Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет комп'ютерних систем і автоматики
(повне найменування інституту, факультету)

_____Кафедра комп'ютерних систем управління_____
(повна назва кафедри)

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Автоматизована система управління адміністративними функціями готелю»

Виконав: студент 2-го курсу, групи
2АКІТ-20м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

_____ Владислав Таламанюк
(ім'я та прізвище)

Керівник: к.т.н., доцент каф. КСУ

_____ Марія Юхимчук
(ім'я та прізвище)
«_15_» _____12_____ 2021 р.

Опонент: к.т.н., доцент каф.. АІІТ

_____ Марія Барабан
(ім'я та прізвище)
«_16_» _____12_____ 2021 р.

Допущено до захисту
Завідувач кафедри КСУ
д.т.н., проф.
___Володимир Дубовой___
(ім'я та прізвище)
«_17_» ____12____ 2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра комп'ютерних систем управління
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 15 Автоматизація та приладобудування
Спеціальність – 151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
завідувач кафедри КСУ
д.т.н. проф. Володимир Дубовой
(прізвище, ім'я, по батькові)
« _01_ » ____10___ 2021 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Таламанюку Владиславу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Автоматизована система управління адміністративними функціями готелю
Керівник проекту (роботи) Юхимчук М. С., к.т.н., доцент каф. КСУ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджена наказом закладу вищої освіти від «24» 09 2021 року № 277
2. Строк подання студентом магістерської кваліфікаційної роботи 10.12.2021р.
3. Вихідні дані до роботи Організація управління готельним бізнесом, загальні алгоритми управління готелем, мова програмування JavaScript
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розкрити) огляд предметної області; реалізація програмного модулю; економічне обґрунтування
5. Перелік графічного матеріалу Use-Case діаграми, вигляд екранів розробленого додатку

6. Консультанти розділів роботи

Розділ змістовної частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний розділ	к.е.н, доц каф. ЕПВМ, Кавецький В. В.		

7. Дата видачі завдання «_01__» _____10_____ 2021_ року.

КАЛЕНДАРНИЙ ПЛАН

№ етапу	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи (проекту)	Примітка
1	Дослідження актуальності поставленої задачі	04.10.2021р.	
2	Загальний огляд автоматизованих систем управління адміністративними функціями готелю	22.10.2021р.	
3	Аналіз функцій автоматизованих систем управління адміністративними функціями готелів	01.11.2021р.	
4	Розробка структури програмного забезпечення системи	03.11.2021р.	
5	Підготовка економічної частини	12.11.2021р.	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	08.12.2021р.	
7	Графічні матеріали: Use-Case діаграми Вигляд екранів розробленого додатку	 01.12.2021р. 03.12.2021р.	
8	Захист МКР	22.12.2021р.	

Студент _____ Владислав Таламанюк
(підпис) (ім'я та прізвище)

Керівник роботи (проекту) _____ Марія Юхимчук
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

В даній магістерській дипломній роботі розглянуто автоматизування управління адміністративними функціями готелю. В ході виконання роботи було створено власні алгоритми, які являються основною частиною опрацювання даних клієнтів і самого персоналу готелю. Програмне забезпечення реалізовано на мові програмування JavaScript в середовищі розробки VisualStudi Code, використовуючи систему управління базами даних MySQL Workbench та MongoDB. Зроблено теоретичну й технічну частину роботи. Було здійснено оптимізацію, й тестування розробленого продукту.

ABSTRACT

In this master's thesis automation of management of administrative functions of hotel is considered. In the course of the work, our own algorithms were created, which are the main part of the processing of customer data and the hotel staff. The software is implemented in the JavaScript programming language in the VisualStudio Code development environment, using the database management system MySQL Workbench and MongoDB. The theoretical and technical part of the work is done. The developed product was optimized and tested.

ЗМІСТ

ВСТУП

Актуальність. В сучасному світі уявити життя людини без цифрової техніки неможливо, до того ж, необхідність у модернізації такої техніки з кожним днем стає все більшою, та більш гострою, оскільки задачі які людина хоче виконувати на цифрових приладах стають все більш тяжчі та об'ємні.

Одним з прикладів результату розвитку цифрової техніки є комп'ютер, після того як людина освоїла його, вона почала делегувати йому задачі, які самостійно робила б довгий період часу, це допомогло людині виконувати масу процесів, не за десятки годин, а за десятки хвилин, одним із прикладів таких процесів є адміністрування сфер послуг.

Раніше процес адміністрування, до прикладу готелю, відбувався доволі примітивно, усі дані про клієнта записувались на листочку, бронювання номеру відбувалось або на місці, або за телефоном. Такий підхід часто призводив до помилок у даних, з якими працював персонал.

У процесі модернізації комп'ютерів та комп'ютерних систем, людство делегувало процес роботи з даними машинам, у випадку з готельним бізнесом це допомогло більш ефективно опрацьовувати дані, та зберігати їх у недоторканій формі.

На сьогоднішній день людство прагне зробити більшість процесів автоматизованими, саме тому почали з'являтись багато автоматизованих систем управління, ціллю яких є вилучення людини з процесу роботи з даними, та зробити ці процеси більш ефективними. Автоматизовані системи управління готелем є яскравим прикладом автоматизації комп'ютерних систем.

Серед таких систем є багато різних зарубіжних аналогів (Opera hotel PMS, Edelweiss, epitome PMS та ін.) які мають велику кількість функціоналу для адміністрування процесів готелю. Основною проблемою всіх цих систем являється перенавантажений інтерфейс, який може зробити роботу персоналу з подібним ПЗ проблемою, та створити труднощі у робочих процесах готелю. Саме тому

необхідний подальший розвиток таких систем щоб впроваджувати нові ідеї та принципи роботи таких систем.

Головним чином такі АСУ допомагають розвивати туристичний бізнес, відповідно створювати робочі місця, стимулювати економічні процеси у такому регіоні, та в країні в цілому. Для України туристичний та готельно-ресторанний бізнес це одна з перспективних галузей розвитку економіки та сфери послуг, саме тому створення автоматизованих систем управління готелем є актуальним і сьогодні.

Наукова новизна.

1. Запропоновано новий підхід до реалізації автоматизованої системи управління адміністративними функціями готелів, який на відміну від існуючих надає більш комфортну взаємодію із застосунком..

2. На основі запропонованого підходу розроблено додаток, який ліг в системи автоматизованого управління адміністративними функціями готелю, яка на відміну від існуючих може працювати у автоматичному та автоматизованому режимі, що дає змогу покращити ефективність роботи при адмініструванні готелю.

Мета роботи. Головним чином потрібно зробити автоматизовану систему управління адміністративними функціями готелю, яка допоможе зробити процеси адміністрування готелю більш ефективними.

Для досягнення мети потрібно виконати поставлені задачі, які наведені нижче:

- Проаналізувати існуючі рішення для автоматизованого адміністрування готелями.
- Використовуючи результати аналізу існуючих рішень запропонувати власну ідею рішення та обґрунтувати його раціональність та конкурентоспроможність по відношенню до існуючих аналогів.
- Розробити модель та алгоритм роботи програми на основі запропонованої ідеї.

- Реалізувати запропоновану модель у вигляді кросплатформеного застосунку для електронно обчислювальних машин які використовують операційні системи з графічним інтерфейсом.
- Протестувати та дослідити готове рішення.

Об’єкт дослідження. Процес управління готелем

Предмет дослідження. Принцип автоматизації управління готелем

Методи дослідження. При виконанні роботи використовувалися економічні методи аналізу роботи адміністративних функцій готелю.

Новизна результату роботи полягає у розробленій концепції автоматизованих систем управління готелями, на основі якої є можливість створювати функціональні програмні продукти.

Практична цінність роботи полягає у програмній реалізації автоматизованої системи адміністративними функціями готелю.

1. ОГЛЯД СТАНУ ПРОБЛЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ АДМІНІСТРАТИВНИМИ ФУНКЦІЯМИ ГОТЕЛЮ

В розділі розкрито поняття автоматизації управління адміністративними функціями готелів, наведено приклад існуючих рішень та обґрунтовано вибір рішення для подальшого дослідження методів автоматизованого управління.

Наведений перелік існуючих рішень автоматизації управління адміністративними функціями готелів, зазначено сфери використання описаних рішень та обґрунтовано вибір оптимального рішення для подальшої розробки аналогу автоматизованої системи управління адміністративними функціями готелів.

1.1. Сутність об’єкту дослідження

Готельно-ресторанний бізнес – це одна з найбільших та найбільш необхідних сфер послуг у світі. Свій початок дана сфера бере ще з XVI століття н. е. Саме тоді англійський король Генріх VIII видав указ про переведення церковної власності (яка надавала безкоштовний притулок усім подорожуючим паломникам в святі місця) у світську, зробивши надання послуг по гостинності платними, стимулювавши, тим самим, початок розвитку сфери.

Згодом в Парижі XVIII ст. вперше з’явився термін «готель». Так в Парижі називали будинки в декілька поверхів, які складались з невеликих окремих апартаментів, які подорожуючі та навіть жителі міста могли орендувати на любий термін. Таким чином американським власникам заїжджих дворів та придорожніх таверн припала до душі така французька назва, якою вони прагнули підкреслити елітність свого закладу.

В той самий час у США, завдяки великому потоку емігрантів, було покладено різкий початок розвитку готельного бізнесу, створенню великих корпорацій готелів. Першим закладом що відчинив двері для відвідувачів став перший

розкішний міський готель люкс класу в Бостоні, Нью Йорк – «Тремонт». З того часу Європа також почала окутувати мода на фешенебельні готелі, оснащені найбільш передовими, на той час, зручностями, які були шедеврами архітектури.[47]

Вперше помітний розвиток готельного бізнесу в Україні почався у другій половині XIX ст. у Києві та інших великих містах. Поштовхом який стимулював розвиток готельного бізнесу стало відкриття у 1889 році регулярного залізничного сполучення, що зумовлює збільшення кількості туристів та подорожуючих, прибуваючих в Київ. До 1880 року у Києві нараховувалось 15 готелів, упродовж наступних 20 років у центральній частині міста (головним чином на Хрещатику та прилеглих вулиць) було побудовано 64 нових готелі, найбільше за всю історію міста.

Надалі почався інтенсивний розвиток готельного бізнесу у всьому світі. З кожним разом створювались все новіші методи приваблення клієнтів, в тому числі побудова готелів поблизу туристичних об'єктів, або місць прибування або відбування людей з міста (вокзали, автовокзали, аеропорти). З розвитком технологій готелі також почали використовувати їх у своїх цілях, в т. ч. для забезпечення більш ефективного адміністрування готелю.[48]

З останніми досягненнями у розвитку технологій найбільшої популярності дістали автоматизовані системи управління готелями, що надало можливість зробити процеси адміністрування більш автономними та ефективними, це дозволило прибрати з ланцюга роботи з даними людський фактор, а відповідно зменшити вірогідність виникнення помилок та некоректних даних при обробці даних клієнта, ведення бухгалтерської звітності та інших процесів керування готелем.[24]

Основні процеси які передбачає автоматизована системи управління адміністративними функціями готелю – це:

- Обробка замовлень для резервування номерів
- Обробка замовлень для резервування столиків у ресторані при готелі
- Автоматизація процесу роботи обслуговуючого персоналу

- Робота зі звітністю по ресурсам готелю
- Робота зі звідністю по дохідності готелю

Детально про роботу основних процесів готелю описано на рис. 1.1.

Згідно з наведеним вище переліком функцій які повинна виконувати автоматизована система управління адміністративними функціями готелю, можна зробити висновок, що така система є комплексним рішенням багатьох різних процесів у готелі, що робить таку систему ефективним інструментом в адмініструванні готелю в цілому.

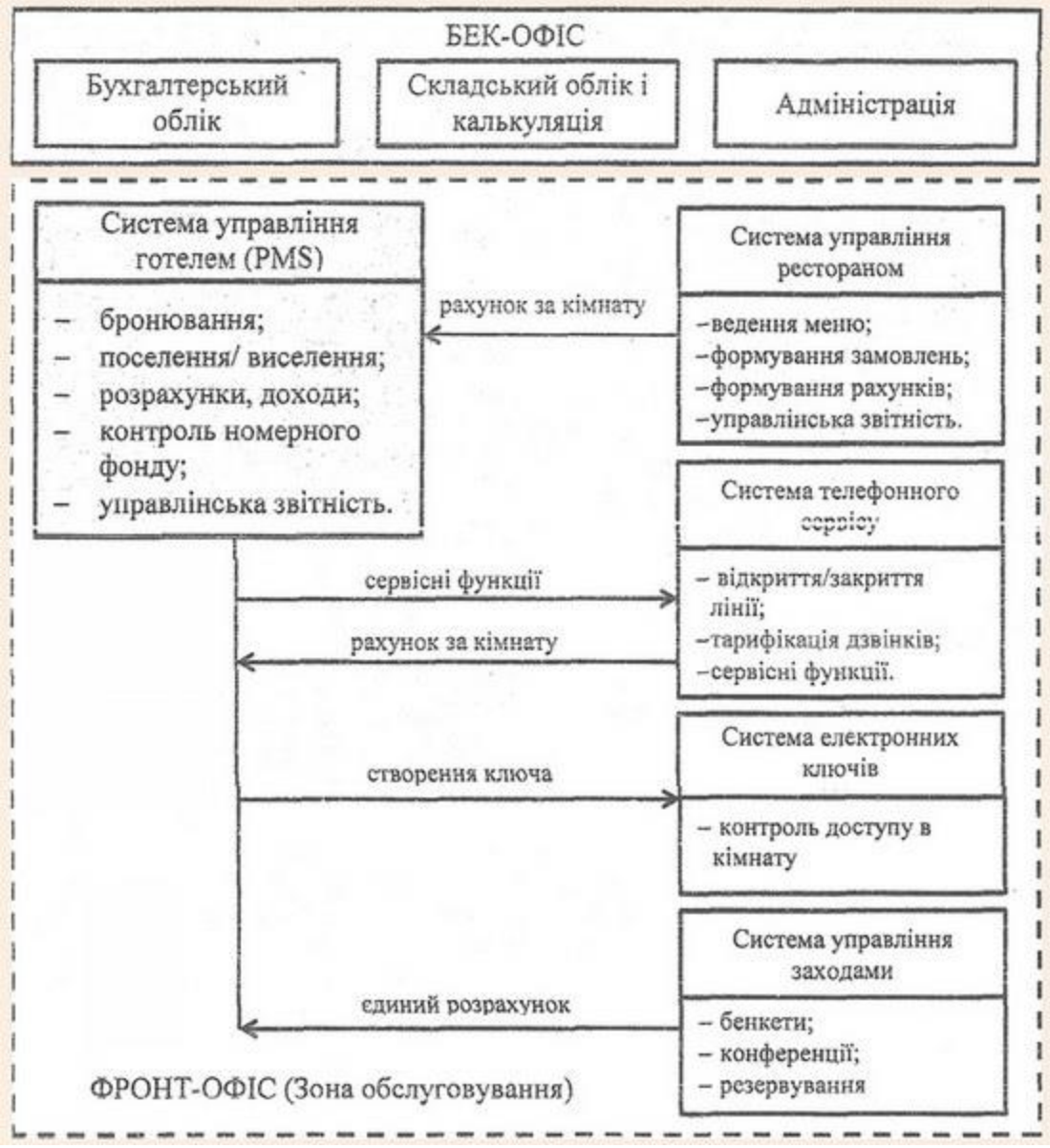


Рисунок 1.1. приклад роботи процесів готелю [46]

1.2. Приклади існуючих рішень для автоматизації управління адміністративними функціями готелів

Для проведення повноцінного аналізу на основі якого буде отримано необхідні дані для створення моделі, та подальшої розробки програмного рішення, потрібно переглянути існуючі рішення для автоматизації управління адміністративними функціями готелів.

1.2.1. Cheerze Connect система готельного менеджменту

Дана система передбачена для комплексного використання у процесах повного циклу адміністрування готелів.

Серед основних функцій даного програмного рішення можна виділити:

- керування номерами готелю – в даному програмному рішенні це допомагає значно спростити процес поселення, виселення, та ведення клієнта під час перебування у готелі. Завдяки хмарним технологіям забезпечується безпека, та недоторканість даних злочинцями, також це надає можливість зробити дані корисними, оскільки відкривається великий перелік функцій по аналізу даних, що надає потужний інструмент для забезпечення більш ефективної роботи готелю та процесу використання номерного фонду готелю;
- керування рестораном готелю – це програмне рішення використовує принцип описаний вище та забезпечує керування бронюванням, та обслуговуванням столику у ресторані готелю.
- Адміністрування обліку активів – це програмне рішення яке надає потужний інструмент для аудиту активів готелю, що значно спрощує процес створення обробки фінансових звітів готелю, це підвищує ефективність ведення бізнесу.
- Адміністрування веб-сайту готелю – дане рішення надає можливість керувати вмістом веб-сайту, для забезпечення безперешкодної актуалізації інформації, яку потрібно донести до користувача.

- Адміністрування персоналу – це рішення яке надає можливість з легкістю керувати персоналом готелю, надавати робітникам роботу, та без будь яких перешкод керувати зайнятістю персоналу.

Дана система має простий та ненавантажений інтерфейс користувача (рис. 1.2.), що робить приємною у використанні, не заплутує, користувача програмного продукту.[49]

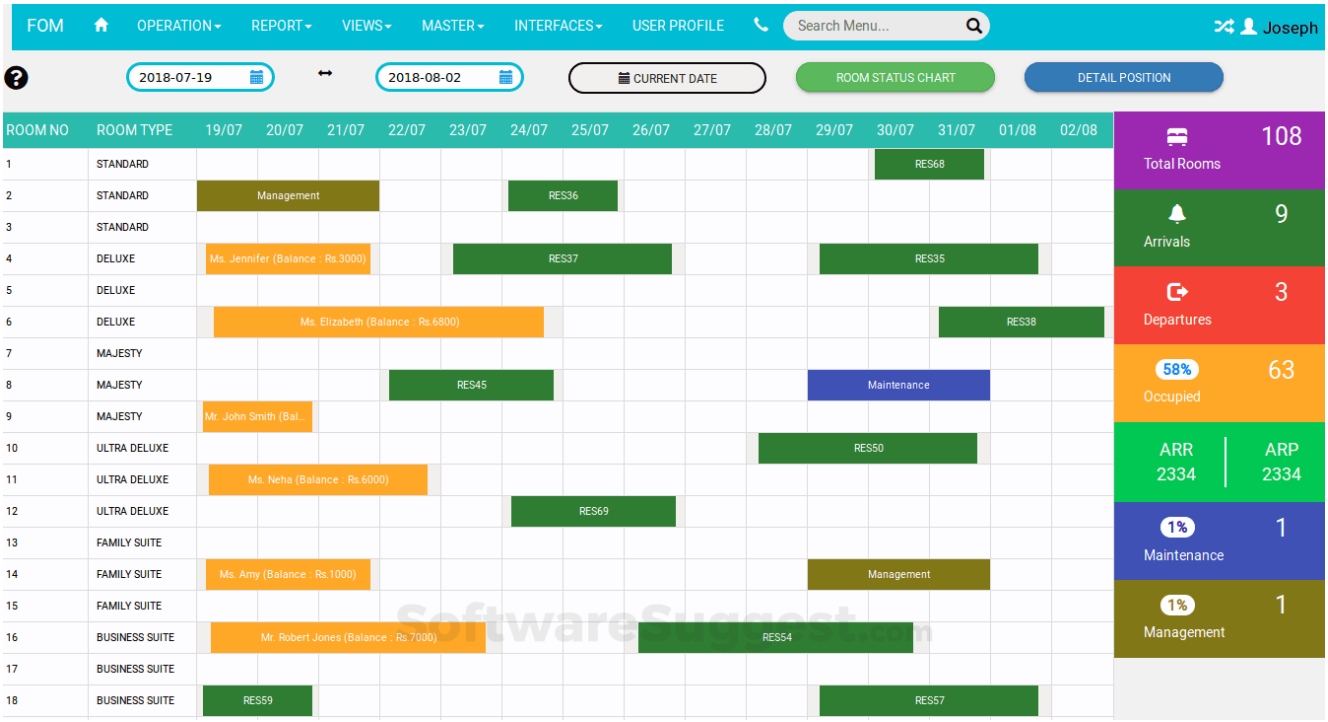


Рисунок 1.2. Приклад інтерфейсу користувача програми Cheerze Connect

1.2.2. StayFlexi – система менеджменту сферами послуг

На відміну від попереднього прикладу автоматизованої системи управління готелем, дана система також передбачає велику кількість інших автоматизованих систем управління, що робить її універсальним інструментом у роботі з комплексом автоматизованих систем управління сфери послуг об'єднаних між собою (рис. 1.3.).

За оцінками різних користувачів, представлене програмне рішення має, дещо, перенавантажений, але інтерактивний інтерфейс користувача. Головною перевагою цієї платформи над іншими можна вважати різноманітний інтерфейс, та різну форму сортування табличних елементів[50]

Рисунок 1.3. Приклад інтерфейсу користувача програми StayFlexi

1.2.3. Oracle Hospitality Property Management Systems – автоматизована система управління готелем

Найбільш популярна система автоматизованого управління готелем. Система розроблена виключно для управління готельним бізнесом.

На відміну від попередніх систем, вона може використовувати як хмарну систему баз даних, так і має можливість розгортання локальної бази даних на серверах готелю. Це надає можливість підвищити безпеку корпоративної таємниці бізнесу.

Оцінка користувачів вказує на те що система організована з використанням багатьох технологій для забезпечення вузького профілю роботи програми, тобто робота з управлінням готелю.

Велика кількість технологій та функціоналу перенавантажує графічний інтерфейс (рис. 1.4.), що робить його менш зрозумілим для непідготовленого користувача.[51]

Рисунок 1.4. Приклад графічного інтерфейсу програми Oracle Hospitality Property Management Systems

1.3. Cheerze Connect як найбільш популярна система автоматизованого управління готелями

Виконавши аналіз існуючих програмних рішень для автоматизації управління адміністративними функціями готелю було прийнято рішення використовувати систему Cheerze Connect.

1.3.1. Детальний аналіз схеми роботи вибраного програмного рішення.

Згідно з даними вказаними на офіційному сайті розробника даного програмного забезпечення, у цій системі використовується велика кількість мікросервісів, які пов’язані між собою єдиним комунікаційним сервером, який дозволяє опрацьовувати дані з різних мікросервісів, та об’єднувати отримані від них дані у єдиній базі даних, це дозволяє обмежити доступ серверу який працює з певним мікросервісом до прямого підключення до бази даних.

Схема роботи мікросервісів даного сервісу виглядає слідуючим чином. Система має 12 мікросервісів:

- Front Office (даний мікросервіс відповідає за керування роботою персоналом, що дозволяє зробити послуги готелю більш ефективними та швидкими)
- POS (це міркосервіс який відповідає за обробку платежів всередині готелю)
- Mobile App (мобільний додаток теж був виведений до міркросервісів, оскільки виконує лише частковий функціонал всієї системи та служить як елемент швидкого доступу до необхідних функцій адміністрування)
- Booking Engine (мікросервіс який виконує керування усім номерним фондом готелю)
- та інші.

Усі ці мікросервіси використовують схему ієрархічну схему зв'язку між собою де на чолі знаходиться база даних усієї системи, далі йде єдиний сервер провайдер даних, який має безпосередній зв'язок з базою даних, згідно з ієрархією, жоден інший сервер не повинен з'єднуватись з базою даних. Наступними у ієрархії розміщені сервери мікросервісів, які виконують функцію обробки даних зі сторони серверу та зі сторони клієнту, в кінці до кожного з серверів з кожним з серверів повинні з'єднуватись клієнтські частини мікросервісів (інтерфейси користувача). Візуальна блок схема роботи мікросервісів наведена на рисунку 1.5.

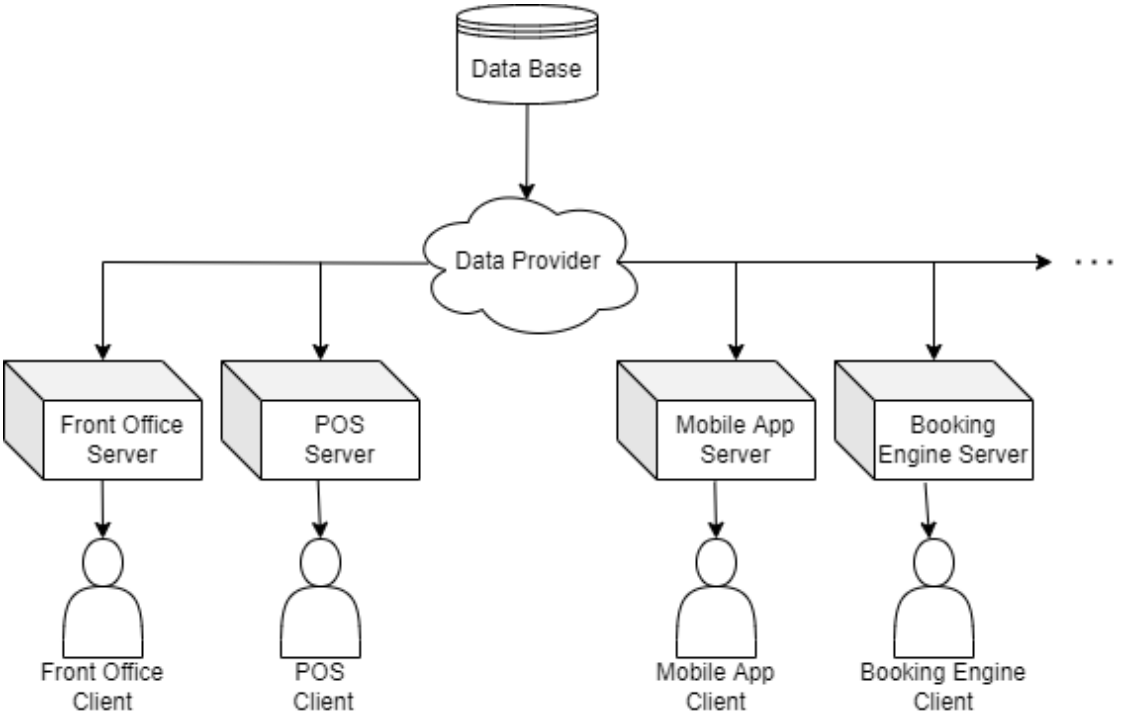


Рисунок 1.5. Візуальна блок схема мікросервісів.

1.3.2. Детальний аналіз інтерфейсу користувача вибраного програмного рішення.

Головним чином інтерфейс користувача, обраного для детального аналізу програмного продукту, використовує структуру, яка дозволяє без будь яких проблем орієнтуватись у інтерфейсі, та швидко розуміти що потрібно натискати для керування програмним продуктом.

На прикладі (рис. 1.6) нижче буде описано послідовність використання інтерфейсу користувача системи роботи з номерним фондом готелю у вибраному програмному рішенні.

Структура зони керування номерним фондом виконана у вигляді так званої «шахматки», яка дозволяє за допомогою лівої грані орієнтуватись за номерами, а верхня грань дозволяє орієнтуватись по датам за якими можна зрозуміти статус номеру (використовується, вільний, або зарезервований).

У верхньому меню можна отримати доступ до інших сервісів програми. Також нижче розташоване меню конфігурації пошуку по датам, та деталі обраного номеру за датою.

З правої сторони розташовані короткі загальні дані про номери готелю (загальна кількість кімнат, кімнати в які заїжджають, кімнати які використовуються, кімнати які звільняють, та інші).

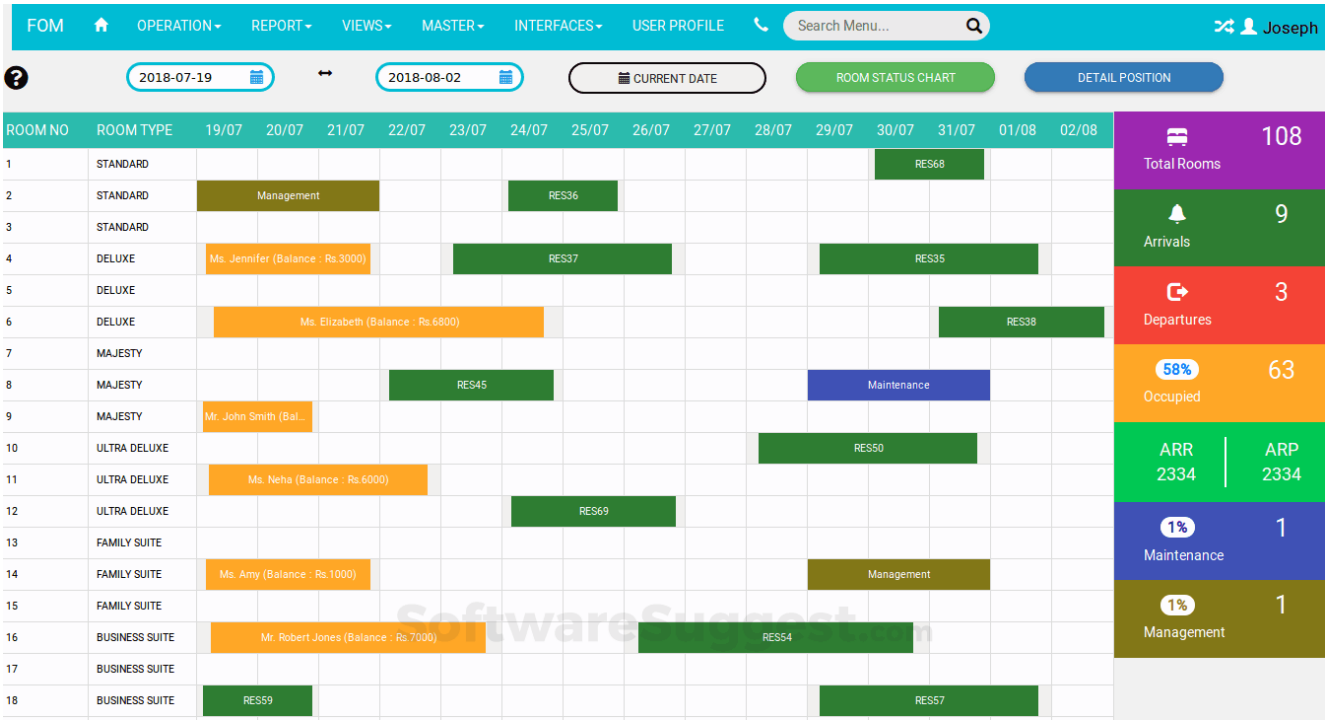


Рисунок 1.6. Приклад інтерфейсу обраного програмного рішення

1.4. Підходи до вирішення проблем

Основним підходом до вирішення поставленої задачі є створення життєвого циклу програмного продукту, це дозволить зробити процес розробки більш швидким та зрозумілим, а в подальшому організувати його підтримку.

