачів, гарантуючи, що всі мають спільне розуміння можливостей системи. Ця чіткість допомагає виявити потенційні прогалини у функціональності та

гарантує, що критичні випадки використання не будуть упущені під час процесу проєктування та розробки.

Розроблена діаграма прецедентів використання представлена на рис. 3.

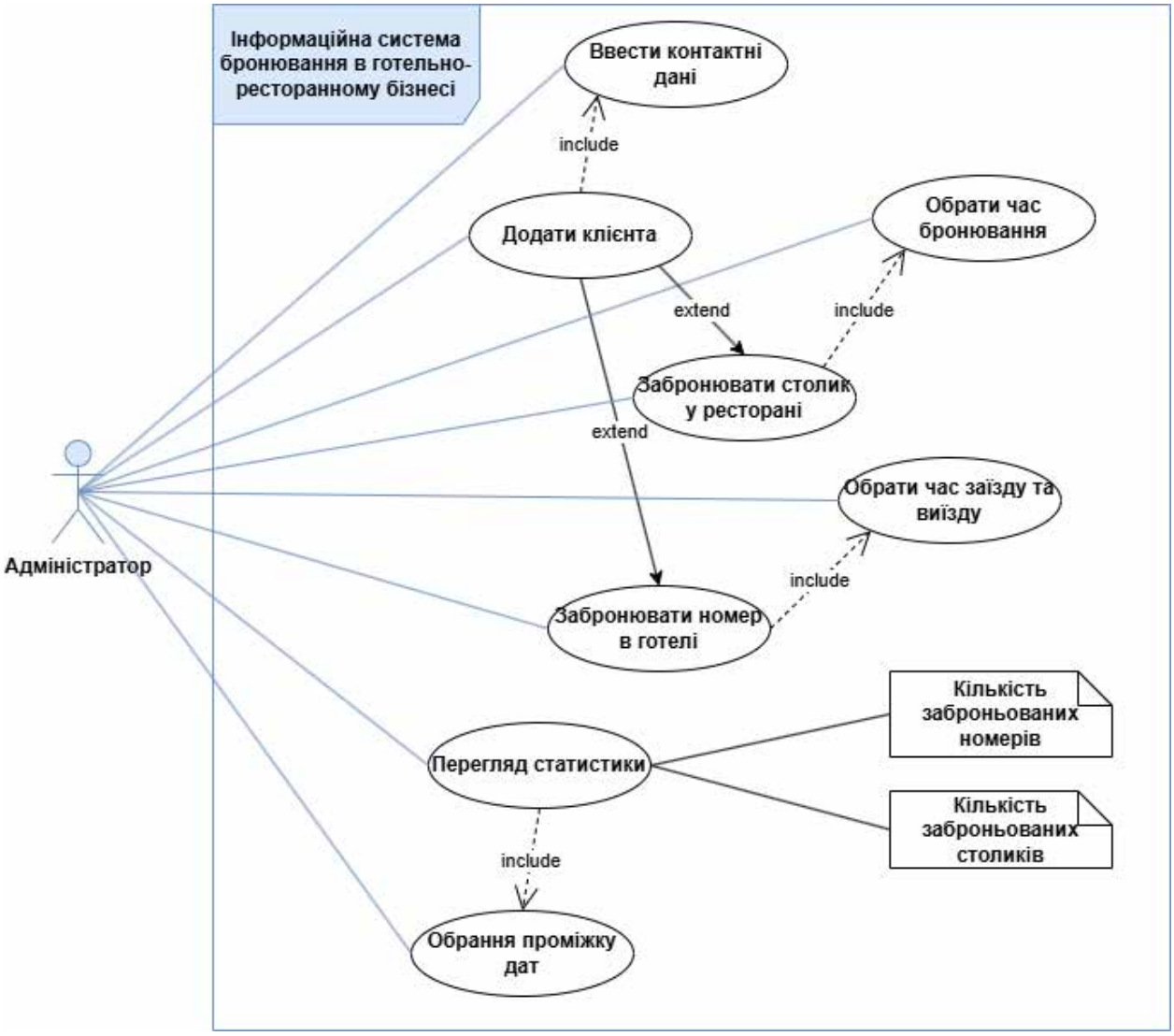

Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Адміністратор”.

Актор «Адміністратор» включає такі прецеденти:

- додати клієнта;
- забронювати столик у ресторані;
- ввести контактні дані;
- обрати час бронювання;

- обрати час заїзду та виїзду;
- забронювати номер в готелі;
- перегляд статистики;
- обрання проміжку дат.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Назва випадку використання: Забронювати номер у готелі

Актор: Адміністратор

Передумови: Адміністратор авторизований в інформаційній системі. У готелі є вільні номери на обрані дати.

Основний потік:

1. адміністратор переходить до розділу «Керування бронюванням» системи та вибирає опцію «Забронювати номер»;
2. система пропонує адміністратору ввести або вибрати дані гостя;
3. система перевіряє наявність номерів за вказаними датами та уподобаннями. Якщо доступний, відображається список відповідних кімнат;
4. адміністратор вибирає потрібну кімнату з доступних варіантів. Система може запропонувати адміністратору підтвердити вибір і відобразити відповідну інформацію про кімнату;
5. адміністратор переглядає інформацію про бронювання, включаючи інформацію про гостя, вибраний номер і загальну

вартість. Система дозволяє вносити будь-які необхідні налаштування до остаточного підтвердження;

6. адміністратор підтверджує бронювання натисканням кнопки «Завершити бронювання». Система обробляє запит і створює новий запис бронювання в базі даних;
7. система оновлює статус доступності забронюваного номера, щоб відобразити, що він більше не доступний на вибрані дати;
8. повідомлення про підтвердження генерується та надсилається гостю через надану контактну інформацію, підтверджуючи деталі бронювання;
9. система реєструє дії бронювання в журналі активності адміністратора для подальшого використання та аудиту.

Альтернативні потоки:

1. якщо на вказані дати немає вільних номерів, система сповіщає про це адміністратора та пропонує вибрати інші дати або запропонувати альтернативне розміщення;
2. якщо виникають проблеми з платіжною інформацією гостя, система сповіщає адміністратора та дозволяє йому повторно ввести або оновити платіжні дані перед завершенням бронювання.

Цей сценарій описує процес бронювання адміністратором номера в готелі за допомогою інформаційної системи. Виконуючи ці кроки, адміністратор може ефективно керувати бронюваннями, забезпечувати точні записи та забезпечувати позитивний досвід для гостей. Цей варіант використання підкреслює важливість ефективного спілкування між адміністратором і системою для спрощення процесу бронювання.

Розглянемо ще один сценарій використання.

Назва випадку використання: Забронювати столик у ресторані

Актор: Адміністратор

Передумови: Адміністратор авторизований в інформаційній системі. У ресторані є вільні столики на обрану дату та час.

Основний потік:

1. адміністратор переходить до розділу «Керування бронюванням» системи та вибирає опцію «Забронювати столик»;
2. система пропонує адміністратору ввести або вибрати дані гостя;
3. адміністратор вводить інформацію про бронювання;
4. система перевіряє наявність столиків за вказаною датою, часом і кількістю гостей. Якщо таблиці доступні, система виводить список відповідних варіантів;
5. адміністратор вибирає потрібний столик із доступних варіантів. Система може запропонувати адміністратору підтвердити вибір і відобразити відповідні відомості про таблицю;
6. адміністратор переглядає деталі бронювання, включаючи інформацію про гостей, вибраний столик і час. Система

дозволяє вносити будь-які необхідні налаштування до остаточного підтвердження;

7. адміністратор підтверджує бронювання натисканням кнопки «Завершити бронювання». Система обробляє запит і створює новий запис бронювання в базі даних;
8. система оновлює статус доступності забронюваного столика, щоб відобразити, що він більше не доступний на вибрану дату та час;
9. повідомлення про підтвердження генерується та надсилається гостю через надану контактну інформацію, підтверджуючи деталі бронювання;
10. система реєструє дії резервування в журналі активності адміністратора для подальшого використання та аудиту.

Альтернативні потоки:

1. якщо на вказану дату й час немає вільних столиків, система сповістить адміністратора та запропонує вибрати інший час або запропонувати альтернативні варіанти обіду;
2. якщо виникають проблеми з контактною інформацією гостя, система сповіщає адміністратора та дозволяє йому повторно ввести або оновити дані перед завершенням бронювання.

Цей сценарій описує процес замовлення адміністратором столика в ресторані за допомогою інформаційної системи. Виконуючи ці кроки, адміністратор може ефективно керувати бронюваннями, забезпечувати точні записи та покращувати загальний досвід обіду для гостей. Цей варіант використання підкреслює важливість ефективного спілкування між адміністратором і системою для оптимізації процесу бронювання столиків.

2.2.2 Діаграма послідовності. Діаграма послідовності — це тип діаграми взаємодії в уніфікованій мові моделювання (UML), яка ілюструє, як об’єкти взаємодіють один з одним у певній послідовності протягом певного часу. Він надає візуальне представлення потоку повідомлень, якими

обмінюються об’єкти, і порядок, у якому відбувається ця взаємодія. Діаграми послідовностей особливо корисні для моделювання поведінки системи під час конкретних сценаріїв або випадків використання, що робить їх важливим інструментом для розуміння динаміки взаємодії об’єктів у розробці програмного забезпечення [10].

Діаграми послідовності особливо цінні на етапі проєктування розробки програмного забезпечення, оскільки вони допомагають прояснити послідовність подій і взаємодій у випадку використання. Вони дають зрозуміти, як різні компоненти системи працюють разом, дозволяючи розробникам виявляти потенційні проблеми, надмірності або вузькі місця в потоці інформації.

Однією з значних переваг діаграм послідовностей є їх здатність представляти складні взаємодії в чіткій і стислій формі. Вони служать ефективними інструментами спілкування, сприяючи обговоренню поведінки та дизайну системи між членами команди, зацікавленими сторонами та клієнтами.

Таким чином, діаграми послідовності є ключовим аспектом UML, пропонуючи візуальне представлення того, як об’єкти взаємодіють протягом певного часу в системі. Відображаючи порядок повідомлень і потік керування, діаграми послідовності допомагають розробникам зрозуміти поведінку системи, покращити зв’язок між членами команди та керувати впровадженням програмних рішень. Їх роль у моделюванні

динамічних взаємодій робить їх важливим інструментом у процесі розробки програмного забезпечення.

Діаграма послідовності, зображена на рис. 4, включає наступні об'єкти:

- адміністратор;
- пошук номеру;
- бронювання номеру;
- оплата.

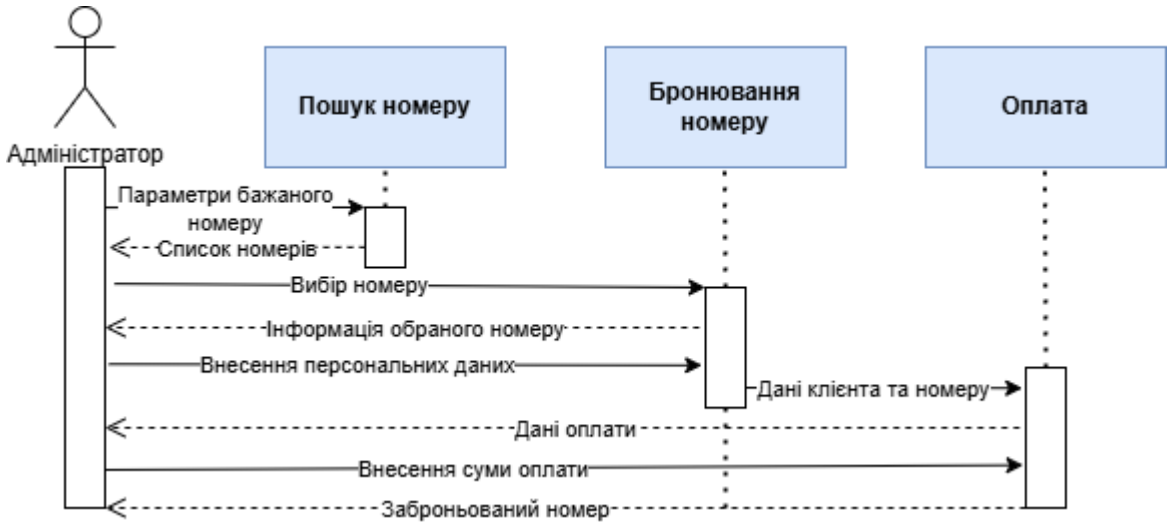

Рис. 4 Діаграма послідовності

Об'єкт «Адміністратор» ініціює запит до об'єкта «Пошук номеру», вказуючи бажані параметри кімнати. У відповідь він отримує список вільних номерів, які відповідають його критеріям. Після цього адміністратор вибирає один із запропонованих номерів і надсилає запит на його бронювання, разом із даними користувача. Ця інформація передається до об'єкта «Бронювання номеру», де необхідні дані заповнюються для подальшої обробки.

На наступному етапі об'єкт «Бронювання номеру» формує запит до об'єкта «Оплата», включаючи дані клієнта. У відповідь система обробляє запит і надсилає результат транзакції. Після успішної обробки оплати

адміністратор отримує підтвердження про бронювання номера і завершує роботу з програмою.

2.2.3 Діаграма активності. Діаграми діяльності — це тип діаграм поведінки в уніфікованій мові моделювання (UML), які ілюструють потік керування або даних у системі. Вони використовуються для моделювання динамічних аспектів системи, показуючи послідовність дій, моменти прийняття рішень і одночасні процеси, задіяні в певному робочому або бізнес-процесі. Діаграми діяльності особливо корисні для візуалізації складних операцій, що робить їх важливим інструментом як для аналізу, так і для проєктування в розробці програмного забезпечення [12].

