МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проєктування»

на тему: «Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел»

Здобувача групи ІТ.м-22 (шифр групи) Глуховцова Дмитра Олександровича (прізвище, ім'я, по батькові)

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

_____ (підпис) Дмитро ГЛУХОВЦОВ (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доц. Вікторія АНТИПЕНКО (посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) _____ (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проєктування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Глуховцову Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи правила бронювання номерів, обов'язки персоналу туристичного готелю

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, проєктування структури web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел, розробка даної web-орієнтованої системи, тестування web-орієнтованої системи

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, мета та задачі дослідження, огляд існуючих аналогів, результати проведеного аналізу аналогів, функціональні вимоги, засоби реалізації, структурно-функціональне моделювання, моделювання варіантів використання, архітектура web-орієнтованої системи, схема реалізованої бази даних засобами PostrgeSQL, програмна реалізація, демонстрація роботи програмного продукту, тестування, апробація, висновок

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка специфікації	20.09.2023	
2	Розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел	20.11.2023	
3	Тестування програмного продукту	16.11.2023	
4	Впровадження в дію web-орієнтовану систему	01.12.2023	
5	Налаштування правильної роботи web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел	01.12.2023	

Магістрант _____ Дмитро ГЛУХОВЦОВ

Керівник роботи к.т.н., доц. Вікторія АНТИПЕНКО

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 58 найменувань, 3 додатки. Загальний обсяг роботи − 118 сторінок, у тому числі 76 сторінок основного тексту, 7 сторінок списку використаних джерел, 35 сторінок додатків.

Актуальність роботи полягає в тому, що більшість процесів переходить в онлайн. Особливо критичним це починає бути у випадку залучення допоміжних онлайн-систем для певних компаній чи бізнесу. Оскільки стає дедалі складно слідкувати за кожною з інтеграцій. Не є виключенням і організація робочих процесів для туристичного готелю. Наприклад, дані про гостей надходять із одного ресурсу, бронювання з іншого, а додаткова інформація для адміністрування – із третього. Усе це потрібно зібрати до купи та синхронізувати для подальшої обробки та аналізу. Саме в такій ситуації може допомогти web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних. Її використання забезпечить підвищення ефективності виконання робочих процесів даного закладу та зменшить навантаження на його персонал. Тому розробка запропонованої web-орієнтованої системи є актуальною.

Мета роботи: розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел. Її використання забезпечить продуктивну роботу адміністратора за рахунок її належної організації. У ході її реалізації було проведено детальний аналіз предметної області та розглянуто вже існуючі подібні програмні продукти. Сформовано функціональні вимоги до даної web-орієнтованої системи. Обрано програмні засоби для її реалізації. Проведено структурно-функціональне

моделювання. Спроєктовано та створено базу даних. Програмно реалізовано описану структуру та забезпечено роботу розроблених модулів. Проведено тестування отриманого програмного продукту.

Результатом даного дослідження є розроблена web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел. Її використання дозволить зменшити час, що витрачає адміністратор на виконання рутинних процесів, надасть механізм перегляду статистики в одному місці, допоможе у формуванні завдань для обслуговуючого персоналу, а також представить гостям функціонал для зручного бронювання номерів, спростить збір інформації для належної організації робочих процесів та аналізу прибутковості.

Результати представленого дослідження були апробовані на науково-практичній конференції «КІТ-2023» у Харківському національному автомобільно-дорожньому університеті в м. Харків

Ключові слова: WEB-ОРІЄНТОВАНА СИСТЕМА, ТУРИСТИЧНИЙ ГОТЕЛЬ, СИНХРОНІЗАЦІЯ ДАНИХ, БРОНЮВАННЯ, DASHBOARD, REACT, NODE.JS.

ЗМІСТ

ВСТУП

Актуальність. Сьогодні світ рухається до тотальної цифровізації та переведення якомога більше процесів в онлайн. Це безумовно відображається на майже кожній сфері сучасного життя. Особливо критичним це є для ситуацій, коли обробка інформації для певної компанії чи бізнесу відбувається за допомогою багатьох допоміжних онлайн-систем і стає складно слідкувати за кожною з них. Як, наприклад, організація робочих процесів для туристичного готелю. Дані про гостей можуть надходити з різних ресурсів, інформація, яка може бути корисною адміністратору – із третього джерела. І все це треба зібрати до купи для обробки й аналізу. У такій ситуації може допомогти web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних. У подальшому це безумовно приведе до позитивних змін у роботі такого закладу, оскільки підвищить її ефективність та зменшить навантаження на персонал. Тому дане дослідження є актуальним сьогодні.

Тема. Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел.

Мета. Розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел.

Для досягнення мети даного проєкту потрібно виконати наступні задачі:

- визначити актуальність та цільову аудиторію розроблюваної web-орієнтованої системи;
- провести детальний аналіз предметної області та огляд сучасних публікацій, пов’язаних із проблематикою поставленої задачі;
- виконати аналіз існуючих продуктів аналогів, ідентифікувати їхні переваги та недоліки;

- визначити функціональні можливості запропонованої web-орієнтованої система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел та технології, які будуть використовуватися для її розробки;
- виконати структурно-функціональне моделювання.
- програмно реалізувати описану структуру та забезпечити роботу модулів, необхідних для забезпечення продуктивної роботи web-орієнтованої системи;
- провести тестування отриманої web-орієнтованої системи для забезпечення якості розробки.

Об’єкт дослідження. Процес адміністрування та оперування інформацією при управлінні туристичним готелем.

Предмет дослідження. Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел.

Практична новизна. Критично важливо для готельного бізнесу мати потужний інструмент, використання якого дозволить більш ефектично організовувати внутрішні процеси та допомагати персоналу краще виконувати свою роботу. Використання запропонованого програмного продукту надасть можливість спростити обов’язки адміністратора та локалізувати їх лише в одній web-орієнтованій системі. У свою чергу моніторинг одразу декількох джерел даних дозволить більш детально аналізувати ринок та знаходити оптимальні рішення в організації задач бізнесу. Також застосування такої web-орієнтованої системи буде корисним для обслуговуючого персоналу, оскільки надасть можливість постійного доступу до актуальної інформації щодо наступних завдань на виконання відповідним працівникам.

Результати роботи були апробовані на всеукраїнській науково-практичній конференції «КІТ-2023» у Харківському національному автомобільно-дорожньому університеті .

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Кожен бізнес безумовно прагне покращити більшість робочих процесів, що однозначно позитивно вплине на прибутки та збільшить комфорт для власних співробітників, а також для зовнішніх користувачів.

Так, наприклад, індустрія гостинності є однією з найбільш перспективних галузей глобальної економіки. У [1-3] було досліджено сучасний її стан, виклики, що постають перед нею в останні роки, а також можливості для подальшого розвитку та потенціал застосування цифровізації як інструменту для покращення та зростання з ефективним використанням засобів сучасності.

Не виключенням є й готельно-ресторанний бізнес, як частина вищезгаданої індустрії. Він входить в число найприбутковіших та таких, на які є попит сьогодні. Це означає, що застосування потужної системи для адміністрування та менеджменту подібних закладів, як представників даного бізнесу, є досить актуальним. А способи досягнення таких цілей є часто обговорюваними [4,5]

Також варто відмітити широковідомий факт, що діджиталізація будь-якого бізнесу – це крок до зростання прибутковості через трансформацію та автоматизацію більшості рутинних процесів всередині нього [6,7]. Це питання є досить актуальним сьогодні. Не є секретом, що прибутковість підприємства, чи як у даному випадку туристичного готелю, є одним із головних чинників його рентабельності. Цій проблематиці присвячено велику кількість статей та досліджень. Наприклад у [8], висвітлюється розробка стратегії управління прибутком. Вона побудована на ціні, можливостях розселення в готелі та інших чинниках. Останні є важко досяжними без використання засобів автоматизації робочих процесів і систематизації даних.

Готельний бізнес орієнтований не лише на прибуток. Це більше про комунікацію людей. Досить важливим є завдання вибудовувати саме

клієнтоцентричну систему, спрямовану на задоволення саме кінцевого отримувача послуг[9-11]. Також зрозумілим є те, що зі зростанням можливостей комп'ютерних технологій і змінами в суспільстві збільшуються й вимоги та побажання споживачів до послуг, які їм надаються, та їхньої якості [12,13]. Саме тому необхідно максимально звужувати прірву між гостями та адміністраторами, готелем та всіма внутрішніми процесами Це означає зменшувати різницю між потребами та можливостями. Цьому звісно може допомогти використання web-орієнтованої системи автоматизації внутрішніх процесів закладу та систематизації релевантної йому інформації [14,15].

Однак гості готелю – це не єдині його клієнти, роботу з якими треба належним чином організувати. Завжди варто враховувати й весь персонал. Управління співробітниками туристичного готелю є невід'ємною частиною робочих процесів [16,17]. Йому треба приділити велику увагу також. Модуль для управління персоналом може покращити досвід співробітників, забезпечивши легкий доступ до актуальної інформації щодо їх поточних завдань і послуг. Також його використання дозволить зменшити навантаження на адміністраторів. Сучасні дослідження прийшли висновку, що покращення загального управління якістю обслуговування позитивно впливає на підвищення корпоративної соціальної відповідальності та продуктивності роботи туристичного готелю [18,19].

Варто прийняти той факт, що сьогодні на ринку готельної індустрії існує досить велика кількість готових рішень, інтегрованих систем та інших продуктів. Однак проблеми розширюваності, обслуговування, ціни, безпеки тощо є актуальними для кожної окремої одиниці бізнесу, яка має намір застосовувати в своєму активі такі технології [20].

Також варто відзначити, що не менш важливим є розуміння того, для чого саме система потрібна бізнесу й які вимоги вона повинна задовольняти. До основних функціональних компонентів зачасту можна віднести наступні:

- модуль бронювання та резервування номерів;
- модуль проведення розрахунів із клієнтами;

– модуль управління персоналом;

– модуль оптимізації прибутку тощо.

Також досить важливим є визначення масштабів, адже програмний продукт, який призначений для організації діяльності малого готелю може бути абсолютно непридатним для великого готельно-ресторанного комплексу [21].

Окрім детального аналізу відповідного функціоналу потрібно ясно розуміти, чим є автоматизована система управління (АСУ). Її основні характеристики є такими:

– інструмент та засіб управління, який дозволяє швидко реагувати на мінливі ситуації в готелі та на ринку;

– можливість покращення рівня обслуговування гостей та якості персоналу;

– система автоматизації всіх етапів взаємодії з гостями;

– інструмент кадрової політики, ефективного контролю дій окремих співробітників;

– нові можливості управління та обслуговування внаслідок отримання нових інструменті та даних [22]

Важливу роль у розробці такого програмного засобу відіграють інтеграції на різноманітних рівнях із глобальними системами бронювання. Наприклад, такими як Booking чи з системами туристичних агенцій (СТА). Подібні інтеграції можуть надавати додаткові канали продажів та реклами [23]. Це позитивно впливає на впізнаваність і прибутковість, адже певний бренд значно частіше перебуває у полі зору кінцевого споживача на просторах Всесвітньої мережі. Саме це було розглянуто в [24]. Однак у даній роботі продемонстровано й негативну сторону таких інтеграцій. Це високі процентні комісії від таких сервісів та недовіра клієнта до прямих продажів через власний сайт тощо. У свою чергу дослідження проведене в [25] допомагає зрозуміти на прикладі конкретних готелей та даних, що навіть знаючи про недоліки додаткових каналів, вони є надзвичайно корисними для збільшення прибутку, якщо користуватися ними в раціональних межах. Отже, це демонструє справжню

необхідність в можливостях опрацювання інформації з різних джерел та її зберігання для подальших маніпуляцій.

Дані в подібних системах є одним із найважливіших компонентів. Без їхньої консистентності неможлива коректна робота всієї подібного інструменту. У свою чергу це призведе до негативного досвіду використання останнього [26]. Наприклад, частою проблемою є ситуація, коли користувач заповнив форму для резервування номеру, підтвердив її, але виникла колізія. На ці дати вже з іншого каналу надійшов запит на бронювання даної кімнати. Однозначно така ситуація викличе негативні емоції потенційних відвідувачів та бажання знову користуватися таким сервісом зникне [27]. У подібних випадках гостро постає проблема синхронізації всіх даних наряду з актуалізацією інформації, наявної на стороні споживача в його роботі з зовнішніми системами. Підтримка таких активностей у ручному режимі досить часто, як можна зрозуміти, є майже нереальною та може спровокувати безліч помилок [28].

Подібні системи зазвичай розробляються на базі web-інструментів та їхніх можливостей. Такий підхід може надати низку наступних переваг:

– простота доступу до інструмента: використання будь-якою людиною, яка має доступ до мережі Інтернет;

– простота розгортання (установки): для подібної web-орієнтованої системи немає необхідності фізичної установки на комп’ютер користувача;

– забезпечення автоматичними оновленнями;

– високий рівень надійності мережевих з’єднань та web-технологій [29];

– точність та інформативність даних, як запорука довіри клієнта [30].

На основі всього вищезазначеного можна зробити висновок, що питання створення та використання web-орієнтованих систем підтримки діяльності туристичних готелей є досить актуальним у наш час. Також варто зауважити, що на них є доволі високий попит і представники даного бізнесу зацікавлені у їх застосуванні. А тому доцільним є розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел.

Результати даного дослідження було апробовано на конференції «Комп’ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» [31].

1.2 Аналіз продуктів-аналогів

У мережі Інтернет все частіше стають доступними різні онлайн інструменти, які спрямовані на автоматизацію робочих процесів готелю. Їхній функціонал постійно збільшується. Однак, як це завжди буває, вони можуть не містити певні специфічні можливості для конкретних потреб малого або великого бізнесу.

Для формування вимог створюваного програмного продукту було розглянуто та досліджено існуючі аналоги подібних web-орієнтованих систем. А саме «SERVIO HMS» [32], «EasyPMS» [33] та «HotelFriend» [34].

«SERVIO HMS» – це програмний інструмент із web-інтерфесом для здійснення ефективної автоматизованої підтримки діяльності підприємств готельного бізнесу та об’єднання усіх функціональних компонентів у єдину інформаційну мережу.

Після завантаження головної сторінки користувач, а саме адміністратор, потрапляє на вкладку з так званою «шахматкою», де представлено дані про всі наявні номери та бронювання. Детальна інформація щодо останніх стає доступною після їх обирання (рис. 1.1).

Також доступною є можливість створення нового заселення, просто протягнувши необхідний відрізок та заповнивши певні дані (рис. 1.2). Або через спеціальне окреме вікно (рис. 1.3).

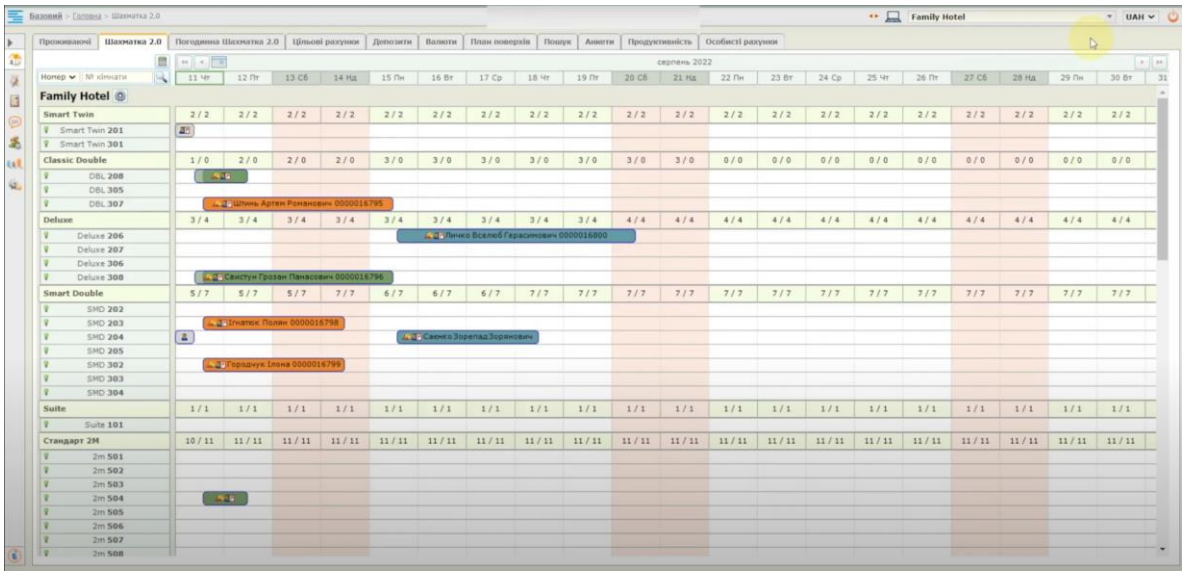


Рисунок 1.1 – Головна сторінка «SERVIO HMS»

Джерело: [32]

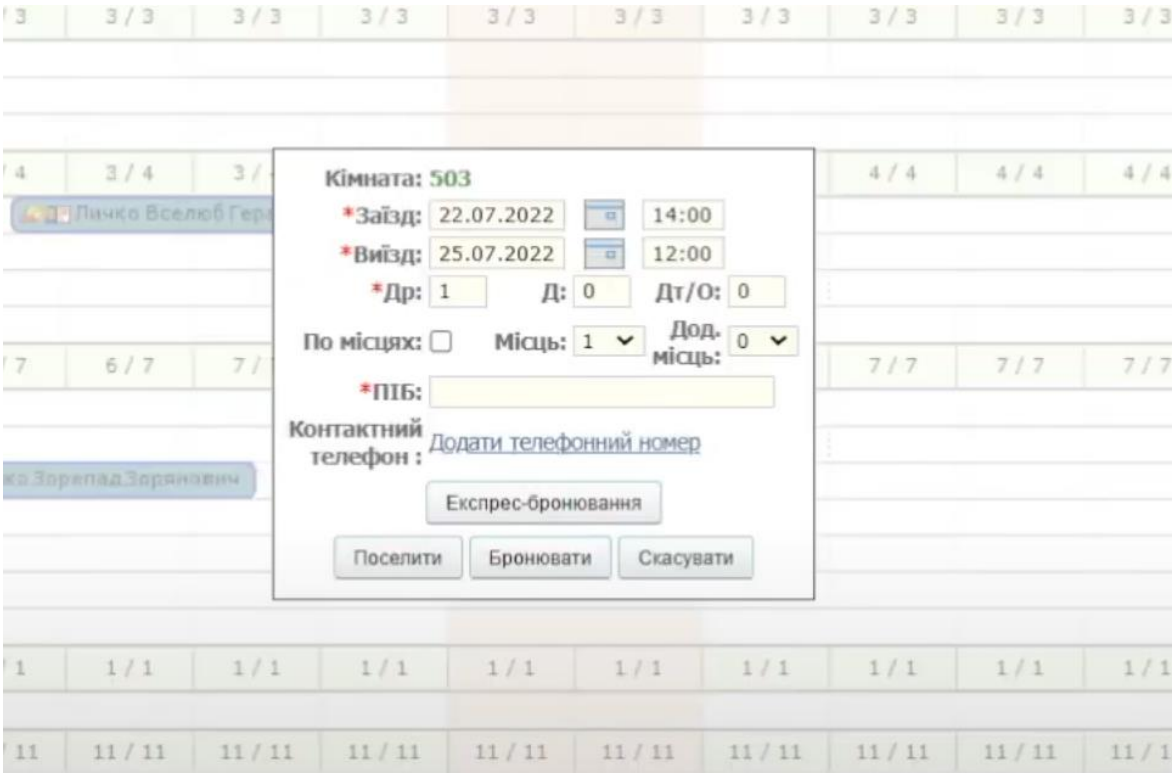


Рисунок 1.2 – Швидке створення нового бронювання

Джерело: [32]

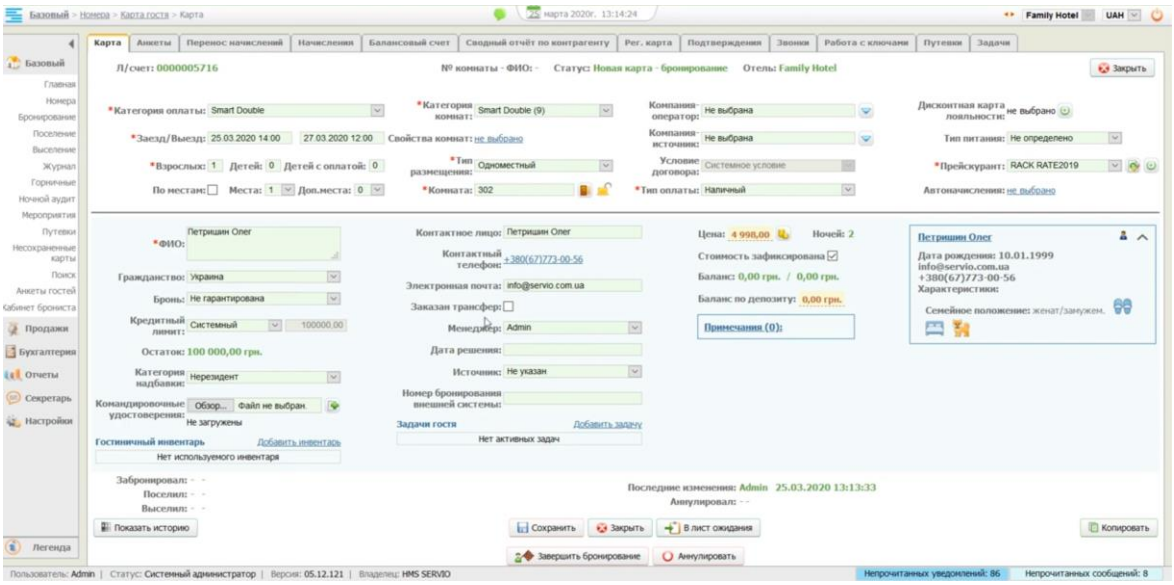


Рисунок 1.3 – Створення бронювання з більшою кількістю налаштувань

Джерело: [32]

Також до функціоналу відноситься можливість перегляду інформації про всіх наявних гостей готелю та різні маніпуляції з цими даними (рис. 1.4).

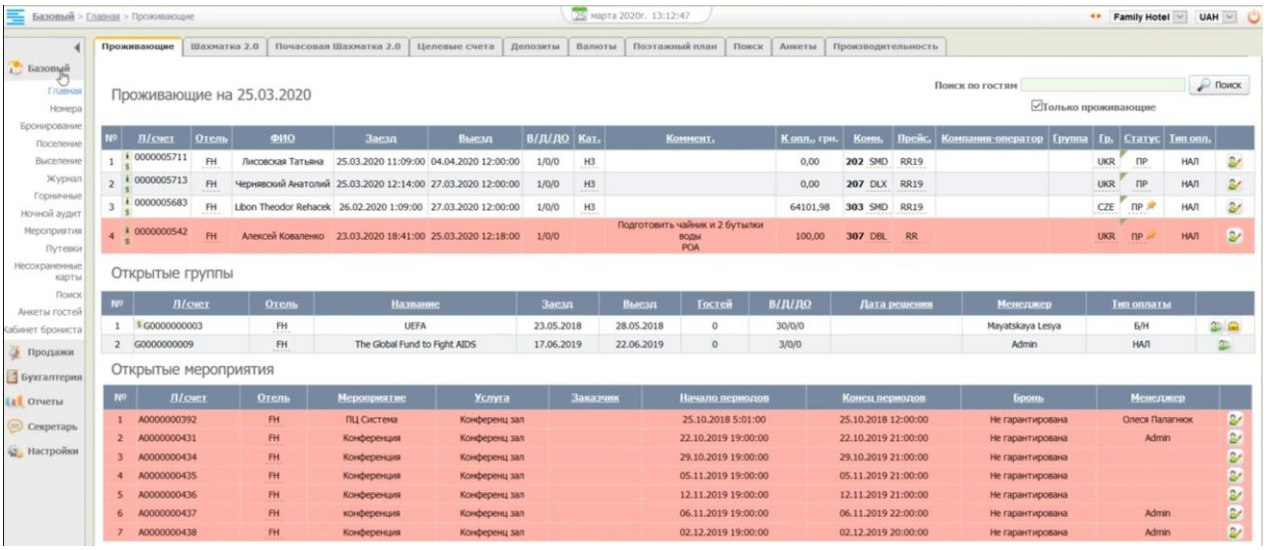


Рисунок 1.4 – Вікно для перегляду даних про поточних гостей

Джерело: [32]

Ще одним модулем даного програмного продукту є «Інструменти для покоївок». Він допомагає управляти процесом прибирання та надавати завдання

обслуговуючому персоналу для кращого догляду за номерами та виконанням побажань клієнтів (рис. 1.5).

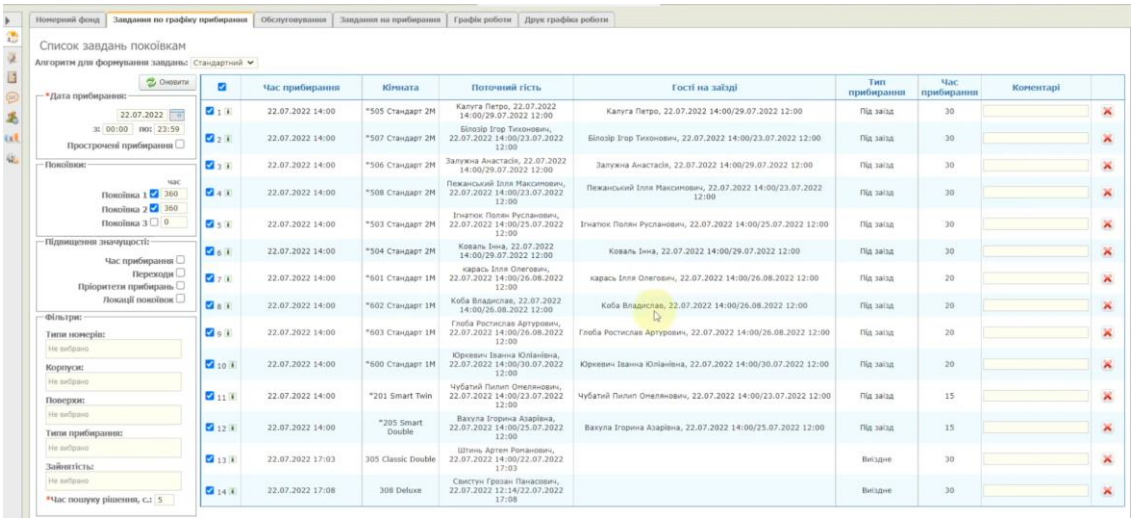


Рисунок 1.5 – Модуль управління процесом прибирання

Джерело: [32]

У «SERVIO HMS» наявними є окремі модулі, які дозволяють автоматизувати роботу бухгалтерії й управління номерним фондом. Також є ті, які надають можливість гнучкого налаштування численних параметрів як для клієнтів, так і для послуг. Додатково варто виділити модуль для побудови звітів, де є велика кількість різноманітних варіантів зведення звітів за певними категоріями.

Дизайн даного програмного продукту можна охарактеризувати як стриманий та приємний, але дещо застарілий. Інтерфейс доволі інтуїтивно зрозумілий, однак перенесичений: щоб розібратися зі специфікою використання даного інструменту, знадобиться певний час.

«EasyPMS» – хмарний web-додаток управління роботою готелю. Це Property Management System (PMS), яка включає в себе менеджер з’єднань, модулі замовлень, оцінювання, менеджменту та багато інших. Після завантаження web-додатку відкривається дашборд з основною статистичною інформацією, такою як наявні

кімнати та їхній статус, різноманітна класифікація відвідувачів та ін., поданою в графічному вигляді (рис. 1.6).

Рисунок 1.6 – Головна сторінка «EasyPMS»

Джерело: [33]

Також представлене рішення має можливість перегляду наявних резервувань у вигляді списку з повною інформацією та графічними позначенням. Це реалізовано для кращого розуміння та оцінки даних (рис. 1.7). Також є шахматка.

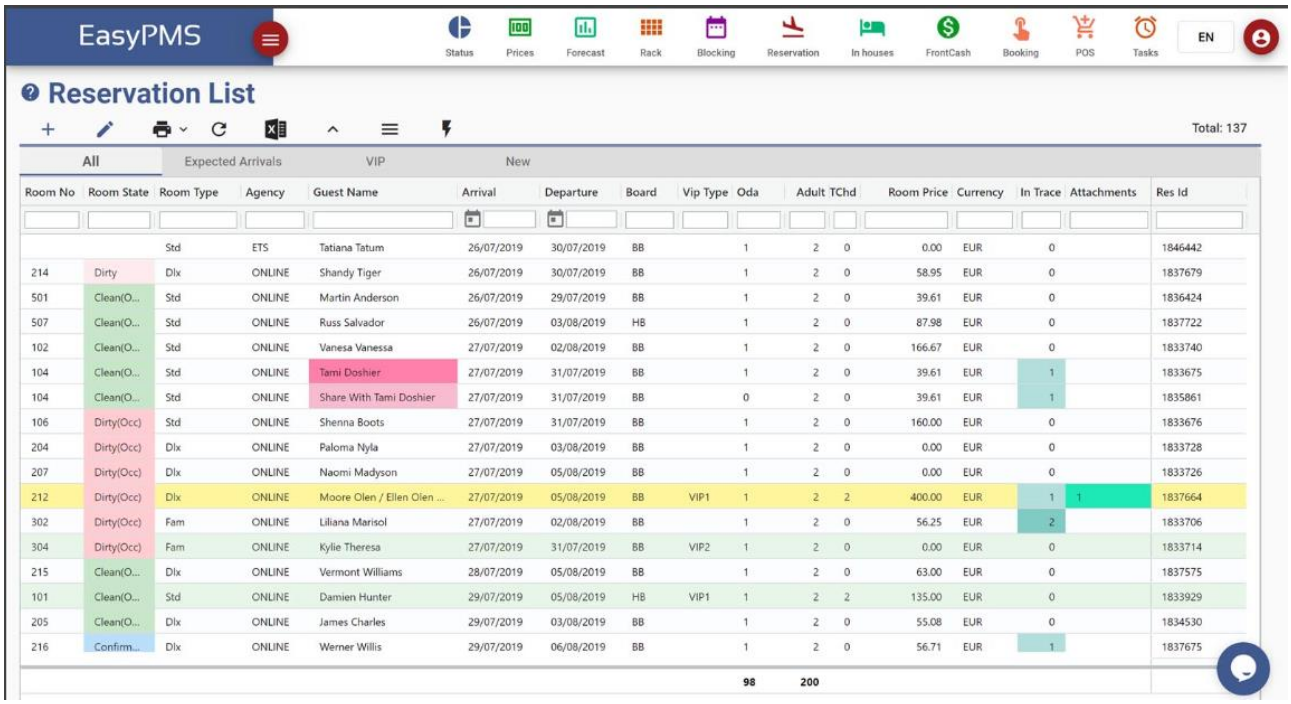


Рисунок 1.7 – Список поточних резервувань

Джерело: [33]

Є можливість заселення гостя вручну (рис. 1.8), але основна кількість бронювань надходить із зовнішніх систем або web-сайту готелю.

