Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти-магістр
за освітньо-професійною програмою
«Кіберфізичні системи в промисловості, бізнесі та транспорті»
зі спеціальності
174 – Автоматизація, комп'ютерно-інтегровані технології
та робототехніка

Тема роботи:
«Автоматизація та оптимізація процесів керування прийому/відправлення літаків і обслуговування пасажирів»

Виконав студент гр.АКІТР 23-1м Семенов І. А.

Нормоконтроль Маринич І.А.

Керівник Тиханський М.П.

Завідувач кафедри Рубан С.А.

Кривий Ріг – 2024

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Магістр*

Спеціальність: *174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка*

Затверджую

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *26* » *червня* 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра

студентові групи АКІТР-23-1м *Семенову Іллі Андрійовичу*

1. Тема кваліфікаційної роботи: *«Автоматизація та оптимізація процесів керування прийому/відправлення літаків і обслуговування пасажирів»*

затверджено наказом по університету № *595с* від *04.07.2024 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2024 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка, додатки, презентація у Microsoft PowerPoint в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-3 доц. Рубан С. А.

Нормоконтроль доц. Маринич І. А.

Керівник доц. Тиханський М.П

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.07.24*
2	*Розділ 1*	*15.07.24*
3	*Розділ 2*	*18.08.24*
4	*Розділ 3*	*19.09.24*
5	*Висновки*	*15.10.24*
6	*Оформлення кваліфікаційної роботи*	*20.11.24*
7	*Підготовка презентації та графічного матеріалу*	*28.11.24*
8	*Підготовка доповіді до захисту*	*01.12.24*

6. Дата видачі завдання: *28.06.2024р.*

Керівник _____/Тиханський М.П. /

7. Запевнення: *Я, Семенов Ілля Андрійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету з метою запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____/ Семенов Ілля Андрійович

АНОТАЦІЯ

Семенов І. А. Автоматизація та оптимізація процесів керування прийому/відправлення літаків і обслуговування пасажирів, 2024 р.

Дана магістерська робота присвячена дослідженню автоматизації та оптимізації процесів керування прийому/відправлення літаків і обслуговування пасажирів.

У першому розділі розглянуто теоретичні основи управління авіаційними процесами. Також було розглянуто сучасні підходи до автоматизації процесів обслуговування пасажирів на авіаційному транспорті. Окрім цього було проаналізовано вже існуючі системи управління прийомом та відправленням літаків та обслуговування пасажирів.

У другому розділі був виконаний опис основ математичного моделювання в авіаційній галузі при обслуговуванні пасажирів. Також для подальшої розробки був проведений аналіз обраного об’єкту. Було зроблено проєктування системи з діаграмами та їх компонентами. В рамках інформаційного забезпечення системи були описані функціональні вимоги, можливості кожного користувача та визначені їх ролі. Також було представлено дії для кожного користувача. Опис бази даних включає її структуру, що складається з трьох таблиць та зв’язків між ними.

У третьому розділі розглянуто програмне забезпечення, використане для розробки системи (HTML, CSS, JS, React та Amadeus). Також проаналізовано технічні вимоги, зокрема характеристики ПК та програмні засоби, необхідні для запуску програми. Наведено опис програми та її основних функцій. Крім того, було проведено тестування системи для перевірки коректності роботи її функціоналу.

АВТОМАТИЗАЦІЯ, ОПТИМІЗАЦІЯ, МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ, СИНТЕЗ СИСТЕМ, АДАПТИВНЕ КЕРУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, ТЕСТУВАННЯ ФУНКЦІОНАЛУ, ВЕБ-ЗАСТОСУНОК, ІНТЕГРАЦІЯ, АНАЛІЗ ПАРАМЕТРІВ.

ANNOTATION

Semenov I. A. Automation and optimization of aircraft reception/departure control processes and passenger service, 2024.

This master's thesis is devoted to the study of automation and optimization of aircraft reception/departure control processes and passenger service.

The first section considers the theoretical foundations of aviation process management. Modern approaches to the automation of passenger service processes in air transport were also considered. In addition, existing aircraft reception and departure control systems and passenger service were analyzed.

The second section describes the basics of mathematical modeling in the aviation industry when servicing passengers. Also, for further development, an analysis of the selected object was carried out. A system design with diagrams and their components was made. Within the framework of the information support of the system, the functional requirements, capabilities of each user were described and their roles were defined. Actions for each user were also presented. The description of the database includes its structure, which consists of three tables and the relationships between them.

The third section discusses the software used to develop the system (HTML, CSS, JS, React and Amadeus). It also analyzes the technical requirements, including PC specifications and software required to run the program. It describes the program and its main functions. In addition, the system was tested to verify the correctness of its functionality.

AUTOMATION, OPTIMIZATION, MATHEMATICAL MODELING, SYSTEM SYNTHESIS, ADAPTIVE CONTROL, INFORMATION SYSTEM, FUNCTIONAL TESTING, WEB APPLICATIONS, INTEGRATION, PARAMETER ANALYSIS.

ЗМІСТ

ВСТУП

Актуальність. Актуальність теми магістерської роботи «Автоматизація та оптимізація процесів керування прийому/відправлення літаків і обслуговування пасажирів» обумовлена сучасними тенденціями розвитку авіаційної індустрії, яка стикається зі стрімким зростанням пасажиропотоку та підвищеними вимогами до ефективності й безпеки авіаперевезень.

В умовах глобалізації та збільшення попиту на авіапослуги, забезпечення своєчасного обслуговування літаків та пасажирів стає важливим фактором конкурентоспроможності авіакомпаній і аеропортів. Автоматизація та оптимізація цих процесів дозволяє значно скоротити час на обробку рейсів, зменшити витрати, покращити якість обслуговування клієнтів, а також підвищити загальний рівень безпеки операцій.

В магістерській роботі було проведено аналіз технологій та інструментів, пов'язаних із розробкою програмного забезпечення, включаючи UML-методи моделювання, засоби програмування на JavaScript, а також пошук інформації в інтернеті та літературних джерелах.

Засоби розробки – середовище розробки Visual Studio Code, HTML, CSS, мова програмування JavaScript, React, MongoDB, Amadeus for Developers.

Мета дослідження. Автоматизація та оптимізація процесів керування прийому/відправлення літаків і обслуговування пасажирів, а саме розробка сучасного вебдодатку, який забезпечить автоматизацію процесів бронювання авіаквитків та ефективне управління авіаційними послугами. Основна увага приділяється аналізу і вибору найбільш ефективних методів та інструментів для створення такої системи, яка буде зручною, функціональною та здатною оптимізувати управлінські процеси.

Задачі дослідження:

1. Аналіз існуючих підходів та інструментів, які можуть бути застосовані для розробки автоматизованої системи управління.
2. Проєктування бази даних із урахуванням специфіки роботи авіаційної галузі.

3. Розробка функціонального вебдодатку, який забезпечує інтерактивну взаємодію користувачів із системою.

4. Забезпечення зручності та простоти використання для пасажирів і адміністраторів.

Об’єкт дослідження. Процес бронювання квитків як ключова операція в системі обслуговування пасажирів аеропорту.

Предмет дослідження. Програмне забезпечення для автоматизації управління авіаційними послугами.

Очікувані результати роботи включають створення програмного забезпечення, яке дозволить автоматизувати ключові операції аеропорту, підвищити якість обслуговування пасажирів і знизити час очікування під час реєстрації та посадки. Отримані результати можуть бути використані для впровадження в аеропортах середнього масштабу, таких як міжнародний аеропорт "Кривий Ріг", що сприятиме його розвитку та конкурентоспроможності.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ

1.1 Теоретичні основи управління авіаційними процесами

Управління авіаційними процесами є важливим і комплексним завданням, яке включає різні аспекти організації роботи аеропортів та авіакомпаній. Основною метою цього управління є забезпечення ефективної, безпечної та надійної роботи авіаційної системи.

Управління починається з детального планування, яке охоплює розклад рейсів, технічне обслуговування літаків, а також управління ресурсами, включаючи персонал і обладнання. Це дозволяє уникати простоїв та забезпечувати високу продуктивність роботи аеропортів і авіакомпаній[1].

Ключовим елементом управління є ефективна координація між диспетчерами, наземним обслуговуванням і адміністрацією. Інформаційні системи допомагають контролювати виконання операцій у реальному часі. Насамперед це слугує мінімізації людських помилкок. Диспетчери АТС надають інструкції пілотам задля забезпечення безпечного пересування між літаками, направляють їх уздовж визначених траєкторій польоту та керують транспортним потоком у контрольованому повітряному просторі. Ефективне управління повітряним простором передбачає розподіл і використання повітряного простору для оптимізації пропускної спроможності та мінімізації заторів, забезпечуючи плавний потік повітряного руху[2].

Основні підходи до управління:

1. Традиційні моделі. Включають централізоване управління та чітку ієрархію прийняття рішень, що все ще використовується на багатьох підприємствах, зокрема в державному секторі.

2. Інноваційні підходи. Використовують інтегровані автоматизовані системи управління, що підвищує точність і швидкість прийняття рішень[3].

Від початку польоту повітряного судна до його безпечного прибуття в пункт призначення авіаційна експлуатація та керівництво здійснюють комплекс дій і рішень. Авіаційні операції та управління охоплюють багато завдань, пов’язаних із забезпеченням безпечного та ефективного переміщення літаків, пасажирів і вантажів у глобальній системі повітряного простору. Це може включати управління повітряним рухом, наземні операції, польоти та завдання з технічного обслуговування[4].

Інтеграція автоматизації та штучного інтелекту (ШІ) має величезний потенціал для революції в експлуатації та управлінні авіацією. Від автономних систем літака та алгоритмів прогнозного технічного обслуговування до рішень управління повітряним рухом на основі ШІ, автоматизація та технології ШІ обіцяють підвищити безпеку, ефективність і прийняття рішень в авіації.

Автоматизуючи рутинні завдання, оптимізуючи польотні операції та покращуючи прогнозну аналітику, авіаційні організації можуть відкрити новий рівень продуктивності та ефективності, одночасно зменшуючи експлуатаційні витрати та людські помилки.

1.2 Сучасні підходи до автоматизації процесів обслуговування пасажирів на авіаційному транспорті

Відомо, що авіаційна галузь швидко впроваджує нові технології для задоволення потреб пасажирів. У майбутньому будуть тенденції та інновації, які змінять спосіб бронювання квитків і досвід польоту. Повітряний транспорт виглядає досить багатообіцяючим для підвищення ефективності, стійкості та простоти доступу. Працюючи разом, дослідники, інженери та авіаційні ентузіасти змінюють наше традиційне розуміння авіаперельотів.

У майбутньому персонал з технічного обслуговування літаків робитиме більше прогнозів щодо проблем, які виникнуть з різними компонентами та обладнанням літака. Вони використовуватимуть прогнозне обслуговування на основі IoT, за допомогою якого вони збиратимуть дані з датчиків, підключених

до машин. Це допоможе їм знати заздалегідь, що незабаром може виникнути проблема з якоюсь частиною машини. Ще одна послуга, яку підтримує IoT, — це доставка критичних даних із частин літака до команди технічного обслуговування. Це допоможе проводити обслуговування, допомагаючи технікам у плануванні графіків, придбанні запчастин і координації ремонту обладнання з кваліфікованими працівниками.

Розумні аеропорти – це аеропорти нового покоління, які користуються такими передовими технологіями, як штучний інтелект, Інтернет речей, аналіз даних і автоматизація, щоб оптимізувати роботу, підвищити безпеку, покращити пасажирські авіаперевезення та віддати пріоритет стійкості. Це досягається шляхом встановлення розумних систем, датчиків і пристроїв по всьому аеропорту, щоб створити централізований цифровий хаб для контроля та планування.

Ключовими сферами цифрової трансформації, які перетворять аеропорти на розумні аеропорти, є:

1. Цифровізація пасажиропотоку:

Аеропорти поступово використовують комп’ютеризовані системи пасажиропотоку для підвищення ефективності та задоволеності клієнтів. Це включає використання передових технологій для обробки даних у реальному часі.

2. Цифровізація інфраструктури:

Щоб сприяти ефективній комунікації та обміну даними, аеропорти починають інтегрувати свої ресурси та інфраструктуру в цифровому вигляді. Він гарантує безперебійне з’єднання та прийняття обґрунтованих рішень, роблячи роботу аеропорту більш зв’язаною та ефективною.

3. Цифрування комерційних потоків:

Аеропорти все частіше звертаються до цифровізації, яка використовує технології для оптимізації руху продуктів і послуг через аеропорт. Він також включає в себе автоматизацію процесів і автоматизацію процедур.

4. Біометрія та персоналізація:

Біометричні технології, такі як розпізнавання обличчя, відбитки пальців і сканування сітківки ока, модернізують і спрощують процес посадки в літак, одночасно підвищуючи безпеку. Наприклад, програмне забезпечення для розпізнавання обличчя порівнює обличчя мандрівника з базою даних підтверджених осіб, зменшуючи потребу в паперових документах і спрощуючи реєстрацію, перевірку безпеки та посадку. Зараз біометричні системи інтегруються в імміграційну службу, прикордонний контроль, реєстрацію в аеропорту та посадку на борт, поступово переходячи до безпаперового процесу перевірки. Цей перехід також охопить отримання багажу, дозволяючи мандрівникам контролювати свій багаж у режимі реального часу [4].

Попит на авіаперельоти значно зростає, що призводить до надмірного пасажиропотоку в аеропортах, які обслуговують відомі напрямки. Процедури керування пасажирами в терміналі під час реєстрації та імміграційних формальностей заважають користувачам аеропорту продуктивно використовувати свій час в аеропорту. Як наслідок, технології самообслуговування (SST), пов’язані біометричні технології та автоматизований прикордонний контроль (ABC) широко впроваджуються для вирішення процесу обслуговування пасажирів в аеропорту [4].

Технології (SSTs) допомагають пасажирам авіаперевізників самостійно виконувати свої дії реєстрації як перший крок до фінального етапу, коли вони сідають на літак через автоматизований канал. Відповідно, авіакомпанії можуть отримати вигоду від зменшення своїх фіксованих витрат на своїх працівників або агентів наземного обслуговування. Навпаки, контроль на кордоні, який активно здійснюється державними імміграційними службовцями, може бути сприянням аеропортом через автоматизований контроль на кордоні (ABC), що використовує біометричну ідентифікацію, а саме розпізнавання обличчя та райдужної оболонки ока, а також безконтактну технологію.

Аналізуючи такий період, як спалах пандемії коронавірусу (COVID-19), пасажири авіакомпаній та користувачі аеропортів отримували допомогу через самостійно працюючі технології, включаючи безконтактну технологію.

Інновація значно допомагає мандрівникам та користувачам аеропортів уникати поширення вірусу. Таким чином, залучені оператори послуг в обслуговуванні пасажирів, такі як авіаперевізники, компанії наземного обслуговування та аеропорти, значно враховують важливість SSTs [5].

Процес обслуговування пасажирів є надзвичайно важливим для управління аеропортом, зокрема для управління чергами. При відправленні з основних аеропортів світу висока щільність пасажирів може затримувати процес обслуговування пасажирів, що призводить до додаткових витрат авіакомпаній та негативної репутації аеропорту. Звичайні процедури реєстрації передбачають особисті взаємодії між працівниками авіакомпаній або агентами наземного обслуговування. Однак наразі взаємодії пасажир-машина значно замінили звичайні взаємодії агент- пасажир. Більше того, функція самообслуговування відігравала важливу роль у запобіганні передачі коронавірусу під час пандемії COVID-19 [5].

Кіоск самообслуговування для реєстрації в аеропорту наразі використовується в різних аеропортах з високим трафіком пасажирів. Кіоски для реєстрації допомагають авіакомпаніям зменшити кількість працівників та ресурси інфраструктури аеропорту, необхідні на стійці реєстрації. Таким чином, авіаперевізники економлять експлуатаційні витрати, ефективно управляють часом очікування в черзі та підвищують продуктивність персоналу. Основна мета кіоска самообслуговування полягає в тому, щоб альтернативно надати послугу пасажирам авіакомпаній замість взаємодій з агентами. Зазвичай фізичний кіоск для реєстрації складається з сенсорного екрану, сканера паспортів, зчитувача кредитних карток, а також принтера посадкових талонів і багажних бирок.[6]

Кіоски самообслуговування для реєстрації можуть не бути включені в прийом багажу в певних аеропортах; пасажири зобов'язані витрачати свій час на стійці прийому багажу. Внаслідок цього кілька аеропортів встановили автоматизовані стійки для здачі багажу, що дозволяє пасажирам скоротити

процедури реєстрації та мати більше часу для огляду безмитних магазинів аеропорту.

Біометричні технології ідентифікації можна визначити як розпізнавання обличчя або райдужної оболонки, а також ідентифікацію за відбитками пальців. Проте малоймовірно, що технології біометрії замінять технології самообслуговування, які в основному доступні в зонах відправлення або реєстрації. З іншого боку, дані біометричної ідентифікації можуть бути зібрані, коли виражається план подорожі, наприклад, під час придбання авіаквитка. Збережені біометричні дані пасажирів дозволяють прискорити процес обслуговування пасажирів на відправленні в аеропорту. Зокрема, з великою кількістю пасажирів у завантажених аеропортах, пасажири обслуговуються своєчасно, що призводить до ефективного управління потоком пасажирів[7].

