МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Веб-застосунок з автоматизації роботи працівників готелю

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Прикладне програмування»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи ПП-42

_____Пелішенко В.С._____
(прізвище та ініціали)

Керівник _____Ващіліна О.В._____
(прізвище та ініціали)
_____к.ф.-м.н., доцент_____
(науковий ступінь, звання)

Унікальність тексту 95%
my.plag.com.ua

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *прикладних інформаційних систем*
Протокол №14 від 23.05.2023 р.
зав. кафедри_____Плескач В.Л.

Київ – 2023
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок з автоматизації роботи працівників готелю»

Освітня програма: Прикладне програмування

Спеціальність: Комп’ютерні науки

ПІБ	Підпис
Пелішенко Владислав Святославович	

Назва роботи українською та англійською мовами:

Веб-застосунок з автоматизації роботи працівників готелю
Web application for automating the work of hotel employees

Мета бакалаврської роботи: ефективна організація роботи працівників готелю на основі веб-застосунку.
План роботи: 1. Дослідження особливостей та сучасного стану автоматизації управління діяльністю готелю, формулювання вимог до програмного продукту; 2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації веб-застосунку; 3. Проектування та програмна реалізація веб-застосунку з автоматизації роботи працівників готелю.

ПІБ, ступінь, звання наукового керівника роботи:

Ващіліна Олена Валеріївна, к.ф-м.н., доцент

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	14.10.2022	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	24.10.2022	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	31.10.2022	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	01.11.2022	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	08.11.2022	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2023	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2023	Виконано
9.	Подання роботи у першому варіанті	28.04.2023	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2023	Виконано

11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**22.05.2023**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	26.05.2023	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	12.06.2023	Виконано
14.	Захист кваліфікаційної роботи бакалавра	28.06.2023	Виконано

Здобувач вищої освіти _____ (підпис)

Керівник _____ (підпис)

ВІДОМІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація(іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	11
2	20
3	23
Висновки	1
Перелік використаних джерел	2
Додатки	4

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розроб н.	Пелішенко В.С.			Відомість дипломної роботи	Лист	Листів
Керівн.	Ващіліна О.В.					66
Н/конт р.	Кравченко К.В.					
Зав.каф	Плескач В.Л.					

.					

АНОТАЦІЯ

Дипломна робота: 66 с., 21 рис., 2 табл., 31 джерело.

Ця дипломна робота присвячена проектуванню та розробленню веб-застосунку з автоматизації роботи працівників готелю.

Метою дипломної роботи є підвищення ефективності роботи працівників готелю на основі веб-застосунку.

Для досягнення поставленої мети треба вирішити такі **завдання:**

- дослідити особливості та сучасний стан автоматизації управління діяльністю готелю, формулювання вимог до програмного продукту;
- провести аналіз архітектурних рішень і вибір програмних засобів для реалізації веб-застосунку;
- спроектувати та програмно реалізувати веб-застосунок з автоматизації роботи працівників готелю;

Об’єкт дослідження

Процеси автоматизації управління діяльністю готелю з використанням інформаційних технологій.

Предмет дослідження

Програмно-технічні, організаційні принципи та підходи щодо побудови веб-застосунку з автоматизації роботи працівників готелю.

Методи дослідження

Аналогії залучені під час проектування, розробки та побудови веб-застосунку з автоматизації роботи працівників готелю.

Ключові слова: програмна система, веб-застосунок, автоматизація процесів, діяльність готелю

ABSTRACT

Thesis: 66 pages, 21 figures, 2 tables, 31 sources.

This thesis is devoted to the design and development of the web application for automation of the work of hotel employees.

The purpose of this thesis is to increase the efficiency of hotel employees based on a web application.

To achieve this goal you need to solve the following **tasks**:

- Research of features and modern state of automation of hotel activities management, formulation of requirements for the software product;
- Analysis of architectural solutions and the selection of software tools for the implementation of web application;
- Design and software implementation of a web application for automation of the work of hotel employees.

Object of study

Processes of automation of hotel activities management using information technologies.

Subject of study

Software and technical, organizational principles and approaches to building the web application for automation of the work of hotel employees.

Research methods

Analogies involved in the design, development and construction of the web application for automation of the work of hotel employees.

Key words: software system, web application, processes of automation, hotel activities

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

АСУ – автоматизована система управління

PMS – Property Management System

CRM – Customer Relationship Management

RMS – Revenue Management System

ШІ – штучний інтелект

БД – база даних

ВСТУП

Двадцять перше століття ознаменувало початок стрімкого розвитку цифрових технологій. І наскільки високими є темпи їхнього розвитку, настільки ж швидко вони впроваджуються в усі сфери нашого життя. І сфера готельного бізнесу не є винятком. Нині важко знайти представників даної сфери, які не використовують продукти та розробки цифрових технологій у процесі управління власним готелем, тощо.

Актуальність цієї теми зумовлена швидким розвитком технологій, які надають можливість використовувати все більш просунуті системи з автоматизації роботи працівників готелю, що забезпечує їхню ще більш ефективну та економічну роботу. Також потрібно враховувати темпи росту готельної сфери, яка після пандемії COVID-19 швидко відновлюється та сталий прогнозований ріст на 8%[9] щороку ринку програмного забезпечення для управління готелями.

Метою кваліфікаційної роботи бакалавра є підвищення ефективності та автоматизація роботи працівників готелю з використанням відповідного веб-застосунку.

Завдання дослідження:

- дослідити особливості та сучасний стан автоматизації управління діяльністю готелю, формулювання вимог до програмного продукту;
- провести аналіз архітектурних рішень і вибір програмних засобів для реалізації веб-застосунку;
- спроектувати та програмно реалізувати веб-застосунок з автоматизації роботи працівників готелю.

Об’єктом дослідження кваліфікаційної роботи бакалавра є процеси автоматизації управління діяльністю готелю.

Предметом дослідження кваліфікаційної роботи бакалавра є програмно-технічні, організаційні принципи та підходи щодо побудови веб-застосунку з автоматизації роботи працівників готелю.

Методи дослідження: аналогії залучені під час проектування, розробки та побудови веб-застосунку з автоматизації роботи працівників готелю.

Практичне значення одержаних результатів полягає у розроблені зручного у використанні та ефективного веб-застосунку, який зможе стати чудовим рішенням для готельного бізнесу для автоматизації чи спрощення процесів в роботі його працівників. Розроблена програмна система має бути створена з урахуванням сучасних вимог ринку.

Структура роботи:

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1

ОСОБЛИВОСТІ ТА СУЧАСНИЙ СТАН АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ДІЯЛЬНІСТЮ ГОТЕЛЮ

1.1 Технічні аспекти автоматизації управління готелем

Сучасний світ важко представити без цифрових технологій, які в нашому столітті розвиваються величезними темпами. Вони мають великий вплив не тільки на наше повсякденне життя, але й на більшість сфер бізнесу. Готельна сфера також зазнала значного впливу, отримавши в своє розпорядження велику кількість інструментів, які автоматизують чи поліпшують різні аспекти даного бізнесу.

АСУ(автоматизована система управління) – комп'ютерна система, призначення якої автоматизувати інформаційні процеси та управління різними технічними, виробничими або бізнес-процесами[2].

Дана система складається з декількох компонентів: програмне забезпечення, бази даних, мережеві засоби зв'язку, контролери, датчики та інші. Ці компоненти дозволяють автоматизувати процеси, які раніше виконувались вручну.

За весь час було розроблені різноманітні АСУ, які відповідають за певну сферу бізнесу. Так, наприклад, в готельному бізнесі сьогодні ефективно впроваджуються системи управління майном(Property Management System), системи управління взаємовідносинами з клієнтами(Customer Relationship Management), системи управління доходами(Revenue Management System) та інші. І для подальшого вивчення питання про особливості та сучасний стан автоматизації управління діяльністю готелю та визначення вимог до розроблення програмного застосунку потрібно розібратися за що відповідає кожна з цих систем.

Property Management System – це комплекс інтегрованих підсистем, що створюють ефективне середовище взаємодії співробітників, клієнтів і ділових партнерів - туристичних агентств, корпоративних клієнтів і туроператорів. До складу даної системи входить декілька модулів:

- система бронювання: цей модуль відповідає за бронювання номерів через різні канали – веб-сайт, онлайн-бронювання, туристичні агенції та інші;
- операційний модуль: дозволяє контролювати рух гостей по готелю, приймати замовлення на послуги, видачу ключів, робити звіти про зайнятість готелю та інші;
- фінансовий модуль: дозволяє автоматизувати процеси обліку фінансових операцій в готелі, наприклад, оплата за номери, послуги, різні види податків[6];
- модуль управління запасами: дозволяє контролювати рівень запасів продуктів та інвентарю, може автоматизувати процес замовлення та доставки товарів[7];
- модуль управління персоналом: дозволяє контролювати розклад роботи працівників готелю, облік робочого часу, нарахування заробітної плати та інші аспекти;
- модуль звітності: дозволяє створювати звіти про діяльність готелю, наприклад, фінансові звіти або звіти про зайнятість готелю.

Customer Relationship Management – це визначення, що відноситься до всіх методів, стратегій, технологій та інструментів, які використовуються бізнесом для залучення, утримання та розвитку бази клієнтів[3].

Основні складові подібних систем:

- база даних клієнтів: займається збором та зберіганням даних про клієнтів, наприклад, персональної інформації, інформації про покупки чи взаємодії з компанією;
- модуль аналітики даних: відповідає за аналіз та інтерпретацію даних з бази даних клієнтів для розуміння потреб та поведінки клієнтів;
- модуль управління взаємодією з клієнтами: дозволяє планувати та відстежувати взаємодію з клієнтами через різні канали зв’язку як-от телефон чи електронна пошта;
- модуль управління продажами: відповідає за відстеження стадій продажів та контроль взаємодії з існуючими та потенційними з ціллю підвищення обсягів продажів;
- модуль управління маркетингом: займається плануванням та впровадженням маркетингових кампаній, орієнтованих на потреби клієнтів;
- модуль управління послугами після продажу: відповідає за відстеження та управління послугами, які надаються після продажу, такі як обслуговування, ремонт, тощо;
- модуль управління зворотним зв’язком: займається збиранням зворотного зв’язку від клієнтів;

- модуль управління звітністю: формує та аналізує звіти про різні сфери діяльності компанії, як-от ефективність маркетингових кампаній, якість обслуговування, обсяги продажів і інші.

Revenue Management System – це система, що використовує технологію визначення найкращої ціни за номер на основі прогнозування попиту, тобто продаж потрібного номеру гостю у потрібний момент за потрібною ціною[4].

Основні модулі, які входять до складу таких систем:

- модуль аналітичних інструментів: включає інструменти збору та аналізу даних, дозволяє зрозуміти попит на готельні послуги та зміни цін на них в різні періоди;
- модуль прогнозування попиту: займається прогнозуванням попиту на готельні послуги в майбутньому, базуючись на даних за попередні періоди;
- модуль оптимізації цін: дозволяє використовувати інструменти з розрахунку оптимальних цін на готельні послуги в залежності від конкуренції, попиту та інших факторів;
- модуль управління наявністю номерів: відповідає за контроль наявності вільних номерів та баланс їх забезпечення бронюванням;
- модуль управління додатковими послугами: дозволяє ефективно управляти додатковими послугами, такими як ресторан, спа-центр, фітнес-зал;
- модуль моніторингу результатів: надає засоби моніторингу результатів застосування системи, аналізу ефективності та внесення змін для досягнення бажаних результатів.

В залежності від розміру бізнесу використання цих систем може комбінуватись. Наприклад, для нещодавно відкритого готелю, який не входить до відомої мережі готелів можуть використовувати лише PMS для підтримання достатнього рівня ефективності його роботи. З розширенням бізнесу або збільшенням бази клієнтів можуть підключати як CRM, так і RMS.

1.2 Історія розвитку АСУ

Історія розвитку АСУ починається з початку XX століття, коли на заводах та виробництвах були використані перші промислові контролери. У 50-х роках були створені перші електронні обчислювальні машини, що сприяло появі перших програмних комплексів для автоматизації виробничих процесів.

У 60-х роках було розроблено перші системи управління виробництвом, що базувалися на збірці та аналізі даних з датчиків та інших пристроїв виробничого процесу. У 70-х роках вже було можливо автоматизувати контроль за багатьма процесами виробництва, застосовуючи новітні технології.

У 80-х роках було започатковано розвиток систем управління на основі мікропроцесорів, що забезпечило їх компактність та швидкість роботи. У 90-х роках відбулася значна експансія комп'ютерних мереж, що дозволило збільшити доступність даних та зменшити вартість систем АСУ.

Сьогодні АСУ поширено в багатьох сферах життя, таких як виробництво, транспорт, енергетика, телекомунікації та інші. Розвиток сучасних технологій, таких як штучний інтелект, хмарні технології та Інтернет речей, дозволяє створювати ще більш складні та ефективні системи АСУ, які забезпечують високу точність та швидкість обробки даних. Наприклад, виробники можуть використовувати системи АСУ для автоматизації виробничих процесів, контролю якості продукції та забезпечення безпеки на робочому місці.

У транспорті системи АСУ можуть забезпечувати автоматизоване управління транспортним потоком, підвищувати безпеку на дорозі та зменшувати час простою транспортних засобів. У сфері енергетики, системи АСУ дозволяють контролювати виробництво та розподіл електроенергії, забезпечуючи ефективне використання ресурсів та зниження витрат.

Крім того, з розвитком Інтернету речей, системи АСУ можуть збирати та обробляти дані з різних джерел, таких як датчики, засоби моніторингу, камери відеоспостереження та інші, що дозволяє створювати більш точні та надійні системи управління.

Усі ці технологічні зміни дозволили системам АСУ стати невід'ємною частиною багатьох галузей життя та вирішувати важливі завдання в автоматизації процесів управління. Розвиток технологій швидко продовжується, тому майбутнє систем АСУ обіцяє бути ще більш захоплюючим і революційним.