Однією з ключових переваг діаграм діяльності є їх здатність забезпечувати чітке та стисле представлення робочих процесів. Вони дозволяють зацікавленим сторонам візуалізувати складні процеси, полегшуючи виявлення вузьких місць, надмірностей або областей, які потребують вдосконалення. Відображаючи потік дій, ці діаграми сприяють обговоренню між членами команди та допомагають переконатися, що всі мають спільне розуміння процесу.

Діаграми діяльності зазвичай використовуються на різних етапах розробки програмного забезпечення, включаючи збір вимог, проєктування системи та оптимізацію процесів. Вони можуть ефективно представляти як прості робочі процеси, так і складні процеси, що включають кілька паралельних дій, що робить їх універсальними інструментами для моделювання бізнес-логіки.

Таким чином, діаграми діяльності служать потужним візуальним представленням потоку керування або даних у системі. Ілюструючи послідовність дій, моменти прийняття рішень і обов’язки, вони допомагають розробникам і зацікавленим сторонам зрозуміти складні робочі процеси, визначити потенційні проблеми та покращити загальний

дизайн системи. Їх роль у моделюванні динамічної поведінки робить їх важливим компонентом процесу розробки програмного забезпечення.

Розроблена діаграма активності представлена на рис. 5.

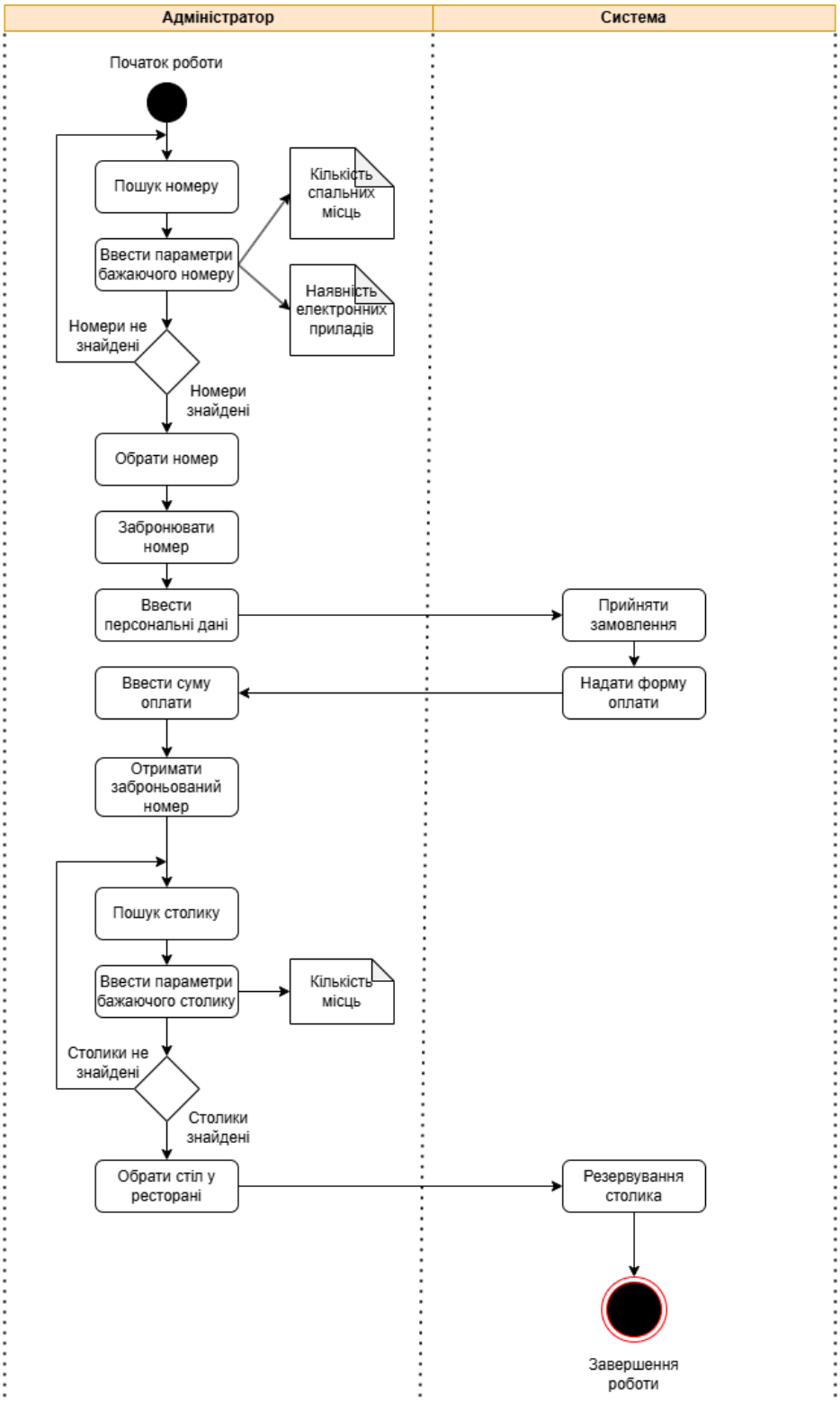

Рис. 5 Діаграма активності

2.3 Абстракції предметної області

Абстракції в предметній області передбачають спрощення складних концепцій, систем або процесів у певній області для полегшення розуміння та спілкування. У сфері розробки програмного забезпечення, особливо для інформаційної системи, призначеної для бронювання в готельному та ресторанному бізнесі, ці абстракції відіграють вирішальну роль. Вони допомагають зацікавленим сторонам зосередитися на основних характеристиках і функціях системи, не перевантажуючись складними деталями [13].

Створення абстракцій дозволяє розробникам, аналітикам і користувачам виділяти основні компоненти системи, що веде до спільного розуміння її роботи. Це колективне розуміння допомагає на різних етапах розробки, включаючи проєктування, впровадження та тестування.

Ключові елементи абстракцій у цьому контексті включають сутності та об’єкти, які представляють важливі поняття, такі як Клієнт, Кімната, Бронювання та Оплата. Кожна з цих сутностей інкапсулює відповідні атрибути та поведінку, служачи основоположним будівельним блоком для системи. Крім того, важливо розуміти зв’язки між цими суб’єктами. Наприклад, Клієнт може мати кілька Бронювань, при цьому кожне Бронювання прив’язане до певної кімнати. Ці з’єднання допомагають визначити структуру бази даних і загальну архітектуру системи.

Абстракції також охоплюють випадки використання, які ілюструють, як різні актори взаємодіють із системою. Такі приклади, як «Забронювати номер» і «Скасувати бронювання», зосереджуються на цілях користувачів і результатах їх взаємодії, а не заглиблюються в технічні деталі. Крім того, ці абстракції представляють процеси та робочі процеси, задіяні в системі бронювання, такі як перевірка доступності, обробка платежів і надсилання підтверджень. Склавши ці робочі процеси, команди

можуть визначити необхідну послідовність дій для користувачів і системи для досягнення своїх цілей.

Крім того, абстракції включають функціональні вимоги, які окреслюють те, що повинна робити система, визначаючи основні функції, необхідні для підтримки взаємодії користувачів, як-от пошук вільних номерів і обробка бронювань. Нефункціональні вимоги, що стосуються таких аспектів, як продуктивність, безпека, зручність використання та масштабованість, також є невід’ємною частиною цих абстракцій, гарантуючи, що система відповідає очікуванням користувачів і працює ефективно за різних умов.

Рис. 6 Абстракції предметної області

2.4 Діаграма класів

Діаграми класів є фундаментальним компонентом уніфікованої мови моделювання (UML), який переважно використовується в об’єктно-орієнтованому проєктуванні для представлення статичної структури системи. Вони надають візуальне зображення класів, їхніх атрибутів,

методів і зв’язків між ними, що робить їх важливими для розуміння архітектури програмного додатку.

В основі діаграми класів знаходяться класи, які є схемами для створення об’єктів. Кожен клас зазвичай представлений прямокутником, поділеним на три частини. Верхній розділ містить назву класу, середній розділ містить список атрибутів (властивостей або полів), а нижній розділ описує методи (функції або операції), пов’язані з цим класом. Наприклад, клас Customer може містити такі атрибути, як name, email і phoneNumber, тоді як його методи можуть містити такі дії, як makeReservation() або cancelReservation().

Діаграми класів виконують кілька важливих завдань у процесі розробки програмного забезпечення. Вони забезпечують чіткий і організований спосіб візуалізації структури системи, допомагаючи розробникам зрозуміти, як різні класи взаємодіють і як дані проходять через додаток. Ця чіткість сприяє кращому спілкуванню між членами команди та зацікавленими сторонами, гарантуючи, що всі мають спільне розуміння дизайну системи [14].

Крім того, діаграми класів є важливими на етапах планування та проєктування розробки програмного забезпечення. Вони допомагають визначити ключові класи та їхні взаємозв’язки, керуючи розробниками у створенні узгодженої та модульної архітектури системи. Відображаючи основні компоненти програми, діаграми класів дозволяють командам виявляти потенційні проблеми або надмірності до початку впровадження.

Таким чином, діаграми класів є життєво важливим інструментом в об’єктно-орієнтованому проєктуванні, пропонуючи комплексне уявлення про класи, їхні атрибути, методи та зв’язки в системі. Вони покращують розуміння, сприяють ефективній комунікації та допомагають у проєктуванні та розробці програмного забезпечення.

Для цього веб-сайту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 7).

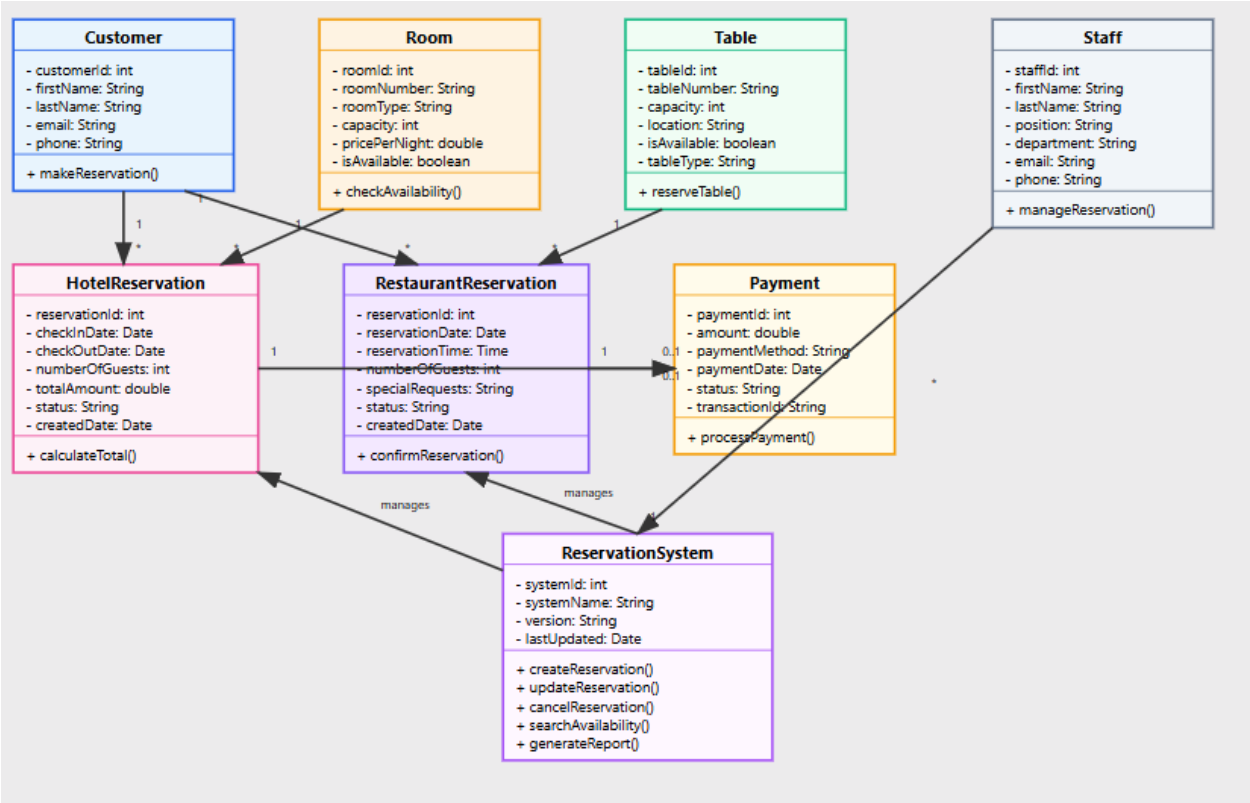

Рис. 7 Діаграма класів

Ця діаграма класів представляє систему бронювання готелів та ресторанів, ілюструючи основні компоненти та їх взаємодію. По суті, клієнти є основними користувачами, кожен з яких ідентифікується унікальними деталями, такими як ім'я, контактна інформація та можливість бронювання. Система обробляє як бронювання номерів у готелях, так і бронювання столиків у ресторанах, кожне з яких керується через окремі, але взаємопов'язані структури.

Номери в готелі характеризуються такими атрибутами, як тип, місткість, ціна та доступність, що дозволяє користувачам перевіряти, чи вільний номер. Аналогічно, столики в ресторанах мають різні ідентифікатори, місткість, розташування та статус доступності, що забезпечує безперебійний процес бронювання. Спеціально розроблений персонал керує цими бронюваннями, а співробітникам призначені певні ролі та обов'язки, що охоплюють різні відділи.