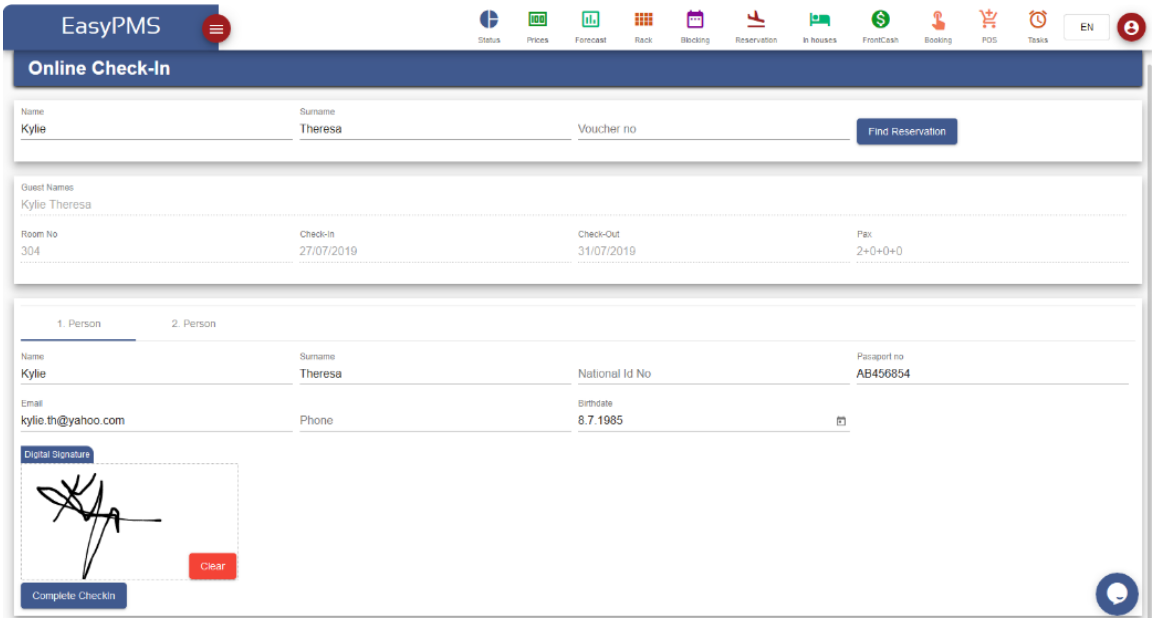


Рисунок 1.8 – Інтрефейс для заселення гостя вручну

Джерело: [33]

Модуль для букингу також доступний для використання і надає повний функціонал для користувачів, які мають бажання забронювати номер безпосередньо з web-сайту готелю (рис. 1.9).

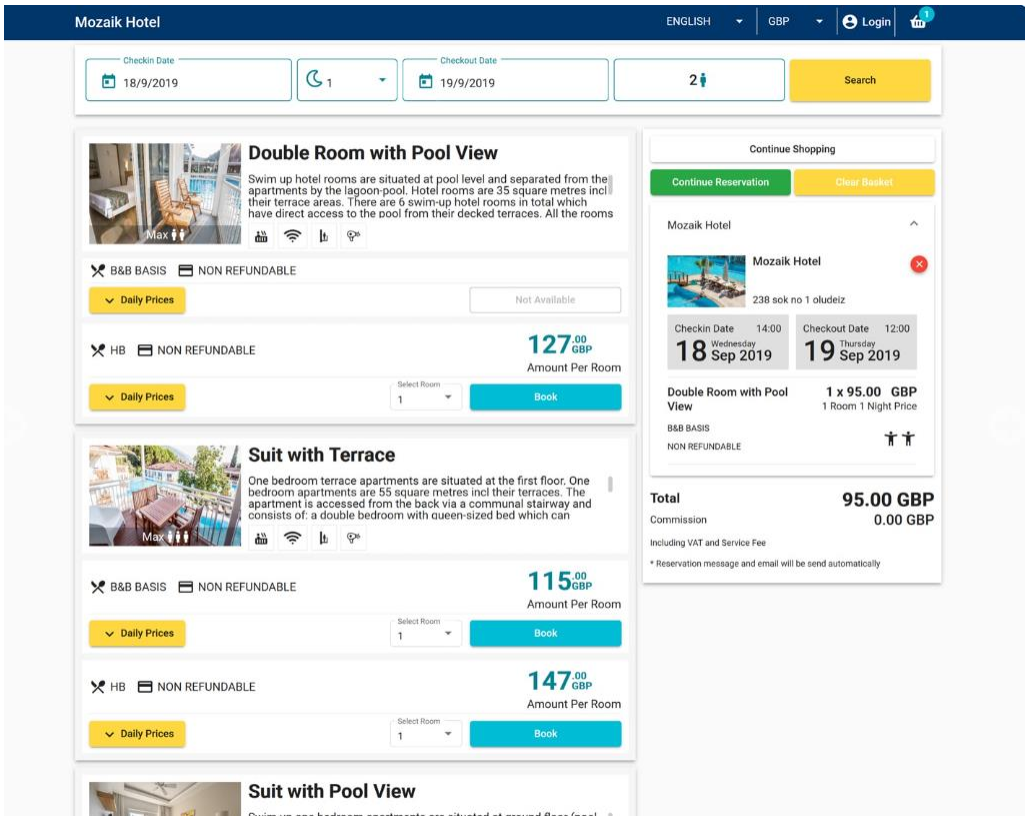


Рисунок 1.9 – Можливості букингу з використанням web-сайту «EasyPMS»

Джерело: [33]

Також є модуль для управління покоївками та їхніми завданнями для автоматизованого та більш якісного здійснення контролю за персоналом та номерним фондом (рис. 1.10).

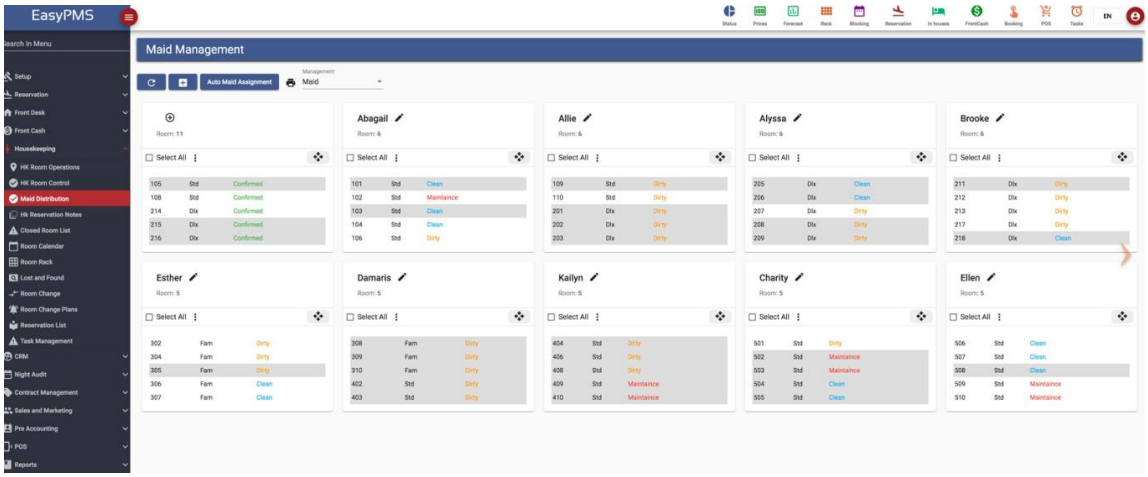


Рисунок 1.10 – Вікно модуля управління персоналом

Джерело: [33]

Дизайн даного web-додатку можна описати як сучасний із приємною кольоровою гаммою та такий, що в основному відповідає сучасним трендам. Інтерфейс є приємним і інтуїтивно зрозумілим. Однак функціонал також є дещо перенасиченим як і у випадку з першим аналогом.

«HotelFriend» – це ще один представник хмарного PMS web-додатку. Він надає доступ до всього необхідного інструментарію для забезпечення управління власним закладом із будь-якого місця, синхронізуючи інформацію з багатьох джерел та СТА. Функціонал цього web-додатку є найбільшим та найзмістовнішим серед досліджуваних аналогів.

Його головна сторінка дозволяє перегляд наявних бронювань і заселень у всьому номерному фонді, знову у формі так званої шахматки (рис. 1.11).

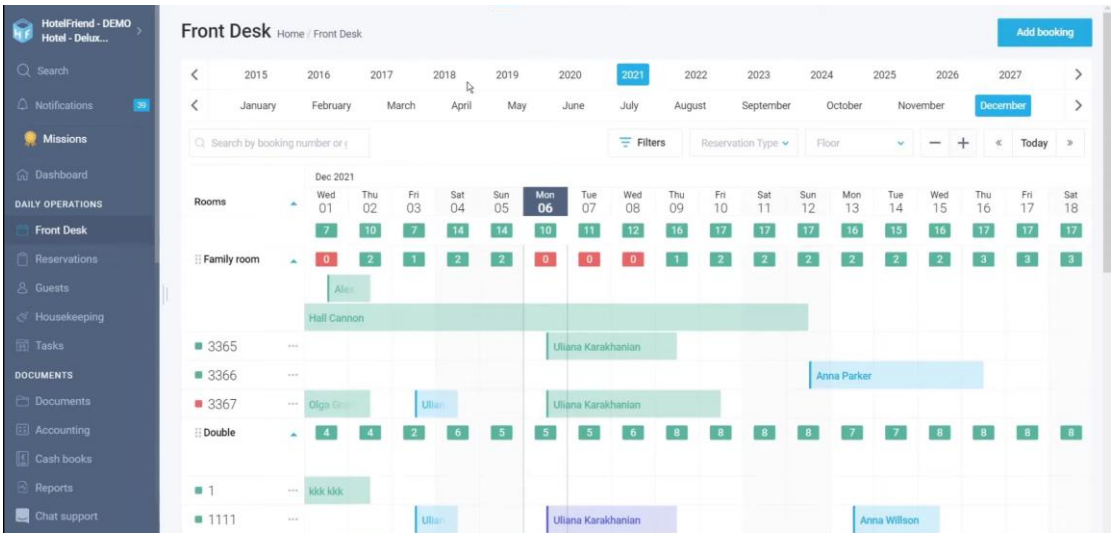


Рисунок 1.11 – Перегляд наявних заселень та бронювань у «HotelFriend»

Джерело: [34]

Також є можливість ручного додавання бронювання. При цьому враховується безліч параметрів та дозволяється заповнювати окремі поля автоматично, підтягуючи з історії та інформації про гостя (рис. 1.12).

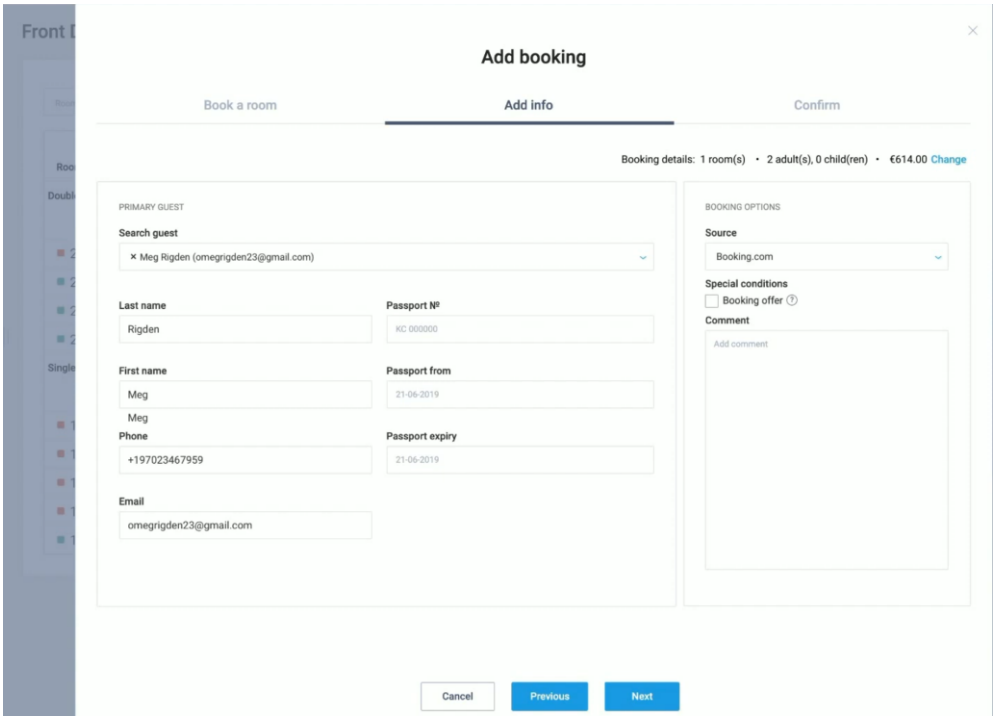


Рисунок 1.12 – Демонстрація ручного додавання бронювання

Джерело: [34]

Як і в інших досліджуваних аналогів, даний програмний продукт має модуль для організації роботи сервісу з прибирань та управління персоналом та номерами. Він включає в себе менеджер завдань та перегляд інформації про їх поточні статуси. Також є можливість для персоналу відмічати виконані ними задачі (рис. 1.13).

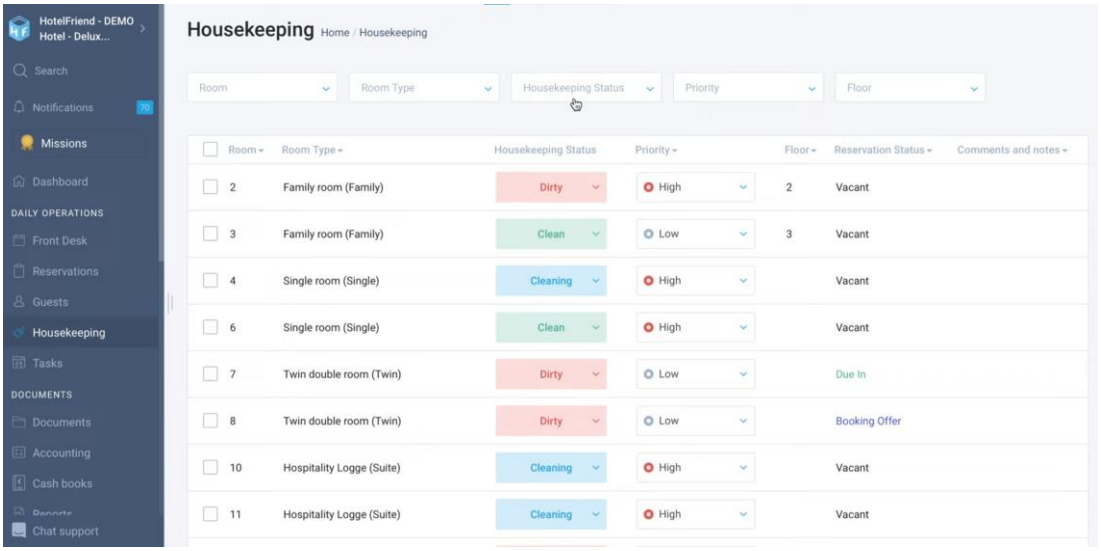


Рисунок 1.13 – Housekeeping модуль

Джерело: [34]

Доступною є функція побудови різноманітних звітів, які наочно відображають статистичні дані та спрощують роботу при їх зведенні (рис. 1.14).

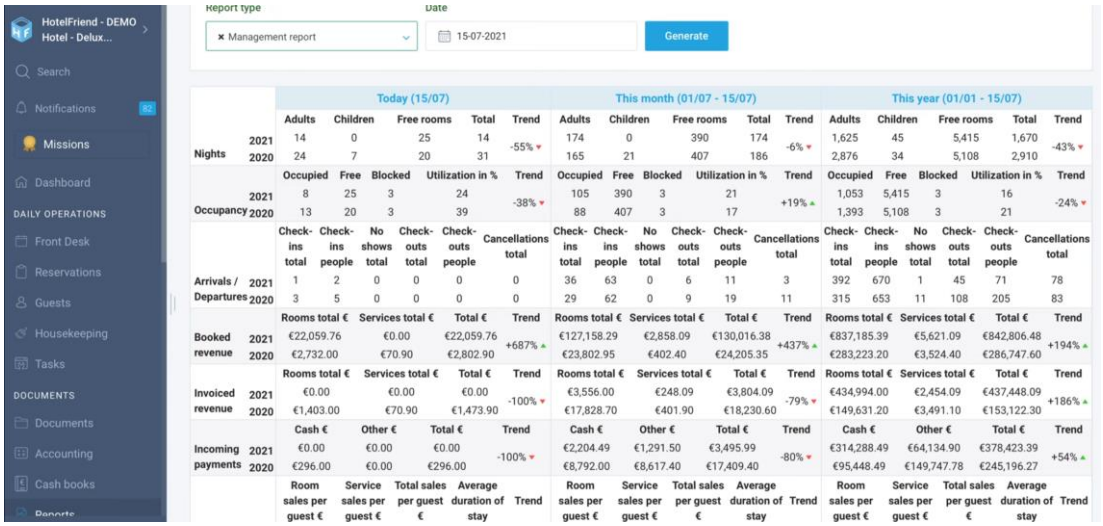


Рисунок 1.14 – Приклад звіту

Джерело: [34]

Дизайн даного web-додатку є сучасним. Його кольорове оформлення є досить приємним та має можливість індивідуального налаштування. Інтерфейс містить певні навчальні матеріали щодо особливостей його застосування, оскільки є вирогідність виникнення ситуації дезорієнтації для нового користувача.

Рельутати проведеного детального аналізу існуючих продуктів-аналогів зі схожими функціями та призначенням до розроблюваної web-орієнтованої системи представлено в таблиці 1.1.

Таблиця 1.1 – Результати порівняння продуктів-аналогів

Характеристика/Web-додаток	**«Servio HMS»**	**«Easy PMS»**	**«HotelFriend»**
Наявність основного функціоналу	+	+	+
Навігація	+	+	+
Авторизація користувачів	+	+	+
Перегляд статистики	-	+	+
Наявність модулю бронювання	-	+	+
Керування персоналом	+	+	+
Можливість персоналу самостійно контролювати виконання завдань	-	-	+
Сповіщення	+	+	+

Джерело: побудовано автором

Дані з таблиці 1.1 допомагають розібратися з основними моментами, на які необхідно звернути увагу під час проєктування структури та визначення вимог до функціоналу розроблюваної web-орієнтованої системи. У першу чергу треба врахувати позитивні моменти, які поєднують всі аналоги, та подолати виявлені їхні недоліки.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою даного дослідження є розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел. Цей програмний продукт повинен забезпечити належну організацію роботи адміністратора даного закладу та вдосконалити внутрішні робочі процеси за рахунок їхньої автоматизації.

Розроблена web-орієнтована система має задовольняти наступні функціональні вимоги:

- перегляд всіх кімнат у номерному фонді даного готелю
- перегляд вільних кімнат у задані гостем дати;
- можливість бронювання номеру на потрібну дату;
- управління бронюваннями з боку адміністратора (їх реєстрацію, редагування та створення нових);
- наявна для адміністратора можливість проведення маніпуляцій із даними про номерний фонд (редагування інформації, її видалення та додавання нової);
- перегляд статистики про прибутковість і популярність номерів, а також інформації про відвідувачів готелю у графічній формі (у діаграмах та таблицях);
- модуль для забезпечення управління та контролю роботи покоївок.

Також створювана web-орієнтована система повинна мати зручний та інтуїтивно зрозумілий інтерфейс. Із боку користувача це можуть бути підказки на формах тощо. Контент має завантажуватися швидко та бути актуальним. Доступ до певного набору функціонала має розмежовуватися за ролями й бути доступним лише для відповідної групи користувачів.

Для досягнення мети даного проєкту необхідно виконати наступні задачі:

- визначити актуальність та цільову аудиторію розроблюваної web-орієнтованої системи туристичного готелю із механізмом синхронізації даних із декількох джерел;
- провести детальний аналіз предметної області та огляд сучасних публікацій, пов’язаних із проблематикою поставленої задачі;
- виконати аналіз існуючих продуктів аналогів, ознайомитися з їхніми позитивними та негативними сторонами для кращого розуміння майбутньої структури власної системи;
- визначити функціональні можливості запропонованої web-орієнтованої система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел та технології, які будуть використовуватися під час її створення для того, щоб мати сучасний стек інструментів для розробки та проєктування;
- виконати структурно-функціональне моделювання процесів, які забезпечуються із застосуванням даного програмного продукту.
- програмно реалізувати описану структуру та забезпечити роботу модулів, необхідних для забезпечення продуктивної роботи web-орієнтованої системи;
- провести smoke-тестування отриманої web-орієнтованої системи для забезпечення якості розробки.

Також, на етапі формалізації мети проєкту було проведено планування його робіт, яке детально описане в Додатку А.

2.2Методи дослідження

Після постановки мети проєкту, визначення функціональних та нефункціональних вимог до розроблюваної web-орієнтованої системи, формування

переліку задач, які мають бути виконаними, варто визначитися з методами даного дослідження.

Загальнонаукові методи дослідження поділяють на три великі групи:

– методи теоретичного дослідження;
– методи емпіричного дослідження;
– загальні методи [35].

Теоретичний метод використовувався під час формалізації мети проєкту. За його допомогою було вибудувано логічний зв’язок та шлях від абстрактного до реального, щоб сформувати та систематизувати всю інформацію в єдине ціле.

До емпіричного методу зазвичай відносяться спостереження, порівняння тощо. Він використовувався під час порівняння продуктів-аналогів та визначення їх позитивних та негативних сторін, а також протягом огляду останніх публікацій.

Загальні методи можуть застосовуватися як на теоретичному, так і на емпіричному рівнях дослідження. До них можна віднести моделювання, а саме проєктування структури даної розробки. Це допомагає конкретизувати та продемонструвати взаємозв’язки та компоненти, із яких складається представлена web-орієнтована система. У контексті роботи над проєктом його було також застосовано під час побудови діаграм IDEF0 та UseCase.

2.3Вибір технологій

Для програмної реалізації даної web-орієнтованої системи було обрано JavaScript бібліотеку React [36]. Її використання значно спрощує процес створення інтерактивних інтерфейсів, збільшує швидкодію програмних продуктів, які розробляються, робить їх простішими та масштабованішими, дозволяючи перезастосовувати свої компоненти та будувати інтуїтивно зрозумілу архітектуру.

Скриптова метамова Sass, а саме її синтаксис SCSS, була обрана для спрощення написання стилів і отримання більшої легкості та свободи у написанні CSS [37]. Бібліотеку MUI [38], яка надає широкий спектр інструментів для розробки інтерфейсів, використано для стильового та функціонального оформлення. Вона пропонує вже готові модулі для пришвидшення розробки та типізації. Для роботи з бекендом даної web-орієнтованої системи було обрано Node.js. Це забезпечило реалізацію асинхронних процесів і взаємодії з базою даних (БД) [39]. Для розробки БД було обрано PostgreSQL. Це передова система реляційних баз даних корпоративного класу з відкритим вихідним кодом, яка підтримує декілька видів запитів, а саме SQL та JSON [40].

3 ПРОЄКТУВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ

Після проведення аналізу предметної області, актуалізації бізнес-потреб, формалізації мети, постановки задач і завершення етапу планування робіт було прийняте рішення про перехід до наступної частини, а саме проєктування структури розроблюваної web-орієнтованої системи та моделювання процесів. Було створено діаграми нотації IDEF0. Описано в деталях процеси та їх послідовність, які беруть участь в автоматизованій підтримці діяльності туристичного готелю та допомагають в прийомі нових бронювань, актуалізації інформації, управлінні обслуговуючим персоналом та у вирішенні аналітичних задач.

3.1Діаграми нотації IDEF0

Для моделювання функцій роботи розроблюваної web-орієнтованої системи використовуються різні підходи та методи. Наприклад, IDEF0 [41]. Це методологія графічного опису зв’язків та формалізації процесів як безлічі взаємозалежних функцій. Із її використанням доступною є можливість досліджувати функціонал об’єкта без прив’язки до конкретної реалізації, а опиратися лише на абстрактність. Діаграма містить різну інформацію. Наприклад, про такі дані, як вхідні, управління, вихідні та механізмів.

Контекстна діаграма зображує найвищий рівень декомпозиції. Вона дозволяє побачити максимально абстраговану систему та описати основні дані, які будуть використані. Так як функціонал є доволі обширним, то було прийнято рішення розділити одну на чотири частини, що відповідають за конкретний функціонал. На рисунку 3.1 зображено контекстну діаграму вибору та бронювання кімнати з точки зору гостя

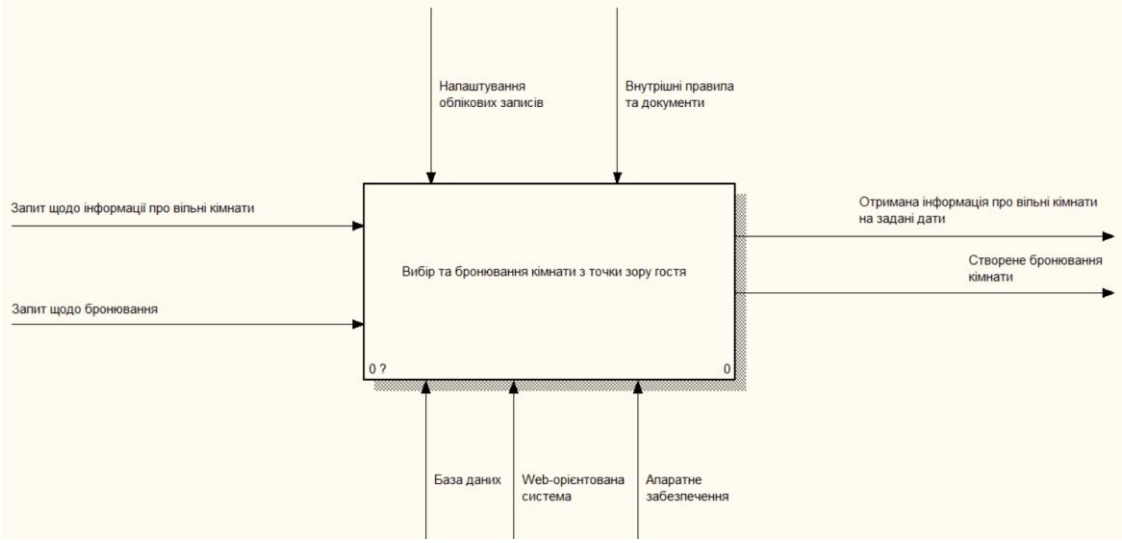


Рисунок 3.1 – Контекстна діаграма вибору та бронювання кімнати з точки зору гостя

Джерело: побудовано автором (знімок з екрану)

Для деталізації процесів в даному компоненті було виконано декомпозицію першого рівня, щоб продемонструвати його роботу детальніше. На рисунку 3.2 зображено декомпозицію процесу вибору та бронювання кімнати з точки зору гостя.

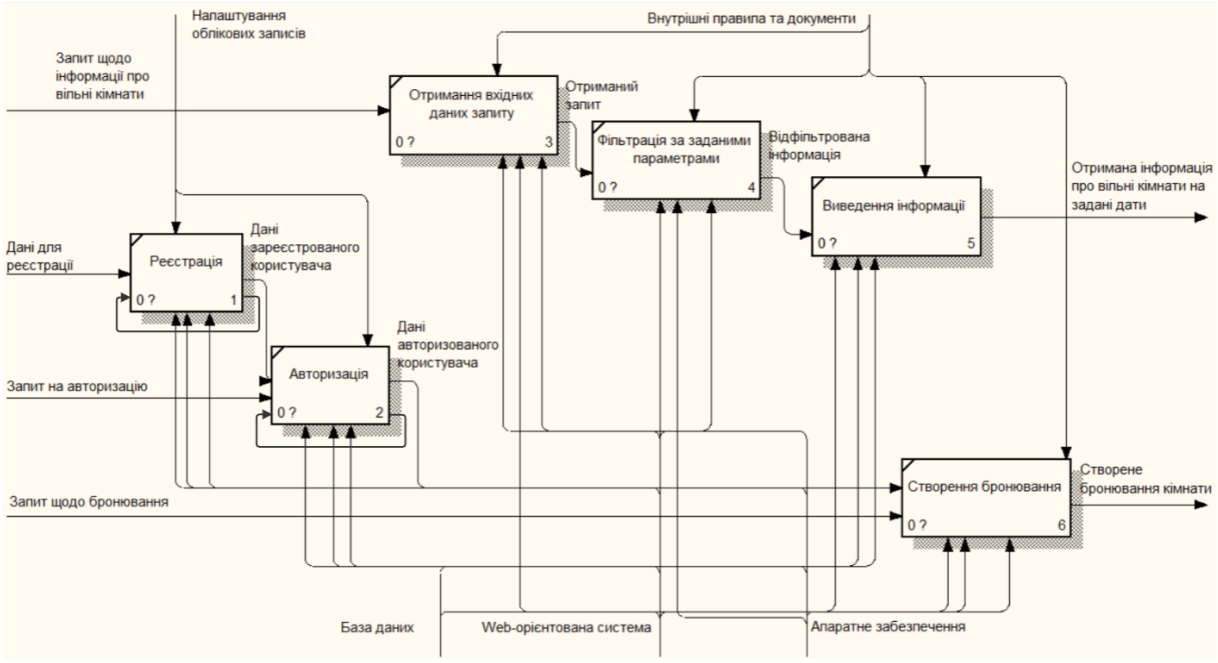


Рисунок 3.2 – Діаграма декомпозиції процесу вибору та бронювання кімнати з точки зору гостя

Джерело: побудовано автором (знімок з екрану)

Інформацію, зображену на діаграмі декомпозиції процесу вибору та бронювання кімнати з точки зору гостя, структуровано у таблиці 3.1.