1.3 Особливості автоматизації різних процесів в готелі

Робота готелю складається з багатьох процесів: від бронювання кімнат до обліку інвентаря. Описані раніше системи займаються їхньою автоматизацією. Ці процеси можна поділити на 4 основні напрямки: автоматизація бронювання номерів та процесу прийому гостей, автоматизація оплати та обліку фінансових операцій, автоматизація управління запасами та контролю їх рівня, автоматизація управління персоналом та контролю робочого часу. Кожен з цих напрямів має свої складові, які допомагають в досягненні автоматизації.

Автоматизація бронювання номерів та процесу прийому гостей:

- онлайн-бронювання: відповідає за бронювання номерів через веб-сайт готелю або одну із систем Інтернет-бронювання. Клієнти самі можуть

обрати тип номера, кількість осіб, дату заїзду та виїзду та інші деталі. Автоматично відбувається перевірка наявності вільних номерів та бронювання;

- сповіщення через СМС та електронну пошту: елемент, який дозволяє комунікувати з клієнтами, сповіщаючи про зміну в бронюванні, нагадуючи про дату заїзду або просячи підтвердити бронювання;
- система керування номерами: відповідає за облік вільних та зайнятих номерів. Автоматично оновлює статус номерів;
- система обліку оплати: автоматично виставляє рахунки за проживання та інші послуги. Дозволяє зберігати інформацію про оплату та автоматично переводити кошти на рахунок готелю;
- система електронного документообігу: відповідає за автоматичну обробку та зберігання документів, пов’язаних з готелем, таких як рахунки, звіти та договори.

Автоматизація оплати та обліку фінансових операцій:

- платіжна система: відповідає за проведення оплати за проживання та додаткові послуги. Оплата може відбуватись як онлайн так і іншими способами;
- система обліку фінансових операцій: цей елемент автоматично веде облік всіх фінансових операцій, відповідальних за оплату та додаткові послуги. Дозволяє стежити за оплатами, повертати кошти та формувати фінансові звіти;
- інтеграція з бухгалтерськими системами: автоматично передає інформацію про фінансові операції до бухгалтерської системи готелю;

- система аналітики: відповідає за приведення різноманітної статистики та аналітики за фінансовими операціями.

Автоматизація управління запасами та контролю їх рівня:

- система управління запасами: відповідає за ведення обліку та контролю рівня запасів. Дана система також може надсилати сповіщення про закінчення запасів;
- система автоматичного замовлення: дозволяє готелю автоматично замовляти певні товари на основі попередніх даних про рівень цих запасів;
- система контролю якості товарів: займається автоматичним відслідковуванням термінів придатності товарів та надавати інформацію про їх стан та якість;
- система аналітики запасів: відповідає за аналіз даних про запаси та вироблення стратегії щодо їх використання та управління.

Автоматизація управління персоналом та контролю робочого часу:

- система управління персоналом: дозволяє вести облік працівників, зберігати детальну інформацію про них, зарплати та години роботи;
- система планування графіків роботи: відповідає за планування графіків роботи для працівників. Може автоматично враховувати розклад роботи, відпустки, хвороби, інші форми відсутності;
- система зарплатної відомості: дозволяє розраховувати зарплату для працівників. Може автоматично враховувати багато факторів, як наприклад, години роботи, оплату праці, додаткові виплати та інші;

- система аналітики персоналу: відповідає за збір та аналіз даних працівників, щоб надалі надавати звіти про якість обслуговування, продуктивність та інші показники.

1.4 Вплив автоматизації на бізнес-процеси готелю

Використання описаних вище систем PMS, CRM або ж RMS дозволяє автоматизувати та спростити значну частину бізнес-процесів готелю, тим самим надаючи таким закладам певну кількість переваг в порівнянні з готелями без даних систем.

Перш за все, потрібно зазначити, що впровадження автоматизованих систем управління підвищує ефективність роботи працівників та швидкість обробки даних[8]. Це відбувається за рахунок зменшення людського фактору при ручному введенні даних, використання апаратного забезпечення, яке значно швидше та точніше може обробити дані, ніж це зробить працівник вручну та зменшення часу для отримання певної інформації. В свою чергу, це надає можливість для подальшого аналізу системою зібраних даних та отримання певних рекомендацій щодо:

- змін, які можна вносити в робочі графіки працівників;
- визначення найбільш популярних видів номерів та послуг;
- визначення найефективніших рекламних кампаній;

тощо.

По-друге, АСУ покращують якість обслуговування гостей та забезпечення їхньої безпеки. Наприклад:

- онлайн-бронювання та реєстрація: для гостей це зручно, оскільки вони самі можуть обрати зручний для них час заїзду та виїзду, а для працівників готелю зекономити час на опрацюванні подібних запитів;

- система керування номерами: забезпечує ефективний контроль доступу до номерів, а також керування енергозбереженням. Наприклад, регулювати температуру в номерах при вході та виході гостей;
- системи безпеки: наявність систем контролю доступу, відеоспостереження та сигналізації запобігають несанкціонованому доступу до готелю та підвищують швидкість реагування охорони.

По-третє, відбувається підвищення ефективності управління фінансами та оптимізація витрат за рахунок:

- системи обліку фінансів, яка веде точний облік фінансових операцій;
- системи прогнозування попиту, яка дозволяє готелю планувати свою роботу, пропонуючи спеціальні акції та знижки на номери в періоди зниженого попиту;
- системи автоматизованого управління запасами, яка зменшує ризик переплати за товари, забезпечуючи наявність необхідних запасів на складах в оптимальних кількостях;
- системи автоматизованого обліку заробітної плати, яка дозволяє швидко та точно нараховувати зарплату співробітникам, зменшуючи ризик помилок та підвищуючи задоволеність працівників.

В загальному, використання АСУ надає бізнесу низку критичних переваг, які потенційно можуть перекрити витрати на, власне, утримання самої АСУ.

1.5 Сучасні тренди в автоматизації управління діяльністю готелю

З кожним роком з’являється все більше нових технологій, які змінюють різні сфери програмного забезпечення. В останній час найбільше помітно прогрес в сфері штучного інтелекту. З’явились нові просунуті мовні моделі такі як ChatGPT, спроможні надати відповідь на питання користувачів або моделі,

які здатні малювати картини відповідно до запитів користувачів. Цей тренд не оминув і АСУ. Штучний інтелект та аналіз даних є одними з найбільш перспективних напрямків в автоматизації управління діяльністю готелю. Ці технології дають можливість ефективно прогнозувати попит на номери та інші послуги, а також оптимізувати ціни в режимі реального часу.

Для цього штучний інтелект аналізує великі об’єми даних про попит на послуги готелю та інші фактори, такі як події в місті, сезонність, культурні заходи та інші. Виходячи з результатів аналізу, ми отримуємо прогноз на наступний сезон та оптимальну стратегію продажу номерів та послуг.

До того ж, система аналізу даних допомагає підлаштовувати ціни на свої послуги відповідно до попиту, що забезпечує максимальний прибуток.

Також в сьогоднішніх умовах потрібно враховувати і такий вагомий фактор, як взаємодія з клієнтами через мобільні пристрої. Оскільки станом на 2023 рік кількість смартфонів наближається до 7 мільярдів, що становить 86.3% від усього населення Землі, то його необхідно використовувати для роботи з гостями.

Наприклад, можна розробити мобільний додаток для взаємодії з готелем та його послугами. Такі програми, зазвичай, надають можливість бронювання номерів, отримання підтвердження бронювання і здійснення заселення при прибутті до готелю онлайн.

Окрім цього, гість також може через додаток приймати повідомлення від готелю, такі як, наприклад, розклад прибирання, акції або спеціальні пропозиції.

Також з «бумом» криптовалют досить популярною стала блокчейн-технологія. Блокчейн – це децентралізована база даних, яка забезпечує безпеку та надійність збереження даних, а також їх ефективну обробку та передачу.

В готельному бізнесі дана технологія використовується для забезпечення безпеки даних гостей та оптимізації фінансових операцій. Наприклад, за допомогою даної технології можна створити систему для ідентифікації гостей, в якій зберігатимуться дані про них в безпечному форматі при цьому забезпечуючи швидкий доступ до них. Або ж блокчейн можна використовувати для оптимізації оплати та взаєморозрахунків між готелем та клієнтами.

У даному розділі було розглянуто особливості та сучасний стан автоматизації управління діяльністю готелю. Автоматизація процесів у готельному бізнесі може значно полегшити та прискорити роботу персоналу, покращити обслуговування гостей та забезпечити більш ефективне управління готелем в цілому.

Серед основних систем автоматизації можна виділити системи управління готельними номерами, рестораном, бронюванням та обліку фінансів. У свою чергу, такі системи можуть бути інтегровані між собою та зовнішніми системами, що дозволяє отримати більш точну та повну інформацію про роботу готелю.

Отже, автоматизація управління діяльністю готелю є важливим елементом розвитку готельного бізнесу в сучасних умовах. Для успішної імплементації таких систем необхідно провести попередню оцінку потреб бізнесу, вибрати підходящі програмні продукти та забезпечити належну підготовку персоналу до роботи з ними.

РОЗДІЛ 2

АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ

2.1 Огляд існуючих програмних засобів для автоматизації роботи працівників готелю

Для автоматизації роботи працівників готелю існує багато програмних засобів. Вони можуть допомогти готелю управляти номерним фондом, бронюваннями, розрахунками, обліком послуг та іншими процесами. Огляд існуючих програмних засобів для автоматизації роботи працівників готелю можна провести залежно від потреб готелю та його розміру.

Для невеликих готелів можуть підійти безкоштовні програми, які пропонують базовий функціонал, наприклад, HotelDruid або Bistone Hotel Management System. Ці програми можуть допомогти управляти резерваціями, обліком оплат, розрахунками, видачею документів та іншими базовими функціями.

Для середніх та великих готелів можуть бути корисними спеціалізовані програмні комплекси, які дозволяють автоматизувати більше процесів. Наприклад, Opera Property Management System, Protel, RoomKeyPMS та інші. Ці системи зазвичай мають більший функціонал, такий як управління номерним фондом, ресурсами готелю, рестораном, баром, курортними послугами, системою безпеки та іншими.

Крім того, існує програмне забезпечення, яке спеціалізується на автоматизації певних процесів готелю, наприклад, бухгалтерській облік, управління персоналом, маркетингових кампаніях та іншому. При виборі програмного забезпечення для автоматизації роботи працівників готелю, слід враховувати ряд факторів, таких як розмір готелю, його потреби, бюджет на

придбання та обслуговування програмного забезпечення, а також його сумісність з іншими системами та обладнанням готелю.

2.2 Види програмного забезпечення

При створенні програмного забезпечення постає вибір в якій формі його створити. Адже веб-застосунок та програмний застосунок мають різні шляхи проектування та розроблення.

Веб-застосунок - це програма, яку можна використовувати через Інтернет, за допомогою веб-браузера. Веб-застосунки можуть бути доступні як локально, так і глобально. До переваг веб-застосунків відноситься те, що вони не вимагають встановлення та оновлення на локальному комп'ютері, оскільки весь код програми розміщується на сервері та доступний через Інтернет. Крім того, веб-застосунки можуть бути доступні з будь-якого пристрою з підтримкою Інтернету, тому вони можуть бути більш доступними для користувачів.

Програмний застосунок - це програма, яку потрібно встановити на локальний комп'ютер або пристрій. Програмний застосунок може бути розроблений на основі конкретних вимог та потреб користувачів та може працювати як в онлайн, так і в офлайн режимі. До переваг програмних застосунків відноситься можливість доступу до даних та функцій програми навіть без підключення до Інтернету, а також більш висока швидкість та ефективність роботи.

Недоліками веб-застосунків можуть бути:

1. Залежність від Інтернету. Веб-застосунки потребують постійного підключення до Інтернету, тому при відсутності зв'язку з мережею вони стають недоступними.

2. Обмежені можливості доступу до пристроїв. Оскільки веб-застосунки запускаються в браузері, їхні можливості доступу до функцій пристрою, таких як камера або мікрофон, можуть бути обмеженими.

3. Небезпека витоку даних. Якщо сервер, на якому розміщено веб-застосунок, не захищений від зломів або атак, то інформація, що зберігається на сервері, може бути скомпрометована.

Недоліками програмних застосунків можуть бути:

1. Потреба встановлення. Перед тим, як користувач зможе почати використовувати програмний застосунок, його потрібно встановити на локальний пристрій, що може зайняти певний час.

2. Потреба в оновленнях. Програмні застосунки потребують постійних оновлень для поліпшення їхньої роботи та виправлення помилок. Це може вимагати витрат часу та ресурсів на регулярне оновлення програмного забезпечення.

3. Обмежена доступність з різних пристроїв. Користувач може мати проблеми з доступом до програмного застосунку з різних пристроїв, оскільки його потрібно встановлювати на кожен з них окремо.

Виходячи з переваг та недоліків обох видів застосунків, можна зробити висновок, що веб-застосунок краще підходить для автоматизації роботи працівників готелю. Оскільки веб-застосунок використовує лише браузер як платформу для власної роботи та інтернет, то такий застосунок можна легко запустити не тільки на локальних комп’ютерах готелю, але й на мобільних пристроях працівників. Також в працівників не виникне проблеми з постійними оновленнями, а в розробників зникне необхідність розробляти та тестувати застосунок на різних платформах як у випадку з програмним застосунком.

2.3 Архітектурні рішення для веб-застосунку

Виходячи з назви бакалаврської роботи випливає, що програмне забезпечення, яке має автоматизувати роботу працівників готелю, має бути створене як веб-застосунок. І для кращого розуміння як має виглядати веб-застосунок потрібно надати вичерпне визначення. Веб-застосунок – це програмне забезпечення, яке можна відкрити та використовувати в браузері. Для розробки користувацького інтерфейсу зазвичай використовують такі технології як HTML та CSS. Проте чим тоді веб-застосунок відрізняється від звичайного веб-сайту? Ключовою відмінністю цих понять є призначення. Веб-застосунок створюють для роботи з даними і взаємодією з користувачем. Звичайний ж веб-сайт розробляють для відображення на ньому певної інформації або статей, які ніяким чином не впливають на користувача. Детальніше про різницю між веб-застосунком та веб-сайтом можна побачити в таблиці нижче.