Система бронювання контролює весь процес бронювання. Для готелів бронювання включає такі деталі, як дати заїзду та виїзду, кількість гостей, загальна вартість та статус бронювання, з методами розрахунку плати. Бронювання ресторанів зосереджується на часі обіду, кількості гостей, спеціальних запитах та процесах підтвердження. У систему інтегрована функція обробки платежів, яка дозволяє здійснювати транзакції різними способами, відстежуючи суми, статус платежу та деталі транзакції.

Загальна система бронювання спрощує створення, оновлення та скасування бронювань, водночас забезпечуючи пошук доступності номерів та створення звітів. Клієнти взаємодіють із системою, розміщуючи бронювання, а персонал керує операціями та підтримує їх. Обробка платежів забезпечує безперебійні транзакції, пов'язуючи фінансові елементи з бронюваннями. Ця структурована структура гарантує ефективність та організованість в управлінні готелями та ресторанами, покращуючи загальний досвід користувачів.

2.5 Функціональна модель

Діаграми SADT (Structured Analysis and Design Technique) — це інструмент візуального моделювання, який використовується переважно для аналізу та проєктування складних систем. Розроблений Дугласом Т. Россом у 1970-х роках, SADT забезпечує структурований підхід до представлення системних функцій та їх взаємодії, полегшуючи зацікавленим сторонам розуміння поведінки та вимог системи. Ця методологія наголошує на декомпозиції процесів на простіші, більш керовані компоненти, що дозволяє чіткіше уявити, як працює система.

В основі діаграм SADT знаходяться блоки процесів, які представляють функції або дії в системі. Кожна коробка зазвичай містить мітку, що описує функцію, яку вона виконує. Блоки процесів з’єднані

стрілками, які ілюструють потік даних і керування між ними, підкреслюючи, як різні функції взаємодіють і залежать одна від одної.

Діаграми SADT пропонують кілька переваг у системному аналізі та проєктуванні. Вони забезпечують чітке та інтуїтивно зрозуміле візуальне представлення функцій системи, полегшуючи для зацікавлених сторін розуміння загальної структури та поведінки системи. Ієрархічна природа SADT дозволяє аналітикам розбивати складні процеси на простіші компоненти, полегшуючи більш детальний аналіз і проєктування [18].

Крім того, діаграми SADT можуть бути особливо ефективними для виявлення надмірностей або неефективності в системі. Візуалізуючи потік даних і контроль, аналітики можуть точно визначити області, які потрібно вдосконалити, і переконатися, що всі необхідні функції враховані в проєкті.

Хоча SADT має свої сильні сторони, важливо зазначити, що він може бути не настільки широко застосований, як інші методи моделювання, такі як UML. Однак його структурований підхід і зосередженість на функціональній декомпозиції роблять його цінним інструментом для

конкретних застосувань, особливо на ранніх етапах системного аналізу та проєктування.

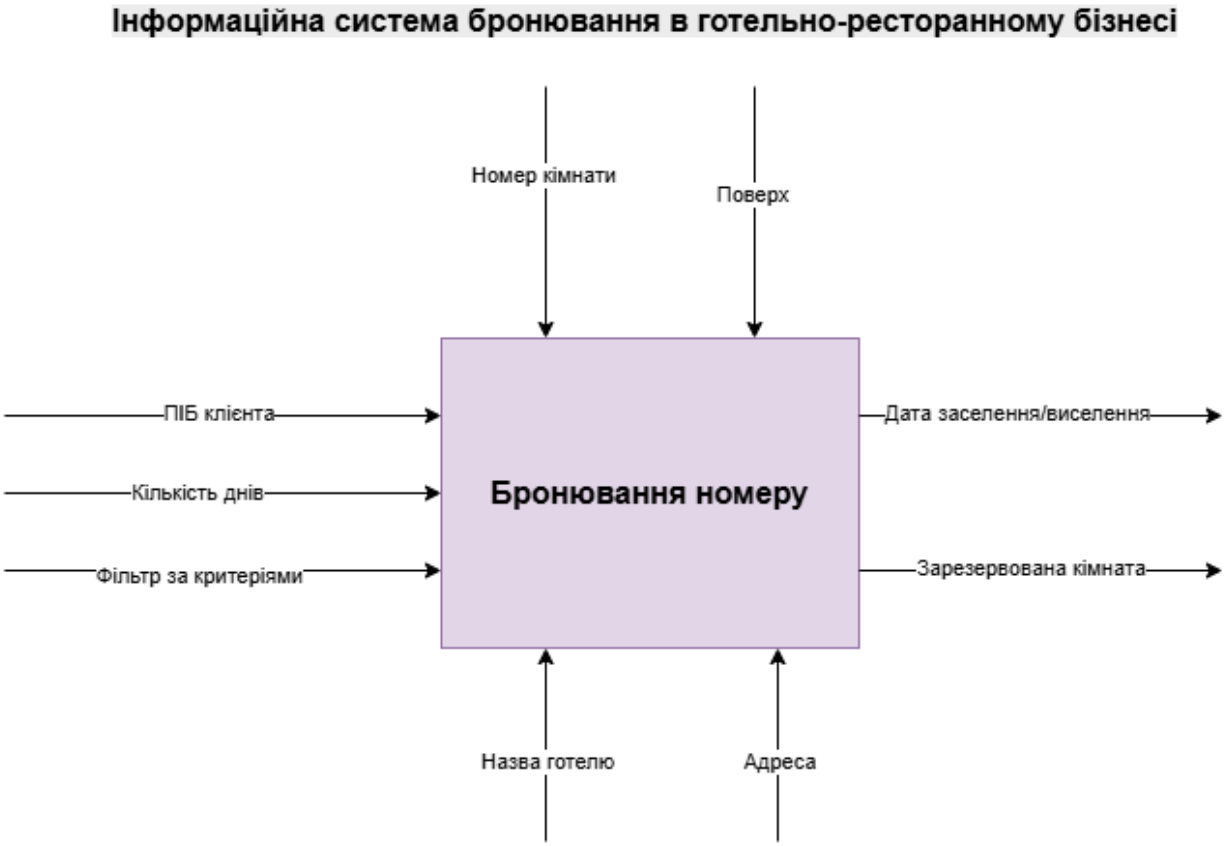

Рис. 8 Функціональна модель

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних представляє організацію, структуру та взаємозв’язки даних у системі, забезпечуючи детальне уявлення про вимоги до даних без розгляду особливостей фізичної реалізації. Ця модель є важливою ланкою між концептуальними проєктами високого рівня та фактичним налаштуванням бази даних, гарантуючи, що дизайн узгоджується з потребами бізнесу чи програми [17].

Центральними для логічної моделі даних є кілька ключових компонентів, починаючи з сутностей, які позначають об’єкти або концепції, що мають відношення до бізнесу та мають чітке існування. Наприклад, у системі бронювання готелів і ресторанів сутності можуть включати клієнта, номер, бронювання та оплату. Кожна з цих сутностей містить атрибути — властивості, які описують сутність. Наприклад, сутність клієнта може містити такі атрибути, як ідентифікатор клієнта, ім’я, електронна адреса та номер телефону.

Зв’язки також є фундаментальним аспектом логічної моделі даних, що визначає, як сутності взаємопов’язані. Ці зв’язки можуть відрізнятися, класифікуючись як «один до одного», «один до багатьох» або «багато до багатьох». Наприклад, один Клієнт може мати кілька Бронювань, тоді як кожне Бронювання відповідає певній кімнаті.

Окрім сутностей і зв’язків, первинні ключі відіграють життєво важливу роль, унікально ідентифікуючи кожен екземпляр сутності. Наприклад, ідентифікатор клієнта може служити первинним ключем для сутності клієнта, забезпечуючи відмінність кожного запису клієнта. Подібним чином зовнішні ключі встановлюють зв’язки між об’єктами, посилаючись на первинні ключі інших об’єктів, таким чином зберігаючи цілісність даних і посилюючи зв’язки. Наприклад, сутність резервування

може містити зовнішній ключ, що посилається на CustomerID від сутності клієнта, щоб вказати, який клієнт зробив конкретне бронювання.

Під час процесу розробки бази даних логічна модель даних в основному використовується на етапі проєктування. Це дозволяє аналітикам і дизайнерам окреслювати структуру даних і взаємозв'язки між різними елементами даних без обмежень, що накладаються фізичними аспектами, такими як формати зберігання чи певні системи керування базами даних. Ця абстракція сприяє гнучкості та дозволяє зосередитися на логічній організації даних.

Однією з істотних переваг логічної моделі даних є її здатність забезпечувати чітку структуру для розуміння вимог системи до даних. Це сприяє ефективній комунікації між зацікавленими сторонами, включаючи бізнес-аналітиків, розробників і адміністраторів баз даних, гарантуючи, що всі мають спільне розуміння того, як дані будуть структуровані та використані.

Крім того, добре розроблена логічна модель даних може допомогти виявити потенційні проблеми, надмірності або неефективність у структурі даних до початку фізичної реалізації. Цей проактивний підхід може призвести до більш ефективних проєктів баз даних, які ефективно відповідають потребам організації.

Таким чином, логічна модель даних є життєво важливим інструментом у процесі проєктування бази даних, пропонуючи структуроване представлення об'єктів даних, атрибутів, зв'язків і ключів. Зосереджуючись на логічній організації даних, він покращує спілкування

між зацікавленими сторонами та закладає основу для фізичної реалізації бази даних.

Логічна модель системи представлена на рисунку 9.

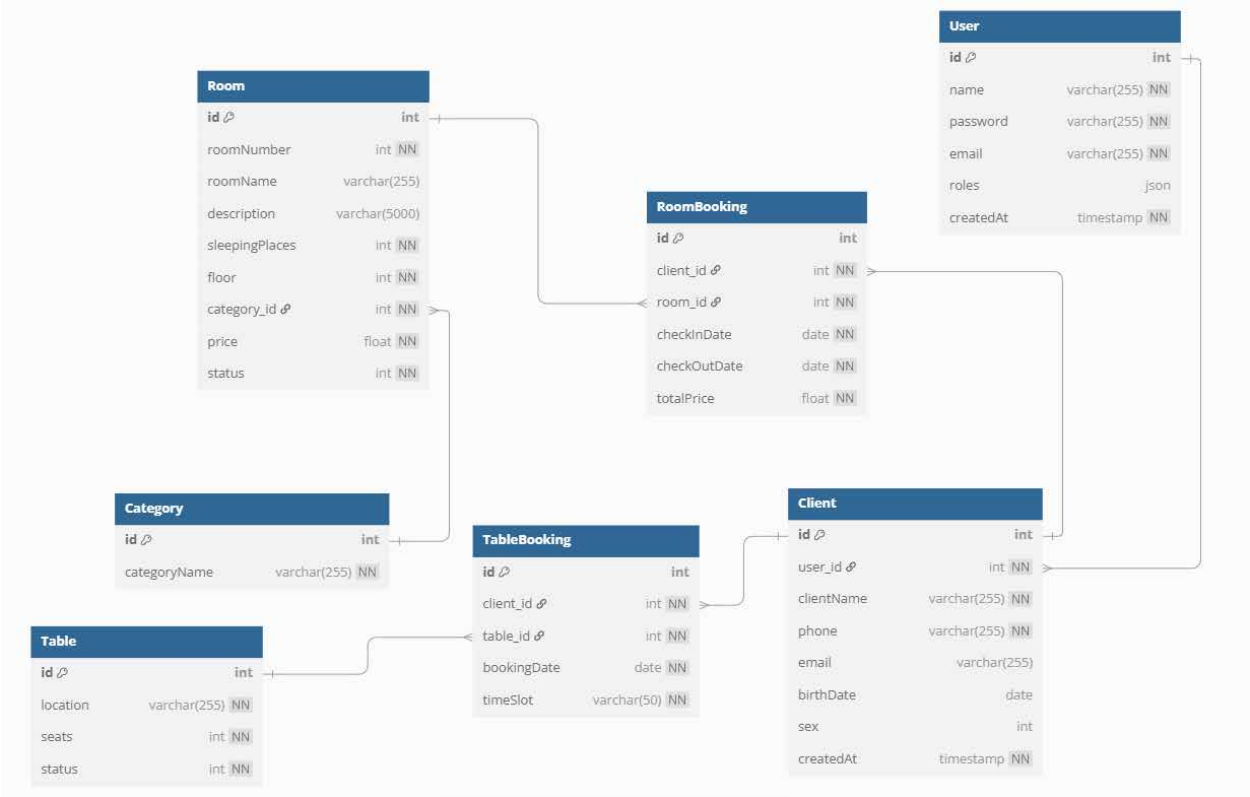

Рис. 9 ER-діаграма

Логічна модель складається з таких сутностей:

- user
- client
- room
- roomBooking
- category
- table
- tableBooking

Опис структури бази даних

User

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожного користувача;
2. name: string (255) - Ім'я користувача;
3. password: string - Пароль для аутентифікації користувача;
4. email: рядок - адреса електронної пошти користувача;
5. roles: array - Ролі, призначені користувачеві (наприклад, ROLE_USER);
6. createdAt: DateTimeImmutable – мітка часу створення користувача.

Client

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожного клієнта;
2. clientName: string (255) - Ім'я клієнта;
3. phone: рядок (255) - Номер телефону клієнта;
4. email: string (255, nullable) - адреса електронної пошти клієнта;
5. birthDate: DateTime (nullable) – дата народження клієнта;
6. sex: int (nullable) - Стать клієнта (1 для чоловічої статі, 2 для жіночої);
7. createdAt: DateTimeImmutable – мітка часу створення клієнта.