Таблиця 3.1 – Дані з діаграми декомпозиції процесу вибору та бронювання кімнати з точки зору гостя

Стрілка/ Процес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Реєстрація	Дані для реєстрації	Налаштування облікових записів	Налаштування облікових записів	Дані зареєстрованого користувача
Авторизація	Запит на авторизацію			Дані авторизованого користувача
Отримання вхідних даних запиту	Запит щодо інформації про вільні кімнати	Внутрішні правила та документи	База даних. Web-орієнтована система. Апаратне забезпечення	Отриманий запит
Фільтрація за заданими параметрами	Отриманий запит	Внутрішні правила та документи		Відфільтрована інформація
Виведення інформації	Відфільтрована інформація			Отримана оновлена інформація про вільні кімнати на задані дати
Створення бронювання	Запит щодо бронювання			Створене бронювання кімнати

Джерело: побудовано автором

Контекстну діаграму процесу роботи з бронюваннями з точки зору адміністратора зображено на рисунку 3.3.

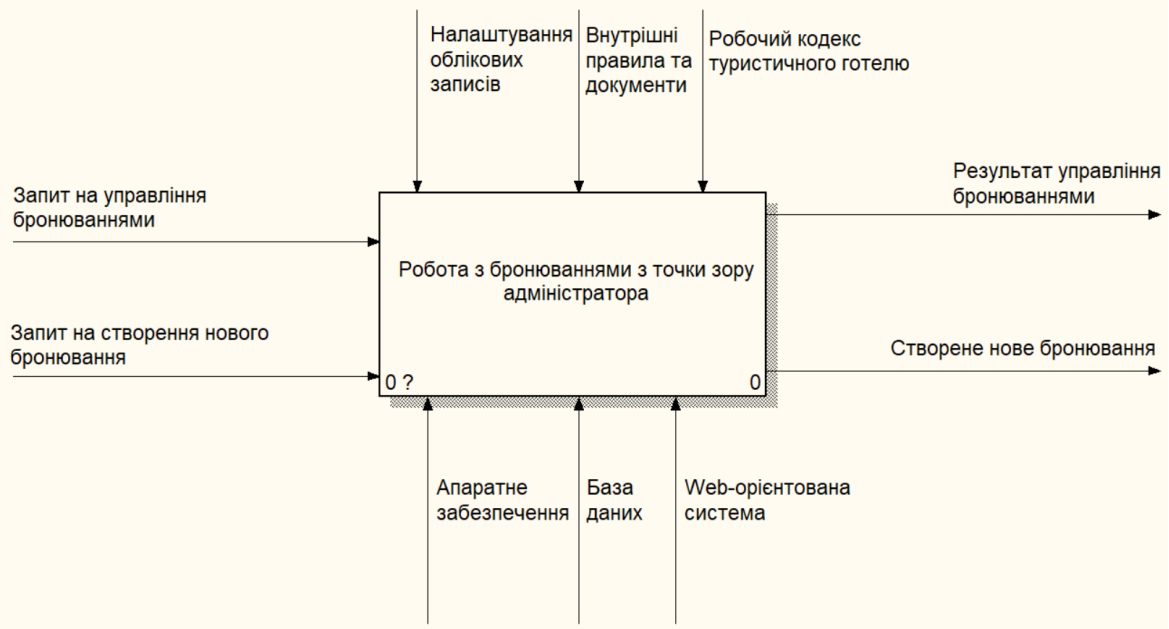


Рисунок 3.3 – Контекстна діаграма процесу роботи
з бронюваннями з точки зору адміністратора
Джерело: побудовано автором (знімок з екрану)

Для деталізації процесів в даному компоненті було виконано декомпозицію першого рівня, щоб продемонструвати його роботу. На рисунку 3.4 зображено декомпозицію процесу роботи з бронюваннями з точки зору адміністратора.

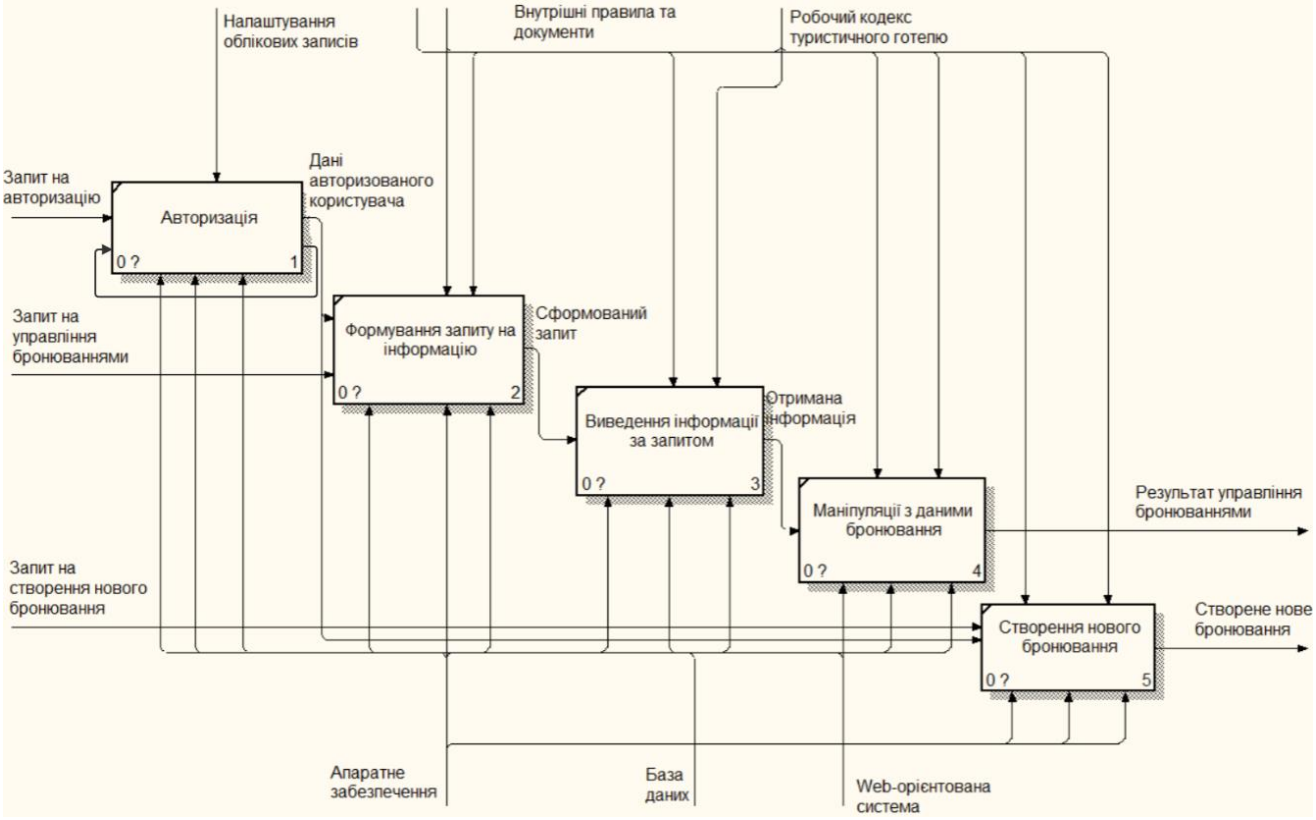


Рисунок 3.4 – Діаграма декомпозиції процесу роботи з бронюваннями з точки зору адміністратора

Джерело: побудовано автором (знімок з екрану)

Для структуризації даних, зображених на діаграмі декомпозиції процесу роботи з бронюваннями з боку адміністратора, було сформовано таблицю 3.2.

Таблиця 3.2 – Дані з декомпозиції процесу роботи з бронюваннями з боку адміністратора

Стрілка/ Процес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Авторизація	Запит на авторизацію	Налаштування облікових записів	База даних. Web-орієнтована система. Апаратне забезпечення	Дані авторизованого користувача

Продовження табл. 3.2

Стрілка/ Процес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Формування запиту на інформацію	Запит щодо інформації про вільні кімнати	Внутрішні правила та документи. Робочий кодекс туристичного готелю	База даних. Web-орієнтована система. Апаратне забезпечення	Сформований запит
Виведення інформації за запитом	Сформований запит			Отримана інформація
Маніпуляції з даними бронювання	Отримана інформація			Результат управління бронюваннями
Створення нового бронювання	Запит на створення нового бронювання			Створене бронювання кімнати

Джерело: побудовано автором

Наступною є контекстна діаграма процесу роботи з завданнями для покоївок з точки зору адміністратора, яку зображено на рисунку 3.5.

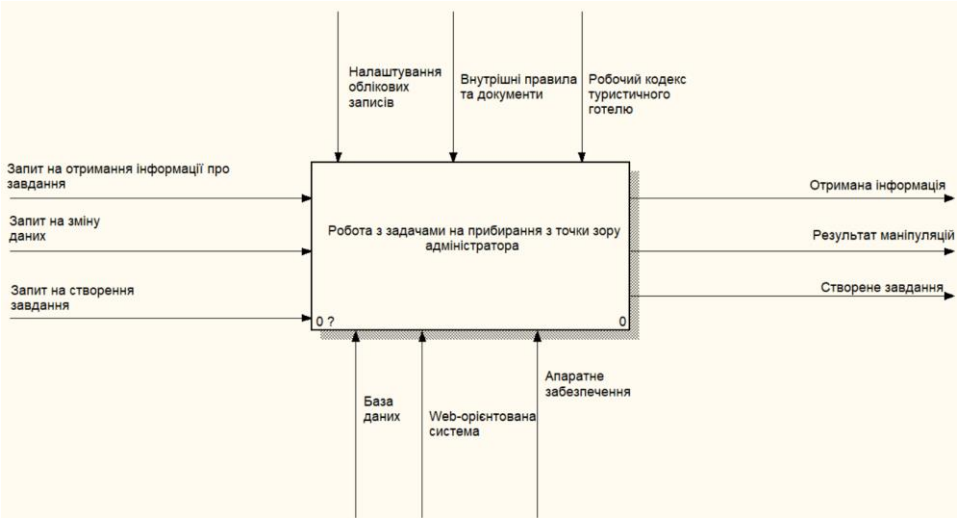


Рисунок 3.5 –Контекстна діаграма процесу роботи з завданнями для покоївок з точки зору адміністратора

Джерело: побудовано автором (знімок з екрану)

На рисунку 3.6 зображено декомпозицію цього процесу.

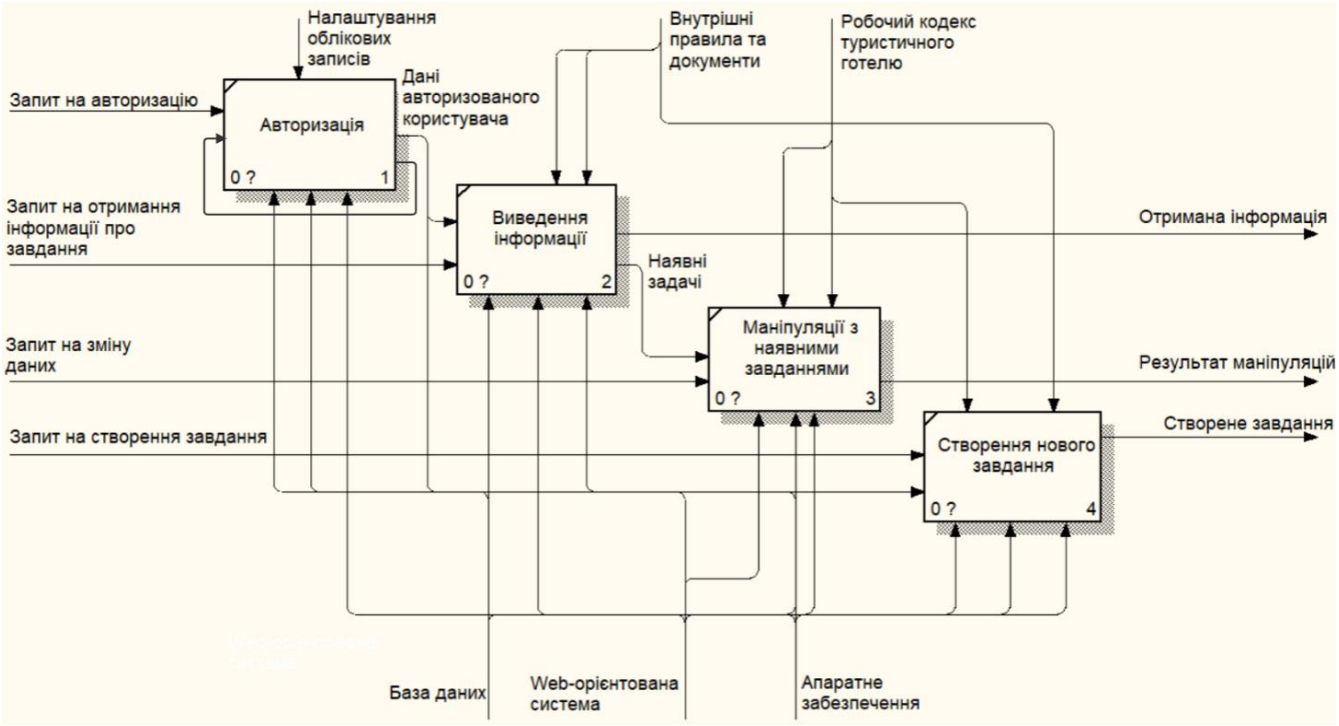


Рисунок 3.6 – Діаграма декомпозиції процесу роботи
з завданнями для покоївок з точки зору адміністратора
Джерело: побудовано автором (знімок з екрану)

У таблиці 3.3 структуровано інформацію з діаграми декомпозиції процесу роботи з завданнями для покоївок з точки зору адміністратора.

Таблиця 3.3 – Дані з декомпозиції процесу роботи з завданнями для покоївок

Стрілка/ Процес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Авторизація	Запит на авторизацію	Налаштування облікових записів	База даних. Web-орієнтована система. Апаратне забезпечення	Дані авторизованог о користувача

Продовження табл. 3.3

Стрілка/ Процес	**Вхідні дані**	**Управління**	**Механізми**	**Вихідні дані**
Виведення інформації	Запит на отримання інформації про завдання	Внутрішні правила та документи. Робочий кодекс туристичного готелю	База даних. Web-орієнтована система. Апаратне забезпечення	Результат управління бронюваннями
				Наявні задачі
Маніпуляції з наявними завданнями	Запит на зміну даних			Результат маніпуляцій
	Наявні задачі			
Створення нового завдання	Запит на створення завдання			Створене завдання

Джерело: побудовано автором

Наступною є контекстна діаграма процесу роботи з задачами на прибирання з точки зору покоївки, яку зображено на рисунку 3.7.

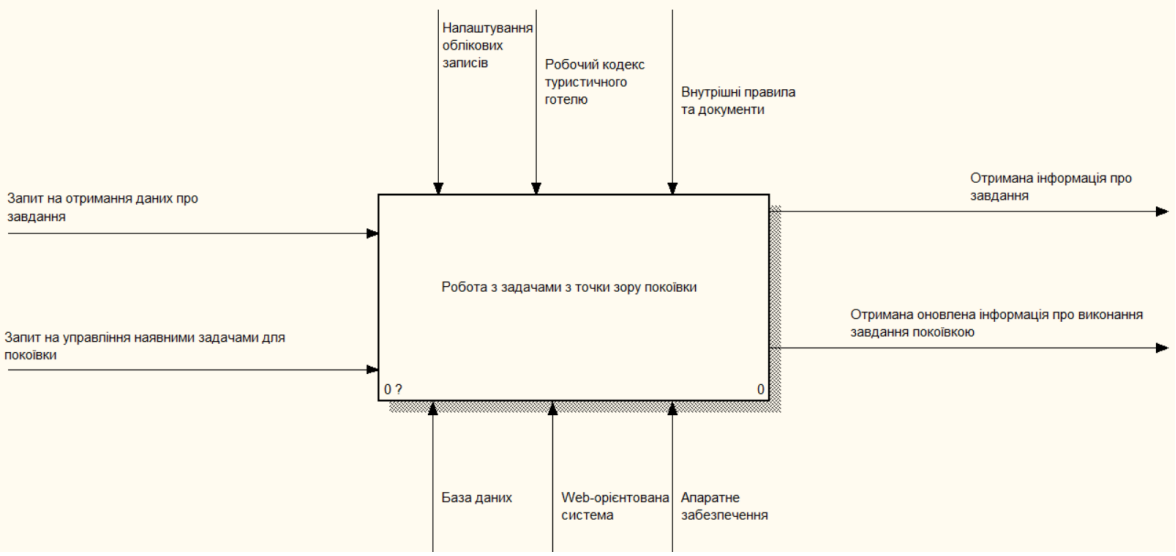


Рисунок 3.7 – Контекстна діаграма процесу роботи з задачами на прибирання з точки зору покоївки

Джерело: побудовано автором (знімок з екрану)

Декомпозицію процесу роботи з задачами на прибирання з точки зору покоївки зображено на рисунку 3.8.

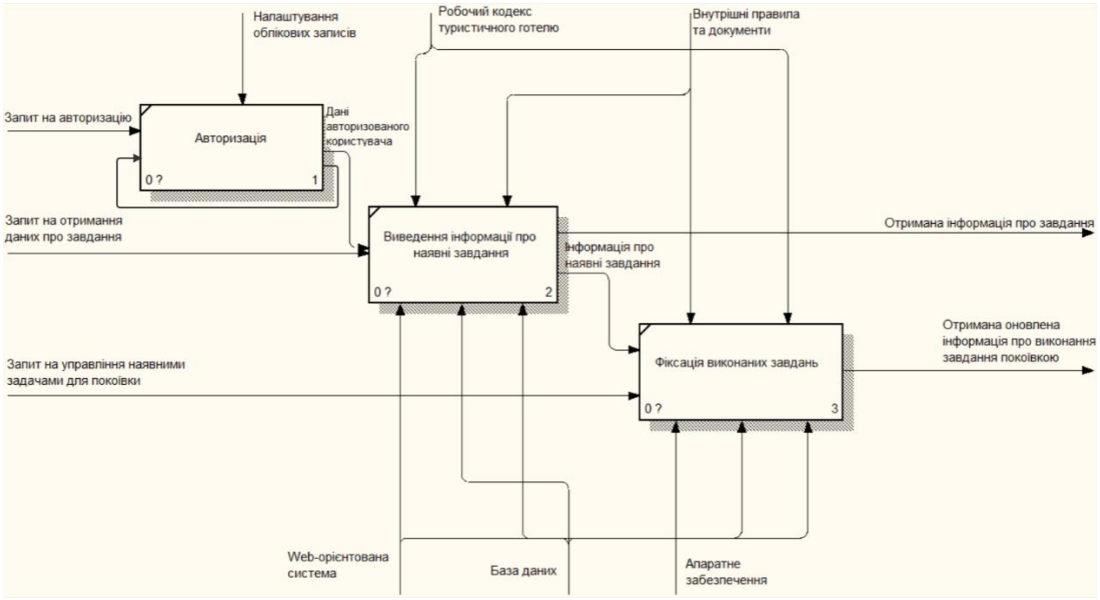


Рисунок 3.8 – Діаграма декомпозиції процесу роботи з задачами на прибирання з точки зору покоївки

Джерело: побудовано автором (знімок з екрану)

Для структуризації даних, зображених на діаграмі декомпозиції процесу роботи з задачами на прибирання з точки зору покоївки, було сформовано таблицю 3.4.

Таблиця 3.4 – Дані з декомпозиції підпроцесу «Робота з задачами з точки зору покоївки»

<table>
<tr><th>Стрілка/ Процес</th><th>Вхідні дані</th><th>Управління</th><th>Механізми</th><th>Вихідні дані</th></tr>
<tr><td>Авторизація</td><td>Запит на авторизацію</td><td>Налаштування облікових записів</td><td>База даних. Web-орієнтована система. Апаратне забезпечення</td><td>Дані авторизованого користувача</td></tr>
<tr><td rowspan="2">Виведення інформації про наявні завдання</td><td rowspan="2">Запит на отримання даних про завдання</td><td rowspan="3">Внутрішні правила та документи. Робочий кодекс туристичного готелю</td><td rowspan="3">База даних. Web-орієнтована система. Апаратне забезпечення</td><td>Отримана інформація про завдання</td></tr>
<tr><td>Інформація про наявні завдання</td></tr>
<tr><td rowspan="2">Фіксація виконаних завдань</td><td>Запит на управління наявними задачами для покоївки</td><td rowspan="2">Інформація про наявні завдання</td></tr>
<tr><td>Інформація про наявні завдання</td></tr>
</table>

Джерело: побудовано автором

3.2 Діаграма Use Case

Після побудови моделі IDEF0 та її декомпозицій потрібно створити діаграму варіантів використання даної web-орієнтованої системи – Use Case [42]. Вона застосовується для опису функціональних можливостей представленого програмного продукту, пов’язаних із його користувачами. На ній зображуються актори. Це ті, хто використовує дану систему. Також продемонтровано варіанти застосування для них.

У web-орієнтованій системі підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел було виділено три основні ролі користувачів: Гість, Покоївка, Адміністратор.

Для користувача «Гість» доступні наступні функціональні можливості:

– бронювання номеру;
– перегляд інформації про кімнати.

Для користувача «Покоївка» доступні такі функції:

– перегляд наявних завдань;
– фіксація виконаних завдань.

Для користувача «Адміністратор» доступні наступні функції:

– перегляд і створення завдань покоївкам;
– управління інформацією та ролями користувачів;
– перегляд та зміна інформації про кімнати;
– управління бронюваннями, заселеннями та виселеннями;
– перегляд статистики.

Актор «База даних» потрібен для того, щоб надати механізм для збереження та надання актуалізованих даних для коректної роботи кожної з вищеперелічених функцій для усіх користувачів системи й забезпечення ефективного використання передбаченого функціоналу.

Діаграма Use Case зображена на рисунку 3.9.

Рисунок 3.9 – Діаграма Use Case

Джерело: побудовано автором (знімок з екрану)

3.3 Проєктування моделі бази даних

Після створення каркасу проєкту необхідно розробити сховище даних, в якому буде зберігатися вся інформація для її коректного функціонування.

База даних є важливою для будь-якої web-орієнованої системи. Адже структурована та зібрана в одному місці інформація спрощує всі процеси, які пов’язані з обробкою даних та різними маніпуляціями з ними.

Під час проєктування БД формується кількість таблиць, їх зміст та будуються взаємозв’язки між ними. Кожна з них має власну структуру, тобто складається з стовпців, які визначають розмір, тип та особливості даних, які зберігаються.

У результаті проєктування та розробки бази даних було сформовано наступні сутності:

– users (Користувачі) – містить інформацію про користувачів системи для ідентифікації;

– staff (Персонал) – містить особисту інформацію про персонал готелю;

– guests (Гості) – містить інформацію про відвідувачів;

– roles (Ролі) – зберігає ролі користувачів для розмежування доступу;

– rooms (Кімнати) – таблиця для зберігання основної інформації про кімнати;

– categories (Категорії) – зберігає категорії номерів;

– room_categories (Категорії кімнат) – допоміжна таблиця для збереження зв’язку кімнат із категоріями;

– amenities (Зручності) – містить інформацію про зручності, які є в готелі;

– room_amenities (Зручності кімнати) – допоміжна таблиця для забезпечення зв’язку таблиць із кімнатами та зручностями;

– bed_info (Інформація про ліжка) – зберігає інформацію про ліжка в кімнатах;

– room_photos (Фотографії кімнат) – таблиця для збереження посилань на зображення, пов’язаниз з кімнатою;

– bookings (Бронювання) – місце для збереження інформації про всі бронювання та їх статуси;

– check_ins (Реєстрація заїзду) – таблиця для фіксування інформації про фактичні заселення та виселення;

– cleaning_tasks (Завдання з прибирання) – зберігає завдання, які видаються покоївкам;

– notification (Сповіщення) – містить інформацію про сповіщення, яке надсилається адміністратору та покоївкам і його пріоритет;

– notification_list (Список сповіщень) – допоміжна таблиця, яка пов’язує сповіщення та їхніх адресатів.

4 РОЗРОБКА WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ПІДТРИМКИ ДІЯЛЬНОСТІ ТУРИСТИЧНОГО ГОТЕЛЮ ІЗ МЕХАНІЗМОМ СИНХРОНІЗАЦІЇ ДАНИХ ПО БРОНЮВАННЮ НОМЕРІВ ІЗ ДЕКІЛЬКОХ ДЖЕРЕЛ

4.1 Архітектура web-орієнтованої системи

Перед тим, як починати програмно реалізовувати web-орієнтовану систему, потрібно розробити її архітектуру. Це є описом її будови та допомагає зрозуміти на які модулі вона поділяється та які складові містить. Також, встановлюється формат та рівень взаємодії між ними. Загалом, архітектура web-орієнтованої системи складається з двох основних компонентів: клієнт (React) та сервер (Node.js). Клієнтська частина складається з роутера, який керує відображенням контенту для конкретного URL, компонента – безпосередньо того, що відображається користувачу як розмітка та інтерфейс, та сервісів – засобів, які забезпечують виконання CRUD-операцій (читання, оновлення, створення та видалення) через HTTP клієнт Axios [43]. Сервер, у свою чергу, поділяється також на роутери, які приймають запити від клієнта через фреймворк Express [44], та після його обробки надсилають запит у сервіс. Останні формують запит до бази даних та повертають результат до клієнта. Схема архітектури запропонованої web-орієнтованої системи зображена на рисунку 4.1.

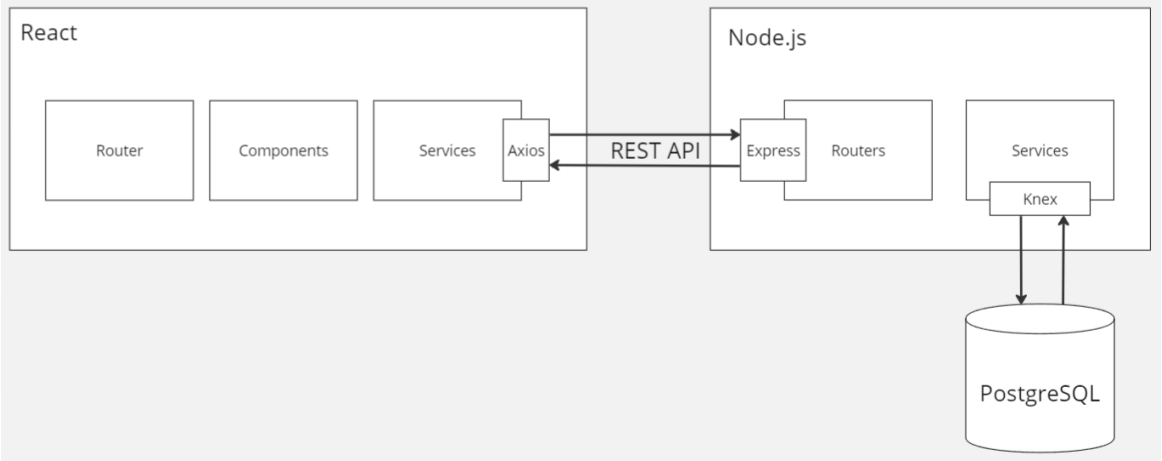


Рисунок 4.1 – Архітектура web-додатку

Джерело: побудовано автором (знімок з екрану)

4.2 Розробка бази даних

Далі було реалізовано базу даних, описану у розділі 3.3. На рисунку 4.2 зображено логічну модель БД представленої web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел

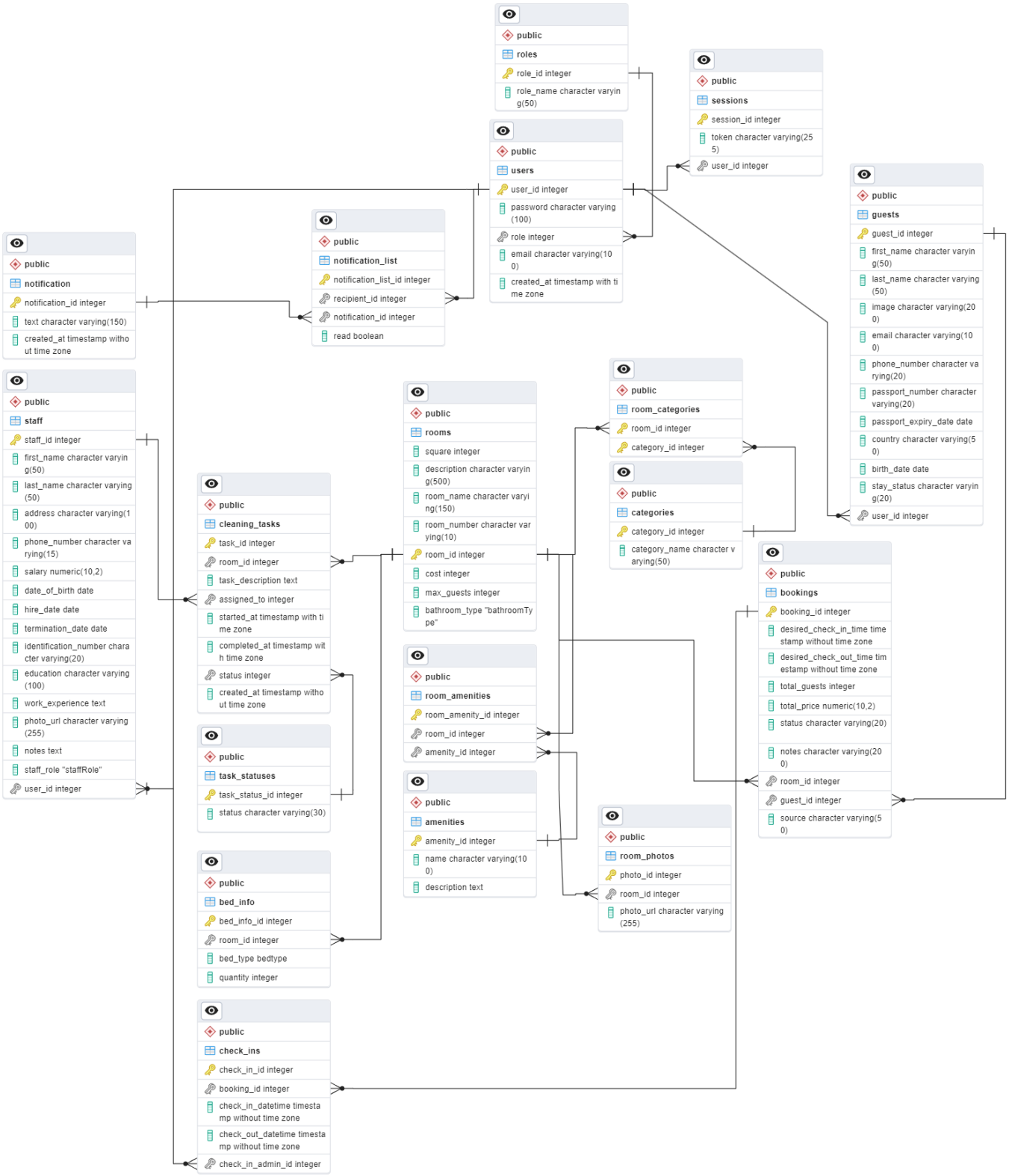


Рисунок 4.2 – Логічна модель бази даних

Джерело: побудовано автором (знімок з екрану)

4.3 Програмна реалізація web-орієнтованої системи

Перед тим, як розпочати безпосередньо реалізацію, потрібно налаштувати робоче середовище. Спочатку необхідно створити робочу папку на персональному комп’ютері для даного програмного продукту. Далі, через консоль ініціюємо розробку нового проєкту за допомогою команди «npx create-react-app «hotel-master»». Далі, клонуємо даний проєкт до віддаленого репозиторію на GitHub [45] за допомогою команди «git init» для легшого контролю версій. Наступним кроком треба встановити всі допоміжні бібліотеки та залежності. Для написання програмного коду було обрано IDE WebStorm [46] та встановлено розширення і бібліотеки eslint [47] для типізації написаного коду. Для пришвидшення дебагінгу та допомоги в написанні невеликих компонентів без логіки використовувався ChatGPT3.5 [48].