Таблиця 2.1 - Різниця між веб-сайтом та веб-застосунком

Параметр	Веб-сайт	Веб-застосунок
Основне призначення	Можна лише переглядати інформацію	Головна ціль – взаємодія з користувачем
Взаємодія з користувачем	Можна лише читати	Можна напряму працювати з даними
Аутентифікація	Не обов’язкова	Бажано мати
Завдання та складність	Досить просто	Залежить від завдання
Параметр	Веб-сайт	Веб-застосунок
Зміна проекту	Досить просто	Необхідні багаторівневі перевірки

Для кращого розуміння як саме працює веб-застосунок, його роботу можна умовно поділити на такі етапи:

- Клієнт запускає веб-браузер і вводить URL-адресу веб-застосунку;
- Браузер відправляє запит на сервер, який містить URL-адресу веб-застосунку;
- Сервер отримує запит і відправляє веб-сторінку браузеру клієнта;
- Браузер отримує веб-сторінку і відображає її на екрані;
- Користувач взаємодіє з веб-застосунком, вводячи дані, натисканням кнопок, переходячи між сторінками і т.д;
- Браузер відправляє дані, які вводить користувач, на сервер веб-застосунку;
- Сервер обробляє дані, що отримав, і відправляє результати клієнту, що відображається в браузері.

Цей процес може повторюватися кілька разів, поки користувач не досягне своєї мети. Кожен раз, коли користувач взаємодіє з веб-застосунком, дані передаються між браузером і сервером.

Наступний крок – визначитись з архітектурою веб-застосунку. Розгляд клієнт-серверної та багатошарової архітектур є важливим етапом при проектуванні веб-застосунку.

Клієнт-серверна архітектура - це модель, яка базується на розділенні додатка на дві частини: клієнтську і серверну. Клієнтська частина додатка знаходиться на біконечних пристроях (комп'ютерах, смартфонах, планшетах) користувачів і відповідає за інтерфейс користувача та взаємодію з ним. Серверна частина знаходиться на віддаленому сервері, який забезпечує функціональність,

обробку та зберігання даних. Приклад даної архітектури можна побачити на рисунку 2.1.

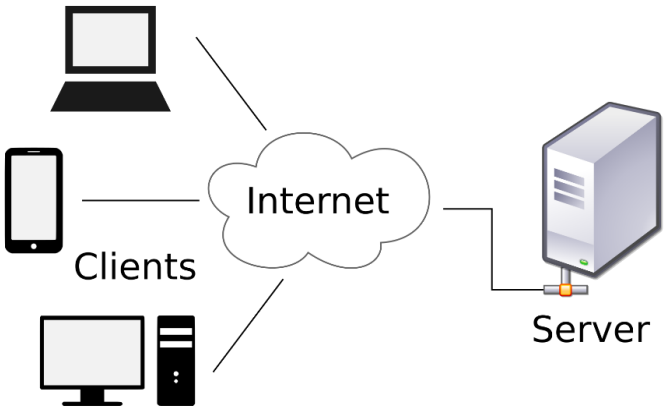

Рисунок 2.1 – Клієнт-серверна архітектура

Переваги клієнт-серверної архітектури полягають у простоті розподілення ресурсів, а також забезпеченні безпеки та конфіденційності даних. Крім того, ця архітектура забезпечує більшу масштабованість та забезпечує доступ до додатку з різних біконечних пристроїв.

Багатошарова архітектура - це архітектурна модель, яка передбачає розділення додатка на різні рівні функціональності. Кожен рівень має власний набір функцій і взаємодіє тільки з найближчими до нього рівнями. Рівні можуть включати, наприклад, презентаційний, бізнес-логіки та рівень доступу до даних. Приклад можна побачити на рисунку 2.2.

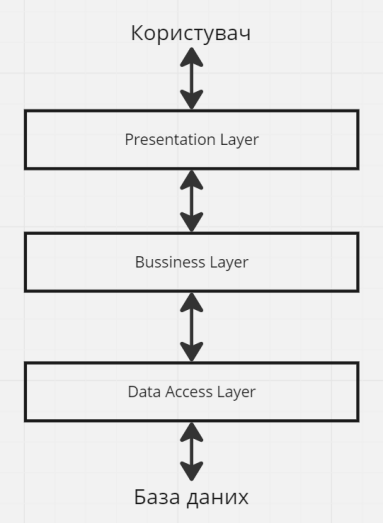

Рисунок 2.2 – Багатошарова архітектура

Переваги багатошарової архітектури полягають у зниженні залежності між рівнями, що дозволяє змінювати функціонал окремих рівнів без впливу на роботу інших рівнів. Крім того, ця архітектура забезпечує більшу гнучкість та можливість повторного використання окремих рівнів у різних додатках.

Однак, багатошарова архітектура також може бути складною в розробці та підтримці, оскільки вимагає ретельної планування та координації між різними рівнями. Крім того, ця архітектура може бути менш ефективною в термінах продуктивності, оскільки кожен рівень потребує власних ресурсів.

Оскільки клієнт-серверна архітектура на даний момент є стандартом для розроблення веб-застосунків, а веб-застосунок для автоматизації роботи працівників готелю не вимагає переваг, які надає багатошарова архітектура, то в процесі розроблення буде використана клієнт-серверна архітектура, яка окрім забезпечення доступу до застосунку з різних біконечних пристроїв також пропонує безпеку та конфіденційність даних.

Також, веб-застосунок можна поділити на декілька видів:

- SPA (Single Page Application) - це веб-застосунок, який працює без перезавантаження сторінок. Весь контент підтягується асинхронно, тому

користувачі не відчувають затримок при переході між сторінками. Основними перевагами SPA є швидкість реакції, відсутність перерв в роботі застосунку та зниження навантаження на сервер. Однак, SPA може мати проблеми з SEO-оптимізацією, оскільки весь контент на одній сторінці, тому важко оптимізувати ключові слова та метатеги;

- MPA (Multi-Page Application) - це веб-застосунок, у якому кожна сторінка завантажується окремо. Користувачі можуть переходити між сторінками, які містять окремий контент та функціонал. Основною перевагою MPA є легкість SEO-оптимізації, оскільки кожна сторінка має своє власне місце в пошуковій системі та може бути оптимізована окремо. Однак, MPA може мати проблеми з швидкістю завантаження сторінок та користувацьким досвідом, оскільки при переході між сторінками зазвичай виникає затримка;
- PWA (Progressive Web Application) - це веб-застосунок, який поєднує в собі переваги SPA та MPA. PWA може працювати як в офлайн-режимі, так і в онлайн-режимі. Він має вбудовану функцію кешування, що дозволяє швидко завантажувати контент при повторному відвідуванні. PWA має швидкість SPA та може мати переваги MPA, оскільки кожна сторінка має своє власне URL, що полегшує SEO-оптимізацію. Крім того, PWA може встановлюватись на робочий стіл або домашній екран мобільного пристрою, як мобільний додаток. Це дозволяє користувачам отримувати швидкий та зручний доступ до застосунку без необхідності завантаження з магазину додатків. Однак, PWA також може мати деякі недоліки. Наприклад, він може бути складним для розробки та підтримки, особливо якщо він потребує специфічної функціональності або інтеграції з іншими системами. Крім того, деякі функції, такі як сповіщення, можуть бути обмеженими або не доступними в деяких браузерах або платформах.

Загальне представлення переваг та недоліків представлене в таблиці 2.2.

Таблиця 2.2 - Переваги та недоліки видів веб-застосунків[18]

Параметр	SPA	MPA	PWA
Швидкість завантаження	висока	низька	висока
SEO-оптимізація	слабка	висока	середня
Offline-режим	відсутній	відсутній	присутній
Складність	середня	середня	висока

В результаті зважування переваг кожного виду, було вирішено, що для розробки веб-застосунку з автоматизації управління працівниками готелю буде використано SPA варіант, оскільки для подібної системи SEO-оптимізація не є необхідною, offline-режим не підходить так як працівники готелю завжди мають взаємодіяти з актуальними даними, а такі переваги SPA як середня важкість розробки та швидкість завантаження є ключовими для подібного застосунку.

2.4 Вибір технологій та інструментів для розробки веб-застосунку

Вибір технологій та інструментів для розробки веб-застосунку є надзвичайно важливим етапом в розробці будь-якого програмного забезпечення. На цьому етапі необхідно ретельно проаналізувати різні можливості та визначити ті, які найкраще відповідають потребам проекту. В даному підрозділі будуть розглянуті різні технології та інструменти, що можуть бути використані для розробки веб-застосунку з автоматизацією роботи працівників готелю. Будуть описані їхні переваги та недоліки, а також

розглянуті фактори, які допоможуть вибрати найбільш оптимальний варіант для даного проекту.

2.4.1 Технології для клієнтської частини

Клієнтська частина веб-застосунку зазвичай відповідає за відображення веб-сторінок, що містять інтерактивний інтерфейс, через який користувач може взаємодіяти з застосунком. Для розробки клієнтської частини веб-застосунку використовуються різні технології, зокрема HTML, CSS та TypeScript.

HTML (Hypertext Markup Language) - це стандартна мова розмітки, яка використовується для створення веб-сторінок. HTML дозволяє розміщувати текст, зображення, посилання та інші елементи на сторінці. Використання правильної HTML-розмітки допомагає покращити доступність та швидкість завантаження веб-сторінок.

CSS (Cascading Style Sheets) - це мова, яка використовується для стилізації веб-сторінок. CSS дозволяє визначати вигляд елементів на сторінці, включаючи кольори, шрифти, розміри та розташування. Використання CSS дозволяє розділити стилі та зовнішній вигляд сторінки від її змісту, що полегшує редагування та зміну дизайну веб-застосунку.

JavaScript - це мова програмування, яка використовується для розробки динамічних та інтерактивних елементів веб-сторінок. JavaScript дозволяє додавати функціональність до веб-сторінок, таку як анімація, перехід на іншу сторінку без перезавантаження, валідація форм та інші елементи, які покращують взаємодію користувачів з веб-застосунком. Проте оскільки це динамічна мова програмування, потрібно відслідковувати більше можливих помилок. Саме тому для розробки буде використовуватись TypeScript.

TypeScript - це мова програмування, яка є розширенням JavaScript, розроблена компанією Microsoft. Вона надає можливості статичного типізації, що дозволяє вказувати типи змінних, параметрів функцій і повертається

значення. TypeScript компілюється до звичайного JavaScript, що дозволяє виконувати його в будь-якому середовищі, яке підтримує JavaScript.

Крім того, існують бібліотеки та фреймворки, які допомагають полегшити розробку клієнтської частини веб-застосунку.

Наприклад одна з найпопулярніших бібліотек для розробки клієнтської частини веб-застосунків - це React. React базується на JavaScript та дозволяє розробникам створювати складні інтерфейси, які можуть бути оновлені без перезавантаження сторінки. Використання React допомагає полегшити розробку та підтримку клієнтської частини веб-застосунку, оскільки він дозволяє розбити інтерфейс на невеликі, повторно використовувані компоненти.

Інша популярна бібліотека- це Angular. Angular також базується на JavaScript, але використовує TypeScript - синтаксис JavaScript, який дозволяє додатково перевіряти код на етапі розробки. Angular дозволяє розробникам створювати більш складні веб-застосунки, які мають багато інтерактивності та динамічності. Однак, Angular може бути більш складним у використанні, порівняно з React.

Vue – це ще одна бібліотека, яка базується на JavaScript та дозволяє розробникам створювати динамічні інтерфейси. Vue добре підходить для невеликих та середніх веб-застосунків, оскільки він дозволяє швидко створювати компоненти та розширювати їх функціональність.

Оскільки вибір для написання інтерфейсів веб-застосунку полягає між даними трьома фреймворками, потрібно визначити їх переваги та недоліки:

React:

Переваги:

- широко використовується в індустрії, є багато матеріалів для вивчення;

- відмінна продуктивність, швидкість рендерингу та оптимізація;
- компонентна архітектура дозволяє легко створювати перевикористовувані компоненти;
- розширюваність та підтримка сторонніх бібліотек;
- можливість використовувати JSX (розширення JavaScript) для відображення компонентів, що спрощує роботу зі створенням UI.

Недоліки:

- висока кількість "бойлерплейту" (базового коду), що потрібен для створення додатку;
- використання JSX може бути не інтуїтивним для деяких розробників;
- часті зміни API та підхід до розробки у нових версіях, що може призвести до проблем при оновленні додатків.

Vue:

Переваги:

- легкий у вивченні та використанні, має документацію зі зрозумілими прикладами;
- компонентна архітектура дозволяє легко створювати перевикористовувані компоненти;
- зручний та гнучкий синтаксис шаблонів, що дозволяє швидко розробляти UI;
- можливість підключення до існуючих проектів;
- розширюваність та підтримка сторонніх бібліотек.

Недоліки:

- менша спільнота розробників порівняно з React або Angular;
- менша кількість сторонніх бібліотек та плагінів;
- менша кількість документації порівняно з React або Angular.

Angular:

Переваги:

- висока продуктивність та оптимізація;
- компонентна архітектура дозволяє легко створювати перевикористовувані компоненти;
- повний набір функцій, включаючи багато стандартних модулів, що дозволяє розробляти додатки відповідно до вимог індустрії;
- розширюваність та підтримка сторонніх бібліотек;
- велика спільнота розробників та велика кількість матеріалів для вивчення.

Недоліки:

- важкий у вивченні та використанні, має складну архітектуру та високий поріг входження;
- високий рівень абстракції може призвести до складнощів у розумінні принципів роботи фреймворку;
- більш обмежений шаблонний синтаксис порівняно з React або Vue.

Мій вибір – React, оскільки він, як і інші фреймворки, підтримує технологію SPA, досить продуктивний, має велику кількість сторонніх бібліотек для реалізації різноманітних функцій і при цьому він не такий складний як Angular.

Так як веб-застосунок буде розроблений за принципом SPA, то такий веб-застосунок має мати власний стан для зберігання та керування даними. Для цього необхідно обрати одну з декількох бібліотек, сумісних з бібліотекою React, які дозволяють це реалізувати.