Room

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожної кімнати;
2. roomNumber: int - номер, призначений кімнаті;
3. roomName: string (255, nullable) - назва кімнати;
4. description: string (5000, nullable) - Опис кімнати;
5. sleepingPlaces: int - Кількість спальних місць у кімнаті;
6. floor: int - номер поверху, на якому знаходиться кімната;
7. ціна: плаваюча - Ціна за добу за номер;

8. status: int - Статус кімнати (1 для вільної, 2 для зайнятої);
9. category: Категорія (ManyToOne) - категорія, до якої належить кімната.

RoomBooking

1. id: int (первинний ключ, автоматичне збільшення) - унікальний ідентифікатор для кожного бронювання номера;
2. client: Client (ManyToOne) - клієнт, який забронював номер;
3. кімната: кімната (ManyToOne) - кімната, яку було заброньовано;
4. checkInDate: DateTime - дата заїзду для бронювання;
5. checkOutDate: DateTime - дата виїзду для бронювання;
6. totalPrice: float - Загальна вартість проживання;

Table

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожної таблиці;
2. location: string (255) - Опис розташування таблиці;
3. seats: int - Кількість місць за столом;
4. status: int - Статус таблиці (1 для доступної, 2 для зайнятої).

TableBooking

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожного бронювання столика;
2. клієнт: Клієнт (ManyToOne) - Клієнт, який забронював столик;
3. table: Table (ManyToOne) - Столик, який було заброньовано;
4. bookingDate: DateTime - Дата бронювання;
5. timeSlot: DateTime - часовий інтервал для бронювання.

Category

1. id: int (первинний ключ, автозбільшення) - унікальний ідентифікатор для кожної категорії;
2. categoryName: рядок (255) - назва категорії;

3. кімнати: Колекція (OneToMany) - Колекція кімнат, які належать до цієї категорії.

Від клієнта до RoomBooking: один клієнт може мати кілька бронювань номерів. (Відношення один до багатьох). RoomBooking.client посилається на Клієнт. Бронювання номера в номері: один номер може мати кілька бронювань протягом певного часу, але в будь-який момент часу його може забронювати лише один клієнт. (Відношення один до багатьох). RoomBooking.room посилання Кімната. Від клієнта до столу: один клієнт може мати кілька бронювань столиків. (Відношення один до багатьох). TableBooking.client посилається на Client. Столик за столом: один столик можна забронювати кілька разів, але в будь-який момент часу його може забронювати лише один клієнт. (Відношення один до багатьох). Посилання на TableBooking.table Таблиця. Кімната до категорії: кімната належить до однієї категорії, а в одній категорії може бути кілька кімнат. (Відношення «багато до одного»). Посилання на Room.category Категорія.

3.2 Вибір системи управління базою даних та її реалізація

Вибір правильної системи управління базами даних (СУБД) є вирішальним кроком у розробці інформаційної системи бронювання в готельно-ресторанному бізнесі. Вибрана СУБД значно впливає на продуктивність, масштабованість і загальну функціональність програми. Важливо переконатися, що СУБД відповідає конкретним потребам системи, беручи до уваги такі фактори, як цілісність даних, зручність використання, підтримка складних запитів і можливості інтеграції.

При оцінці потенційних варіантів СУБД враховуються кілька ключових факторів. По-перше, розглянемо вимоги до даних системи, яка керуватиме різними типами інформації, включаючи дані про клієнтів, записи про бронювання, наявність номерів і платіжні операції. Система керування реляційною базою даних (RDBMS), як-от MySQL або

PostgreSQL, часто добре підходить для цього типу програм завдяки своїй здатності обробляти структуровані дані та підтримувати зв’язки через зовнішні ключі.

Ще одним важливим фактором є масштабованість. У міру зростання готелю чи ресторану база даних повинна вміщувати все більшу кількість даних і одночасних запитів користувачів. Вибір СУБД, яка підтримує горизонтальне масштабування, наприклад MySQL Cluster або хмарних рішень, таких як Amazon RDS, може допомогти забезпечити ефективне керування майбутнім зростанням.

Продуктивність також важлива. Вибрана СУБД повинна забезпечувати швидкий час відповіді на запити та ефективні механізми індексування для оптимізації таких операцій, як пошук вільних кімнат або обробка бронювань. Крім того, надійні функції безпеки необхідні для захисту конфіденційних даних, включаючи інформацію про клієнтів і платіжні деталі. СУБД повинна пропонувати такі можливості, як шифрування даних, автентифікація користувачів і контроль доступу для захисту цієї інформації [19].

Варто також ретельно оцінити вартість і ліцензування. Розгляньте загальні витрати, пов’язані з СУБД, включаючи ліцензійні збори, поточне обслуговування та витрати на підтримку. Варіанти з відкритим кодом, такі як MySQL або PostgreSQL, можуть бути більш бюджетними для малого та середнього бізнесу порівняно з пропрієтарними системами. Крім того, СУБД має бездоганно інтегруватися з мовою програмування та структурою, що використовується при розробці системи резервування, забезпечуючи сумісність і простоту використання.

Після вибору відповідної СУБД наступним кроком є її впровадження, яке зазвичай включає кілька етапів. Почніть із розробки схеми бази даних на основі логічної моделі даних. Цей дизайн має визначати таблиці для ключових сутностей, таких як «Клієнт», «Комната», «Бронювання» та «Оплата», а також їхні атрибути та зв’язки. Належна

нормалізація є важливою для усунення надмірності та підтримки цілісності даних.

Налаштування бази даних передбачає встановлення вибраної СУБД на сервері або хмарній платформі та налаштування середовища, включаючи облікові записи користувачів, дозволи та налаштування безпеки. Після налаштування створіть необхідні таблиці за допомогою мови структурованих запитів (SQL), забезпечивши визначення первинних ключів для кожної таблиці та встановлення зв'язків зовнішніх ключів для забезпечення цілісності посилань.

Впровадження індексів є ще одним важливим кроком для підвищення продуктивності запитів. Створивши індекси для стовпців, які часто шукають, наприклад імена клієнтів, наявність номерів і дати бронювання, ви можете значно підвищити швидкість операцій пошуку даних.

Якщо існують дані, які потрібно перенести в нову систему, сплануйте та виконайте процес міграції даних. Це може включати вилучення даних із застарілих систем, перетворення їх відповідно до нової схеми та завантаження в нову базу даних. Після цього проведіть ретельне тестування бази даних, щоб переконатися, що всі запити та транзакції функціонують належним чином. Оптимізуйте конфігурацію бази даних і запити на основі результатів фази тестування для підвищення загальної продуктивності.

Після впровадження постійний моніторинг продуктивності та безпеки бази даних є життєво важливим. Щоб забезпечити безперебійну роботу з часом, слід запланувати регулярні завдання технічного обслуговування, включаючи резервне копіювання, оновлення та налаштування продуктивності.

Рішення вибрати MySQL як систему управління базами даних для інформаційної системи бронювання в готельно-ресторанному бізнесі базується на кількох переконливих факторах, які відповідають вимогам і

цілям проєкту. MySQL є однією з найпоширеніших систем керування реляційними базами даних з відкритим кодом, відома своєю надійністю, продуктивністю та простотою використання.

Однією з основних причин вибору MySQL є її надійна продуктивність і здатність ефективно обробляти великий обсяг транзакцій. Враховуючи характер системи бронювання, яка оброблятиме численні бронювання, взаємодії з клієнтами та платіжні операції одночасно, надзвичайно важливо мати базу даних, яка може керувати одночасними користувачами без шкоди для швидкості та надійності. Оптимізована обробка запитів та можливості індексування MySQL забезпечують швидкий пошук даних і ефективну обробку складних запитів, що є важливим для забезпечення бездоганної взаємодії з користувачем [20].

Ще одна суттєва перевага MySQL — це потужна підтримка цілісності та безпеки даних. У індустрії гостинності управління конфіденційною інформацією про клієнтів, зокрема особистими та платіжними даними, має першочергове значення. MySQL пропонує різні функції для забезпечення цілісності даних, такі як підтримка зовнішніх ключів, транзакцій і відповідності ACID (атомність, узгодженість, ізоляція, довговічність). Ці функції допомагають підтримувати узгоджені та точні дані в системі, мінімізуючи ризик помилок або пошкодження даних.

MySQL також добре масштабується, що робить його придатним для прогнозованого зростання готельного та ресторанного бізнесу. Зі збільшенням обсягу бронювань і взаємодії з клієнтами MySQL може легко масштабуватися для розміщення додаткових даних і користувачів. Його здатність обробляти великі бази даних гарантує, що система може розвиватися разом з бізнесом, дозволяючи розширюватися без необхідності повного перегляду бази даних.

Природа MySQL з відкритим кодом дає додаткові переваги для проєкту. Будучи безкоштовним у використанні, MySQL зменшує початкові витрати на впровадження, що робить його економічно вигідним варіантом

для малого та середнього бізнесу. Крім того, широка підтримка спільноти та документація надають цінні ресурси для усунення несправностей та оптимізації бази даних, покращуючи загальний процес розробки.

Сумісність – ще один фактор, який вплинув на рішення. MySQL легко інтегрується з різними мовами програмування та фреймворками, включаючи PHP і Symfony, які використовуються в розробці інформаційної системи. Ця сумісність спрощує процес розробки, забезпечуючи ефективний зв’язок між програмою та базою даних.

Крім того, MySQL пропонує ряд інструментів і функцій, які підтримують адміністрування та управління базами даних. За допомогою таких інструментів, як MySQL Workbench, розробники можуть легко проєктувати, моделювати та керувати базою даних, підвищуючи продуктивність і оптимізуючи зусилля щодо розробки.

Підсумовуючи, MySQL було обрано для реалізації інформаційної системи бронювання завдяки її надійній продуктивності, надійній підтримці цілісності та безпеки даних, масштабованості, економічній ефективності, сумісності з існуючими технологіями та комплексним інструментам управління. Ці атрибути роблять MySQL ідеальним вибором для підтримки операційних потреб готельного та ресторанного бізнесу, одночасно забезпечуючи надійне та ефективне бронювання для клієнтів.

Отриману схему бази даних зображено на малюнку 10.

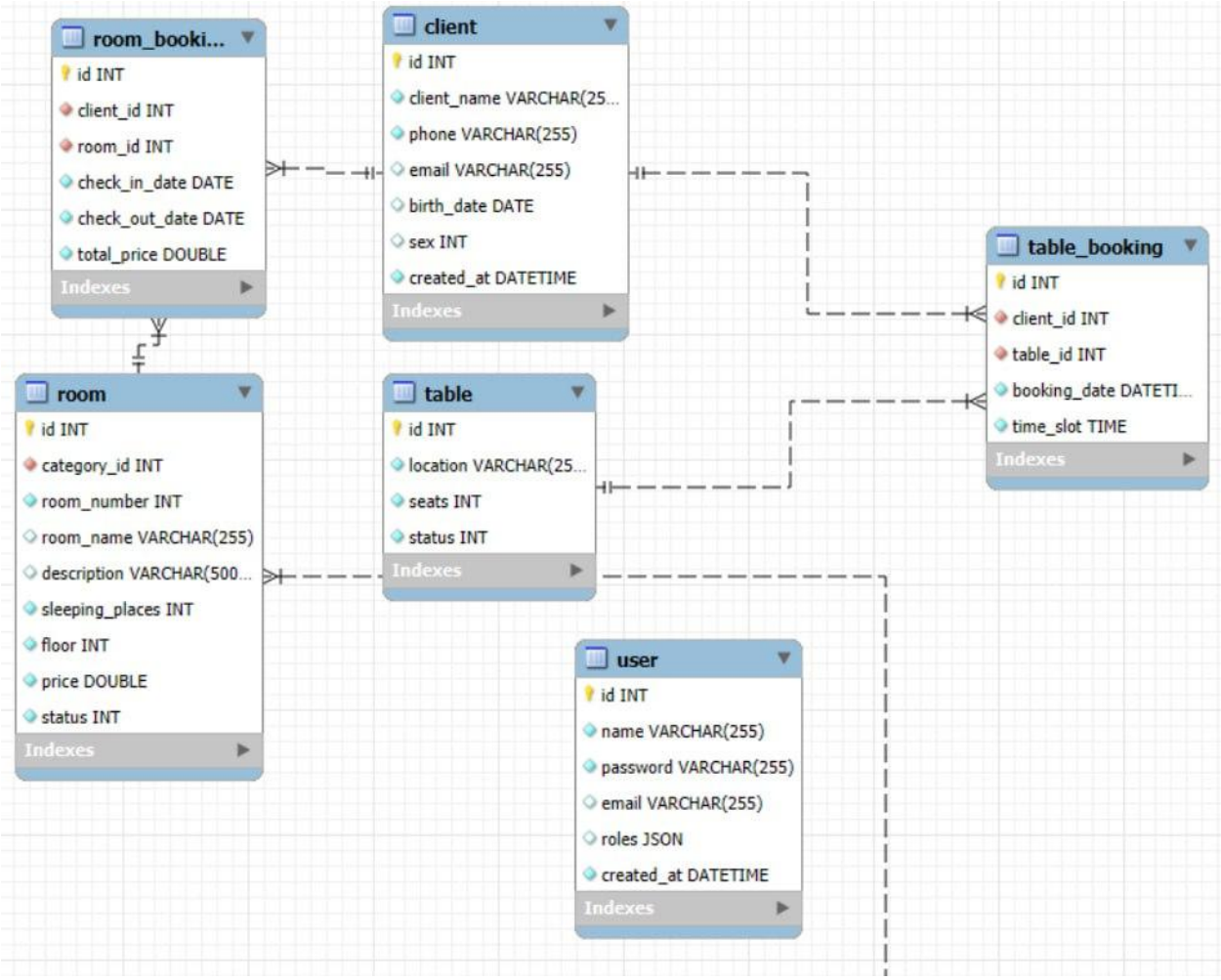

Рис. 10 База даних системи

3.3 Архітектура програмного забезпечення

Програмна архітектура інформаційної системи бронювання в готельно-ресторанному бізнесі визначає структуру та компоненти системи, що забезпечує масштабованість, ремонтопридатність і функціональність. Ця архітектура складається з кількох окремих рівнів, кожен з яких служить певній меті для підтримки загальної роботи системи.