На рисунку 1.7 наведено життєвий цикл програмного продукту каскадного типу. Головними перевагами даного методу являються:

- жодних, або майже жодних редагувань
- гарна специфікація, яка частіше перетікаю в гарну документацію
- зрозуміла модель
- програмісти можуть мати низьку кваліфікацію

Також даний метод життєвого циклу має ряд недоліків, а саме:

- необхідне досягнення досконалості на кожному етапі
- може бути складно вносити зміни
- надлишкове проектування
- поділ розробників на «ідеальних» та «недолугих»

Незважаючи на недоліки, які буди представлені по відношенню до даного методу життєвого циклу, його можна вважати найбільш правильним, оскільки проект, до якого буде його використано, має невеликий об’єм.

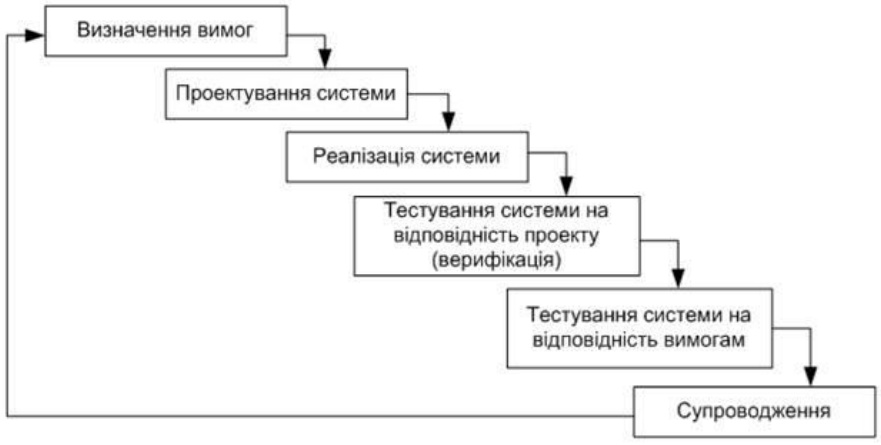


Рисунок 1.7. Метод «Каскадного» або «Водоспадного» життєвого циклу програмного продукту.

1.5. Автоматизація адміністративних функцій готелю

Автоматизація являється одним з головних напрямів науково технічного прогресу, який призваний залучити у процеси виробництва саморегульовані технічні засоби, звільнивши людину від участі у цих процесах.

Аналогічним чином такий напрям можна використовувати також для автоматизації сфери послуг, що також звільнить людину від процесів обслуговування передавши всі подібні функції саморегульованим технічним засобам (надалі – автоматизованим системам управління).

Для того щоб можна було автоматизувати адміністративні функції готелю, необхідно створити алгоритм, який самостійно буде реагувати на певні події які викликаються певними діями.

Оскільки передбачається велика кількість функцій які потрібно автоматизувати, то відповідно є необхідність створення алгоритму для кожної автоматизованої системи окремо. Такий підхід може викликати створення великої кількості мікросервісів, які використовуватимуть особистий алгоритм автоматизації, саме тому більшість розробників вдаються до використання вже готово коду, який дозволяє використовувати як сторонні сервіси, так і сервіси, які знаходяться на відкритих репозиторіях.

1.6. Запропонований метод вирішення проблеми

Для автоматизації системи створюваного програмного рішення, необхідно використовувати підхід, який дозволить зробити систему універсальною (такою що зможе працювати на різних операційних системах), для цього необхідно буде обрати мову програмування, яка дозволяє працювати з різними операційними системами, та мати однаковий набір функціоналу на будь-якій з них. Для того щоб виконати повноцінну автоматизацію програмного продукту необхідно буде розділити програмний продукт на деяку кількість мікросервісів, які допоможуть розділити обов’язки кожного з них, та передбачити перетинання обов’язків між

собою, такий підхід повинен зробити програмний продукт більш стійким до неочікуваних збоїв у роботи програмного продукту, та дозволить в майбутньому розширювати функціонал програмного продукту.

Після автоматизації необхідно забезпечити недоторканість даних, та безпеку автентифікації, це можна реалізувати за допомогою ієрархії клієнт-серверного зв’язку описаного у пункті 1.3.1, але додатково необхідно розділити бази даних які працюють з даними користувачів готелю, та даними автентифікації клієнтської частини програмного продукту, за допомогою розділення їх по різним серверам.

2. ДОСЛІДЖЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СУЧАСНИХ ПРОГРАМНИХ РІШЕНЬ

2.1. Класифікація систем за архітектурою

Архітектура інформаційно-обчислювальної системи базується на апаратних (комп'ютерах), телекомунікаціях і програмному забезпеченні. Рівень розвитку кожної із складових визначається досконалістю інформаційної системи, технології обробки даних, що призвело до появи таких схем обробки даних: телеобробка; файловий сервер; клієнт-сервер; Інтернет система; сховище даних та система оперативно-аналітичної обробки даних тощо.

2.1.1. Архітектура файл-сервер

Інформаційна система такого типу складається з трьох компонентів: сервера баз даних, клієнта (ПК з клієнтськими додатками та СУБД), мережевого та комунікаційного програмного забезпечення (рис. 2.1.).

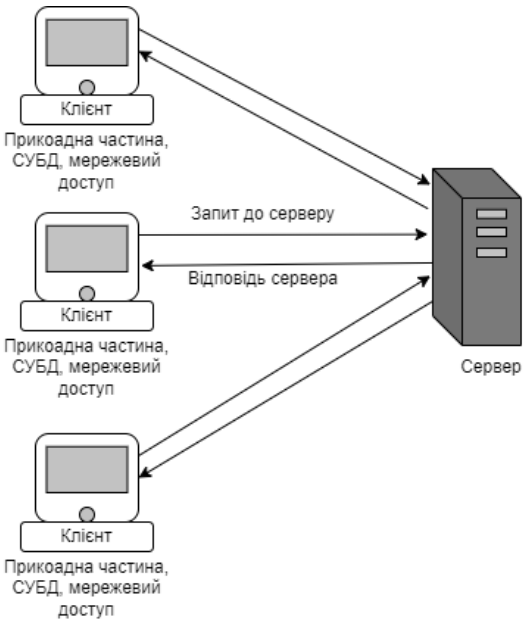


Рисунок 2.1. – Архітектура файл-сервер

Сервер містить СУБД і файли, необхідні для роботи клієнтських програм. Клієнтські програми та їх персональні СУБД розташовані та працюють на окремих

робочих станціях і звертаються до файлового сервера лише тоді, коли необхідно отримати доступ до файлів. Сервер вибирає необхідні файли з бази даних (а не їх окремі записи), які відправляються мережею клієнту для обробки. Таким чином, файловий сервер діє як спільний жорсткий диск. Архітектура файлового сервера характеризується такими основними недоліками: великий обсяг мережевої графіки; кожна робоча станція повинна мати повну копію СУБД користувача; Управління паралельністю, відновленням та цілісністю бази даних є складнішим, оскільки до тих самих файлів одночасно звертаються кілька СУБД.

2.1.2. Архітектура клієнт-сервер

Інформаційна система клієнт-сервер складається з трьох основних компонентів: серверного програмного забезпечення; програмне забезпечення кінцевого користувача; проміжне програмне забезпечення (рис. 2.2.).

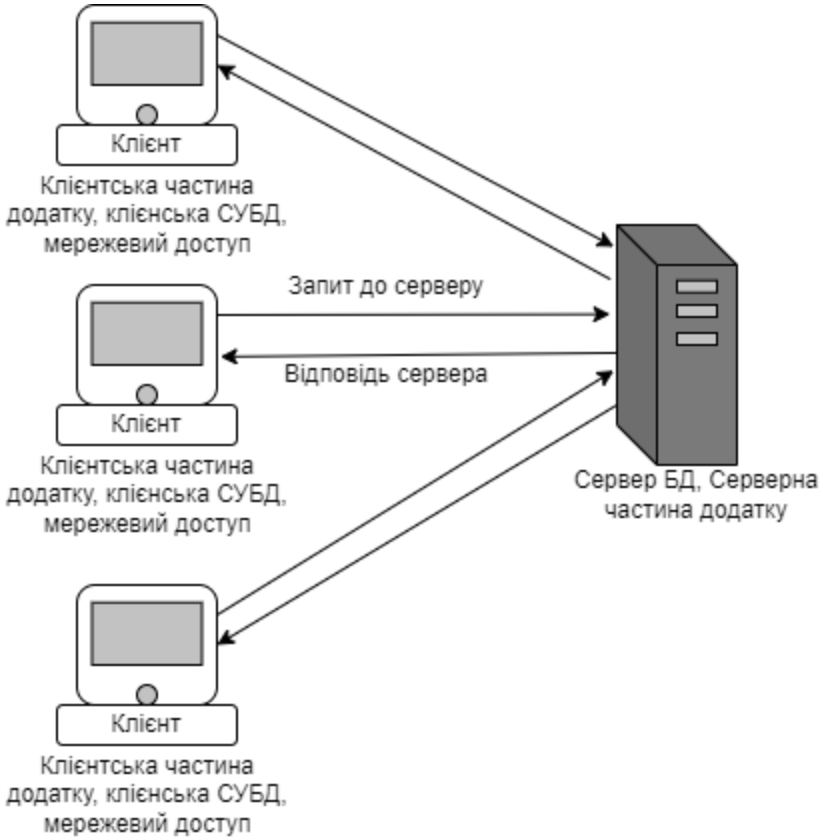


Рисунок 2.2. – Дворівнева архітектура клієнт-сервер.

Серверне програмне забезпечення забезпечує обслуговування клієнтів. Для реалізації архітектури клієнт-сервер зазвичай використовуються багатокористувацькі СУБД, наприклад, Oracle або Microsoft SQL Server. Ці СУБД забезпечують механізми блокування та багатокористувацький контроль доступу для захисту даних від небезпек одночасного доступу. Крім того, сервер баз даних повинен захищати дані від несанкціонованого доступу, оптимізувати запити до бази даних, забезпечувати цілісність даних і контролювати завершення транзакцій.

В організації структури «клієнт-сервер» можуть бути досить «тонкими», а сервер повинен бути достатньо товстим, щоб задовольнити потреби всіх клієнтів. Програмне забезпечення для кінцевих користувачів включає засоби розробки програмного забезпечення та генератори звітів, включаючи електронні таблиці та текстові процесори.

За допомогою цього програмного забезпечення користувачі встановлюють з'єднання з сервером, формують запити, які автоматично формуються в SQL-запити і відправляються на сервер. Сервер приймає та обробляє запити, а потім передає результати клієнтам.

Проміжне програмне забезпечення — це частина системи клієнт-сервер, яка пов’язує програмне забезпечення кінцевого користувача із сервером. Схема клієнт-сервер проста: клієнт відправляє серверу запит на необхідні дані; сервер приймає їх, обробляє та надсилає клієнту лише ті дані, які були замовлені.

Дворівнева модель клієнт/сервер оптимальна для підприємств з менш ніж 100 користувачами, оскільки операційна система сервера перевантажена керуванням кількома підключеннями до сервера при обслуговуванні великої кількості клієнтів.

Трирівнева модель, на відміну від дворівневої, вирішує проблеми масштабування. При використанні трирівневої моделі, крім клієнта і сервера, є ще додатковий проміжний ланцюжок (сервер додатків), який керує транзакціями – аналізує запити, організовує їх чергу, направляє запити на виконання тощо (рис. 2.3.).

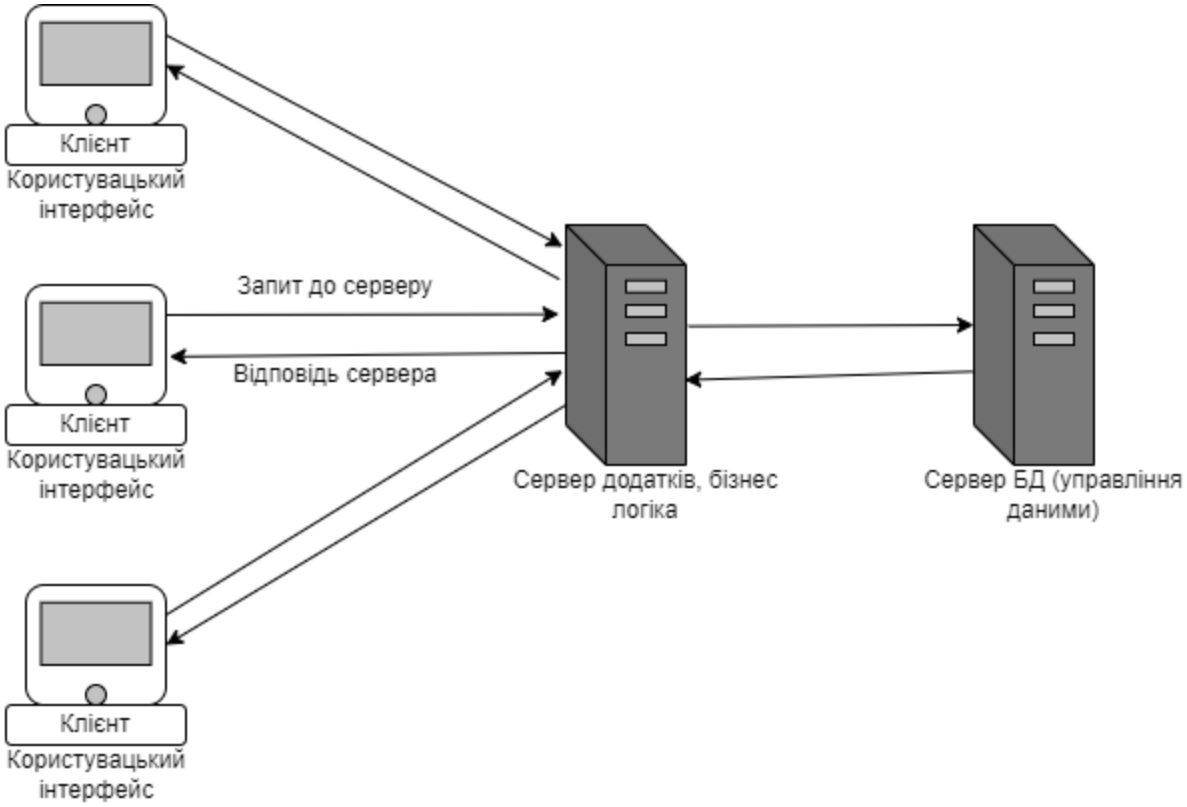


Рисунок 2.3. – Трирівнева архітектура клієнт-сервер

Інформаційні системи клієнт-сервер мають ряд переваг перед файлово-серверними інформаційними системами.

- По-перше, це зменшує мережевий трафік при виконанні запитів.
- По-друге, архітектура клієнт-сервер стає незамінною, коли кількість користувачів, які одночасно використовують одні й ті ж дані, перевищує тисячі користувачів.

Ще однією перевагою архітектури клієнт-сервер є можливість зберігати бізнес-правила на сервері, що дозволяє уникнути дублювання коду в різних програмах, які використовують загальну базу даних.

Крім перерахованих переваг, сучасні серверні СУБД мають широкі можливості для управління привілеями користувачів і правами доступу до різних об’єктів бази даних, резервного копіювання та архівування даних, а також оптимізації виконання запитів.

2.2. Вибір архітектури системи

Ця система має трирівневу архітектуру, яка передбачає наявність наступних програмних компонентів: клієнтський додаток (як правило, його називають «тонким клієнтом» або термінал), підключений до сервера додатків, який, у свою чергу, підключений до сервера бази даних.

Клієнт — це (зазвичай графічний) компонент, який представляє перший рівень для кінцевого користувача. Перший рівень не повинен мати прямих підключень до бази даних (для вимог безпеки), не повинен завантажуватися основною бізнес-логікою (для вимог до масштабованості) і зберігати стан програми (для вимог надійності). Найпростіша бізнес-логіка може бути і зазвичай виноситься на перший рівень: інтерфейс авторизації, алгоритми шифрування, перевірка введених значень на дійсність і відповідність формату, прості операції (сортування, групування, підрахунок значень) з уже завантаженими даними. термінал.

Сервер додатків розміщений на другому рівні. Другий рівень містить більшу частину бізнес-логіки. За його межами знаходяться фрагменти, експортовані в термінали (див. вище), а також збережені процедури та тригери, розташовані на третьому рівні.

У «правильній» (з точки зору безпеки, надійності, масштабованості) конфігурації сервер баз даних міститься на виділеному комп'ютері (або кластері), до якого через мережу підключено один або кілька серверів додатків, до яких, у свою чергу, термінали. підключені через мережу.

У порівнянні з архітектурою клієнт-сервер або файл-сервер можна виділити наступні переваги трирівневої архітектури:

- масштабованість;
- конфігураційність
- ізоляція рівнів один від одного дозволяє (при правильному розгортанні архітектури) швидко і легко переналаштувати систему у разі збоїв або під час планового обслуговування на одному з рівнів;
- високий рівень безпеки;

- висока надійність;
- низькі вимоги до швидкості каналу (мережі) між терміналами та сервером додатків;
- низькі вимоги до продуктивності та технічних характеристик терміналів, внаслідок зниження їх вартості. Терміналом може бути не тільки комп'ютер, а й, наприклад, мобільний телефон.

Недоліки випливають із переваг. У порівнянні з архітектурою клієнт-сервер або файл-сервер можна виділити наступні недоліки трирівневої архітектури:

- більша складність створення додатків;
- більш складні в розгортанні та адмініструванні;
- високі вимоги до продуктивності серверів додатків і сервера баз даних, і, як наслідок, висока вартість серверного обладнання;

Незважаючи на недоліки, більшою мірою пов'язані з вимогами продуктивності окремих частин сервера, буде використовуватися трирівнева архітектура (рис. 2.4.), оскільки переваг набагато більше, ніж недоліків ніш.

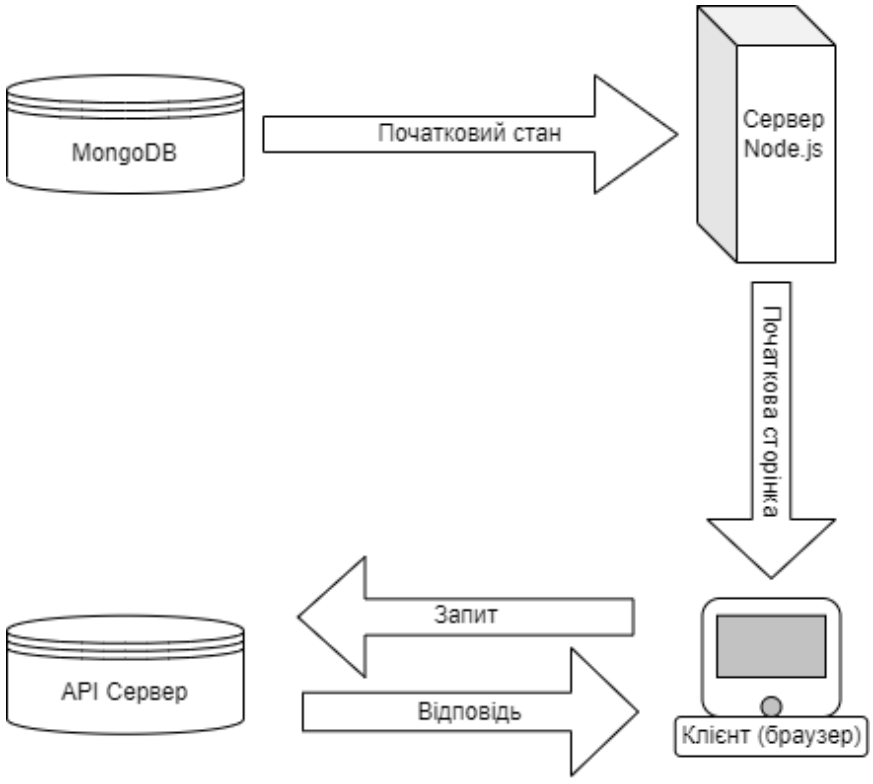


Рисунок 2.4. Трирівнева архітектура клієнт-сервер системи виявлення плагіату

2.3. Вибір системи управління базами даних

MongoDB використовує новий підхід до створення баз даних без таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які є поширеними в об'єктно-реляційних базах даних.

Ще в 2000-х роках було загальноприйнятою практикою зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). Було не так велико значення, чи підходять реляційні бази даних для зберігання такого типу даних чи ні.

На відміну від реляційних баз даних, MongoDB пропонує модель даних, орієнтовану на робочий процес, що робить MongoDB швидшим, масштабнішим і простішим у використанні.

Але, навіть беручи до уваги всі недоліки традиційних баз даних і переваги MongoDB, важливо розуміти, що завдання різні, а методи їх вирішення різні. У деяких ситуаціях MongoDB дійсно покращить продуктивність вашої програми, наприклад, якщо вам потрібно зберігати дані, які мають складну структуру. Інакше було б краще використовувати традиційні реляційні бази даних. Крім того, ви можете використовувати змішані підходи: зберігати один тип даних у MongoDB, а інший — у традиційних базах даних.[29]

Вся система MongoDB може представляти більше однієї бази даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розміщувати декілька баз даних на кількох фізичних серверах, і ці бази даних можуть легко обмінюватися даними та підтримувати узгодженість.

Одним із популярних стандартів для обміну та зберігання даних є JSON (JavaScript Object Notation). JSON ефективно описує дані, які містять складну структуру. У цьому відношенні спосіб зберігання даних MongoDB подібний до JSON, хоча формально JSON не використовується. MongoDB зберігається у форматі BSON (Beeson) або скорочення від бінарного JSON.[31]

BSON дозволяє працювати з даними швидше: швидше пошук і обробка. Хоча слід зазначити, що BSON, на відміну від зберігання даних у форматі JSON, має

невеликий недолік: загалом дані у форматі JSON займають менше місця, ніж у форматі BSON, з іншого боку, цей недолік більш ніж окупається. в швидкості.

MongoDB написана на C ++, тому його легко перенести на найрізноманітніші платформи. MongoDB можна розгорнути на платформах Windows, Linux, MacOS, Solaris. Ви також можете завантажити вихідний код і скомпілювати MongoDB самостійно, але рекомендується використовувати бібліотеки з офіційного сайту.

Якщо реляційні бази даних зберігають рядки, MongoDB зберігає документи. На відміну від рядків, документи можуть зберігати інформацію, яка має складну структуру. Документ можна розглядати як сховище ключів і значень.[29]

Ключ являє собою просту мітку, з якою пов'язана певна частина даних.

Однак, за всіх відмінностей, є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційній СУБД існує таке поняття, як первинний ключ. Ця концепція описує певний стовпець з унікальними значеннями. MongoDB має унікальний ідентифікатор для кожного документа під назвою _id. І якщо ви явно не вкажете його значення, MongoDB автоматично згенерує значення для нього. Кожен ключ пов’язаний з певним значенням. Але тут також потрібно враховувати одну особливість: якщо в реляційних базах даних є чітко визначена структура, де є поля, і якщо яке поле не має значення, її (залежно від налаштувань конкретної бази даних) можна призначити значення NULL. MongoDB відрізняється.[31]

Якщо будь-який ключ не пробує значення, то цей ключ просто опускається в документі і не використовується. Тоді як у традиційному світі SQL є таблиці, у світі MongoDB є колекції. І якщо в реляційних базах даних таблиці зберігають один і той же тип жорстко структурованих об’єктів, то колекція може містити різноманітні об’єкти з іншою структурою та іншим набором властивостей.

Система зберігання в MongoDB являє собою набір реплік. Цей набір має первинний вузол, а також може мати набір вторинних вузлів. Усі вторинні вузли залишаються недоторканими і автоматично оновлюються, коли оновлюється головний вузол. І якщо головний вузол з якоїсь причини виходить з ладу, то один із другорядних вузлів стає основним.

Відсутність жорсткої схеми бази даних і, отже, необхідність найменшої зміни концепції зберігання даних, перетворення цієї схеми значно полегшують роботу з базами даних MongoDB та їх подальше масштабування.[30] Це також економить час розробників. Їм більше не потрібно думати про перезапуск бази даних і витрачати час на створення складних запитів.

Однією з проблем будь-якої системи баз даних є зберігання великих даних. Можна зберігати дані у файлах, використовуючи різні мови програмування. Деякі СУБД пропонують спеціальні типи даних для зберігання двійкових даних у базі даних (наприклад, BLOB в MySQL).

На відміну від реляційної СУБД, MongoDB дозволяє зберігати різні документи з різними наборами даних, однак розмір документа обмежений 16 МБ. Але MongoDB пропонує рішення – спеціальну технологію GridFS, яка дозволяє зберігати дані розміром понад 16 МБ.[31]

Система GridFS складається з двох колекцій. Перша колекція, яка називається файлами, зберігає імена файлів, а також їх метадані, наприклад розмір. Інша колекція, яка називається chunks, зберігає дані файлів невеликими фрагментами, як правило, частинами по 256 КБ.

2.4. Вибір мови програмування та допоміжних засобів

Для розробки клієнтської сторони використовувався JavaScript. Використовувалися також допоміжні мови та інструменти. Розглянемо їх більш детально.

2.4.1. Мова розмітки гіпертекстових документів

HTML (англійська мова розмітки гіпертексту - мова розмітки гіпертекстові документи) - стандартна мова розмітки для веб-сторінок інтернет. Більшість веб-сторінок створюються за допомогою HTML (або XHTML). HTML-документ обробляється браузером і відтворюється екран у звичному для людини вигляді.

HTML є похідним від SGML, успадковуючи визначення тип документа та ідеологія структурної розмітки тексту, хоча HTML є штучною комп'ютерною мовою, це не мова програмування. HTML разом із каскадними таблицями стилів та вбудованими скрипти — це три основні технології створення веб-сторінок. Введення HTML означає:

- створення структурованого документа шляхом позначення структурний склад тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації зі всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення в текст зображень, звуку, відео та інших об'єктів.

2.4.2. Каскадні таблиці стилів CSS

Каскадні таблиці стилів (або скорочено CSS) — це спеціальна мова, яка використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовується для візуального представлення сторінок, написаних у HTML і XHTML, але формат CSS можна застосувати до інших видів документів XML.

Специфікації CSS були створені та розробляються Консорціумом World Wide Web.

CSS має різні рівні та профілі. Наступний рівень CSS спирається на попередні, додаючи нові функції або розширюючи існуючі. Рівні називаються CSS1, CSS2 і CSS3. Профілі - набір правил CSS одного або кількох рівнів, призначених для конкретних типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадний або блоковий макет) замінив табличне розташування веб-сторінок. Основна перевага блочного макета — розділення вмісту сторінки (даних) та її візуального представлення.

2.4.3. Мова програмування JavaScript

JavaScript (JS) — це динамічна, об'єктно-орієнтована мова програмування. Реалізація ECMAScript. Найчастіше він використовується як частина браузера, що забезпечує можливість клієнтського коду (виконаного на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно спілкуватися з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. JavaScript також використовується для програмування на стороні сервера (подібно до мов програмування, таких як Java і C#), розробки ігор, настільних і мобільних додатків, написання сценаріїв у програмному забезпеченні (наприклад, у програмах Adobe CreativeSuite), всередині документів PDF, тощо...

JavaScript класифікується як прототип (підмножина об'єктно-орієнтованої), динамічно типізованої мови сценаріїв. Окрім прототипного JavaScript, він також частково підтримує інші парадигми програмування (імперативні та частково функціональні) та деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам’яттю, імітація прототипу, функціонує як першокласні об’єкти.

Незважаючи на подібність імен, Java і JavaScript є двома різними мовами з різною семантикою, хоча вони мають схожість у стандартних бібліотеках та умовах найменування. Синтаксис обох мов успадковано від мови C, але на семантику та дизайн JavaScript впливають мови Self і Scheme.