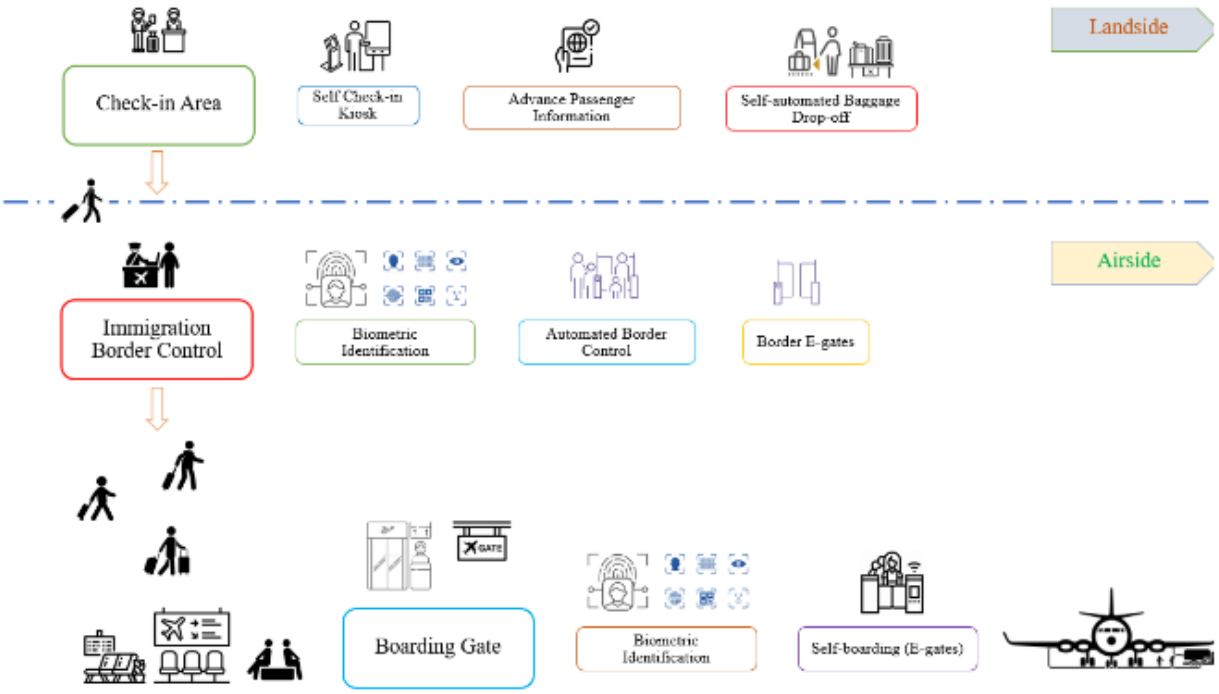

Рисунок 1.1 – Подорож пасажира через автоматизовані технології в процесі обслуговування пасажирів в аеропорту

Автоматизовані технології в процесі обслуговування пасажирів суттєво допомагають аеропортам ефективно управляти потоком пасажирів. Інновація

також дозволяє аеропортам підвищити свої компетенції як постачальників послуг, тим самим підвищуючи репутацію аеропортів. Більше того, рейтинг аеропорту сприяє залежності авіакомпаній-клієнтів від якості послуг аеропорту та ефективності обслуговування пасажирів. Доходи аеропорту можуть бути підвищені за допомогою раніше згаданих підходів.

Безсумнівно, що авіаперевізники суттєво досягають успіху в операціях зі зниження фіксованих витрат, зокрема, на персонал та вимоги до стійок. Незважаючи на всі вищезгадані переваги, аеропорти, авіакомпанії та їхні зацікавлені сторони повинні інвестувати значну кількість капіталу в розвиток технологій.

На противагу цьому, з обмеженнями постачальників автоматизованих сервісних технологій, зацікавлені сторони будуть терпіти повільнішу, ніж необхідно, технологію. З іншого боку, інвестори в технології повинні зосередитися в основному на даних про конфіденційність клієнтів, щоб запобігти витоку даних. Деякі аеропорти все ще не мають достатньої інфраструктурної бази і можуть не реагувати продуктивно на передові технології. Крім того, певні SST не відповідають критеріям усіх спеціальних категорій пасажирів, включаючи неповнолітніх без супроводу (UMNR), новачків у польотах, літніх пасажирів та пасажирів з обмеженою мобільністю. Інвестори в SST повинні розробити ідеальні функціональні інновації для полегшення обслуговування спеціальних категорій пасажирів. Аналогічно, технології самообслуговування повинні бути зручними для користувачів і спрощеними, особливо щодо літніх пасажирів.

Можна зробити висновок, що автоматизовані технології в процесі обслуговування пасажирів в аеропорту повинні бути далі вивчені з різних перспектив. Оскільки обслуговування пасажирів є складною операцією, кожна окрема функція повинна відповідати вимогам пасажирів відповідно до міжнародних норм. Перш за все, аеропорти, авіакомпанії та їхні зацікавлені сторони повинні розглянути, чи є відповідні технології самообслуговування

доречними з точки зору їхнього контексту та характеристик; також інвестиційний фонд має приносити задовільну норму прибутку.

1.3 Аналіз існуючих систем управління прийомом та відправленням літаків та обслуговування пасажирів

В наш час якість обслуговування в комерційній транспортній індустрії є одним з головних конкурентних показників кожного транспортного агентства задля залучення пасажирів. Інфраструктура аеропортів розвивається попри застарілі системи та процеси, які обмежують можливість повністю реалізувати потенціал ефективності дорогих реконструкцій. Аеропорти продовжують використовувати сучасні технології. Проте, щоб такі системи працювали максимально ефективно, пасажир повинен відігравати активну роль.

В цьому підрозділі буде проаналізовано , як сучасні процеси автоматизації обслуговування пасажирів впливають на розвиток аеропортів. Тут буде розглянуто один із типів автоматизації, який в цілому можна назвати «розумним пасажиром». В досліді буде проаналізовано суттєво різні типи аеропортів: Лондон Сіті (LCY), який призначений для обслуговування великої частки бізнес-пасажирів на коротких рейсах, де ефективність є ключовою для реалізації його ціннісної пропозиції; та Пальма де Майорка (PMI), яка обслуговує переважно відпочивальних пасажирів, де дуже високі піки попиту навантажують інфраструктуру та процеси. Екстремальні та протилежні характеристики цих аеропортів роблять випадки актуальними для вивчення потенційних наслідків у різних категоріях аеропортів, принаймні в Європі[10].

Впровадження цих технологій повинно прискорити обслуговування пасажирів, а також розвантажити трафік, завдяки автоматизації процесів реєстрації, оформлення багажу і біометричній ідентифікації. "Розумний пасажир" отримає ексклюзивний доступ до процесів відправлення. Цей аналіз має на меті розкрити потенційну потужність терміналу аеропорту залежно від типів пасажирів, які його використовують.

У цьому дослідженні основна увага приділяється руху пасажирів у межах терміналів аеропорту та їхній взаємодії з навколишнім середовищем. На основі плану терміналу створюється мережа локацій, яка дозволяє моделювати маршрути пасажирів і аналізувати ключові показники ефективності, що впливають на їхню подорож. Це допомагає виявити залежності між різними процесами в аеропорту.

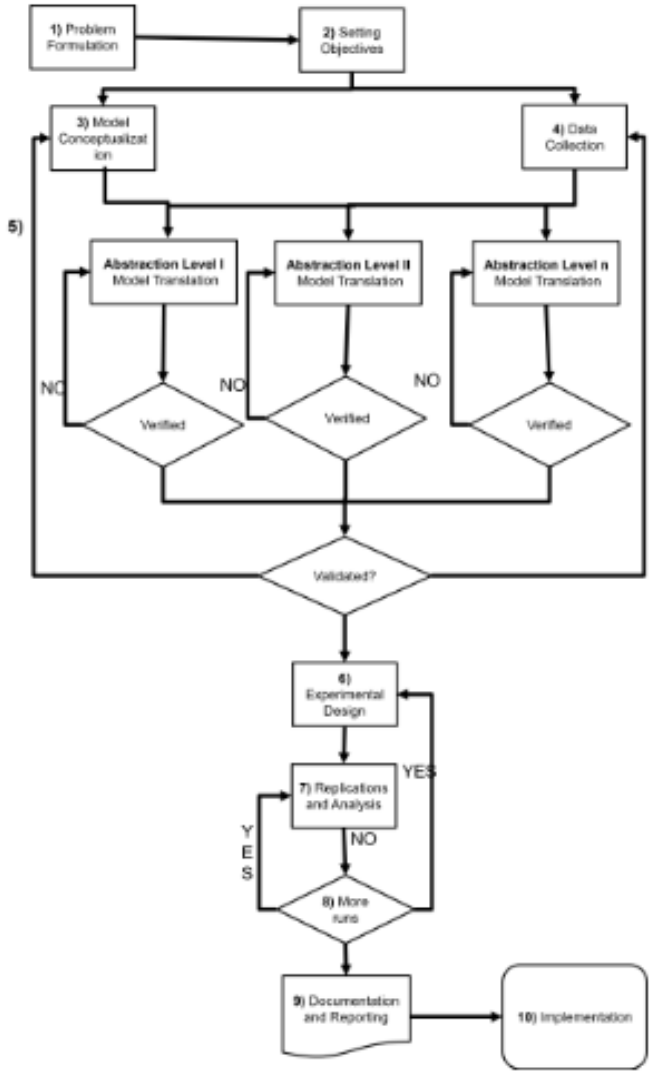

Рисунок 1.2 – Багатошарова методологія моделювання

Як показано на рис.1.2, після постановки мети та проведення перевірки й валідації моделей (V&V) розробляються сценарії, які враховують як нові характеристики пасажирів, так і особливості терміналу. У цьому дослідженні розглядається комбінація обох факторів — поведінки пасажирів та змін у

конфігурації терміналу, що дозволяє оцінити ефективність та оптимізувати процеси[10].

Ці моделі були випробувані і перевірені та є зразковими прикладами для різноманітних типів аеропортів. Оцінка цих двох випадків дає огляд переваг і впливу на аеропорти, деякі з яких здебільшого пов’язані з бізнесом, а інші орієнтовані на відпочинок. Тоді це дає точку зору на те, які важелі використовувати для покращення ефективності аеропорту залежно від профілю користувача.[10]

Для отримання бажаного результату необхідно, щоб «розумний пасажир» отримував свій посадковий талон онлайн, а в аеропорту використовував самостійно окремі багажні термінали для реєстрації свого багажа. В наслідку пасажир буде пересуватись аеропортом без багажу швидше, а також скоротить час процесу перевірки, оскільки не потрібно буде сканувати багаж і використовувати підноси для сканування своїх речей. На рис.1.3 продемонстровано, що спеціалізовані кіоски самообслуговування для здачі багажу та спеціалізовані черги для процесу безпеки можуть стимулювати «розумну» поведінку та покращити як продуктивність аеропорту, так і досвід подорожей пасажирів.

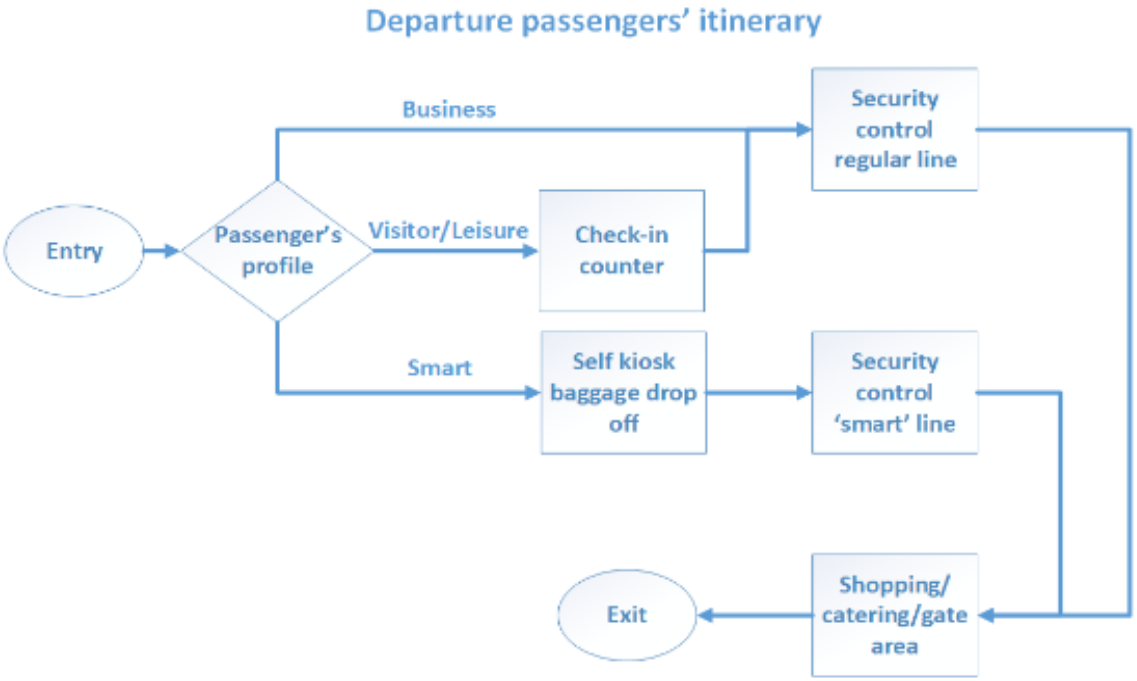

Рисунок 1.3 – Маршрут пасажирів, що відправляються, в терміналі на основі профілю пасажирів

Наступні графіки демонструють різницю між пасажирами, у цьому випадку розумні та звичайні. На рис. 1.4 показана довжина черги на реєстрацію, тут видно, що довжина черги пасажирів покращується, завдяки розумним пасажирам, а з іншого боку, довжина черги розумних пасажирів різко збільшується, коли ми збільшуємо їх відсоток на 20% і 30%. Ця ситуація не є ідеальною, оскільки це негативно вплине на користь бути розумним пасажиром. Це явище підтверджено на рис. 1.5, який показує, що час черги для розумних пасажирів зростає і це перевищує час черги звичайних пасажирів. Тому найкращий сценарій для розумних пасажирів знайдено в S1, а для звичайних пасажирів - S3. Однак компроміс можна знайти у S2, де час черги зменшився для звичайного пасажира, а розумні пасажири все ще можуть отримати вигоду меншого часу черги в порівнянні зі звичайними. Ця тенденція показує, що зі збільшенням відсотка розумних пасажирів, дієздатність для них обмежена, у наслідку довші черги на реєстрації[10].

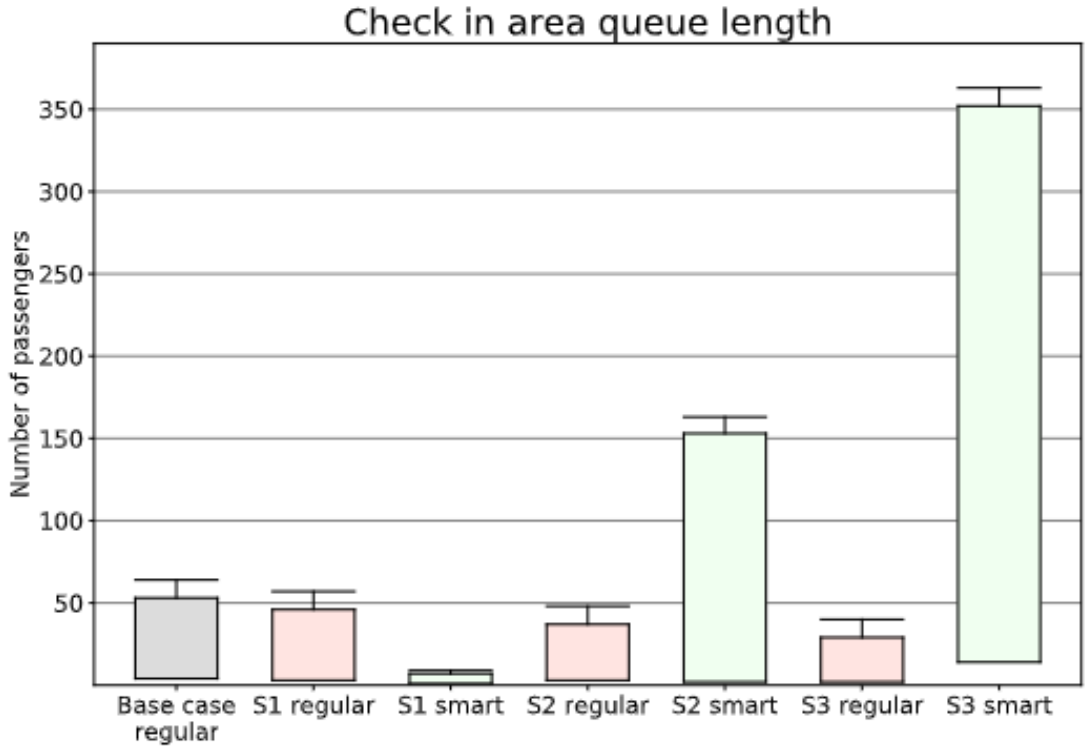

Рисунок 1.4 – Довжина черги в зоні реєстрації

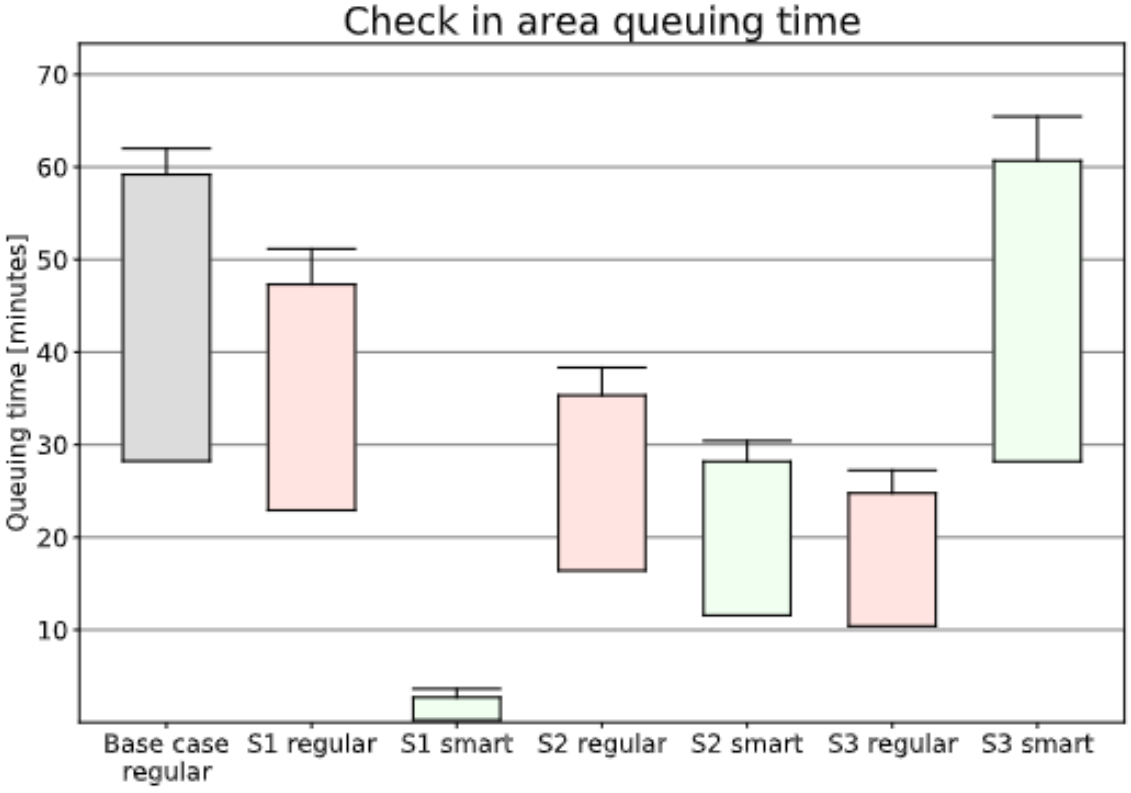

Рисунок 1.5 – Час очікування в зоні реєстрації

1.4 Оцінка ефективності обслуговування пасажирів

Оцінка ефективності обслуговування пасажирів є важливою складовою управління авіаційними процесами, оскільки дозволяє виявляти слабкі місця та оптимізувати процеси. Основними критеріями ефективності є:

1. Час обслуговування пасажирів.