У React є декілька популярних бібліотек для управління станом (state) додатків. Одні з найпопулярніших:

1. Redux - це бібліотека для управління станом додатків у React. Вона дозволяє зберігати стан додатку в одному об'єкті, який називається "store". Дані можуть бути змінені за допомогою функцій, які називаються "reducers". Redux також має розширення для React, яке називається "React Redux", що дозволяє легко підключати Redux до компонентів React.

2. MobX - це ще одна бібліотека для управління станом додатків у React. Вона пропонує інший підхід до управління станом, де стан представлений як "спостерігачі" (observers), які автоматично перемальовують компоненти, коли вони отримують нові дані. MobX також має розширення для React, яке дозволяє підключати MobX до компонентів React.

3. Zustand - це нова бібліотека для управління станом додатків у React. Вона пропонує простий API, який дозволяє зберігати стан додатку та змінювати його за допомогою функцій, які називаються "updaters". Zustand також пропонує можливість створювати локальний стан для компонентів, що дозволяє зробити їх більш незалежними.

4. React Context - це вбудований механізм у React для управління станом. Він дозволяє передавати дані від батьківського компонента до дочірніх компонентів безпосередньо через "контекст". Цей механізм не такий потужний,

як Redux або MobX, але досить простий і добре підходить для менших додатків або тих, де стан не має складної структури.

Найкращим вибором є Redux, оскільки його поєднання з React є найпопулярнішим і його ефективність перевірено тисячами проектів. Проте в Redux є бібліотека-доповнення Redux Toolkit, яка робить розробку з Redux більш простою та ефективною. Вона включає в себе набір корисних утиліт, які допомагають зменшити більшість більш детальних налаштувань, що зазвичай пов'язані з Redux.

Redux Toolkit має кілька переваг, таких як:

- економія часу на підготовчих роботах, оскільки більшість типових шаблонів вже вбудовано в бібліотеку;
- менше коду для написання, оскільки Redux Toolkit дозволяє скоротити кількість коду, що необхідно написати, знизивши кількість шаблонного коду;
- збільшення читабельності коду, оскільки Redux Toolkit дозволяє дещо зменшити розмір коду, а також забезпечує більш чистий та організований підхід до розробки додатків з Redux.

Оскільки React має компонентну архітектуру це також дозволяє зручно використовувати бібліотеки готових компонент з готовим дизайном та логікою роботи.

Нижче наведено деякі з популярних бібліотек готових компонент для React:

1. Material-UI - це бібліотека готових компонент, яка використовує дизайн-мову Google Material Design. Вона містить широкий спектр компонентів, таких як кнопки, форми, таблиці та інші, які можуть бути легко налаштовані та розширені за допомогою тем та стилів.

2. React Bootstrap - це бібліотека готових компонент, яка використовує дизайн-мову Twitter Bootstrap. Вона містить широкий спектр компонентів, таких як кнопки, форми, навігаційні панелі, каруселі та інші, які можуть бути легко налаштовані та розширені.

3. Ant Design - це бібліотека готових компонент, яка використовує дизайн-мову Ant Design. Вона містить широкий спектр компонентів, таких як кнопки, форми, таблиці, графіки та інші, які можуть бути легко налаштовані та розширені.

4. Semantic UI React - це бібліотека готових компонент, яка використовує дизайн-мову Semantic UI. Вона містить широкий спектр компонентів, таких як кнопки, форми, навігаційні панелі, каруселі та інші, які можуть бути легко налаштовані та розширені.

Серед наявних варіант я обрав React Bootstrap тому що вона надійна через довгий шлях розробки та тестування, легка в використанні через простий та зрозумілий API, широкий вибір готових компонент та підтримку Bootstrap.

2.4.2 Технології для серверної частини

Серверна частина веб-застосунку (backend) складається з наступних компонентів:

1. Веб-сервер: Він є основою для роботи веб-застосунку, відповідає за обробку запитів, формування відповідей та їх передачу до клієнта. Популярні веб-сервери, що використовуються для розробки веб-застосунків, це Apache, Nginx, IIS.

2. База даних: Даний компонент забезпечує зберігання даних, що використовуються в веб-застосунку. База даних може бути реляційною (наприклад, MySQL, PostgreSQL, MS SQL Server) або нереляційною (наприклад, MongoDB, Couchbase, Cassandra).

3. Бізнес-логіка: Це компонент, що відповідає за логіку додатку. Це може бути процес обробки запитів, розрахунки, перевірка прав доступу та інші операції, які пов'язані з функціоналом веб-застосунку.

4. Автентифікація та авторизація: Даний компонент відповідає за ідентифікацію користувача та його прав доступу до різних ресурсів веб-застосунку. Це може бути реалізовано за допомогою вбудованих функцій аутентифікації та авторизації, або з використанням сторонніх бібліотек.

5. API: Це компонент, який надає можливість взаємодії з веб-застосунком ззовні. API може бути реалізований з використанням різних протоколів, таких як REST або GraphQL, та забезпечувати доступ до різних функціональних можливостей веб-застосунку.

6. Додаткові бібліотеки та сервіси: Це можуть бути сторонні бібліотеки, сервіси, що забезпечують додатковий функціонал веб-застосунку. Наприклад, це можуть бути бібліотеки для роботи зі зображеннями, текстовий процесор, сервіси для розсилки електронних листів та інші. Також можуть використовуватись сервіси для забезпечення безпеки, моніторингу, логування, тестування та інші, що полегшують розробку та підтримку веб-застосунків.

Усі компоненти серверної частини взаємодіють між собою та забезпечують роботу веб-застосунку. Розробники веб-застосунків повинні бути ознайомлені з усіма компонентами та знати, як правильно їх об'єднати, щоб забезпечити ефективну та надійну роботу веб-застосунку.

Проте, щоб створити власну серверну частину для веб-застосунку, особливо у випадку коли він планується масштабним потрібно витратити багато ресурсів та часу на проектування, розробку та тестування коду. Саме тому на ринку існують сервіси, які надають можливість використовувати як сервер готові рішення.

Ось декілька з популярних сервісів:

1. Amazon Web Services (AWS): це платформа хмарних обчислень, яка надає багато сервісів, що можуть бути використані як бекенд-частина для веб-застосунків. Наприклад, Amazon S3 - це сервіс зберігання об'єктів, який може бути використаний для зберігання статичного контенту (такого як зображення, відео, аудіо тощо). Amazon EC2 - це сервіс віртуальних машин, який може бути використаний для розгортання власних серверів, додатків та баз даних.

2. Microsoft Azure: це інший сервіс хмарних обчислень, який надає багато сервісів для веб-розробки. Наприклад, Azure App Service - це сервіс, який дозволяє розгортати веб-додатки на базі .NET, Node.js, Java, Python та інших технологій. Azure SQL Database - це повністю керована реляційна база даних, яка може бути використана для зберігання даних веб-додатків.

3. Google Cloud Platform: це сервіс хмарних обчислень, який надає різноманітні сервіси, такі як Google Cloud Storage - це область зберігання файлів та об'єктів; Google Cloud Functions - це сервіс, який дозволяє розробникам виконувати код відповідно до певних подій або запитів API без необхідності налаштування власних серверів.

4. Firebase: це платформа для розробки мобільних та веб-додатків, яка надає різноманітні сервіси, такі як база даних в реальному часі, аутентифікація, зберігання та інше. Firebase може бути використаний як бекенд-частина для веб-застосунків та мобільних додатків, а також для ігор та IoT-проектів. Firebase також надає інструменти для моніторингу та аналізу використання додатків, що дозволяє розробникам покращувати їхню продуктивність та користувацький досвід.

5. Heroku: це хмарна платформа, яка дозволяє розробникам легко розгортати веб-додатки на базі різних мов програмування, включаючи Ruby, Node.js, Python та інші. Heroku також надає підтримку для різних баз даних, таких як PostgreSQL, MySQL та MongoDB.

6. DigitalOcean: це інша хмарна платформа, яка спеціалізується на віртуальних машинах та інших сервісах хмарного хостингу. DigitalOcean надає легкий спосіб розгортання веб-додатків, а також підтримку для різних баз даних та інших інфраструктурних сервісів.

Ці сервіси можуть допомогти розробникам веб-застосунків швидко та ефективно розгортати бекенд-частини своїх веб-застосунків, що дозволяє їм більше уваги приділити фронтенд-розробці та користувацькому досвіду.

Але для фінального рішення треба розглядати як переваги, так і недоліки обох підходів.

Переваги створення власної бекенд-частини веб-застосунку:

1. Повний контроль: при створенні власної бекенд-частини веб-застосунку, розробники мають повний контроль над кодом та можуть налаштувати все до найменших деталей.

2. Незалежність: створюючи власну бекенд-частину, розробники можуть уникнути залежності від сторонніх сервісів та джерел даних.

3. Масштабування: розробники можуть налаштувати свою бекенд-частину так, щоб вона легко масштабировалась у майбутньому, якщо з'являться нові вимоги.

Недоліки створення власної бекенд-частини веб-застосунку:

1. Витрати часу та ресурсів: створення власної бекенд-частини може вимагати значних витрат часу та ресурсів, зокрема, на розробку, тестування та налаштування.

2. Складність: створення бекенд-частини може бути складним процесом, особливо якщо розробники не мають достатнього досвіду у цій області.

3. Непередбачуваність: створення власної бекенд-частини може стати непередбачуваним, оскільки можуть виникати проблеми з безпекою, продуктивністю та іншими технічними питаннями, які потрібно вирішувати.

Переваги використання сервісів для створення бекенд-частини веб-застосунку:

1. Швидкість: використання готових сервісів дозволяє значно скоротити час, необхідний для розробки бекенд-частини, оскільки розробники можуть використовувати готовий код та інфраструктуру.

2. Низькі витрати: використання готових сервісів може значно зменшити витрати на розробку та підтримку бекенд-частини, оскільки сервіси зазвичай пропонують оптимізовану інфраструктуру та технічну підтримку.

3. Простота: використання готових сервісів може бути більш простим процесом, ніж створення власної бекенд-частини, оскільки розробники можуть використовувати готові інструменти та бібліотеки.

Недоліки використання сервісів для створення бекенд-частини веб-застосунку:

1. Залежність: використання сервісів може створити залежність від сторонніх постачальників, що може призвести до проблем з безпекою та стійкістю.

2. Обмеження: сервіси можуть мати обмеження в функціональності та налаштуваннях, що може ускладнити вирішення деяких технічних завдань.

3. Вартість: деякі сервіси можуть бути досить дорогими, особливо якщо використовувати їх у великому масштабі або з високим рівнем навантаження.

Беручи до уваги усі вищеперераховані порівняння, найкращим варіантом для створення бекенд-частини веб-застосунку з автоматизації роботи працівників готелю буде використання готових рішень.

Серед наявних сервісів досить популярним рішенням є Firebase. Він має необхідні для розробки веб-застосунку з автоматизації роботи працівників готелю сервіси, такі як Firestore Database та Firebase Authentication.

Firestore Database є NoSQL базою даних, що зберігає дані у вигляді документів. Документи зберігаються у колекціях та мають унікальний ідентифікатор. Кожен документ містить набір ключ-значення пар, де ключ є рядком, а значення може бути різних типів даних, таких як рядки, числа, булеві значення, масиви та об'єкти. Документи можуть містити вкладені колекції, що дозволяє створювати складні структури даних.

Однією з особливостей Firestore є швидкість та масштабованість. Firestore дозволяє виконувати запити до бази даних в режимі реального часу, що дозволяє відстежувати зміни даних в режимі реального часу. Firestore може масштабуватися горизонтально, тобто може розширюватися за допомогою додавання додаткових вузлів. Це дозволяє забезпечувати швидку та стабільну роботу при великій кількості запитів до бази даних.

Іншою особливістю Firestore є можливість синхронізувати дані між додатками, що працюють на різних пристроях. Firestore забезпечує миттєву синхронізацію даних між пристроями, що дозволяє користувачам отримувати оновлення даних в режимі реального часу.

Firebase Authentication - це сервіс аутентифікації користувачів для додатків, розроблених на платформі Firebase. Цей сервіс дозволяє додавати функціональність аутентифікації користувачів до додатків без необхідності розробляти власний сервер аутентифікації.

Firebase Authentication підтримує різні методи аутентифікації, включаючи електронну пошту та пароль, соціальні мережі, такі як Google, Facebook, Twitter та інші, а також інші методи, такі як аутентифікація за допомогою номера телефону.

Сервіс Firebase Authentication забезпечує безпеку користувачів за допомогою різноманітних методів аутентифікації, таких як багаторазова перевірка пароля, тимчасова блокування після блокування, перевірка електронної пошти та інші. Крім того, Firebase Authentication також дозволяє встановлювати обмеження на доступ користувачів до певних функцій додатку, забезпечуючи додаткову безпеку та захист.

Firebase Authentication також забезпечує зручний інтерфейс для управління користувачами, що дозволяє додавати, видаляти та редагувати профілі користувачів, а також додавати права доступу для адміністраторів та модераторів. Для розробників Firebase Authentication надає зручні інструменти для інтеграції сервісу в додатки, зокрема SDK для різних мов програмування, включаючи JavaScript, Android, iOS та інші.

В ході аналізу архітектурних рішень було проведене дослідження будови веб-застосунків, їхньої архітектури і технологій, які можна використовувати при розробці. Також було проведене порівняння та обрано технології, які будуть використані при проектуванні та розробці веб-застосунку з автоматизації роботи працівників готелю.

РОЗДІЛ 3

ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ З АВТОМАТИЗАЦІЇ РОБОТИ ПРАЦІВНИКІВ готелю

3.1 Проектування та розробка серверної частини

Для створення серверної частини будуть використані можливості сервісу Firebase, а саме його Firestore Database та Firebase Authentication.

3.1.1 Структура бази даних

Почати розробку варто з проектування бази даних Firestore. Як зображено на рисунку 3.1. база даних складається з 3 колекцій: hotels, users та cleaning.