2.4.4. Бібліотека React.js

React.js, який зазвичай називають React, — це бібліотека JavaScript з відкритим кодом для створення користувацьких інтерфейсів, яка вирішує проблеми часткового оновлення вмісту веб-сторінки, які виникають при розробці односторінкових програм. Розроблено Facebook, Instagram та спільнотою індивідуальних розробників.[18]

React дозволяє розробникам створювати великі веб-додатки, які споживають дані, які змінюються з часом, не перезавантажуючи сторінку. Його мета – бути швидким, простим, масштабованим. React обробляє лише інтерфейс користувача в програмах. Це відповідає View у шаблоні Model-View-Controller (MVC) і може використовуватися разом з іншими бібліотеками JavaScript або у великих платформах MVC, таких як AngularJS. Його також можна використовувати з React на основі доповнень, щоб подбати про частини створення веб-програм, які не є користувацькими.[16]

2.4.5. Бібліотека Redux

Redux — це призначений контейнер стану для додатків JavaScript. Це дозволяє створювати програми, які однаково поводяться в різних середовищах (клієнтські, серверні та власні програми), а також легко тестувати. Крім того, він забезпечує чудовий досвід налагодження, наприклад редагування коду в реальному часі в поєднанні з подорожами в часі. Ви можете використовувати redux разом з react або будь-якою іншою бібліотекою представлення. Це крихітна бібліотека (2 КБ, включаючи залежності).[17] Оскільки вимоги до односторінкових javascript-додатків зростають, ми змушені керувати все більшою кількістю станів за допомогою javascript. Ці стани можуть включати відповіді сервера, кешовані дані та дані, згенеровані локально, але ще не збережені на сервері. Це також стосується станів інтерфейсу користувача, таких як активний маршрут, виділена вкладка, відображення обертання або розбиття на сторінки тощо.

Важко керувати станами, які постійно змінюються. Якщо модель може оновлювати іншу модель, то подання може оновлювати модель, що оновлює іншу модель, що, у свою чергу, може призвести до оновлення іншого подання. У якийсь момент ви більше не знаєте, що відбувається у вашій програмі. Ви більше не можете контролювати, коли, чому і як стан оновлювався. Коли система стає непрозорою і недетермінованою, стає важко виявити помилки або додати нові функції.

Це досить погано, враховуючи, що нові вимоги стають звичними для інтерфейсної розробки, наприклад, обробка оптимістичних оновлень, відтворення на сервері, отримання даних перед переходом на сторінку тощо. Як фронтенд-розробники, ми намагаємося впоратися зі складністю, з якою ніколи раніше не мали справи, і тому неминуче виникає питання: чи пора здаватися?

Ця складність виникає через те, що ми змішуємо два поняття, які дуже важко зрозуміти: мутація та асинхронність. Я називаю їх Mentos і Circles. Обидва ці поняття можуть бути чудовими самі по собі, але разом вони перетворюються на безлад. Бібліотеки, такі як react, намагаються вирішити цю проблему на рівні представлення, видаляючи асинхронність і пряме маніпулювання домом. Однак react залишає контроль над станом даних вам. Ось тут на допомогу приходить redux. Наслідуючи flux, cqrs і джерело подій, redux намагається зробити зміни стану передбачуваними, вводячи деякі обмеження щодо того, як і коли можуть відбуватися оновлення. Ці обмеження знаходять своє відображення в трьох китах налагодження зменшення помилок, таких як редагування коду в реальному часі в поєднанні з подорожами в часі.

Єдине джерело істини. Стан всієї вашої програми зберігається в дереві об’єктів в одному сховищі.

Це полегшує створення універсальних програм. Стан на сервері можна серіалізувати та надіслати клієнту без додаткових зусиль. Це полегшує налагодження програми, коли ми маємо справу з одним деревом станів. Ви також можете зберегти стан своєї програми, щоб прискорити процес розробки. А за допомогою єдиного дерева станів ви отримуєте функціональні можливості скасування/повторення з коробки.

Стан лише для читання. Єдиний спосіб змінити стан - застосувати дію - об'єкт, що описує те, що сталося.

Це гарантує, що перегляди або зворотні виклики мережі ніколи не змінюють стан безпосередньо. Оскільки всі зміни централізовані і застосовуються послідовно в строгому порядку, немає необхідності стежити за гонкою держав. Дії — це лише

прості об’єкти, тому їх можна розмістити, серіалізувати, зберегти, а потім відтворити для налагодження чи тестування.

Мутації записуються як чисті функції. Щоб визначити, як дерево станів буде трансформовано за допомогою дій, ви пишете чисті редуктори. Редуктори — це просто чисті функції, які приймають попередній стан і дію і повертають новий стан. Не забудьте повернути новий об’єкт стану замість того, щоб змінювати попередній. Ви можете почати з одного редуктора, але пізніше, коли ваша програма буде рости, ви можете розділити його на менші редуктори, які керують окремими частинами дерева станів. Оскільки редуктори — це лише функції, ви можете контролювати порядок їх виклику, надсилати додаткові дані або навіть писати повторно використовувані редуктори для звичайних завдань, таких як розбиття на сторінки.

2.4.6. Транспайлер Babel.js

Babel.JS — це транспілер, який переписує код ES-2015 у попередній стандарт ES5.

Він складається з двох частин. Фактично транспілер, який переписує код.

Поліфіл, метод додавання Array.from, String.prototype.repeat та інші.

На сторінці можна поекспериментувати з транспілятором: зліва ви вводите код в ES-2015, а праворуч з'являється результат його перетворення в ES5.

Зазвичай Babel.JS працює на сервері як частина системи збору коду JS (наприклад, webpack або бранч) і автоматично переписує весь код у ES5.

Налаштувати таке перетворення тривіально, єдине, що потрібно піднімати саму систему збірки, а додати до неї Babel легко, для будь-якого з них є плагіни.

Якщо ви хочете «пограти», то можете використовувати браузерну версію Babel.[18]

2.4.7. Утиліта Webpack

Webpack — це утиліта для створення пакетів та оптимізації модулів JavaScript та інших інтерфейсних ресурсів. Якщо ви раніше використовували RequireJS або Browserify, є ймовірність, що вам сподобається webpack так само, як і мені. При розробці великого проекту дуже часто виникає необхідність розбити код на окремі модулі, які будуть взаємодіяти один з одним. Сьогодні для цього існує два підходи: AMT і CommonJS. Обидва вони дозволяють розробляти ізольовані модулі, не думати про порядок їх завантаження, зібрати всі наші модулі в один js-файл і безпечно підключати сторонні бібліотеки.[16]

2.4.8. Методологія BEM

BEM (Block, Element, Modifier) — це компонентний підхід до веб-розробки. Він заснований на принципі поділу інтерфейсу на незалежні блоки. Це дозволяє швидко та легко розробляти інтерфейси будь-якої складності та повторно використовувати існуючий код, уникаючи «Копіювати-Вставити».

Блок — це функціонально незалежний компонент сторінки, який можна використовувати повторно. У HTML блоки представлені атрибутом class.

Блоки можуть бути вкладені в інші блоки. Наприклад, головний блок може включати логотип, форму пошуку та блок авторизації (рис. 2.5.).

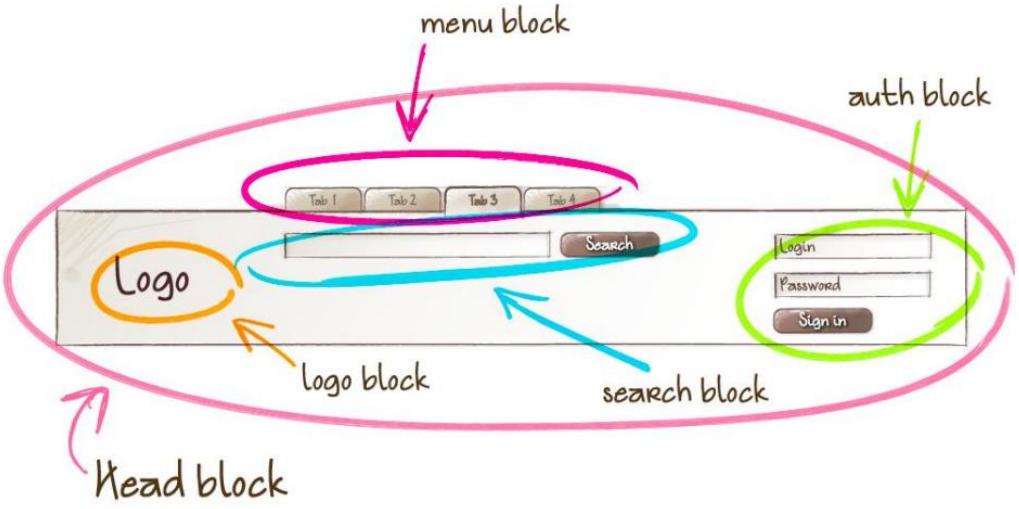


Рисунок 2.5. Структура BEM

Блоки можна переміщувати в межах однієї сторінки, різних сторінок або проектів. Незалежна реалізація блоку дозволяє змінювати положення на сторінці та забезпечує коректну роботу та зовнішній вигляд.

Так, наприклад, логотип і форму авторизації можна замінити місцями. У цьому випадку не потрібно вносити зміни в код CSS або JavaScript блоку (рис. 2.6.).

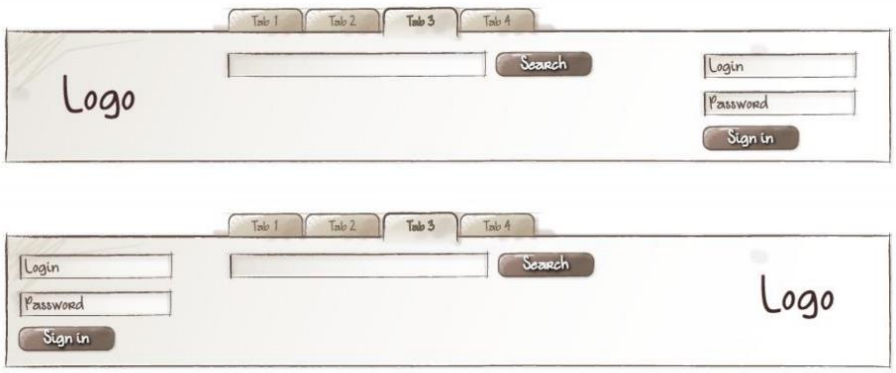


Рисунок 2.6. Реструктуризація в методології BEM

Інтерфейс може одночасно вміщувати декілька екземплярів одного і того ж блоку (рис. 2.7)

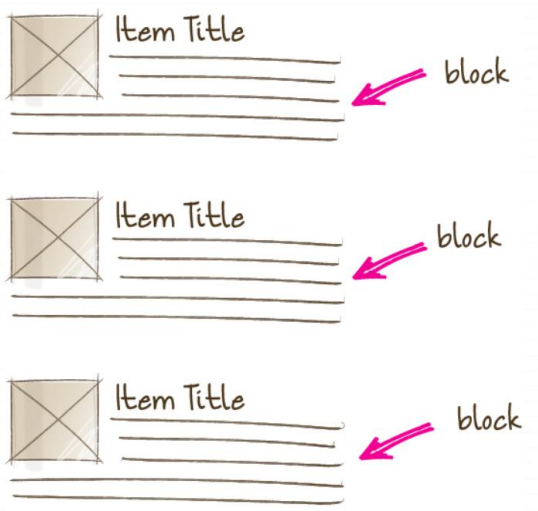


Рисунок 2.7. Приклад декількох екземплярів одного блоку в методології BEM

Елемент – це основна частина блоку, яка не повинна використовуватись поза ним.

До прикладу, один із пунктів не використовується поза контекстом блоку меню, це означає, що цей пункт є елементом (рис 2.8.)

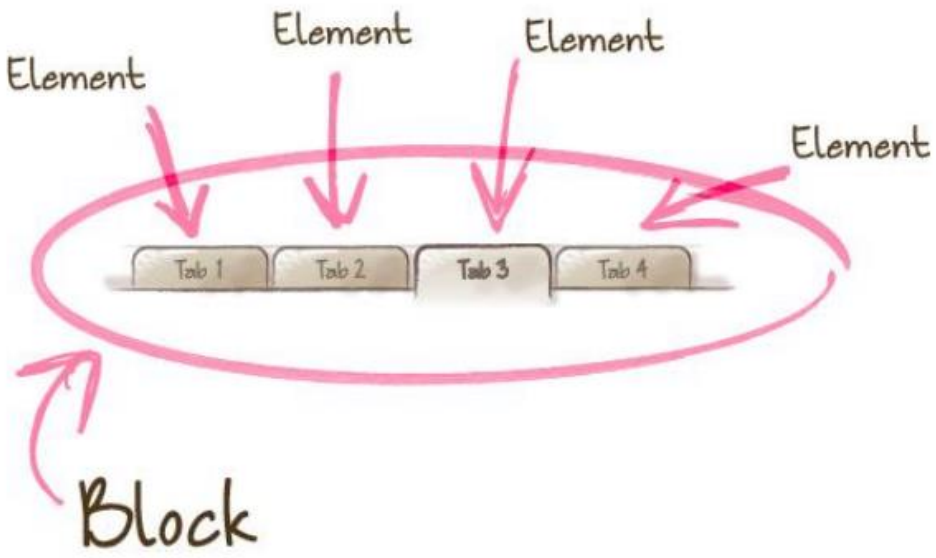


Рисунок 2.8. Елемент в методології BEM

Модифікатор — це сутність BEM, яка визначає зовнішній вигляд, стан і поведінку блоку або елемента.

Використання модифікаторів необов'язкове. За своєю суттю модифікатори схожі на атрибути HTML. Один і той же блок виглядає по-різному через використання модифікатора. Наприклад, вигляд блоку меню (меню) може змінюватися залежно від застосованого модифікатора (рис. 2.9.).

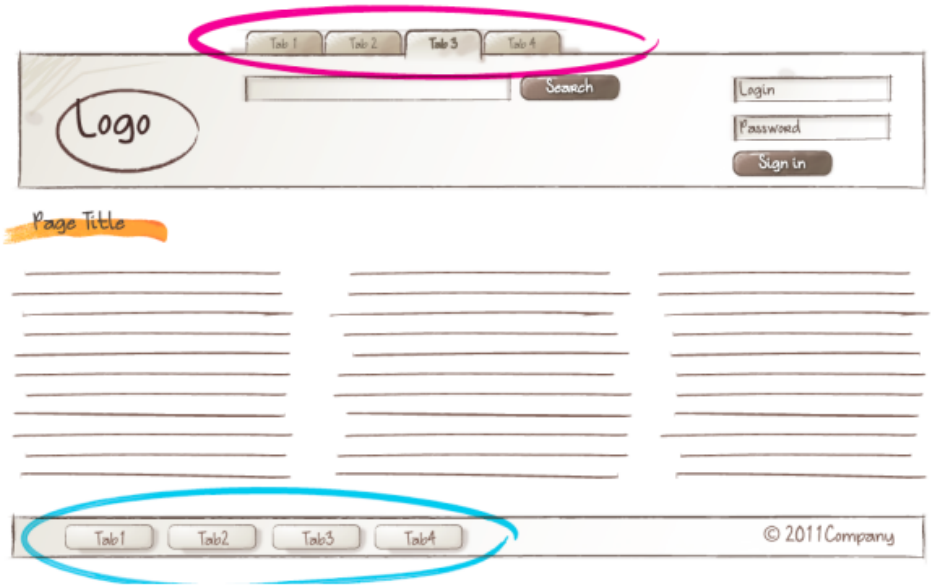


Рисунок 2.9. Модифікатори в методології BEM

Модифікатори можна змінювати як під час роботи блоку (наприклад, як реакція на DOM-події блоку), так і за запитом інших блоків.

Наприклад, коли ви натискаєте кнопку «Увійти» (подія кліку DOM), у разі неправильно заповнених полів «логін» або «пароль», установіть модифікатор (видимий) на прихований блок повідомлень про помилки.

Перевизначення блоку означає зміну реалізації блоку шляхом додавання до нього нових функцій на іншому рівні.

Остаточна реалізація блоку може бути розділена на різні рівні перевизначення. Кожен наступний рівень додає або перекриває вихідну реалізацію блоку. Кінцевий результат збирається за технологіями реалізації окремих блоків з усіх рівнів перевизначення послідовно в заданому порядку

2.4.9. Скриптова метамова SASS

Sass (Syntacically Awesome Stylesheets) — метамова сценаріїв, яка компілюється у звичайні стилі CSS. Якщо ви знайомі з CSS + HTML, ви зможете зрозуміти це з SASS за кілька днів. Кожен, хто стикається з CSS розміром більше 500 рядків, відчуває головний біль, як його спростити. На жаль, з моменту розробки стандартів каскадних стилів їх структура кардинально не змінилася. Вимоги до верстки, кому я буду брехати, значно ускладнилися. Якщо колись 50-70 рядків стилів могли спроектувати простий сайт, то сьогодні цього обсягу вистачає лише для заголовка.

Розширення для файлів SASS може бути .sass і .scss, залежно від вибраного синтаксису. Однак жодного з них він не розуміє, тому для взаєморозуміння потрібно скористатися компілятором. Його робота полягає в тому, щоб перевести SASS у розбірливий, класичний CSS, розпізнаний будь-яким браузером. Навіть шостий Провідник, запряжений конем.

Роль компілятора може виконувати сервер js або програма, встановлена на вашій робочій машині, і відстежує зміни в робочих файлах.

Мова має два основних «обличчя»: SASS і новіший SCSS. Відмінності між ними невеликі, але порушення правил синтаксису завадить компіляції файлу.

У синтаксисі SASS немає фігурних дужок, вкладення елементів у ньому реалізовано за допомогою відступів, а правила стилю обов'язково розділяються новими рядками.

Незалежно від синтаксису, SCSS зворотно сумісний із CSS. Тобто будь-який CSS обов'язково буде дійсним кодом SCSS.

Спочатку я розповім вам в двох словах, а потім більш детально. Мені завжди не вистачало змінних у CSS, і я заважав хакерству між браузерами. У наведених нижче прикладах ви побачите, що SASS блискуче вирішує ці дві проблеми.

Sass дозволяє призначати змінні, і це одна з його ключових переваг. Змінна, за аналогією з php, починається зі знака долара ($), значення призначаються за допомогою двокрапки. Змінні в Sass можна розділити на 4 типи:

- число (int);
- струна (струна);
- логічний тип (так / ні, логічний);
- кольори (ім'я, імена).

2.5. Вибір серверу

NodeJS часто використовується як сервер в односторінкових програмах. Node або Node.js — це програмна платформа, заснована на механізмі V8 (перекладає JavaScript у машинний код), який перетворює JavaScript із вузькоспеціалізованої мови в мову загального призначення. Node.js додає в JavaScript можливість взаємодіяти з пристроями вводу-виводу через свій API (написаний на C++), підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них з коду JavaScript.[8] Node.js використовується в основному на сервері, виконуючи роль веб-сервера, але можлива розробка на Node.js і віконних додатків на робочому столі (за допомогою NW.js, AppJS або Electron для Linux, Windows і Mac OS) і навіть програми

мікроконтролери (наприклад, tessel і espruino). В основі Node.js лежить кероване подіями та асинхронне (або реактивне) програмування з неблокуючим введенням-виводом.[4]

Node.js добре працює в програмах реального часу, оскільки використовує технологію push через веб-сокети. Ну, як уже було сказано, після 20 років використання вищезгаданої парадигми з'явилися такі двонаправлені програми, де комунікація може бути ініційована як клієнтом, так і сервером, а потім перейти до вільного обміну даними. Ця технологія різко контрастує з типовою парадигмою веб-оглядів, де спілкування завжди ініціюється клієнтом. Крім того, вся технологія заснована на відкритому веб-стеку (HTML, CSS і JS), що працює на стандартному порту 80.

Багато людей заперечують, що все це було у нас не один рік – у вигляді Flash і Java-аплетів – але насправді це були лише пісочниці, які сприймали Інтернет як транспортний протокол для доставки даних клієнта. Крім того, вони працювали ізольовано і часто працювали на нестандартних портах, які могли потребувати додаткових прав доступу тощо.

Node.js, незважаючи на всі свої переваги, наразі відіграє ключову роль у технологічному стеку багатьох відомих компаній, які безпосередньо залежать від унікальних властивостей Node.

Основна ідея Node.js полягає в тому, щоб використовувати неблокуючий, керований подіями ввод-вивод, щоб залишатися замисленим і ефективним під час роботи з додатками з інтенсивним використанням даних у реальному часі, що працюють на розподілених пристроях.[8]

Насправді це означає, що Node.js не є універсальною платформою, яка буде домінувати у світі веб-розробки. Навпаки, це платформа для вирішення строго визначених завдань. Це абсолютно необхідно розуміти. Звичайно, ви не повинні використовувати Node.js для операцій з інтенсивним процесором, більше того, використання Node.js для складних обчислень фактично зведе нанівець усі його переваги. Node.js дійсно хороший для створення швидких, масштабованих мережевих додатків, оскільки він може обробляти величезну кількість з'єднань

одночасно з великою пропускною здатністю, що дорівнює високій масштабованості.

Тонкощі Node.js під капотом досить цікаві. У порівнянні з традиційними версіями веб-сервісів, де кожне з'єднання (запит) породжує новий потік, завантажуючи оперативну пам'ять системи і, зрештою, безслідно розбираючи цю пам'ять, Node.js набагато економніший: він працює в одному потоці. , викликаний за допомогою неблокуючого введення-виводу, дозволяє підтримувати десятки тисяч одночасних з'єднань (існуючих у циклі подій).

Простий розрахунок: припустимо, що кожен потік потенційно може вимагати 2 МБ пам'яті і працює в системі з 8 ГБ оперативної пам'яті. У цьому випадку теоретично можна очікувати максимум 4000 одночасних підключень плюс вартість перемикання контексту між потоками.[10]

Це сценарій, з яким ми стикаємося під час використання традиційних веб-сервісів. Node.js, уникаючи всього цього, може масштабуватися до понад мільйона одночасних підключень (як експеримент для підтвердження концепції).

Звичайно, виникає питання про розподіл єдиного потоку між усіма клієнтськими запитами, це основна «пастка» при написанні додатків за допомогою Node.js. По-перше, складні обчислення можуть заблокувати один потік Node.js, що може спричинити проблеми для всіх клієнтів (докладніше про це нижче), оскільки вхідні запити будуть заблоковані, доки запитане обчислення не буде завершено. По-друге, розробники повинні бути дуже обережними, щоб не дозволити виняткам потрапити в базовий (найвищий) цикл подій Node.js, інакше екземпляр Node.js завершиться (насправді вся програма вийде з ладу).[8]

Щоб уникнути виключення винятків до самої поверхні, застосовується такий трюк: помилки передаються тому, хто викликає, як параметри зворотного виклику (а не викидаються, як в інших середовищах). У випадку, якщо необроблений виняток пролітає і випливає, існує багато парадигм та інструментів, які дозволяють відстежувати процес Node і виконувати необхідне відновлення екземпляра, що збився (хоча відновити сеанси користувача буде неможливо). Найпоширенішими є модуль Forever або робота із зовнішніми системними інструментами upstart і monit.

2.5.1. NPM Менеджер пакетів Node

Обговорюючи Node.js, просто необхідно згадати про вбудовану підтримку керування пакетами, яка використовує інструмент NPM, який за замовчуванням постачається з будь-якою інсталяцією Node.js. Ідея модулів NPM багато в чому схожа на Ruby Gems: це набір компонентів з відкритим кодом, багаторазового використання, які легко встановити через онлайн-сховище; вони підтримують керування версіями та залежностями.

Повний список запакованих модулів можна знайти на сайті NPM npmjs.org, а також доступний за допомогою інструмента CLI NPM, який автоматично встановлюється разом із Node.js. Екосистема модулів повністю відкрита, будь-хто може опублікувати там свій власний модуль, який з'явиться у списку репозиторіїв NPM. Короткий вступ до NPM (трохи застарілий, але все ще актуальний) на сайті howtonode.org/introduction-to-npm.[9]

Деякі з найбільш популярних сучасних модулів NPM:

- express.js, фреймворк веб-розробки для Node.js, написаний у дусі Сінатри, де-факто є стандартом для більшості існуючих сьогодні додатків Node.js;
- connect: Connect — це розширюваний фреймворк, який працює з протоколами підключення до бази даних;
- node.js як HTTP-сервер, що забезпечує набір високопродуктивних «плагінів», спільно відомих як проміжне програмне забезпечення; служить основою для Express;
- socket.io і sockjs - серверна сторона двох найпоширеніших компонентів веб-сокета на сьогоднішній день;
- Jade - один з популярних інструментів шаблонування, написаний в дусі HAML, за замовчуванням використовується в Express.js;
- mongo і mongojs - обгортки MongoDB, які надають API для баз даних об'єктів MongoDB в Node.js;
- redis - клієнтська бібліотека Redis;

- coffee-script - компілятор CoffeeScript, який дозволяє розробникам писати програми на Node.js за допомогою Coffee;
- підкреслення (lodash, lazy) - Найпопулярніша допоміжна бібліотека JavaScript, запакована для використання з Node.js, а також дві подібні бібліотеки, які забезпечують підвищену продуктивність, оскільки реалізовані дещо іншим чином;
- forever - Мабуть, найпоширеніша утиліта для забезпечення безперебійного виконання скрипту на заданому вузлі. Підтримує роботу процесу Node.js у разі будь-яких неочікуваних збоїв;

Ось що відбувається коли один із клієнтів надсилає повідомлення:

- браузер отримує натискання кнопки «Відправити» за допомогою обробника JavaScript, вибирає значення з поля введення (тобто текст повідомлення) і видає повідомлення веб-сокета за допомогою клієнта веб-сокета, підключеного до нашого сервера (це клієнт ініціалізується разом із веб-сторінкою);
- Компонент сервера підключення websocket отримує повідомлення та передає його всім іншим підключеним клієнтам.
- усі клієнти отримують нове push-повідомлення за допомогою компонента веб-сокета, що працює на веб-сторінці. Потім вони захоплюють вміст публікації та оновлюють веб-сторінку, додаючи нову публікацію на форум;

2.5.2. Черги вводу

Якщо ви отримуєте велику кількість одночасних даних, база даних може стати вузьким місцем. Як показано вище, Node.js з легкістю обробляє одночасні з’єднання як такі. Але оскільки доступ до бази даних є операцією блокування (в даному випадку), у нас виникають проблеми. Рішення полягає в тому, щоб зафіксувати поведінку клієнта до того, як дані фактично будуть записані в базу даних.

При такому підході чутливість системи підтримується під високим навантаженням, що особливо корисно, якщо клієнту не було потрібно підтвердження того, що запис даних пройшов успішно. Типовими прикладами є: реєстрація або запис даних відстеження користувачів, які обробляються пакетами та не використовуються в подальшому; операції, результат яких має відображатися миттєво (наприклад, оновлення кількості «лайків» у Facebook), де прийнятна узгодженість, зрештою, використовується так часто у світі NoSQL.

Дані поміщаються в чергу за допомогою спеціальної інфраструктури кешування та черги повідомлень (наприклад, RabbitMQ, ZeroMQ) і перевар юються за допомогою окремого процесу пакетного запису бази даних або спеціальних служб інтерфейсу бази даних із інтенсивними обчисленнями. Подібну поведінку можна реалізувати за допомогою інших мов/фреймворків, але на іншому обладнанні і не з такою високою та стабільною пропускною здатністю.

На більш звичайних веб-платформах HTTP-запити та відповіді розглядаються як ізольовані дії; але насправді це потоки. На цьому етапі ви можете використовувати Node.js для створення деяких цікавих функцій. Наприклад, ми можемо обробляти файли під час їх завантаження, оскільки дані надходять у потоці, і ми можемо працювати з ними онлайн. Це можна зробити, наприклад, при кодуванні відео чи аудіо в режимі реального часу, а також при встановленні проксі між різними джерелами даних (докладніше дивіться в наступному розділі).

Node.js цілком можна використовувати як проксі-сервер, і в цьому випадку він може обробляти велику кількість одночасних підключень неблокуючим способом. Це особливо корисно при посередництві між різними службами, які мають різний час відповіді, або при зборі даних з багатьох джерел.

Наприклад, давайте розглянемо серверний додаток, який спілкується із сторонніми ресурсами, збирає інформацію з різних джерел або зберігає такі ресурси, як зображення та відео, які потім відкриваються стороннім хмарним сервісам.

Хоча виділені проксі-сервери існують, натомість Node зручно використовувати, особливо якщо інфраструктурний проксі-сервер не існує або

якщо вам потрібне рішення для локальної розробки. Тут я маю на увазі, що ви можете створити клієнтську програму, де буде використовуватися сервер розробки Node.js, де ми будемо зберігати ресурси та робити проксі/заглушки для запитів API, а в реальних умовах такі взаємодії вже здійснюватимуться за допомогою виділеного проксі. обслуговування. (nginx, HAProxy тощо).

Тепер поговоримо про інфраструктурні аспекти. Скажімо, у вас є постачальник SaaS, який хоче запропонувати користувачам сторінку відстеження послуг (наприклад, сторінку стану GitHub). За допомогою циклу подій Node.js ви можете створити потужний веб-інтерфейс, де стани служби асинхронно перевіряються в режимі реального часу, а дані надсилаються клієнту через веб-сокети.

Ця технологія дозволяє звітувати про статуси як внутрішніх (внутрішньокорпоративних), так і державних служб в режимі реального часу. Давайте розглянемо цю ідею трохи далі і спробуємо уявити центр мережевих операцій (NOC), який контролює роботу оператора, хмарного провайдера/хостингу чи якоїсь фінансової установи. Усе це працює на відкритому веб-стеку на основі Node.js і websockets, а не Java та/або аплетів Java.

Коли справа доходить до складних обчислень, Node.js залишає бажати кращого. Звичайно, ви не збираєтеся програмувати на сервері Node для обчислень Фібоначчі. По суті, будь-яка обчислювальна операція, яка створює велике навантаження на процесор, знецінить приріст пропускної здатності, якого Node досягає за допомогою керованого подіями, неблокування вводу-виводу. Справа в тому, що будь-які вхідні запити будуть заблоковані, поки один потік зайнятий переробкою. числа.