Рівень презентації, або рівень інтерфейсу користувача, є найвищим рівнем і відповідає за взаємодію з користувачами. Він забезпечує зручний інтерфейс, через який і клієнти, і адміністратори можуть отримати доступ до різноманітних функцій системи. Інтерфейс розроблено з використанням HTML, CSS і JavaScript, що дозволяє користувачам взаємодіяти з програмою

через веб-браузери. Цей інтерфейс розроблено таким чином, щоб бути чуйним, забезпечуючи сумісність на різних пристроях, таких як настільні ПК, планшети та смартфони. Основна увага приділяється створенню інтуїтивно зрозумілого користувальницького досвіду, який дозволяє користувачам легко орієнтуватися в системі, шукати вільні кімнати або столики, робити бронювання та керувати своїми бронюваннями.

Під презентаційним рівнем знаходиться прикладний рівень, який обробляє основні функції та бізнес-логіку системи. Розроблений за допомогою PHP і фреймворку Symfony, цей рівень обробляє запити користувачів і керує робочими процесами. Основні функції включають керування бронюванням, яке охоплює створення, зміну та скасування бронювань готельних номерів і столиків у ресторанах. Крім того, програма перевіряє наявність номерів і столиків на основі запитів користувачів і конкретних дат. Іншим важливим аспектом є обробка платежів, де система інтегрується з платіжними шлюзами для безпечної обробки транзакцій. Автоматичні сповіщення також є частиною цього рівня, гарантуючи, що користувачі отримують підтвердження та оновлення щодо своїх бронювань.

Рівень даних необхідний для керування зберіганням і пошуком даних. MySQL було обрано як систему керування реляційною базою даних для проєкту, яка зберігає життєво важливу інформацію, таку як деталі клієнтів, записи про бронювання, наявність номерів та платіжну інформацію. Логічна модель даних описує структуру бази даних, визначаючи ключові сутності, такі як Клієнт, Номер, Бронювання та Оплата, а також їхні атрибути та зв’язки. Цей дизайн має вирішальне значення для підтримки цілісності даних і підтримує ефективне надсилання запитів. Рівень доступу до даних додатково абстрагує взаємодію з базою даних, дозволяючи прикладному рівню виконувати операції без прямого підключення до СУБД. Це розділення покращує зручність обслуговування та спрощує можливі майбутні зміни в структурі бази даних.

Безпека є першорядним фактором в архітектурі, особливо з огляду на конфіденційний характер інформації про клієнтів і платіжних деталей. Система реалізує безпечні механізми автентифікації користувачів, такі як хешування паролів і керування сеансами, для захисту облікових записів користувачів. Протоколи шифрування даних використовуються для захисту передачі конфіденційної інформації між клієнтами та сервером. Крім того, встановлено заходи контролю доступу, щоб обмежити функції на основі ролей користувачів, гарантуючи, що лише авторизований персонал може виконувати певні дії.

Архітектура розгортання визначає, як програма буде розміщена та доступна для користувачів. Веб-сервер, такий як Apache або Nginx, розміщуватиме програму, керуватиме вхідними запитами та обслуговуватиме зовнішній інтерфейс. Рішення для хмарного хостингу, такі як AWS, Google Cloud або DigitalOcean, можна використовувати для забезпечення масштабованості та надійності. Впровадження балансування навантаження може розподілити трафік між кількома примірниками сервера, зберігаючи високу доступність і продуктивність, особливо в періоди пікового використання [21].

Підсумовуючи, програмна архітектура інформаційної системи бронювання в готельно-ресторанному бізнесі побудована так, щоб забезпечити масштабовану, ефективну та безпечну платформу для керування бронюваннями. Завдяки розмежуванню окремих рівнів — презентації, програми та даних — архітектура сприяє зручності обслуговування, покращує взаємодію з користувачем і забезпечує надійну продуктивність. Такий організований підхід дозволяє системі адаптуватися та розвиватися разом із бізнесом, ефективно задовольняючи мінливі операційні потреби.

Нижче буде продемонстровано багатошарова архітектура даного програмного забезпечення (рис. 11)

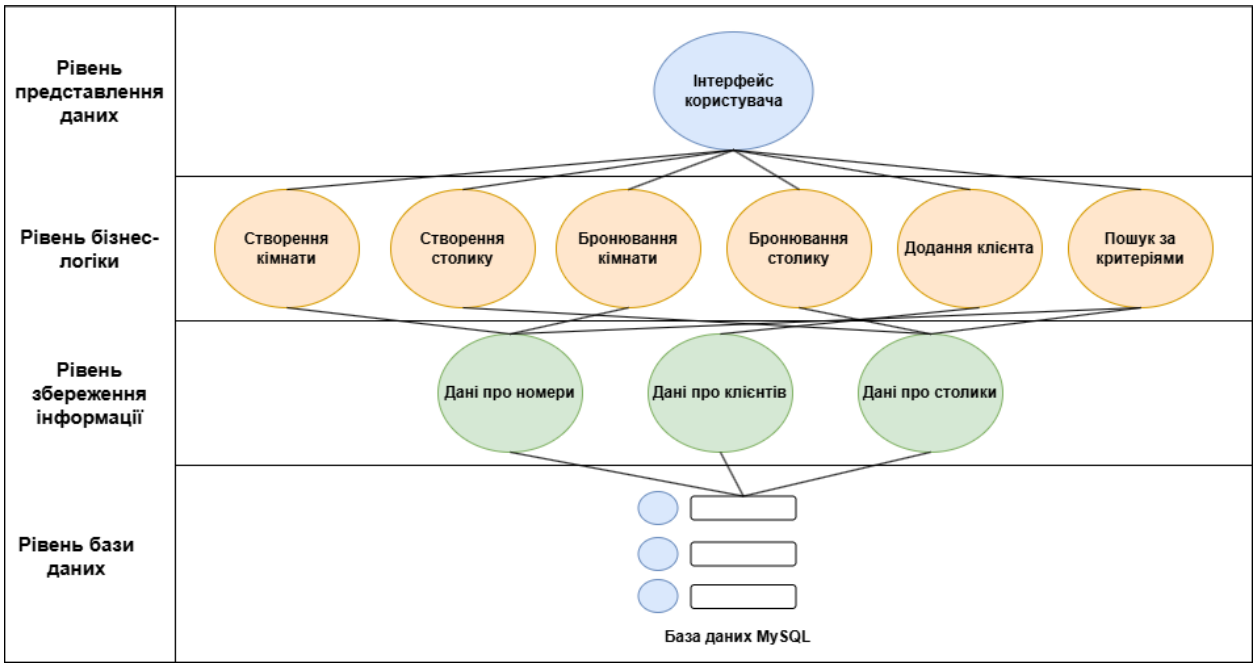

Рис. 11 Багатошарова архітектура веб-сайту

Архітектура нашої системи розроблена як багаторівнева структура, що складається з чотирьох окремих рівнів: рівня презентації, рівня бізнес-логіки, рівня збереження даних та рівня бази даних. Таке налаштування дозволяє інформації плавно перетікати з одного рівня на інший, створюючи цілісну та організовану структуру.

Перший рівень — це інтерфейс користувача, який відображається на екрані комп'ютера або мобільного пристрою. Це місце, де користувачі взаємодіють із системою, отримуючи доступ до різноманітних функцій і функцій, щоб покращити свій досвід.

Другий рівень охоплює основні функції програми, якими користувачі активно займаються. На цьому рівні користувачі можуть виконувати операції, пов'язані з номерами, столиками та бронюванням, оптимізуючи загальний досвід роботи на платформі.

В основі архітектури лежить третій рівень, який складається з бази даних. Цей рівень зберігає всю необхідну системі інформацію. Структурований дизайн архітектури ефективно організовує компоненти системи, забезпечуючи плавний зв'язок між ними. Таке розташування не тільки полегшує ефективну обробку даних, але й покращує взаємодію з

користувачем, що в кінцевому підсумку сприяє бездоганному досвіду для всіх користувачів.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграми пакетів — це тип структурної діаграми в уніфікованій мові моделювання (UML), яка організовує та групує пов’язані елементи системи в пакети. Вони забезпечують високорівневе уявлення про архітектуру системи, ілюструючи, як різні пакети взаємодіють один з одним. Діаграми пакетів допомагають керувати складністю системи, розбиваючи її на більш дрібні, більш керовані компоненти, полегшуючи розуміння та підтримку загальної структури [22].

На діаграмі пакетів пакунки представлені у вигляді прямокутних коробок, часто з невеликою вкладкою у верхньому лівому куті, яка вказує на те, що це пакунки. Кожен пакет може містити різні елементи UML, такі як класи, інтерфейси, компоненти та навіть інші пакети. Ця ієрархічна організація дозволяє краще відокремити проблеми та сприяє модульному дизайну.

Діаграми пакетів служать кільком важливим цілям у проєктуванні та розробці програмного забезпечення. Вони надають чіткий огляд архітектури системи, полегшуючи зацікавленим сторонам розуміння організації різних компонентів і способів їх взаємодії. Ця ясність допомагає спілкуватися між членами команди та допомагає визначити потенційні проблеми, пов’язані із залежностями та зв’язком між пакетами.

Крім того, пакетні діаграми є корисними на етапах планування та проєктування проєкту. Вони полегшують ідентифікацію модулів, які можна розробити та протестувати незалежно, сприяючи більш ефективному процесу розробки. Візуалізуючи взаємозв’язки між пакетами, розробники

можуть краще оцінити вплив змін, внесених до одного пакета, на інші, що призводить до прийняття більш обґрунтованих дизайнерських рішень.

Таким чином, пакетні діаграми є цінним інструментом для організації та візуалізації структури системи. Групуючи пов’язані елементи в пакети та ілюструючи їхні зв’язки, ці діаграми допомагають керувати складністю, покращувати комунікацію та керувати проєктуванням і розробкою програмних додатків.

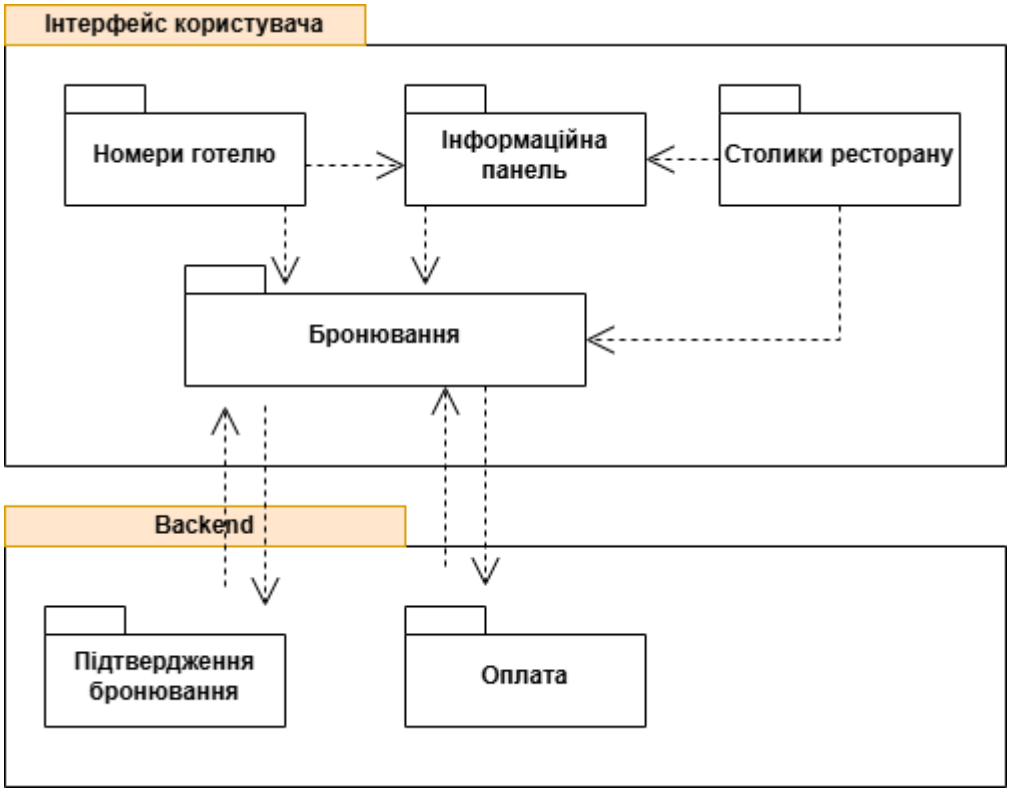

Рис. 12 Діаграма пакетів

3.5 Вибір інструментарію для створення програмного забезпечення

При розробці інформаційної системи бронювання в готельно-ресторанному бізнесі вибір правильних інструментів має вирішальне значення для забезпечення успішного впровадження. Вибрані технології не тільки впливають на продуктивність і масштабованість програми, але й впливають на процес розробки та загальний досвід користувача. Для цього

проєкту були обрані такі інструменти: PHP, Symfony, MySQL, Doctrine ORM, PHPStorm, HTML, CSS і JavaScript. Кожен інструмент відіграє певну роль у стеку розробки, сприяючи надійній та ефективній системі.