Рисунок 3.1 - Структура бази даних

Колекція hotels відповідає за зберігання даних про готель та бронювання, де кожен документ має унікальний ID та має таку структуру:

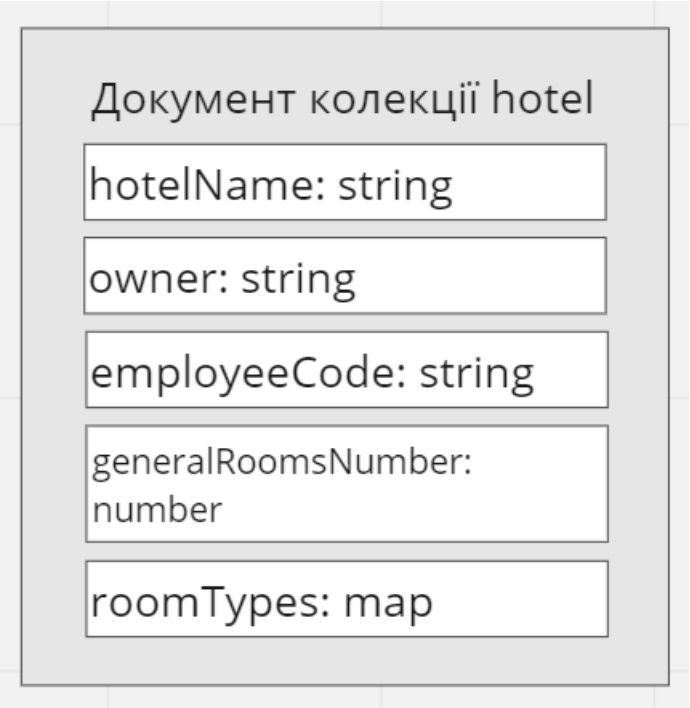

Рисунок 3.2 - Структура документу колекції hotels

В свою чергу поле roomTypes містить поля ключ-значення, де ключ – назва типу кімнати, а значення зберігає інший map тип даних, як на рисунку 3.3. Також потрібно зазначити, що тип даних map подібний до типу даних object.

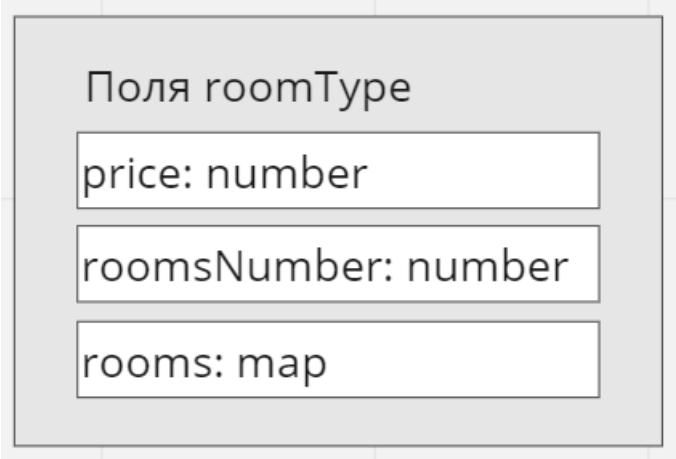

Рисунок 3.3 - Структура об’єкту roomType

Поле rooms також зберігає дані у вигляді ключ-значення, де ключ – номер кімнати, а значення – масив об’єктів з інформацією про бронювання даної кімнати, які мають структуру як на рисунку 3.4.

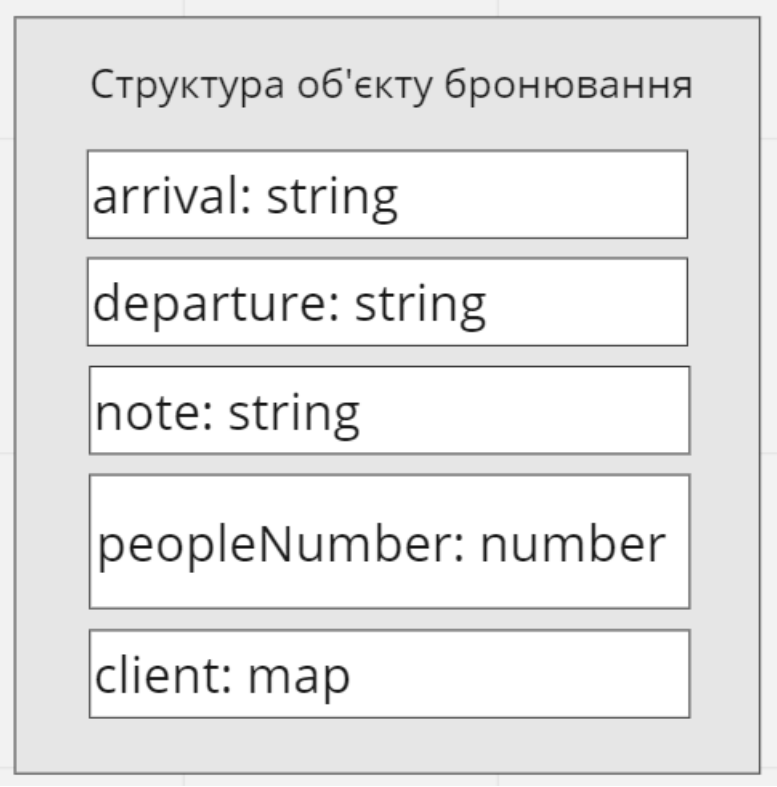

Рисунок 3.4 - Структура об’єкту бронювання

Поле client відповідає за зберігання інформації про клієнта, який оформив замовлення, має тип map та містить такі полі як рисунку 3.5.

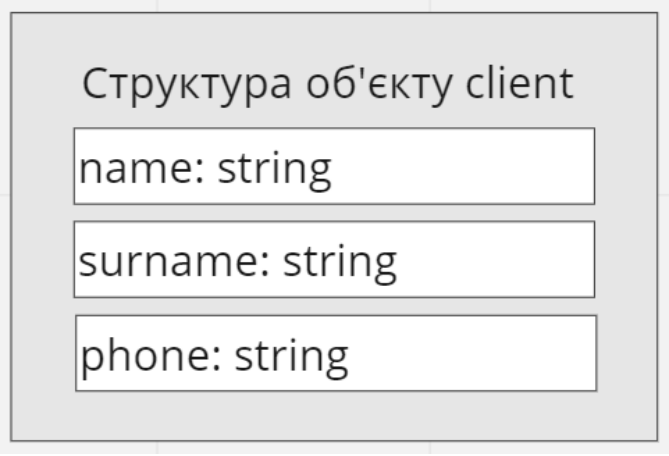

Рисунок 3.5 - Структура об’єкту client

Далі йде структура колекції users, яка відповідає за зберігання інформації про зареєстрованих користувачів(в даному випадку власників готелів та працівників) та їх ролі. Відповідно, колекція users зберігає об’єкти типу map користувачів, які мають структуру як на рисунку 3.6.

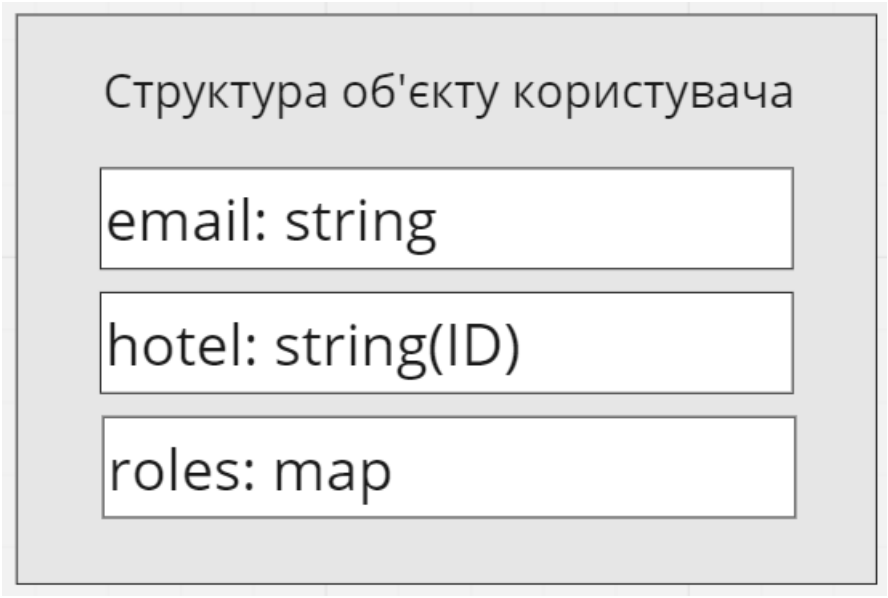

Рисунок 3.6 - Структура об’єкту користувача

В полі roles зберігається три поля: owner, cleaner, receptionist. Кожне з них має тип boolean і показує, яку роль має користувач.

Остання колекція в базі даних cleaning відповідає за зберігання об’єктів з інформацією про графік прибирань, які мають структуру як на рисунку 3.7.

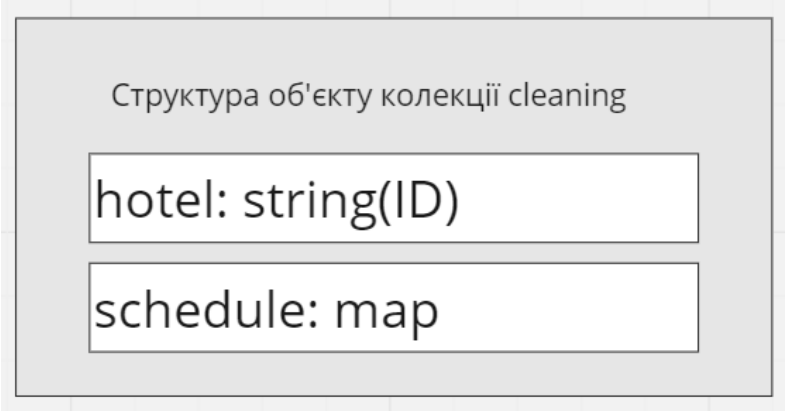

Рисунок 3.7 - Структура об’єкту колекції cleaning

Об’єкт schedule складається з пар ключ-значення, де ключ – назва дня тижня англійською мовою, а значення має тип map та складається з інших пар ключ-значення, де ключ – пошта користувача готелю, чия роль cleaner має значення true. Відповідно значення для такого ключа має структуру як на рисунку 3.8.

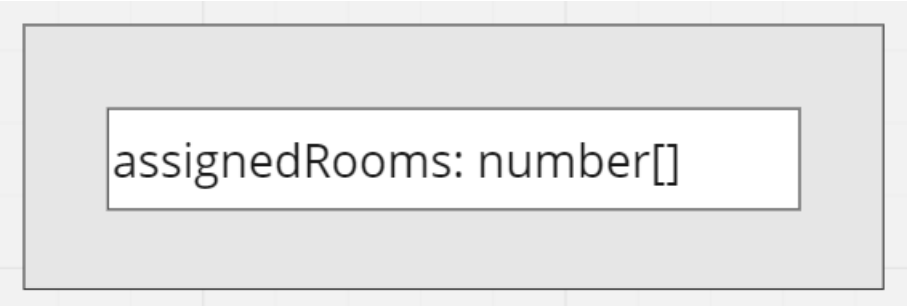

Рисунок 3.8 - Структура значення для ключа дня тижня

3.1.2 Методи авторизації

Firebase Authentication підтримує кілька методів авторизації, які дозволяють користувачам автентифікуватися за допомогою різних соціальних мереж та електронної пошти.

Основні методи авторизації в Firebase Authentication включають наступне:

1. Електронна пошта та пароль - цей метод авторизації дозволяє користувачам створювати облікові записи, ввійти у свій обліковий запис та відновлювати свій пароль.

2. Соціальні мережі - Firebase Authentication підтримує авторизацію за допомогою Facebook, Twitter, Google та інших соціальних мереж.

3. Анонімна авторизація - цей метод дозволяє користувачам використовувати додаток без реєстрації та входу.

4. Авторизація з використанням мобільного номера - цей метод дозволяє користувачам авторизуватися за допомогою свого мобільного номера та коду перевірки.

5. Авторизація з використанням ідентифікатора Firebase - цей метод дозволяє користувачам авторизуватися з використанням свого ідентифікатора Firebase.

Для веб-застосунку з автоматизації роботи працівників готелю я обрав спосіб реєстрації та авторизації за допомогою електронної пошти та паролю. Також Firebase надає можливість переглядати акаунти, зареєстровані через методи, які надає його сервіс автентифікації. Дана можливість зображена на рисунку 3.9.

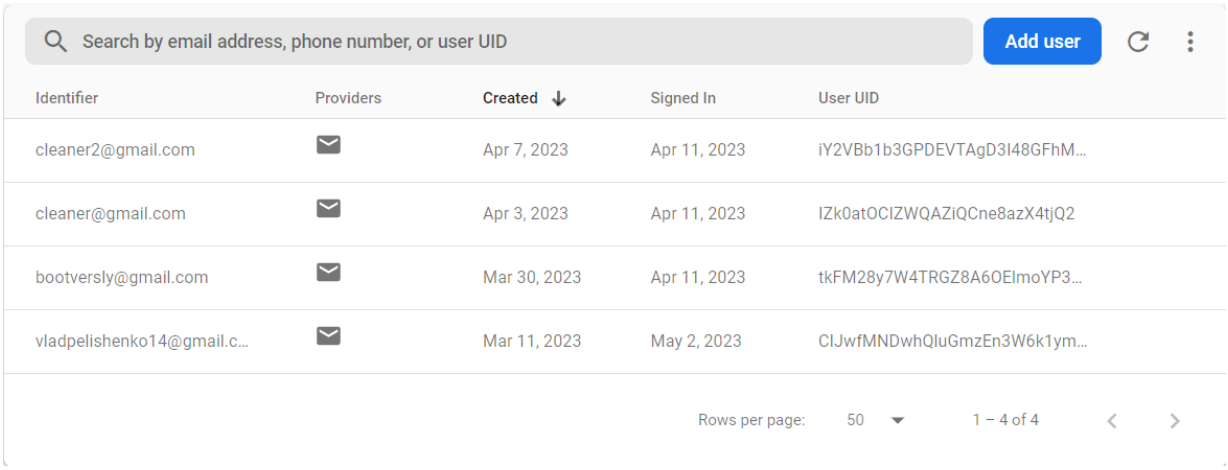

Рисунок 3.9 - Таблиця зареєстрованих акаунтів

3.2 Проектування та розробка клієнтської частини

Перед початком розробки клієнтської частини окрім етапу вибору технологій потрібно також визначити набір функцій для веб-застосунку та на його основі спроектувати структуру проекту.