Як зазначалося вище, Node.js є однопоточним і використовує лише одне ядро процесора. Може знадобитися реалізувати паралельність на багатоядерному сервері; для цього команда ядра Node вже готує спеціальний кластерний модуль. Крім того, ви можете легко запустити кілька екземплярів сервера Node.js через зворотний проксі-сервер за допомогою nginx.

Кластеризація зберігає можливість перевантажувати всі складні обчислення на фонові процеси, що виконуються в більш підходящому середовищі, і дозволяє спілкуватися між ними через сервер черги повідомлень, такий як RabbitMQ.

Хоча фонова обробка спочатку може виконуватися на одному сервері, цей підхід забезпечує дуже високу масштабованість. Такі послуги фонової обробки можна легко розподілити на окремі виробничі сервери без необхідності налаштовувати навантаження на «фронт-енд» веб-сервери.

Звичайно, такий підхід доречний і на інших платформах, але у випадку з Node.js купується та величезна пропускна здатність, про яку ми говорили вище, оскільки кожен запит — це невелике завдання, яке обробляється дуже швидко та ефективно.

2.6. Аналіз роботи прикладного програмного забезпечення з базами даних

Дане програмне рішення являється універсальним програмним забезпеченням, яке потребує виключно часткового втручання користувача у внутрішні процеси роботи програми. Звичайно, користувач має повний контроль над усіма процесами, якщо потрібно відмінити дію, в обробці даних, або якимось іншим чином змінити дані клієнтів готелю, також користувач має можливість власноруч додавати певні зміни у вигляді створення нового запису на реєстрацію, виселення, або поселення клієнта готелю. Але перш за все, програма орієнтована на автоматизовану роботу, тобто усі процеси у програмні повинні виконуватись автоматично без впливу користувача, це дозволить системі працювати завжди правильно та без збоїв. Першочергово таке правило повинно відноситись до бази даних програмного продукту, оскільки жодна система не може функціонувати з повністю доступними базами, в які можна безперешкодно вносити будь які зміни.

Для зберігання та редагування даних, програма повинна з’єднуватись з базою даних, безпечніше за все це можна зробити за допомогою виведення підключення до бази даних на захищений сервер, який матиме єдині права на читання, зміну, запис та видалення даних з бази даних

Додаток – це програма або комплекс програм, які використовують базу даних і забезпечують автоматизацію обробки інформації в певній предметній області. Додатки можна створювати як в середовищі систем управління базами даних, так і без них за допомогою системи програмування, наприклад, C++ або Java, використовуючи засоби доступу до баз даних.

Для роботи з базою даних у багатьох випадках можна використовувати лише інструменти систем управління базами даних, наприклад, створені запити та звіти.

Перевага використання баз даних в інформаційних системах полягає в тому, щоб забезпечити незалежність даних від додатків (рис). Немає необхідності розглядати питання розміщення даних у пам'яті та способів доступу до них [3].

Рисунок 2.10. – взаємодія користувача з базою даних

2.7. Дослідження функціональних можливостей сучасних програмних рішень

З метою підвищення швидкості та ефективності розробки програмного забезпечення можна використовувати готові фреймворки та програми, які значно полегшують роботу, автоматично виконуючи необхідні для розробника дії.

2.7.1. MySQL workbench

MySQL Workbench — це єдиний візуальний інструмент для архітекторів, розробників та адміністраторів баз даних. MySQL Workbench пропонує моделювання даних, розробку SQL та комплексні інструменти адміністрування для конфігурації сервера на адміністративній панелі (Рисунок 2.11.), адміністрування користувачів, створення резервних копій тощо. MySQL Workbench доступний для більшості операційних систем [3].

MySQL Workbench дозволяє адміністратору бази даних, розробнику або архітектору даних візуально проектувати, моделювати, створювати та керувати базами даних. Він включає в себе все необхідне розробнику даних для створення складних моделей ER, а також надає ключові функції для складних завдань управління змінами та документування, які зазвичай вимагають від розробника багато часу і зусиль для створення бази даних.

Він пропонує візуальні інструменти для створення, виконання та оптимізації запитів SQL. Редактор SQL забезпечує підсвічування синтаксису, автозаповнення, повторне використання фрагмента SQL та історію виконання SQL. Панель підключень до бази даних дозволяє розробникам легко керувати стандартними підключеннями до бази даних, включаючи MySQL Fabric. Засіб перегляду об’єктів забезпечує миттєвий доступ до схеми та об’єктів бази даних [3].

Також можна надати візуальну консоль для легкого адміністрування середовищ MySQL та кращого огляду баз даних. Розробники та адміністратори баз даних можуть використовувати візуальні інструменти для налаштування серверів, адміністрування користувачів, створення резервних копій і відновлення, перевірки даних аудиту та перегляду стану бази даних.

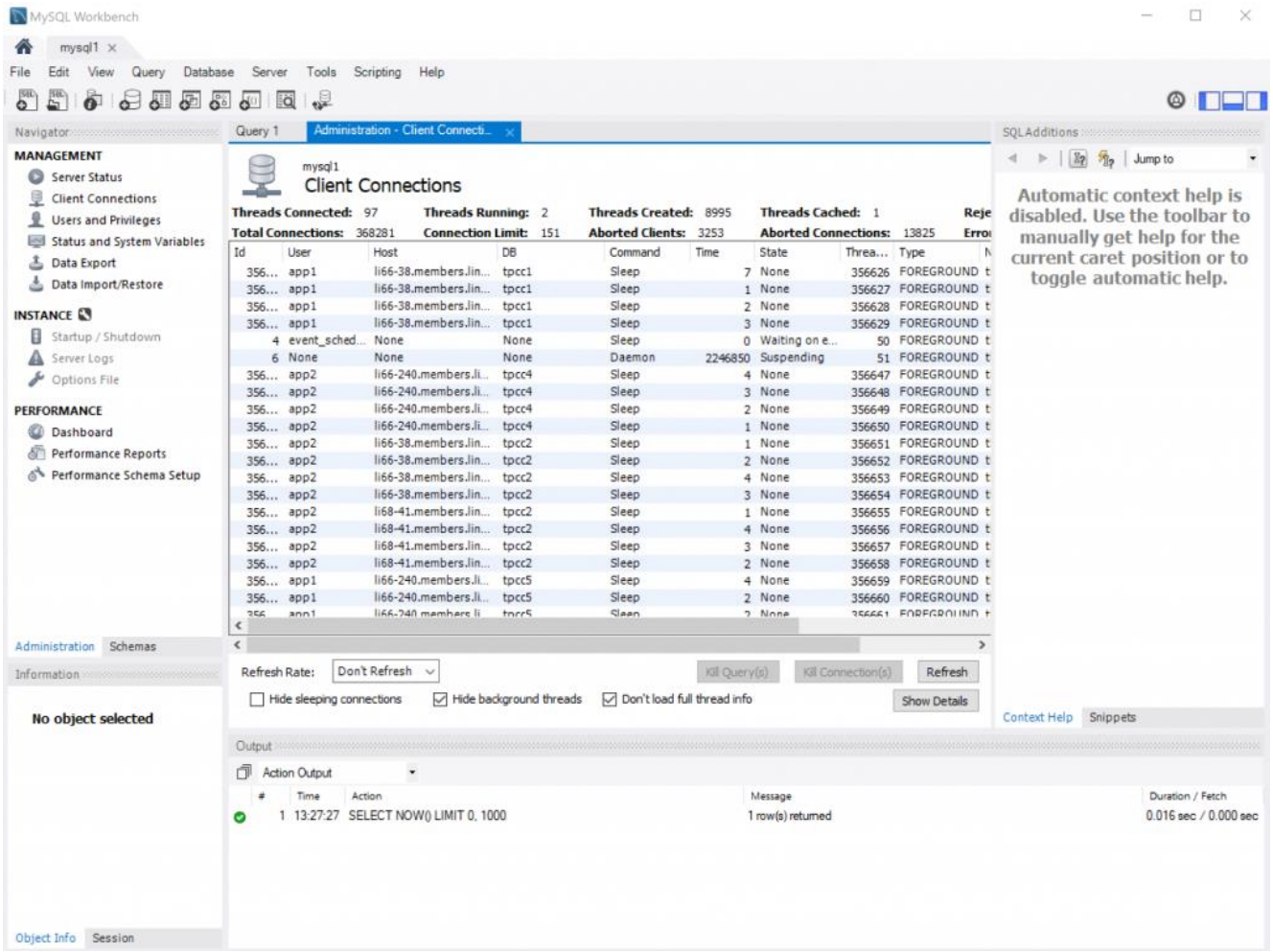


Рисунок 2.11. – Дошка адміністрування в MySQL workbench

2.7.2. DBManager

DBManager — це програма для керування даними. Є вбудована підтримка MySQL (Рисунок 2.12.). Він доступний у двох версіях, тому ви можете вибрати те, яке буде відповідати потребам користувача: стандартне та розширене [4]. Стандартна версія повністю функціональна, і деякі функції навіть відсутні в будь-якому іншому безкоштовному програмному забезпеченні для керування базами даних. DBManager спеціально розроблений для особистого використання, має мінімальний набір функцій, що полегшує роботу з новими користувачами бази даних [4].

Це програмне забезпечення безкоштовне для некомерційного використання. Для комерційного використання ви можете придбати ліцензію, яка надає додаткову функціональність.

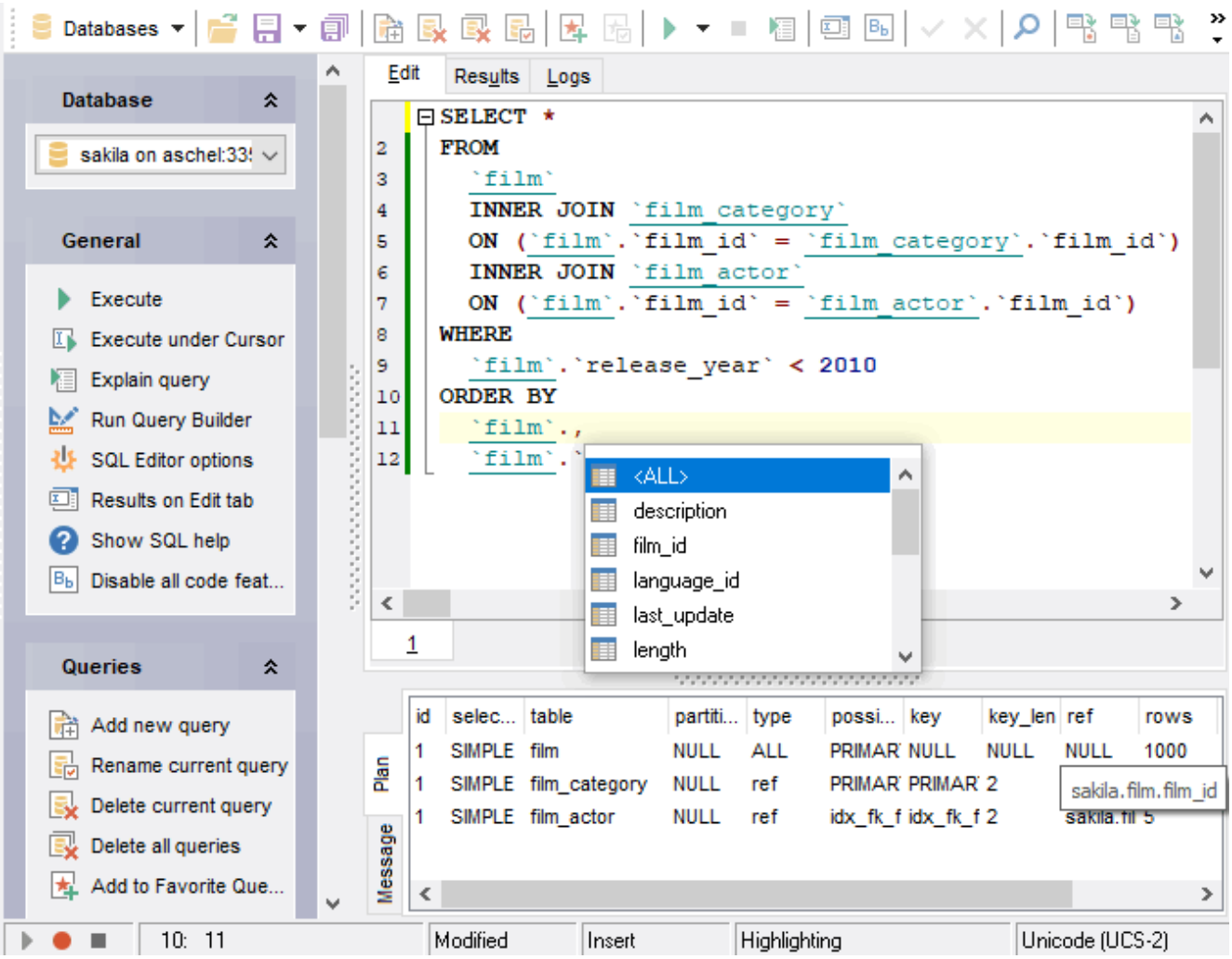


Рисунок 2.12. – підтримка MySQL в DBManager

2.7.3. MyDB Studio

MyDB Studio - безкоштовний інструмент для адміністрування БД MySQL, який дозволяє створювати, редагувати і видаляти записи, таблиці і бази даних (Рисунок – 2.13.). Працює виключно на платформі Windows [5].

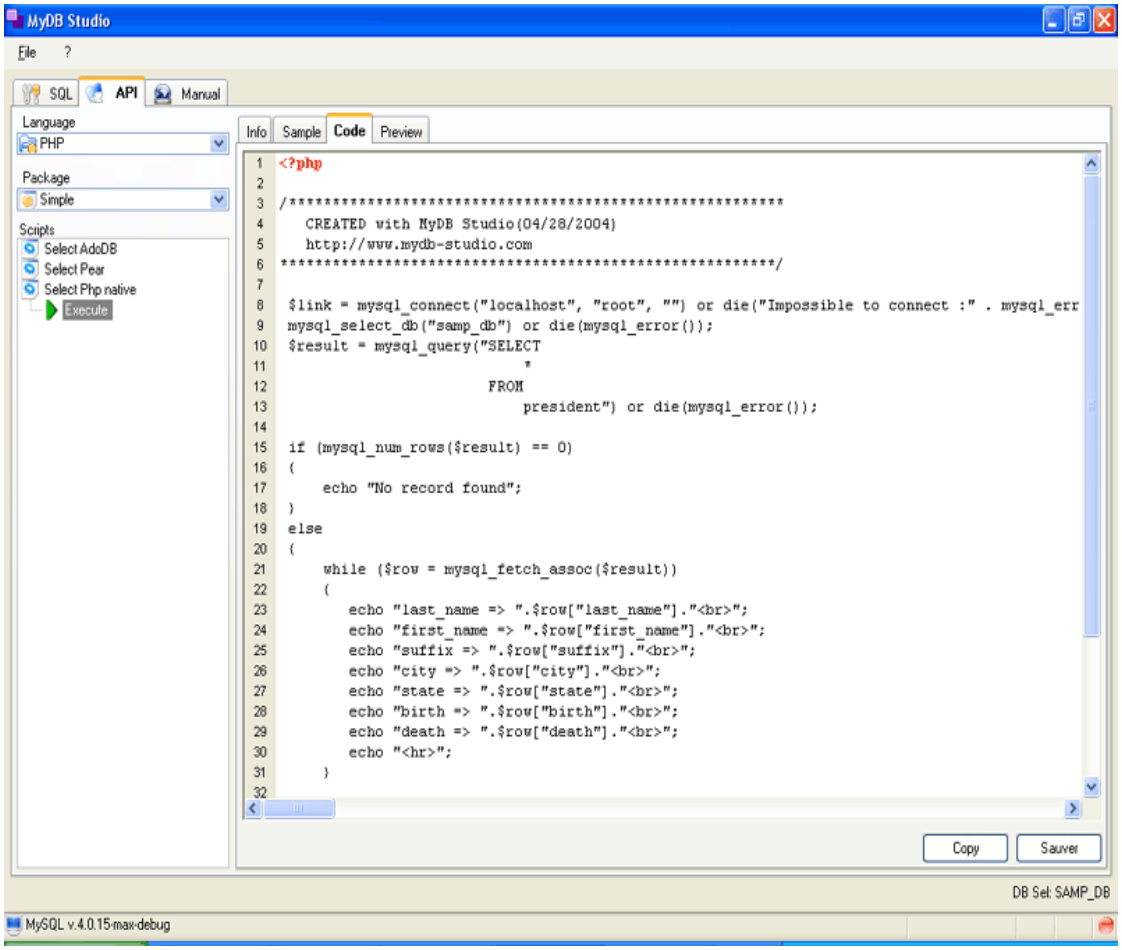


Рисунок 2.13. – приклад роботи MyDB Studio з MySQL

MyDB Studio дозволяє SSH-тунелювання для безпечних з'єднань, дозволяючи підключатися до вашого власного хоста без дозволу комутованих з'єднань. Адміністратор може надати користувачам право перезавантажувати сервер, очищати внутрішні кеші використання MySQL, переглядати / видаляти процес сервера, експортувати серверні змінні в XML, Excel, Word, CSV [5].

2.7.4. SQLyog Ultimate

SQLyog Ultimate дозволяє розробникам баз даних, адміністраторам та архітекторам візуально порівнювати, оптимізувати та документувати схеми.

SQLyog Ultimate містить інструмент для автоматизації та планування синхронізації даних між двома хостами MySQL (рис. 21). За допомогою інтерактивного майстра можна створити файл визначення завдання. Інструмент не

вимагає встановлення на хостах MySQL. Для запуску інструменту можна використовувати будь-який хост [5].

SQLyog дозволяє інтерактивну синхронізацію даних. Використовуючи дисплей, ви можете порівняти вихідні та цільові дані для кожного рядка, щоб вирішити, чи синхронізуватися в певному напрямку.

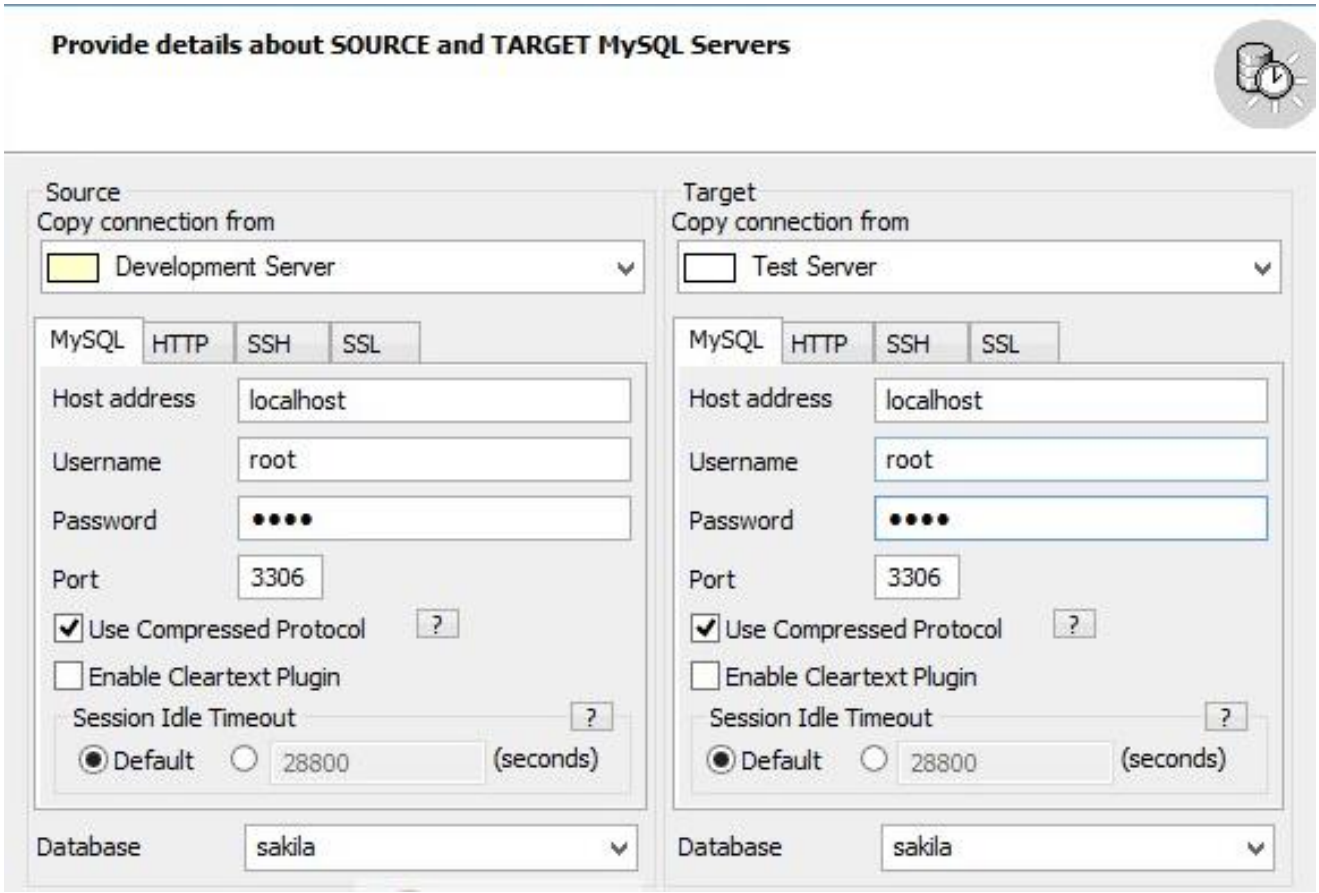


Рисунок 2.14.– приклад синхронізації хостів в SQLyog Ultimate

2.8. Порівняння існуючих рішень

Кожне рішення має свої особливості. Але для об’єктивного порівняння ми розглянемо переваги та недоліки кожної з програм за запропонованими критеріями:

- Модель бази даних у графічному вигляді. Для більш ефективної розробки проекту іноді необхідно використовувати графічну модель бази даних, оскільки це забезпечує більш наочний опис структури даних. Серед розглянутих технологій графічна модель представлена лише в MySQL

Workbench (рис. 22), коли вона відсутня в MyDB Studio, DBManager і SQLyog Ultimate.

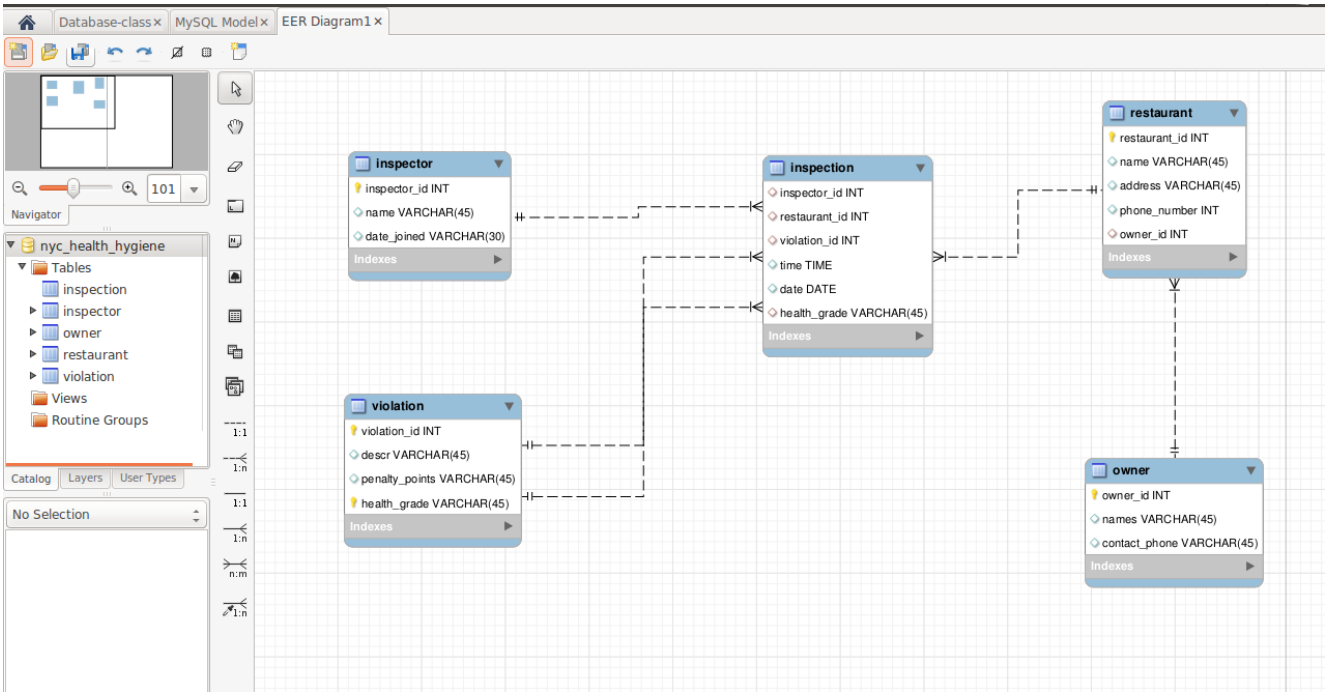


Рисунок 2.15. – модель бази даних у графічному вигляді MySQL workbench

- Мультиплатформенність. Ця властивість дозволяє програмі працювати на кількох платформах. Дозволяє скоротити витрати і час на розробку програмного забезпечення. Ця властивість присутня в таких розглянутих продуктах, як MySQL Workbench і SQLyog. MyDB Studio та DBManager працюють лише на платформі Windows;

- Наявність редактора для SQL-запитів. При відправці даних на сервер бажано отримати відповідь у зручній формі, у вигляді таблиць. З розглянутих програмних продуктів цей редактор доступний лише в програмному забезпеченні MySQL Workbench (рис. 23).

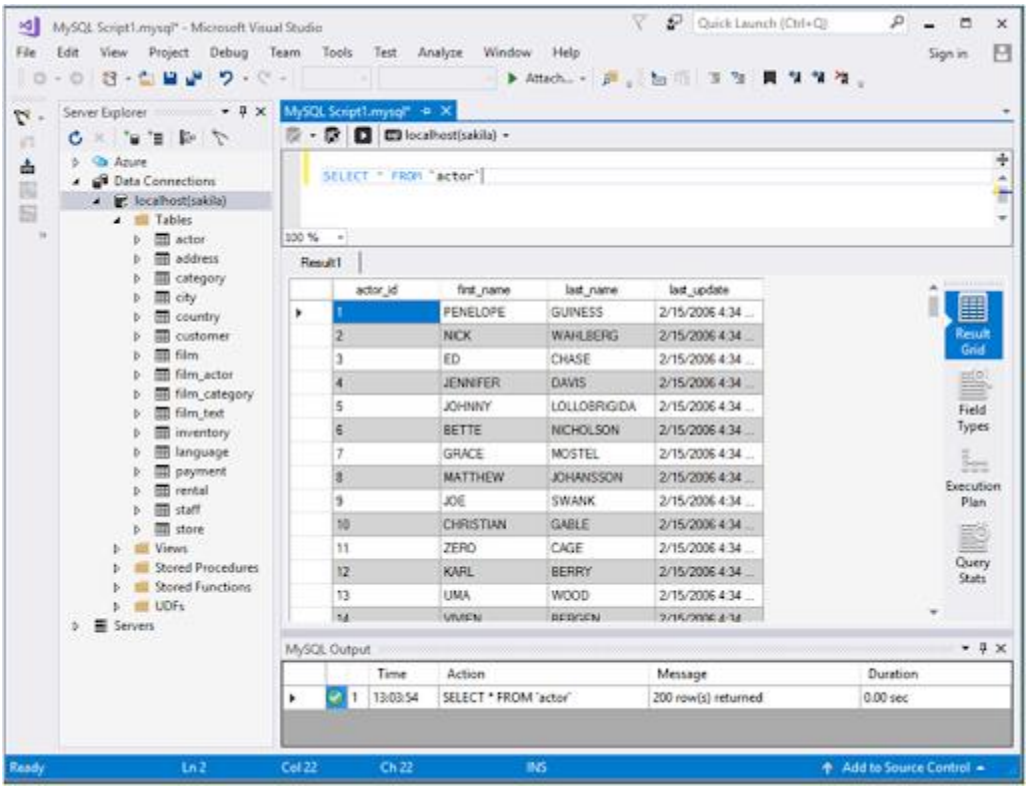


Рисунок 2.16. – Редактор запитів SQL в MySQL Workbench

- функціональний механізм створення зв'язків між полями. Ця функція дозволяє створювати зв’язки між полями з функцією створення таблиці зв’язків. Цей механізм зустрічається в таких програмах, як MySQL Workbench і SQLyog.

- доступність. Цей критерій не менш важливий, оскільки багато програмних продуктів вимагають значних фінансових витрат. Для того щоб товар користувався великим попитом, він стає безкоштовним. MySQL Workbench, SQLyog і MyDB Studio наразі є безкоштовними програмними продуктами.

3. КОДУВАННЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

В розділі проведено аналіз та порівняння існуючих рішень для кодування автоматизованої системи управління адміністративними функціями персоналу. Обґрунтовано вибір мови програмування та її додатків для розробки та тестування готового рішення. Представлено описовий алгоритм керуючої програми написаної на мові програмування JavaScript та наведено опис функцій, розроблених для програмного рішення. Подано огляд розробленої системи, зазначено функції, які система виконує. Проведено аналіз результатів тестування системи.

3.1. Аналіз сучасних рішень для кодування функціоналу та візуальної частини.