PHP — це широко поширена серверна мова сценаріїв, яка особливо добре підходить для веб-розробки. Його простота використання, гнучкість і сильна підтримка спільноти роблять його чудовим вибором для створення динамічних веб-додатків. Великі бібліотеки та фреймворки PHP дозволяють розробникам створювати складні функціональні можливості з відносною простотою, що важливо для системи резервування, яка вимагає різноманітних інтерактивних функцій.

Symfony — це потужний фреймворк PHP, який надає структурований і багаторазово використовуваний набір компонентів для розробки веб-додатків. Прийнявши Symfony, команда розробників отримує переваги від її модульної архітектури, яка просуває найкращі практики та полегшує розробку коду, який можна масштабувати та підтримувати. Вбудовані функції Symfony, такі як маршрутизація, створення шаблонів і безпека, спрощують процес розробки, дозволяючи команді зосередитися на реалізації основних функцій системи бронювання.

MySQL було обрано як систему управління реляційною базою даних для проєкту. Відомий своєю надійністю та продуктивністю, MySQL ефективно обробляє структуровані дані, необхідні для керування бронюваннями, інформацією про клієнтів і платіжними записами. Його підтримка властивостей ACID (атомність, узгодженість, ізоляція, довговічність) забезпечує цілісність даних, що має вирішальне значення для системи, яка обробляє конфіденційну інформацію. Крім того, масштабованість MySQL дозволяє базі даних розвиватися разом із бізнесом, враховуючи збільшення обсягу даних і активності користувачів.

Doctrine ORM (Object-Relational Mapping) інтегровано з Symfony для спрощення взаємодії з базою даних. Doctrine надає потужний рівень абстракції, який дозволяє розробникам працювати із записами бази даних як

об’єктами PHP, полегшуючи маніпуляції та керування даними. Цей інструмент підвищує продуктивність, дозволяючи розробникам зосередитися на бізнес-логіці, а не працювати з необробленими запитами SQL. Підтримка Doctrine для різних платформ баз даних додає гнучкості, дозволяючи проєкту адаптуватися до майбутніх змін вимог до баз даних.

Для середовища розробки було обрано PHPStorm як інтегроване середовище розробки (IDE). PHPStorm відомий своїми надійними функціями, призначеними для розробки PHP, включаючи інтелектуальне завершення коду, потужні інструменти налагодження та повну інтеграцію з системами контролю версій. Його зручний інтерфейс і широка підтримка плагінів сприяють більш ефективному процесу розробки, дозволяючи розробникам писати, тестувати та підтримувати код з більшою легкістю.

На передній частині HTML, CSS і JavaScript використовуються для створення адаптивного та привабливого інтерфейсу користувача. HTML служить основою веб-сторінок, ефективно структуруючи вміст. CSS використовується для покращення візуального дизайну та макета, забезпечуючи привабливу взаємодію з користувачем, яка відповідає сучасним стандартам дизайну. JavaScript додає програмі інтерактивність, уможливлюючи такі динамічні функції, як перевірка форм, оновлення в реальному часі та сповіщення користувачів, що є важливими для покращення загальної взаємодії з користувачем.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Розробка інформаційної системи бронювання в готельно-ресторанному бізнесі вимагає ретельного врахування як апаратних, так і програмних вимог. Ці специфікації мають важливе значення для забезпечення ефективної роботи, задоволення вимог користувачів і забезпечення бездоганного досвіду для клієнтів і адміністраторів.

Що стосується апаратного забезпечення, вимоги залежатимуть від очікуваної кількості користувачів, обсягу даних і загальних потреб у продуктивності. Рекомендується виділений сервер або хмарна інфраструктура, в ідеалі обладнана принаймні вісьмома ядрами ЦП для ефективної обробки кількох одночасних запитів. Мінімум 16 ГБ оперативної пам’яті має важливе значення для безперебійної роботи, особливо під час пікового використання, коли до системи одночасно звертаються багато користувачів. Крім того, використання твердотільного накопичувача (SSD) ємністю щонайменше 500 ГБ покращить пошук і зберігання даних, значно покращуючи продуктивність бази даних.

На стороні клієнта пристрої користувача, такі як настільні ПК, ноутбуки, планшети або смартфони, повинні мати принаймні двоядерний процесор для підтримки безперебійної роботи веб-браузерів і програм. Для ефективної багатозадачності та роботи з веб-додатками рекомендовано мінімум 4 ГБ оперативної пам’яті. Також важливо, щоб користувачі мали доступ до сучасних веб-браузерів, таких як Google Chrome, Mozilla Firefox або Safari, які підтримують останні веб-стандарти, включаючи HTML5, CSS3 і JavaScript. Стабільне підключення до Інтернету з мінімальною швидкістю 10 Мбіт/с необхідне для надійного доступу до системи, особливо

якщо вона розміщена в хмарі. Крім того, захищена локальна мережа (LAN) у готелі чи ресторані полегшить зв’язок між пристроями та сервером.

Що стосується вимог до програмного забезпечення, сервер повинен працювати на системі на базі Linux, такій як Ubuntu Server або CentOS, через її стабільність, безпеку та надійну підтримку веб-додатків. Встановлення програмного забезпечення веб-сервера, наприклад Apache або Nginx, буде необхідним для обслуговування програми користувачами. На сервері також має бути встановлений PHP разом із необхідними розширеннями (наприклад, MySQLi або PDO), щоб увімкнути зв’язок із базою даних MySQL.

Для управління даними MySQL використовуватиметься як система керування реляційною базою даних, яка ефективно зберігатиме всі відповідні дані, пов’язані з клієнтами, бронюваннями та платежами. Фреймворк Symfony буде використовуватися для внутрішньої розробки, пропонуючи надійну структуру та основні компоненти для швидкої розробки додатків. Щоб полегшити взаємодію між програмою та базою даних, буде інтегровано Doctrine ORM, що дозволить розробникам працювати із записами бази даних як з об’єктами, а не мати справу з необробленими запитами SQL.

Для розробки інтерфейсу використовуватимуться HTML, CSS і JavaScript для створення адаптивного та інтерактивного інтерфейсу користувача. Для подальшого покращення взаємодії з користувачем можна включити такі бібліотеки та фреймворки, як Bootstrap для CSS і jQuery для JavaScript. Середовище розробки підтримуватиметься PHPStorm, який відомий своїми розширеними функціями, які допомагають розробці PHP, налагодженню та інтеграції контролю версій.

Інструменти тестування та розгортання відіграватимуть важливу роль у забезпеченні надійності програми. Такі інструменти, як PHPUnit для модульного тестування та рішення для розгортання, як Docker для

керування контейнерами додатків і середовищами, сприятимуть плавному процесу розробки та тестування.

На рисунку 13 зображено діаграму розгортання даної системи. Клієнт-серверна архітектура реалізована на двох окремих серверах.

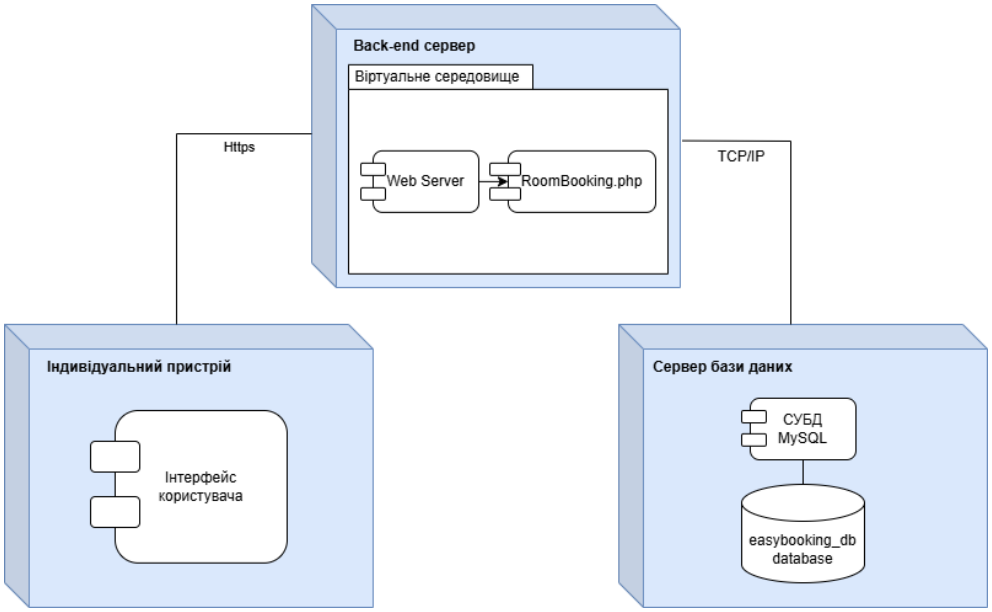

Рис. 13 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, бачимо головну сторінку програмного забезпечення (рис. 14).

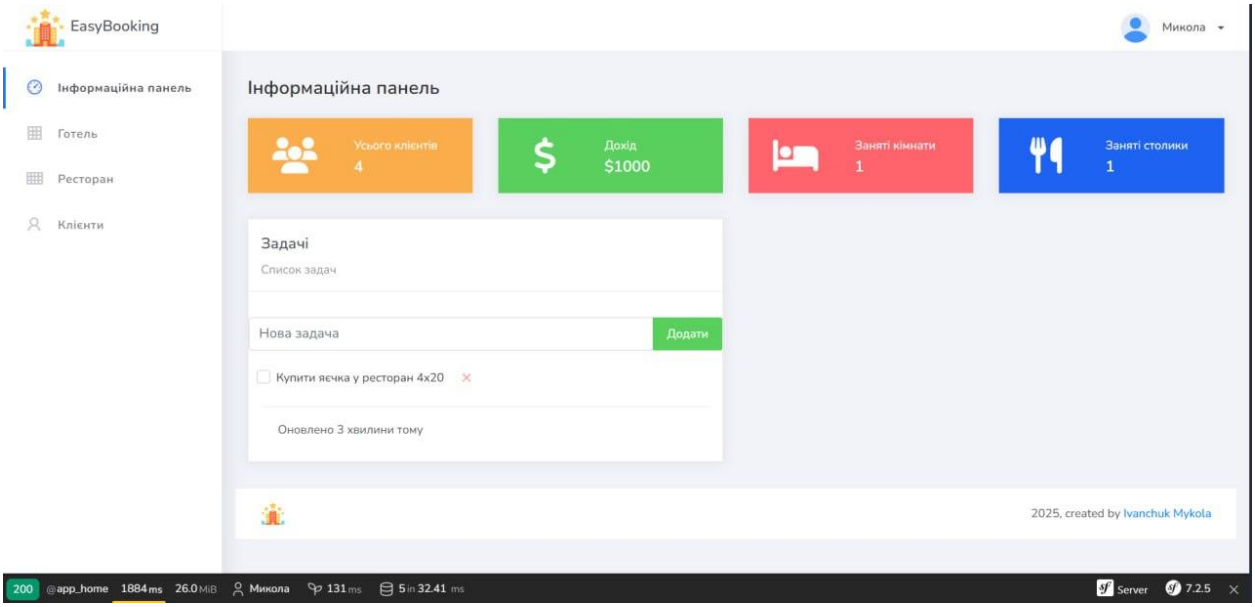

Рис. 14 Головна сторінка

На головній сторінці ми бачимо основні статистичні дані про готель на кшталт кількості зайнятих столиків, номерів, кількості доходів за добу, маємо можливість щось записати до свого списку завдань.

Перейдемо на сторінку клієнтів. Тут адміністратор нашої системи може зареєструвати клієнта, ввівши його персональні дані для подальшої взаємодії із системою, щоб резервувати номер у готелі або столик у ресторані на ім'я цього клієнта (Рис. 15).

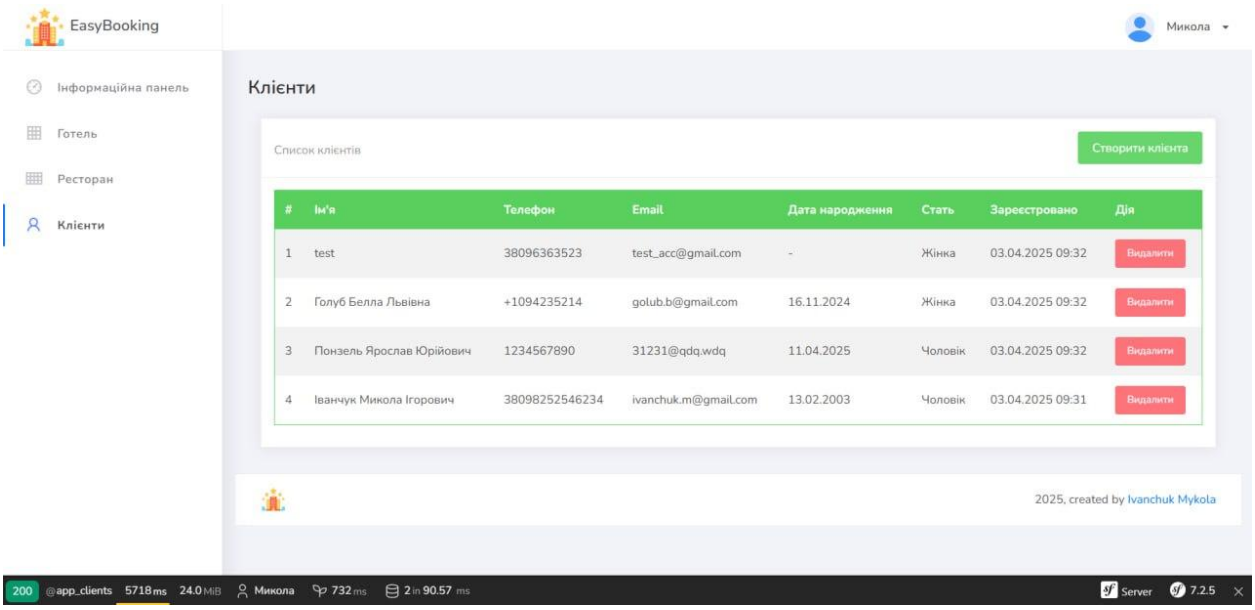

Рис. 15 Сторінка “Клієнти”

Натиснувши на кнопку "Створити клієнта", у нас відкривається форма для введення даних про клієнта і після успішного введення та підтвердження даний клієнт з'являється у нас у базі даних (Рис. 16).