Час, який пасажир проводить у чергах на реєстрацію, контроль безпеки або паспортний контроль, є важливим показником. Скорочення цього часу підвищує загальну ефективність аеропорту. Для цього використовуються моделі масового обслуговування та імітаційне моделювання[12].

2. Пропускна спроможність аеропорту.

Це визначає кількість пасажирів, яких аеропорт може обслужити за певний період часу. Імітаційні моделі, як-от ті, що застосовуються у програмному забезпеченні AnyLogic, допомагають прогнозувати, як зміни в організації вплинуть на цей показник.

3. Рівень задоволеності пасажирів.

Використання опитувань та інших зворотних зв'язків від пасажирів дозволяє оцінити якість обслуговування, що безпосередньо впливає на довіру до аеропорту та авіакомпаній. Наприклад, впровадження систем автоматизації черг з використанням штучного інтелекту може суттєво підвищити комфорт пасажирів.

4. Ефективність використання ресурсів.

Це включає оптимальне розподілення персоналу, технічних засобів і площ аеропорту, щоб мінімізувати витрати і збільшити швидкість обслуговування. Відповідно, автоматизовані системи управління допомагають контролювати і планувати робочі ресурси аеропорту більш ефективно.

Оцінка цих показників дозволяє приймати рішення про впровадження нових технологій, таких як штучний інтелект та автоматизовані системи обслуговування, для підвищення ефективності роботи аеропорту.

Помітно, що коли кількість розумних пасажирів зростає, вся система отримує вигоду. Це виявляється у зменшенні середніх і дисперсійних значень продуктивності для звичайних і розумних пасажирів, що забезпечує загально кращий рівень обслуговування для всіх пасажирів у терміналі. Ці результати свідчать про те, що технології разом із заохоченнями для розумних пасажирів у терміналах відкриють цінну ємність, що, в свою чергу, позитивно вплине на всіх пасажирів (розумних і традиційних). Це дослідження відкриває нові напрямки досліджень, які будуть вивчені авторами в майбутньому. Наприклад, враховуючи, що управління ресурсами є важливим, аналіз чутливості може допомогти визначити, яка буде правильна кількість ресурсів (наприклад, черги безпеки або каси), виділених для розумних пасажирів. Крім того, оскільки задокументовано, що варіабельність відіграє важливу роль у динамічних системах, аналіз варіації для визначення, які елементи найбільше сприяють варіабельності системи, може надати управлінське уявлення. Можливість гнучких (спільних) черг, коли кількість розумних пасажирів низька, а також інші цікаві концепції, такі як віртуальне чергування або попередньо замовлені послуги, також можуть бути оцінені. Крім того, результати проекту IMHOTEP і дані опитувань, доступні онлайн, можуть покращити точність параметрів, які моделюють поведінку[13].

Висновки до розділу

Аналіз підходів до підвищення ефективності процесу обслуговування пасажирів на авіаційному транспорті продемонстрував важливість комплексного підходу до управління та автоматизації цих процесів.

Теоретичні основи управління авіаційними процесами показують, що ефективне управління пасажиропотоками потребує використання сучасних математичних моделей та алгоритмів для прогнозування попиту, оптимізації роботи персоналу та ресурсів.

Сучасні підходи до автоматизації процесів обслуговування пасажирів демонструють значний потенціал для підвищення ефективності через впровадження систем на основі штучного інтелекту. Ці системи оптимізують управління чергами, зменшують час очікування та покращують загальне задоволення пасажирів. Важливими інструментами стають імітаційні моделі, що дозволяють тестувати різні сценарії і обирати найбільш продуктивні варіанти.

Аналіз існуючих систем управління прийомом та відправленням літаків виявив, що аеропорти, які впроваджують автоматизовані системи управління та симуляційні моделі, досягають вищої ефективності за рахунок швидшого обслуговування та оптимального використання інфраструктури. Імітаційні моделі AnyLogic та інші програмні рішення дозволяють оптимізувати процеси без необхідності втручання людини.

Оцінка ефективності обслуговування пасажирів підтверджує, що використання математичних моделей і технологій штучного інтелекту дозволяє значно скоротити час очікування, підвищити пропускну спроможність аеропортів та покращити загальний рівень комфорту пасажирів. Ці системи також забезпечують кращу управлінську гнучкість у випадку непередбачуваних подій, таких як затримки рейсів або збої в роботі. Таким чином, впровадження сучасних технологій автоматизації та математичного моделювання стає важливим інструментом для підвищення ефективності обслуговування пасажирів і управління авіаційними процесами в цілому.

РОЗДІЛ 2
ПРОЕКТУВАННЯ РОЗРОБЛЮВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Комплексний аналіз потреб та обґрунтування розробки системи

У цьому розділі ми хочемо обґрунтувати розробку вебдодатку для обслуговування пасажирів міжнародного аеропорту Кривий Ріг, використовуючи математичні моделі для оптимізації функціональних процесів. Міжнародний аеропорт Кривий Ріг, хоча і є регіональним транспортним вузлом, стикнувся з низьким рівнем автоматизації обслуговування пасажирів. Не дивлячись на призупинення роботи всіх аеропортів на території України, ми маємо на меті розробити сучасну онлайн-платформу, яка включатиме:

— перегляд рейсів у реальному часі;

— онлайн-бронювання квитків;

— популяризацію туристичних напрямків та спеціальних пропозицій.

Розроблений вебзастосунок дозволить автоматизувати обслуговування пасажирів і вирішити кілька ключових завдань:

— підвищення зручності користування послугами аеропорту;

— зменшення навантаження на персонал завдяки автоматизації процесів;

— стимулювання попиту на рейси через акційні пропозиції;

— створення платформи для довготривалого зростання пасажиропотоку.

Мешканці Кривого Рогу мають обмежений вибір міжнародних рейсів і потребують більш доступних цифрових інструментів для подорожей. Наявні сервіси обслуговування пасажирів обмежуються стандартними офлайн-процедурами. Інтеграція вебзастосунку створить цифрову основу для розширення спектра послуг аеропорту.

Для розробки системи необхідно здійснити аналіз, визначити функціональні вимоги та особливості реалізації різних процесів, які будуть виконуватись. Тому далі буде продемонстровано і проаналізовано дані щодо

пасажиропотоку, обробки вантажів, а також рейсів у період з 2018 по 2023 рр. Цей аналіз допоможе виявити потреби у розробці веб-додатку[16-17].

Таблиця 2.1 – Вантажо- та пасажиропотік КП “Міжнародний аеропорт Кривий Ріг”

Показники	2018	2019	2020	2021	2022	2023
Обслуговано пасажирів, осіб	219644	21329	341	2403	0	0
Оброблено вантажів, пошти, т.	19,3	33,0	11,8	4,9	0	0
Обслуговано рейсів	706	1051	527	788	0	0

У період з 2018 по 2019 рр аеропорт демонстрував стабільну роботу. Кількість обслугованих пасажирів незначно зменшилася: з 21964 осіб у 2018 році до 21329 осіб у 2019 році. Аналогічна динаміка спостерігається у кількості рейсів, які збільшились із 706 до 1051. Це свідчить про зростання попиту на послуги аеропорту в регіоні.

Суттєве падіння відбулось у 2020 році через пандемію COVID-19 . А саме різке зниження пасажиропотоку до 341 особи (майже у 62 рази порівняно з 2019 роком). Ці зміни можна пояснити глобальними обмеженнями на авіаперельоти. Подібна ситуація відображається і в кількості оброблених вантажів: з 33 тонн у 2019 році до 11,8 тонн у 2020 році.

У 2021 році спостерігається часткове відновлення роботи: кількість обслугованих пасажирів зросла до 2403 осіб, а рейсів – до 788. Проте цей рівень все ще значно нижчий, ніж до пандемії.

З 2022 року аеропорт припинив роботу через повномасштабне вторгнення рф в Україну. Усі авіаційні процеси були зупинені з міркувань безпеки, що підтверджується нульовими показниками в таблиці.

Показники діяльності аеропорту прямо залежать від глобальних та регіональних подій. Глобальні кризи, як-от пандемія COVID-19, суттєво впливають на авіаційні перевезення. Регіональні конфлікти, як війна, призводять до повної зупинки діяльності. Аеропорту важливо буде відновлювати роботу та пасажиропотік, орієнтуючись на нові виклики. Цифровізація та впровадження вебзастосунків (наприклад, автоматизація бронювання квитків) можуть стати ключовим фактором у післявоєнному відновленні.

До повномасштабного вторгнення обслуговування пасажирів у багатьох регіональних аеропортах України, включно з Кривим Рогом, відбувалось переважно в офлайн-режимі. Це створює такі проблеми:

— високі витрати часу пасажирів на отримання інформації про рейси, реєстрацію та бронювання;

— залежність від людського фактора, що може призводити до помилок. Вебзастосунок усуне ці недоліки, надавши можливість пасажирам отримувати всі послуги дистанційно та швидко.

Кривий Ріг не є великим хабом, і його пасажиропотік значно нижчий, ніж у провідних аеропортах України (Бориспіль, Одеса). Впровадження вебзастосунку дозволить:

— стимулювати попит на авіаперевезення за рахунок зручності доступу до рейсів та акційних пропозицій;

— залучити нових користувачів, які звикли до цифрових сервісів для подорожей.

Адміністрація аеропорту отримає такі переваги:

— централізоване управління рейсами через систему, що спрощує додавання, редагування та видалення інформації;

— аналітичні інструменти, які допоможуть оцінювати попит на рейси, оптимізувати розклад та запроваджувати акційні пропозиції.

Вебзастосунок стане стратегічним інструментом для автоматизації та модернізації роботи міжнародного аеропорту Кривий Ріг. Його впровадження

сприятиме підвищенню зручності для пасажирів, оптимізації ресурсів аеропорту та стимулюванню попиту на авіаперевезення, що є важливим у сучасних реаліях української авіації.

2.2 Розробка системи обслуговування пасажирів

Система обслуговування пасажирів є основним інструментом для підвищення якості сервісу в аеропортах. У рамках розробки для міжнародного аеропорту Кривий Ріг, головною метою є створення зручного, функціонального вебзастосунку, що автоматизує ключові операції, такі як бронювання квитків, управління рейсами, та інтеграцію туристичних пропозицій. Це дозволить зменшити ручну працю, оптимізувати взаємодію між адміністрацією аеропорту та пасажирами, а також підвищити конкурентоспроможність.

Інформаційна система, що розробляється, являє собою вебдодаток, що надає авіаційні послуги користувачам. Система орієнтована на вивчення потреб користувачів у сфері авіаперельотів і туристичних пропозицій. Вона включає дві основні ролі: користувач та адміністратор. Користувачі можуть переглядати доступні авіарейси, бронювати квитки, вивчати актуальні тури та спеціальні пропозиції, а також знайомитись з матеріалами блогу сайту. Адміністратори системи мають розширені права і можуть керувати списком рейсів, створювати та редагувати туристичні пропозиції, а також керувати іншими адміністраторами.

Система повинна забезпечувати швидку обробку запитів та можливість одночасної роботи великої кількості користувачів. Також необхідно передбачати надійні механізми захисту даних. Інформаційна модель включає в себе опис головних сутностей, таких як користувачі (адміністратор та клієнт)[18].

Система має забезпечувати такі основні функції:

Функції користувача.

1. Реєстрація та аутентифікація: користувач має можливість зареєструватися та
увійти до системи, щоб отримувати доступ до персоналізованих функцій.

2. Перегляд доступних рейсів: система надає користувачам інформацію про доступні рейси, включаючи пункти відправлення та прибуття, дату вильоту, вартість квитка.

3. Бронювання квитків: користувач може вибрати відповідний рейс та забронювати квиток, вказавши свої особисті дані.

4. Перегляд турів та спеціальних пропозицій: система відображає список доступних турів та спеціальних пропозицій, які користувач може вивчити та вибрати.

5. Перегляд блогу: користувачі можуть читати статті та новини, опубліковані у блозі сайту.

Функції адміністратора.

1. Керування рейсами: адміністратор може переглядати заброньовані рейси.

2. Управління турами та спеціальними пропозиціями: адміністратор має можливість створювати нові тури та спеціальні пропозиції, а також редагувати та видаляти існуючі.

3. Керування адміністраторами: адміністратор може додавати та видаляти інших адміністраторів.

Для кращого розуміння як система використовується, необхідно розробити діаграму використання (Use Case Diagram). Діаграми створені у Visual Paradigm [19]. Така діаграма демонструє, як користувачі взаємодіють з системою. Діаграма варіантів використання представлена на рис. 2.1.

Визначені актори:

1. Адміністратор – користувач, який може керувати системою, переглядати заброньовані рейси, створювати тури, спеціальні пропозиції, та керувати іншими адміністраторами.

2. Клієнт – може бронювати квитки на рейси, тури та спеціальні пропозиції, зареєструватися в системі.

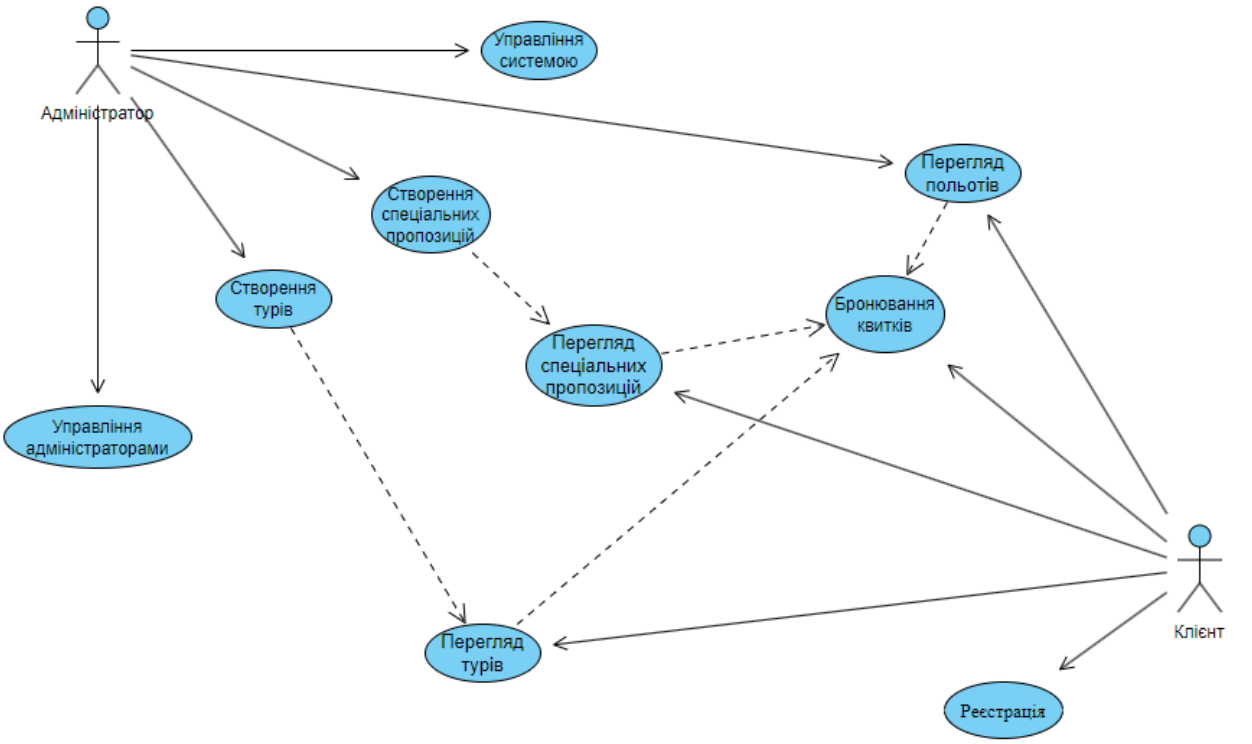

Рисунок 2.1 – Діаграма варіантів використання вебдодатку для визначених акторів

Варіанти використання вебдодатку для адміністратора:

— створення турів;
— створення спеціальних пропозицій;
— управління адміністраторами;
— перегляд польотів;
— управління системою.

Варіанти використання вебдодатку для клієнта:

— бронювання квитків;
— перегляд турів;
— перегляд спеціальних пропозицій;
— реєстрація;
— перегляд польотів.

Діаграма послідовності демонструє, як об'єкти інформаційної системи взаємодіють у певному порядку. Вона дозволяє моделювати різні процеси, зокрема взаємодію користувача з системою. На діаграмі представлені актори та об'єкти, які пов'язані між собою через повідомлення. Кожне повідомлення демонструє, як відбувається обмін інформацією між ними.

Діаграма послідовності представлена на рис. 2.2.