Одним із головних компонентів фронтенду є роутингова навігація. За її допомогою відбувається вся навігація по інтерфейсу та забезпечується рендеринг потрібних компонентів за адресою, на яку перейшов користувач. На рисунку 4.3 зображено приклад використання роутів.

```
<Routes>
  <Route>
    <Route path="/" element={<MainPage />} />
    <Route
      path="/rooms-availability-check-result"
      element={<SearchResultPage />}
    />
    <Route path="/rooms" element={<RoomsPage />} />
  </Route>
  <Route path="/login" element={<LoginPage />} />
  <Route path="/registry" element={<RegisterPage />} />
  <Route
    path="/balance"
    exact
    element={<BalanceRoute user={context.user} />}
  />
  <Route element={<BookRoute token={context.token} />}>
    <Route
      path="/complete-booking/:roomID"
      element={<BookPage />}
    />
  </Route>
```

Рисунок 4.3 – Приклад відображення основних маршрутів та компонентів, які за ними відображаються

Джерело: побудовано автором (знімок з екрану)

Кожен елемент для відображення, який представлений в маршрутизаторі представляє шаблон. Він складається з дрібніших частин, які можна повторно використовувати. Це є однією з основних переваг застосування React. Приклад використання такого шаблону зображено на рисунку 4.4.

```
const MainPage = () => {
  const { isFetching, data } = useQuery( queryKey: "favRooms",  queryFn: () => getFavRooms());
  const rooms = data?.data;
  return (
    <Container className="main-page-container">
      <SliderContainer />
      <AvailabilityCheckContainer />
      <FavoriteRoomsContainer rooms={rooms} isFetching={isFetching} />
    </Container>
  );
};
```

Рисунок 4.4 – Побудова шаблону головної сторінки з підкомпонентами

Джерело: побудовано автором (знімок з екрану)

Для отримання актуальної інформації на клієнті потрібно налаштувати зв’язок із сервером. Він відбувається завдяки API [49]. Саме через нього клієнтська частина звертається до сервера, а той відповідно опрацьовує його, зв’язується з базою та повертає сформовані дані в інтерфейс для відображення. Для можливості отримувати запити, потрібно було налаштувати роутери всередині express сервера, які приймають запити типу GET, PUT, DELETE, ADD (рис. 4.5).

```
// Get All Bookings
router.get(
   path: "/",
  asyncHandler( fn: async (req, res) => {...})
);

// Get Booking by ID
router.get(
   path: "/:bookingId",
  asyncHandler( fn: async (req, res) => {...})
);

// Add Booking
router.post(
   path: "/",
  asyncHandler( fn: async (req, res) => {...})
);

// Update Booking
router.put(
   path: "/:bookingId",
```

Рисунок 4.5 – Приклад використання роутера для роботи з бронюваннями

Джерело: побудовано автором (знімок з екрану)

Для отримання актуальної інформації з бази даних роутери звертаються до спеціальних сервісів. Вони представляють собою запити до БД, написані за допомого бібліотеки knex [50] (рис. 4.6).

```
deleteBooking: async (bookingId) =>
  db("bookings")
    .where( columnName: "booking_id", bookingId)
    .delete()
    .returning( column: "booking_id"),
getBookingsByRoomId: async (roomId) =>
  db.select( columnName: "*").from( tableName: "bookings").where( columnName: "bookings.room_id", roomId),
getBookingsByUserId: async (roomId) =>
  db.select( columnName: "*").from( tableName: "bookings").where( columnName: "bookings.room_id", roomId),
```

Рисунок 4.6 – Програмний код використання запитів у функціях

Джерело: побудовано автором (знімок з екрану)

Хоча частина функціоналу модуля бронювання гостя, що відповідає за відображення інформації та пошук вільних кімнат не потребує авторизації, але доступ до функціональних модулів створення бронювання, адміністрування та для забезпечення роботи покоївок розмежовано між відповідними ролями в системі. Саме тому було створено 3 основні користувачі: Гість (Guest), Адміністратор (Admin), Покоївка (Housemaid). Гість може реєструватися та авторизуватися в даній системі для забезпечення ідентифікації. Інші можуть бути створеними лише через адміністративну панель, але також мають авторизовуватися. Цей механізм було реалізовано за допомогою бібліотеки Passport JS [51]. Це є проміжним програмним забезпеченням для виконання задач аутентифікації та JWT токенами [52], які зберігаються в сесії та мають в собі основну інформацію про авторизованого користувача. Реалізація даної стратегії зображена на рисунку 4.7.

```
passport.use(
  new LocalStrategy(
    options: {
      usernameField: "signinEmail",
      passwordField: "signinPassword",
    },
    verify: async (email, password, done) => {
      const user = await userService.getByEmail(email);
      if (user) {
        if (user.password) {
          bcrypt.compare(password, user.password, (err : Error , res : boolean ) => {
            if (!err) {
              if (res) {
                return done(null, user);
              }
              return done(null, false, { message: "Incorrect password" });
            }
            console.log(err);
          });
        } else {
          return done(null, false, {
            message: "Please enter the password",
          });
        }
      } else {
        return done(null, false, { message: "Incorrect login" });
      }
    }
  )
```

Рисунок 4.7 – Програмний код стратегії для аутентифікації

Джерело: побудовано автором (знімок з екрану)

Для розмежування доступу на клієнті використовувалися захищені роути, які витягують з сесії токен, якщо він існує, парсять його та звіряють із заданою роллю для перевірки чи підходить рівень доступу. Якщо так, то відбувається подальша переадресація на запитувану адресу. У противному випадку авторизований користувач переспрямовується або на сторінку авторизації або, у випадку невідповідного рівня доступу, просто на головну сторінку туристичного готелю (рис. 4.8).

```
const AdminProtectedRoute = ({ user, redirectPath : string  = "/" }) => {
  if (!user || Object.keys(user).length === 0 || user.role !== 1) {
    const localToken = localStorage.getItem( key: "token");
    if (!localToken) {
      return <Navigate to="/registry" replace />;
    }

    const logUser = jwtDecode(JSON.parse(localToken).accessToken);

    if (logUser && logUser.role !== 1) {
      return <Navigate to={redirectPath} replace />;
    }
  }

  return <Outlet />;
};
```

Рисунок 4.8 – Приклад обробки користувача в захищеному роутері

Джерело: побудовано автором (знімок з екрану)

Також було враховано, що зв’язок із сторонніми сервісами може відбуватися через API інтерфейс. Тому потрібно подбати про валідації на стороні сервера. Наприклад, перевірка заданих дат при створенні нового бронювання, щоб не створювалося колізій та дублів у системі. А це є критично важливим для туристичного готелю. Для цього було створено окреме проміжне програмне забезпечення, яке відповідає за різноманітні валідації (рис. 4.9). На рисунку 4.10 показано як це middleware використовується під час отримання API запита.

```
for await (const ruleValidate of rules) {
  const [rule, param] = ruleValidate.split(":");
  // eslint-disable-next-line default-case
  switch (rule) {
    case "required":
      if (!req.body[field]) {
        fieldErrors.push("Field is required");
      }
      break;
    case "email":
      if (
        !/^\w+([.-]?\w+)*@\w+([.-]?\w+)*(\.\w{2,3})+$/.test(req.body[field])
      ) {
        fieldErrors.push("Valid email is required");
      }
      break;
    case "min":
      {
        const minValue = parseInt(param,  radix: 10);
        if (req.body[field] && req.body[field].length < minValue) {
          fieldErrors.push(`Too short. min = ${minValue}`);
        }
      }
      break;
```

Рисунок 4.9 – Приклад валідаційних правил

Джерело: побудовано автором (знімок з екрану)

Рисунок 4.10 – Приклад використання валідаційного middleware

Джерело: побудовано автором (знімок з екрану)

Також для адміністраторів та покоївок є важливим отримувати інформацію як тільки вона стає доступною в системі. Наприклад, нові бронювання або нові задачі на прибирання. Тому було розроблено систему сповіщень. Вони відразу повідомляють про подібну подію. Сповіщення виводяться на екран потрібному користувачу. Це було створено за допомогою платформи реального часу Ably [53]. Вона надає інструменти для забезпечення миттєвого обміну даними. Ably прослуховує спеціально визначені канали та реагує на зміну в них, отримуючи поточну інформацію. Приклад її використання зображено на рисунках 4.11-4.12.

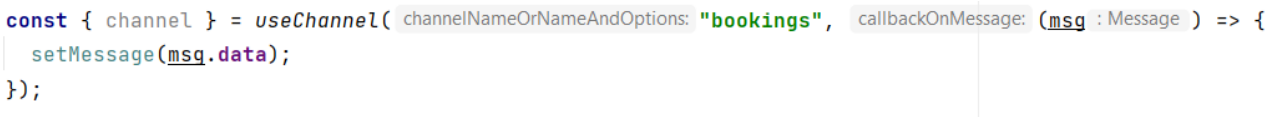


Рисунок 4.11 – Приклад прослуховування каналу та отримання інформації на фронтенді

Джерело: побудовано автором (знімок з екрану)

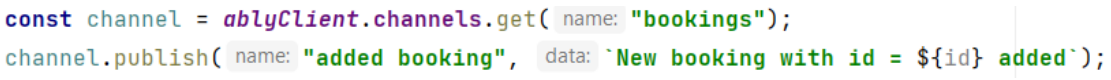


Рисунок 4.12 – Приклад прослуховування каналу та відправки інформації на бекенді

Джерело: побудовано автором (знімок з екрану)

Також існує можливість додавати фото до кімнати. Це було організовано за допомогою бібліотеки multer [54] та Google Cloud Storage [55]. Нові фото отримуються через запит, відбувається їх обробка й вони надсилаються до сервісу у певну задану папку, яка відповідає окремій кімнаті. Частину логіки зображено на рисунку 4.13

```
const bucket = storage.bucket(config.cloud.bucket);

const cloudStorage = multerGoogleStorage.storageEngine( opts: {
  autoRetry: true,
  projectId: config.cloud.projectId,
  keyFilename: "./src/services/valiant-adviser-403118-6a5d33645bf7.json",
  bucket: config.cloud.bucket,
  filename: (req, file, cb) => {
    const roomNum = req.params.roomNum || null;
    let folder = "";
    if (roomNum) {
      folder = `rooms/${roomNum}`;
    } else {
      folder = `rooms/temp`;
    }
    const ext = file.mimetype.split("/")[1];
    const filename = `${uuidv4()}.${ext}`;
    const filePath = `${folder}/${filename}`;
    cb(null, filePath);
  },
});
```

Рисунок 4.13 – Частина функції для роботи з файлами

Джерело: побудовано автором (знімок з екрану)

Лістинг описаних у цьому розділі, а також інших основних модулів web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел, описано у додатку В. Також код всіх файлів знаходиться в репозиторії GitHub [56].

4.4Демонстрація роботи web-орієнтованої системи

Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел підтримує наступні 4 типи користувачів, які відрізняються рівнями доступу до даних та обмеженнями в використанні функціоналу:

– Admin – це роль із найбільшими правами в системі; він може додавати, редагувати та видаляти інформацію пов’язану з туристичним готелем;

– Housemaid – це для користувачів покоївок, які можуть переглядати доступні задачі на прибирання, а також змінювати їхній статус;

– Guest – це користувач авторизованого гостя, який може створювати бронювання та переглядати інформацію про готель та кімнати;

– Неавторизований користувач – стандартний початковий користувач, який може лише переглядати інформацію про готель та номери.

При першому відвідуванні, якщо система не знаходить сесії, пов’язаної з користувачем, то вона перенаправляє на головну сторінку туристичного готелю. Там розміщено загальні фото даного закладу, форма з пошуком номеру на задані дати та список найпопулярніших номерів (рис. 4.14-4.15).

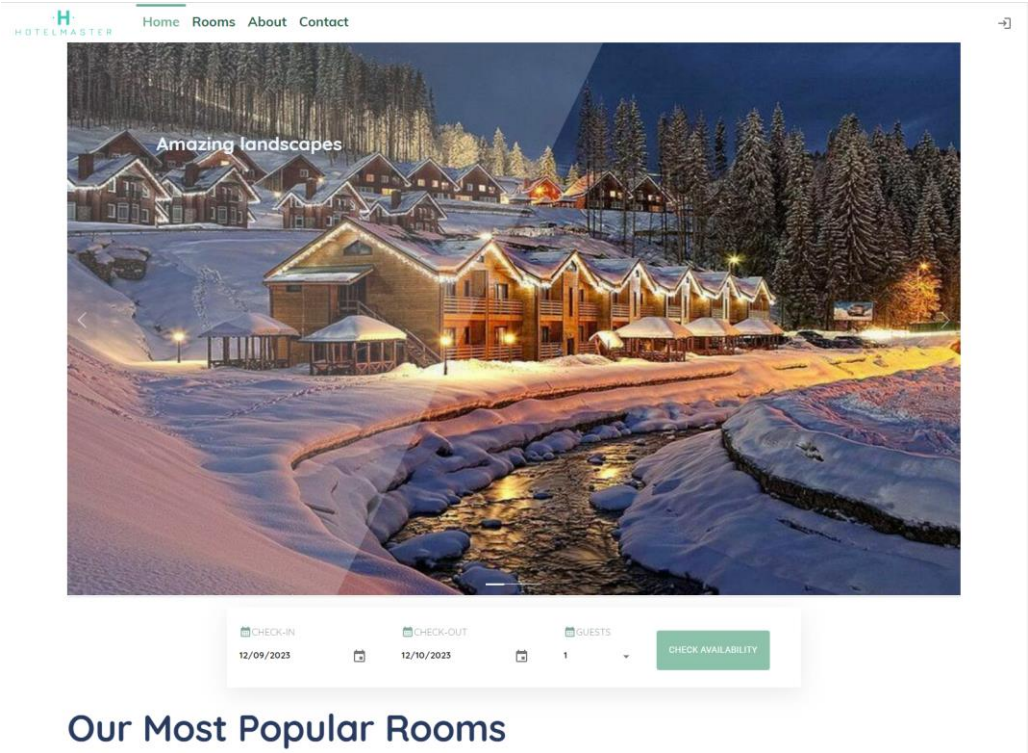


Рисунок 4.14 – Вигляд головної сторінки туристичного готелю (верхня частина)

Джерело: побудовано автором (знімок з екрану)

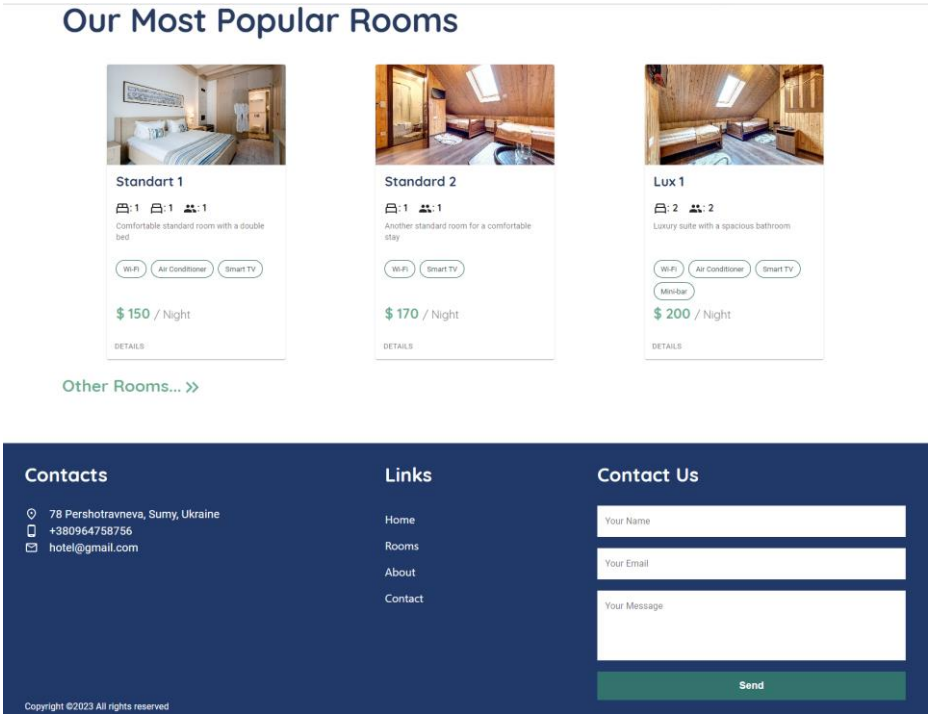


Рисунок 4.15 – Вигляд головної сторінки готелю (нижня частина)

Джерело: побудовано автором (знімок з екрану)

На головній сторінці можна ввести дати, які цікавлять користувача, та виконати пошук вільних номерів. Їх список відобразиться на новій сторінці (рис. 4.16).

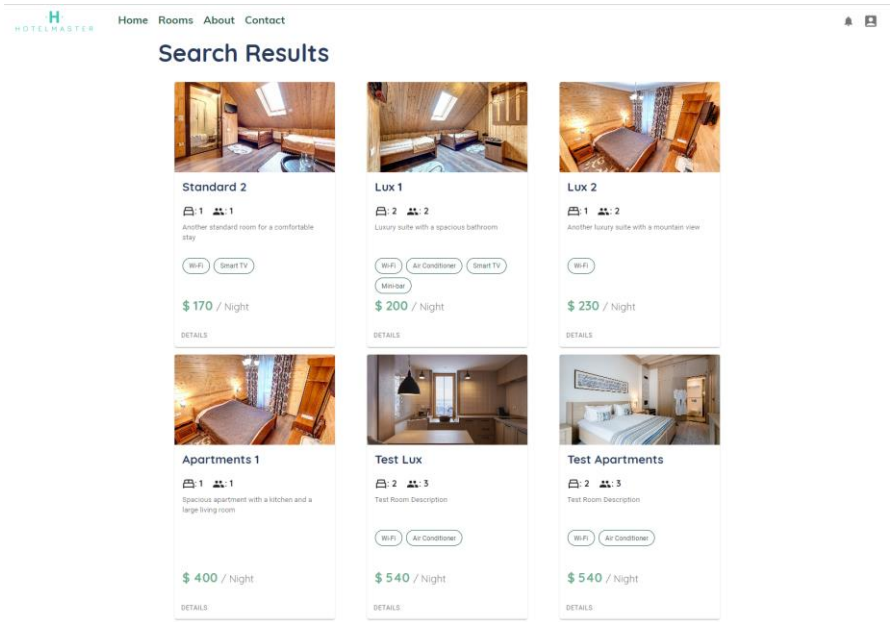


Рисунок 4.16 – Вигляд сторінки з результатами пошуку

Джерело: побудовано автором (знімок з екрану)

Перейшовши на вкладку «Rooms», можна побачити повний список номерів готелю (рис. 4.17), а також фільтри. Вони мають певні категорії, як дати та кількість гостей. Результат застосування фільтру продемонстровано на рисунку 4.18.

Натиснувши на кнопку «Details» на картці кімнати, можна відкрити всю інформацію про неї, а також календар доступності у модальному вікні (рис. 4.19-4.20).

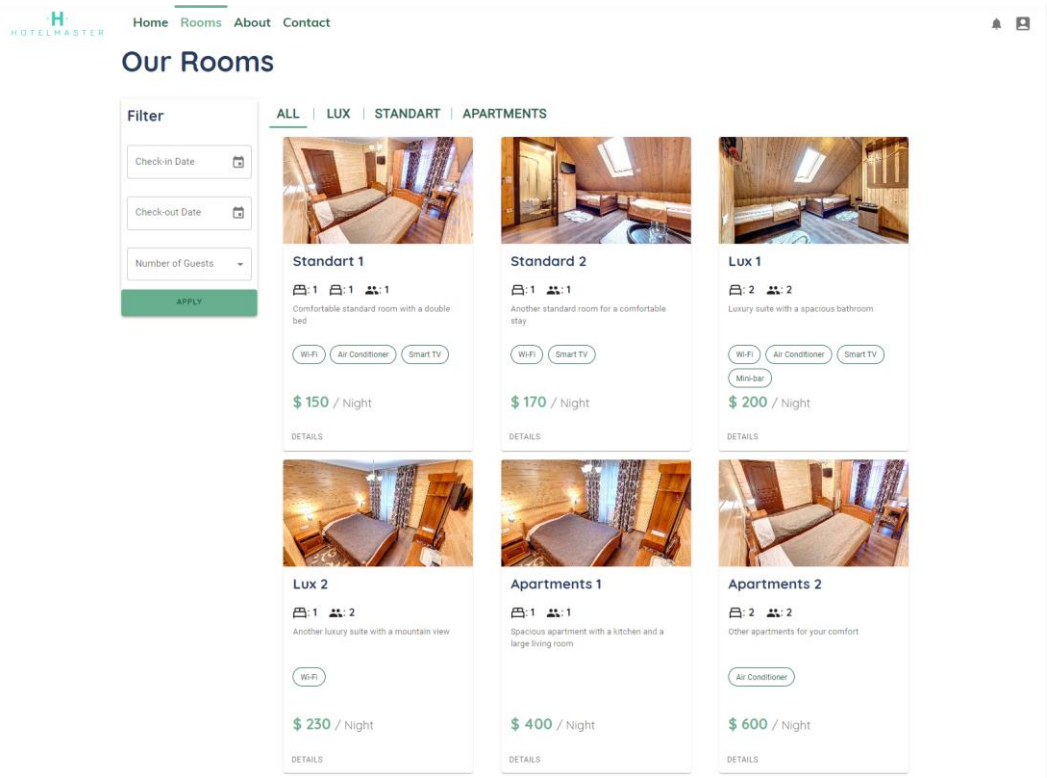


Рисунок 4.17 – Вигляд сторінки з усіма кімнатами

Джерело: побудовано автором (знімок з екрану)

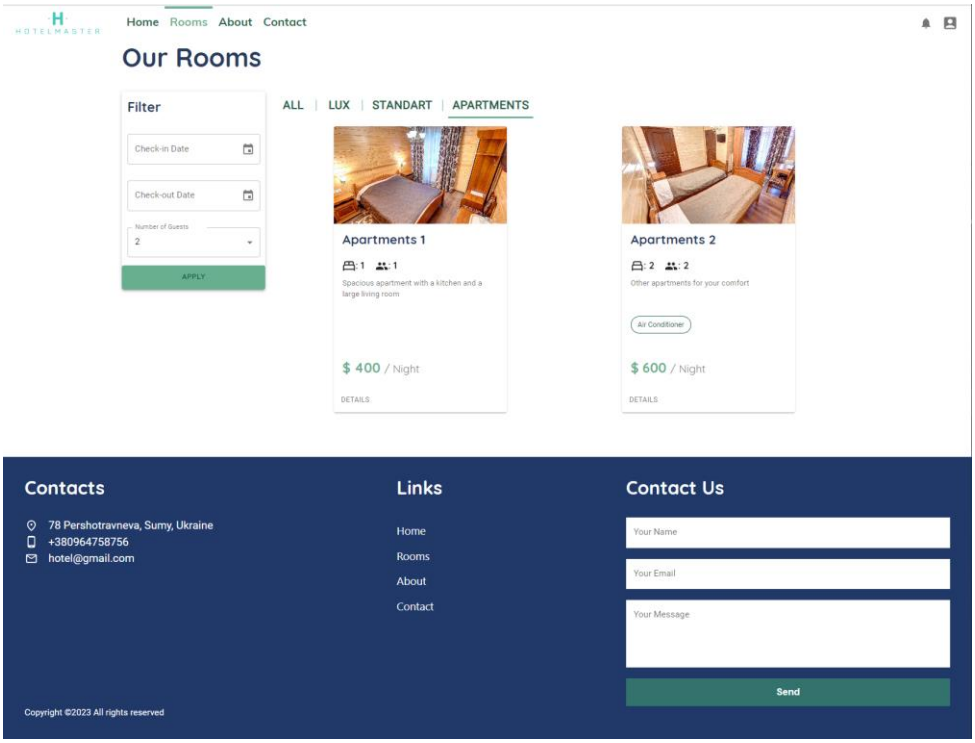


Рисунок 4.18 – Вигляд сторінки з усіма кімнатами із застосованим фільтром

Джерело: побудовано автором (знімок з екрану)

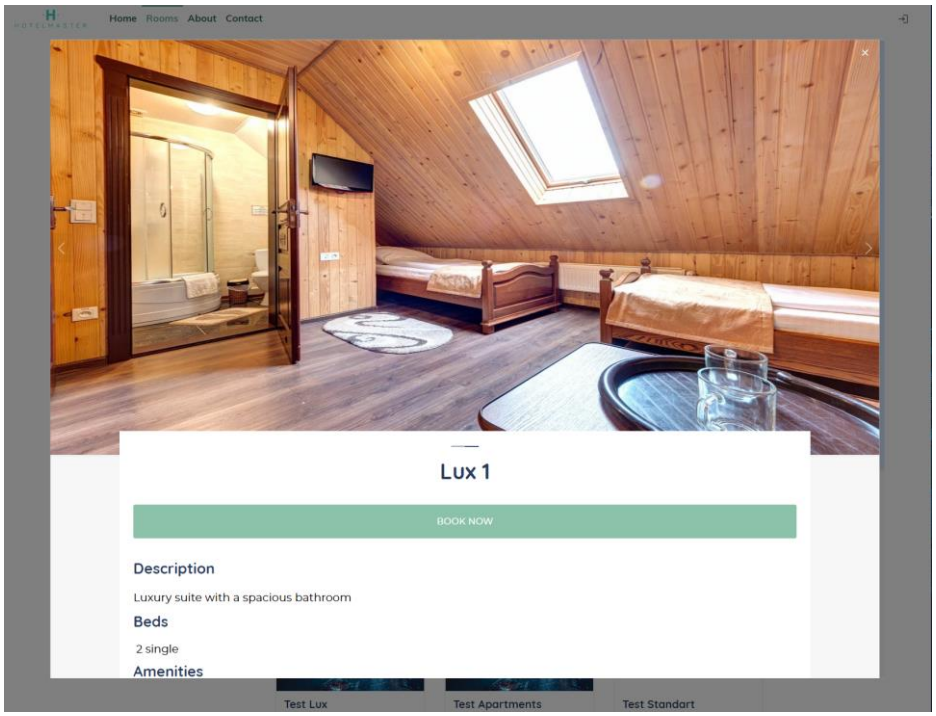


Рисунок 4.19 – Вигляд вікна з інформацією про номер (верх)

Джерело: побудовано автором (знімок з екрану)

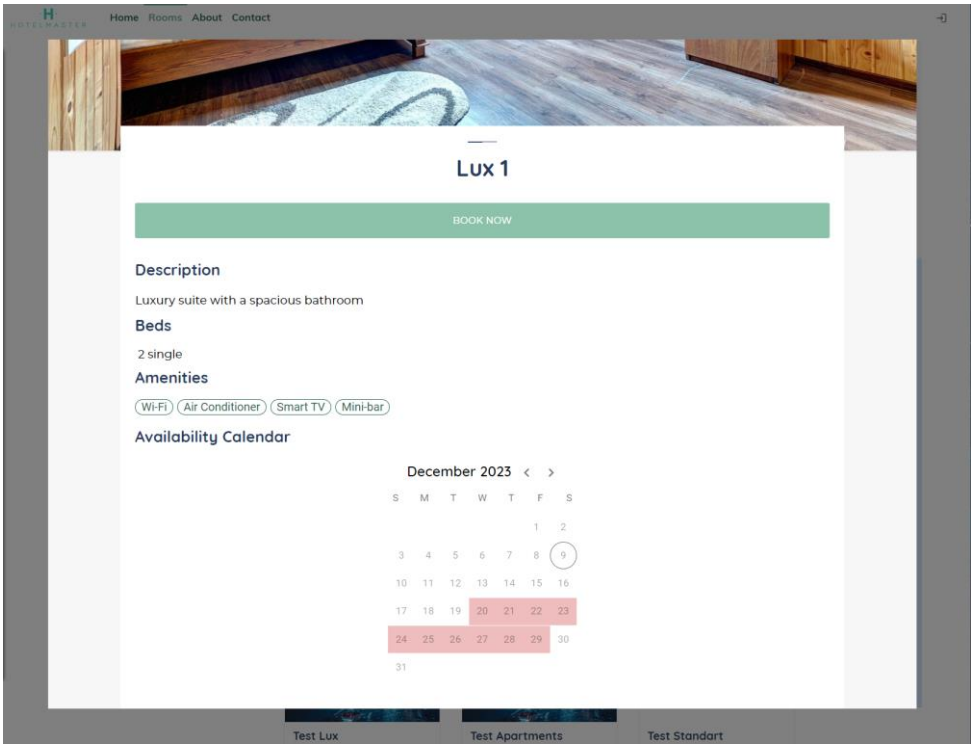


Рисунок 4.20 – Вигляд вікна з інформацією
про номер (частина з календарем)

Джерело: побудовано автором (знімок з екрану)

Для переходу до бронювання кімнати можна натиснути кнопку «Book Now». Однак, якщо користувач ще не авторизований та не має токена, то йому виведеться повідомлення, що потрібно зареєструватися або авторизуватися (рис.4.21).