Для створення програмного продукту було обрано мови програмування які дозволяють зробити програму крос-платформенною, тобто такою що може виконуватись на різних операційних системах, таки як:

- OS Windows
- Операційні системи на архітектурі Linux
- Mac OS

Такою мовою програмування стала мова JavaScript, оскільки на сьогоднішній день це найбільш популярна, та найбільш підтримувана мова, яка розвивається з кожним днем все більше і більше. Вихідний код цієї мови відкритий, що дозволяє розширювати її і додавати різні модифікації синтаксисів у цій мові, завдяки цьому на її основі було створено велику кількість бібліотек, які залишаються популярними і сьогодні, до прикладу ReactJS.[16]

ReactJS – це бібліотека яка призначена для створення одно-сторінкового додатку, що дозволяє уникнути постійного переходу за посиланнями, а натомість, виконувати перемальовування сторінки видаливши непотрібний вміст та додавши необхідний.[16]

Щоб була можливість використовувати дане програмне рішення як додаток а не як веб сайт, потрібно використовувати обгортку, яка дозволяє запускати веб сайти як додатку, така обгортка називається ElectronJS

ElectronJS – це обгортка яка дозволяє файли html, css та js запускати у вікні операційної системи, таким чином можна без довгої компіляції розробити повноцінний програмний продукт, який буде працювати не гірше ніж скомпільована програма.[2]

Також для легкого створення серверної частини програми потрібно створити Node проект, який дозволить виконати запуск сервера, та підключення до бази даних програми.[10]

NodeJS – це платформа з відкритим кодом яка дозволяє виконувати високопродуктивні мережеві застосунки, написані мовою JavaScript. Платформа має вбудований функціонал, який дозволяє створити підключення по мережевим протоколам, та дуже легко ними користуватись.[8]

3.2. Описовий алгоритм керуючої програми

3.2.1. Опис роботи бази даних з клієнтською частиною програми

Головний алгоритм програми повинен виконувати більшість завдань з обробкою отриманих даних з сервера, оскільки основна частина автоматизації повинна знаходитись на сервері.

На рисунку 3.1 наведено схему послідовності виконання процесів програми, фактичним початком роботи програми являється вибір функції, саме цей етап забезпечує набір запитів до серверу, та підготування нового вікна інтерфейсу.

Після цього повинні проводитись певні маніпуляції з даними, які призначені для зміни цих даних. Задум програми побудований таким чином, що навіть якщо користувач не вносив корективи у дані, зміни до бази даних все одно будуть записані, цими зміненими даними являються log-записи, які дозволять

відслідковувати не тільки користувача який змінював дані, але й користувача який авторизувався.

Наступним етапом є зберігання даних, цей етап передбачає надсилання даних до серверу, який необхідним

Рисунок 3.1. Послідовність процесів роботи бази даних при використанні програми

3.2.2. Опис роботи клієнтської частини

Алгоритм роботи клієнтської частини програми (рис. 3.2.) повинен бути послідовним, та мати якомога меншу кількість розгалужень, це допоможе зробити систему максимально зрозумілою та швидкою.

Перший крок алгоритму роботи клієнтської частини це авторизація. При авторизації з’являється діалогове вікно, в яке необхідно ввести дані користувача, який вже зареєстрований в системі, якщо авторизація була неуспішна можна спробувати авторизуватись ще раз, або вийти з програми.

Другим кроком після авторизації являється вибір функції в програмі з якою необхідно працювати, після чого програма надсилає запит до серверу, який повертає дані необхідні для роботи з функцією програми. Якщо дані не були отримані, або були отримані некоректно, система спробує отримати дані ще раз, після декількох безрезультатних спроб, система сповістить про помилку в отримані даних з серверу, та поверне користувача у головне меню програми.

Якщо дані були отримані коректні, система виводить їх у візуальній частині програми та дає змогу редагувати дані, зміна яких дозволена програмою, якщо данні введені коректні, система збереже їх та відправить їх на сервер для збереження у базі даних, якщо дані були введені некоректні, система відправить запит на повторне отримання актуальних даних щоб не було можливості змінити дані.

Останнім етапом йде розгалуження яке передбачає повторний вибір функції програми, або завершення програми.

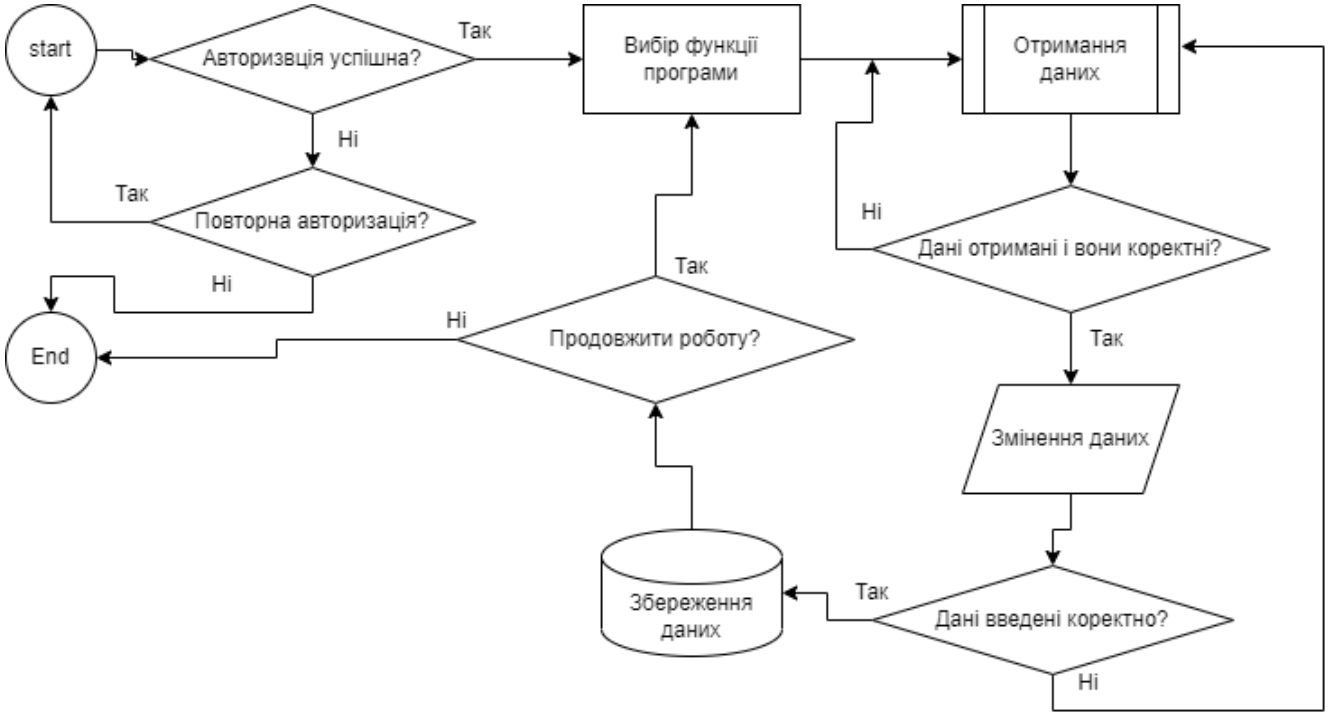


Рисунок 3.2. Алгоритм роботи з клієнтською частиною програми.

3.2.3. Опис роботи сервісу обробки резервування номерів клієнта.

Номера клієнтів резервуються на сайті готелю, система надсилає запит на сервер, та запитує у бази даним чи можливо зарезервувати певний номер на певний період часу, якщо сервер повернув відповідь із дозволом, сервер надсилає дані клієнта до бази даних для резервування номеру, потім система синхронізує дані у базі даних та у застосунку програми (рисунок 3.3.)

Рисунок 3.3. Схема роботи сервісу обробки резервування номерів клієнта.

3.2.4. Опис роботи сервісу обробки резервування столиків у ресторані готелю.

Резервування столиків у ресторані готелю проходить за допомогою сайту готелю. Клієнт обирає дату, час та столик за яким він хоче провести час. При резервуванні з сайту відправляється запит на сервер, який повертає відповідь з дозволом на резервування, якщо сервер дозволяє резервування, він відправляє до бази даних дані клієнта, у випадку якщо дозволу немає, сервер відправляє на сайт відповідь із повідомленням про те що резервація за даним столиком та в даний час недоступна (рисунок 3.4.)

Рисунок 3.4. Схема роботи сервісу обробки резервування столиків у ресторані готелю.

3.3. Розробка системи управління

Для того щоб можна було дуже легко розробляти візуальну частину програми було використано мову програмування JavaScript бібліотеку для цієї мови – ReactJS. Також щоб зробити вікно для робочого стола різних операційних систем було використано бібліотеку для платформи NodeJS – ElectronJS.[1]

JavaScript – це мова програмування яку використовують найпопулярніші браузери, оскільки вона майже повністю складається з об’єктів, що дає змогу дуже легко обробляти дані, та модифікувати їх без жодних обмежень за типами даних.

Також головною перевагою JavaScript над іншими мовами програмування є можливість додавання особистого синтаксису, який можна налаштувати під використання, до прикладу, html розмітки у коді JavaScript. На прикладі коду

основної частини інтерфейсу програми, на рисунку 3.5. продемонстровано переваги використання можливості додавання особистого синтаксису.

```
20  const changeNightMod = (bool) => {
21    console.log(bool)
22    setNightMod(bool, () => {
23      setStyle({
24        css: {
25          backgroundColor: nightMod ? '#424242
26        }
27      }, () => {
28        console.log('bg changed')
29      })
30    })
31  }
32
33  return (
34    <div className="App" style={state.css} >
35      <Header setNightMod={changeNightMod} />
36    </div>
```

Рисунок 3.5. Приклад коду основної частини інтерфейсу програми

На строках з 34 по 36 показано використання html коду у коді JavaScript. Також на рисунках 3.6. та 3.7. продемонстровано результат виведення даного коду із світлим та темним інтерфейсом.

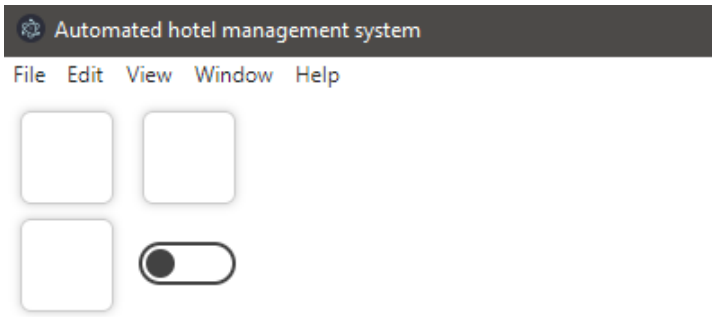


Рисунок 3.6. Результат виведення коду із світлим інтерфейсом

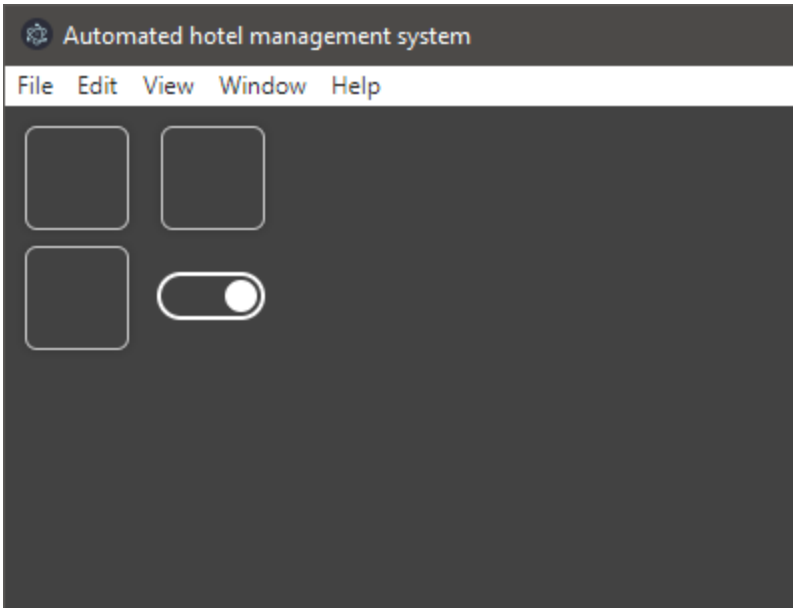


Рисунок 3.7. Результат виведення коду із темним інтерфейсом

Для реалізації вікна робочого стола операційної системи було використано бібліотеку ElectronJS, для того, щоб вікно було виведено використовувався код, для створення вікна інтерфейсу, представленому на рисунку 3.7.[3]

```
function createWindow() {
  // Create the browser window.
  const win = new BrowserWindow({
    width: 800,
    height: 700,
    minWidth: 800,
    minHeight: 700,
    webPreferences: {
      nodeIntegration: true,
      webSecurity: false
    }
  });

  win.loadURL(
    isDev
      ? 'http://localhost:3000'
      : `file://${path.join(__dirname, '../build/index.html')}`
  );
}

app.whenReady().then(createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});
```

Рисунок 3.7. Приклад коду створення вікна робочого стола операційної системи.

В даному випадку створюється об’єкт який отримує як параметри: висоту, ширину, максимальну та мінімальну висоту і ширину, та додаткові конфігурації, за допомогою яких можна керувати характеристиками вікна.

3.4. Огляд розробленої системи

При запуску програми першим з'являється екран автентифікації. Цей екран складається із строк вводу логіну та паролю, також є кнопка увійти (рисунок 3.8.).

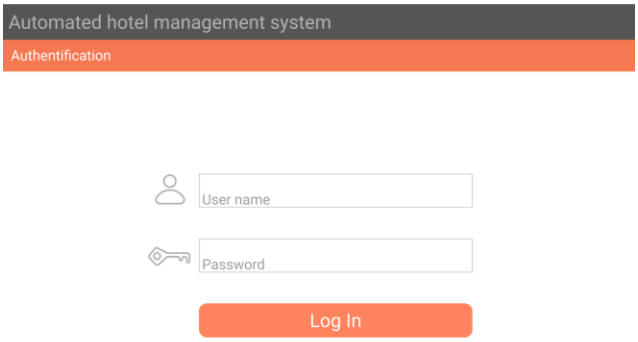


Рисунок 3.8. Екран автентифікації

У випадку якщо автентифікація пройшла не успішно (неправильний логін або пароль) з'являється повідомлення про некоректно введені дані та прохання звернутись до свого системного адміністратора (рисунок 3.9)

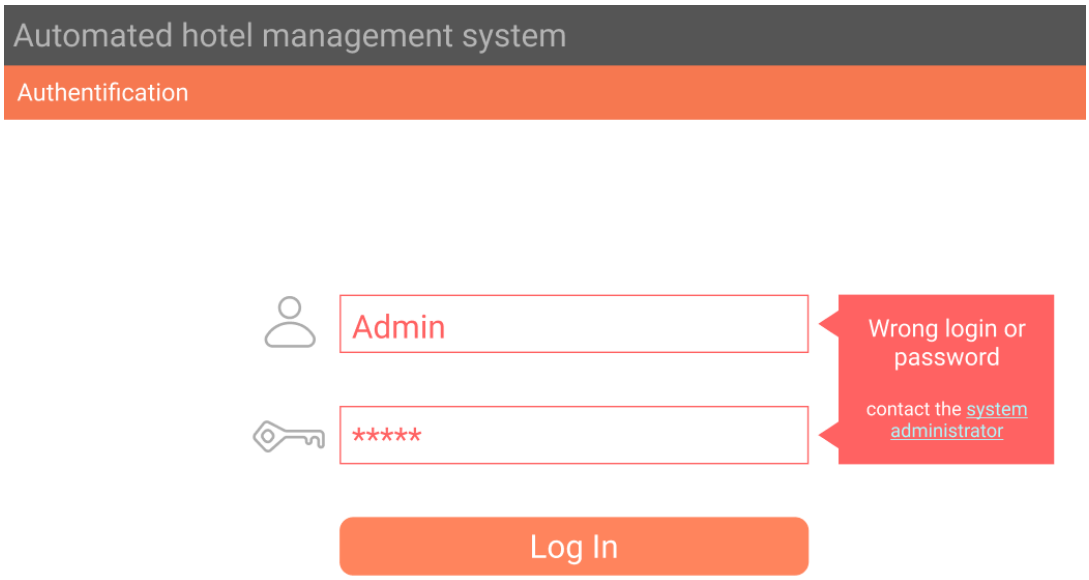


Рисунок 3.9. Екран неправильно введених даних при автентифікації

При введені вірних даних система перейде у стан очікування відповіді від серверу, під час чого буде крутитись так званий «прелодер», який буде сигналізувати про те що система все ще очікує відповідь від серверу (рисунок 3.10.).

Після того як прийде відповідь від сервера з дозволом подальшого використання програми, система змінить інтерфейс на екран головного меню.

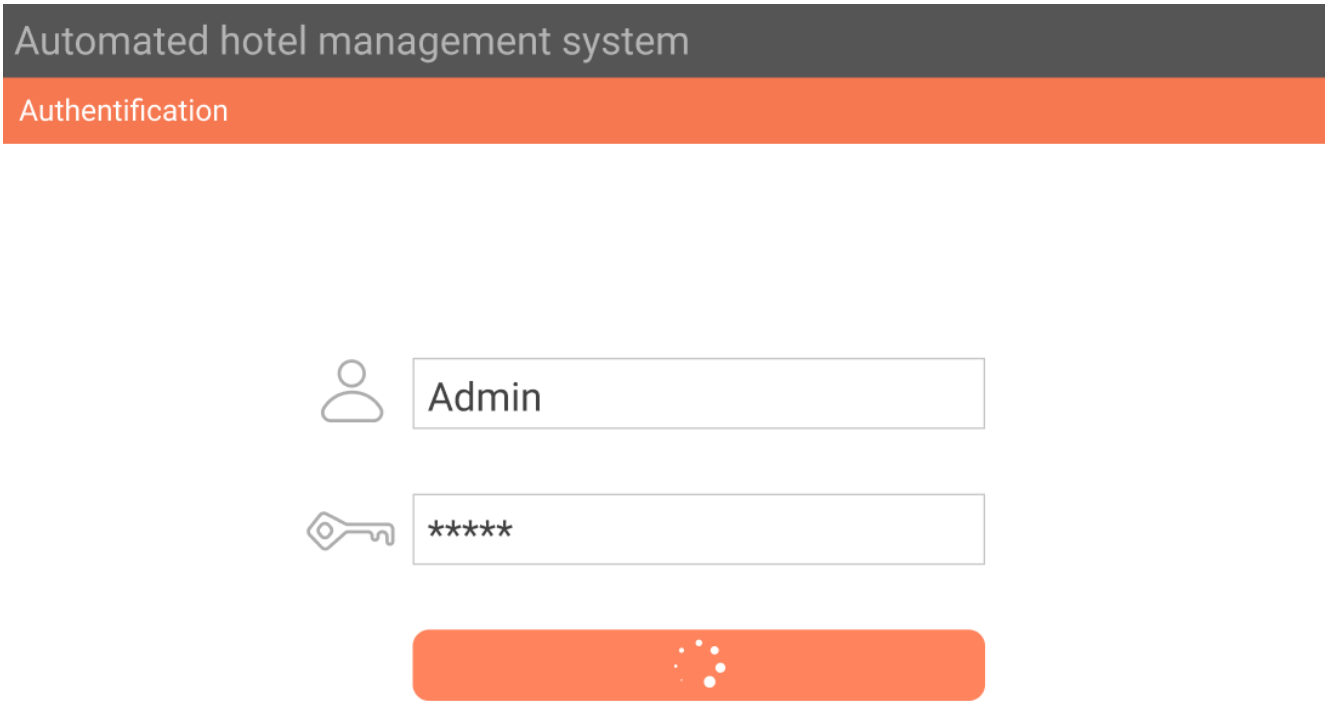


Рисунок 3.10. Очікування відповіді від бази даних

Головне меню інтерфейсу користувача складається з шести сегментів, які представляють собою функціонал програми.

Екран меню вміщує в себе стрічку з назвою програмного продукту, нижче елементи керування профілем користувача, та кнопка вийти з облікового запису. Нижче знаходиться панель з функціями які програма може виконувати, а саме:

- Керування номерами готелю

Цей пункт панелі дозволяє проводити маніпуляції відносно резервації поселення та виселення клієнта з готелю.

- Керування рестораном готелю

Цей пункт панелі дозволяє проводити маніпуляції відносно резервації столику та обслуговування клієнта у ресторані готелю.

- Керування аудитом

Цей пункт панелі дозволяє проводити маніпуляції відносно даних про внутрішні ресурси готелю.

- Керування веб-сайтом готелю

Цей пункт панелі дозволяє проводити маніпуляції відносно веб-сайту готелю, вносити в нього зміни, додавати новий матеріал у стрічку новин на сайті.

- Керування персоналом готелю

Цей пункт панелі дозволяє керувати персоналом готелю, перенаправляти робочу силу у необхідні для обслуговування зони готелю, кімнати, ресторани, та інше.

- Управління загальними налаштуваннями

У цьому пункті панелі зібрані усі загальні налаштування системи, від зміни інтерфейсу користувача, мови, до налаштувань підключення до локальної бази даних.

Усі ці пункти панелі з функціями програми представлені на рисунку 3.11.

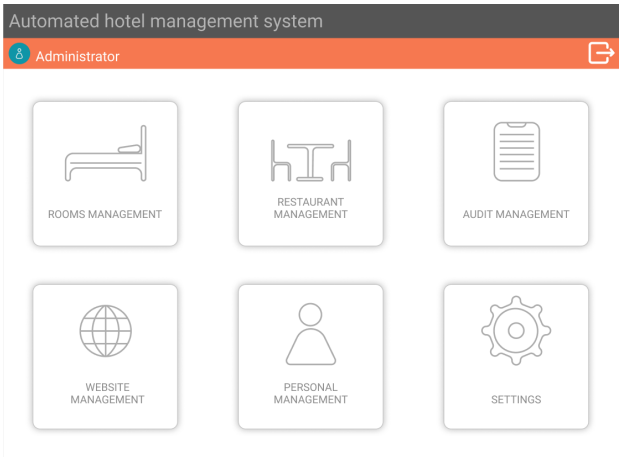


Рисунок 3.11. Інтерфейс головної пане функцій програми.

Одними з основних функцій програми являються управління номерами готелю та управління рестораном готелю.

На екрані управління номерами готелю зроблено сітку яка допомагає орієнтуватись у зарезервованих, або використовуваних номерах готелю, адмініструвати їх та редагувати дані у їхніх записах.

Сітка виконана у вигляді так званої дошки «шахматки», це дуже легка для розуміння та орієнтації у списку номерів дошка (рисунок 3.12).

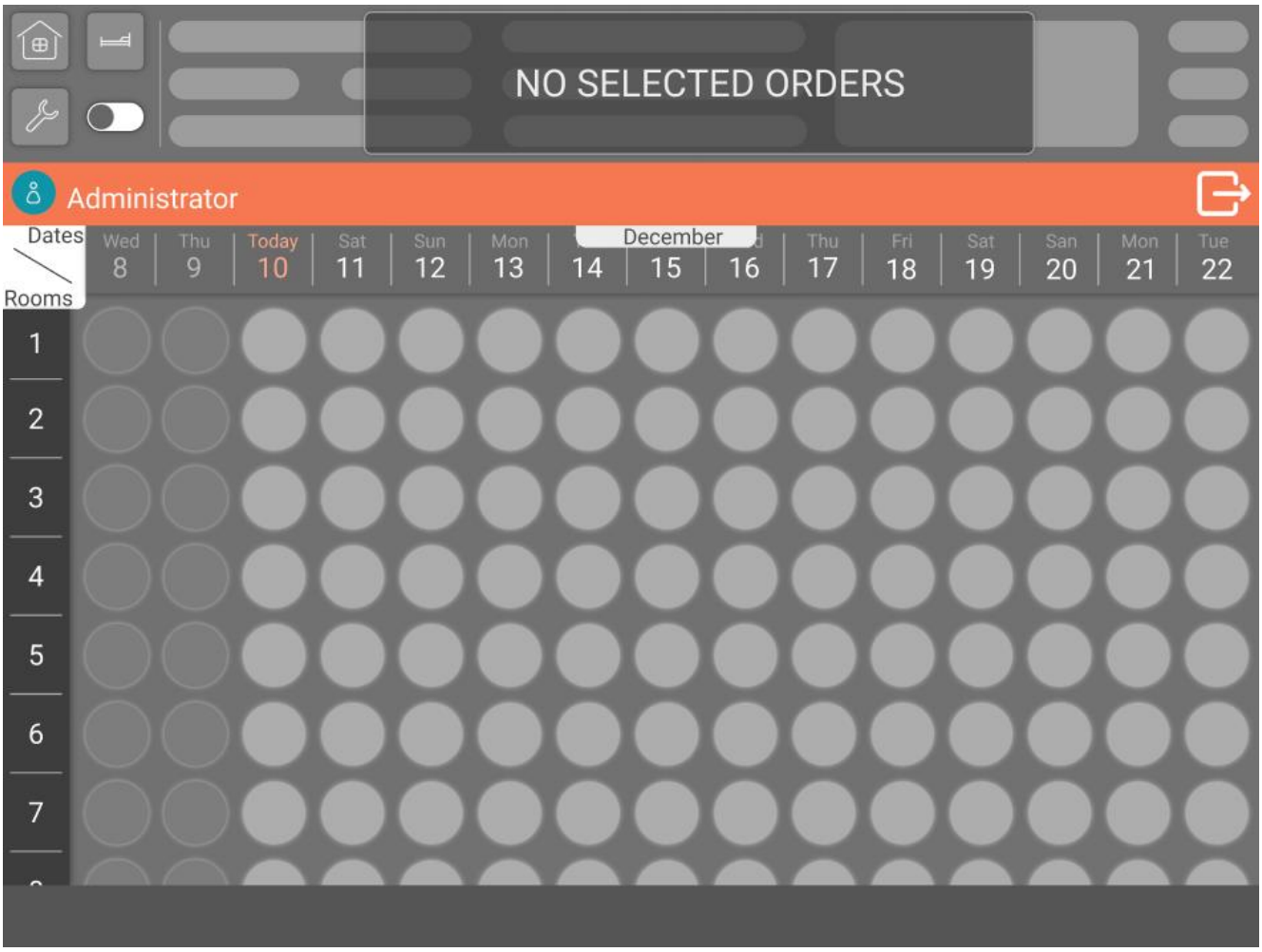


Рисунок 3.12. Екран управління номерами готелю.

Наступний екран основних функцій – це екран управління рестораном готелю. Тут розташована робоча область, на які схематично, подібно до розташування у самому ресторані, адміністратор розташовує столики по області, та

надає їм номери по яким можна орієнтуватись у пошуку потрібного столика. Пошук столика виконується за допомогою, дати та часу резервації, імені клієнта, або по його номеру у готелі за відповідну дату та час (рисунок 3.13.).

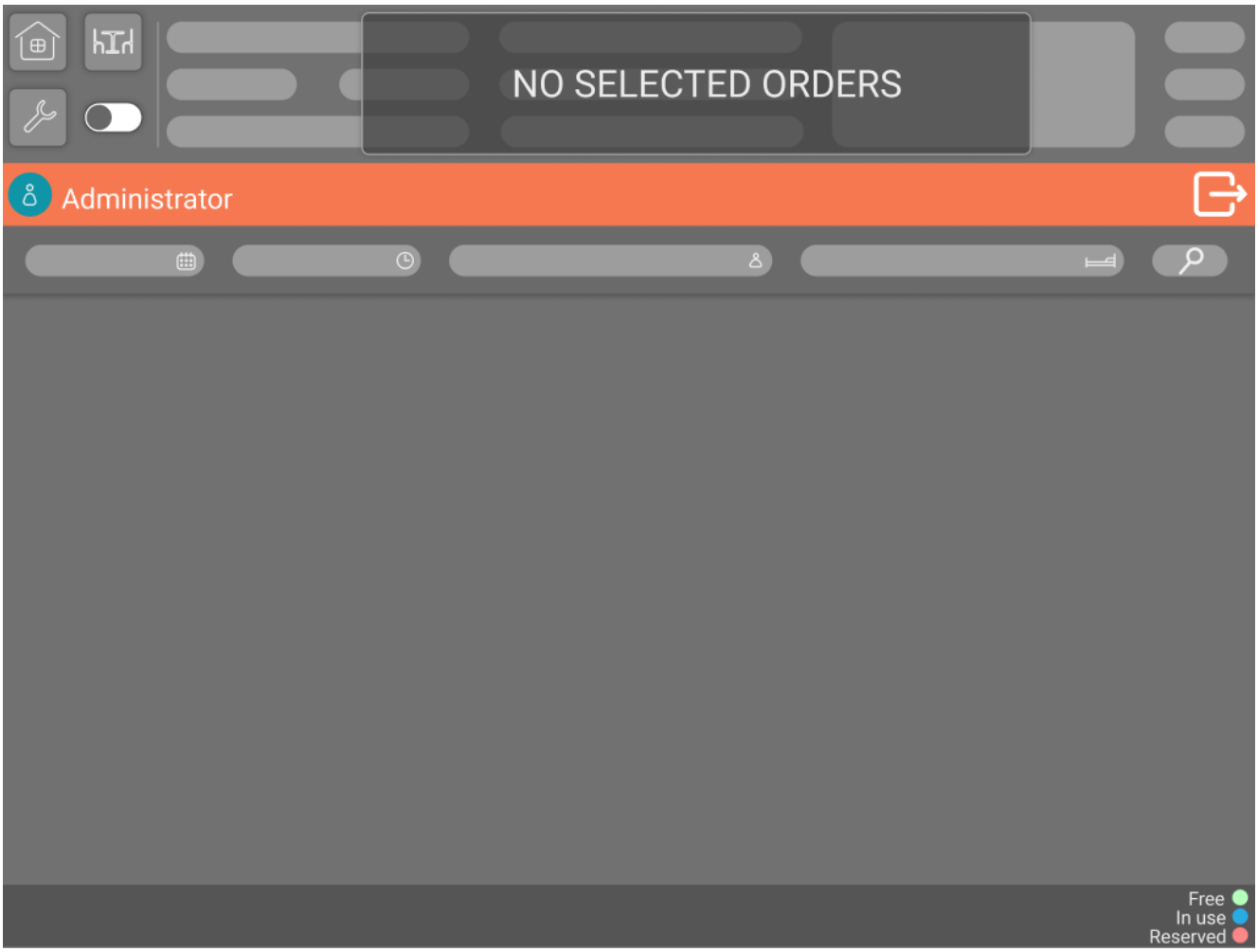


Рисунок 3.13. Екран упрвління рестораном готелю.