3.2.1 Визначення функціоналу веб-застосунку

Перед процесом проектування розпочнемо з визначення ролей, після чого перейдемо до розподілу функцій до яких кожна з ролей буде мати доступ. Для

свого веб-застосунку з автоматизації роботи працівників готелю я визначив три ролі: власник готелю, прибиральник та менеджер з бронювання.

Загальне розподілення функціоналу по ролях таке:

- прибиральник: особа, яка має можливість зареєструватись під роллю прибиральника в певному готелі та переглядати графік прибирань;
- менеджер з бронювання: особа, яка має можливість зареєструватись під роллю менеджера з бронювання, формувати бронювання, переглядати загальний графік бронювання та переглядати графік заселень та виселень з кімнат на актуальний день;
- власник готелю: особа, яка має можливість зареєструватись як власник готелю, зареєструвати власний готель, мати ті ж функції, що й попередні ролі, також можливість редагувати графік прибирань і перегляд списку працівників готелю.

Відштовхуючись в даного розподілення функціоналу по ролях можна перейти до етапу визначення сторінок веб-застосунку та їх структури. Отже, можна виділити такі сторінки як:

- сторінка розкладу прибирань, де можна подивитись графік прибирань та його редагувати;
- сторінка створення бронювання;
- сторінка перегляду усіх бронювань;
- сторінка перегляду працівників готелю;
- сторінка профілю користувача;
- домашня сторінка, де можна побачити графік заселень та виселень на актуальний день.

На рисунку 3.10 можна побачити структуру сторінок та таких елементів, як модальні та спливаючі вікна.

Рисунок 3.10 - Структура сторінок веб-застосунку

3.2.2 Розробка функціоналу веб-застосунку

Перш ніж почати розробляти сторінки, SPA веб-застосунок потребує власне локальне сховище, яке буде зберігати інформацію, отриману у відповідях на запити за поточну сесію. Окрім зберігання це сховище також відповідальне за роботу з даними, тобто видалення та оновлення. Не менш важливим є і той факт, що використовуючи на серверній частині Firebase на клієнтській для роботи з базою даних та сервісом автентифікації використовується бібліотека «firebase», яка дозволяє нам зручно формувати та надсилати запити одразу зі сховища. Тож для початку, я за допомогою засобів redux-toolkit створив загальне сховище.

```
export const store = configureStore({reducer: rootReducer});

export type RootState = ReturnType<typeof store.getState>;

export type AppDispatch = typeof store.dispatch;
```

Проте саме по собі загальне сховище не несе ніякої функціональної цінності. Таке сховище має складатися з окремих станів. Як приклад, для зберігання інформації про працівників готелю я створив файл userReducer.ts в

якому окрім інформації про співробітників також зберігається логіка опрацювання запитів, які відповідають за роботу з колекцією «users» з бази даних та функції, які відповідають за роботу з даними стану. Ось приклад запиту, який отримує з колекції «users» усіх користувачів, які працюють в готелі:

```
export const getHotelEmployees =
createAsyncThunk('emloyees/getHotelEmployees', async(hotelId: string) => {
   const employeesRef = collection(db, 'users');
   const q = query(employeesRef, where('hotel', '==', hotelId));
   let snapshot = await getDocs(q);
   return {snapshot}
});
```

Також приклад створення стану:

```
const employeeSlice = createSlice({
   name: 'employees',
   initialState,
   reducers: {},
   extraReducers: (builder) => {}
});
```

Використовуючи цей алгоритм дій було створено також такі стани:

- hotelReducer: відповідає за інформацію про готель та бронювання та запити пов’язані з цими даними;
- cleaningReducer: відповідає за інформацію про графік прибирань та запити, пов’язані з цими даними;

- userReducer: відповідає за інформацію про авторизованого в сесії користувача та запити пов’язані з цими даними;
- reservationReducer: відповідає за інформацію про бронювання зі сторінки створення бронювання та запити пов’язані з цими даними.

Після завершення створення станів, я додав їх до загального сховища, яке надалі буде керувати з якого стану діставати дані, необхідні компонентам. Таким чином, дане сховище є «єдиним джерелом правди», яке дозволяє не дублювати дані в різних компонентах, що може спричинити конфлікти через різницю між ними.

Розпочати розробку сторінок варто з процесу створення готелю. Для цього необхідно визначити, які дані ми маємо отримати від користувача для створення в базі даних об’єкту готелю. Беручи до уваги структуру об’єкту готелю в колекції hotels користувач має надати інформацію про назву готелю, загальну кількість кімнат, типи кімнат, їх кількість, ціну та присвоїти номери кімнат до типів. Для передачі зібраних даних використовуються функції для створення документів в базі даних, отримані з бібліотеки «firebase». Приклад такої функції:

```
export const createHotel = createAsyncThunk('hotel/createHotel',
async({hotelName, owner, generalRoomsNumber, roomTypes}: HotelData) => {
    let response = await addDoc(collection(db, 'hotels'), {
    hotelName, owner, generalRoomsNumber, roomTypes, isCleaningSet: false
    })
    return {response};
});
```

Також окрім реєстрації в ролі власника готелю, потрібно додати реєстрацію як працівника. Для цього потрібно, щоб користувач надав свою пошту, придумав пароль для акаунту, ввів код готелю, щоб система змогла його автоматично додати до списку працівників потрібного закладу і обрав одну з двох ролей: прибиральник та менеджер з бронювання. Приклад форми реєстрації зображено на рисунку 3.11.

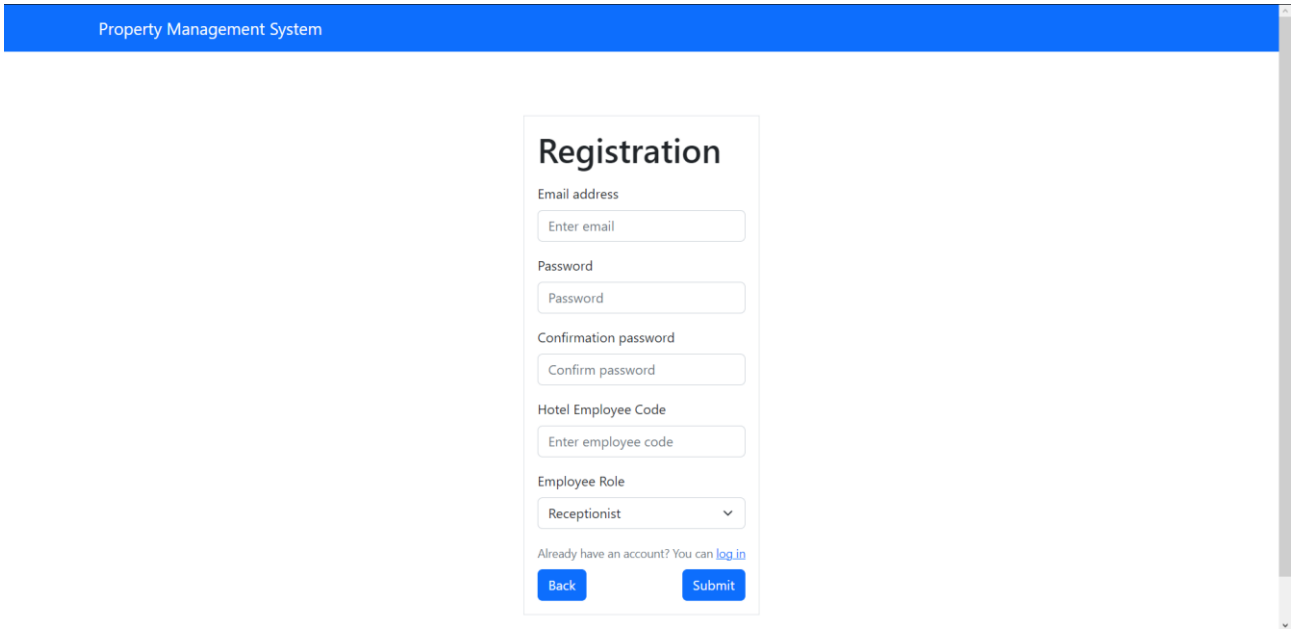

Рисунок 3.11 - Форма реєстрації працівника готелю

Наступний етап створення веб-застосунку – сторінка створення бронювання: функція, яка значно спрощує процес бронювання, зменшує людський фактор похибки та в подальшому надає можливість аналізувати результати роботи готелю, використовуючи записані таким чином дані про бронювання, отримані з бази даних. Для створення бронювання потрібно мати такі дані: дата заселення, дата виселення, кількість людей, тип кімнати, прізвище та ім’я клієнта, який оформляє бронювання та його номер телефону. Далі інформація передається в базу даних за допомогою функції updateDoc, яка дозволяє оновити поля документу, не замінюючи його або створити цей документ, якщо його немає. Приклад такого запиту:

```
{
   const docRef = doc(db, 'hotels', hotelId);
   const path = `roomTypes.${roomType}.rooms.${roomNumber}`;
   const newReservation = {
      arrival,
      departure,
      client,
      note,
      peopleNumber
   }
   let response = await updateDoc(docRef, {
      [path]: arrayUnion(newReservation)
   })
});
```

Таким чином, ми оновлюємо документ готелю, додаючи до масиву бронювань певної кімнати об’єкт нового бронювання.

Для користувача з роллю менеджера з бронювання було створено зручний інтерфейс, який дозволяє не тільки швидко створити бронювання, але й відфільтрувати кімнати за типом, як зображено на рисунку 3.12. Як видно на рисунку, менеджер також може одразу отримати інформацію про кількість вільних кімнат на період, визначений клієнтом, номер кімнати та ціну за проживання.

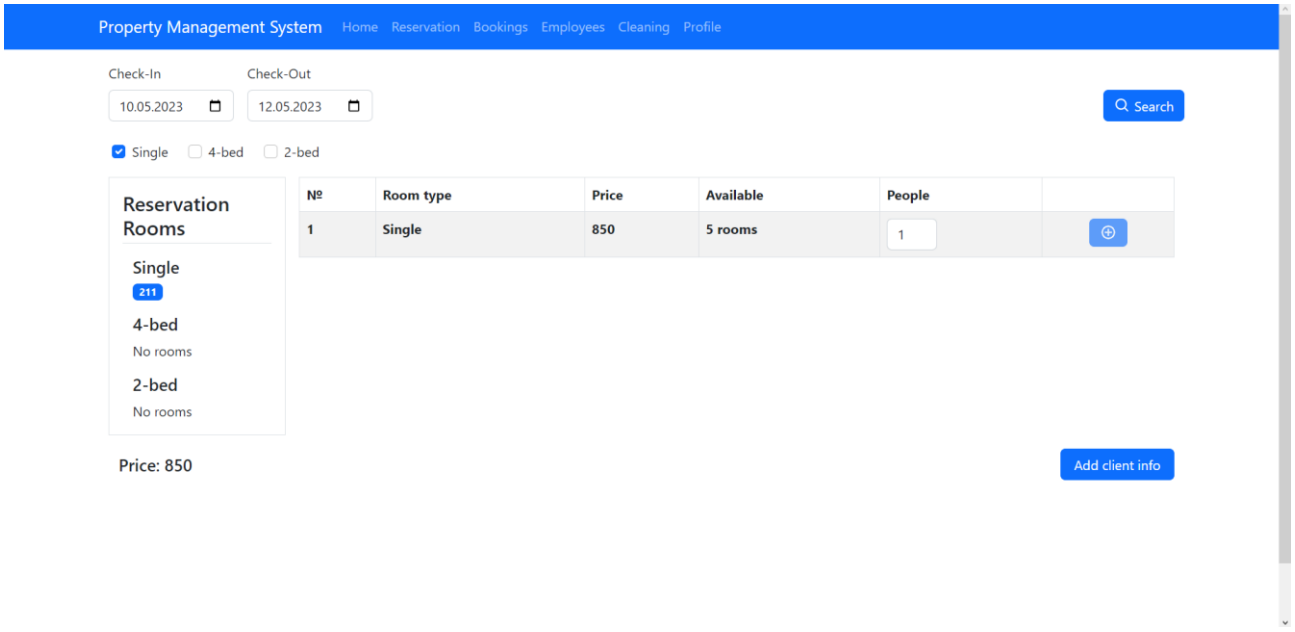

Рисунок 3.12 - Сторінка створення бронювання з відфільтрованими кімнатами

Для внесення інформації про клієнта наявне модальне вікно з якого після внесення даних можна відразу відправити бронювання до бази даних.