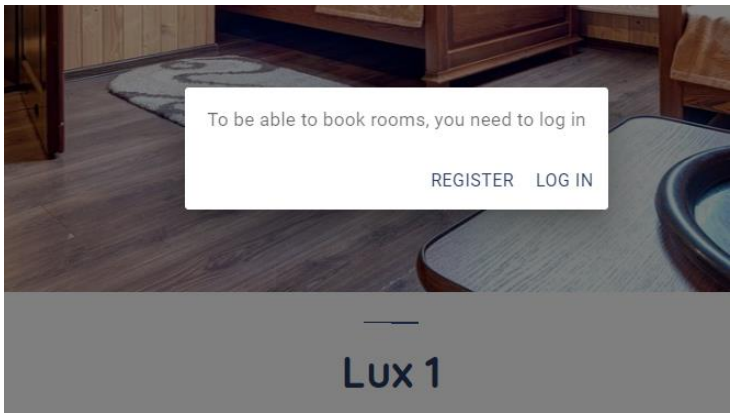


Рисунок 4.21 – Вигляд вікна з інформацією про необхідність авторизації

Джерело: побудовано автором (знімок з екрану)

Для авторизації та реєстрації було створено дві форми, які зображено на рисунках 4.22 та 4.23 відповідно.

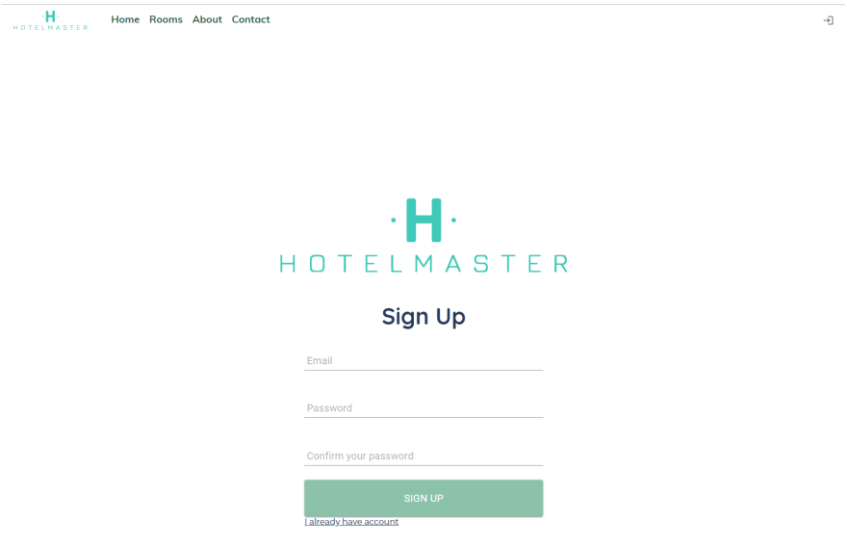


Рисунок 4.22 – Вигляд форми для реєстрації

Джерело: побудовано автором (знімок з екрану)

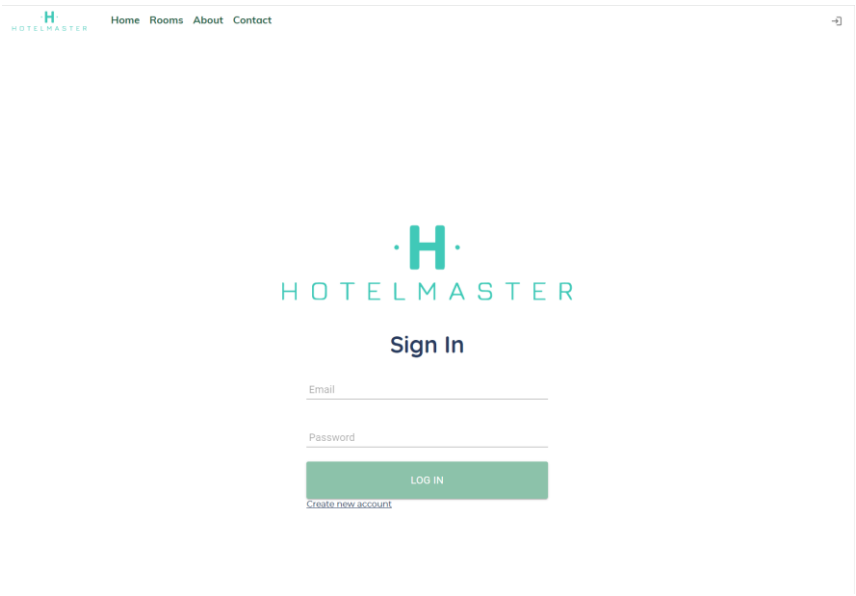


Рисунок 4.23 – Вигляд форми для авторизації

Джерело: побудовано автором (знімок з екрану)

Після успішної авторизації користувач може повторити попередню процедуру та натиснути кнопку «Book Now». Після цього він перенаправляється на сторінку створення замовлення (рис. 4.24).

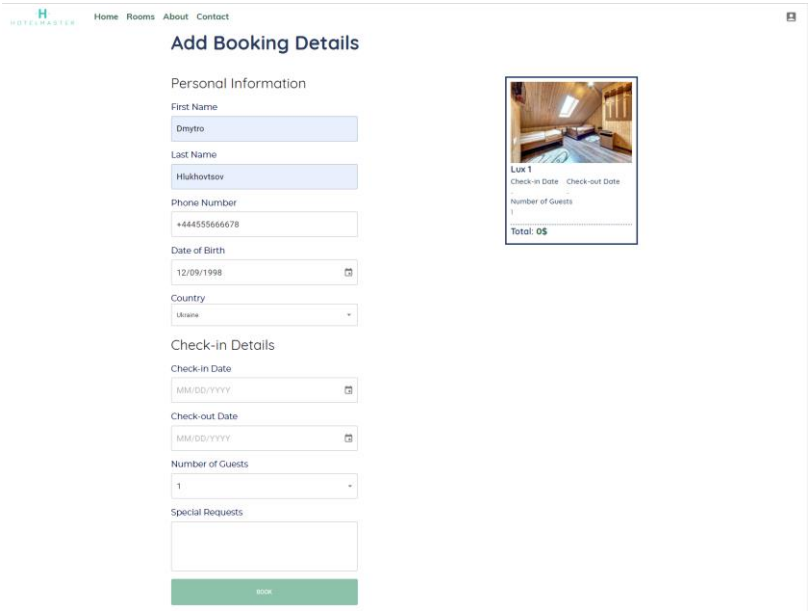


Рисунок 4.24 – Вигляд сторінки для оформлення бронювання

Джерело: побудовано автором (знімок з екрану)

Також для сторінки з створенням бронювання було розроблено компонент календаря. Він успадкований від MUI-X календаря. Однак із власним рендерингом компонентів для відображення вже зайнятих дат та візуалізації обраних (рис. 4.25).

Після заповнення усіх полів у правій частині вікна відобразиться основна інформація про замовлення (рис. 4.26).

У разі успішного додавання бронювання буде відображено повідомлення у верхній частині екрану про успіх (рис. 4.27) та відбудеться редирект до головної сторінки туристичного готелю.

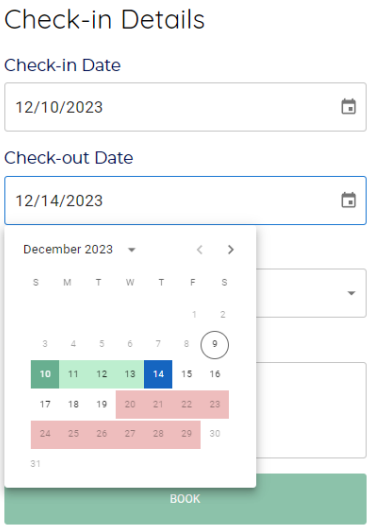


Рисунок 4.25 – Вигляд кастомного поля для введення дати

Джерело: побудовано автором (знімок з екрану)

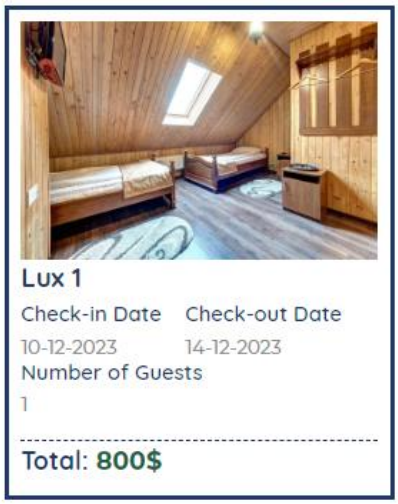


Рисунок 4.26 – Вигляд короткої інформації про бронювання

Джерело: побудовано автором (знімок з екрану)

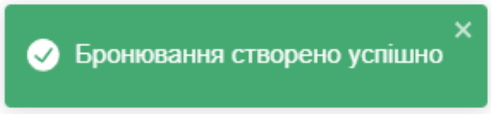


Рисунок 4.27 – Вигляд сповіщення про успішність бронювання

Джерело: побудовано автором (знімок з екрану)

Далі розглянемо функціонал для адміністратора. Спочатку авторизуємось за допомогою форми та відразу потрапляємо до сторінки з таблицею бронювань (рис. 4.28). Також можна помітити, що шапка web-орієнтованої системи для адміністратора змінилась

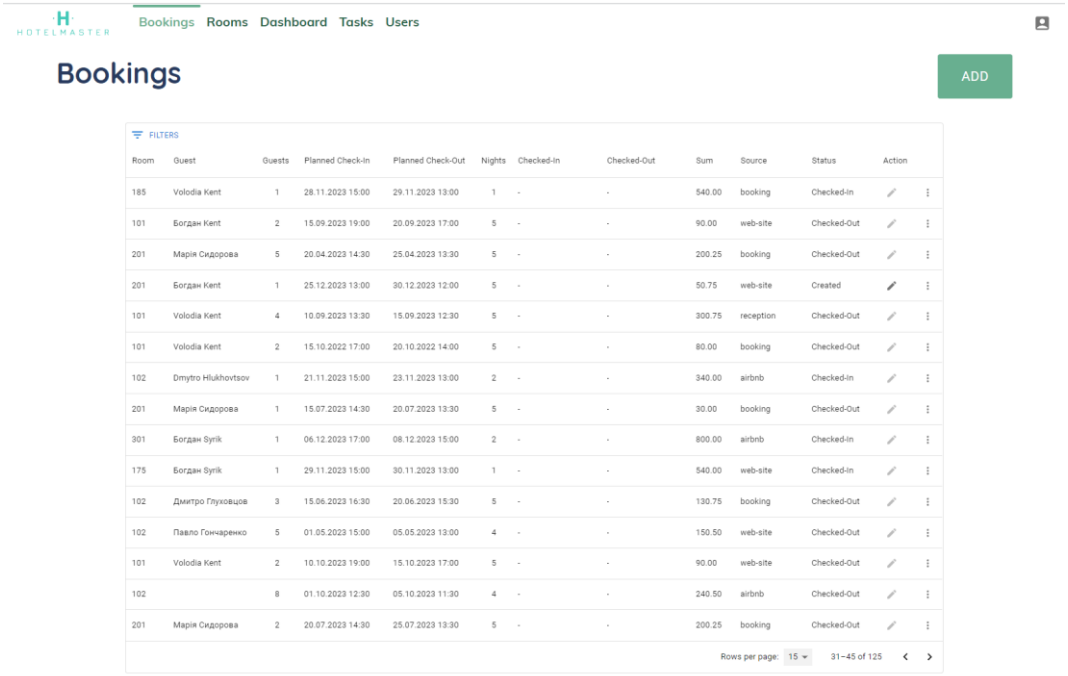


Рисунок 4.28 – Вигляд сторінки з бронюваннями

Джерело: побудовано автором (знімок з екрану)

У залежності від статусу бронювання адміністратор може відповідно редагувати його, змінювати статус або відміняти (наприклад, бронювання має статус «Created» або дата його виселення не раніше поточної, тож можна виконати будь-яку з перелічених дій) (рис. 4.29). Або також є можливості редагувати (рис. 4.30) та створювати нове бронювання (рис. 4.31).

Для цієї форми було застосовано декілька кастомних компонентів, які походять від об’єктів з бібліотеки MUI. Наприклад, поле користувача, яке після вибору існуючого користувача автоматично заповнює доступні поля (рис. 4.32), та поле кімнати, після вибору якої підвантажуються зайняті дати в календарі (рис. 4.33).

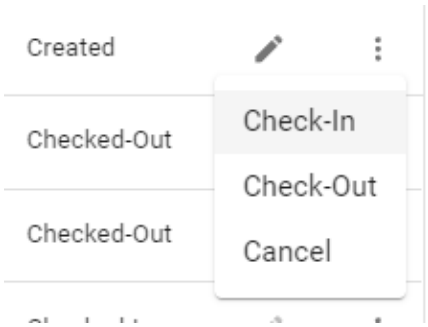


Рисунок 4.29 – Вигляд підменю для бронювання зі статусом «Created»

Джерело: побудовано автором (знімок з екрану)

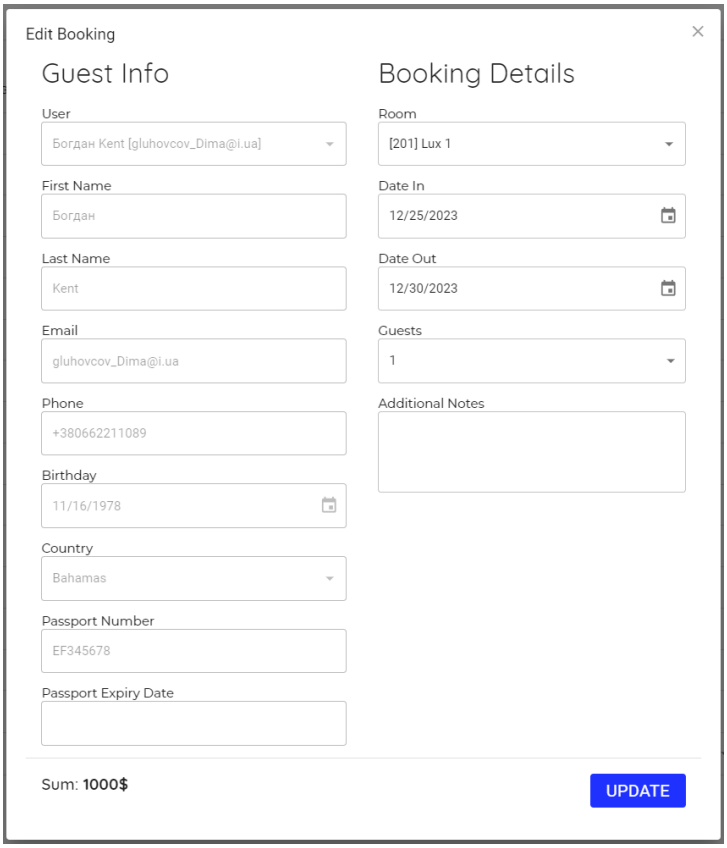


Рисунок 4.30 – Вигляд модального вікна для редагування бронювання

Джерело: побудовано автором (знімок з екрану)

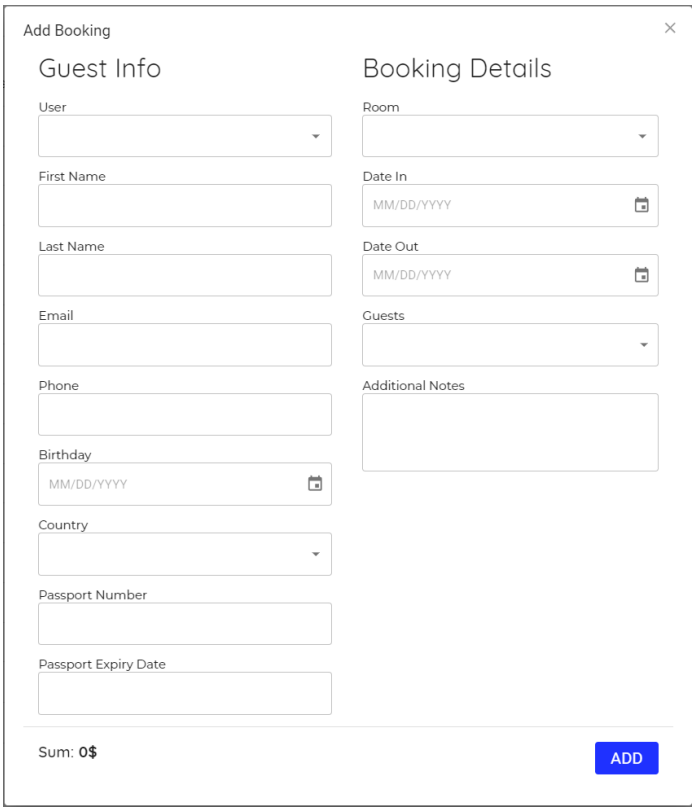


Рисунок 4.31 – Вигляд модального вікна для створення бронювання

Джерело: побудовано автором (знімок з екрану)

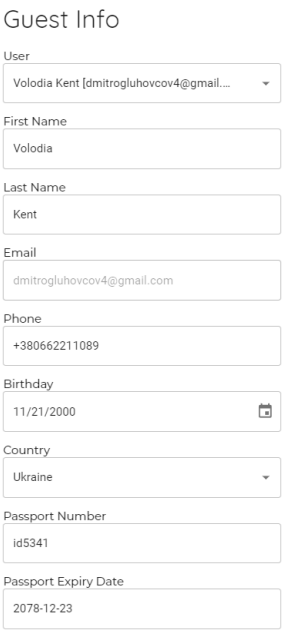


Рисунок 4.32 – Вигляд інформації про користувача під час додавання бронювання

Джерело: побудовано автором (знімок з екрану)

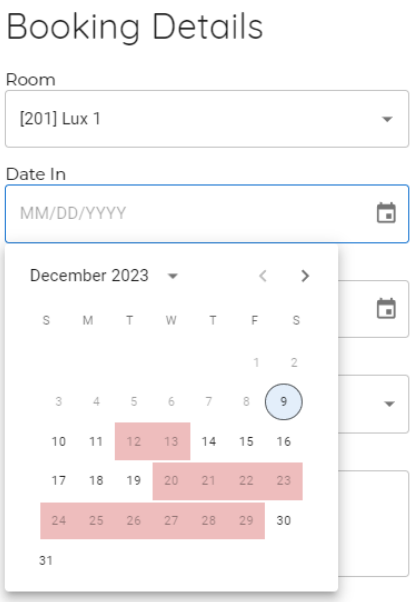


Рисунок 4.33 – Вигляд поля для введення дати після вибору кімнати

Джерело: побудовано автором (знімок з екрану)

Також, для таблиці з бронюваннями є можливість відфільтрувати дані за значеннями будь-якої колонки. Наприклад, вивести лише ті, які в статусі «Checked-In» (рис. 4.34).

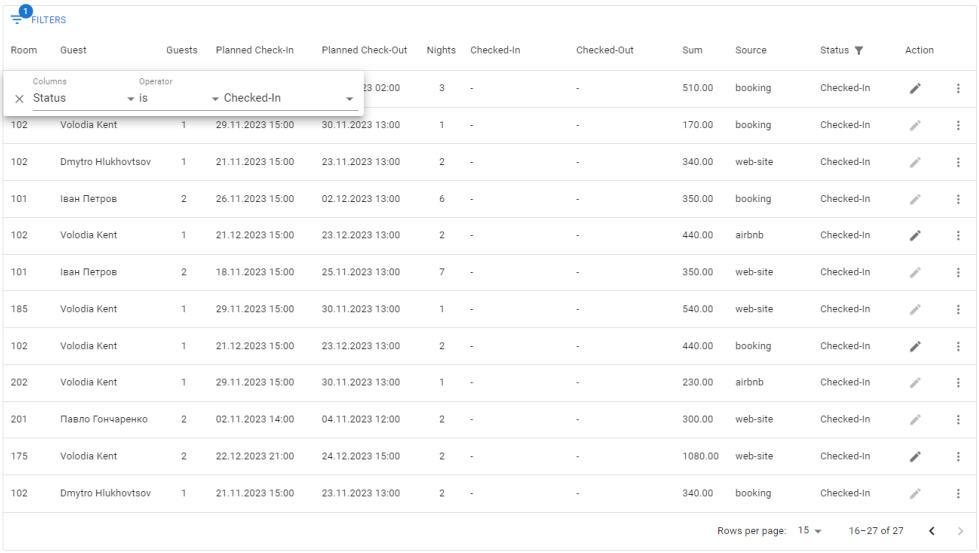


Рисунок 4.34 – Вигляд таблиці після фільтрації

Джерело: побудовано автором (знімок з екрану)

Наступною є сторінка з таблицею, яка містить основну інформацію про кімнати та є доступним функціонал детального її перегляду, редагування, додавання та видалення (рис. 4.35).

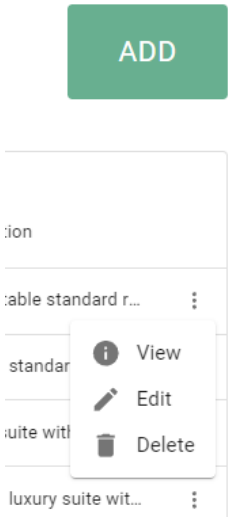


Рисунок 4.35 – Вигляд функціональних кнопок таблиці з кімнатами

Джерело: побудовано автором (знімок з екрану)

На рисунках 4.36 та 4.37 зображено саме вигляд модального вікна для перегляду детальної інформації про кімнату та вікно для її редагування відповідно.

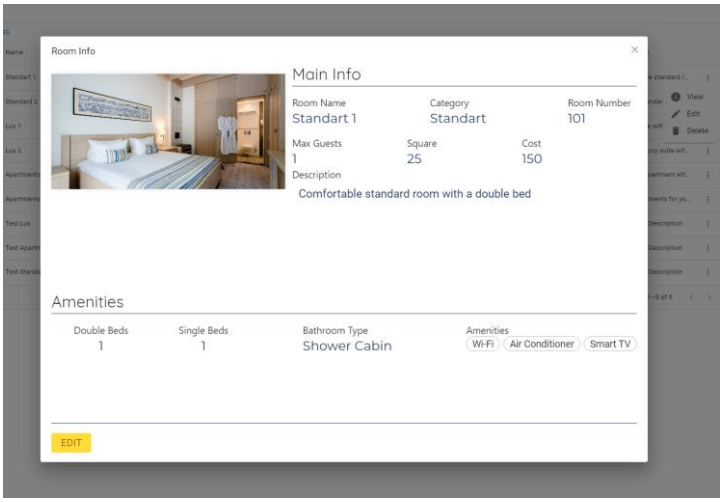


Рисунок 4.36 – Вигляд вікна перегляду інформації про кімнату

Джерело: побудовано автором (знімок з екрану)

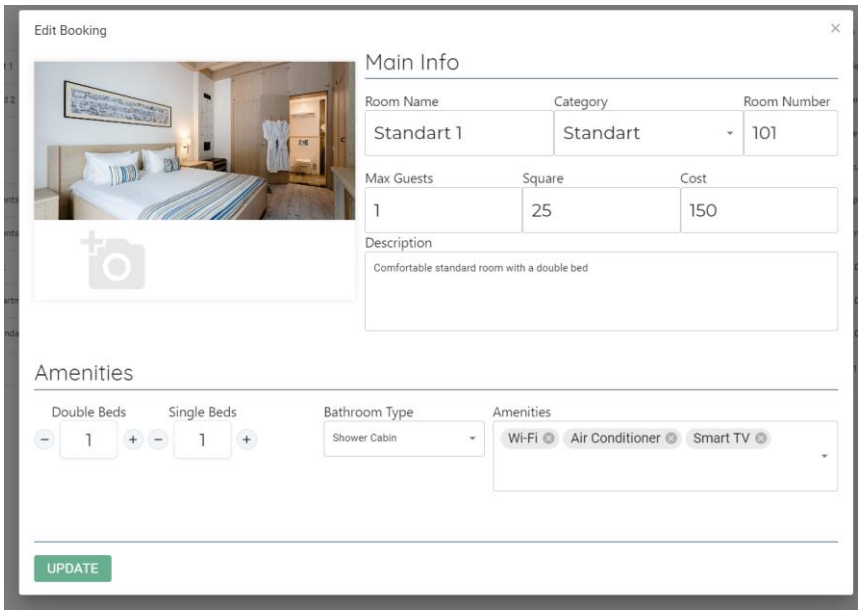


Рисунок 4.37 – Вигляд вікна редагування інформації про кімнату

Джерело: побудовано автором (знімок з екрану)

Також доступною є можливість додавати та видаляти фото кімнати. Перед цим потрібно клікнути на відповідну кнопку з плюсиком, обрати потрібне зображення та, за необхідності, обрізати його (рис. 4.38).

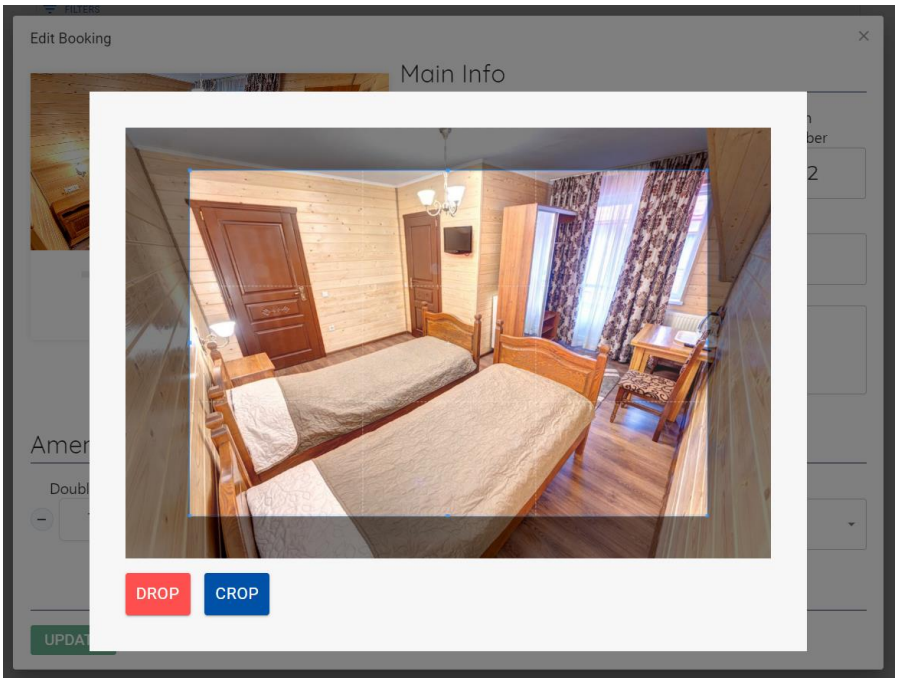


Рисунок 4.38 – Вигляд обрізки фото для додавання до кімнати

Джерело: побудовано автором (знімок з екрану)

Наступним пунктом є «Dashboard». Це вкладка, на якій зображено основну інформацію в графічному вигляді про деякі метрики та показники. Наприклад, поточну кількість заселень і виселень, статистику відвідувань за минулий рік по місяцях та статистику надходження бронювань із різних джерел (рис. 4.39-4.40).

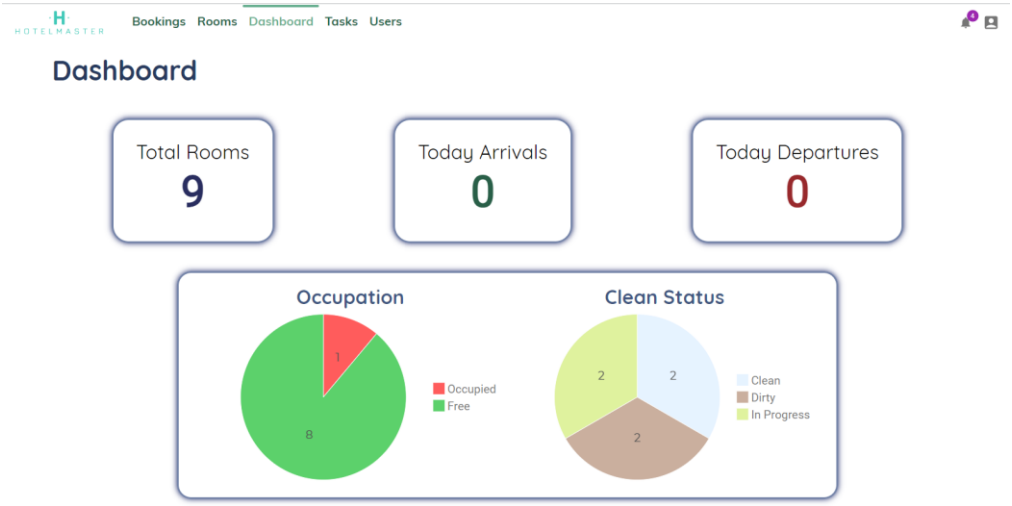


Рисунок 4.39 – Вигляд візуальної інформації про кімнати

Джерело: побудовано автором (знімок з екрану)

Рисунок 4.40 – Вигляд візуальної інформації про бронювання

Джерело: побудовано автором (знімок з екрану)

Наступною є вкладка з задачами на прибирання для покоївок. Вона містить основну інформацію про завдання та різноманітні статуси кімнати (рис. 4.41). Також доступною є можливість для редагування та додавання нових задач (4.42).