3.5. Створення бази даних для застосунку

Щоб упорядкувати свої дані та керувати ними, вам потрібно додати базу даних, створену за допомогою системи керування базами даних MySQL Workbench.

Перевагою цієї системи є інформаційна панель продуктивності та звіти, які дозволяють легко переглядати загальну продуктивність сервера, а різні звіти надають уявлення про точки доступу, оператори SQL, мережу, платформу даних тощо. Він надає багато функцій, які можна використовувати для проектування та моделювання. Ви можете створювати складні моделі, виконувати зворотний і

прямий інжиніринг, а також легко вносити зміни та керувати документами, які можна використовувати для розробки своєї бази даних.

Запити SQL створені й оптимізовані, їх можна виконувати за допомогою візуальних інструментів, наданих робочим середовищем MySQL. Інші функції, які допомагають і спрощують завдання розробки та виконання запитів, включають автозаповнення, підсвічування синтаксису, надання історії виконання запиту та повторне використання фрагментів SQL. Панель підключень може зберігати та керувати різними підключеннями до баз даних, включаючи структуру MySQL. До об'єктів схеми та бази даних можна отримати миттєвий доступ за допомогою Object Explorer [7].

Існує також візуальна консоль, яка надається спеціально MySQL Workbench, яку можуть використовувати адміністратори та розробники баз даних для перевірки всього середовища. Інші доступні інструменти можуть бути корисними для конфігурації сервера, адміністрування користувачів, аудиту даних для перевірки, перевірки справності та резервного копіювання бази даних, а також відновлення даних.

Продуктивність додатків баз даних можна проаналізувати та покращити за допомогою набору інструментів, наданих середовищем робочого столу MySQL. Можна знайти точки доступу, а також запити та оператори SQL.

Показники продуктивності можна переглядати та аналізувати за допомогою панелі керування продуктивністю робочого столу. Інші місця, де можна оптимізувати запити, також пропонуються та наочно пояснюються на робочому місці. Основна частина програми, яка включає пошук і зберігання даних, заснована на виконанні запиту та його продуктивності.

Інструмент Workbench надає рішення та функції, які можна використовувати для міграції даних на різні платформи, такі як Sybase ASE, Microsoft Access, PostgreSQL, Microsoft SQL Server, а також об'єктів, даних, таблиць у реляційній базі даних MySQL. Існуючі програми можна легко конвертувати для роботи на

будь-якій іншій платформі, як-от Windows, Linux тощо. Дані можна перенести зі старої версії MySQL до новішої за допомогою робочого середовища.

Давайте розглянемо створення таблиці бази даних. Для того, щоб створити макет таблиці, необхідно ввести таку команду «СТВОРИТИ ТАБЛИЦЮ». VARCHAR є хорошим вибором для стовпців імені та прізвища користувача, оскільки значення стовпців відрізняються за довжиною. Довжини цих стовпчиків не повинні бути однаковими. Зазвичай ви можете вибрати будь-яку довжину від 1 до 65535, що здається найбільш бажаним варіантом. Якщо згодом буде знайдено неправильний вибір і стане зрозуміло, що потрібне довше поле, MySQL пропонує оператор "ALTER TABLE". Для стовпців дати та часу для прийому пацієнта доцільно використовувати тип даних «ДАТА». Після створення таблиці маємо наступний результат (рис. 3.14):

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idusers	INT	☑	☑	☐	☐	☐	☐	☑	☐	
name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
lastName	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
login	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
password	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
date	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
time	TIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Рисунок 3.14. – Поля даних користувачів

3.6. Аналіз результатів тестування

Результатом всього процесу розробки став готовий програмний продукт, який може виконувати основні адміністративні функції автоматизованої система управління готелем.

В ході розробки було створено деякі екрани автоматизованої системи управління записами клієнтів. Створені екрани мають досить широкий функціонал, який дозволяє швидко та просто модифікувати дані, виконувати адміністрування готелю.

3.6.1. Навігація у записах клієнтів.

Дана система передбачена для комплексного використання у процесах повного циклу адміністрування готелів.

Серед основних функцій даного програмного рішення можна виділити:

- керування номерами готелю – в даному програмному рішенні це допомагає значно спростити процес поселення, виселення, та ведення клієнта під час перебування у готелі. Завдяки хмарним технологіям забезпечується безпека, та недоторканість даних злочинцями, також це надає можливість зробити дані корисними, оскільки відкривається великий перелік функцій по аналізу даних, що надає потужний інструмент для забезпечення більш ефективної роботи готелю та процесу використання номерного фонду готелю;
- керування рестораном готелю – це програмне рішення використовує принцип описаний вище та забезпечує керування бронюванням, та обслуговуванням столику у ресторані готелю.
- Адміністрування обліку активів – це програмне рішення яке надає потужний інструмент для аудиту активів готелю, що значно спрощує процес створення обробки фінансових звітів готелю, це підвищує ефективність ведення бізнесу.
- Адміністрування веб-сайту готелю – дане рішення надає можливість керувати вмістом веб-сайту, для забезпечення безперешкодної актуалізації інформації, яку потрібно донести до користувача.
- Адміністрування персоналу – це рішення яке надає можливість з легкістю керувати персоналом готелю, надавати робітникам роботу, та без будь яких перешкод керувати зайнятістю персоналу.

Дана система має простий та ненавантажений інтерфейс користувача, що робить приємною у використанні, не заплутує, користувача програмного продукту.

Одним із основних екранів – це екран навігації у записах клієнтів (рисунок 3.15). Цей екран дозволяє дуже легко орієнтуватись в усіх записах клієнтів.

Записи клієнтів поділяються на 4 типи (рисунок 3.15):

- Підтверджений. Під типом «підтверджений», мається на увазі, що після резервації, клієнт сплатив необхідні тарифи, та повністю підтвердив, що буде використовувати номер готелю, який він зарезервував. Цей тип запису підсвічується зеленим кольором (рисунок 3.16).
- Зарезервований. Під типом «зарезервований», мається на увазі, що номер був зарезервований на певний період часу, на який інші клієнти не зможуть зарезервувати цей самий номер. Цей тип запису підсвічується білим кольором (рисунок 3.17).
- У використанні. Під типом «у використанні», мається на увазі, що клієнт заселився до номеру, та використовує його на протязі оголошеного періоду часу. Цей тип запису підсвічується синім кольором (рисунок 3.18)
- Відмінений. Під типом «відмінений», мається на увазі що клієнт відмінив резервацію номеру, або під час використання номеру відмінив подальше використання номеру готелю. Цей тип запису підсвічується червоним кольором (рисунок 3.19).

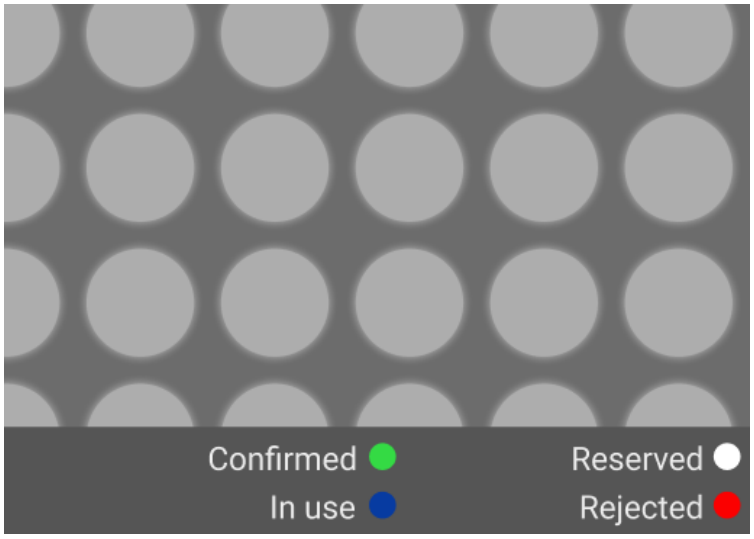


Рисунок 3.15. Типи записів клієнтв

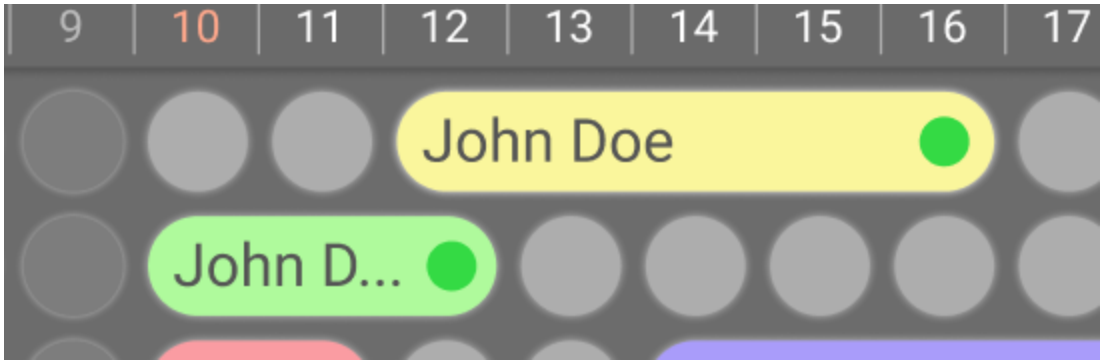


Рисунок 3.16. Тип запису «Підтверджено»

Рисунок 3.17. Тип запису «Зарезервовано»

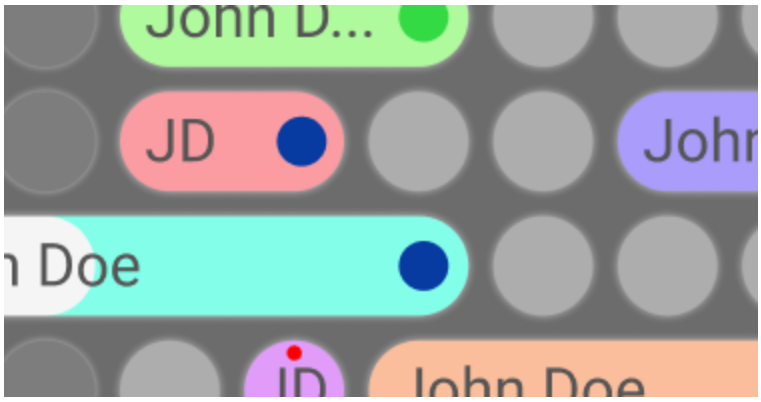


Рисунок 3.18. Тип запису «У використанні»

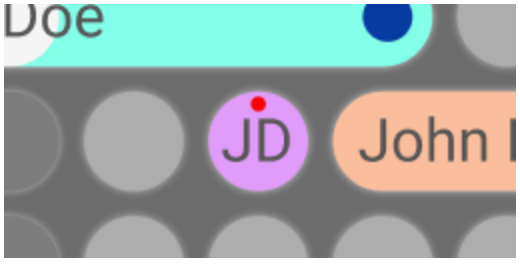


Рисунок 3.19. Тип запису «Відмінений»

Одними з основних функцій програми являються управління номерами готелю та управління рестораном готелю.

На екрані управління номерами готелю зроблено сітку яка допомагає орієнтуватись у зарезервованих, або використовуваних номерах готелю, адмініструвати їх та редагувати дані у їхніх записах. Сітка виконана у вигляді так званої дошки «шахматки», це дуже легка для розуміння та орієнтації у списку номерів дошки (рисунок 3.20).

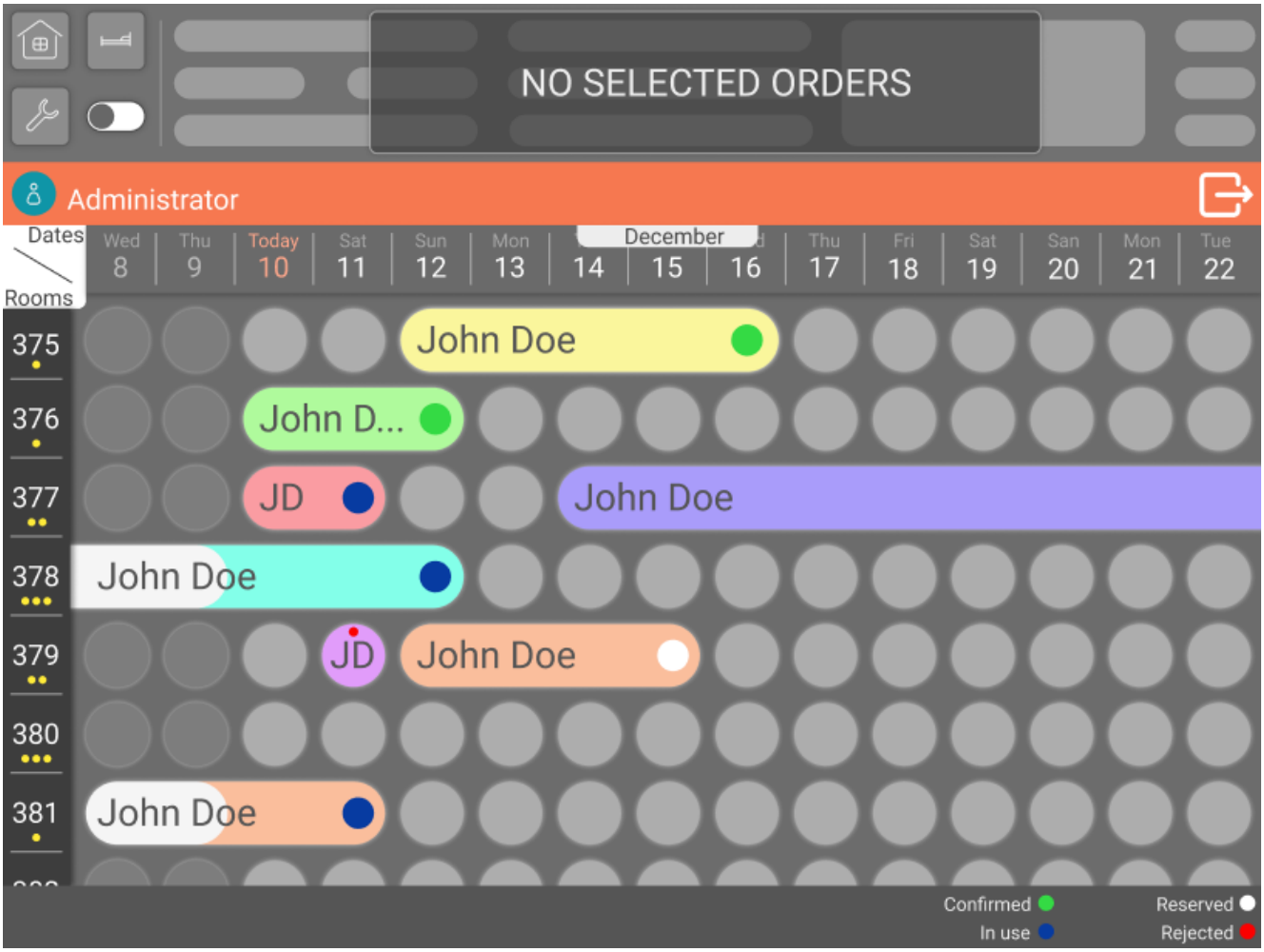


Рисунок 3.20 Навігаційна панель списку номерів готелю

Для автоматизації системи створюваного програмного рішення, було використано підхід, який дозволив зробити систему універсальною (такою що зможе працювати на різних операційних системах), для цього було обрано мову програмування, яка дозволяє працювати з різними операційними системами, та

мати однаковий набір функціоналу на будь-якій з них. Виконавши повноцінну автоматизацію програмного продукту, буде розділено програмний продукт на деяку кількість мікросервісів, які допомагають розділити обов'язки кожного з них, та передбачає перетинання обов'язків між собою, такий підхід зробив програмний продукт більш стійким до неочікуваних збоїв у роботи програмного продукту, та дозволив розширювати функціонал програмного продукту.

Після автоматизації було забезпечено недоторканість даних, та безпеку автентифікації за допомогою ієрархії клієнт-серверного зв'язку описаного у пункті 35, але додатково було розділено бази даних які працюють з даними користувачів готелю, та даними автентифікації клієнтської частини програмного продукту, за допомогою розділення їх по різним серверам.

У наступному екранному інтерфейсі представлено функціонал, який дозволяє виконувати корегування даних.

Для того щоб можна було отримати інформацію до певної комірки із зарезервованим номером готелю, потрібно натиснути на потрібну комірку, завдяки чому у верхній частині екрану з'явиться інформація про клієнта який зарезервував номер (рисунок 3.21). У ці дані входять:

- Ім'я – дане поле може змінювати лише головний адміністратор системи, який має відповідні права доступу, у випадку якщо ім'я та фамілія клієнта не збігаються із фактичними.
- Дати – дані поля зазначають термін перебування у готелі. Зміна цього поля може відбуватись виключно за правами головного адміністратора.
- Номер кімнати – дані у цьому полі не можуть бути змінені, для їх зміни потрібно видалити резервацію та створити нову.
- Номер телефону – це поле не може бути змінене та слугує елементом зв'язку з клієнтом, додатково система може присилати одноразові коди на номер телефону клієнта для верифікації резервування номеру готелю.

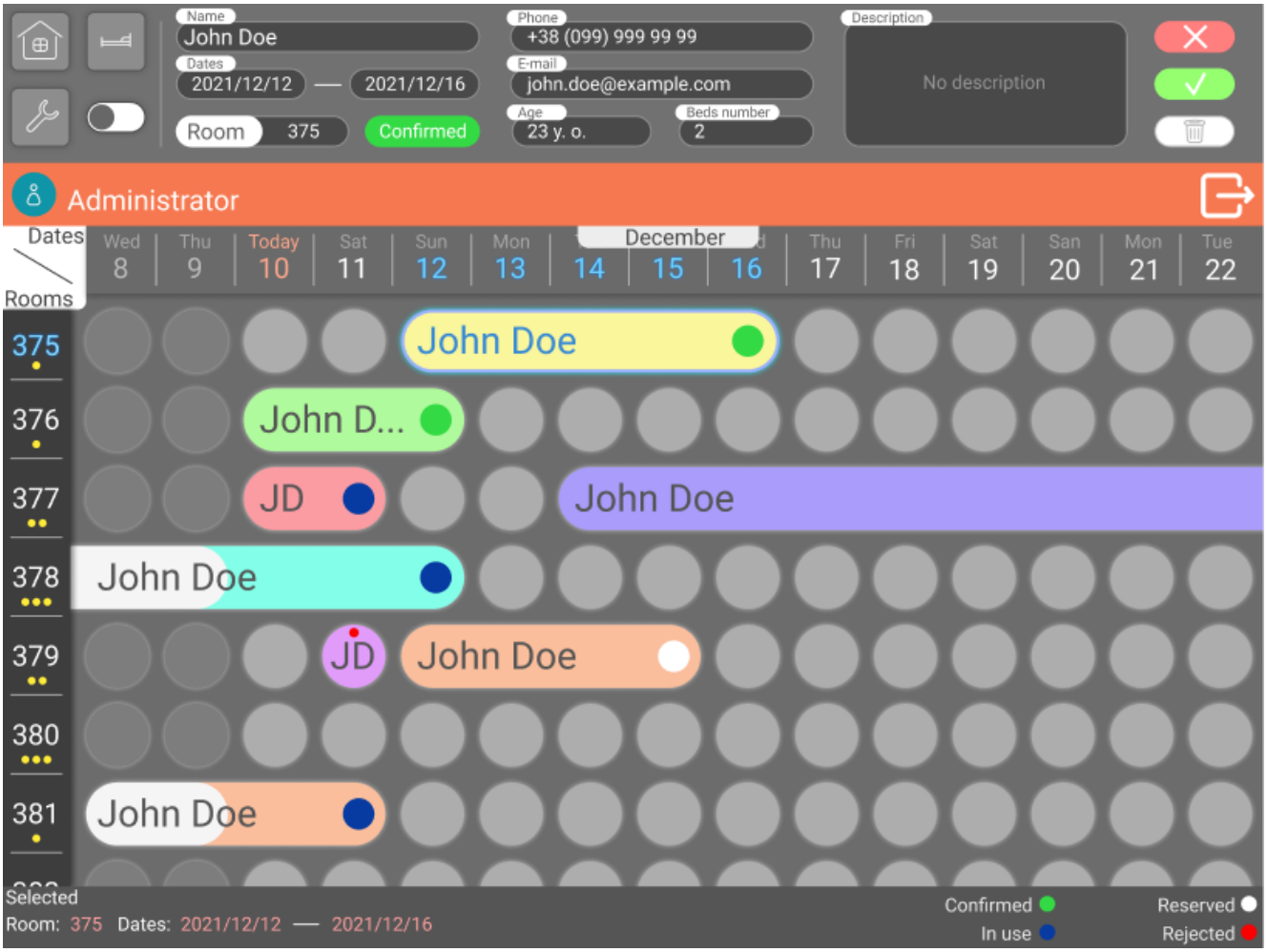


Рисунок 3.21. Екран функціоналу, який дозволяє виконувати корегування даних

Для швидкої навігації по додатку у верхній лівий кут інтерфейсу винесено швидке меню доступу до інших функцій додатку (рисунок 3.23).

Серед такого функціоналу можна виділити:

- Керування номерами готелю

  Цей пункт панелі дозволяє проводити маніпуляції відносно резервації поселення та виселення клієнта з готелю.

- Керування рестораном готелю

  Цей пункт панелі дозволяє проводити маніпуляції відносно резервації столику та обслуговування клієнта у ресторані готелю.

- Керування аудитом

Цей пункт панелі дозволяє проводити маніпуляції відносно даних про внутрішні ресурси готелю.

- Керування веб-сайтом готелю

Цей пункт панелі дозволяє проводити маніпуляції відносно веб-сайту готелю, вносити в нього зміни, додавати новий матеріал у стрічку новин на сайті.

- Керування персоналом готелю

Цей пункт панелі дозволяє керувати персоналом готелю, перенаправляти робочу силу у необхідні для обслуговування зони готелю, кімнати, ресторани, та інше.

- Управління загальними налаштуваннями

У цьому пункті панелі зібрані усі загальні налаштування системи, від зміни інтерфейсу користувача, мови, до налаштувань підключення до локальної бази даних.

У збільшеному розміні усі пункти меню можна побачити на рисунку 3.22.

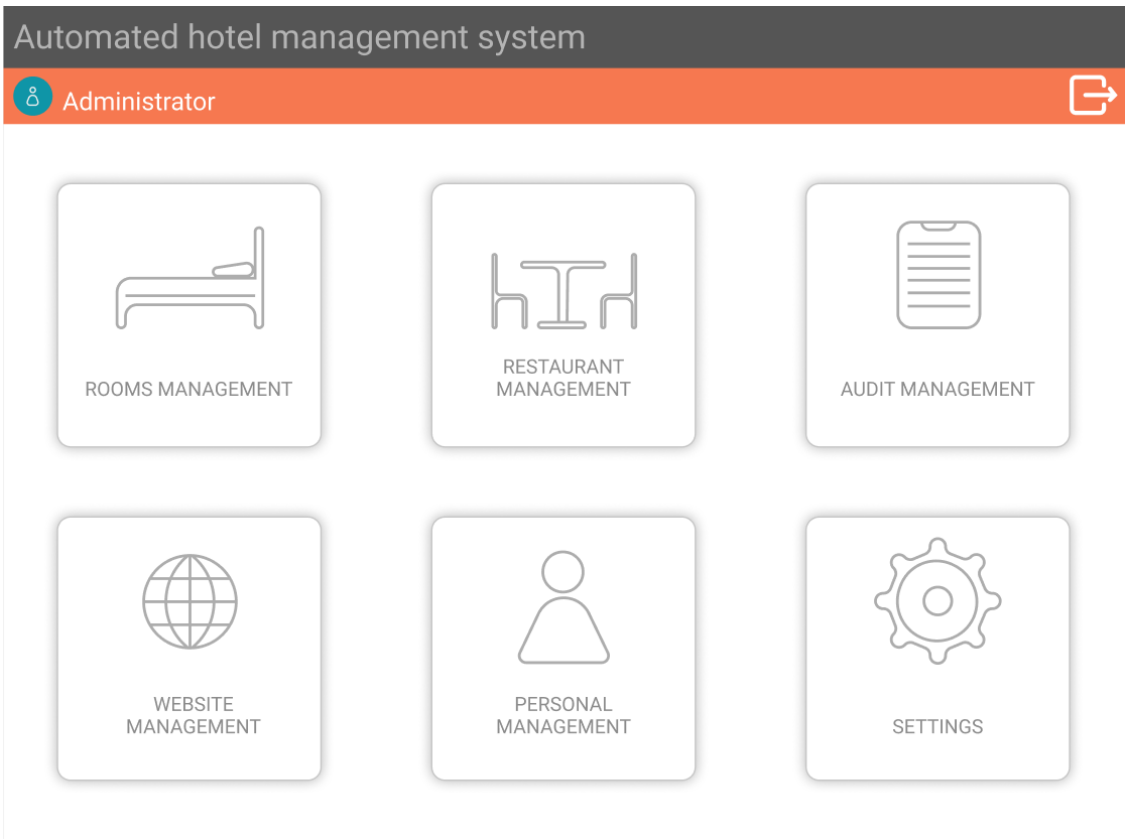


Рисунок 3.22. Інтерфейс головної пане функцій програми.

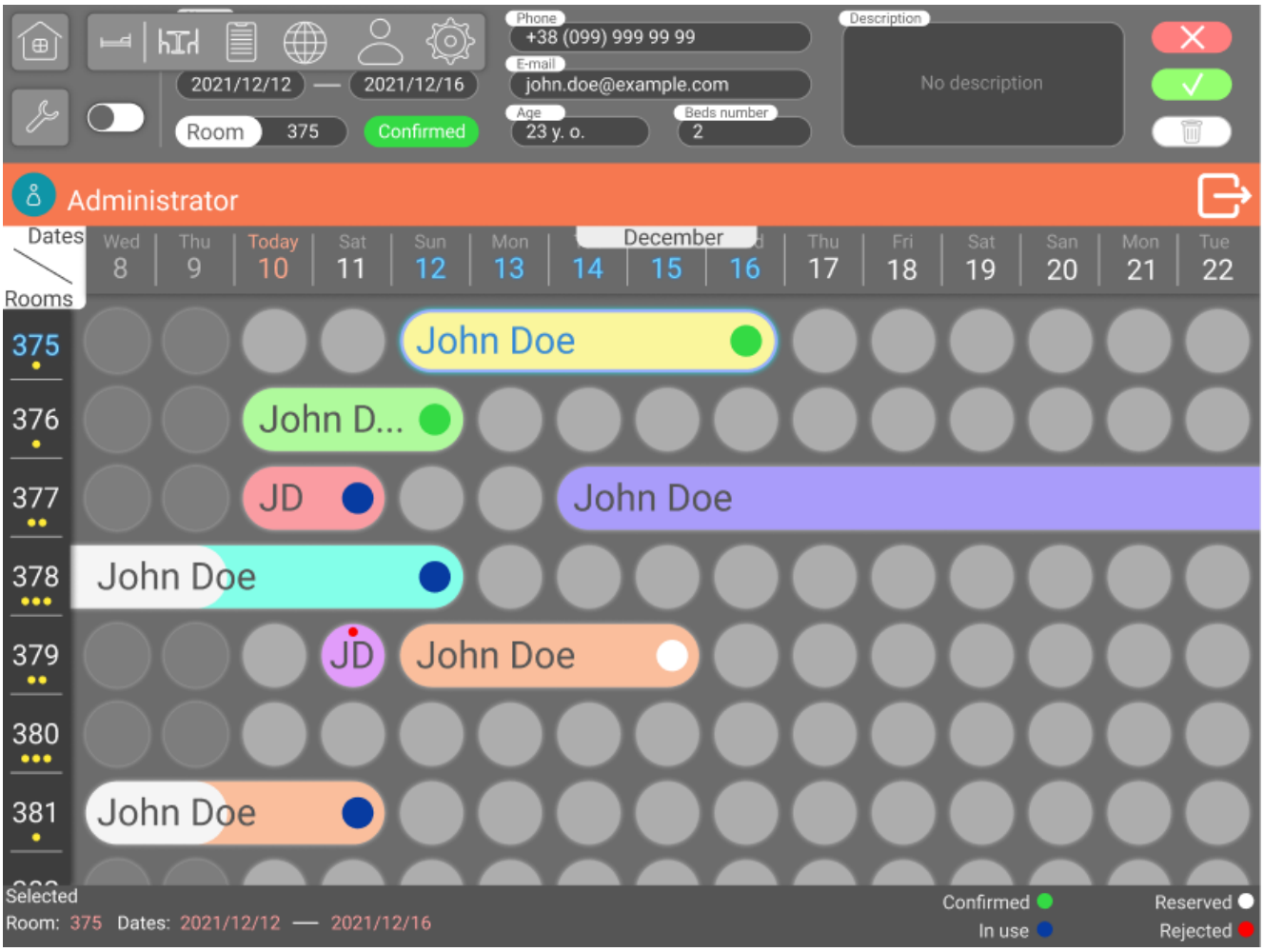


Рисунок 3.23. Швидке меню доступу до інших функцій додатку

Досить важливим сервісом для готелю являється ресторан, в якому відвідувачі мають змогу провести час з комфортом та ситістю. Для того щоб ресторан був пов'язаний з готелем не тільки будівлею, було створено додаток який дозволяє керувати резервацією столиків у ресторані.