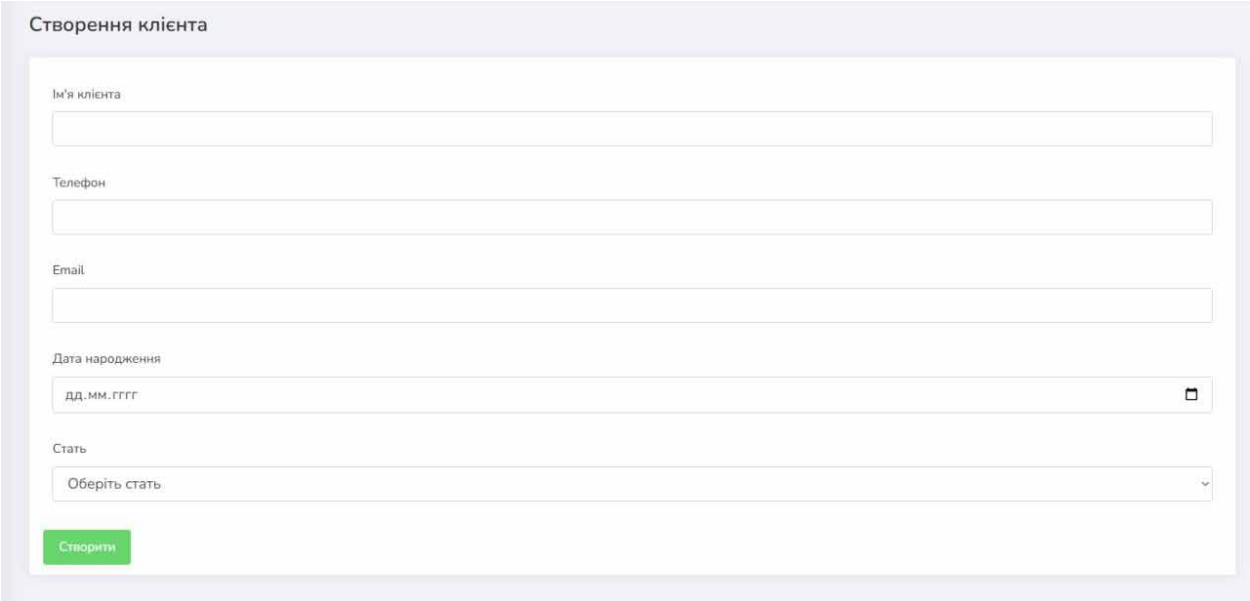

Рис. 16 Форма створення клієнта

Тепер ми можемо перейти на сторінку з готелем. Тут у нас представлені доступні номери, що класифікуються за категоріями (Економ, Середній, Преміум, Люкс). Якщо номер зелений означає він вільний для розміщення, а якщо сірий, то в даний момент він зайнятий (Рис. 17).

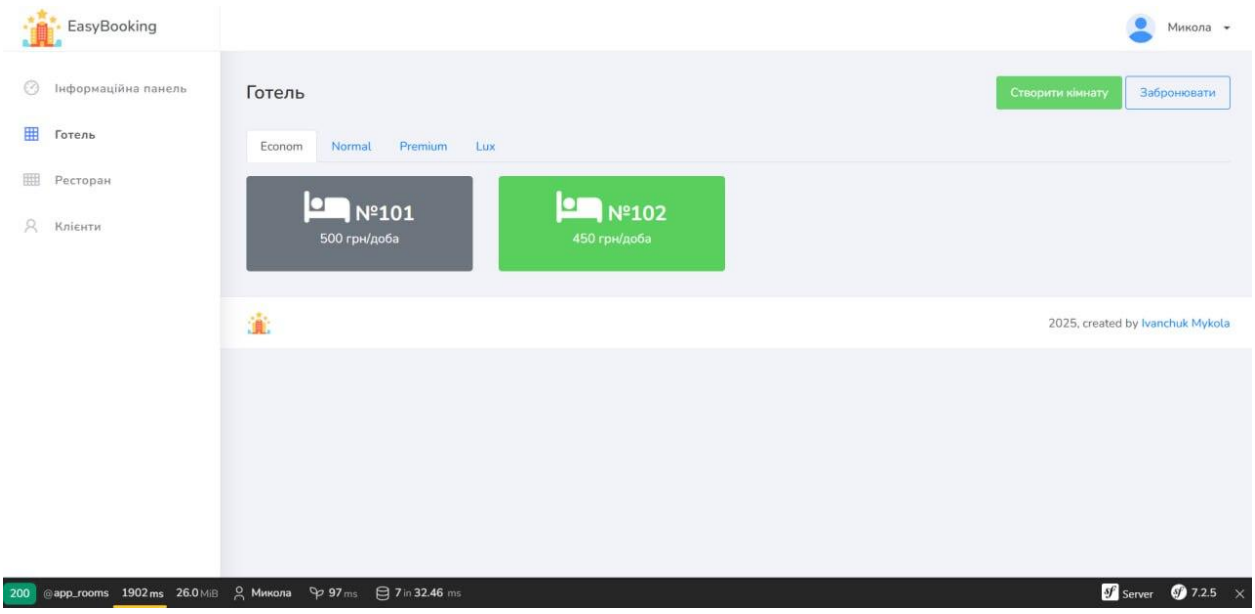

Рис. 17 Сторінка “Готель”

Натиснувши кнопку створення кімнати, перед нами відкриється форма для введення даних. Вводимо всі необхідні дані в поля, натискає підтвердити, і кімната успішно створюється (Рис. 18).

Рис. 18 Форма створення номеру

Після цього ми можемо забронювати кімнату, вибравши зі списку клієнтів та зі списку доступних вільних кімнат (Рис. 18).

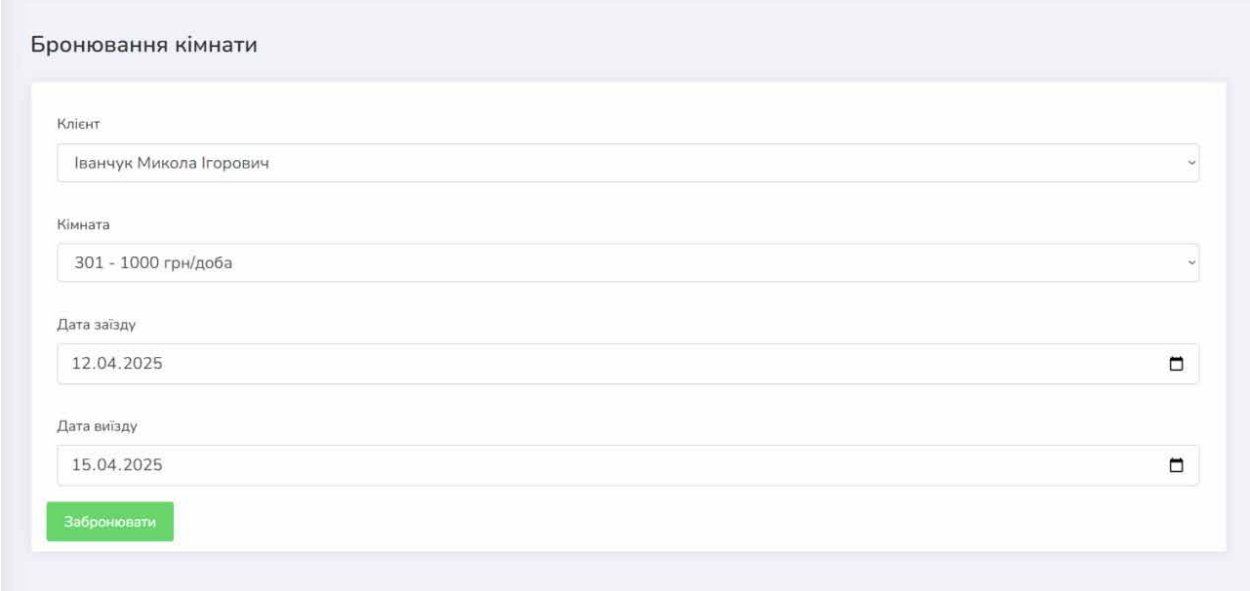

Рис. 19 Форма бронювання номеру

Відкриваємо сторінку ресторану. Тут у нас також представлені столики, на них вказано номер, розташування та кількість місць. Якщо столик зелений то він вільний, якщо сірий він зайнятий (Рис. 20).

Рис. 20 Сторінка “Ресторан”

Натиснувши кнопку створення столику, перед нами відкриється форма для введення даних. Вводимо всі необхідні дані в поля, натискаємо підтвердити, і столик успішно створюється (Рис. 21).

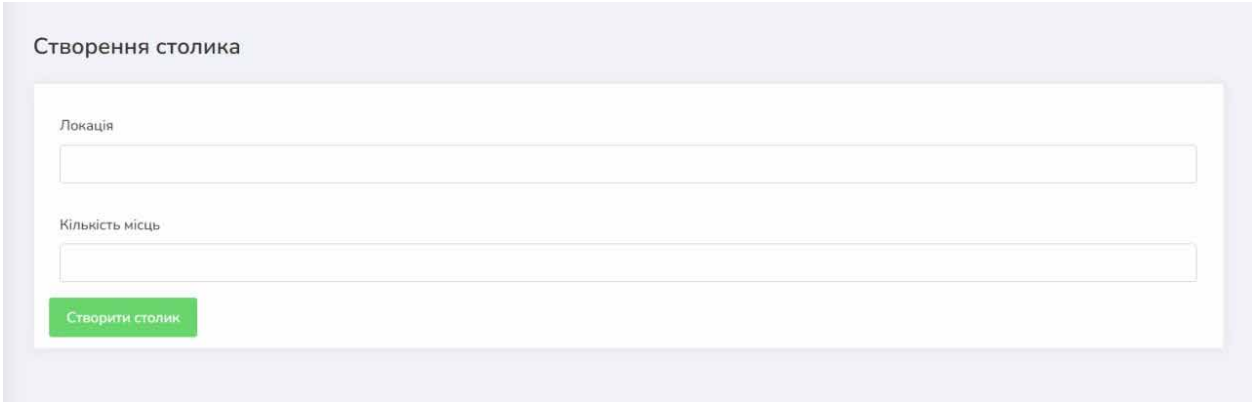

Рис. 21 Форма створення столику

Після цього ми можемо забронювати столик, вибравши зі списку клієнтів та зі списку доступних вільних столів (Рис. 22).

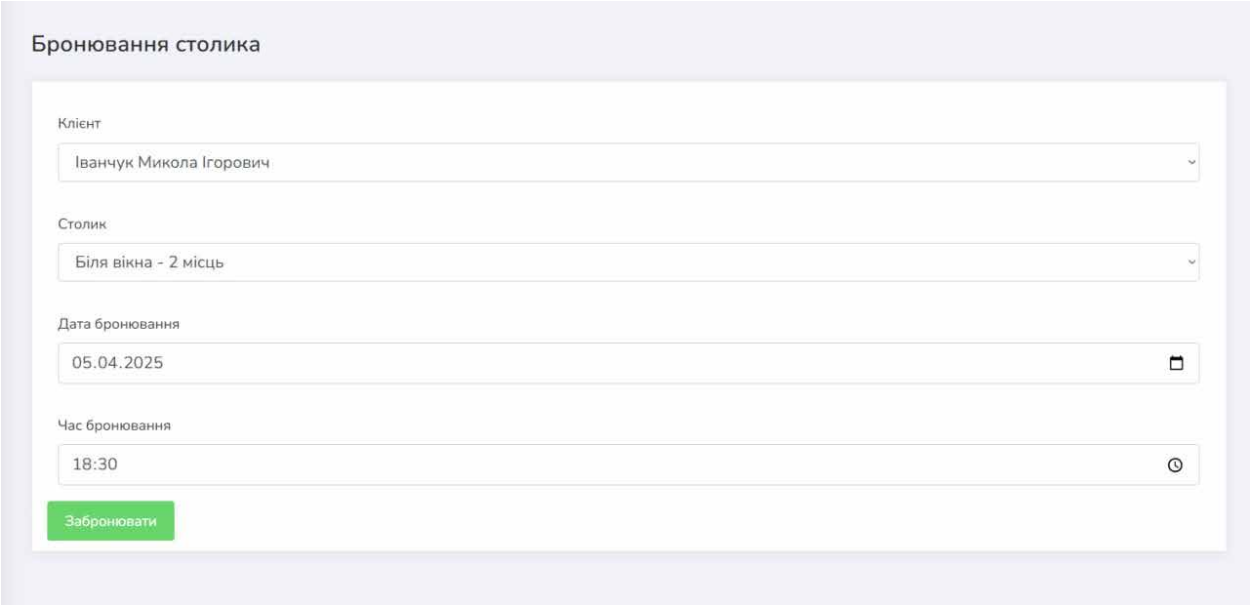

Рис. 22 Форма бронювання столику

Ще ми можемо натиснути на зайнятий столик і подивитися актуальну інформацію щодо його бронювання, дізнавшись на який час він зайнятий (Рис. 23).