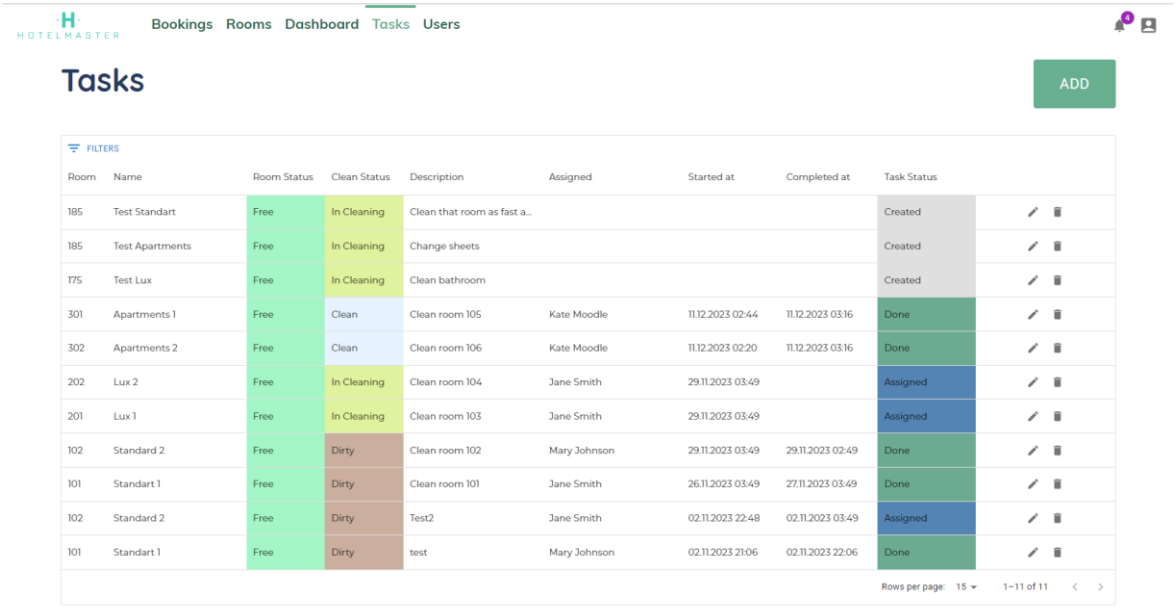


Рисунок 4.41 – Вигляд таблиці з інформацією про задачі покоївкам

Джерело: побудовано автором (знімок з екрану)

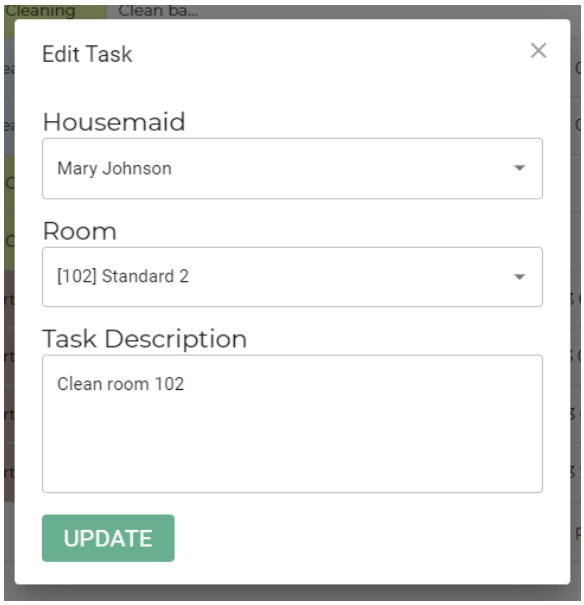


Рисунок 4.42 – Вигляд вікна для зміни інформації завдання

Джерело: побудовано автором (знімок з екрану)

Ще одним функціональним модулем є сторінка з користувачами даної системи. На ній можна переглянути, які з них є наявними в ній (рис. 4.43). Доступною є можливість додати нового користувача (рис. 4.44). Також можна змінити інформацію та перевизначити роль. Таких користувачів, як адміністратор та покоївка можна створити лише через представлену панель.

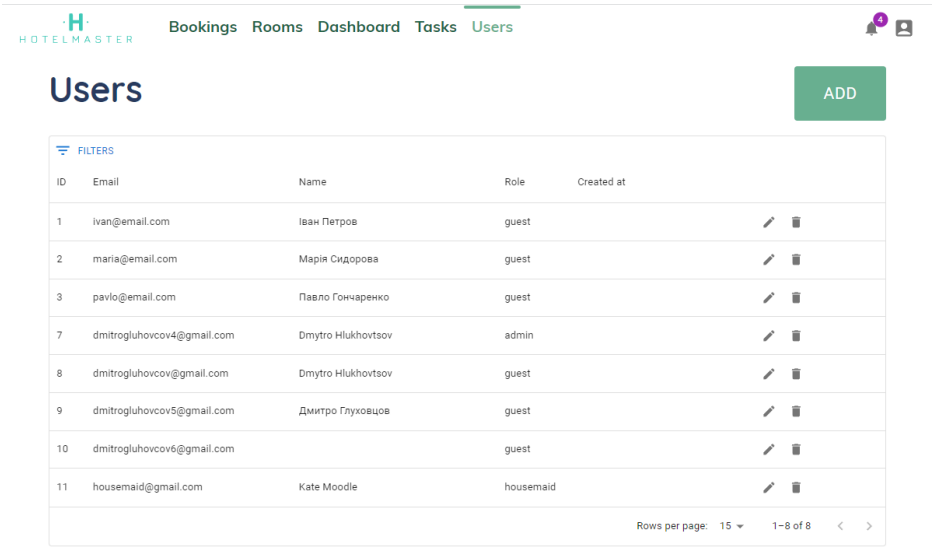


Рисунок 4.43 – Вигляд сторінки з користувачами

Джерело: побудовано автором (знімок з екрану)

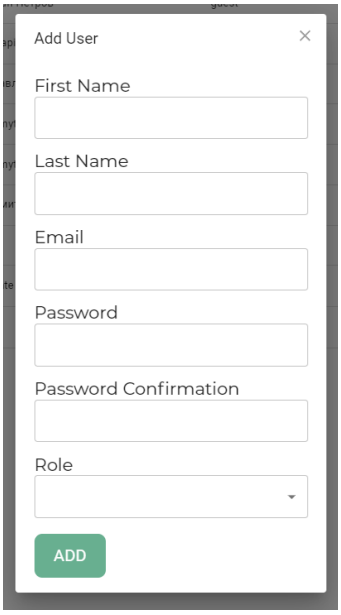


Рисунок 4.44 – Вигляд форми додавання нового користувача

Джерело: побудовано автором (знімок з екрану)

Для покоївки є доступною така ж сторінка з задачами (рис. 4.45). Однак із обмеженим функціоналом. Тобто, вона може переглядати свої задачі та ті, які ще без виконавця, а також, змінювати статус виконання та призначати завдання на себе (рис. 4.46).

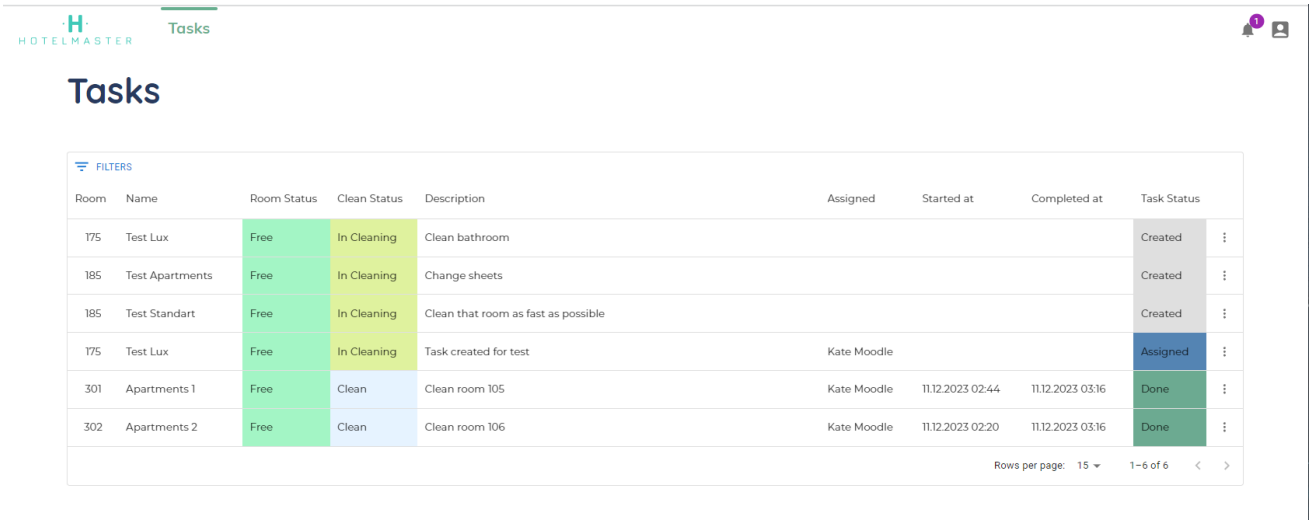


Рисунок 4.45 – Вигляд сторінки з задачами для користувача покоївки

Джерело: побудовано автором (знімок з екрану)

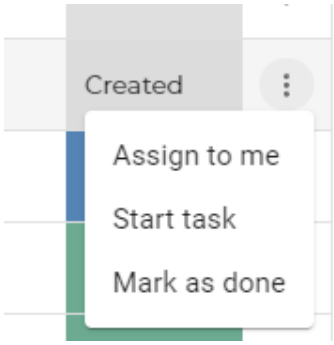


Рисунок 4.46 – Вигляд функціональних кнопок для зміни статусу завдання

Джерело: побудовано автором (знімок з екрану)

Ще одним функціональним компонентом, який реалізовано як для покоївок, так і для адміністратора, є сповіщення. Для адміністратора вони стають доступними у випадку надходження нового бронювання з боку клієнта або через API (рис. 4.47). Також при зміні покоївкою статусу задачі на прибирання (рис. 4.48). Це зручно для

відслідковування виконання таких завдань у реальному часі. Покоївці сповіщення приходять, коли додається нова задача для неї (рис. 4.49).

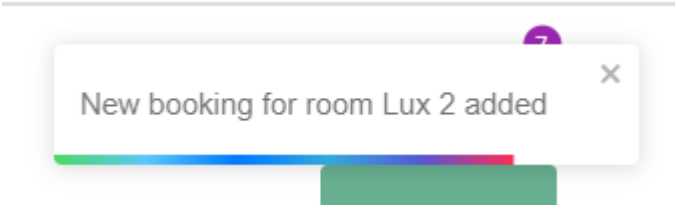


Рисунок 4.47 – Вигляд сповіщення адміністратору
про нове бронювання
Джерело: побудовано автором (знімок з екрану)

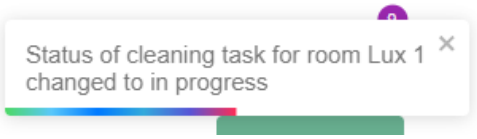


Рисунок 4.48 – Вигляд сповіщення про зміну
статусу завдання на прибирання
Джерело: побудовано автором (знімок з екрану)

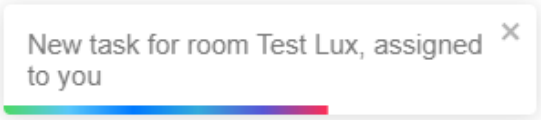


Рисунок 4.49 – Вигляд сповіщення про створення
нового завдання для покоївки
Джерело: побудовано автором (знімок з екрану)

Також для обох цих користувачів існує список із сповіщеннями. Там зберігаються всі, які надійшли. Користувач може позначати їх як прочитані, щоб не була активною позначка непрочитаних повідомлень, а також видаляти непотрібні (рис. 4.50).

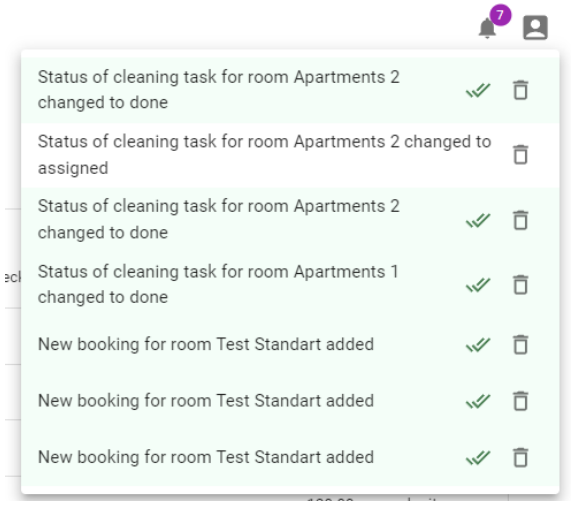


Рисунок 4.50 – Вигляд контейнера з сповіщеннями

Джерело: побудовано автором (знімок з екрану)

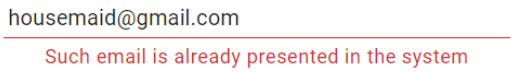
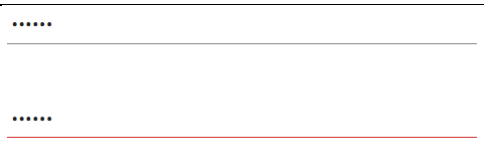
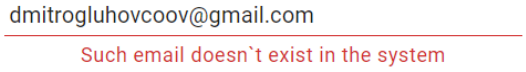
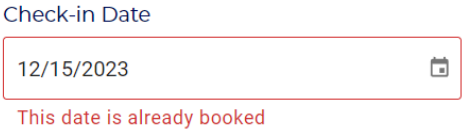
4.5 Тестування web-орієнтованої системи

Після виконання програмної реалізації необхідно дослідити працездатність розробленого прграмного продукту та відповідність очікуваним результатам роботи. Тому було проведено функціональне тестування. При цьому досліджено коректність роботи посилань та маршрутизації, адекватність реакцій на введення невалідних даних до форм та якість працездатності основного функціоналу.

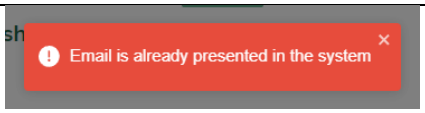
Першочергово, було переглянуто функції реєстрації та авторизації. Також коректність відпрацювання валідаційних правил у різноманітних формах. Перевірено те, як дана система переспрямовує користувача в залежності від його ролі, а також при кліках на посилання. Розглянуто можливості календаря, а також коректність відпрацювання через API за допомогою платформи Postman [57]. Також, перевірено функціонування сповіщень та їх правильне надходження до адресатів.

Під час тестування застосовувався метод «чорного ящика» [58]. Тобто перевірялася функціональність представленої web-орієнтованої системи без доступу до коду. Основні результати проведеного тестування представлено в таблиці 4.1.

Таблиця 4.1 – Результати тестування web-орієнтованої системи

№	Назва	Очікуваний результат	Фактичний результат	0/1
1	Перевірка форми реєстрації на введений email вже існуючий у системі	Поле підсвічено червоним та виведено відповідне повідомлення про існування такого email		1
2	Перевірка форми реєстрації на невідповідність паролів	Поля підсвічено червоним і виведено відповідне повідомлення про неправильне введення даних		1
3	Перевірка форми авторизації на введення email не існуючого в системі	Виведення відповідного повідомлення про відсутність такого логіну		1
4	Перевірка форми авторизації на невірний пароль	Виведення повідомлення про неправильно введені дані		1
5	Перевірка форми на введення валідних даних авторизації	Переадресація до сторінки, що відповідає ролі авторизованого користувача	Відбувається переадресація на відповідну ролі початкову сторінку	1
6	Перевірка форми бронювання на неможливість забронювати на зайняту дату	Виведення повідомлення про те, що дата вже зайнята		1
7	Перевірка роботи фільтру на сторінці з кімнатами для гостя	Фільтрація відбувається успішно	Дані в списку виведених кімнат відфільтровано та отримано вибірку з відповідними фільтру	1

Продовження табл. 4.1

№	Назва	Очікуваний результат	Фактичний результат	0/1
8	Перевірка коректної зміни статусу чистоти кімнати після зміни статусу завдання покоївкою	Перехід до коректного статусу	Статус чистоти кімнати змінено відповідно до статусу завдання пов’язаного з нею	1
9	Перевірка неможливості додати нового користувача з вже існуючим email адміну	Виведення повідомлення про існування такого користувача		1
10	Перевірка надходження сповіщення про нове бронювання в системі всім адміністраторам	Спливання сповіщення у правій верхній частині екрану та його поява у списку сповіщень	Нове сповіщення стало доступним для всіх адміністраторів	1
10	Перевірка надходження сповіщення про нове завдання на прибирання для покоївки	Спливання сповіщення у правій верхній частині екрану його поява у списку сповіщень	Нове сповіщення стало досутпним для відповідної покоївки	1
11	Перевірка на неможливість додати бронювання через Postman на вже зайняті дати	У відповіді сервера є поле з помилкою та, власне, сам текст помилки		1

Після проведення тестування істотних багів не було виявлено й все працює відповідно до поставлених функціональних вимог.

ВИСНОВКИ

У ході виконання кваліфікаційної дипломної роботи було здійснено проєктування структури та програмну реалізацію web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел відповідно до таких основних функціональних вимог:

– перегляд всіх кімнат у номерному фонді даного готелю
– перегляд вільних кімнат у задані гостем дати;
– можливість бронювання номеру на потрібну дату;
– управління бронюваннями з боку адміністратора (їх реєстрацію, редагування та створення нових);
– наявна для адміністратора можливість проведення маніпуляцій із даними про номерний фонд (редагування інформації, її видалення та додавання нової);
– перегляд статистики про прибутковість і популярність номерів, а також інформації про відвідувачів готелю у графічній формі (у діаграмах та таблицях);
– модуль для забезпечення управління та контролю роботи покоївок.

Створений програмний продукт є актуальним для готельного бізнесу, адже дозволить автоматизувати вищеперелічені процеси, а також систематизувати всі дані в одному місці з можливістю оперування ними.

Під час виконання даної роботи було визначено потребу в розробці web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел у підсумку проведення детального аналізу предметної області. Також досліджено декілька аналогів. У результаті було визначено їхні переваги та недоліки. Цю інформацію враховано при розробці власної web-орієнтованої системи.

Після цього було формалізовано мету проєкту, представлено функціональні та нефункціональні вимоги до системи та сформовано задачі на реалізацію. Проведено детальне планування робіт проєкту.

Наступним етапом було змодельовано роботу реальних кейсів і варіанти використання даної web-орієнтованої системи за допомогою діаграм нотації IDEF0 і її декомпозицій. Це допомогло пропрацювати список ролей для даного програмного продукту, рівні доступу та функціональні вимоги для кожної з них, а також декомпозувати задачі на атомарні.

Далі була визначено архітектура даної web-орієнтованої системи, створена база даних на основі її моделі, програмно реалізований функціонал, що відповідає визначеним вимогам. Також організовано коректну роботу фронтенду та бекенду розроблюваної системи, а також взаємодію між ними. Розробка відбувалася за допомогою сучасних засобів та інструментів.

Після цього було проведено тестування для визначення рівня якості розробленої web-орієнтованої системи. У підсумку проведених заходів було підтверджено коректну її роботу, яка відповідає поставленим функціональним вимогам.

Отже, використання web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел дозволить зменшити час, який витрачає адміністратор на виконання рутинних процесів, надасть механізм перегляду статистики в одному місці, допоможе у формуванні завдань для обслуговуючого персоналу, а також представить гостям функціонал для зручного бронювання номерів, спростить здійснення збору інформації для покращення робочих процесів та аналізу прибутковості.

Результати роботи були апробовані на науково-практичній конференції «КІТ-2023» у Харківському національному автомобільно-дорожньому університеті в м. Харків (Додаток Б).

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Тищенко С. В. Цифрові технології в індустрії гостинності. Таврійський науковий вісник. серія: економіка. 2021. № 7. С. 131-139. URL: https://doi.org/10.32851/2708-0366/2021.7.16 (дата звернення: 09.11.2023).

2. Kansakar P., Munir A., Shabani N. Technology in the Hospitality Industry: Prospects and Challenges. *IEEE Consumer Electronics Magazine*. 2019. Vol. 8, no. 3. P. 60-65. URL: https://doi.org/10.1109/mce.2019.2892245 (date of access: 12.11.2023).

3. Milovanović V. Hotel management in the twenty-first century: opportunities, threats, and implications. *Crisis after the crisis: economic development in the new normal*. Cham, 2023. P. 287-298. URL: https://doi.org/10.1007/978-3-031-30996-0_21 (date of access: 12.11.2023).

4. Nesterenko S., Bocharova N., Yarchuk A. Modern aspects of management and administration in the hotel and restaurant business. *Scientific papers OF DMYTRO MOTORNYI TAVRIA STATE AGROTECHNOLOGICAL UNIVERSITY (ECONOMIC SCIENCES)*. 2021. Vol. 43. P. 82-87. URL: https://doi.org/10.31388/2519-884x-2021-43-82-87 (date of access: 14.11.2023)

5. Kaschuck K. M., Mosiichuk I. V., Saukh I. V. Modern management technologies in the hotel and restaurant business: practices and innovations. *Business inform*. 2023. Vol. 6, no. 545. P. 93-99. URL: https://doi.org/10.32983/2222-4459-2023-6-93-99 (date of access: 13.11.2023).

6. Фостолович В. Сучасні інструменти системи управління бізнесом у сфері готельно-ресторанної справи modern tools of business management system in the field of hotel and restaurant affairs. *Інвестиції: практика та досвід №11*.

2022. С. 18-25. URL: https://doi.org/10.32702/2306-6814.2022.11-12.18 (дата звернення: 03.11.2023).

7. Patrick M., Paulo W. M., Sumarwan A. How digital technology enhances hotel experience? Examining experiential marketing of two four-star hotels in yogyakarta. *E-Journal of tourism*. 2023. P. 33. URL: https://doi.org/10.24922/eot.v10i1.98359 (date of access: 15.11.2023).

8. M. Ara, M. Foysal, M. A. Huda and S. Bairagi, «A model to develop hotel management system to optimize revenue», Khulna university Studies, vol. 19, no. 2. pp. 167-188, 2022

9. Inversini A., De Carlo M., Masiero L. The effects of customer-centricity in hospitality. *International journal of hospitality management*. 2020. Vol. 86. P. 102436. URL: https://doi.org/10.1016/j.ijhm.2019.102436 (date of access: 03.11.2023).

10. Customer-Centricity: the quest for customer focus / M. K. Dash et al. *Customer-Centricity in organized retailing*. Singapore, 2023. P. 9-23. URL: https://doi.org/10.1007/978-981-19-3593-0_2 (date of access: 13.11.2023).

11. Perišić Prodan M. IMPLEMENTATION OF CUSTOMER ORIENTATION IN THE HOSPITALITY INDUSTRY: IMPLICATIONS FOR BUSINESS PERFORMANCE. *4th International Scientific Conference – EMAN 2020 – Economics and Management: How to Cope With Disrupted Times*. 2020.

12. Teng A. Hotel reservation management information system based on CRM. *Lecture notes on data engineering and communications technologies*. Cham, 2023. P. 189-197.

13. Chourasia S., Shrivastava A., Jodhana L. S. The impact of consumers intention on online booking. *International journal of social science & management studies*. 2023. Vol. 9, no. 3. P. 73-80. URL: https://www.researchgate.net/publication/373601903_The_Impact_of_Consumers_Intention_on_Online_Booking(date of access: 15.11.2023)

14. Forman N., Udvaros J. Digital innovation in hospitality: bridging the gap between concierge services and hotel guests. *Journal of environmental management and tourism*. 2023. Vol. 14, no. 6. P. 2673-2673.

15. Chopde A. M., Arote M. Automated and efficient hotel management system. *Journal of information technology and cryptography*. 2023. Vol. 1, no. 1. P. 22-30.

16. Jiang X. Hotel human resource management strategy during the pandemic: a case study of hilton hotel. Advances in economics, management and political sciences. 2023. Vol. 15, no. 1. P. 265-269.

17. Huhul O. Innovative approaches to personnel management in the hotel and restaurant business. *Innovative economy*. 2023. No. 2. P. 110-115. URL: https://doi.org/10.37332/2309-1533.2023.2.14 (date of access: 15.11.2023).

18. Гірняк Л. І., Сопіга В. Б. Формування культури та якості обслуговування у готельно-ресторанних підприємствах. *Науковий вісник Ужгородського національного університету. Серія : міжнародні економічні відносини та світове господарство*. 2018. № 21(1). С. 50-55.

19. Viktor S., Aliluiko M. Integration of management systems in the hotel and restaurant enterprises. *Problems of systemic approach in the economy*. 2019. No. 4 (72). URL: https://doi.org/10.32782/2520-2200/2019-4-46 (date of access: 14.11.2023).

20. Betrand C. U., Onyema C. J., Awaji N. M. Automated hotel management information system. Journal of research in engineering and computer sciences. 2023. Vol. 1, no. 1. P. 19-28. URL: https://www.researchgate.net/publication/370801940_Automated_Hotel_Management_Information_System (date of access: 15.11.2023)

21. Січка І. Сучасні технології в індустрії гостинності. *Географія, економіка і туризм: національний та міжнародний досвід*. 2020. С. 317-321.

22. Доценко О. АВТОМАТИЗОВАНІ СИСТЕМИ УПРАВЛІННЯ ГОТЕЛЯМИ В УКРАЇНІ (за матеріалами готелю «HOTEL 39» у м.

львові). *РЕАЛІЇ, ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ ГЕОГРАФІЇ ТА ТУРИЗМУ В УКРАЇНІ Матеріали XXI-ої Всеукраїнської студентської наукової конференції*. 2020. С. 87-91. URL: https://geography.lnu.edu.ua/wp-content/uploads/2022/10/2020_Stud-konf21-Realiyi-persp.pdf#page=87 (дата звернення: 15.11.2023).

23. The level of hotel reservation synchronization processes using information technology tools. *ResearchGate*. URL: https://www.researchgate.net/publication/327393657_THE_LEVEL_OF_HOTEL_RESERVATION_SYNCHRONIZATION_PROCESSES_USING_INFORMATION_TECHNOLOGY_TOOLS (date of access: 17.11.2023).

24. Han S., Anderson C. K. The effect of private customer-manager social engagement upon online booking behavior. *Cornell hospitality quarterly*. 2020. P. 193896552097533. URL: https://doi.org/10.1177/1938965520975330 (date of access: 14.11.2023).

25. tanasova I., Ivanov I. “Hotel – OTA” interaction and its influence on the optimization of urban hotels’ distribution channels. *SHS web of conferences*. 2021. Vol. 129. P. 11. URL: https://doi.org/10.1051/shsconf/202112906001 (date of access: 16.11.2023).

26. Buhalis D., Leung R. Smart hospitality–Interconnectivity and interoperability towards an ecosystem. *International journal of hospitality management*. 2018. Vol. 71. P. 41-50. URL: https://doi.org/10.1016/J.IJHM.2017.11.011 (date of access: 15.11.2023).

27. Battiti R., Brunato M., Battiti F. RoomTetris: an optimal procedure for committing rooms to reservations in hotels. *Journal of hospitality and tourism technology*. 2020. Vol. 11, no. 4. P. 589-602. URL: https://doi.org/10.1108/jhtt-08-2019-0108 (date of access: 16.11.2023).

28. Tereshchuk N., Tranchenko L. Automation of business processes as a mechanism to increase the efficiency of a hotel enterprise. *Innovations and technologies in the service sphere and food industry*. 2021. No. 1-2(3-4).

URL: https://doi.org/10.24025/2708-4949.1-2(3-4).2021.241949 (date of access: 16.11.2023).

29. ПЕРЕВАГИ ТА НЕДОЛІКИ WEB-ДОДАТКІВ - Проєктування і розробка web-додатків - Підручники для студентів онлайн. URL: https://stud.com.ua/97611/informatika/perevagi_nedoliki_dodatkiv (дата звернення: 16.11.2023).

30. Research on hotel management system / W. P. S. W. Weerasinghe et al. *International journal of engineering and management research*. 2022. Vol. 12, no. 5. P. 218-226. URL: https://doi.org/10.31033/ijemr.12.5.27 (date of access: 17.11.2023).

31. Глуховцов Д. О., Антипенко В. П. Web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних із декількох джерел. *Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві* : Матеріали всеукр. науково-практ. конф. здобувачів вищ. освіти і молодих уч., м. Харків, 22 листоп. 2023 р. Харків, 2023. С. 139–142. URL: https://mf.khadi.kharkov.ua/fileadmin/F-MECHANIC/Автоматизації_та_комп'ютерно-інтегрованих_технологій/publications/Conf_AKIT/Матеріали__КІТ_2023.pdf (дата звернення: 16.11.2023)

32. SERVIO HMS - система автоматизації гостинності. URL: https://expertsolution.com.ua/uk/modul-servio-hms (дата звернення: 16.11.2023).

33. Hotel management software - all-in-one cloud PMS - easypms. URL: http://www.easypms.com/ru/homeru/ (дата звернення: 09.11.2023).

34. HotelFriend property management system - hotel software. URL: https://hotelfriend.com/ (дата звернення: 16.11.2023).

35. Медвідь В. Ю., Данько Ю. І., Коблянська І. І. Методологія та організація наукових досліджень (у структурно-логічних схемах і таблицях) : навч. посіб. Суми : Унів. кн., 2020. 220 с.

36. React. URL: https://react.dev/ (дата звернення: 17.11.2023).

37. Що таке SCSS і які його особливості. URL: https://highload.today/uk/scss/ (дата звернення: 17.11.2023).

38. Overview - material UI. MUI: The React component library you always wanted. URL: https://mui.com/material-ui/getting-started/ (date of access: 17.11.2023).

39. Що таке Node JS простими словами - застосування Node JS у програмуванні | DAN-IT. URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення: 17.11.2023).

40. What is PostgreSQL. URL: https://www.postgresqltutorial.com/post-gresql-getting-started/what-is-postgresql/ (date of access: 17.11.2023).

41. Методологія idef0. *Підручники для студентів онлайн*. URL: https://stud.com.ua/87184/ekonomika/metodologiya_idef0. (дата звернення: 17.11.2023).

42. Діаграма прецедентів – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 18.11.2023).

43. Починаючи | axios docs. *Axios*. URL: https://axios-http.com/uk/docs/intro (дата звернення: 19.11.2023).

44. Express - Node.js web application framework. *Express - Node.js web application framework*. URL: https://expressjs.com/ (date of access: 19.11.2023).

45. GitHub: let’s build from here. *GitHub*. URL: https://github.com/ (date of access: 19.11.2023).

46. JetBrains. WebStorm. *JetBrains*. URL: https://www.jetbrains.com/ru-ru/webstorm/ (date of access: 19.11.2023).

47. Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter. *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*. URL: https://eslint.org/ (date of access: 20.11.2023).