Головними атрибутами інтерфейсу системи управління рестораном є основна панель зі столиками, які можна конфігурувати із необхідною розстановкою, для полегшення обслуговування клієнтів за цими столиками.

У верхній частині екрану зосереджено поля з даними про клієнта ресторану, це допоможе легко контактувати з клієнтом.

У нижній частині екрану розташована полоса статусу, та короткої інструкції. На ній зазначаються всі можливі статуси для столиків в залежності від набору даних у пошуковому меню (рисунок 3.24).

Рисунок 3.24. Екран із незазначеним пошуковим меню.

У випадку якщо у меню вводиться тільки дата за якою є можливість знайти клієнта по записам, то система виведе всі столики у світло синьому кольорі, що сигналізуватиме про те що столики мають багато записів на конкретну дату, щоб передивитись усі записи на цю дату, потрібно натиснути на столик, після чого з’явиться випадаюче меню, що дозволить побачити усі записи клієнтів у списку. Деякі записи мають приховані дані, які можна розгорнути, та оглянути детально (рисунок 3.25).

Рисунок 3.25 – Екран пошуку столику клієнта по даті.

На відміну від пошуку даних про клієнта який зарезервував столик, по даті, ведення більш точних даних, таких як дата та час, дають більш точний результат пошуку, та виводять конкретного клієнта за конкретним столиком. Натиснувши на столику з’явиться випадаючий список, який виведе тільки одного клієнта (рисунок 3.26).

Після того як випадаюче меню з’явилось, є змога натиснути на запис конкретного клієнта, завдяки чому з’явиться детальна інформація відносно клієнта. Деякі поля неможливо редагувати, інші може редагувати виключно головний адміністратор (рисунок 3.27).

Рисунок 3.26. Екран пошуку столику клієнта по даті та часу.

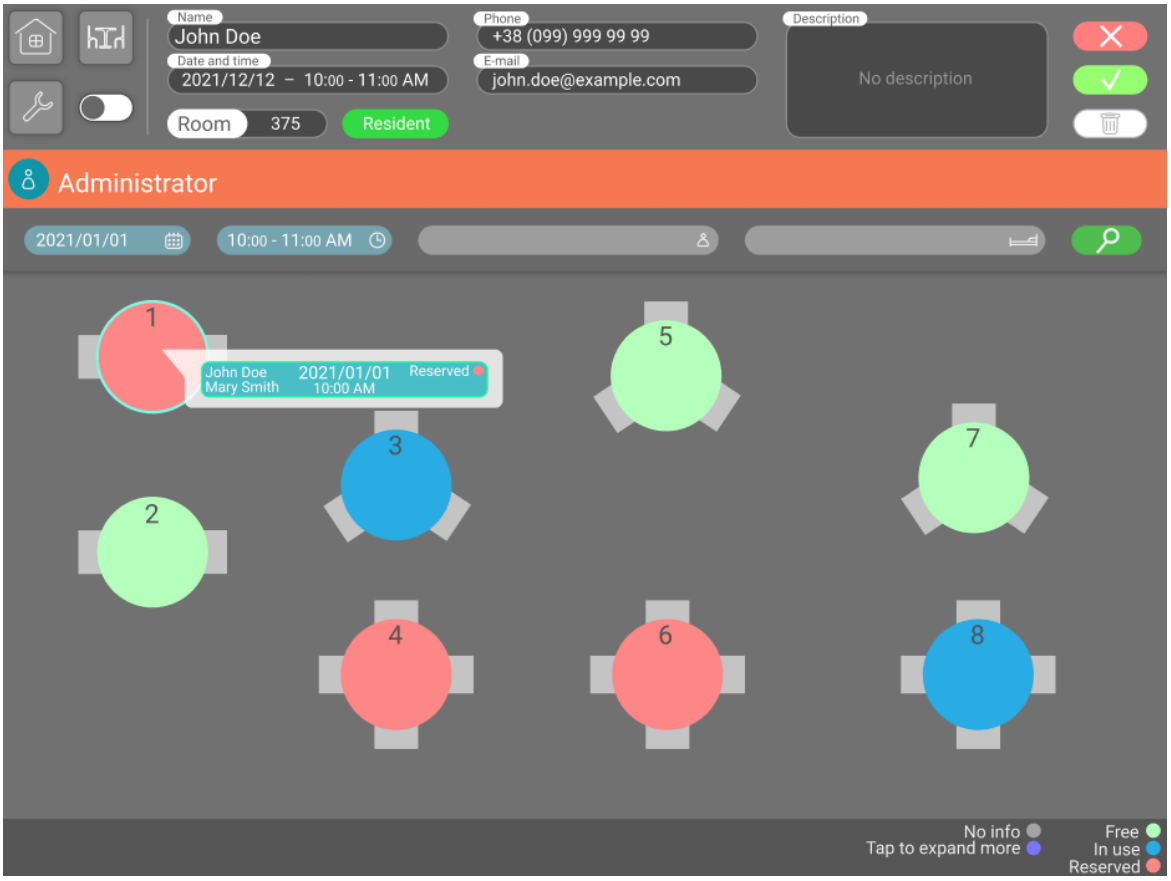


Рисунок 3.27. Екран керування даними резервації столику клієнтом

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Автоматизована система управління адміністративними функціями готелю» відноситься до науково-технічних робіт, які плануються для використання безпосередньо самим розробником (замовником), тобто її результатами буде користуватися тільки одна особа – розробник (або замовник). У цьому випадку нам потрібно довести ефективність інвестицій, вкладених у цей проект самим розробником (замовником).

Для цього випадку необхідно виконати такі етапи робіт:

1) провести технологічний аудит власної науково-технічної розробки, тобто встановити її науково-технічний рівень;

2) розрахувати витрати на здійснення науково-технічної розробки;

3) провести розрахунок економічної ефективності науково-технічної розробки у випадку її впровадження розробником (замовником) на власному підприємстві та обґрунтувати економічну доцільність впровадження розробником (замовником) розробленого науково-технічного проекту.

4.1. Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Автоматизована система управління адміністративними функціями

готелю» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [52].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни
4	Технічні та споживчі властивості продукту	Технічні та споживчі влас-тивості продукту	Технічні та споживчі властивості продукту на	Технічні та споживчі властивості продукту	Технічні та споживчі властивості продукту
5	Експлуатаційні витрати значно вищі, ніж в	Експлуатаційні витрати дещо вищі, ніж в	Експлуатаційні витрати на рівні експлуатаційни	Експлуатаційні витрати трохи нижчі, ніж в	Експлуатаційні витрати значно нижчі, ніж в
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти	Необхідне незначне навчання фахівців та збільшення їх	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	4	4	4
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	2	3	2
5. Ринкові переваги (експлуатаційні витрати)	3	4	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	4	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	39	39
Середньоарифметична сума балів $СБ_c$	38,7		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [52].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизована система управління адміністративними функціями готелю» становить 38,7 бала, що, відповідно до таблиці 4.3, свідчить про

комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2. Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [53]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \qquad (4.1)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

- для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \,, \qquad (4.2)$$

де $I_{нi}$ та $I_{на}$ – чисельні значення конкретного *i*-го технічного показника якості відповідно для нової розробки та аналога;

- для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{нi}} \,; \qquad (4.3)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість підтримуваних ОС	шт.	2	4	2	0,1
Кількість підтримуваних адміністративно-управлінських функцій	шт.	18	56	3,11	0,3
Можливість взаємодії з наявними базами даних	бал	5	8	1,6	0,15
Інтегрування з функціональними підсистемами	бал	4	8	2	0,25
Рівень захищеності баз даних	бал	7	9	1,26	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2 \cdot 0{,}15 + 3{,}11 \cdot 0{,}1 + 1{,}6 \cdot 0{,}3 + 2 \cdot 0{,}25 + 1{,}26 \cdot 0{,}2 = 2{,}13.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,13 рази.

4.3. Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Автоматизована система управління адміністративними функціями готелю», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1. Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [52]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 12500{,}00 \cdot 44 / 22 = 25000{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	12500,00	568,18	44	25000,00

Продовження таблиці 4.5

Інженер-програміст	11650,00	529,55	24	12709,09
Консультант з функціонування готельного бізнесу	12800,00	581,82	8	4654,55
Інженер-проектувальник автоматизованих систем управління	11500,00	522,73	26	13590,91
Консультант (менеджер-адміністратор)	12400,00	563,64	5	2818,18
Всього				58772,73

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Автоматизована система управління адміністративними функціями готелю» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=2379,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [52];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 22 дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 2379,00 · 1,10 · 1,65 / (22 · 8) = 24,53 *грн.*
$З_{р1}$ = 24,53 · 6,30 = 154,56 *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка офісного обладнання	6,30	2	1,10	24,53	154,56
Підготовка робочого місця розробника системи	7,20	3	1,35	30,11	216,79
Інсталяція програмного забезпечення розробки програмних модулів	5,00	5	1,70	37,92	189,58
Формування інформаційної бази даних функцій готелю	11,00	4	1,50	33,45	368,00
Тренування системи взаємодії програмних функціональних блоків управління	8,00	4	1,50	33,45	267,64
Тестування підсистем	12,00	3	1,35	30,11	361,31
Монтаж інтерфейсних блоків функціонального управління	6,00	6	2,00	44,61	267,64
Всього					1825,51

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (58772,73 + 1825,51) · 10 / 100% = 6059,82 грн.

4.3.2. Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (58772,73 + 1825,51 + 6059,82) · 22 / 100% = 14664,77 грн.

4.3.3. Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Автоматизована система управління адміністративними функціями готелю».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 2,00 \cdot 116,00 \cdot 1,1 - 0,000 \cdot 0,00 = 255,20$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний AMOCOOL-500 (A4)	116,00	2,00	0,000	0,00	255,20
Папір для заміток AMOCOOL-B (A5)/70	66,00	2,00	0,000	0,00	145,20
Начиння канцелярське AMOCOOL	173,00	4,00	0,000	0,00	761,20
Органайзер офісний AMOCOOL light	122,00	4,00	0,000	0,00	536,80
Картридж для принтера HP-7021	950,00	2,00	0,000	0,00	2090,00
Диск оптичний COOL-CD/RW	11,80	4,00	0,000	0,00	51,92
FLASH-пам'ять KingsBAFF 32GB	308,00	1,00	0,000	0,00	338,80
Всього					4179,12

4.3.4. Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Автоматизована система управління адміністративними функціями готелю», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де H_j – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*).

$К_в = 4 \cdot 960,00 \cdot 1,1 = 4224,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Інтерфейсні блоки взаємозв'язків функціональних систем управління	4	960,00	4224,00
Всього			4224,00

4.3.5. Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец} = 15700,00 \cdot 1 \cdot 1,1 = 17270,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.11 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверний блок системи управління	1	15700,00	17270,00
Автоматизована система відеоспостереження	1	12200,00	13420,00
Інтерфейсний блок обміну інформацією з системою безпеки готелю	1	9400,00	10340,00
Всього			41030,00

4.3.6. Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 5515,00 \cdot 1 \cdot 1,1 = 6066,50$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	1	5515,00	6066,50
Прикладний пакет Microsoft Office 2019	1	4385,00	4823,50

Продовження таблиці 4.9

Прикладний пакет MATLab VisProject	1	6720,00	7392,00
Всього			18282,00

4.3.7. Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (24690{,}00 \cdot 2) / (2 \cdot 12) = 2057{,}50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Програмно-аналітичний комплекс	24690,00	2	2	2057,50
Графічно-обчислювальний комплекс обробки даних	27200,00	2	2	2266,67

Продовження таблиці 4.10

Програмне забезпечення розробки автоматизованих систем	10620,00	2	2	885,00
Обладнання виводу інформації	8600,00	4	2	358,33
Оргтехніка	8730,00	4	2	363,75
Приміщення лабораторії	245000,00	25	2	1633,33
Всього				7564,58

4.3.8. Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,25 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,32 \cdot 300,0 \cdot 4,25 \cdot 0,95 / 0,97 = 408,00$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Програмно-аналітичний комплекс	0,32	300,0	408,00
Графічно-обчислювальний комплекс обробки даних	0,46	300,0	586,50

Продовження таблиці 4.11

Інтерфейсні блоки взаємодії	0,01	90,0	3,83
Обладнання виводу інформації	0,40	30,0	51,00
Оргтехніка	0,56	50,0	119,00
Серверний блок системи управління	0,20	90,0	76,50
Автоматизована система відеоспостереження	0,20	90,0	76,50
Інтерфейсний блок обміну інформацією з системою безпеки готелю	0,02	90,0	7,65
Всього			1328,98

4.3.9. Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Автоматизована система управління адміністративними функціями готелю» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (4.15)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 25%.

$B_{св} = (58772,73 + 1825,51) \cdot 25 / 100\% = 15149,56$ *грн.*

4.3.10. Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$B_{сп} = (58772,73 + 1825,51) \cdot 35 / 100\% = 21209,38$ *грн.*

4.3.11. Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 62%.

$І_в = (58772,73 + 1825,51) \cdot 62 / 100\% = 37570,91$ *грн.*

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (4.18)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 115%.

$B_{нзв} = (58772,73 + 1825,51) \cdot 115 / 100\% = 69687,97$ *грн.*

Витрати на проведення науково-дослідної роботи на тему «Автоматизована система управління адміністративними функціями готелю» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}\ . \qquad (4.19)$$

$B_{заг} = 58772,73 + 1825,51 + 6059,82 + 14664,77361 + 4179,12 + 4224,00 + 41030,00 + 18282,00 + 7564,58 + 1328,98 + 15149,56 + 21209,38 + 37570,91 + 69687,97 = 301549,34$ *грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}\ , \qquad (4.20)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,9.

$ЗВ = 301549,34 / 0,9 = 335054,82$ *грн.*

4.4. Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо розробником (замовником)

При виконанні даної роботи за темою «Автоматизована система управління адміністративними функціями готелю» розглядається ситуація, коли замовник певної науково-технічної розробки використовує її тільки на своєму підприємстві (чи в організації) і не виводить її на ринок. У цьому випадку позитивним результатом від впровадження цієї науково-технічної розробки може бути покращення певних економічних та фінансових показників діяльності підприємства.

Для визначення величини майбутнього економічного ефекту та ефективності розробки визначимо певні характеристики підприємства.

Таблиця 4.12 – Вихідні дані замовника

Показник	Рік розробки	1-й рік	2-й рік	3-й рік	4-й рік
Чисельність працівників готельного бізнесу, які виконують визначені функції вручну, осіб	6	-	-	-	-
Середня заробітна плата працівника, який виконує відповідну функцію вручну, грн	12000,00	-	-	-	-
Приблизні витрати на розробку автоматизованої системи управління, грн	335054,82	-	-	-	-
Економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб	-	2	1	1	1
Кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, шт	30000	-	-	-	-
Прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт	-	20000	8000	1000	1000

В даному випадку майбутній економічний ефект та ефективність буде формуватися на основі використання таких показників: $\Delta П_я$ – зростання прибутку підприємства внаслідок зниження витрат на оплату праці працівників, які

виконують окремі виробничі чи інформаційно-технічні управлінські функції, грн. Причому $\Delta П_я$ може бути визначено як:

$$\Delta П_я = \frac{ЧП \cdot ЗП \cdot 12}{N} - \frac{(0,2...0,6) \cdot ЗВ}{\Delta N_i}, \quad (4.21)$$

де $ЧП$ – чисельність працівників, які виконують визначені функції вручну, приймемо 6 осіб; $ЗП$ – середня заробітна плата працівника, який виконує відповідну функцію вручну, приймемо 12000,00 грн; $ЗВ$ – приблизні витрати на розробку автоматизованої системи управління, приймемо 335054,82 грн; N – кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, 30000 шт; ΔN_i – прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт.

Зростання прибутку підприємства в 1-й рік впровадження розробки

$\Delta П_я$ =6·12000,00·12/30000 - 0,4·335054,82/20000 = 22,10 грн/функц.

Зростання прибутку підприємства в 2-й рік впровадження розробки

$\Delta П_я$ =6·12000,00·12/30000 - 0,4·335054,82/28000 = 24,01 грн/функц.

Зростання прибутку підприємства в 3-й рік впровадження розробки

$\Delta П_я$ =6·12000,00·12/30000 - 0,4·335054,82/29000 = 24,18 грн/функц.

Зростання прибутку підприємства в 4-й рік впровадження розробки

$\Delta П_я$ =6·12000,00·12/30000 - 0,4·335054,82/30000 = 24,33 грн/функц.

$П_я$ – прибуток, який отримує підприємство від автоматизації виконання окремої виробничої чи інформаційно-технічної управлінської функції у кожному із років після впровадження науково-технічної розробки, грн. Даний прибуток можна приблизно оцінити виходячи з формули:

$$П_я = \frac{\Delta ЧП \cdot ЗП \cdot 12}{N}, \quad (4.22)$$

де $\Delta ЧП$ – економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб;

Прибуток який отримує підприємство від автоматизації функції в 1-й рік $П_я$ =2·12000,00·12/30000 = 9,6 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 2-й рік $П_я$ =1·12000,00·12/30000 = 4,8 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 3-й рік $П_я$ =1·12000,00·12/30000 = 4,8 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 4-й рік $П_я$ =1·12000,00·12/30000 = 4,8 грн/функц.

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\Delta П_я \cdot N + П_я \cdot \Delta N)_i, \quad (4.23)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки у році що аналізується;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки;

$П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки;

ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки у році що аналізується.

Збільшення чистого прибутку підприємства в 1-й рік впровадження

$\Delta П_i$=22,10·30000+9,6·20000=854967,11 грн.

Збільшення чистого прибутку підприємства в 2-й рік впровадження

$\Delta П_i$=24,01·30000+4,8·(20000+8000)=854805,08 грн.

Збільшення чистого прибутку підприємства в 3-й рік впровадження

$\Delta П_i$=24,18·30000+4,8·(20000+8000+1000)=864556,63 грн.

Збільшення чистого прибутку підприємства в 4-й рік впровадження

$\Delta П_i$=24,33·30000+4,8·(20000+8000+1000+1000)=873978,07 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати замовник від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.24)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,13;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання замовником додаткових чистих прибутків у цьому році.

$$ПП = 854967,11/(1+0,13)^1+854805,08/(1+0,13)^2+864556,63/(1+0,13)^3+$$
$$+873978,07/(1+0,13)^4=756608,06+669437,76+599181,11+536027,12=$$
$$=12210801,08 \text{ грн.}$$

Величина початкових інвестицій PV, які замовник має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.25}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 335054,82 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 335054{,}82 = 670109{,}64$ грн.

Абсолютний економічний ефект $E_{абс}$ для розробника від можливого впровадження науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.26}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2561254,05 грн;

PV – теперішня вартість початкових інвестицій, 670109,64 грн.

$E_{абс} = ПП - PV = 2561254{,}05 - 670109{,}64 = 1891144{,}41$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.27}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 1891144,41 грн;

PV – теперішня вартість початкових інвестицій, 670109,64 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1891144{,}41/670109{,}64)^{1/4} = 0{,}398.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.28}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,1.

$\tau_{мін}$ = 0,1+0,1 = 0,2 < 0,398 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Автоматизована система управління адміністративними функціями готелю» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.29}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,398 = 2,51 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника профінансувати впровадження даної розробки для застосування в діяльності підприємства.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Автоматизована система управління адміністративними функціями готелю» становить 38,7 бала, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,13 рази.

Також термін окупності становить 2,51 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника до впровадження даної розробки при отриманні ефекту в розмірі 1891144,41 грн.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Автоматизована система управління адміністративними функціями готелю».

ВИСНОВКИ

За результатами магістерської роботи проаналізовано процес авирматизованого управління адміністративними функціями готелю та розроблено програмне рішення цієї проблеми. Проаналізовано процес обробки даних клієнтів та визначено основні компоненти системи автоматизованого управління адміністративними функціями готелю:

- Адміністрування номерів готелю;
- Адміністрування ресторану готелю;
- Адміністрування персоналу;
- Адміністрування аудиту;
- Адміністрування сайту;

Розроблено та розглянуто метод обробки даних клієнтів (серверна обробка з використанням хмарних сховищ).

В першому розділі проведено дослідження функціональних можливостей сучасних програмних рішень для управління готелю, а саме систем: Cheerze Connect, StayFlexi, Opera Hotel PMS.

В другому розділі було проведено аналіз програмних засобів, що використовувались в ході розробки автоматизованої системи, а також проведено тестування працездатності заявлених функцій системи управління готелю.

В третьому розділі було виконано проектування автоматизованої системи управління готелем, для чого розроблено структуру бази даних системи.

В четвертому розділі було проведено дослідження рівня комерційного потенціалу розробки, згідно результатів якого можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Автоматизована система управління управління адміністративними функціями готелю. Частина 2. Розробка модуля управління готелю»

Для роботи з даним програмним забезпеченням необхідний будь-який комп’ютер та система.

Додатково було розроблено функціонал, який дозволяє керувати користувацькими функціями програмного продукту.

За допомогою баз даних було розроблено структуру даних програми, що дозволило ефективно зберігати дані клієнтів, та організовувати порядок їх зберігання.

В кінці було проведено тестування програмного продукту, що дало розуміння того, що програмний продукт виконує поставлені перед ним задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Electron | Создавайте кросс-платформенные приложения при помощи JavaScript, HTML и CSS. (electronjs.org) [Електронний ресурс] Ружим доступу: https://www.electronjs.org
2. Пишем настольное JS-приложение с Electron (tproger.ru) [Електронний ресурс] Ружим доступу: https://tproger.ru/translation/desktop-js-app-with-electron/
3. Electron: разработка настольных приложений с использованием HTML, CSS и JavaScript / Блог компании RUVDS.com / Хабр (habr.com) [Електронний ресурс] Ружим доступу: https://habr.com/ru/company/ruvds/blog/436466/
4. electron/electron: Build cross-platform desktop apps with JavaScript, HTML, and CSS (github.com) [Електронний ресурс] Ружим доступу: https://github.com /electron/electron
5. electron - npm (npmjs.com) [Електронний ресурс] Ружим доступу: https://www.npmjs.com/package/electron
6. Brainhub: What Is Electron.js? Features, Architecture and Tools [Електронний ресурс] Ружим доступу: https://brainhub.eu/library/what-is-electron-js/
7. Node.js (nodejs.org) [Електронний ресурс] Ружим доступу: https://nodejs.org/uk/
8. Run JavaScript Everywhere. (nodejs.dev) [Електронний ресурс] Ружим доступу: https://nodejs.dev/
9. nodejs/node: Node.js JavaScript runtime (github.com) [Електронний ресурс] Ружим доступу: https://github.com/nodejs/node
10. Express - фреймворк веб-приложений Node.js (expressjs.com) [Електронний ресурс] Ружим доступу: https://expressjs.com/ru/
11. express - npm (npmjs.com) [Електронний ресурс] Ружим доступу: https://www.npmjs.com/package/express
12. Node.JS | Начало работы с Express (metanit.com) [Електронний ресурс] Ружим доступу: https://metanit.com/web/nodejs/4.1.php
13. expressjs/express: Fast, unopinionated, minimalist web framework for node.

(github.com) [Електронний ресурс] Ружим доступу: https://github.com /expressjs/express

14. React – JavaScript-бібліотека для створення користувацьких інтерфейсів (reactjs.org) [Електронний ресурс] Ружим доступу: https://uk.reactjs.org/

15. Oracle OPERA PMS: Reviews & Pricing 2021 | Hotel Tech Report [Електронний ресурс] Ружим доступу: https://hoteltechreport.com/operations/property-management-systems/opera-hotels-software

16. Tutorial | React.js and Spring Data REST [Електронний ресурс] Ружим доступу: https://spring.io/guides/tutorials/react-and-spring-data-rest/

17. Изучение React. Полное руководство по React (learn-reactjs.ru) [Електронний ресурс] Ружим доступу: https://learn-reactjs.ru/home

18. React JavaScript Tutorial in Visual Studio Code [Електронний ресурс] Ружим доступу: https://code.visualstudio.com/docs/nodejs/reactjs-tutorial

19. Eloquent JavaScript [Електронний ресурс] Ружим доступу: https://eloquentjavascript.net/

20. 8 benefits of using automated systems | Cimpl (uplandsoftware.com) [Електронний ресурс] Ружим доступу: https://uplandsoftware.com /cimpl/resources/blog/8-benefits-of-using-automated-systems/

21. Difference Between Manual And Automated System - Manual System vs Automated System | PadaKuu.com [Електронний ресурс] Ружим доступу: https://padakuu.com/article/1-difference-between-manual-and-automated-system-manual-system-vs-automated-system

22. What is automation? | IBM [Електронний ресурс] Ружим доступу: https://www.ibm.com/topics/automation

23. What Is an Automated System? (reference.com) [Електронний ресурс] Ружим доступу: https://www.reference.com/world-view/automated-system-c85583d0f17a632

24. Automated system for hotel room service | IEEE Conference Publication | IEEE Xplore [Електронний ресурс] Ружим доступу: https://s3-us-west-2.amazonaws.com/ieeeshutpages/xplore/xplore-ie-notice.html?

25. Hotel Automation: Everything the Modern Hotelier Needs To Know | Operto [Електронний ресурс] Ружим доступу: https://operto.com/hotel-automation/
26. The Automatic Hotel (hotelmanagement-network.com) [Електронний ресурс] Ружим доступу: https://www.hotelmanagement-network.com/features/feature103426/
27. MySQL [Електронний ресурс] Ружим доступу: https://www.mysql.com/
28. Что Такое MySQL: Объяснение MySQL Для Начинающих (hostinger.ru) [Електронний ресурс] Ружим доступу: https://www.hostinger.ru/rukovodstva/shto-takoje-mysql/
29. The most popular database for modern apps | MongoDB [Електронний ресурс] Ружим доступу: https://www.mongodb.com/
30. MongoDB Hosting: Database-as-a-Service by mLab [Електронний ресурс] Ружим доступу: https://mlab.com/
31. MongoDB Documentation [Електронний ресурс] Ружим доступу:
32. JSON [Електронний ресурс] Ружим доступу: https://docs.mongodb.com/
33. Работа с JSON - Изучение веб-разработки | MDN (mozilla.org) [Електронний ресурс] Ружим доступу: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON
34. JSON Introduction (w3schools.com) [Електронний ресурс] Ружим доступу: https://www.w3schools.com/js/js_json_intro.asp
35. Visual Studio Code - Code Editing. Redefined [Електронний ресурс] Ружим доступу: https://code.visualstudio.com/
36. microsoft/vscode: Visual Studio Code (github.com) [Електронний ресурс] Ружим доступу: https://github.com/microsoft/vscode
37. Visual Studio Live Share: средство для совместной работы с кодом в реальном времени (microsoft.com) [Електронний ресурс] Ружим доступу: https://visualstudio.microsoft.com/ru/services/live-share/
38. Brainhub: What Is Electron.js? Features, Architecture and Tools [Електронний ресурс] Ружим доступу: https://brainhub.eu/library/what-is-electron-js/
39. Git (git-scm.com) [Електронний ресурс] Ружим доступу: https://git-scm.com/
40. Start using Git on the command line | GitLab [Електронний ресурс] Ружим

доступу: https://git-scm.comhttps//docs.gitlab.com/ee/gitlab-basics/start-using-git.html

41. Программа для гостиниц / баров / ресторанов / салонов красоты / фитнес-центров / бильярда / саун. Программа для отеля и ресторана. (indom.com.ua) [Електронний ресурс] Ружим доступу: https://www.indom.com.ua/

42. Maestro PMS | Property Management Software Solution [Електронний ресурс] Ружим доступу: https://maestropms.com/

43. Maestro PMS (hospitalitynet.org) [Електронний ресурс] Ружим доступу: https://www.hospitalitynet.org/organization/17003663/northwind-maestro.html

44. Сучасний стан та інноваційні процеси розвитку готельно-ресторанного бізнесу в Україні (tourlib.net) [Електронний ресурс] Ружим доступу: https://tourlib.net/statti_ukr/dominska.html

45. ПЕРСПЕКТИВИ РОЗВИТКУ ГОТЕЛЬНО-РЕСТОРАННОГО БІЗНЕСУ В УКРАЇНІ В УМОВАХ ГЛОБАЛІЗАЦІЇ СВІТОВОГО ГОСПОДАРСТВА (uzhnu.uz.ua) [Електронний ресурс] Ружим доступу: http://www.visnyk-econom.uzhnu.uz.ua/archive/23_1_2019ua/28.pdf

46. Автоматизовані системи управління готелями [Електронний ресурс] Ружим доступу: https://tourlib.net/statti_ukr/gudzovata.htm

47. Hotel [Електронний ресурс] Ружим доступу: https://en.wikipedia.org/wiki/Hotel

48. Історія розвитку готельного бізнесу в Україні [Електронний ресурс] Ружим доступу: https://pidru4niki.com/15941024/turizm/istoriya_rozvitku_gotelnoyi_sferi_ukrayini

49. Cheerze Connect [Електронний ресурс] Ружим доступу: https://cheerzeconnect.com/

50. StayFlexi [Електронний ресурс] Ружим доступу: https://business.stayflexi.com/

51. Hotel Property Management and POS Solutions [Електронний ресурс] Ружим доступу: https://www.oracle.com/industries/hospitality/hotel-property-management/

52. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький.