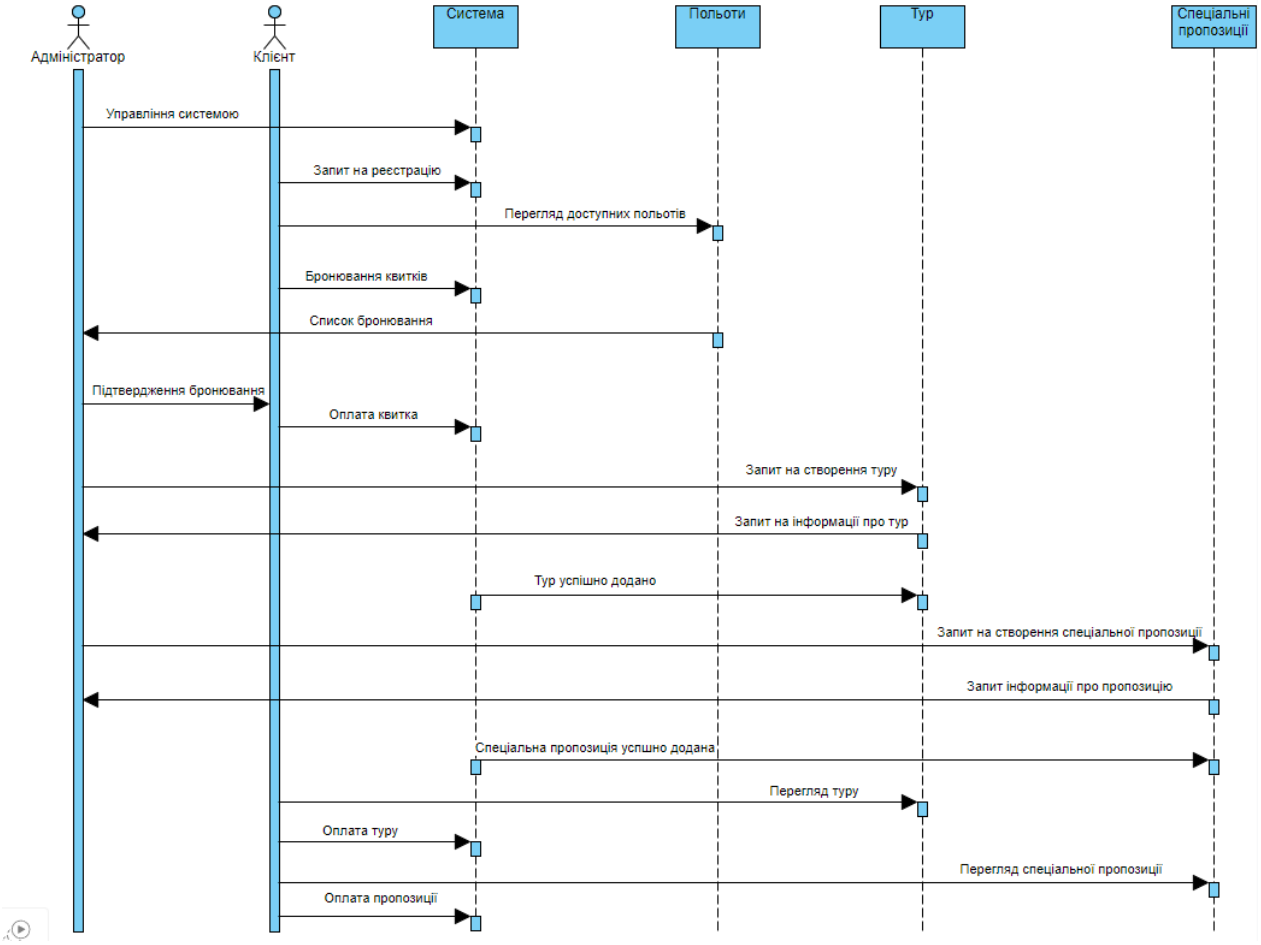

Рисунок 2.2 – Діаграма послідовності

Динаміка роботи інформаційної системи може змінюватися під впливом різних чинників, таких як впровадження нових функцій або зміни у вимогах до технологій. Тому важливо регулярно аналізувати ці зміни, щоб вчасно вносити необхідні корективи. Слід постійно моніторити продуктивність системи, відслідковувати використання ресурсів, аналізувати дії користувачів і виявляти можливі помилки. Тестування допоможе перевірити стабільність роботи

системи за умов високого навантаження. Якщо з'являться нові функції, їх необхідно детально дослідити. На основі отриманих даних можна впроваджувати заходи для покращення продуктивності та вдосконалення системи.

На діаграмі представлені лінії акторів: адміністратор та користувач. А також компоненти: система, польоти, тур і спеціальні пропозиції. Діаграма послідовності показує що є адміністратор який керує системою, підтверджує бронювання квитків користувачів, створює нові тури та спеціальні пропозиції. Користувач може переглядати доступні йому рейси, але для оформлення бронювання йому необхідно зареєструватися, також є можливість оформлення турів та спеціальних пропозицій. Клієнт бронює квитки, і адміністратор отримує список броньованих рейсів, де відображається уся інформація, зв’язується з клієнтом для підтвердження бронювання, і вже після підтвердження бронювання від адміністратора, клієнт може заплатити за квитки.

Для опису поведінки компонентів системи використовується діаграма станів. Діаграма станів зображена на рис. 2.3.

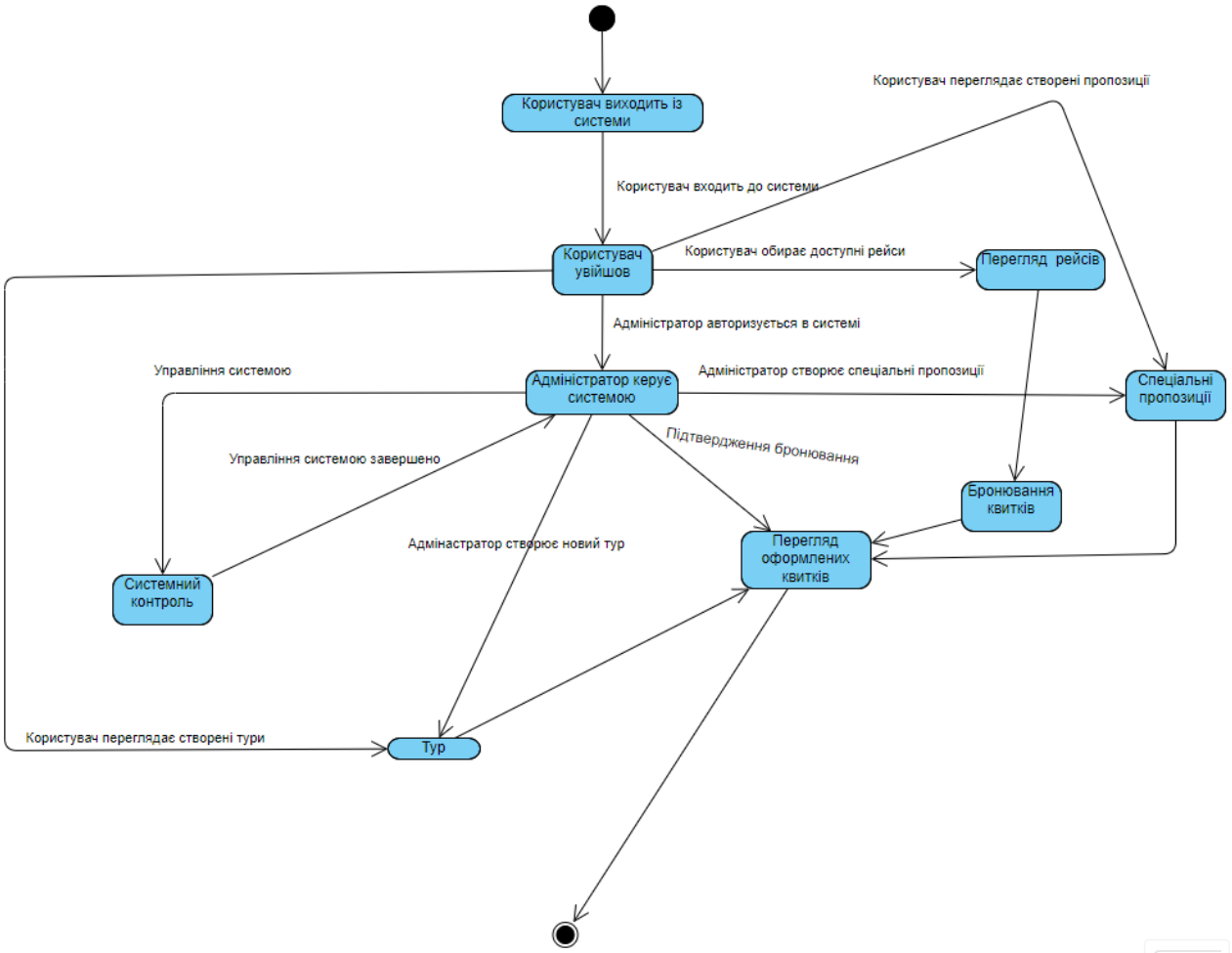

Рисунок 2.3 – Діаграма станів

Діаграма описує основні дії та переходи між станами в системі. Вона починається з «Користувач виходить із системи», який представляє відсутність входу в систему, якщо користувач увійшов до системи, то перехід відбувається до «Користувач увійшов». З цього стану можливі декілька виходів. Вхід Адміністратора де він керує системо, стан «Адміністратор керує системою». Зі стану «Користувач увійшов» є декілька переходів «Перегляд рейсів», де користувач переглядає доступні пропозиції, які присутні. Та до стану «Бронювання квитків», коли користувач обрав задовільний для нього рейс. Зі стану «Адміністратор керує системою» є також декілька переходів до інших станів, «Перегляд оформлених квитків» де адміністратор підтверджує бронювання квитків користувача. Стан «Тур», де адміністратор створює нові тури, до цього стану також іде перехід від користувача, якщо він обирає тури.

Від адміністратора іде також стан до «Спеціальні пропозиції», де адміністратор створює спеціальні пропозиції, також до цього стану іде зв’язок від користувача коли він обирає спеціальні пропозиції. Все завершується підтвердженням бронювання квитків від адміністратора.

Діаграма розгортання представлена на рис. 2.4

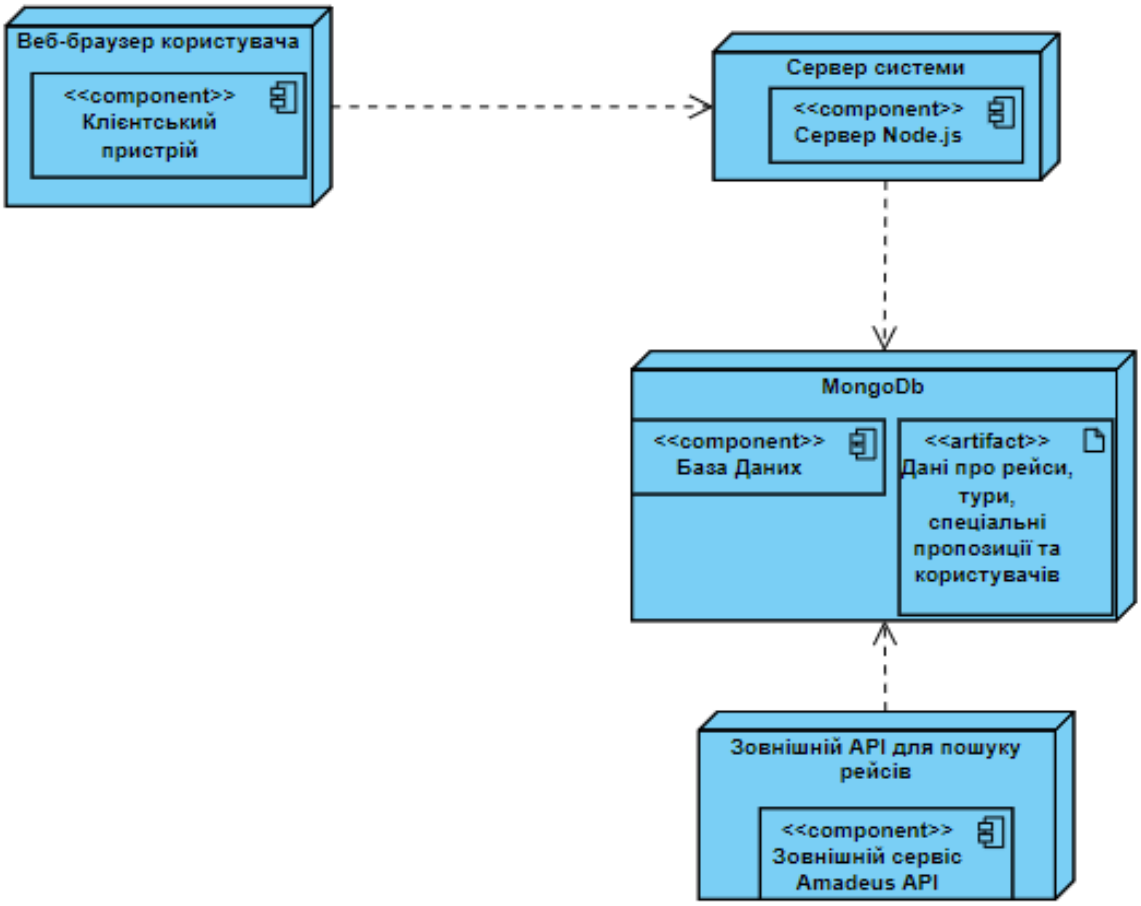

Рисунок 2.4 – Діаграма розгортання

Клієнтський пристрій це інтерфейс користувача, з яким взаємодіють клієнти та адміністратори через браузер. Тут починається вся взаємодія із системою. Потім сервер, на якому працює додаток, реалізований на Node.js. Тут обробляються запити користувачів та взаємодія із зовнішніми сервісами та базою даних. Система взаємодіє із зовнішнім API Amadeus для отримання даних про рейси. У системі працює база даних MongoDB, яка зберігає інформацію про користувачів, замовлення, тури та спеціальні пропозиції.

Система взаємодіє з базою даних для виконання операцій (створення, читання, оновлення, видалення даних).

У діаграмі класів представлено розробку моделі програмної системи, починаючи з логічної моделі, для чого застосовується діаграма класів. (рис. 2.5).

Взаємозв'язки між класами на діаграмі відображають їхню взаємодію. На діаграмі виділено 5 класів, які описують різні об'єкти в системі:

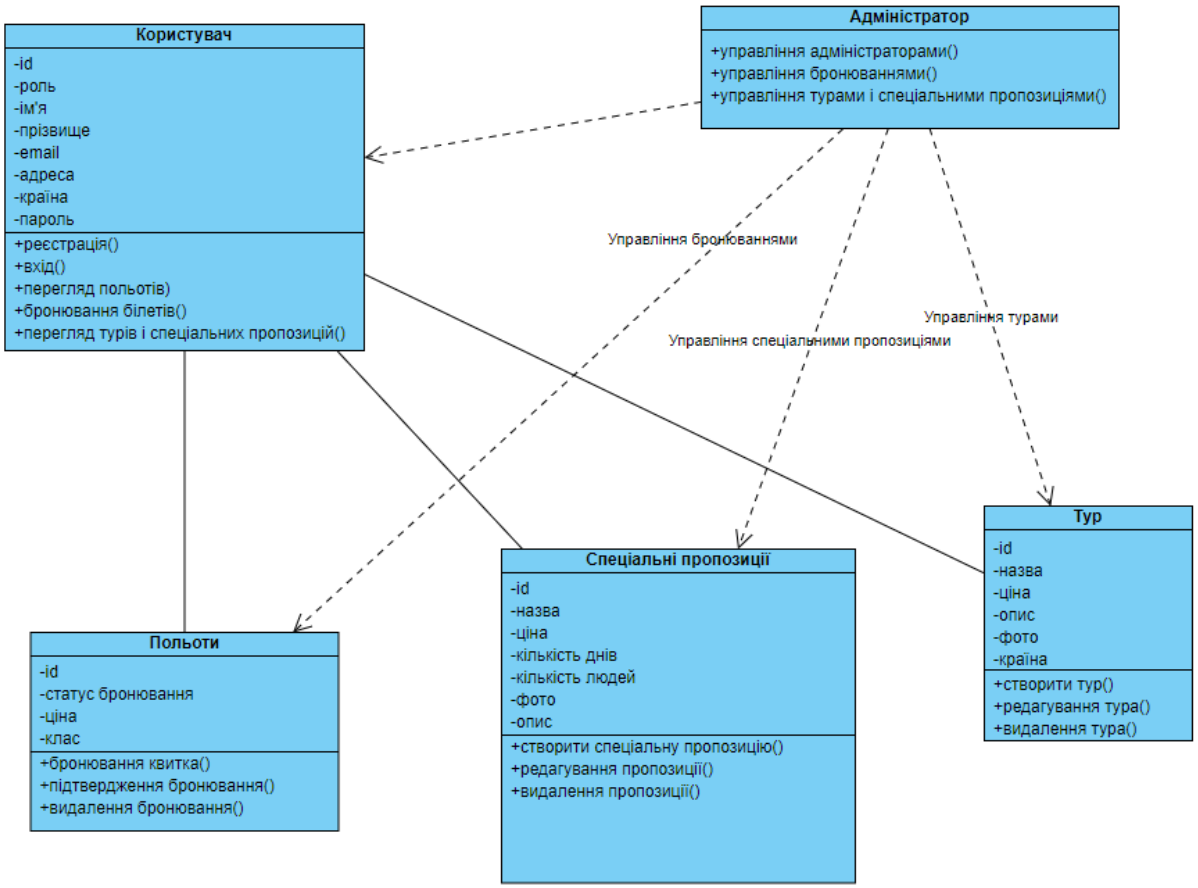

Рисунок 2.5 – Діаграма класів

1. Клас «Користувач» – який представляє звичайного користувача системи. Користувачі можуть реєструватися, входити до системи та взаємодіяти з різними функціями, такими як бронювання квитків та перегляд інформації про тури.

2. Клас «Адміністратор» – який наслідується від класу «Користувач», та представляє адміністратора системи, який має додаткові права, порівняно зі

звичайним користувачем. Адміністратори керують вмістом та функціональністю системи.

3. Клас «Польоти» – що представляє рейс, який включає інформацію про маршрут, час вильоту та вартість квитка.

4. Клас «Тур» – що представляє туристичний пакет, що пропонується користувачам. Включає інформацію про назву, опис та вартість туру.

5. Клас « Спеціальні пропозиції» – який представляє спеціальні пропозиції для користувачів.

2.3 Інформаційне забезпечення для оптимізації процесів

Інформаційне забезпечення відіграє ключову роль у розробці програмного забезпечення. У цьому розділі наводяться структури та схеми інформаційних об'єктів і ресурсів, а також схеми інформаційних потоків і бази даних.

Структури та схеми інформаційних об'єктів і ресурсів. У рамках розроблюваної системи можна визначити конкретні інформаційні об'єкти та ресурси. База даних складається з трьох програмних кодів. Структура представлена на рис. 2.6.