48. ChatGPT. *OpenAI*. URL: https://openai.com/chatgpt (date of access: 21.11.2023).

49. Що таке API? Просте пояснення від Петра Газарова. dev.ua. URL: https://dev.ua/news/chto-takoe-api-prostym-yazykom (дата звернення: 21.11.2023)

50. SQL query builder for javascript | knex.js. *SQL Query Builder for Javascript | Knex.js*. URL: https://knexjs.org/ (date of access: 21.11.2023).

51. Passport.js. *Passport.js*. URL: https://www.passportjs.org/ (date of access: 25.11.2023).

52. Плюси та мінуси JWT: короткий огляд тонкощів цієї технології - highload.today. *Highload.today - медіа для розробників*. URL: https://highload.today/uk/plyusy-i-minusy-jwt-kratkij-obzor/ (дата звернення: 23.11.2023).

53. Ably: the platform to power synchronized digital experiences in realtime | Ably Realtime. *Ably Realtime*. URL: https://ably.com/ (date of access: 25.11.2023).

54. Multer. *npm*. URL: https://www.npmjs.com/package/multer (date of access: 26.11.2023).

55. Cloud storage. *Google Cloud*. URL: https://cloud.google.com/storage?hl=ru (date of access: 26.11.2023).

56. Dmytro-Hlukhovtsov/hotel-master. URL: https://github.com/Dmytro-Hlukhovtsov/hotel-master (date of access: 26.11.2023).

57. Postman. URL: https://www.postman.com/ (дата звернення: 27.11.2023).

58. Яка різниця між black box & white box testing?. URL: https://training.qatestlab.com/blog/technical-articles/whats-the-difference-between-black-box-white-box-testing/ (дата звернення: 28.11.2023).

ДОДАТОК А

Планування робіт

Попит на автоматизацію процесів всередині компаній та бізнесу, а також на механізми, які допоможуть оперувати даними, стрімко зростає з кожним роком. Але в свою чергу подібні рішення є не лише засобом досягнення прибутку. Насамперед це допомагає збільшити ефективність робочого персоналу, знизити навантаження за рахунок зменшення часу на виконання рутинних процесів тощо. Саме тому, завдяки створенню web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів можна досягти як і забезпечення гнучкості бізнесу, так і його прибутковості за рахунок надання інструментарію для більш продуктивної роботи всіх складових цієї розробки.

Деталізація мети методом SMART. Для збільшення успішності та ефективності проєкту є досить важливим визначити його мету, у чому може допомогти SMART-метод. Результат деталізації цього методу зображено в таблиці А.1. Мета за SMART-методом має таке формулювання: «Розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел на основі затверджених функціональних компонентів для забезпечення автоматизації робочих процесів даного закладу та їх належної організації до 15 грудня 2023 року».

Таблиця А.1 – Деталізація мети проєкту методом SMART

Specific (конкретна)	Розробка web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел та забезпечення продуктивної роботи адміністратора за рахунок її належної організації.
Measurable (вимірювана)	Результатом роботи має бути розроблена web-орієнтована система підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів, яка містить модулі бронювання та резервування номерів, управління завданнями для покоївок, редагування інформації про користувачів та кімнати, а також модуль для перегляду статистичної інформації.
Achievable (досяжна, узгоджена)	Для розробки проєкту потрібні знання HTML, CSS, мови програмування JavaScript(React, Node.js), Post-greSQL та навички написання документації. Також є затверджене технічне завдання, враховані та розподілені наявні ресурси, тому ціль є досяжною.
Relevant (реалістична)	Створена web-орієнтована система допоможе автоматизувати та пришвидшити такі процеси, як робота з бронюваннями, збір та перегляд статистики, створення та редагування специфічних даних, а також сприятиме збільшенню прибутку за рахунок належної організації автоматизації внутрішніх процесів туристичного готелю.
Time-framed (обмежена в часі)	Термін досягнення мети проєкту визначено за навчальним планом, закінчити проєкт до 15 грудня 2023 року.

Джерело: побудовано автором

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний план компонентів проєкту, які розташовані ієрархічно й поєднуються з продуктом проєкту. Також WBS надає представлення для подальшої оцінки термінів і налаштування групової роботи.

На найвищому рівні даної структури знаходиться кінцевий результат проєкту. На другому – вже дії та заходи, які спрямовані на досягнення головної мети. Таким способом відбувається декомпозиція до того моменту, поки дії стають атомарними. Це такі, які спрямовані лише на одного виконавця, для якого можна розрахувати терміни роботи та затрати на працю, а також такими, що мають однозначний і чіткий результат. На рисунку А.1 зображено WBS з розробки web-орієнтованої системи підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів.

Планування структури виконавців. Після декомпозиції процесів необхідно розробити структуру виконавців проєкту – OBS (Organization Breakdown Structure). Вона зображується у вигляді графічної структури й відображає учасників, відповідальних за певний етап реалізації. Останні – це особи, які виконують елементарну роботу, яка зазначена в WBS. Організаційну структуру проєкту зображено на рисунку А.2.

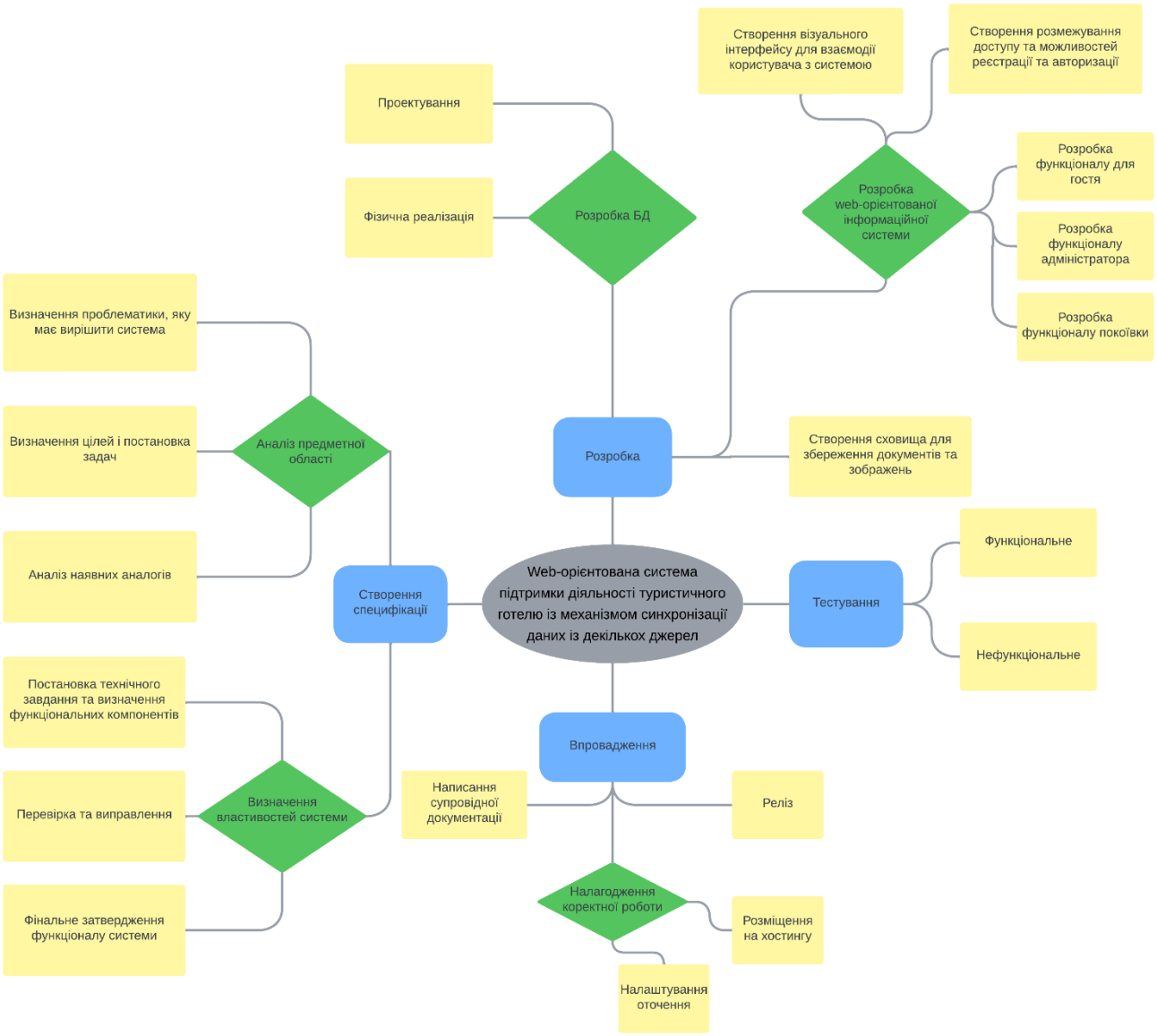


Рисунок А.1 – WBS-структура робіт проєкту

Джерело: побудовано автором (знімок з екрану)

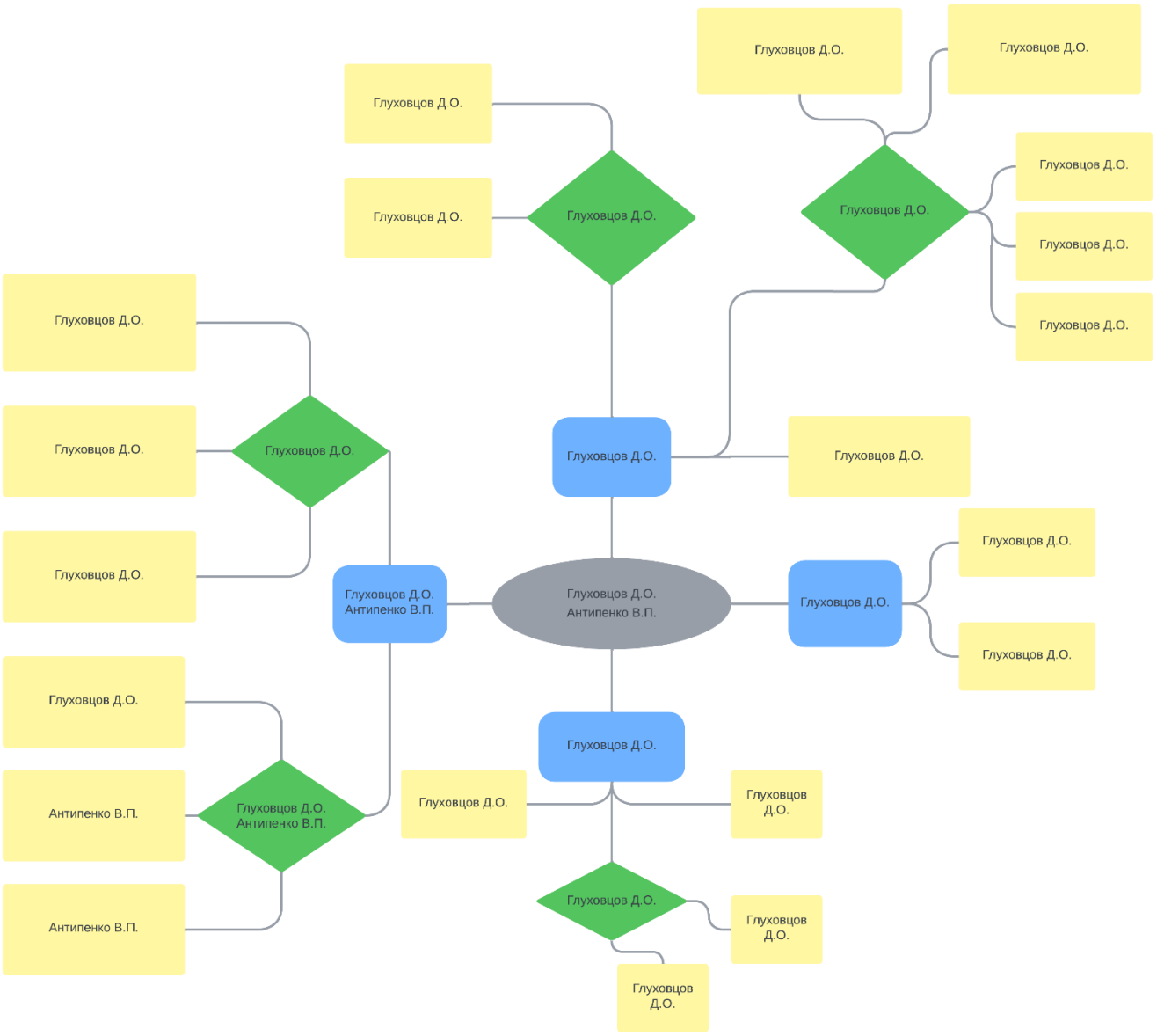


Рисунок А.2 – OBS-структура робіт проєкту

Джерело: побудовано автором (знімок з екрану)

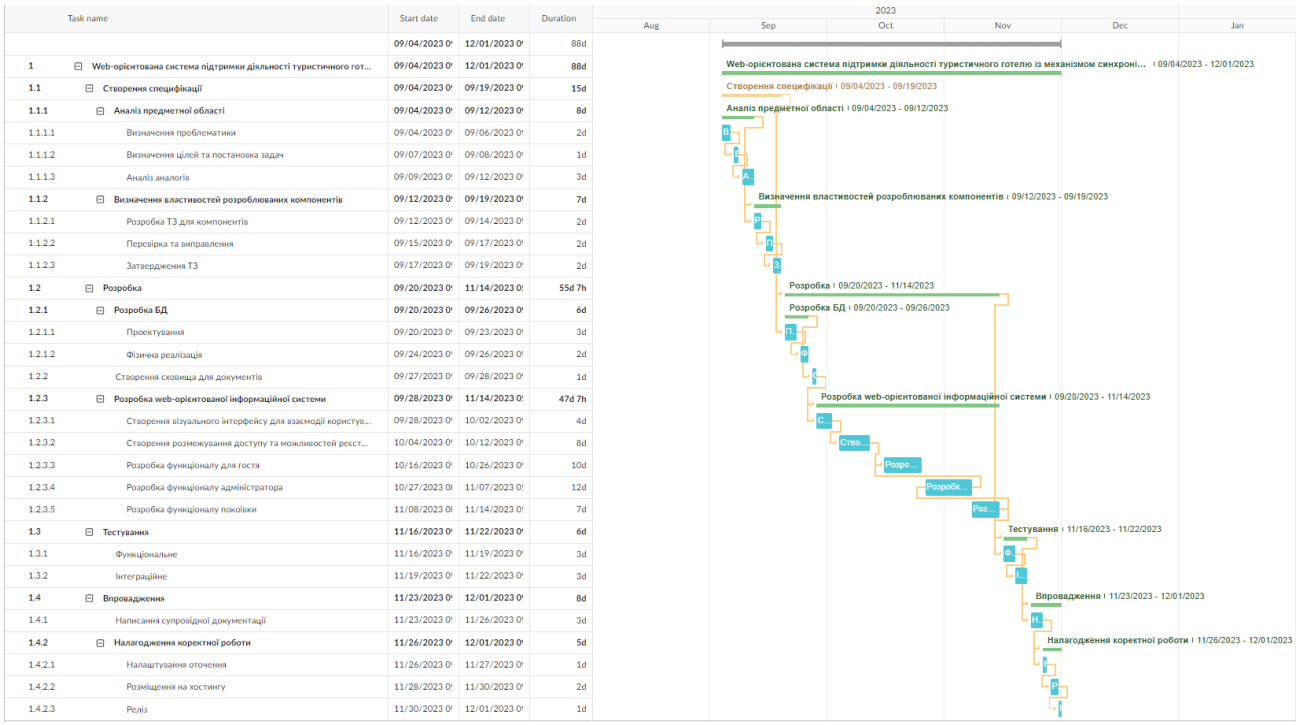
Список виконавців даного проєкту описано в таблиці А.2.

Таблиця А.2 – Виконавці проєкту

Роль	**Ім’я**	**Проєктна роль**
Розробник	Глуховцов Д.О.	Виконує розробку модулів web-орієнтованої системи.
Проєктувальник	Глуховцов Д.О	Відповідає за проєктування бази даних та структуру розроблюваної системи.
Тестувальник	Глуховцов Д.О.	Здійснює тестування дизайну та функціональності створених компонентів.
Керівник проєкту	Антипенко В.П.	Видає завдання на розробку проєкту.
Менеджер проєкту	Глуховцов Д.О.	Відповідає за розподіл завдань і ресурсів, а також за дотримання термінів та своєчасне виконання проєкту.

Джерело: побудовано автором

Діаграма Ганта. Створення календарного плану, пов’язаного з вищезгаданими діаграмами, є одним із найважливіших етапів проєкту, адже під час нього розбивається виконання робіт з прив’язкою до конкретних дат. Це дозволяє мати коректне уявлення про тривалість процесів та розподілення ресурсів, враховуючи обмеження останніх. Календарний графік даного проєкту зображено на рисунку А.3.



Task name	Start date	End date	Duration
	09/04/2023 0	12/01/2023 0	88d
1 Web-орієнтована система підтримки діяльності туристичного гот...	09/04/2023 0	12/01/2023 0	88d
1.1 Створення специфікації	09/04/2023 0	09/19/2023 0	15d
1.1.1 Аналіз предметної області	09/04/2023 0	09/12/2023 0	8d
1.1.1.1 Визначення проблематики	09/04/2023 0	09/06/2023 0	2d
1.1.1.2 Визначення цілей та постановка задач	09/07/2023 0	09/08/2023 0	1d
1.1.1.3 Аналіз аналогів	09/09/2023 0	09/12/2023 0	3d
1.1.2 Визначення властивостей розроблюваних компонентів	09/12/2023 0	09/19/2023 0	7d
1.1.2.1 Розробка ТЗ для компонентів	09/12/2023 0	09/14/2023 0	2d
1.1.2.2 Перевірка та виправлення	09/15/2023 0	09/17/2023 0	2d
1.1.2.3 Затвердження ТЗ	09/17/2023 0	09/19/2023 0	2d
1.2 Розробка	09/20/2023 0	11/14/2023 0	55d 7h
1.2.1 Розробка БД	09/20/2023 0	09/26/2023 0	6d
1.2.1.1 Проєктування	09/20/2023 0	09/23/2023 0	3d
1.2.1.2 Фізична реалізація	09/24/2023 0	09/26/2023 0	2d
1.2.2 Створення сховища для документів	09/27/2023 0	09/28/2023 0	1d
1.2.3 Розробка web-орієнтованої інформаційної системи	09/28/2023 0	11/14/2023 0	47d 7h
1.2.3.1 Створення візуального інтерфейсу для взаємодії користув...	09/28/2023 0	10/02/2023 0	4d
1.2.3.2 Створення розмежування доступу та можливостей реєст...	10/04/2023 0	10/12/2023 0	8d
1.2.3.3 Розробка функціоналу для гостя	10/16/2023 0	10/26/2023 0	10d
1.2.3.4 Розробка функціоналу адміністратора	10/27/2023 0	11/07/2023 0	12d
1.2.3.5 Розробка функціоналу покоївки	11/08/2023 0	11/14/2023 0	7d
1.3 Тестування	11/16/2023 0	11/22/2023 0	6d
1.3.1 Функціональне	11/16/2023 0	11/19/2023 0	3d
1.3.2 Інтеграційне	11/19/2023 0	11/22/2023 0	3d
1.4 Впровадження	11/23/2023 0	12/01/2023 0	8d
1.4.1 Написання супровідної документації	11/23/2023 0	11/26/2023 0	3d
1.4.2 Налагодження коректної роботи	11/26/2023 0	12/01/2023 0	5d
1.4.2.1 Налаштування оточення	11/26/2023 0	11/27/2023 0	1d
1.4.2.2 Розміщення на хостингу	11/28/2023 0	11/30/2023 0	2d
1.4.2.3 Реліз	11/30/2023 0	12/01/2023 0	1d

Рисунок А.3 – Діаграма Ганта

Джерело: побудовано автором (знімок з екрану)

Управління ризиками проєкту. Наступним важливим етапом при створенні проєкту є визначення та оцінка його ризиків. Спочатку необхідно їх ідентифікувати та створити їхній список, оцінити вплив та ймовірність виникнення. І, звісно, потрібно розробити стратегії відповідного реагування на ризики певного рангу. У таблиці А.3 надано перелік ризиків для даного проєкту. Результати оцінки цих ризиків знаходяться в таблиці А.4. В таблиці Б.5 надано шкалу для класифікації ризиків за їх рівнем впливу на проєкт і ймовірністю їхнього виникнення.

Таблиця А.3 – Ризики проєкту

№ ризику	**Назва (опис) ризику**
R1	Розбіжності у баченнях ситуацій між розробником та замовником
R2	Вихід альтернативного продукту
R3	Допущення помилок під час проєктування проєкту
R4	Виникнення невідповідностей у оточеннях розробника та замовника
R5	Вимкнення світла під час розробки

Продовження табл. А.3.

R6	Втрата результатів розробки через необачність
R7	Зміна ТЗ під час розробки проєкту
R8	Реалізація зайвого функціоналу
R9	Проблеми розподіленням часу
R10	Низька кваліфікація виконавців
R11	Збої в роботі програмного забезпечення
R12	Вибір неефективних технологій розробки

Джерело: побудовано автором

Таблиця А.4 – Результати визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
R1	Розбіжності у баченнях ситуацій між розробником та замовником	0,7	0,2	0,14
R2	Вихід альтернативного продукту	0,5	0,1	0,05
R3	Допущення помилок під час проєктування проєкту	0,7	0,2	0,14
R4	Виникнення невідповідностей у оточеннях розробника та замовника	0,3	0,1	0,03
R5	Вимкнення світла під час розробки	0,7	0,1	0,07
R6	Втрата результатів розробки через необачність	0,1	0,8	0,08
R7	Зміна ТЗ під час розробки проєкту	0,5	0,2	0,1
R8	Реалізація зайвого функціоналу	0,1	0,1	0,01
R9	Проблеми з розподіленням часу	0,7	0,2	0,14
R10	Низька кваліфікація виконавців	0,5	0,2	0,1
R11	Збої в роботі програмного забезпечення	0,3	0,2	0,06
R12	Вибір неефективних технологій розробки	0,5	0,2	0,1

Джерело: побудовано автором

Таблиця А.5 – Шкала оцінювання ризиків за ймовірністю виникнення та величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: побудовано автором

Для зниження негативного впливу ризиків на проєкт потрібно розробити план щодо реагування на них. До цього процесу входять оцінка наслідків впливу на проєкт і розробка заходів для відповіді на них. Виконується цей аналіз за показниками, що описані в таблиці А.4. Результатом планування заходів реагування на ризики є матриця ймовірності виникнення та впливу ризиків (рис. А.4). Зеленим кольором позначаються прийнятні ризики, жовтим – виправдані, а червоним, в свою чергу – недопустимі.

Ймовірність виникнення ризику	Вплив ризику				
	0,05	0,1	0,2	0,4	0,8
0,9	0,045	0,09	0,18	0,36	0,72
0,7	0,035	0,07 R5	0,14 R1, R3, R9	0,28	0,56
0,5	0,025	0,05 R2	0,1 R7, R10, R12	0,2	0,4
0,3	0,015	0,03 R4	0,06 R11	0,12	0,24
0,1	0,005	0,01 R8	0,02	0,04	0,08 R6

Рисунок А.4 – Матриця ймовірності та впливу

Джерело: побудовано автором (знімок з екрану)

У таблиці А.6 представлено класифікацію ризиків проєкту за рівнем, що відповідає отриманому значенню індексу. Ризики та стратегії реагування на них представлено в таблиці А.7.

Таблиця А.6 – Шкала оцінювання ризику за рівнем

№	**Назва**	**Межі**	**Ризики, які входять (номера)**
1	Прийнятні	$0,005 \leq R \leq 0,05$	2, 4, 8
2	Виправдані	$0,05 < R \leq 0,14$	1, 3, 5, 6, 7, 9, 10, 11,12
3	Недопустимі	$0,14 < R \leq 0,72$	

Джерело: побудовано автором

Таблиця А.7 – Ризики та стратегії управління

№	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
R1	Новий	Розбіжності у баченнях ситуацій між розробником та замовником	Висока	Середній	0,14	Зменшення	Визначити можливі першопричини виникнення розбіжностей. Дотримуватись ділового етикету у спілкуванні	Створити здорову атмосферу у міжособистісних стосунках
R2	Новий	Вихід альтернативного продукту	Середня	Низький	0,05	Прийняття	Провести попереднє дослідження наявних продуктів. Створювати оригінальний та актуальний для замовника продукт	Додати унікальні функції
R3	Новий	Допущення помилок під час проєктування проєкту	Висока	Середній	0,14	Зменшення	Контроль всіх вхідних даних. Залучення контролюючої особи.	Проведення постійного контролю та перевірки етапів проєктування.
R4	Новий	Виникнення невідповідностей у оточеннях розробника та замовника	Низька	Середній	0,03	Зменшення	Враховувати вимоги до програмного оточення Модифікувати та уніфікувати продукт	Розробити шляхи швидкого вирішення та проконсультуватись у спеіаліста

Продовження табл. А.7.

№	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
R5	Новий	Вимкнення світла під час розробки	Висока	Низький	0,07	Зменшення	Мати резервний засіб живлення або техніку з акумуляторними батареями.	Моніторити повідомлення про планові відключення, погіршення погоди та результати обстрілів
R6	Новий	Втрата результатів розробки через необачність	Низька	Високий	0,08	Зменшення	Використовувати ПЗ, з автоматичним збереженням. Використовувати резервне копіювання.	Використовувати резервні сховища та шляхи відновлення даних
R7	Новий	Зміна ТЗ під час розробки проєкту	Середня	Середній	0,1	Зменшення	Своєчасно сформувати вимоги до проєкту. Обговорити з замовником всі технічні питання і можливості зміни.	Переоцінка та перепланування проєкту після зміни.
R8	Новий	Реалізація зайного функціоналу	Низька	Низький	0,01	Зменшення	Попередити замовника про вірогідність додавання додаткового функціоналу.	Обговорити і можливі плюси та мінуси від додатких функцій.

Продовження таблиці А.7.

№	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
R9	Новий	Проблеми розподілення м часу	Висока	Середній	0,14	Зменшення	Провести реорганізацію робочого процесу Провести аналіз актуальності процесів	Змінити пріоритетність робіт та провести оптимізацію існуючої системи
R10	Новий	Низька кваліфікація виконавців	Середня	Середній	0,1	Зменшення	Підвищити кваліфікацію Використовувати необхідні допоміжні матеріали	Переоцінка процесів та розподілу часу з врахуванням навичок
R11	Новий	Збої в роботі програмного забезпечення	Низька	Середній	0,06	Зменшення	Мати резерв програмних засобів Мати можливість залучити спеціаліста	Використовувати лише перевірене програмне забезпечення.
R12	Новий	Вибір неефективних технологій розробки	Середня	Середній	0,1	Зменшення	Проаналізувати можливі технології на початку проєктування Сформувати потреби та шляхи їх задоволення Обрати оптимальну	Отримати поради від компетентних осіб

Джерело: побудовано автором

ДОДАТОК Б

Апробація результатів дослідження

На рисунках Б.1-Б.4 представлено тези, опубліковані за результатами даного дослідження.

Матеріали конференції КІТ-2023, Харків, ХНАДУ, 22.11.2023

УДК 004

WEB-ОРІЄНТОВАНА СИСТЕМА ПІДТРИМКИ ДІЯЛЬНОСТІ ТУРИСТИЧНОГО ГОТЕЛЮ ІЗ МЕХАНІЗМОМ СИНХРОНІЗАЦІЇ ДАНИХ ІЗ ДЕКІЛЬКОХ ДЖЕРЕЛ

Глуховцов Д.О., Антипенко В.П.

Сумський державний університет, Суми

У зв'язку з тотальною цифровізацією світ постійно рухається до переведення значної кількості процесів в онлайн. Не є виключенням і бізнес, адже кожна компанія прагне бути в тренді та максимально ефективною в сучасних реаліях. Сьогодні особливо можна виділити такий його напрямок, як індустрія гостинності. Вона є однією з найперспективніших і входить до таких, які мають найбільший попит серед населення. Її невід'ємною частиною є готельно-ресторанний бізнес. Він є потужним та конкурентним гравцем на світовому ринку.

Зараз готельний бізнес є досить актуальним і таким, який швидко розвивається. Тому він також прагне постійного вдосконалення. Висвітленню цієї тематики присвячується багато робіт. Особлива увага приділяється організації процесів всередині подібних закладів. Так, зокрема, в [1] описуються основні аспекти сучасного бізнесу, а також шляхи, якими можна досягти максимальної ефективності адміністрування та обґрунтовується необхідність наявності системи, яка дозволить реалізувати управлінську діяльність.

Загальновідомим є той факт, що прибутковість підприємства або, у даному випадку, готелю є одним із головних критеріїв його рентабельності. Це питання також є часто згадуваним в багатьох сучасних публікаціях. Так у [2] описується розробка стратегії управління прибутком. Вона будується на різноманітних чинниках. Наприклад, ціні за номери та послуги, можливостях розселення та інших, які є важливими, але і такими, якими не зручно оперувати без засобів автоматизації та систематизації.

Результати дослідження, які представлені у [3], описують що для досягнення амбітних цілей з популяризації готельного бізнесу та підвищення його прибутковості

Рисунок Б.1 – Перша сторінка тез

Джерело: побудовано автором (знімок з екрану)

часто використовуються особливі механізми. Це зовнішні інтеграції із системами туристичних агенцій або іншими майданчиками, що провокує появу великого об'єму інформації. У свою чергу увесь цей потік даних має бути опрацьованим та конвертованим у позитивні результати. Рішенням такої задачі є використання автоматизованої системи. Без її застосування це є максимально складним завданням для реалізації.

Отже, на основі вищесказаного, можна зробити висновок, що сьогодні для управління готельним бізнесом критично важливо мати потужну адміністративну систему. Її використання необхідно для ефективної організації робочих процесів, оскільки такий інструмент допоможе персоналу краще виконувати їхню роботу. Застосування подібної системи однозначно зменшить навантаження на адміністратора та звузить сектор його уваги до одного об'єкта. Її функціонал покриває як і банальний контроль за бронюваннями з прямих продажів, які можуть надходити безпосередньо через сайт готелю, так і за запитами гостей з інших ресурсів, а також управління персоналом. А моніторинг й аналіз усіх цих даних, який здійснюється в одній системі, може допомогти в дослідженні ринку та власної статистики, а також у знайденні оптимальних рішень щодо організації задач бізнесу.