– Вінниця : ВНТУ, 2021. – 42 с.

53. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А
(обов'язковий)
ВНТУ

ЗАТВЕРДЖУЮ

Завідувач кафедри КСУ

д.т.н., проф. В.М. Дубовой

_____

" 30 " вересня 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Автоматизована система управління адміністративними функціями готелю.
Частина 2. Розробка модуля управління готелем
08-01.МКР.013.00.000 ТЗ

Виконав: студент 2-го курсу, групи
2АКІТ-20м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Владислав Таламанюк _____
(ім'я та прізвище) (підпис)

Керівник: к.т.н., доцент каф. КСУ

Марія Юхимчук _____
(ім'я та прізвище) (підпис)

Вінниця 2021

1. Назва та галузь застосування
   1.1. Назва – Автоматизована система управління адміністративними функціями готелю. Частина 2. Розробка модуля управління готелем.
   1.2. Галузь застосування – Комп'ютеризовані системи управління готелями.

2. Підстава для проведення розробки.
   Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ №277 від "24" _09_ 2021 р.

3. Мета та призначення розробки.
   Метою магістерської кваліфікаційної роботи є підвищення ефективності роботи й автоматизація систем управління готелями, що дозволить зручно заносити записи клієнтів в базу даних.

4. Джерела розробки.
   Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:
   1. Готельний бізнес: підручник для студентів /за ред. Мальська М. П. – Київ, 2012. – 133 с.
   2. Трегуб В. Проектування систем автоматизації. – М.: Видавництво «Ліра-К», 2014. - 162 с.
   3. Джонсон Р. Патерни об'єктно-орієнтованого програмування / Гамма Э., Хелм Р., Влиссидес Д.. - М.: Издательство «ЛОРИ», 2020. - 159 с.
   4. Осипов Д.Л. Технології проектування баз даних / М.: Издательство «ДМК Пресс», 2007. - 284 с.

5. Вимоги до розробки.
   5.1. Перелік головних функцій:
   - реєстрація та авторизація в системі;
   - використання бази даних клієнтів;
   - призначення дати й часу поселення та виселення
   - резервація клієнта.
   - резервація столику у ресторані.
   - показ інформації зареєстрованого клієнта
   - перегляд існуючих записів клієнтів.

   5.2. Основні технічні вимоги до розробки.
   5.2.1. Вимоги до програмної платформи:
   - WINDOWS 7\8\10.

   5.2.2. Умови експлуатації системи:
   - робота на стандартних ПЕОМ в приміщеннях зі стандартними умовами;

- можливість цілодобового функціонування системи;
- текст програмного забезпечення системи є цілком закритим.

6. Стадії та етапи розробки.
   6.1 Пояснювальна записка:
   - Дослідження актуальності поставленої задачі 04.10.2021р.
   - Загальний огляд автоматизованих систем управління адміністративними функціями готелю 22.10.2021р
   - Аналіз функцій автоматизованих систем управління адміністративними функціями готелю 01.11.2021р.
   - Розробка структури програмного забезпечення 03.11.2021р.
   - Оформлення пояснювальної записки, графічних матеріалів і презентації 08.12.2021р.

   6.2 Графічні матеріали:
   - Use-case діаграми 01.12.2021р.
   - Вигляд екранів розробленого додатку 03.12.2021р.

7. Порядок контролю і приймання.
   7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «17» грудня 2021 р.
   7.2. Атестація проекту здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «17» грудня 2021 р.
   7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК.
   7.4. Захист магістерської кваліфікаційної роботи провести до «29» грудня 2021 р.

ДОДАТОК Б
ЛІСТИНГИ

Код програми

Код серверу

```
const path = require('path');

const { app, BrowserWindow } = require('electron');
const isDev = require('electron-is-dev');

function createWindow() {
  // Create the browser window.
  const win = new BrowserWindow({
    width: 800,
    height: 700,
    minWidth: 800,
    minHeight: 700,
    webPreferences: {
      nodeIntegration: true,
      webSecurity: false
    }
  });

  win.loadURL(
    isDev
      ? 'http://localhost:3000'
```

```
      : `file://${path.join(__dirname,  '../build/index.html')}`
  );
}

app.whenReady().then(createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length  === 0) {
    createWindow();
  }
});
```

Код меню упрвління резервацією номерів

```
import React, { useEffect } from 'react';
import { useStateWithCallbackLazy  } from 'use-state-with-callback';
const useState = useStateWithCallbackLazy

const CheckBox = (props) => {
    console.log('CheckBox', props)
    const [btnState, setBtnState] = useState(false)
    const [state, setState] = useState({
        css: {
```

```
        container: {

        },
        label: {
            backgroundColor: btnState ? "#424242" : "#ffffff",
            border: btnState ? "2px solid #ffffff" : "2px solid #424242"
        },
        circle: {
            backgroundColor: btnState ? "#ffffff" : "#424242",
        }
    },
})

useEffect(() => {
    console.log('componentDidMount')
    changeBtnState(false)
}, [])

const changeBtnState = (bool) => {
    setBtnState(bool, () => {
        setState({
            css: {
                container: {

                },
                label: {
                    backgroundColor: btnState ? "#ffffff" : "#424242",
                    border: btnState ? "2px solid #424242" : "2px solid #ffffff"
                },
```

```
            circle: {
                backgroundColor: btnState ? "#424242" : "#ffffff"
            }
        },
    })
})
props.setNightMod(!bool)
}

return (
    <>
        <input id="custom-checkbox" type="checkbox" onChange={() =>
changeBtnState(!btnState)} />
        <label htmlFor="custom-checkbox">
            <div className={`custom-checkbox-label night-${btnState ? 'off' : 'on'}`}
style={state.css.label}>
                <div className="circle" style={state.css.circle}></div>
            </div>
        </label>
<input id="custom-checkbox" type="checkbox" onChange={() =>
changeBtnState(!btnState)} />
        <label htmlFor="custom-checkbox">
            <div className={`custom-checkbox-label night-${btnState ? 'off' : 'on'}`}
style={state.css.label}>
                <div className="circle" style={state.css.circle}></div>
            </div>
        </label>
<input id="custom-checkbox" type="checkbox" onChange={() =>
changeBtnState(!btnState)} />
        <label htmlFor="custom-checkbox">
```

```
            <div className={`custom-checkbox-label night-${btnState ? 'off' : 'on'}`}
style={state.css.label}>
                <div className="circle" style={state.css.circle}></div>
            </div>
        </label>

    </>
  )
}

export default CheckBox;
```

Головний клас програми

```
import React, { useEffect } from 'react';
import { useStateWithCallbackLazy } from 'use-state-with-callback';
import './App.css';
import Header from './components/Header';

const useState = useStateWithCallbackLazy

function App() {
 const [nightMod, setNightMod] = useState(false)
 const [state, setStyle] = useState({
   css: {
     backgroundColor: nightMod ? '#424242' : '#ffffff'
   }
 })
```

```
useEffect(() => {
  document.querySelector('.preloader').style.display = "none"
})

const changeNightMod  = (bool) => {
  console.log(bool)
  setNightMod(bool, () => {
    setStyle({
      css: {
        backgroundColor: nightMod  ? '#424242' : '#ffffff'
      }
    }, () => {
      console.log('bg changed')
    })
  })
}

return (
  <div className="App" style={state.css} >
    <Header setNightMod={changeNightMod}  />
        <div className="header-room-manager">
                <div className="nav-menu">
                    <div className="home"></div>
                    <div className="quick-access"></div>
                    <div className="tools"></div>
                    <CheckBox {...props} />
                </div>
          </div>
        <div className="header-room-manager">
```

```
                <div className="nav-menu">
                    <div className="home"></div>
                    <div className="quick-access"></div>
                    <div className="tools"></div>
                    <CheckBox {...props} />
                </div>
            </div>

        <div className="header-room-manager">
                <div className="nav-menu">
                    <div className="home"></div>
                    <div className="quick-access"></div>
                    <div className="tools"></div>
                    <CheckBox {...props} />
                </div>
            </div>
        <div className="header-room-manager">
                <div className="nav-menu">
                    <div className="home"></div>
                    <div className="quick-access"></div>
                    <div className="tools"></div>
                    <CheckBox {...props} />
                </div>
            </div>
    </div>
  )
}

export default  App;
```

Додаток В
(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизованої системи управління адміністративними функціями готелю

Перелік ілюстративних матеріалів:

1. Use-Case діаграма вікна головного меню програми
2. Use-Case діаграма вікна адміністрування номерів готелю
3. Use-Case діаграма вікна адміністрування ресторану готелю
4. Use-Case діаграма вікна налаштувань
5. Екран головного меню
6. Екран адміністрування номерів готелю
7. Екран адміністрування ресторану готелю
8. Екран авторизації
9. Екран невдалої авторизації
10. Екран загальних налаштувань системи

Виконав: студент 2-го курсу, групи
2АКІТ-20м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

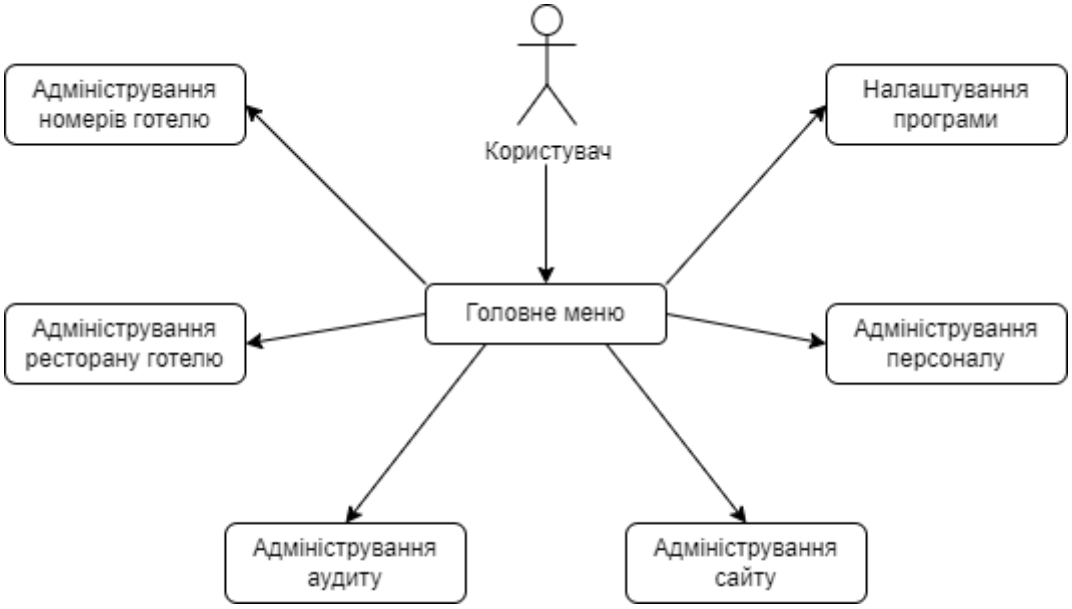
_____Владислав Таламанюк
(ім'я та прізвище)

Керівник: к.т.н., доцент каф. КСУ

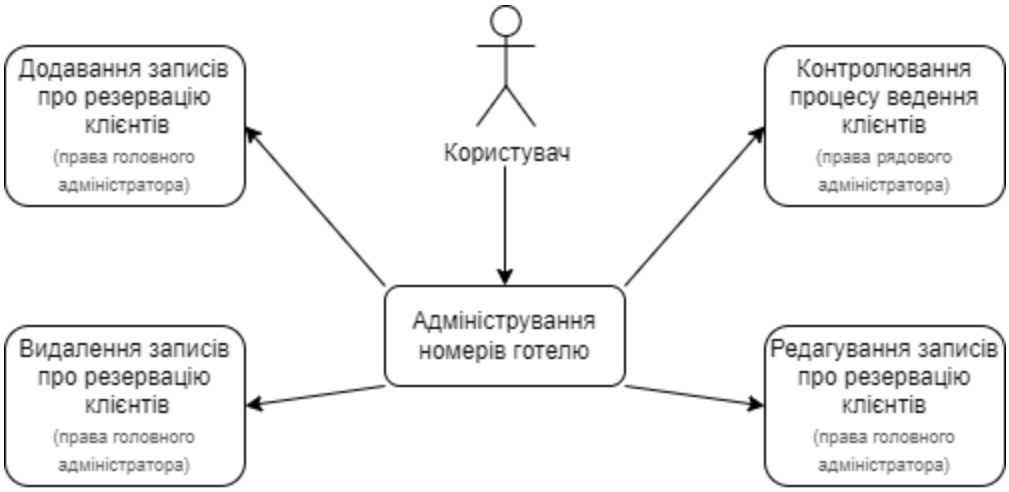
_____Марія Юхимчук
(ім'я та прізвище)
«_15_» _____12_____ 2021 р.

Опонент: к.т.н., доцент каф. АІІТ

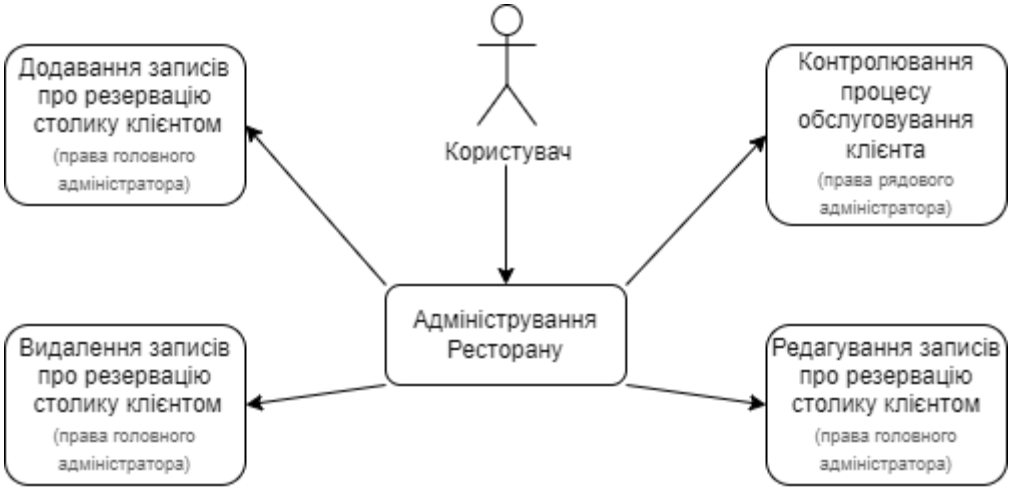
_____Марія Барабан
(ім'я та прізвище)
«_16_» _____12_____ 2021 р.



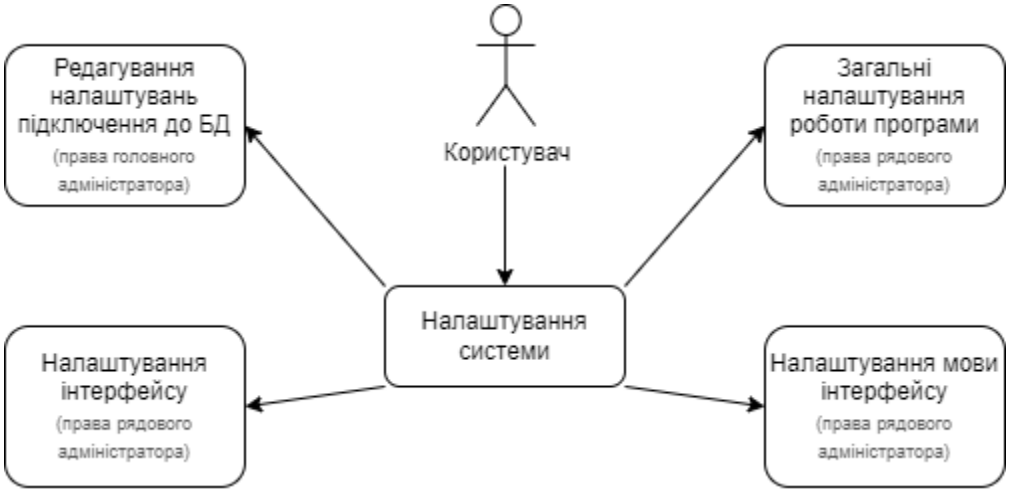
Use-Case діаграма вікна головного меню програми



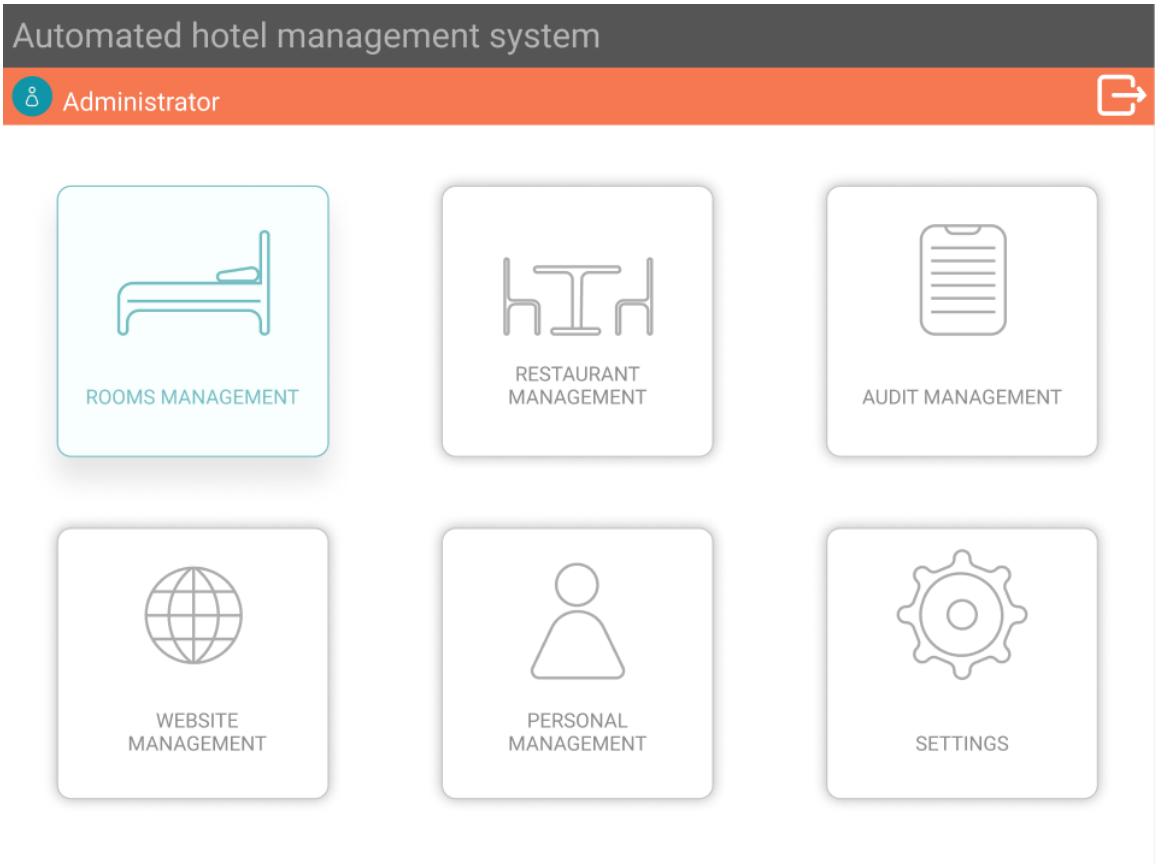
Use-Case діаграма вікна адміністрування номерів готелю



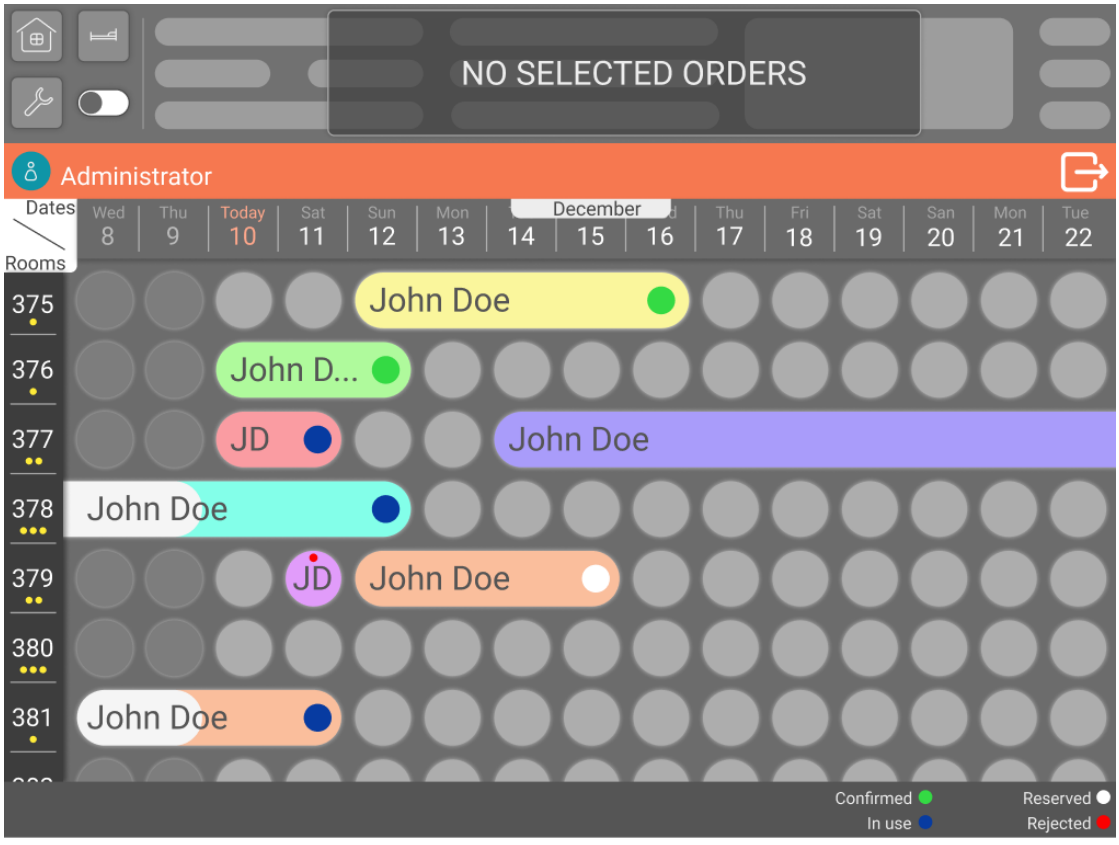
Use-Case діаграма вікна адміністрування ресторану готелю



Use-Case діаграма вікна налаштувань

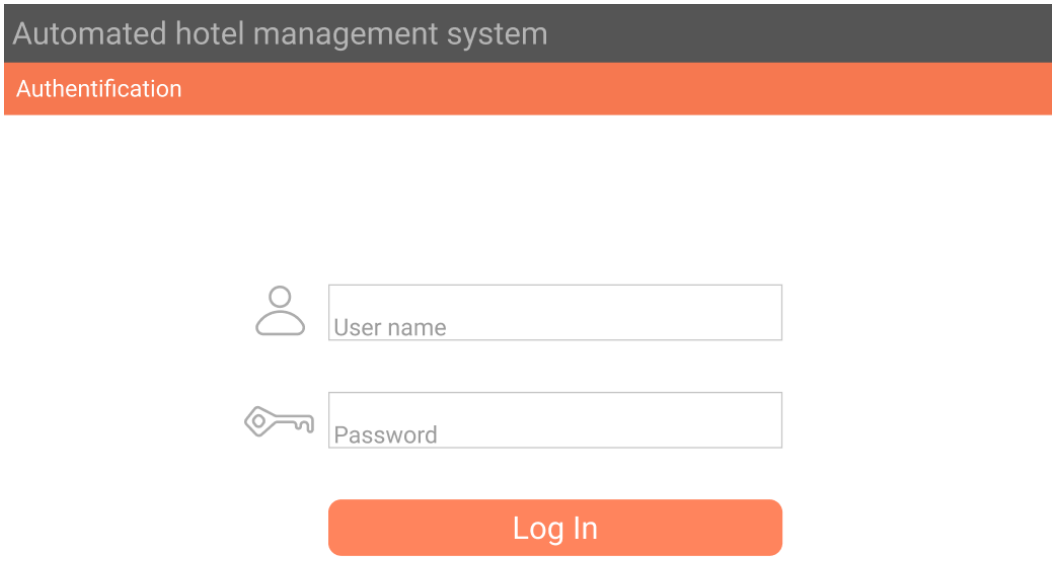


Екран головного меню

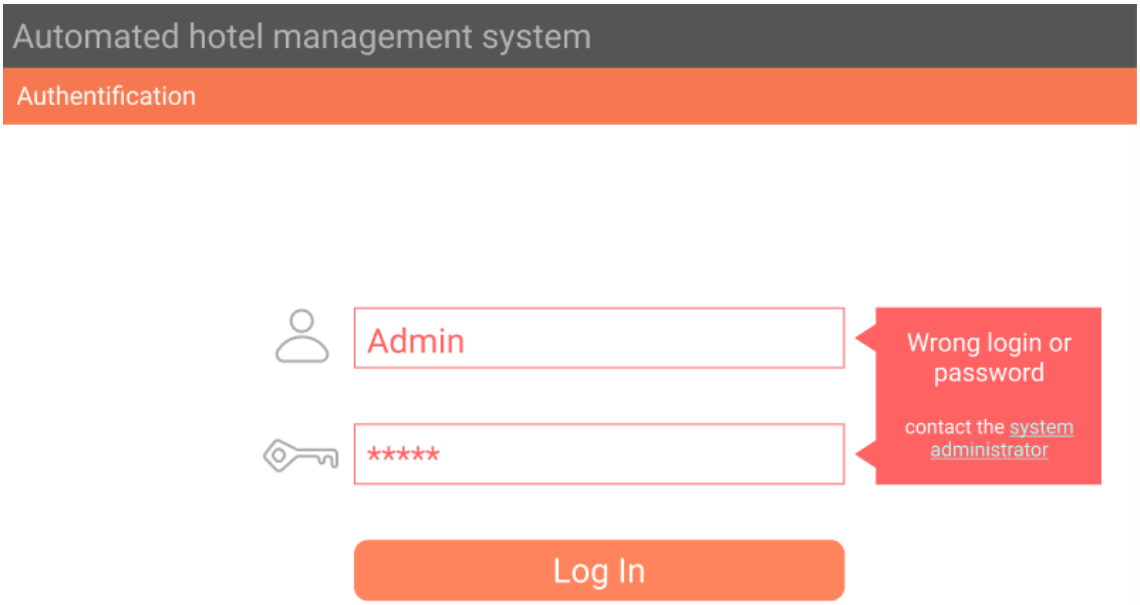


Екран адміністрування номерів готелю

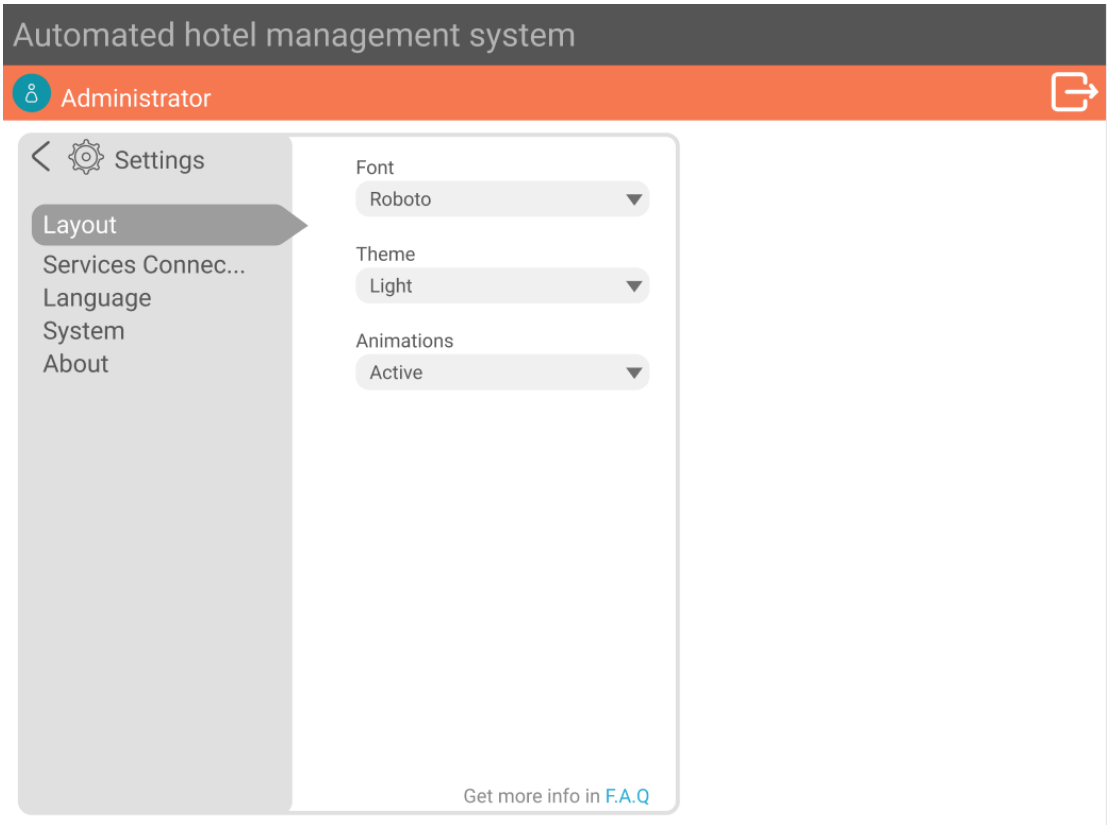
Екран адміністрування ресторану готелю



Екран авторизації



Екран невдалої авторизації



Екран загальних налаштувань системи