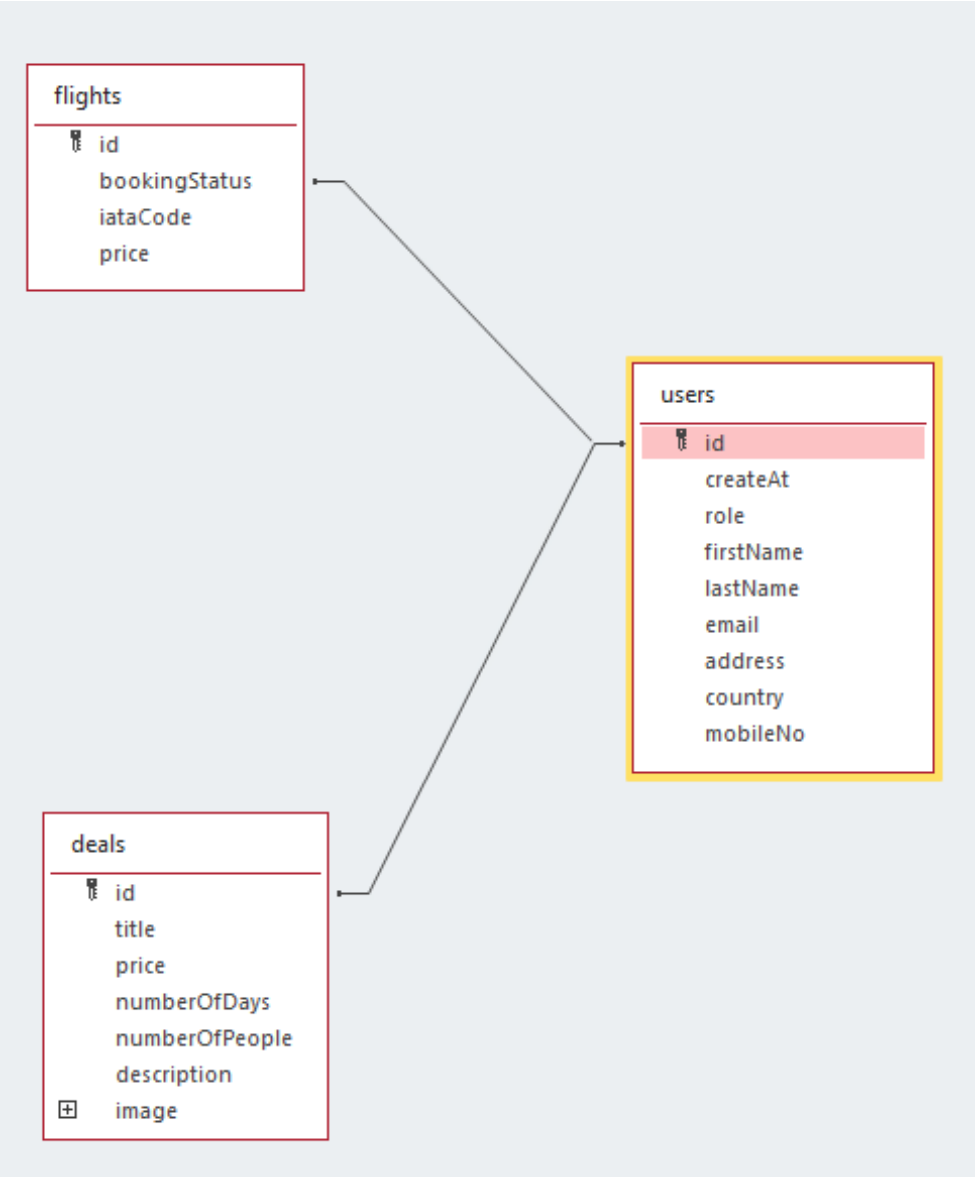

Рисунок 2.6 – Структура бази даних

Документ users складається з користувачів системи таких як Адміністратори та Клієнти. Документ users, представлений на рисунку 2.7, містить дані користувачів системи, які виконують різні ролі, зокрема адміністраторів і клієнтів. Він має структуру бази даних у форматі JSON. Кожен об'єкт (запис) представляє окремого користувача з детальною інформацією про нього.

Рисунок 2.7 – Документ users

Цей документ містить поля:

— id – ідентифікатор користувача;

— createdAt – дата створення облікового запису;

— role – роль користувача, адміністратор чи клієнт;

— firstName – ім’я користувача;

— lastName – прізвище користувача;

— email – електронна пошта;

— address – адреса користувача;

— country – країна користувача;

— mobileNo – номер телефону.

Документ flights містить інформацію про оформленні білети клієнтів з підтвердженим бронюванням чи ні. Документ flights, представлений на рисунку 2.8, є основним елементом зберігання інформації про авіарейси, доступні для

клієнтів. Він містить детальні дані про квитки, статус бронювання та пов'язані з ними параметри. Основною функцією документа є забезпечення запису інформації, необхідної для відстеження рейсів, обробки бронювань та подальшого оформлення квитків.

Рисунок 2.8 – Документ flights

Цей документ містить поля:

— bookingStatus – статус бронювання;

— id – ідентифікатор білету;

— iataCode – код авіакомпанії відправника та прибуття;

— price – ціна за білет;

Документ deals, представлений на рис. 2.9, містить інформації про тури та спеціальні пропозиції. Він є важливим для реалізації комерційних та маркетингових функцій системи. Він сприяє залученню клієнтів через надання актуальних спеціальних пропозицій, підвищуючи привабливість продукту. Завдяки добре структурованій базі даних, адміністратори можуть легко додавати нові тури, оновлювати інформацію та відслідковувати актуальність пропозицій.

Рисунок 2.9 – Документ deals

Цей документ містить поля:

— id – ідентифікатор угоди;

— title – назва угоди;

— price – ціна угоди;

— numberOfDays – кількість днів для спеціальних пропозицій ;

— numberOfPeople – кількість людей ;

— description – опис ;

— image – ціна.

База даних для системи авіаційних вебпослуг складається з кількох ключових документів, що забезпечують ефективне зберігання та управління даними користувачів, авіаквитків, турів та спеціальних пропозицій. Їх структура забезпечує оптимальну взаємодію між адміністраторами та користувачами системи, дозволяючи керувати бронюваннями, турами та користувачами.

Документ flights відіграє ключову роль в управлінні процесом бронювання авіаквитків. Він містить інформацію про статус бронювання кожного квитка, ціну, а також коди аеропортів або авіакомпаній для відправлення і прибуття. Це дозволяє системі відслідковувати всі етапи замовлення квитка, від його створення до підтвердження бронювання. Інформація з цього документу використовується для того, щоб користувачі могли бачити свої активні або минулі бронювання, а також дізнаватися про вартість і деталі кожного рейсу. Статус бронювання є ключовим параметром, що вказує на етап бронювання (наприклад, підтверджено чи очікується оплата), що дозволяє системі автоматично оновлювати дані та надавати актуальну інформацію користувачам. Така система важлива для надання точних даних, особливо в контексті авіаперевезень, де зміни можуть відбуватися досить часто.

Документ deals відповідає за зберігання даних про доступні тури та спеціальні пропозиції. Він містить ключову інформацію, необхідну для управління цими угодами, включаючи назву туру, його вартість, кількість днів та людей, на яких він розрахований. Така інформація дозволяє адміністраторам створювати і публікувати нові тури та спеціальні пропозиції, доступні для користувачів, а також забезпечує користувачам можливість отримати повну інформацію про кожну пропозицію, порівнювати їх і робити вибір. Крім того, система дозволяє додавати опис і зображення до кожної угоди, що робить її

більш привабливою для потенційних клієнтів. Опис допомагає детальніше пояснити умови і переваги туру або пропозиції, а зображення надає візуальну складову, що сприяє залученню користувачів.

Документ users містить дані про користувачів системи, включаючи клієнтів і адміністраторів. Цей документ є важливою складовою бази даних, оскільки вона не лише забезпечує зберігання особистої інформації (ім’я, прізвище, контактні дані), але й визначає роль користувача в системі. Це дозволяє системі визначати, які права доступу має кожен користувач — клієнти можуть переглядати рейси, замовляти квитки та спеціальні пропозиції, в той час як адміністратори мають змогу керувати контентом системи (додавати нові тури, створювати спеціальні пропозиції). Крім того, поля для адреси та контактної інформації забезпечують можливість швидкого зв’язку з користувачами, що може бути важливим для відправлення оновлень про рейси, підтверджень бронювань або надання іншої важливої інформації. Рольова система гарантує безпеку, оскільки тільки адміністратори можуть мати доступ до функцій управління.

База даних побудована таким чином, щоб забезпечувати тісну взаємодію між документами. Наприклад, документ flights пов’язаний з документом users, щоб кожен користувач міг бачити свої активні та минулі бронювання. Це створює чітку структуру взаємодії між даними різних типів. Так само документ deals надає можливість користувачам бачити актуальні спеціальні пропозиції та тури.

Адміністратори системи мають доступ до документів flights і deals, щоб керувати створенням і публікацією нових турів та пропозицій, а також відслідковувати статуси бронювань, але не мають можливості безпосередньо створювати або керувати рейсами (що обмежено логікою системи). Для створення турів і спеціальних пропозицій необхідно заповнити певну інформацію про ці компоненти. Коли адміністратор їх створив у нього є також можливість і видалити їх. Клієнт в свою чергу після створення турів і пропозицій може забронювати їх і одразу заплатити за них так як уже вся

необхідна інформація в них міститься на відміну від бронювання звичайних рейсів де адміністратор ще має підтвердити бронювання.

Ця структура бази даних є гнучкою та ефективною для підтримки основних функцій системи авіаційних вебпослуг. Вона забезпечує управління квитками, турами та користувачами, дозволяючи автоматизувати багато процесів і зберігати важливу інформацію в зручному для доступу вигляді

В рамках розробленої системи для зберігання та керування даними використовується база даних MongoDB. Вона є нереляційною документно-орієнтованою СУБД, яка дозволяє гнучко організовувати дані у вигляді колекцій та документів. Це забезпечує високу продуктивність під час роботи з великими обсягами даних і дозволяє масштабувати систему зі збільшенням навантаження. У системі MongoDB зберігається інформація про користувачів, тури, спеціальні пропозиції та рейси, що дозволяє ефективно обробляти запити та оперативно оновлювати дані в реальному часі.

2.4 Методи синтезу систем керування для оптимізації процесів

Синтез систем керування спрямований на створення моделей, алгоритмів і структур для забезпечення ефективного функціонування різних процесів. В авіаційній галузі, зокрема для оптимізації обслуговування пасажирів, синтез систем керування базується на математичному моделюванні, обчислювальних методах та теорії управління.

Алгоритм дій для користувача Адміністратора виглядає наступним чином: спочатку він проходить авторизацію в системі, після чого система показує панель з доступними опціями, з яких користувач обирає одну (календар, польоти, світові тури, спеціальні пропозиції, керування адміністраторами та виходу із системи).

Алгоритмічні дії для користувача Клієнта: він потрапляє на головну сторінку де може переглядати необхідні йому рейси, світові тури, спеціальні пропозиції, але для бронювання йому необхідно зареєструватися або увійти

якщо в нього вже створений акаунт. Алгоритмічна схема користувача «Користувач» представлена на рис. 2.10.

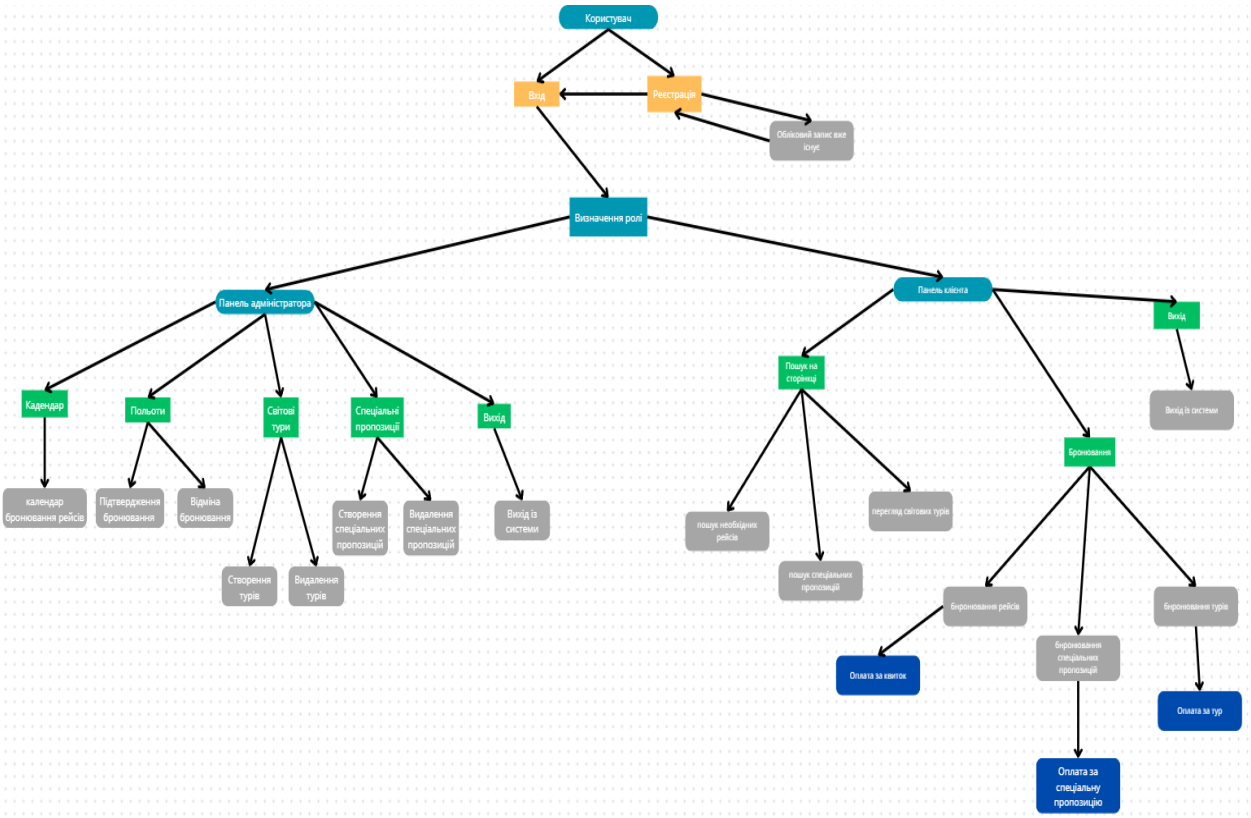

Рисунок 2.10 – Схема «Користувач»

При виборі рейсу система повинна перевірити, чи доступні квитки для обраного рейсу. Дані про наявність місць оновлюються за допомогою зовнішнього API, Amadeus. У адміністраторів є можливість додавати нові тури до системи. Система валідує дані, такі як тривалість туру, кількість осіб та ціна, перевіряє коректність введених даних, після чого зберігає новий запис до бази даних. Коли новий користувач (клієнт) реєструється в системі, алгоритм перевіряє унікальність введених даних (електронна пошта, номер телефону), шифрує пароль, та зберігає інформацію про користувача базу даних, якщо такий користувач вже існує в базі даних то система не дасть створити обліковий запис. Після реєстрації або під час входу в систему користувач вводить логін та пароль. Система порівнює введені дані із інформацією в базі даних. При успішній авторизації створюється сесія для користувача, яка зберігає

інформацію про його роль (клієнт або адміністратор), що визначає доступ до функцій системи.

У системі реалізовано можливість управління правами доступу користувачів на основі їх ролей. Система перевіряє роль користувача (клієнт або адміністратор) та надає доступ лише до тих дій, які дозволені для цієї ролі. Клієнт переглядає необхідні йому рейси або створені адміністраторам тури і спеціальні пропозиції. Коли клієнт бронює квитки на рейс, бронювання переходить в стан очікування, поки адміністратор не зв’яжеться з клієнтом і не підтвердить бронювання, тоді клієнт має змогу заплатити за квиток. Під час вибору туру або спеціально пропозиції, оплата доступна одразу.

Коли адміністратор авторизується в системі він може переглянути календар польотів де розміщенні заброньовані рейси, точна дата вильоту та прильоту. Є можливість переглядати заброньовані рейси на їх різних стадіях: в очікуванні (коли клієнт чекає на підтвердження бронювання зі сторони адміністратора), затверджено (коли адміністратор підтвердив бронювання клієнта), підтверджено (коли клієнт заплатив за квиток) і скасовано (рейси які були скасовані від бронювання). Адміністратор також має право створювати тури та спеціальні пропозиції, заповнивши їх певною інформацією (назва, ціна, країна, опис, зображення і т.д.), після їх створення у клієнтів є можливість їх забронювати. Також адміністратор може створювати облікові записи для інших адміністраторів в разі потреби. Система надсилає запит на сервер Amadeus API для отримання списку доступних рейсів. Запит містить параметри, такі як номер авіакомпанії, дата вильоту, ціна та дата прильоту, після чого система обробляє отриману відповідь та виводить дані користувачеві.

Найважливіші аспекти системи включають управління бронями, обробку цінових пропозицій, а також роботу з даними користувача. Основним завданням системи є управління процесом бронювання квитків та спеціальними пропозиціями. Система використовує пошук, фільтрація даних та моделювання стану бронювання, для зручного управління процесами та для перегляду усіх заброньованих рейсів на різних їх етапах. Кожне бронювання може знаходитися

в одному з кількох станів: очікується підтвердження, затверджено, підтверджено, скасовано. Переходи між цими станами можуть описуватися з допомогою кінцевих результатів.

В залежності від дій користувачів система відображає бронювання в цих станах. Наприклад коли клієнт бронює рейс який йому необхідний бронювання переходить в стан очікування до поки адміністратор не підтвердить бронювання, і тоді вже воно знаходиться в статусі затверджено і клієнт на цьому етапі може перейти до оплати за квиток, після чого воно перейде в стан підтверджено. Система здійснює пошук та фільтрацію інформації про рейси, використовуючи API Amadeus, що дозволяє забезпечити актуальність даних про доступні рейси та ціни.

Для пошуку рейсів використовується пошук у базі даних, що дозволяє швидко знаходити необхідні пропозиції щодо запитів користувачів. Запити API Amadeus формуються на основі введених користувачем параметрів, таких як пункт відправлення, пункт призначення та дата вильоту. Система отримує інформацію про рейси в реальному часі через API Amadeus.

Бронювання є ключовим процесом у системі, який дозволяє користувачеві знайти та забронювати авіаквиток. Користувач вводить запит на пошук рейсів, вказуючи точки відправлення та прибуття, дату та інші параметри. Система надсилає запит до API Amadeus, який повертає список доступних рейсів за заданими параметрами. Користувач вибирає потрібний рейс і система відправляє підтвердження бронювання, зберігаючи дані в базі MongoDB. І в залежності подальших дій перевіряється статус бронювання, і користувачу відображає статус успішного або невдалого бронювання (успіх або помилка). Створення, зміни та видалення турів та спеціальних пропозицій здійснюється адміністраторами системи. Адміністратор заповнює форму з інформацією про новий тур або спеціальну пропозицію (назва, опис, ціна, кількість людей, днів, країна та опис). Система зберігає новий запис у таблиці "deals" бази даних MongoDB. Адміністратор може також оновлювати та видаляти тури або спеціальні пропозиції, вносячи зміни до бази даних через

інтерфейс системи. Система підтримує два типи користувачів: клієнти та адміністратори. Під час реєстрації дані (ім'я, прізвище, email, пароль і т.д.) зберігаються у базі даних. Вхід до системи з перевіркою облікових даних (електронна пошта та пароль) перевіряються чи існує такий користувач. У разі успішної перевірки створюється сесія для користувача. Розмежування доступу, клієнти мають доступ до бронювання рейсів та перегляду пропозицій, тоді як адміністратори керують турами та користувачами. Керування обліковими записами адміністраторів включає додавання та видалення адміністраторів поточними адміністраторами. Приклад створення нового туру представлено на рис. 2.11.

```
const details = JSON.parse(req.body.details);
const deal = await Deals.findOne({ "details.country": details.country });
if (deal) {
  const updatedDeal = await Deals.findOneAndUpdate(
    { "details.country": details.country },
    {
      type: "WorldTour",
      "details.country": details.country,
      $push: {
        "details.packages": {
          _id: mongoose.Types.ObjectId(),
          title: details.packageTitle,
          price: details.packagePrice,
          description: details.packageDescription,
          image: req.file.filename,
          bookedBy: []
        }
      }
    }
  );
  await res.json({
    success: true,
    message: `Тур успішно створено`
  });
} else {
  const newDeal = await Deals.create({
    type: "WorldTour",
    "details.country": details.country,

    "details.packages": {
      _id: mongoose.Types.ObjectId(),
      title: details.packageTitle,
      price: details.packagePrice,
      description: details.packageDescription,
      image: req.file.filename,
      bookedBy: []
    }
  });
  await res.json({
    success: true,
    message: `Тур успішно створено`
  });
}
} catch (error) {
  console.log("error", error.message);
  await res.json({ message: error.message });
```

Рисунок 2.11 – Фрагмент програмного коду створення туру

На рисунку зображений код, є функцією createWorldTour, з використанням бібліотеки Mongoose для взаємодії з MongoDB. Основне завдання цієї функції — створення або оновлення світового тура для визначеної країни в колекції Deals.