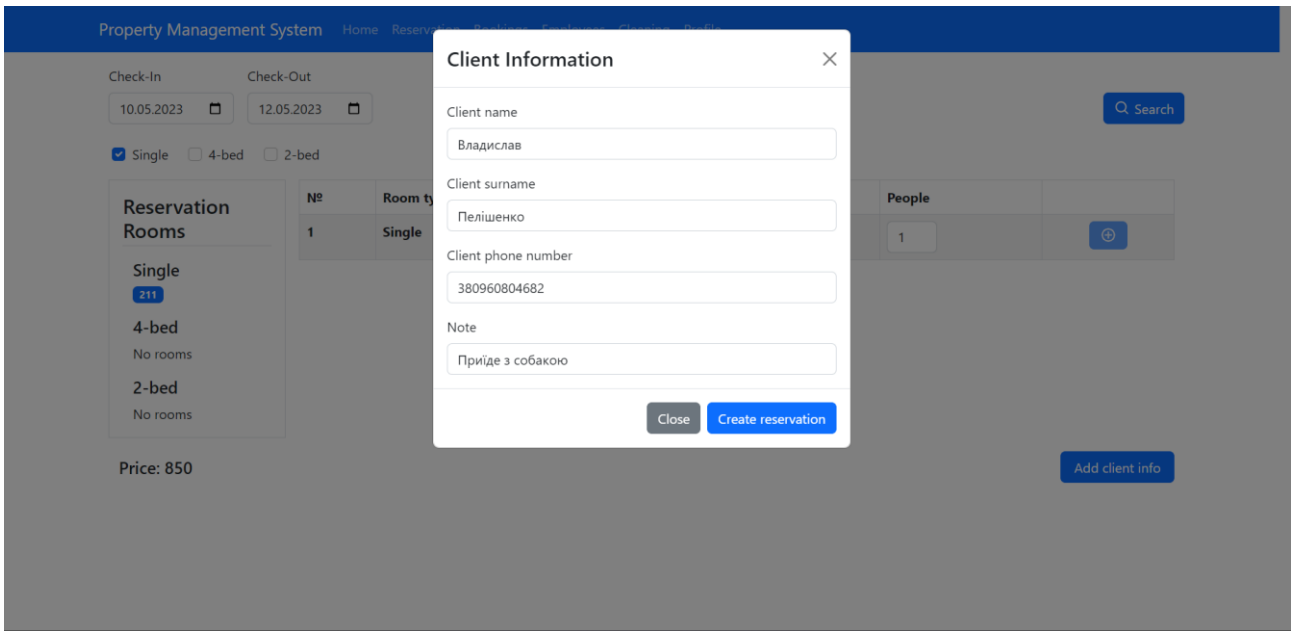

Рисунок 3.13 - Модальне вікно

Наступна сторінка – сторінка перегляду всіх бронювань. Вона використовує об’єкти бронювань з бази даних та відображає їх, змінюючи колір фону кожного бронювання відповідно до його стану. Всього є три стани: заселення –

синій, проживання – зелений та виселення – червоний. Приклад сторінки зображений на рисунку 3.14.

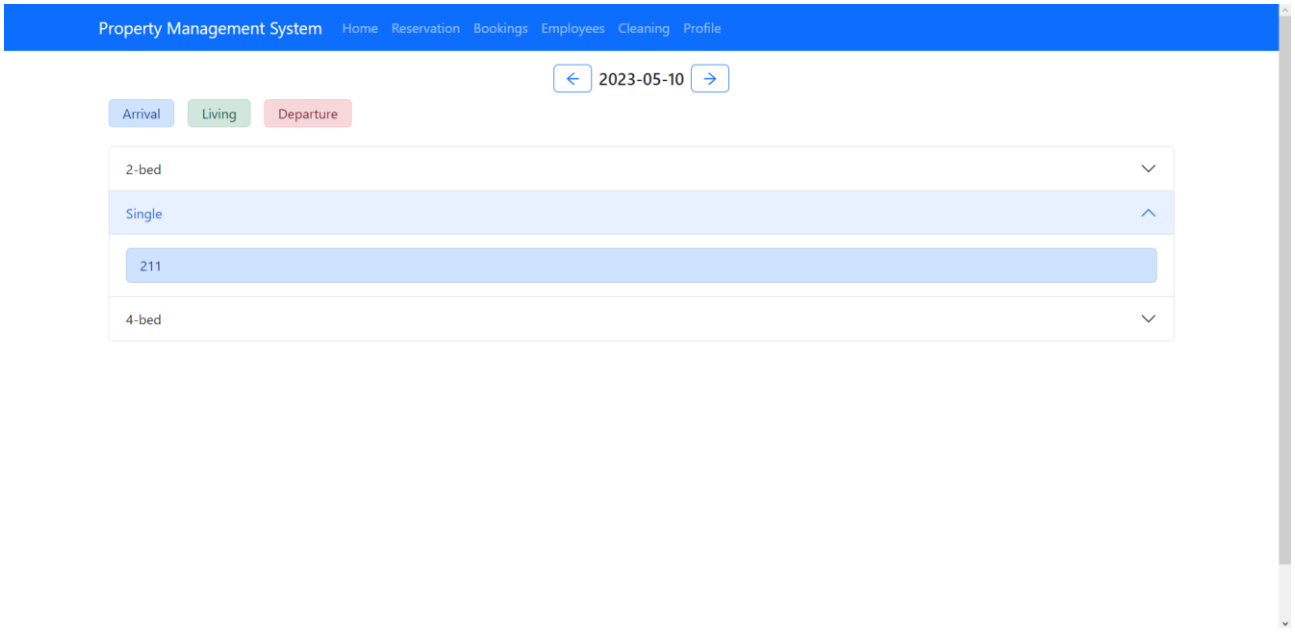

Рисунок 3.14 - Сторінка перегляду усіх бронювань

Наступна сторінка – сторінка перегляду працівників готелю. На ній відображається таблиця з даними про працівників готелю: їхні імена та прізвища, електронні пошти та ролі. Для отримання інформації з бази даних використовується функція getDocs, яка приймає першим аргументом посилання на певну колекцію, а другим проте необов’язковим умову фільтрування даних.

```
{
   const employeesRef = collection(db, 'users');
   const q = query(employeesRef, where('hotel', '==', hotelId));
   let snapshot = await getDocs(q);


   return {snapshot};
}
```

Також на даній сторінці відображається код готелю, який можуть використати працівники при реєстрації. Код дістається з інформації про готель. Приклад сторінки зображений на рисунку 3.15.

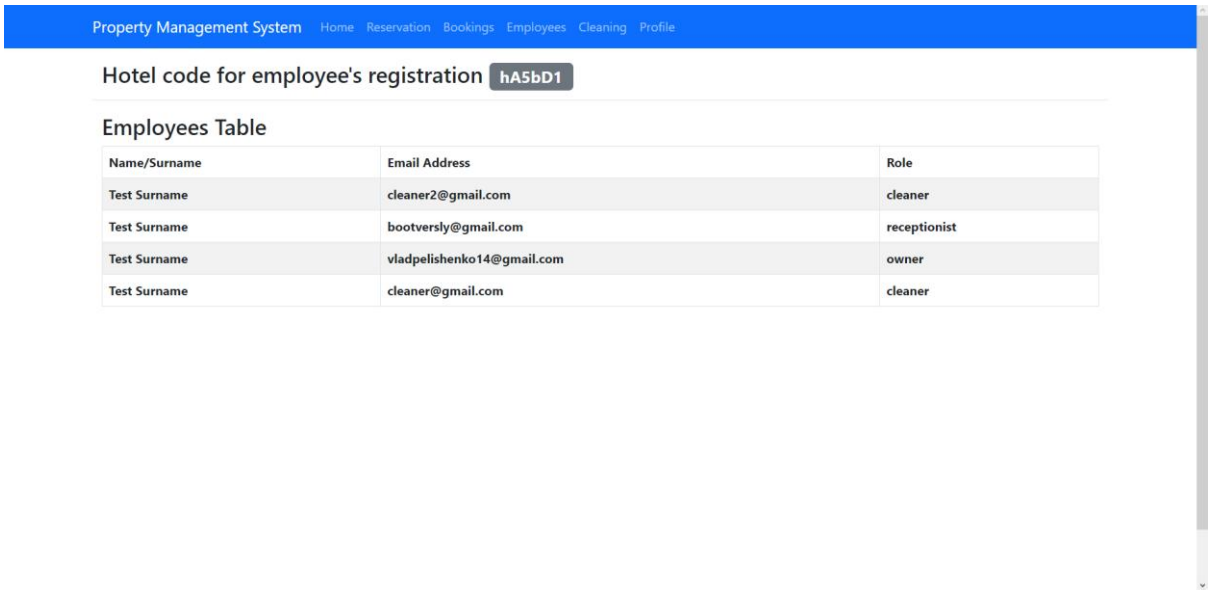

Рисунок 3.15 - Сторінка зі списком працівників

Далі йде сторінка з графіком прибирань. Коли власник створює свій готель, інформація про графік прибирань відсутня тому при відвідуванні даної сторінки в такому випадку веб-застосунок просить створити графік. Приклад такої сторінки на рисунку 3.16.

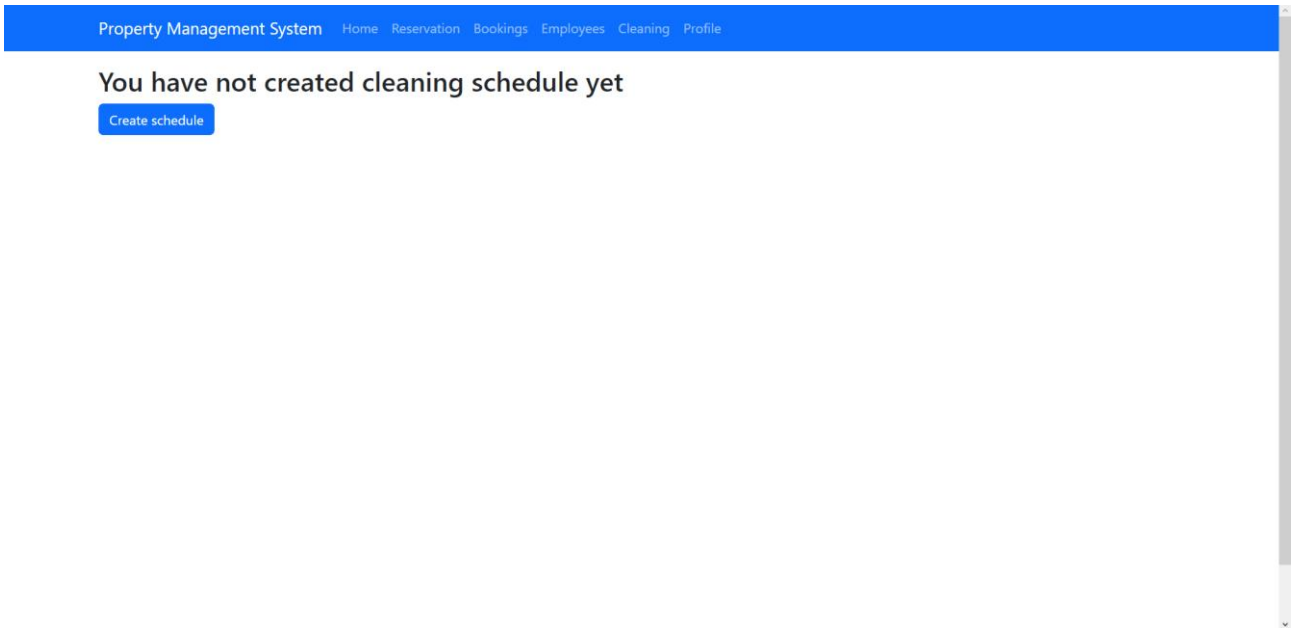

Рисунок 3.16 - Сторінка графіку прибирань без графіку

При натисканні кнопки «Створити розклад» з’являється модальне вікно в якому можна поденно задати вибраним користувачам з роллю прибиральника кімнати, які він має прибрати. При натисканні на пошту доданого прибиральника, вона змінює колір на зелений, що сигналізує, що тепер саме цьому прибиральнику будуть призначатись кімнати.

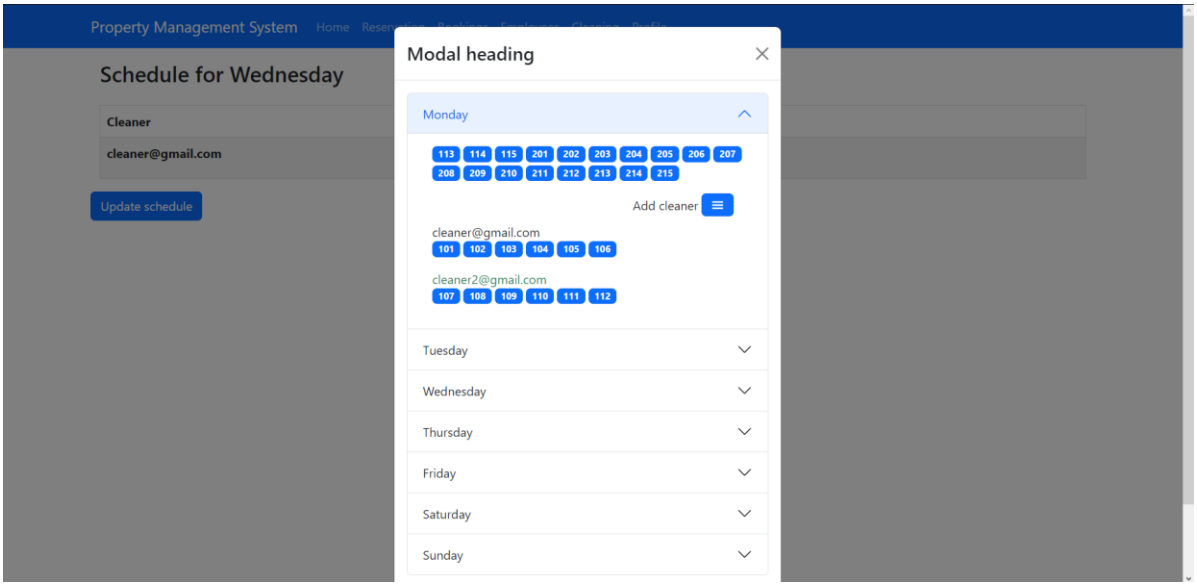

Рисунок 3.17 - Модальне вікно створення/редагування графіку прибирань

Після натискання кнопки «Створити графік» об’єкт даного графіка відправляється в колекцію «cleaning». Для цього використовується функція setDoc, яка створює або оновлює документ за ID. Приклад такого запиту:

```
{
  let docId:string|null = cleaningId;
  if (!cleaningId) {
    const newDocRef = doc(collection(db, 'cleaning'));
    docId = newDocRef.id;
  }
```

```
    if (docId) {
        const cleaningRef = doc(db, 'cleaning', docId);
        await setDoc(cleaningRef, {hotel: hotelId, schedule: newCleaningSchedule},
        {merge: true});
    }
    return docId;
}
```

Після створення об’єкту в базі даних сторінка оновлюється і відображає таблицю з відповідною інформацією на поточний день.