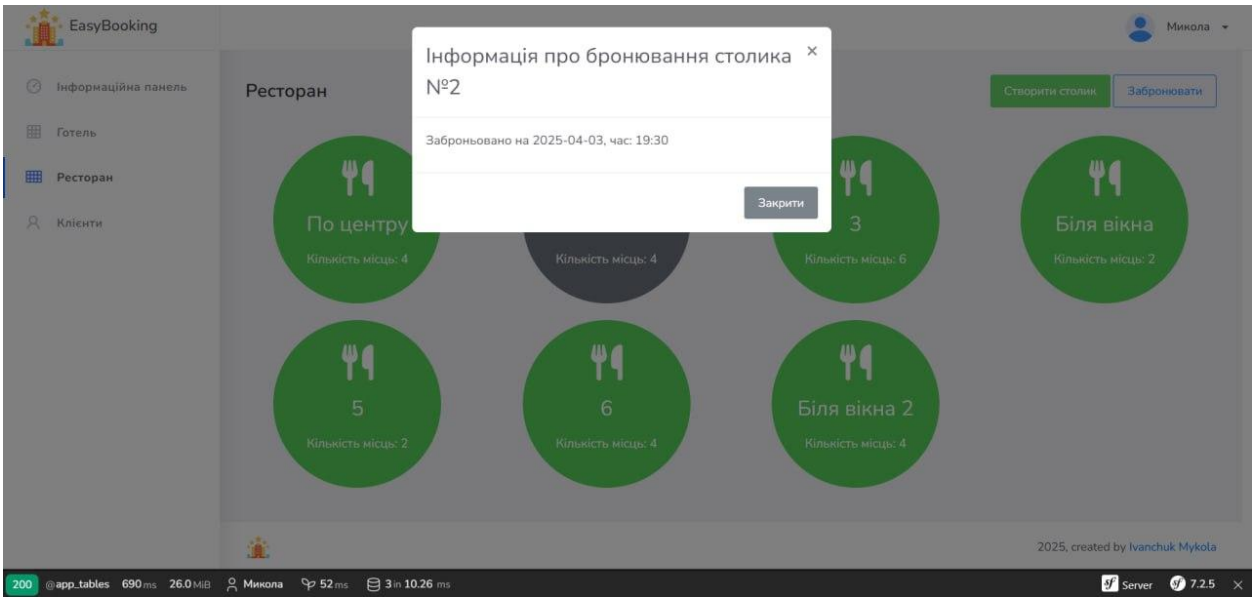

Рис. 23 Вікно з обраним столиком

На рис. 24 представлена форма авторизації в системі, яку необхідно заповнити при першому вході в систему.

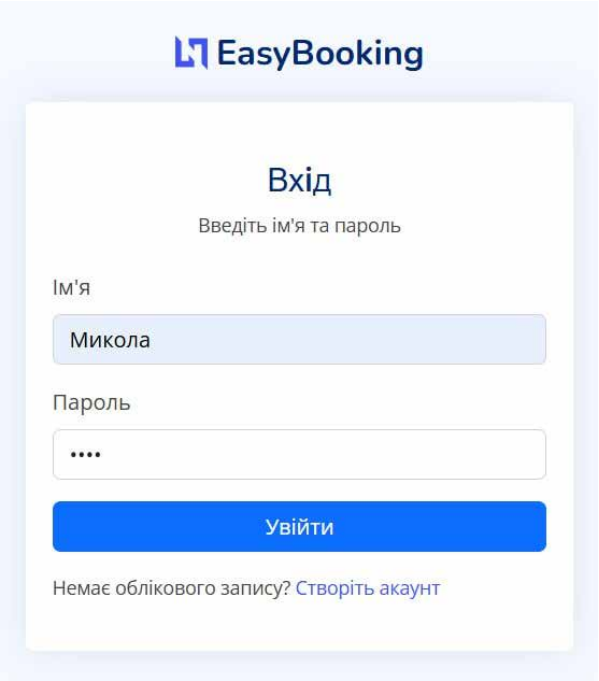

Рис. 24 Форма авторизації

Наступним кроком є реєстрації в системі – створення профілю користувача та заповнення його особистими даними. Ця форма представлена на рис. 25.

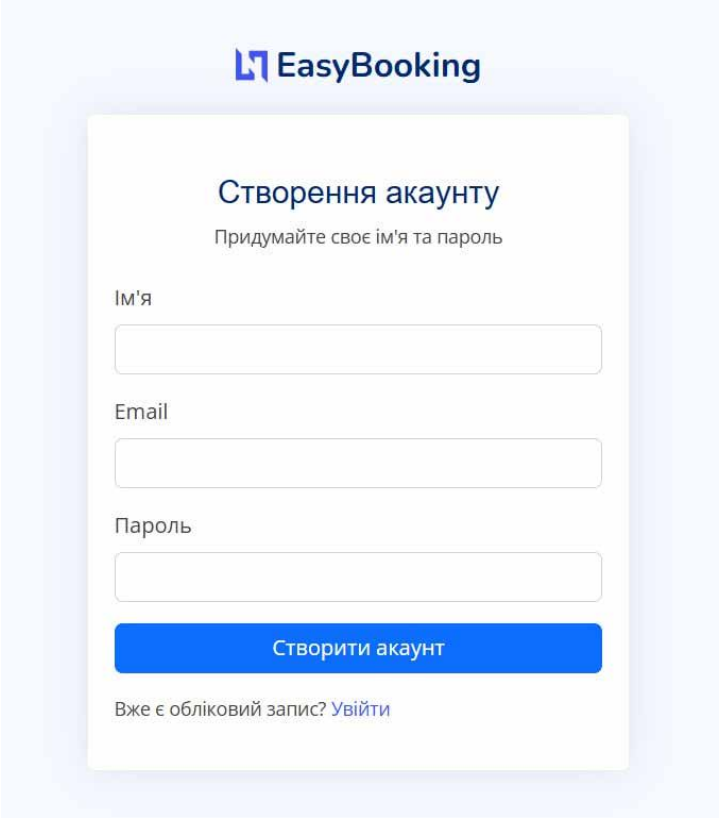

Рис. 25 Форма реєстрації

ВИСНОВКИ

Розвиток інформаційної системи бронювання в готельно-ресторанному бізнесі має важливе значення для підвищення ефективності роботи, підвищення рівня задоволеності клієнтів і оптимізації процесу бронювання. Протягом цього дослідження ми досліджували різні аспекти системи, включаючи її архітектуру, вибір відповідних технологій і критичні апаратні та програмні вимоги, необхідні для її успішного впровадження.

Архітектура системи була розроблена як багаторівнева структура, що охоплює рівень презентації, рівень додатків і рівень даних. Такий розподіл завдань сприяє кращій організації та зручності обслуговування, одночасно забезпечуючи безперебійний потік інформації між різними компонентами. Використовуючи зручний інтерфейс, система дозволяє клієнтам і адміністраторам легко переміщатися, підвищуючи загальний досвід користувача.

Вибір таких технологій, як PHP, Symfony, MySQL і Doctrine ORM, довів свою перевагу для створення надійної та масштабованої програми. Ці інструменти не лише підтримують ефективне керування даними, але й сприяють передовому досвіду розробки програмного забезпечення. Інтеграція сучасних веб-технологій, таких як HTML, CSS і JavaScript, додатково сприяє створенню інтерактивного та чутливого інтерфейсу користувача, забезпечуючи доступність на різних пристроях.

Встановивши чіткі вимоги до обладнання та програмного забезпечення, ми заклали міцну основу для продуктивності та безпеки системи. Акцент на використанні надійного апаратного забезпечення в поєднанні з перевагами програмного забезпечення з відкритим вихідним кодом забезпечує економічно ефективне рішення, яке відповідає потребам як малого, так і середнього бізнесу в секторі гостинності.

Просуваючись вперед, важливо враховувати потенціал для майбутніх удосконалень інформаційної системи. Включення розширених функцій, таких як аналітика даних, керування взаємовідносинами з клієнтами (CRM) і підтримка мобільних додатків, може ще більше підвищити якість роботи користувачів і надати цінну інформацію для бізнес-операцій. Використовуючи аналіз даних, система може аналізувати поведінку та вподобання клієнтів, дозволяючи менеджерам готелів і ресторанів приймати обґрунтовані рішення щодо маркетингових стратегій, ціноутворення та пропозицій послуг. Крім того, інтеграція модуля CRM може сприяти кращому спілкуванню з клієнтами, дозволяючи компаніям будувати міцніші стосунки та підтримувати лояльність завдяки персоналізованому досвіду.

Підсумовуючи, можна сказати, що інформаційна система бронювання в готельно-ресторанному бізнесі є значним кроком до модернізації діяльності та покращення надання послуг. Завдяки ефективному управлінню бронюванням, платежами та взаємодією з клієнтами система не тільки покращує ефективність роботи, але й створює позитивний досвід для користувачів. Оскільки індустрія гостинності продовжує розвиватися, впровадження таких інноваційних рішень матиме вирішальне значення для компаній, які прагнуть залишатися конкурентоспроможними на динамічному ринку. Висновки та рекомендації, викладені в цій дисертації, можуть слугувати цінним довідником для майбутнього розвитку та вдосконалення системи бронювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Booking – [Електронний ресурс] – Режим доступу: https://www.booking.com
2. OpenTable – [Електронний ресурс] – Режим доступу: https://www.opentable.com
3. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ua/glossary/mnogourovnevaya-arhitektura/
4. The Clean Architecture – [Електронний ресурс] – Режим доступу: https://medium.com/clean-code-channel/clean-architecture-the-solution-to-have-a-reusable-flexible-and-testable-code-ac7e296d1a75
5. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
6. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
7. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektuvannya-er-diagrammy/
8. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/fd_use-case-diagram.html
9. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektuvannya -use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

10. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
11. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
12. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/gpg_class-diagram.html
13. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ua/uml-classes/
14. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
15. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
16. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.
17. W3Schools – [Електронний ресурс] – Режим доступу: www.w3schools.com
18. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
19. What is Component Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
20. Deployment Diagram in UML: Definition, Examples & Components – [Електронний ресурс] – Режим доступу: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html

21. What is a data flow diagram? – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram
22. Лі, С. (2020). «Схеми придбання електронних книг: погляд на схожість і поведінку читачів». Дослідження в галузі бібліотечної та інформаційної науки, 42 (2), 135-150.
23. Chen, J., & Zhan, Y. (2018). «Вплив систем онлайн-бронювання на задоволеність клієнтів у готельній індустрії». Journal of Hospitality Marketing & Management, 27(5), 573-590.
24. Гійє, Б. Д., Ку, Д. (2019). «Роль технологій у бронюванні номерів у готелях: дослідження впливу платформ онлайн-бронювання». Міжнародний журнал готельного менеджменту, 76, 143-150.
25. Квортнік, Р. Дж., і Томпсон, Г. М. (2020). «Об'єднання маркетингу послуг і операцій із управлінням досвідом обслуговування». Журнал дослідження послуг, 23 (1), 1-18.
26. Іванов, С., Вебстер, К. (2019). «Переосмислення ролі технологій в управлінні готелем: уроки Airbnb». Журнал технологій гостинності та туризму, 10 (1), 35-49.
27. Барроуз К. В. та Пауерс Т. (2018). Введення в індустрію гостинності. Джон Вайлі та сини.
28. Пун, А., Лоу, К. (2017). «Вплив електронної комерції на індустрію гостинності: дослідження онлайн-туристичних агентств». Міжнародний журнал досліджень туризму, 19(6), 703-710.
29. Сігала, М. (2020). «Соціальні медіа та залучення клієнтів в індустрії гостинності». Журнал технологій гостинності та туризму, 11 (1), 125-143.
30. Бухаліс, Д., Ло, Р. (2018). «Прогрес в інформаційних технологіях і менеджменті туризму: 20 років і 10 років після Інтернету — стан досліджень електронного туризму». Управління туризмом, 29 (4), 609-623.

31. Чжа, К., і Цю, Х. (2021). «Дослідження ролі мобільних технологій у покращенні досвіду клієнтів у секторі гостинності». Journal of Hospitality Marketing & Management, 30(3), 305-321.
32. О'Коннор, П. (2019). «Еволюція онлайн-туристичних агентств: огляд їхнього впливу на готельну індустрію». Журнал дослідження подорожей, 58 (1), 28-41.
33. Лян С. та Ван Ю. (2020). «Вплив стратегій цифрового маркетингу на залучення клієнтів у готельну індустрію». Журнал менеджменту гостинності та туризму, 45, 123-132.
34. Hsu, C. H. C. & Huang, S. (2018). «Вплив відгуків клієнтів онлайн на наміри бронювання готелів: перспектива соціального впливу». Journal of Hospitality Marketing & Management, 27(5), 635-652.