Функція отримує дані про тур через запит (req.body.details). Вона перевіряє, чи існує тур в базі даних для вказаної країни, використовуючи метод Deals.findOne. Якщо тур для цієї країни вже існує (якщо знайдено поле "details.country": details.country), то відбувається оновлення запису. У протилежному випадку створюється новий запис. Функція оновлює існуючий тур за допомогою методу findOneAndUpdate(). Нова інформація про пакет (назва, ціна, опис, зображення) додається до масиву пакетів (details.packages) через $push. Це означає, що новий пакет буде додано до вже існуючого в цьому турі. Після успішного оновлення або створення тура функція повертає JSON-відповідь із повідомленням про те, що тур був успішно створений.

Висновки до розділу

У цьому розділі було обґрунтувано розробку вебдодатку для обслуговування пасажирів міжнародного аеропорту Кривий Ріг, використовуючи математичні моделі для оптимізації функціональних процесів. Було розроблено діаграму використання (Use Case Diagram), яка демонструє, як користувачі взаємодіють з системою. Також було здійснено проєктування системи, яка забезпечуватиме швидку обробку запитів та матиме можливість одночасної роботи великої кількості користувачів.

Інформаційне забезпечення відіграє ключову роль у розробці програмного забезпечення. У цьому розділі наводяться структури та схеми інформаційних об'єктів і ресурсів, а також схеми інформаційних потоків і бази даних. В рамках інформаційного забезпечення системи були описані функціональні вимоги, можливості кожного користувача та визначені їх ролі.

Також було представлено дії для кожного користувача. Опис бази даних включає її структуру, що складається з трьох документів та зв’язків між ними.

В рамках розробленої системи для зберігання та керування даними використовується база даних MongoDB. Вона є нереляційною документно-орієнтованою СУБД, яка дозволяє гнучко організовувати дані у вигляді колекцій та документів. Це забезпечує високу продуктивність під час роботи з великими обсягами даних і дозволяє масштабувати систему зі збільшенням навантаження. У системі MongoDB зберігається інформація про користувачів, тури, спеціальні пропозиції та рейси, що дозволяє ефективно обробляти запити та оперативно оновлювати дані в реальному часі.

РОЗДІЛ 3 ПРОГРАМНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ПРОЦЕСОМ

3.1 Розробка структури програмної системи для автоматизації

Метою розробки є створення програмної системи для автоматизації процесів бронювання авіаквитків та управління авіаційними послугами в рамках діяльності міжнародного аеропорту "Кривий Ріг". Система повинна забезпечувати ефективну взаємодію між користувачами (пасажирами) та адміністраторами аеропорту, спрощуючи бронювання рейсів, управління даними про рейси та підвищення ефективності внутрішніх процесів.

Програмна система включає наступні основні модулі:

1. Модуль користувача:

— функції реєстрації та аутентифікації;

— перегляд доступних рейсів;

— бронювання квитків;

— перегляд турів, спеціальних пропозицій та блогу.

2. Модуль адміністратора:

— управління рейсами;

— створення та редагування туристичних пропозицій;

— управління іншими адміністраторами.

База даних:

Структура бази даних включає три основні таблиці:

— таблиця користувачів, що зберігає дані про клієнтів і адміністраторів;

— таблиця рейсів, що містить інформацію про авіарейси.

— таблиця бронювань, що реєструє дані про здійснені бронювання.

Інтерфейс взаємодії:

— вебінтерфейс для користувачів та адміністраторів;

— реалізація зручної навігації для доступу до функцій системи.

У системі використовується програмне забезпечення, яке забезпечує необхідну функціональність. Системне програмне забезпечення: як основа, застосовується операційна система Windows, яка відповідає за стабільне управління ресурсами та надає базові функції для роботи[20].

Вимоги до програмного забезпечення:

— операційна система: Windows;

— не менше 8 гігабайт оперативної пам’яті, рекомендовано 16 гігабайт;

— процесор: i3 7100 і вище;

— рекомендовано використовувати SSD накопичувач для операційної системи і програмного забезпечення для більш швидкої роботи;

— підключення до інтернету.

З огляду на специфіку роботи аеропорту "Кривий Ріг", система дозволяє вирішити такі ключові завдання:

— залучення нових пасажирів через інтеграцію спеціальних пропозицій;

— збільшення ефективності роботи адміністрації завдяки автоматизації обробки заявок на бронювання;

— забезпечення прозорості процесів управління рейсами та розширення спектру послуг аеропорту;

Розроблена структура програмної системи спрямована на автоматизацію роботи міжнародного аеропорту "Кривий Ріг". Її впровадження сприятиме покращенню якості обслуговування, збільшенню пасажиропотоку та оптимізації управлінських процесів.

3.2 Вибір технологій та інструментів для розробки

Розробка вебзастосунку для автоматизації обслуговування пасажирів аеропорту вимагає використання сучасних технологій, які забезпечують гнучкість, продуктивність і зручність у використанні. Вибрані технології охоплюють усі етапи розробки: від створення інтерфейсу користувача до роботи з базами даних і інтеграції з зовнішніми сервісами.

Для розробки цього проєкту були використані такі технології:

1. HTML (HyperText Markup Language) – мова розмітки, яка відповідає за структурування та організацію контенту вебсторінки. За допомогою HTML створюються заголовки, списки, таблиці, форми, кнопки та інші елементи, які дозволяють користувачам взаємодіяти із системою.

2. CSS (Cascading Style Sheets) – технологія для стилізації зовнішнього вигляду елементів на вебсторінці. CSS дозволяє налаштовувати кольори, шрифти, розміри, відступи та інші аспекти дизайну, забезпечуючи гармонійний та привабливий вигляд інтерфейсу.

3. JavaScript – скриптова мова програмування, яка використовується для реалізації логіки на стороні клієнта. Вона додає інтерактивність до вебсторінок, дозволяє обробляти дії користувача, виконувати асинхронні запити до сервера та проводити валідацію даних без необхідності перезавантаження сторінки.

4. React – JavaScript-бібліотека, яка використовується для створення інтерфейсів користувача. Вона дозволяє розробникам працювати зі складними структурами даних завдяки компонентному підходу, що сприяє модульності та повторному використанню коду. React також забезпечує високу продуктивність завдяки віртуальному DOM, який мінімізує кількість змін у реальному DOM під час оновлення інтерфейсу.

Інтерфейс користувача розроблений із використанням HTML, CSS, JavaScript та React що забезпечує зручну взаємодію користувачів із системою. Створено сторінки для авторизації та різних ролей користувачів, які включають елементи управління, такі як кнопки, форми та таблиці. React використовується для реалізації динамічної взаємодії з інтерфейсом, обробки подій та виконання асинхронних запитів до сервера.

Основні функції включають:

— сторінки для реєстрації та авторизації користувачів;

— динамічний пошук рейсів із фільтрацією за датами, напрямками та цінами;

— модуль бронювання квитків із підтвердженням через email;

— панель адміністратора для управління рейсами, пропозиціями та аналітичними даними;

Технології для серверної частини:

1. Node.js - використовується для реалізації серверної логіки. Завдяки подієво-орієнтованій архітектурі, Node.js забезпечує високу продуктивність під час обробки численних запитів. Серверна частина керує авторизацією користувачів, обробкою запитів на бронювання рейсів і взаємодією з базою даних.

2. Express.js - це мінімалістичний фреймворк для Node.js, який спрощує створення веб-сервісів і API. Express.js дозволяє організовувати маршрути для запитів користувачів, обробляти дані форм і передавати відповіді на запити.

3. MongoDB- це нереляційна база даних, що забезпечує гнучке зберігання даних у форматі JSON. MongoDB ідеально підходить для динамічних систем, які потребують масштабованості та швидкої обробки великих обсягів інформації. У проєкті MongoDB використовується для зберігання інформації про користувачів, рейси, бронювання та спеціальні пропозиції.

На серверній стороні логіка реалізована за допомогою Node.js. Всі запити, які надсилаються користувачами, обробляються контролерами, організованими у вигляді функцій і класів. Додавання, редагування або видалення даних виконується через MongoDB, яка забезпечує ефективне зберігання та маніпулювання інформацією.

Інтеграція зі сторонніми сервісами:

— Amadeus API - ця платформа використовується для отримання інформації про рейси, ціни та маршрути. Інтеграція з Amadeus API дозволяє в режимі реального часу оновлювати дані про доступні авіарейси, забезпечуючи актуальність інформації для користувачів.

— Системи онлайн-платежів - інтеграція з платіжними системами (наприклад,LiqPay або Stripe) забезпечує безпечні транзакції при оплаті квитків.

В залежності від ролі користувача, система дозволяє виконувати різні операції. Наприклад, адміністратори можуть створювати нові спеціальні пропозиції або редагувати існуючі дані. Комбінація цих технологій, включаючи Node.js, React, MongoDB та Amadeus API, забезпечує функціональність, продуктивність і зручність використання системи.

Переваги вибраних технологій:

— Гнучкість і масштабованість:

Використання MongoDB і Node.js дозволяє масштабувати систему при збільшенні кількості користувачів або даних.

— Швидкодія:

React.js та віртуальний DOM мінімізують час оновлення інтерфейсу, забезпечуючи швидкий відгук системи.

— Зручність для розробників:

Компонентний підхід React.js і простота Express.js спрощують написання, тестування та підтримку коду.

— Інтеграція зі сторонніми сервісами:

Використання Amadeus API забезпечує доступ до актуальної інформації про авіаперевезення, значно скорочуючи час і витрати на реалізацію цього функціоналу.

Вибрані технології та інструменти дозволяють створити сучасний вебзастосунок, який відповідає вимогам функціональності, продуктивності та зручності. Завдяки цим технологіям система обслуговування пасажирів аеропорту Кривий Ріг є не лише ефективним рішенням для автоматизації процесів, але й платформою, яка легко адаптується до майбутніх потреб.

3.3 Розробка алгоритмів обробки даних та управління процесами

У системі створено окремі модулі, кожен із яких відповідає за виконання конкретних функцій під час роботи з програмою. Серед них: модуль для авторизації, модуль для керування бронюваннями, для управління світовими

турами, спеціальними пропозиціями, адміністраторами, для бронювання існуючих рейсів.

Для забезпечення ефективної роботи вебзастосунку реалізовано модульну структуру системи. Кожен модуль виконує певні функції, пов'язані з обробкою даних або управлінням бізнес-процесами. Це забезпечує гнучкість, легкість модифікації та інтеграцію нових функцій у майбутньому. Основні модулі системи включають:

1. Модуль авторизації, який відповідає за ідентифікацію та автентифікацію користувачів.

Алгоритм роботи:

— користувач вводить email та пароль;

— дані надсилаються на сервер, де вони перевіряються на відповідність записам

у базі даних;

— разі успішної перевірки користувач отримує токен доступу, який використовується для подальшої взаємодії з системою;

— на основі ролі користувача (пасажир, адміністратор) відбувається перенаправлення на відповідну сторінку.

На рисунку 3.1 (умовний приклад) представлено блок-схему алгоритму авторизації користувача. У ній описано основні етапи: перевірка введених даних, взаємодія з базою даних і перенаправлення залежно від ролі.

Модуль авторизації забезпечує процес входу користувачів до системи. Для цього необхідно ввести адресу електронної пошти та пароль, які перевіряються на відповідність. У разі успішного входу користувач автоматично перенаправляється на відповідну сторінку, залежно від призначеної йому ролі (рис. 3.1).

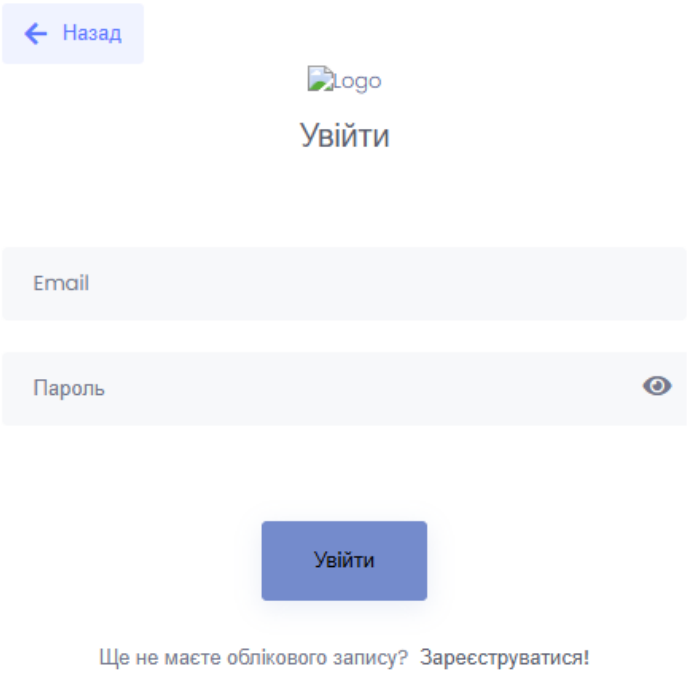

Рисунок 3.1 – Вікно авторизації

2. Модуль бронювання рейсів, що автоматизує процес бронювання квитків на
доступні рейси.

Алгоритм роботи:

— користувач обирає рейс із доступного списку;

— система перевіряє доступність місць і ціни;

— користувач вводить свої дані (ім’я, паспортні дані) та підтверджує бронювання;

— інформація про бронювання зберігається в базі даних;

— надсилається електронне підтвердження на email користувача.

Модуль управління бронюваннями дозволяє адміністратору переглядати список усіх заброньованих рейсів, а також погоджувати бронювання або відхиляти (рис. 3.2).

Рисунок 3.2 – Таблиця з бронювання рейсів

3. Модуль управління турами, який забезпечує додавання та редагування інформації про туристичні пропозиції.

Алгоритм роботи:

— адміністратор вводить дані про тур (напрямок, ціна, дати);

— система перевіряє коректність даних;

— інформація публікується для перегляду користувачами.

Усі запити до сервера обробляються асинхронно за допомогою технологій Node.js та Promise API, що дозволяє уникнути зависань програми під час великих навантажень.

Модуль бронювання використовує Amadeus API для отримання актуальної інформації про рейси, маршрути та ціни.

Алгоритм:

— система надсилає запит до API на основі введених параметрів (місто вильоту, дата);

— отримані дані обробляються й відображаються у вигляді списку для користувачів.

Модуль світових турів, призначений для перегляду та бронювання їх зі сторони клієнта, та створення і редагування зі сторони адміністратора (рис. 3.3 та 3.4).

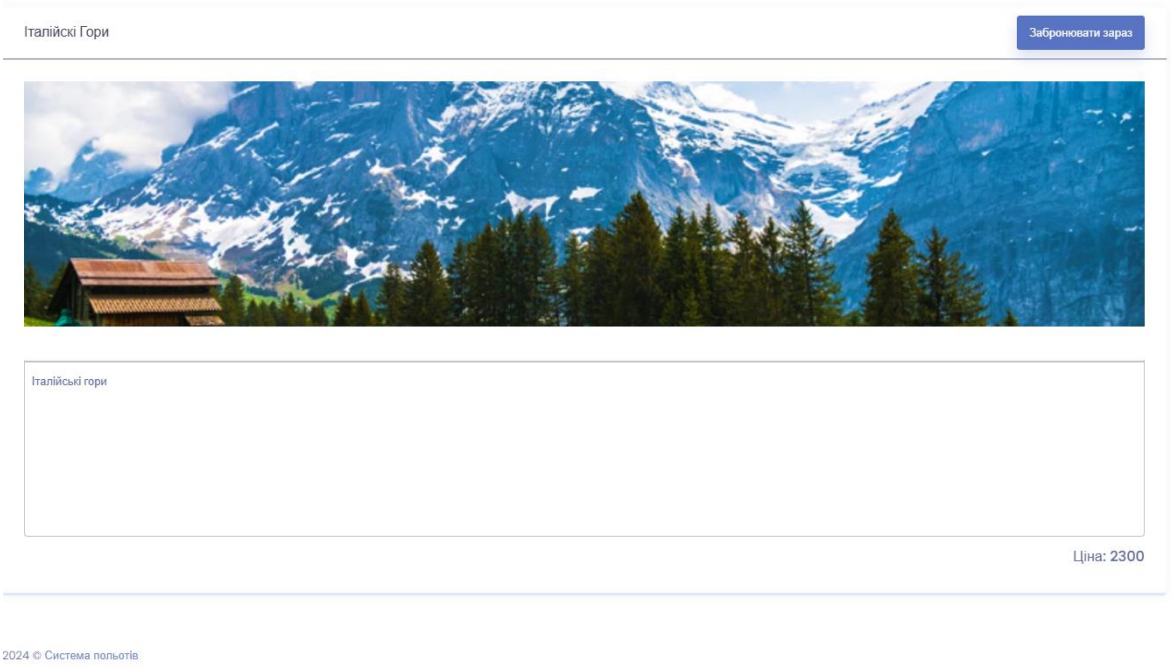

Рисунок 3.3 – Світовий тур(Клієнт)

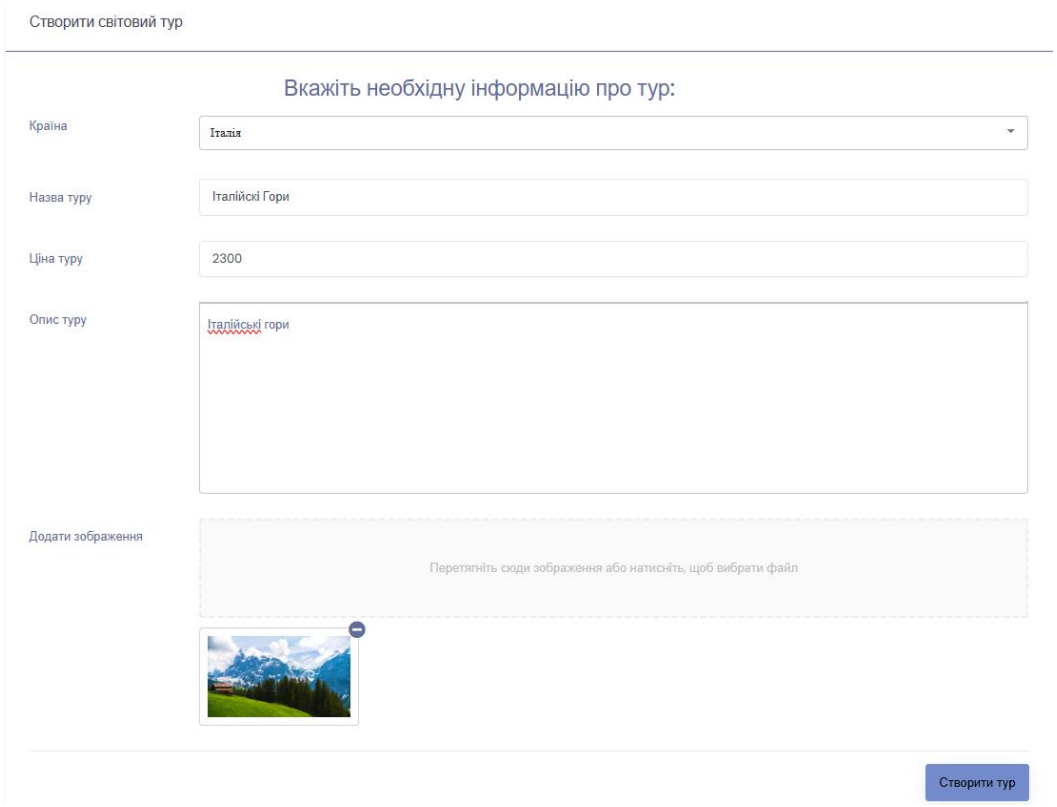

Рисунок 3.4 – Світовий тур(Адміністратор)

4. Модуль управління адміністраторами, що використовується для додавання, редагування або видалення облікових записів адміністраторів.