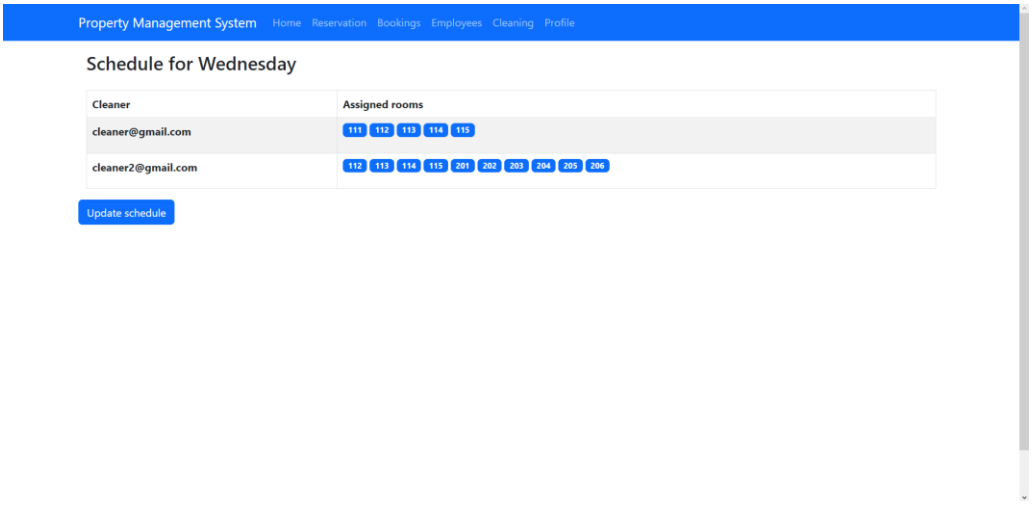

Рисунок 3.18 - Таблиця з графіком прибирання на поточний день

Домашня сторінка – на ній можна побачити інформацію про кімнати в які заселяються та з яких виселяються на поточний день гості, а також інформацію про них.

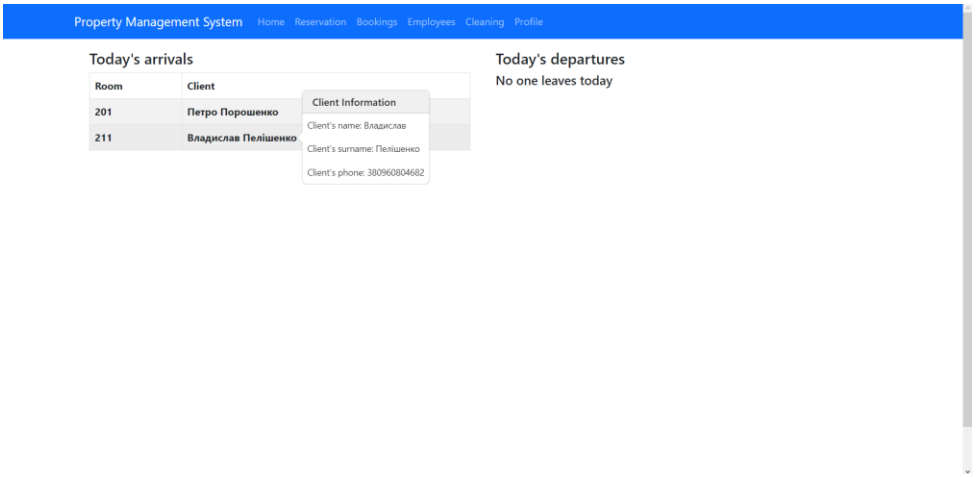

Рисунок 3.19 - Домашня сторінка

Отже, в даному розділі було розглянуто процес та етапи побудови веб-застосунку з автоматизації роботи працівників готелю. Було показано принцип роботи та функціонал веб-застосунку та приведено приклади коду. Також було продемонстровано призначення та взаємодію різних бібліотек. В результаті поставлені цілі були досягнуті.

ВИСНОВОК

В результаті виконання кваліфікаційної роботи бакалавра було розглянуто такі питання:

- дослідження особливостей та сучасного стану автоматизації управління діяльністю готелю, формулювання вимог до програмного продукту;
- проведено аналіз архітектурних рішень і вибір програмних засобів для реалізації веб-застосунку;
- спроектовано та програмно реалізовано веб-застосунок з автоматизації роботи працівників готелю.

А саме, було досліджено особливості, причини та стан розвитку автоматизованих систем управління, розглянуто їх призначення, складові таких систем, процеси, які вони автоматизують. Також було розглянуто вплив автоматизації на бізнес-процеси готелю та сучасні тенденції розвитку автоматизованих систем управління.

Також було проведено аналіз видів веб-застосунків та архітектурних рішень, визначено переваги та недоліки знайдених рішень. Крім цього було розглянуто програмно-технологічні засоби для побудови клієнтської та серверної частин веб-застосунку.

Для проектування та програмної реалізації веб-застосунку з автоматизації роботи працівників готелю було обрано набір технологій з React та Redux-toolkit для клієнтської частини та Firebase для серверної. Визначено набір необхідного функціоналу для веб-застосунку. А також розроблено інтерфейс користувача з використанням технологій HTML, CSS та бібліотеки react-bootstrap.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hospitality property management system (PMS) market size & share analysis – growth trends & forecasts(2023 – 2028). *Mordor Intelligence* : веб-сайт. URL: https://www.mordorintelligence.com/industry-reports/hospitality-property-management-software-market (дата звернення 05.02.2023)

2. Автоматизовані системи управління готелями. *Все про туризм* : веб-сайт. URL: https://tourlib.net/statti_ukr/gudzovata.htm (дата звернення 05.02.2023)

3. Що таке CRM-система та як вона працює? *Creatio* : веб-сайт. URL: https://www.creatio.com/page/uk/definition-crm (дата звернення 05.02.2023)

4. Застосування Revenue management в готельному бізнесі. *Expert Solution* : веб-сайт. URL:

https://expertsolution.com.ua/uk/primenenie-revenue-management-v-gostinichnom-biznese- (дата звернення 05.02.2023)

5. Роглєв Х.Й. Основи готельного менеджменту. *Все про туризм* : веб-сайт. URL: https://tourlib.net/books_ukr/roglev04-3.htm (дата звернення 07.02.2023)

6. Організація автоматизованого обліку фінансово-розрахункових операцій. *Букліб* : веб-сайт. URL: https://buklib.net/books/27801/ (дата звернення 07.02.2023)

7. Можливості системи управління запасами ABM Inventory. *Abmcloud* : веб-сайт. URL: https://abmcloud.com/uk/abm-soft/abm-inventory/ (дата звернення 10.02.2023)

8. Терещук Н., Транченко Л. Автоматизація бізнес-процесів як механізм підвищення ефективності діяльності готельного підприємства. 2021. №1-2

Mordor Intelligence : веб-сайт. URL: http://itsf.chdtu.edu.ua/article/view/241949 (дата звернення 10.02.2023)

9. How many smartphones are in the world? *Bank my cell* : веб-сайт. URL https://www.bankmycell.com/blog/how-many-phones-are-in-the-world (дата звернення 15.02.2023)

10. Порівнюємо React, Angular і Vue — найпопулярніші бібліотеки й фреймворки у 2022 році. *DOU* : веб-сайт.. URL: https://dou.ua/forums/topic/39933/ (дата звернення 21.02.2023)

11. 10 Best Free React UI Libraries in 2023. *We are developers* : веб-сайт. URL: https://www.wearedevelopers.com/magazine/best-free-react-ui-libraries (дата звернення 21.02.2023)

12. React Documentation. *React* : веб-сайт. URL: https://react.dev/learn (дата звернення 01.03.2023)

13. Redux Toolkit Documentation. *Redux toolkit* : веб-сайт. URL: https://redux-toolkit.js.org/tutorials/overview (дата звернення 01.03.2023)

14. Bootstrap Documentation. *Bootstrap* : веб-сайт. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення 03.03.2023)

15. React Bootstrap. *React-bootstrap*: веб-сайт. URL: https://react-bootstrap.github.io/components/alerts (дата звернення 04.03.2023)

16. React Router Documentation. *React Router* : веб-сайт. URL: https://reactrouter.com/en/main (дата звернення 05.03.2023)

17. Firebase Docs. *Firebase* : веб-сайт. URL: https://firebase.google.com/docs/firestore (дата звернення 11.03.2023)

18. Що таке веб додаток? Різниця між сайтом, веб-додатком, SPA і PWA. *Mordor Webcase* : веб-сайт. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/#f1 (дата звернення 13.03.2023)

19. Hotels | A brief history. *Hospitalitynet* : веб-сайт. URL: https://www.hospitalitynet.org/opinion/4017990.html (дата звернення 15.03.2023)

20. Property Management System Buyer's Guide. *Property management system* : веб-сайт. URL: https://propertymanagementsystem.com/guide/ (дата звернення 15.03.2023)

21. Tablets – T-Commerce: Innovative Guest-facing Application. *Hospitality Upgrade* : веб-сайт. URL: https://www.hospitalityupgrade.com/_magazine/magazine_Detail.asp/?ID=686 (дата звернення 17.03.2023)

22. Property Management System. *Tech Target* : веб-сайт. URL: https://www.techtarget.com/whatis/definition/property-management-system-PMS (дата звернення 19.03.2023)

23. Web Application Architecture: How the Web works. *Altexsoft* : веб-сайт. URL: https://www.altexsoft.com/blog/engineering/web-application-architecture-how-the-web-works/ (дата звернення 20.03.2023)

24. Client-Server Architecture for QA Engineers. *Qamadness* : веб-сайт. URL: https://www.qamadness.com/knowledge-base/client-server-architecture-for-qa-engineers/ (дата звернення 24.03.2023)

25. Архітектура застосунків на прикладі Golang. *Codeguida :* веб-сайт. URL: https://codeguida.com/post/1394 (дата звернення 13.03.2023)

26. Архітектура програмного забезпечення. *Wezom:* веб-сайт. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення 13.03.2023)

27. Що таке SPA-додатки. *Wezom:* веб-сайт. URL: https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya (дата звернення 15.03.2023)

28. Що таке SPA-додатки. *Wezom:* веб-сайт. URL: https://wezom.com.ua/ua/blog/chto-takoe-spa-prilozheniya (дата звернення 15.03.2023)

29. Багатосторінкові та односторінкові додатки, їх переваги та недоліки. *Dou:* веб-сайт. URL: https://dou.ua/forums/topic/25444/ (дата звернення 10.03.2023)

30. Learn React in 5 minutes - A React.js tutorial for beginners. *Freecodecamp* : веб-сайт. URL: https://www.freecodecamp.org/news/learn-react-js-in-5-minutes-526472d292f4/ (дата звернення 21.03.2023)

31. PWA (Progressive Web Application) - що це таке та в чому його особливості. *Highload:* веб-сайт. URL: https://highload.today/uk/pwa/ (дата звернення 17.03.2023)

ДОДАТКИ

ДОДАТОК А

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";


const firebaseConfig = {
    apiKey: "AIzaSyBpzNM8XDBL-MsEtMyf8TZd0FJdglQrrL4",
    authDomain: "property-management-syst-91e65.firebaseapp.com",
    projectId: "property-management-syst-91e65",
    storageBucket: "property-management-syst-91e65.appspot.com",
    messagingSenderId: "276767848201",
    appId: "1:276767848201:web:54e18a3f429c6aedce05d8"
};


export const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
```

Рисунок 1А – Файл конфігурації Firebase

ДОДАТОК Б

```
26  const AuthProvider = ({children}: Props) => {
27      const [currentUser, setCurrentUser] = useState<User|null>(null);
28      const [loading, setLoading] = useState(true);
29
30      function signUp(email: string, password: string) {
31          return createUserWithEmailAndPassword(auth, email, password)
32      }
33
34      function logIn(email: string, password: string) {
35          return signInWithEmailAndPassword(auth, email, password);
36      }
37
38      function logOut() {
39          return signOut(auth);
40      }
41
42      async function setName(username: string) {
43          if (currentUser) {
44              await updateProfile(currentUser, {
45                  'displayName': username
46              })
47          }
48      }
```

Рисунок 1Б – Компонент-провайдер функцій та даних автентифікованого користувача

```
50      useEffect(() => {
51          const unsubscribe = auth.onAuthStateChanged((user) => {
52              setCurrentUser(user);
53              setLoading(false);
54          })
55
56          return unsubscribe;
57      }, [])
58
59      const value: Context = {
60          currentUser,
61          signUp,
62          logIn,
63          logOut,
64          setName
65      };
66
67      return <AuthContext.Provider value={value}>
68          {!loading && children}
69      </AuthContext.Provider>
70  }
```

Рисунок 2Б - Компонент-провайдер функцій та даних автентифікованого користувача

ДОДАТОК В

```
import { combineReducers, configureStore } from "@reduxjs/toolkit";
import hotelReducer from "./hotelReducer";
import reservationReducer from "./reservationReducer";
import userReducer from "./userReducer";
import employeesReducer from "./employeesReducer";
import cleaningReducer from "./cleaningReducer";
import { getDefaultMiddleware } from '@reduxjs/toolkit';


const rootReducer = combineReducers({hotelReducer, reservationReducer, userReducer, employeesReducer, cleaningReducer});

export const store = configureStore({reducer: rootReducer});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

Рисунок 1В – Файл створення сховища даних та під’єднання його станів

ДОДАТОК Г

```tsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import { BrowserRouter } from 'react-router-dom'
import '@fontsource/roboto/300.css';
import '@fontsource/roboto/400.css';
import '@fontsource/roboto/500.css';
import '@fontsource/roboto/700.css';
import './index.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import AuthProvider from './contexts/AuthContext';
import { Provider } from 'react-redux';
import { store } from './store/store';

ReactDOM.createRoot(document.getElementById('root') as HTMLElement).render(
  <BrowserRouter>
    <AuthProvider>
      <Provider store={store}>
        <App />
      </Provider>
    </AuthProvider>
  </BrowserRouter>
)
```

Рисунок 1Г – Файл відмальовування кореневого компоненту веб-застосунку