Отже, метою даного проєкту є розробка web-орієнтованої системи для підтримки діяльності туристичного готелю із механізмом синхронізації даних по бронюванню номерів із декількох джерел для забезпечення ефективної організації робочих процесів у рамках даного закладу.

Для досягнення представленої мети необхідно виконати наступні задачі:

- визначити актуальність та цільову аудиторію розроблюваної системи;
- виконати детальний аналіз предметної області та огляд сучасних публікацій, пов'язаних з проблематикою поставленої задачі;
- провести аналіз аналогів вже існуючих рішень на ринку;
- реалізувати структуру та функціонал зазначеної web-системи;
- визначити технології, які будуть використовуватися під час розробки даної web-системи;
- розробити та реалізувати структуру web-системи;

Рисунок Б.2 – Друга сторінка тез

Джерело: побудовано автором (знімок з екрану)

– реалізувати відповідні функціональні можливості для адміністрування готелю та оперування даними;

– провести тестування отриманої web-системи.

Розроблювана web-орієнтована система має задовольняти наступні функціональні вимоги:

– перегляд всіх кімнат в номерному фонді готелю, а також тих, які є вільними у задані гостем дати;

– можливість бронювання на потрібну дату;

– управління бронюваннями з боку адміністратора, а саме їх затвердження, редагування та можливість створювати нові;

– можливість роботи з даними про номерний фонд для адміністратора, тобто редагувати інформацію, видаляти її та додавати нову;

– перегляд статистики щодо прибутковості та популярності номерів, а також інформації про гостей у графічному вигляді (у діаграмах та таблицях);

– модуль для забезпечення контролю роботи покоївок.

Для програмної реалізації даної web-системи було обрано JavaScript бібліотеку React. Її використання значно спрощує створення інтерактивних інтерфейсів, збільшує швидкодію розроблюваних систем, робить їх простішими та масштабованішими, дозволяючи перевикористовувати власні компоненти та будувати інтуїтивно зрозумілу архітектуру. Скриптову метамову Sass, а саме її синтаксис SCSS, було обрано для спрощення написання стилів і отримання більшої легкості та свободи у написанні CSS. Також для стильового та функціонального оформлення використовуватиметься бібліотека MUI. Вона надає широкий спектр інструментів для розробки інтерфейсів і пропонує вже готові модулі для пришвидшення розробки та типізації. Робота з бекендом даної web-системи буде забезпечена завдяки використанню Node.js. Це необхідно для реалізації асинхронних процесів та взаємодії з базою даних (БД). Для розробки БД було обрано PostgreSQL. Це є передовою системою реляційних баз даних корпоративного класу з відкритим вихідним кодом. Вона підтримує декілька видів запитів, а саме SQL та JSON.

Рисунок Б.3 – Третя сторінка тез

Джерело: побудовано автором (знімок з екрану)

Загалом, застосування такої web-орієнтованої системи для підтримки діяльності туристичного готелю дозволить належним чином організувати роботу його адміністратора та зекономить час за рахунок автоматизації виконання рутинних та монотонних процесів агрегації даних. Її використання допоможе оптимізувати внутрішні процеси даного закладу за рахунок їхньої автоматизації та систематизації. Також дана розробка надасть потенційним гостям додаткове місце для бронювань без комісій від інших платформ наряду з забезпеченням зручного механізму ефективного управління роботою покоївок, а, отже, і активного контролю за чистотою номерів.

У висновку можна сказати, що представлений проєкт є справді актуальним і може значно допомогти суб'єкту готельного бізнесу оптимізувати власні робочі процеси та зменшити навантаження на персонал, надавши зручні та функціональні інструменти.

Література:

1. С. Нестеренко, Н. Бочарова, А. Ярчук, «Сучасні аспекти менеджменту й адміністрування в готельно-ресторанному бізнесі», Збірник наукових праць Таврійського державного агротехнологічного університету імені Дмитра Моторного (економічні науки), т. 43, № 1. С. 82-87, 2021.
2. M. Ara, M. Foysal, M. A. Huda and S. Bairagi, «A model to develop hotel management system to optimize revenue», Khulna university Studies, vol. 19, no. 2. pp. 167-188, 2022.
3. S. Han and C. K. Anderson, «The Effect of Private Customer-Manager Social Engagement Upon Online Booking Behavior», Cornell Hospitality Quarterly, vol. 63, no. 2. pp. 141-151, 2022.

Рисунок Б.4 – Четверта сторінка тез

Джерело: побудовано автором (знімок з екрану)

ДОДАТОК В

Лістинг програмного коду основних модулів web-орієнтованої системи

Модулі реєстрації та авторизації

LoginPage.js

```
import { Link } from "react-router-dom";
import * as React from "react";
import { Typography } from "@mui/material";
import LoginForm from "../../containers/auth/LoginForm";

const LoginPage = () => (
  <div className="auth-page-container">
    <img
      src="https://storage.cloud.google.com/hotel_master_bucket/ho-
tel/logo.svg?authuser=0"
      alt="Логотип"
    />
    <Typography variant="h3">Sign In</Typography>
    <div className="log-form-wrapper">
      <LoginForm />
      <Link to="/registry">Create new account</Link>
    </div>
  </div>
);

export default LoginPage;
```

LoginForm.js

```
import * as Yup from "yup";
import React, { useContext } from "react";
import { useFormik } from "formik";
import Button from "@mui/material/Button";
import { TextField } from "@mui/material";
import { useMutation } from "react-query";
import jwtDecode from "jwt-decode";
import { useNavigate } from "react-router-dom";
import authContext from "../../authContext";
import { loginUser } from "./api/authApi";

const validationSchema = Yup.object({
  signinEmail: Yup.string("Enter your email")
    .email("Enter correct email")
    .required("Email is required"),
  signinPassword: Yup.string("Enter your password")
    .min(6, "Password should be at least 6 symbols")
    .required("Password is required"),
});
```

```
const LoginForm = () => {
  const { setContext } = useContext(authContext);
  const navigate = useNavigate();
  const login = useMutation("login", (values) => loginUser(values));
  const formik = useFormik({
    initialValues: {
      signinEmail: "",
      signinPassword: "",
    },
    validationSchema,
    onSubmit: async (values) => {
      await login.mutate(
        { ...values },
        {
          onSuccess: ({ data }) => {
            console.log(data);
            if (data.error) {
              if (data.error.login)
                formik.setErrors({
                  signinEmail: "Such email doesn`t exist in the system",
                });
              if (data.error.password)
                formik.setErrors({ signinPassword: data.error.password });
            } else {
              setContext({ token: data, user: jwtDecode(data.accessToken) });
              localStorage.setItem("token", JSON.stringify(data));
              navigate("/");
              window.location.reload();
            }
          },
          onError: (err) => console.log(err),
        }
      );
    },
  });

  return (
    <div className="log-form">
      <form onSubmit={formik.handleSubmit} id="login-form">
        <TextField
          fullWidth
          className="input-block"
          id="signinEmail"
          name="signinEmail"
          placeholder="Email"
          value={formik.values.signinEmail}
          onChange={formik.handleChange}
          error={
            formik.touched.signinEmail && Boolean(formik.errors.signinEmail)
          }
          helperText={formik.touched.signinEmail   &&   formik.errors.signinE-
mail}
          variant="standard"
        />
        <TextField
```

```
          className="input-block"
          fullWidth
          id="signinPassword"
          name="signinPassword"
          placeholder="Password"
          type="password"
          value={formik.values.signinPassword}
          onChange={formik.handleChange}
          error={
            formik.touched.signinPassword &&
            Boolean(formik.errors.signinPassword)
          }
          helperText={
            formik.touched.signinPassword && formik.errors.signinPassword
          }
          variant="standard"
        />

        <Button
          id="signup-btn"
          color="primary"
          variant="contained"
          type="submit"
        >
          Log In
        </Button>
      </form>
    </div>
  );
};

export default LoginForm;
```

RegistryPage.js

```
import { Link } from "react-router-dom";
import { Typography } from "@mui/material";
import * as React from "react";
import RegisterForm from "../../containers/auth/RegisterForm";

const RegisterPage = () => (
  <div className="auth-page-container">
    <img
      src="https://storage.cloud.google.com/hotel_master_bucket/ho-
tel/logo.svg?authuser=0"
      alt="Логотип"
    />
    <Typography variant="h3">Sign Up</Typography>
    <div className="log-form-wrapper">
      <RegisterForm />
      <Link to="/login">I already have account</Link>
    </div>
  </div>
);

export default RegisterPage;
```

RegisterForm.js

```
import React, { useContext } from "react";
import { useFormik } from "formik";
import * as Yup from "yup";
import Button from "@mui/material/Button";
import { TextField } from "@mui/material";
import { useMutation } from "react-query";
import jwtDecode from "jwt-decode";
import { useNavigate } from "react-router-dom";
import { registerUser } from "./api/authApi";
import authContext from "../../authContext";

const validationSchema = Yup.object({
  signupEmail: Yup.string("Enter your email")
    .email("Enter correct email")
    .required("Email is required"),
  signupPassword: Yup.string("Enter your password")
    .min(6, "Password should be at least 6 symbols")
    .required("Password is required"),
  repeatPassword: Yup.string("Confirm your password")
    .min(6, "Password should be at least 6 symbols")
    .required("Password is required")
    .oneOf([Yup.ref("signupPassword"), null], "Passwords should be the same"),
});

const RegisterForm = () => {
  const { setContext } = useContext(authContext);
  const navigate = useNavigate();

  const registry = useMutation("registry", (values) => registerUser(values));
  const formik = useFormik({
    initialValues: {
      signupEmail: "",
      signupPassword: "",
      repeatPassword: "",
    },
    validationSchema,
    onSubmit: async (values) => {
      await registry.mutate(
        { ...values },
        {
          onSuccess: ({ data }) => {
            setContext({ token: data, user: jwtDecode(data.accessToken) });
            localStorage.setItem("token", JSON.stringify(data));
            navigate("/");
            window.location.reload();
          },
          onError: (err) => {
            console.log(err.response.data.error);
            if (err.response.data.error.signupEmail) {
              formik.setErrors({
                signupEmail: err.response.data.error.signupEmail[0],
              });
            }
          },
```

```
        }
      );
    },
  });

  return (
    <div className="log-form">
      <form onSubmit={formik.handleSubmit} id="register-form">
        <TextField
          fullWidth
          className="input-block"
          id="signupEmail"
          name="signupEmail"
          placeholder="Email"
          value={formik.values.signupEmail}
          onChange={formik.handleChange}
          error={
            formik.touched.signupEmail && Boolean(formik.errors.signupEmail)
          }
          helperText={formik.touched.signupEmail     &&     formik.errors.sig-
nupEmail}
          variant="standard"
        />
        <TextField
          className="input-block"
          fullWidth
          id="signupPassword"
          name="signupPassword"
          placeholder="Password"
          type="password"
          value={formik.values.signupPassword}
          onChange={formik.handleChange}
          error={
            formik.touched.signupPassword &&
            Boolean(formik.errors.signupPassword)
          }
          helperText={
            formik.touched.signupPassword && formik.errors.signupPassword
          }
          variant="standard"
        />
        <TextField
          className="input-block"
          fullWidth
          id="repeatPassword"
          name="repeatPassword"
          type="password"
          value={formik.values.repeatPassword}
          onChange={formik.handleChange}
          error={
            formik.touched.repeatPassword &&
            Boolean(formik.errors.repeatPassword)
          }
          helperText={
            formik.touched.repeatPassword && formik.errors.repeatPassword
          }
```

```
          placeholder="Confirm your password"
          variant="standard"
        />
        <Button
          id="signup-btn"
          color="primary"
          variant="contained"
          type="submit"
        >
          Sign Up
        </Button>
      </form>
    </div>
  );
};

export default RegisterForm;
```

Сторінка з виведенням усіх кімнат для користувача Гість

```
import React, { useEffect, useState } from "react";
import { Container, Typography } from "@mui/material";
import { useQuery } from "react-query";
import RoomsList from "../components/room/roomsList";
import { getAllRoomsWithFilters } from "./api/crudRooms";
import CategoryFilter from "../components/filters/categoryFilter";
import SideRoomsFilter from "../components/filters/sideRoomsFilter";
import Loader from "../components/Loader";

const RoomsPage = () => {
  const [rooms, setRooms] = useState([]);
  const [filter, setFilter] = useState({
    category: null,
    dateIn: null,
    dateOut: null,
    visitors: null,
  });

  const { isFetching: isFetchingFilteredRooms, data: filteredRoomsData } =
    useQuery(["rooms-all-guest-filtered", filter], () =>
      getAllRoomsWithFilters(filter)
    );

  const handleFilterChange = (newCategory) => {
    setFilter((prevFilter) => ({
      ...prevFilter,
      category: newCategory,
    }));
    console.log(filter);
  };

  const handleSideFilterChange = (filterData) => {
    setFilter((prevFilter) => ({
      ...prevFilter,
      dateIn: filterData.dateIn,
```

```
      dateOut: filterData.dateOut,
      visitors: filterData.visitors,
    }));
    console.log(filter);
  };

  const categories = [
    { id: "1", name: "Lux" },
    { id: "2", name: "Standart" },
    { id: 3, name: "Apartments" },
  ];

  useEffect(() => {
    if (!isFetchingFilteredRooms && filteredRoomsData) {
      setRooms(filteredRoomsData.data || []);
      console.log(filteredRoomsData.data);
    }
  }, [filteredRoomsData, isFetchingFilteredRooms]);

  return (
    <Container id="rooms-page-container">
      <Typography variant="h3">Our Rooms</Typography>
      <div id="rooms-container-wrapper">
        <div id="side-filter-container">
          <SideRoomsFilter onApplySideFilter={handleSideFilterChange} />
        </div>
        <div id="rooms-container">
          <CategoryFilter
            categories={categories}
            selectedCategory={filter.category}
            onFilterChange={handleFilterChange}
          />
          {isFetchingFilteredRooms && <Loader />}
          {rooms && rooms.length !== 0 && <RoomsList rooms={rooms} />}
        </div>
      </div>
    </Container>
  );
};

export default RoomsPage;
```

Модальне вікно для виведення інформації про кімнату

roomModal.js

```
import React, { useContext, useEffect, useState } from "react";
import {
  Box,
  Button,
  Dialog,
  DialogActions,
  DialogContent,
  DialogContentText,
  Modal,
```

```
  Typography,
} from "@mui/material";
import IconButton from "@mui/material/IconButton";
import CloseIcon from "@mui/icons-material/Close";
import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";
import { DateCalendar, LocalizationProvider } from "@mui/x-date-pickers";
import { useQuery } from "react-query";
import { useNavigate } from "react-router-dom";
import ShowSlider from "../mainPage/slider";
import FeaturesList from "./featuresList";
import { getRoomReservedDates } from "../../pages/api/crudRooms";
import CustomCalendarDay from "../formComponents/CustomCalendarDay";
import authContext from "../../authContext";

const RoomModal = ({ room }) => {
  const [open, setOpen] = useState(false);
  const [showAlert, setShowAlert] = useState(false);
  const [dates, setDates] = useState(null);
  const { context } = useContext(authContext);
  const navigate = useNavigate();
  const { isFetching, data } = useQuery(
    ["room-reserved-dates", room?.room_id],
    () => getRoomReservedDates(room.room_id),
    {
      enabled: open,
    }
  );
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);
  const handleAlertClose = () => setShowAlert(false);

  const handleLoginBtn = () => navigate("/login");
  const handleRegistryBtn = () => navigate("/registry");

  const handleBookClick = () => {
    if (context && Object.keys(context.token).length !== 0) {
      navigate(`/complete-booking/${room.room_id}`);
    } else {
      setShowAlert(true);
    }
  };

  useEffect(() => {
    if (!isFetching && data) {
      setDates(data.data || null);
    }
  }, [data, isFetching]);

  const images = room?.photo_urls.map((photo, index) => ({
    img: <img src={photo} key={`photo-key-${index}`} alt="room" />,
  }));

  return (
    <>
      <Dialog
        open={showAlert}
```

```
  onClose={handleAlertClose}
  className="authorization-error-modal"
>
  <DialogContent>
    <DialogContentText>
      To be able to book rooms, you need to log in
    </DialogContentText>
  </DialogContent>
  <DialogActions>
    <Button onClick={handleRegistryBtn}>Register</Button>
    <Button onClick={handleLoginBtn}>Log In</Button>
  </DialogActions>
</Dialog>
<Button onClick={handleOpen} size="small">
  Details
</Button>
<Modal open={open} onClose={handleClose}>
  <Box id="room-modal">
    <IconButton
      aria-label="close"
      className="close-modal-btn"
      onClick={handleClose}
      sx={{
        position: "absolute",
        right: 8,
        top: 8,
        color: (theme) => theme.palette.grey[500],
      }}
    >
      <CloseIcon />
    </IconButton>
    {room ? (
      <>
        <ShowSlider slides={images} cover={false} />
        <div className="room-modal-body">
          <Typography variant="h3" className="room-modal-name">
            {room.room_name}
          </Typography>
          <Button
            className="room-modal-book-btn"
            onClick={handleBookClick}
          >
            Book Now
          </Button>
          <div className="room-modal-description-block wrapper-block">
            <Typography variant="h4">Description</Typography>
            <Typography className="room-modal-description">
              {room.description}
            </Typography>
          </div>
          <div className="room-modal-beds-block wrapper-block">
            <Typography variant="h4">Beds</Typography>
            {room.bed_info && room.bed_info.length > 0 && (
              <div className="beds-info-container">
                {" "}
                {room.bed_info.map(
```

```
                        (bed, index) =>
                          bed && (
                            <span key={index} className="beds-info">
                              {bed.quantity} {bed.bed_type}{" "}
                              {index !== room.bed_info.length - 1 &&
                                index + 1 !== room.bed_info.length &&
                                " and "}
                            </span>
                          )
                      )}
                    </div>
                  )}
                </div>
                <div className="room-modal-amenities-block wrapper-block">
                  <Typography variant="h4">Amenities</Typography>
                  {room.amenities.length > 0 && (
                    <FeaturesList fList={room.amenities} />
                  )}
                </div>
                <div className="room-modal-description-block wrapper-block">
                  <Typography variant="h4">Availability Calendar</Typography>
                  <LocalizationProvider dateAdapter={AdapterDayjs}>
                    <DateCalendar
                      className="room-modal-calendar"
                      slots={{ day: CustomCalendarDay }}
                      slotProps={{
                        day: { dates },
                      }}
                      disabled
                    />
                  </LocalizationProvider>
                </div>
              </div>
            </>
          ) : (
            <Typography>Something went wrong</Typography>
          )}
        </Box>
      </Modal>
    </>
  );
};

export default RoomModal;
```

Сторінка з усіма бронюваннями для адміністратора

BookingsPage.js

```
import React, { useEffect, useState } from "react";
import { useMutation, useQuery } from "react-query";
import { LinearProgress, Typography } from "@mui/material";
import { DataGrid, GridActionsCellItem } from "@mui/x-data-grid";
import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";
import { LocalizationProvider } from "@mui/x-date-pickers";
```

```
import Button from "@mui/material/Button";
import { toast } from "react-toastify";
import { changeBookingStatus, getAllBookings } from "../api/crudBookings";
import bookingColumns from "../../services/admin/bookingColumns";
import BookingModal from "../../containers/admin/modals/bookingModal";
import BookingActions from "../../services/admin/actionBtns/bookingActions";
import customToolbar from "../../services/admin/actionBtns/customToolbar";

const BookingsPage = () => {
  const [bookings, setBookings] = useState([]);
  const [rowData, setRowData] = useState({ open: false, value: {} });
  const [isUpdating, setIsUpdating] = useState(true);

  const { isFetching: isFetchingBookings, data: bookingsData } = useQuery(
    ["all-bookings"],
    () => getAllBookings(),
    {
      enabled: isUpdating,
    }
  );

  useEffect(() => {
    if (!isFetchingBookings && bookingsData) {
      setBookings(bookingsData.data || []);
      setIsUpdating(false);
      console.log(bookingsData.data);
    }
  }, [bookingsData, isFetchingBookings]);

  const updateStatus = useMutation("addBooking", ({ id, stat }) =>
    changeBookingStatus(id, stat)
  );

  const commonToastOptions = {
    position: "top-center",
    autoClose: false,
    hideProgressBar: true,
    closeOnClick: true,
    rtl: false,
    pauseOnHover: true,
    theme: "colored",
  };

  const cancelBooking = async (e, id) => {
    e.stopPropagation();
    await updateStatus.mutate(
      { id, stat: "Cancelled" },
      {
        onSuccess: ({ data }) => {
          toast.success(data.msg, {
            ...commonToastOptions,
            autoClose: 3000,
          });
        },
        onError: (error) => {
          if (error.response.data.error.email)
```

```
          toast.error(
            error.response.data.error.email[0].charAt(0).toUpperCase() +
              error.response.data.error.email[0].slice(1),
            commonToastOptions
          );
      },
    }
  );
};

const checkIn = async (e, id) => {
  e.stopPropagation();
  await updateStatus.mutate(
    { id, stat: "Checked-In" },
    {
      onSuccess: ({ data }) => {
        toast.success(data.msg, {
          ...commonToastOptions,
          autoClose: 3000,
        });
      },
      onError: (error) => {
        if (error.response.data.error.email)
          toast.error(
            error.response.data.error.email[0].charAt(0).toUpperCase() +
              error.response.data.error.email[0].slice(1),
            commonToastOptions
          );
      },
    }
  );
};

const checkOut = async (e, id) => {
  e.stopPropagation();
  await updateStatus.mutate(
    { id, stat: "Checked-Out" },
    {
      onSuccess: ({ data }) => {
        toast.success(data.msg, {
          ...commonToastOptions,
          autoClose: 3000,
        });
      },
      onError: (error) => {
        if (error.response.data.error.email)
          toast.error(
            error.response.data.error.email[0].charAt(0).toUpperCase() +
              error.response.data.error.email[0].slice(1),
            commonToastOptions
          );
      },
    }
  );
  console.log("booking_id", id);
};
```

```
const handleAdd = () => {
  setRowData({ open: true, value: {} });
};

const handleClose = () => {
  setRowData({ open: false, value: {} });
};

const editButton = {
  field: "action",
  headerName: "Action",
  width: 60,
  renderCell: (params) => <BookingActions params={params} />,
  filterable: false,
  sortable: false,
};
const actionButtons = {
  field: "actions",
  type: "actions",
  width: 40,
  getActions: (params) => [
    <GridActionsCellItem
      label="Check-In"
      disabled={
        params.row.status === "Cancelled" ||
        params.row.status === "Checked-Out" ||
        params.row.status === "Checked-In"
      }
      onClick={(e) => checkIn(e, params.id)}
      showInMenu
    />,
    <GridActionsCellItem
      label="Check-Out"
      disabled={
        params.row.status === "Cancelled" ||
        params.row.status === "Checked-Out"
      }
      onClick={(e) => checkOut(e, params.id)}
      showInMenu
    />,
    <GridActionsCellItem
      label="Cancel"
      disabled={
        params.row.status === "Cancelled" ||
        params.row.status === "Checked-Out" ||
        params.row.status === "Checked-In"
      }
      onClick={(e) => cancelBooking(e, params.id)}
      showInMenu
    />,
  ],
  filterable: false,
  sortable: false,
};
```

```
  const getRowId = (row) => row.booking_id;

  return (
    <div className="admin-page">
      <div className="admin-table-header">
        <Typography variant="h3">Bookings</Typography>
        <Button onClick={() => handleAdd()} className="add-booking-btn">
          Add
        </Button>
        <BookingModal show={rowData} handleModal={handleClose} />
      </div>
      <div id="admin-bookings-table-container">
        <LocalizationProvider dateAdapter={AdapterDayjs}>
          <DataGrid
            className="bookings-table"
            rows={bookings}
            columns={[...bookingColumns, editButton, actionButtons]}
            getRowId={getRowId}
            initialState={{
              pagination: {
                paginationModel: { pageSize: 15, page: 0 },
              },
            }}
            pageSizeOptions={[10, 15, 20]}
            disableSelectionOnClick
            autoHeight
            slots={{
              loadingOverlay: LinearProgress,
              toolbar: customToolbar,
            }}
            loading={isFetchingBookings}
          />
        </LocalizationProvider>
      </div>
    </div>
  );
};

export default BookingsPage;
```

Конфігураційний файл для роботи з API бронювань

crudBookings.js

```
import { apiClient } from "../../config/axios";

const accessToken = localStorage.getItem("token")
  ? `Bearer ${JSON.parse(localStorage.getItem("token")).accessToken}`
  : null;

export const addBooking = async (data) =>
  apiClient.post("/bookings", data, {
    headers: {
      Authorization: `${accessToken}`,
    },
  });
```

```
export const addAdminBooking = async (data) =>
  apiClient.post("/bookings/admin", data, {
    headers: {
      Authorization: `${accessToken}`,
    },
  });

export const updateBooking = async (data, id) =>
  apiClient.put(`/bookings/${id}`, data, {
    headers: {
      Authorization: `${accessToken}`,
    },
  });

export const getAllBookings = async () =>
  apiClient.get("/bookings", {
    headers: {
      Authorization: `${accessToken}`,
    },
  });

export const changeBookingStatus = async (id, stat) =>
  apiClient.put(
    `/bookings/${id}/setStatus`,
    { newStatus: stat },
    {
      headers: {
        Authorization: `${accessToken}`,
      },
    }
  );
```

Кастомізований компонент для календаря

DatePickerField.js

```
import React from "react";
import { DatePicker } from "@mui/x-date-pickers/DatePicker";
import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";
import { LocalizationProvider } from "@mui/x-date-pickers";

const DatePickerField = ({ field, form, dependentField, ...otherProps }) => {
  const currentError = form.errors[field.name];
  const currentTouched = form.touched[field.name];

  const onChange = (date) => {
    form.setFieldValue(field.name, date);
    if (
      dependentField &&
      dependentField.value &&
      ((dependentField.name === "dateIn" && dependentField.value >= date) ||
        (dependentField.name === "dateOut" && dependentField.value <= date))
    ) {
      form.setFieldValue(dependentField.name, null);
    }
  };
```

```
  const onBlur = () => {
    form.setFieldTouched(field.name, true);
  };

  return (
    <LocalizationProvider dateAdapter={AdapterDayjs}>
      <DatePicker
        {...otherProps}
        value={field.value}
        onChange={onChange}
        onBlur={onBlur}
        slotProps={{
          textField: {
            error: currentTouched && Boolean(currentError),
            helperText: currentTouched && currentError,
          },
        }}
      />
    </LocalizationProvider>
  );
};

export default DatePickerField;
```

CustomCalendarDay.js

```
import dayjs from "dayjs";
import { PickersDay } from "@mui/x-date-pickers";
import React from "react";

const CustomCalendarDay = ({
  day,
  startDate = null,
  endDate = null,
  dates = null,
  ...props
}) => {
  const date = dayjs(day);
  const currentDate = dayjs();
  let matchedStyles = { margin: "0px" };

  const isDateSelected = (sdate) =>
    startDate && endDate && sdate >= startDate && sdate <= endDate;

  if (
    (startDate && startDate.isSame(date, "day")) ||
    (endDate && endDate.isSame(date, "day"))
  )
    matchedStyles = {
      backgroundColor: "rgba(63,154,114,0.79)",
      color: "rgb(255,255,255)",
      ...matchedStyles,
    };
  if (isDateSelected(date))
    matchedStyles = {
      backgroundColor: "rgba(75,210,126,0.37)",
      borderRadius: "0px",
```

```
      ...matchedStyles,
    };
  // console.log(date);

  if (dates && dates.length > 0) {
    dates.forEach((range) => {
      if (
        range.startDate &&
        range.endDate &&
        date.isBetween(range.startDate, range.endDate, null, "[)") &&
        date > currentDate
      ) {
        matchedStyles = {
          pointerEvents: "none",
          color: "rgba(0, 0, 0, 0.38)",
          backgroundColor: "rgba(210,75,75,0.37)",
          borderRadius: "0px",
          ...matchedStyles,
        };
      }
    });
  }

  return <PickersDay day={day} {...props} sx={{ ...matchedStyles }} />;
};

export default CustomCalendarDay;
```

Функція для обробки додавання нового бронювання

bookingsRouter.js

...

```
router.post(
  "/",
  validationMiddleware({
    room_id: ["occupied", "non-past"],
  }),
  asyncHandler(async (req, res) => {
    const addInfo = req.body;

    if (addInfo) {
      const bookingInfo = {
        desired_check_in_time: dayjs(addInfo.desired_check_in_time)
          .add(13, "hour")
          .format(),
        desired_check_out_time: dayjs(addInfo.desired_check_out_time)
          .add(11, "hour")
          .format(),
        total_guests: addInfo.total_guests,
        total_price: addInfo.total_price,
        status: "Created",
        notes: addInfo.notes,
        room_id: addInfo.room_id,
```

```
        guest_id: addInfo.guest_id,
      };
      const guestInfo = {
        first_name: addInfo.firstName,
        last_name: addInfo.lastName,
        email: req.auth.email,
        phone_number: addInfo.phoneNumber,
        passport_number: addInfo.passportNumber,
        passport_expiry_date: addInfo.passportExpiryDate,
        country: addInfo.country,
        birth_date: addInfo.birthdate,
        stay_status: "Booked",
      };

      try {
        const { success, bookingId } = await bookingService.addBooking(
          bookingInfo,
          guestInfo
        );

        console.log("booking:", success);
        const channel = ablyClient.channels.get("admin-messages");
        if (success) {
          const room = await bookingService.getRoomByBookingId(bookingId);
          await notificationService.addNotification({
            text: `New booking for room ${room[0].room_name} added`,
            role: 1,
            initiator: 0,
            toUser: null,
          });
          console.log(room, bookingId);
          channel.publish("added booking", {
            message: `New booking for room ${room[0].room_name} added`,
            recipient: null,
          });
          res.send({ msg: "Booking created successfully", success: "success"
});
        } else {
          res.send({
            message: "Booking wasn't created",
            success: "error",
          });
        }
      } catch (error) {
        console.log(error);
        res.send({
          msg: "Booking wasn't created",
          success: "error",
        });
      }
    }
  })
);
```

…