Алгоритм роботи:

— головний адміністратор переглядає список наявних адміністраторів;

— може створювати нові облікові записи, призначати або знімати ролі;

— усі зміни логуються для забезпечення прозорості.

Модуль управління адміністраторами, призначений для створення нових адміністраторів, та додавання їх до системи з певними правами, представлено на рис. 3.5.

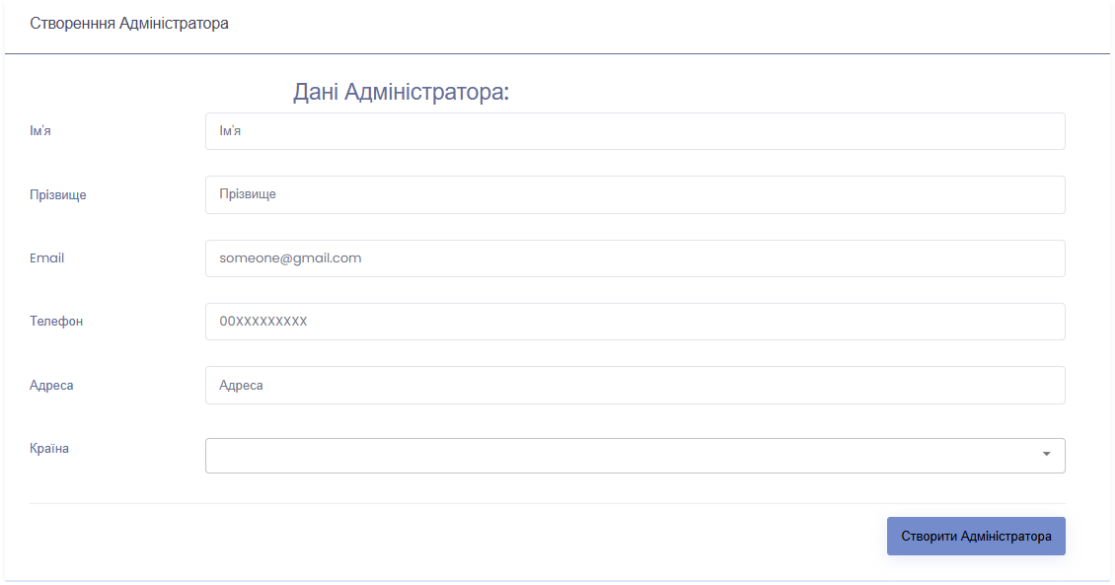

Рисунок 3.5 – Форма створення адміністратора

3.4 Опис функціональності та перевірка працездатності вебзастосунку

Для роботи із системою необхідний комп’ютер, який відповідає усім технічним вимогам, а також доступ до інтернету. Під час запуску системи користувач побачить вікно авторизації, де необхідно ввести адресу електронної пошти та пароль. Оскільки система має чітко визначені права доступу, введені дані перевіряються, і в залежності від ролі користувача він перенаправляється на відповідну сторінку. Там можна виконувати функції, передбачені для нього.

Система для авіаційних послуг надає користувачам можливість бронювання авіаквитків, перегляду спеціальних турів, пропозицій та блогу, а адміністраторам — керувати контентом і користувачами. Інструкція містить кроки для роботи з системою як для звичайного користувача, так і для адміністратора.

Використання системи як клієнта:

1. Реєстрація:

— перейдіть на сторінку реєстрації;

— заповніть усі обов’язкові поля (ім'я, електронна пошта, номер паспорта тощо);

— підтвердіть реєстрацію, натиснувши "Зареєструватися".

2. Авторизація

— введіть ваші облікові дані (електронна пошта та пароль) на сторінці входу;

— після успішного входу ви будете перенаправлені на головну сторінку системи.

3. Бронювання авіаквитків

— перейдіть до розділу "Бронювання";

— вкажіть маршрут (місто вильоту, місто прибуття), дати та кількість пасажирів;

— оберіть відповідний рейс із запропонованого списку;

— натисніть "Забронювати". Після цього ви будете перенаправлені на сторінку оплати.

4. Перегляд турів та спеціальних пропозицій

— у розділі "Тури" ви знайдете доступні пропозиції, такі як світові тури;

— натисніть на тур, щоб переглянути деталі, та, якщо бажаєте, здійсніть бронювання.

Тестування програми є важливим етапом розробки, оскільки під час цього процесу можна виявити недоліки системи, несправності у функціонуванні, некоректну роботу певних елементів або їх неправильне відображення.

Наприклад, пошук необхідного рейсу та його бронювання. Представлено на рисунках, 3.6 та 3.7.

На рис. 3.6 видно основні елементи для пошуку рейсів, такі як вибір пунктів вильоту та прибуття, дати, кількості дорослих та дітей. Це забезпечує користувача можливістю легко налаштовувати параметри пошуку. Список рейсів представлений у вигляді таблиці з основними даними — назва авіакомпанії, дата і час вильоту/прибуття, тривалість рейсу та ціна. Таке оформлення дозволяє користувачеві швидко оцінити доступні варіанти.

На рис. 3.7 видно детальну інформацію про вибраний рейс: маршрут, тарифи, класи обслуговування та індивідуальні ціни. Є чітка кнопка "Забронюйте зараз", що забезпечує інтуїтивність інтерфейсу для кінцевого користувача. Інформація структурована, що знижує ймовірність помилок і покращує користувацький досвід.

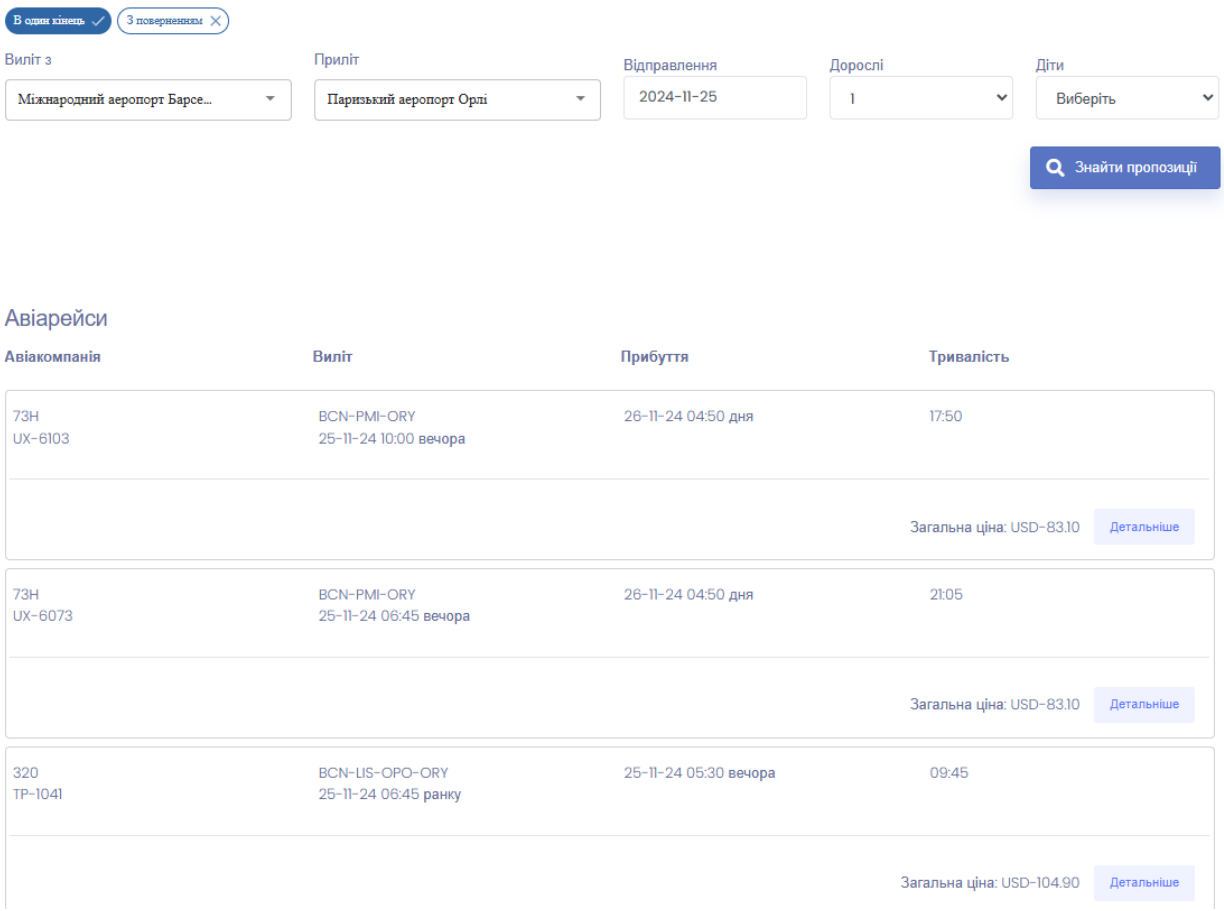

Рисунок 3.6 – Пошук рейсу

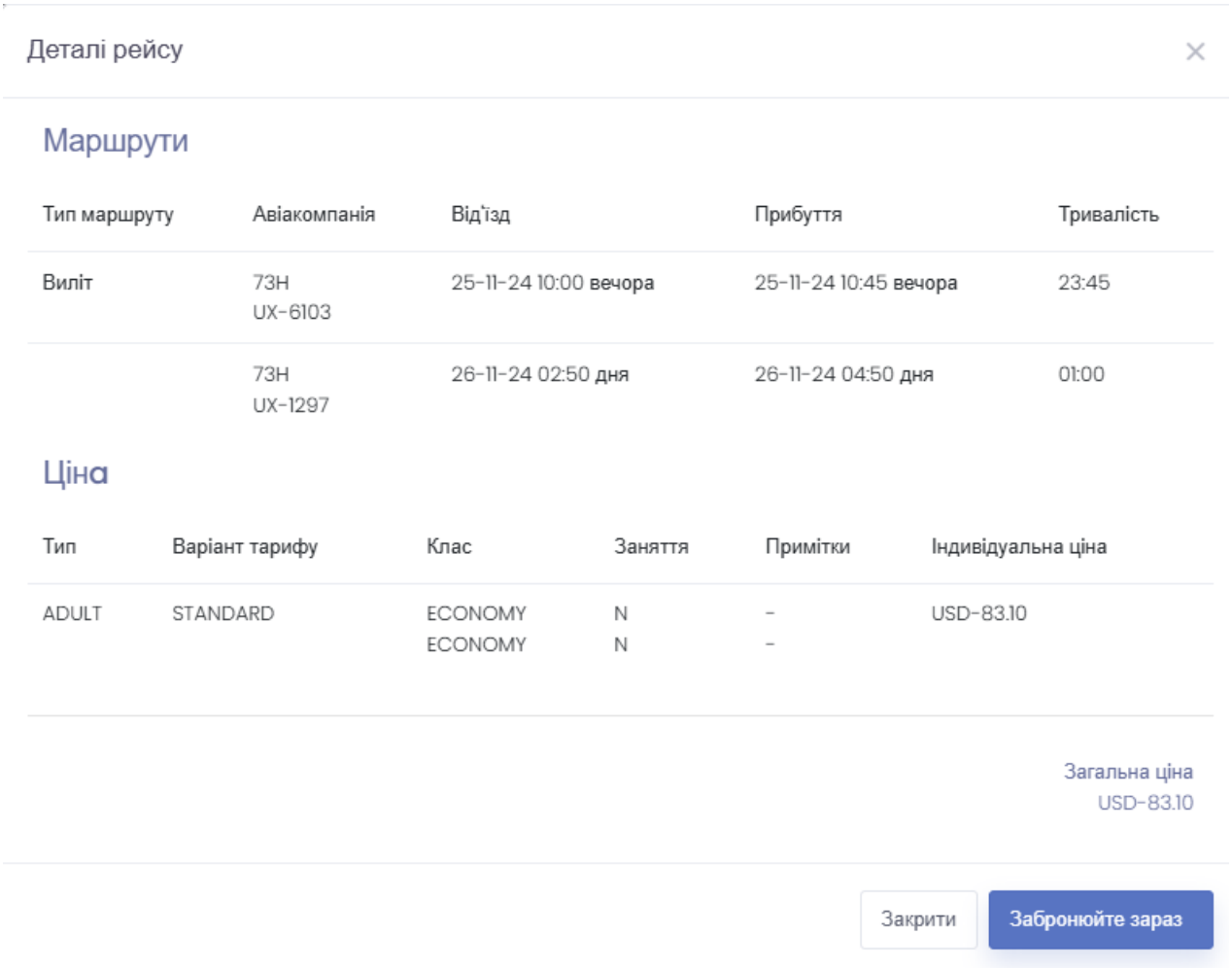

Рисунок 3.7 – Бронювання рейсу

Програмне рішення успішно автоматизує процеси пошуку та бронювання авіарейсів. Зображені екрани демонструють високий рівень зручності користування, що позитивно впливатиме на пасажирський досвід. Тестування функцій, таких як пошук і бронювання, дозволяє виявити можливі помилки або проблеми відображення, забезпечуючи надійну роботу системи після її впровадження.

На рисунку 3.8 показано, що після заповнення даних користувача (ім'я, прізвище, паспортні дані тощо) та вибору рейсу, користувач може здійснити бронювання, однак для завершення транзакції потрібне підтвердження зі сторони адміністратора. Це додає контроль з боку аеропорту для перевірки даних і уникнення дублювань чи некоректних бронювань.

Модуль бронювання передбачає автоматичну валідацію введених даних на відповідність (наприклад, перевірку формату електронної пошти чи номера телефону). Після підтвердження з боку адміністратора, клієнт отримує фінальне підтвердження бронювання разом із деталями рейсу (інформація про авіакомпанію, клас місця, ціну та додаткові примітки). Централізована перевірка адміністратором мінімізує можливі технічні помилки чи проблеми в обробці бронювань. Користувачі отримують можливість швидко уточнювати статус бронювання у разі потреби.

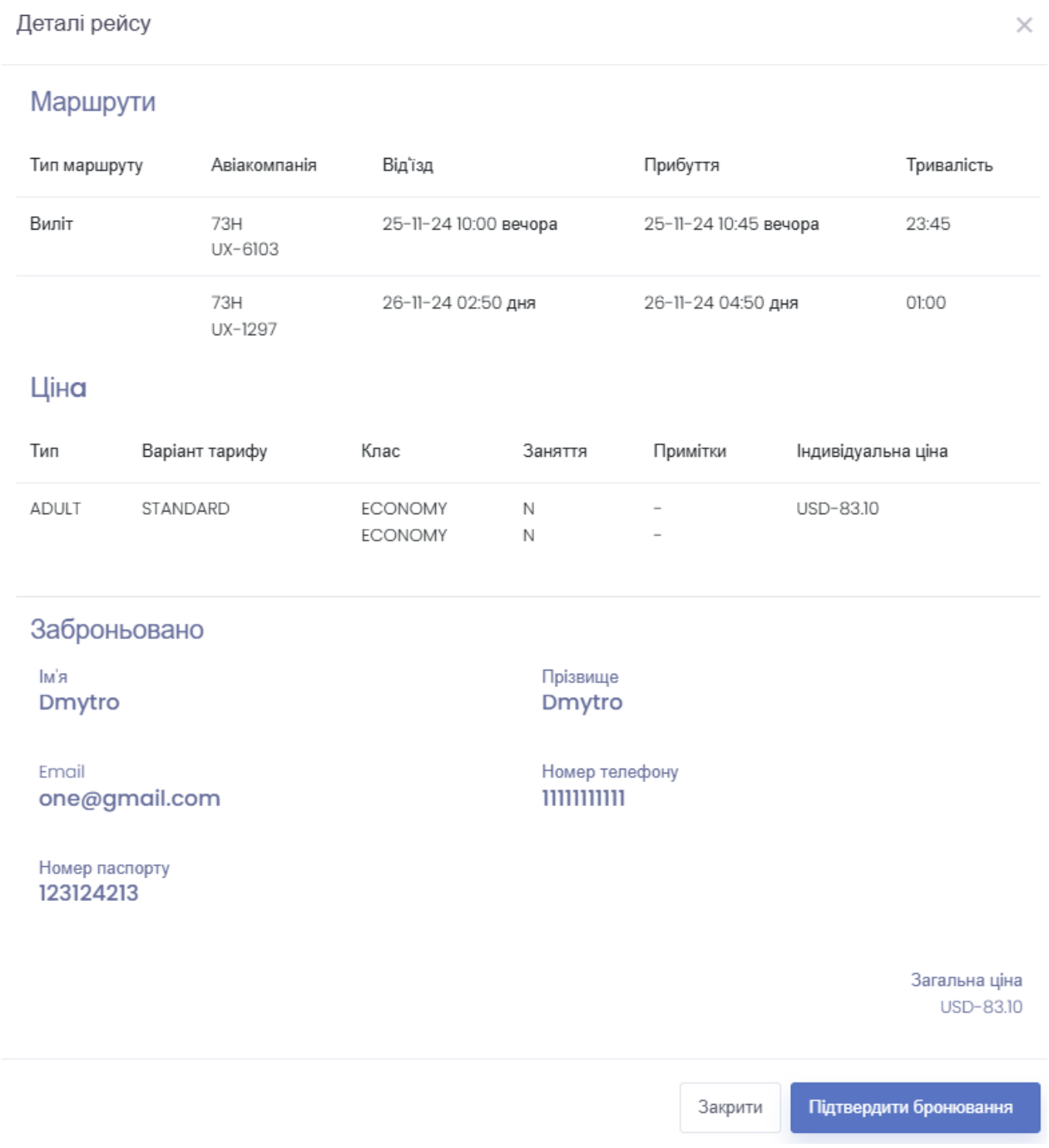

Рисунок 3.8 – Підтвердження бронювання

Тепер, після підтвердження бронювання клієнт може перейти до оплати свого квитка. На рисунку 3.9 представлено інтерфейс етапу оплати квитка після підтвердження бронювання. У цьому вікні користувач має можливість внести платіж за обраний рейс через введення наступних даних:

— Email — поле для введення електронної пошти пасажира, на яку буде надіслано підтвердження оплати і квиток;

— номер картки — основне поле для введення даних платіжної картки клієнта;

— MM/YY — поле для введення строку дії картки (місяць/рік);

— CVC — поле для введення тризначного коду безпеки картки.

Також представлена інформація про рейс:

— маршрут: показано авіакомпанію, номер рейсу, час відправлення та прибуття.

— ціна: вказана загальна сума до оплати в доларах США (USD 108.20).

Кнопка "Pay 108,00 $" (Сплатити) завершує процес оплати. Натискання на цю кнопку підтверджує транзакцію, після чого користувач отримує підтвердження бронювання на вказану електронну пошту.

Цей інтерфейс орієнтований на простоту використання, що дозволяє клієнту швидко і без зайвих труднощів здійснити оплату за рейс.

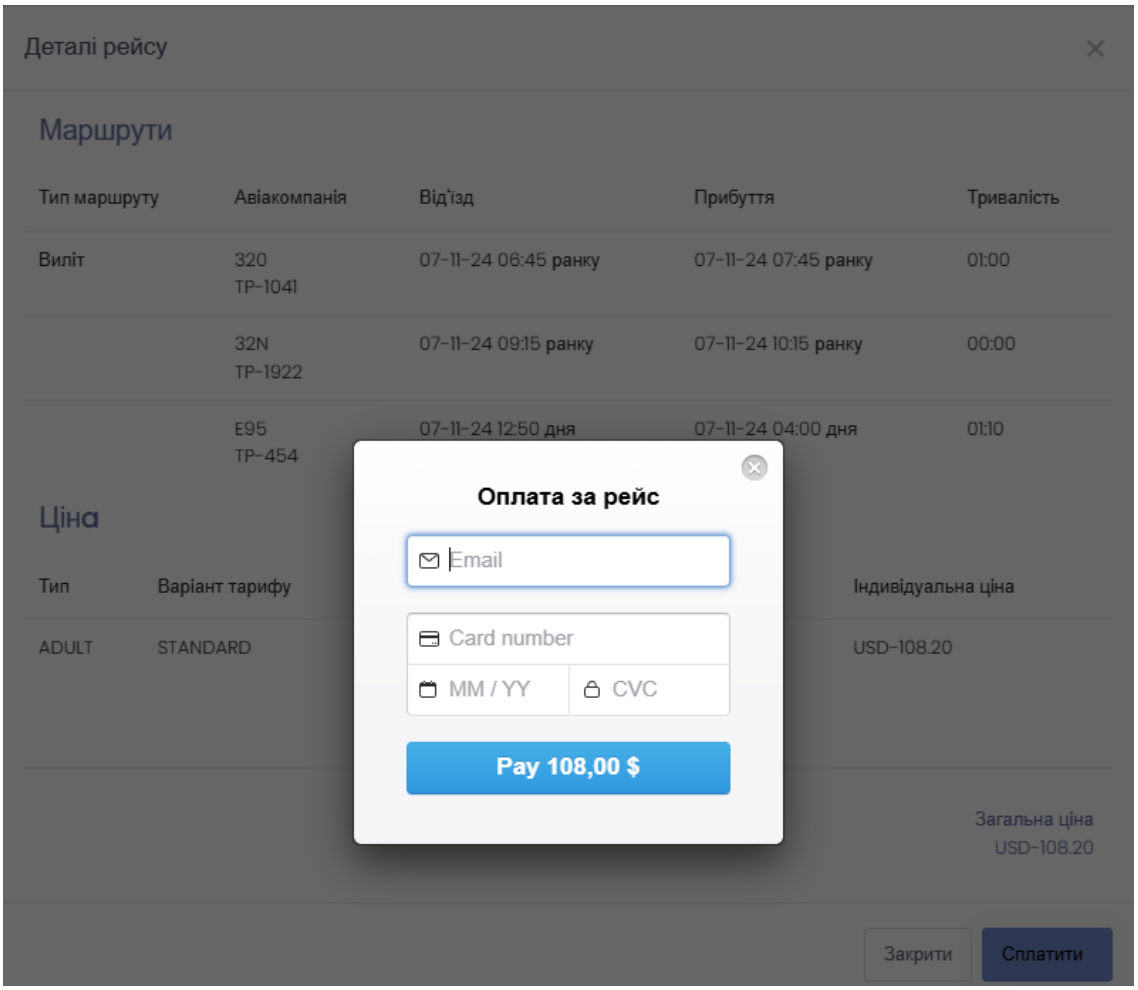

Рисунок 3.9 – Оплата квитка

Ця схема працює і зі світовими турами і зі спеціальними пропозиціями, але на відміну від бронювання рейсів інформація про які надходить з Amadeus пропозиції і тури створює адміністратор.

Висновки до розділу

У рамках цього розділу було проведено комплексний аналіз та розробку системи для автоматизації бронювання та оплати авіаквитків. Для реалізації системи було застосовано сучасні технології веб-розробки, а саме: HTML для створення структури веб-сторінок, CSS для стилізації інтерфейсу користувача, JavaScript та React для забезпечення інтерактивності та динамічної взаємодії з користувачем, Amadeus API як засіб інтеграції з базами даних авіарейсів, що дозволяє автоматизувати пошук рейсів і обробку запитів клієнтів. Для забезпечення стабільної роботи програми було визначено необхідні апаратні та програмні ресурси.

Система надає такі функції:

— пошук рейсів;

— бронювання;

— оплата;

— адміністрування;

Проведено детальне тестування функціоналу, яке включало перевірку таких аспектів:

— коректність роботи пошуку рейсів: система знаходить рейси відповідно до заданих параметрів;

— надійність оплати: всі тестові транзакції були успішно проведені без помилок у передачі даних;

— зручність інтерфейсу: користувачі змогли швидко виконати всі необхідні дії (пошук, бронювання, оплата).

Тестування підтвердило високу стабільність і зручність роботи системи. Користувачі можуть швидко виконувати всі етапи процесу — від пошуку рейсу до оплати. Крім того, адміністратор має всі необхідні інструменти для контролю за виконанням бронювань.

ВИСНОВКИ

Дана магістерська робота розкриває актуальні питання щодо автоматизації та оптимізації процесів керування прийому/відправлення літаків і обслуговування пасажирів. У ній було розглянуто вже поширені методи і підходи автоматизації та оптимізації цих процесів, проведено аналіз вже існуючих наукових робіт з цього питання.

В роботі було виконано розробку інформаційної системи, яка являється вебдодатком, що надає авіаційні послуги користувачам. Система орієнтована на вивчення потреб користувачів у сфері авіаперельотів і туристичних пропозицій. Найважливіші аспекти системи включають управління бронями, обробку цінових пропозицій, а також роботу з даними користувача. Основним завданням системи є управління процесом бронювання квитків та спеціальними пропозиціями. Система використовує пошук, фільтрація даних та моделювання стану бронювання, для зручного управління процесами та для перегляду усіх забронюваних рейсів на різних їх етапах.

Для розробки цього проєкту були використані такі технології:

— HTML –за допомогою HTML створюються заголовки, списки, таблиці, форми, кнопки та інші елементи, які дозволяють користувачам взаємодіяти із системою.

— CSS –дозволяє налаштовувати кольори, шрифти, розміри, відступи та інші аспекти дизайну, забезпечуючи гармонійний та привабливий вигляд інтерфейсу.

— JavaScript –додає інтерактивність до вебсторінок, дозволяє обробляти дії користувача, виконувати асинхронні запити до сервера та проводити валідацію даних без необхідності перезавантаження сторінки.

— React –дозволяє розробникам працювати зі складними структурами даних завдяки компонентному підходу, що сприяє модульності та повторному використанню коду. React також забезпечує високу продуктивність завдяки

віртуальному DOM, який мінімізує кількість змін у реальному DOM під час оновлення інтерфейсу.

— Amadeus API- за допомогою API отримується інформація про доступні рейси, маршрути, ціни та інші дані, необхідні для користувачів системи. Ця технологія значно спрощує доступ до актуальної інформації про авіапослуги.

— MongoDB – забезпечує гнучке зберігання інформації у вигляді документів у форматі JSON, що дозволяє ефективно працювати з великими обсягами даних. MongoDB також підтримує масштабування та високу продуктивність при виконанні запитів.

У ході тестування було виявлено, що функціонал, призначений для користувачів, працює належним чином. Зокрема, клієнт може здійснювати пошук відповідних рейсів на основі вхідних даних, та бронювати і сплачувати їх після підтвердження адміністратором.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Панасюк І. П. Формування тарифної політики авіакомпанії : дис. канд. екон. наук : 08.00.04 / Ірина Петрівна Панасюк ; Національний авіаційний університет. Київ, 2015. 220 с. Режим доступу: http://er.nau.edu.ua/handle/NAU/13992(DSpace).

2. Essentials of Aviation Operations and Management. *Defense and Military Contracting for Aerospace Parts*. URL: https://www.governmentprocurement.com/news/aviation-operations-management#:~:text=Aviation%20operations%20and%20management%20encompass,flight%20operations,%20and%20maintenance%20tasks.

3. Шляхи формування раціональної організаційної структури авіаційної частин. URL: : https://www.ukrmilitary.com/2017/03/aviation-organizational-structure.html

4. The Future of Air Travel: Innovations and Trends - Airport Gurús. *Airport Gurus*. URL: https://www.airportgurus.com/en/the-future-of-air-travel-innovations-and-trends/ (date of access: 07.08.2024).

5. Choi J. H. Changes in airport operating procedures and implications for airport strategies post-COVID-19. *Journal of Air Transport Management*. 2021. Vol. 94. P. 102065. URL: https://doi.org/10.1016/j.jairtraman.2021.102065 (date of access: 07.08.2024).

6. Department of Aeronautical Sciences Institute of Civil Aviation. The 3rd Aviation National Symposium – Proceeding book. Bangkok : Institute of Civil Aviation Ministry of Transport, 2023. 232 p.

7. Manasak Pamornmalirat. The Rise of Automated Technologies in Passenger Handling Process at the Airport. *The 3rd Aviation National Symposium – Proceeding book*. Thailand, 2023. P. 62–70 (date of access: 10.08.2024).

8. Academia.edu. Academia.edu. URL: https://www.academia.edu/RegisterToDownload/undefined(date of access: 11.08.2024).

9. Ayodeji Y., Rjoub H., Özgit H. Achieving sustainable customer loyalty in airports: The role of waiting time satisfaction and self-service technologies. *Technology in Society*. 2022. P. 102106. URL: https://doi.org/10.1016/j.techsoc.2022.102106 (date of access: 10.08.2024).

10. Miguel Mujica Mota, Paolo Scala, Michael Schultz. The rise of the Smart Passenger I: analysis of impact on departing passenger flow in airports. 7–9 December 2021.

11. Dvorakova T., Vittek P., Nagy I. Impact of Smart Technologies on Passenger Flow and Queueing at Airports. *2022 New Trends in Civil Aviation (NTCA)*, Prague, Czech Republic, 26–27 October 2022. 2022. URL: https://doi.org/10.23919/ntca55899.2022.9934165 (date of access: 15.08.2024).

12. The Analysis and AI Simulation of Passenger Flows in an Airport Terminal: A Decision-Making Tool. *MDPI*. URL: https://www.mdpi.com/2071-1050/16/3/1346 (date of access: 15.08.2024).

13. Imhotep - integrated multimodal airport operations for efficient flow management. *IMHOTEP*. URL: https://www.imhotep-h2020.eu/ (date of access: 16.08.2024).

14. Optimization of resource demand for passenger services at airports during system failures such as blackouts / L.-M. Brause et al. *European Transport Research Review*. 2020. Vol. 12, no. 1. URL: https://doi.org/10.1186/s12544-020-00446-2 (date of access: 16.08.2024).

15. Processing passengers efficiently: An analysis of airport processing times for international passengers / J. Pitchforth et al. *Journal of Air Transport Management*. 2015. Vol. 49. P. 35–45. URL: https://doi.org/10.1016/j.jairtraman.2015.06.016 (date of access: 16.08.2024).

16. Як змінювався пасажиропотік найбільших аеропортів України. *Слово і Діло*. URL: https://www.slovoidilo.ua/2021/07/20/infografika/suspilstvo/yak-zminyuvavsya-pasazhyropotik-najbilshyx-aeroportiv-ukrayiny (дата звернення: 26.08.2024).

17. Статистичні дані в галузі авіатранспорту. *Міністерство інфраструктури України*. URL: https://mtu.gov.ua/content/statistichni-dani-v-galuzi-aviatransportu.html (дата звернення: 26.08.2024).

18. Шаховська Н. Б. Проектування інформаційних систем: Навчальний посібник / Н. Б. Шаховська, В. В. Литвин; за ред. В. В. Пасічника. Львів: «Магнолія 2006», 2017. – 380 с.

19. Ideal Modeling & Diagramming Tool for Agile Team Collaboration. *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. URL: https://www.visual-paradigm.com/ (date of access: 30.08.2024).

20. Петрик М. Р. Моделювання програмного забезпечення : науково-методичний посібник / М. Р. Петрик, О. Ю. Петрик. Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2015. – 200 с.

21. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т. І. Каплієнко, О. А. Петрова. Запоріжжя : Дике Поле, 2016. – 250 с.

22. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дудук В. І. Інформаційні технології та моделювання бізнес-процесів. Навч. посібник. / Телишевський О. М., Цегелик Г. Г., Вітер М. Б., Дідук В. І. К.: Видавництво «Центр учбової літератури», 2012. – 296 с.

23. Конєва А. Перспективи застосування сучасних систем автоматизації. *AUTOMATION AND DEVELOPMENT OF ELECTRONIC DEVICES*. 2022. УДК 687.02. URL: https://openarchive.nure.ua/server/api/core/bitstreams/641885b5-d764-433e-b4a9-3ec9b74930d9/content.

24. Репозитарій Національного авіаційного університету. Особливості організації та розвитку аеропортів України в сучасних умовах // er.nau.edu.ua. – 2021. URL: https://er.nau.edu.ua (дата звернення: 17.09.2024).

25. Цифровізація бізнес-процесів у цивільній авіації України // Економіка та держава, 2022. – №5. – URL: https://economics.org.ua.

26. Тронь В. В., Маринич І. А. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 174 - Автоматизація, комп’ютерно-інтегровані технології та робототехніка”. Кривий Ріг: Видавничий центр КНУ, 2022. 50 с.

27. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с.(Інформація та документація).

28. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с.(Інформація та документація).

29. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с.(Інформація та документація).

30. ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с.(Інформація та документація).

ДОДАТОК А

Створення Адміністратора-----

```
import React, {useState} from 'react';
import {Alert} from "react-bootstrap";
import {Portlet, PortletBody, PortletHeader} from "../../partials/content/Portlet";
import {ErrorMessage, Field, Form, Formik} from "formik";
import {createAdmin} from "../../crud/auth.crud";
import {formErrorMessage} from "../errors/FormErrorMessage";
import InputCountry from "../../Components/input/InputCountry";
import clsx from "clsx";
import {useHistory} from 'react-router-dom'
import {adminCreateValidations} from "../../../utils/validations/adminCreateValidations";
const CreateAdmin = () => {
  const history = useHistory()
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState({show: false, message: ''});
  const [success, setSuccess] = useState({show: false, message: ''});
  const [loadingButtonStyle, setLoadingButtonStyle] = useState({
    paddingRight: "1rem"
  });

  const enableLoading = () => {
    setLoading(true);
    setLoadingButtonStyle({ paddingRight: "3.5rem" });
  };

  const disableLoading = () => {
    setLoading(false);
    setLoadingButtonStyle({ paddingRight: "1rem" });
  };
  const closeAlert = () => {
    setTimeout(() => {
      setError({show: false, message: ''})
      setSuccess({show: false, message: ''})
    }, 3000)
  }
  return (
    <div>
      <Alert show={success.show} variant="success">{success.message}</Alert>
      <Alert show={error.show} variant="danger">{error.message}</Alert>
```

```
    <Portlet    className="kt-portlet--height-fluid-half    kt-portlet--border-bottom-
brand">
      <PortletHeader
       title='Створеннння Адміністратора'
      />
      <PortletBody>
       <div className="row container">
        <Formik
         initialValues={{
          firstName: '',
          lastName: '',
          email: '',
          address: '',
          country: '',
          mobileNo: ''
         }}
         validate={adminCreateValidations}
         onSubmit={(values, { setStatus, setSubmitting, resetForm }) => {
          enableLoading();
          createAdmin({...values})
           .then(res => {
            if (!res.data.success) {
             disableLoading();
             setError({show: true, message: res.data.message})
             closeAlert()
            } else {
             disableLoading();
             setSuccess({show: true, message: res.data.message})
             closeAlert()
             setTimeout(() => {
              history.push('/admins')
             }, 2000)
            }
            setSubmitting(false)
           })
           .catch((error) => {
            disableLoading();
            setError({show: true, message: 'Could not create lawyer!'})
            setSubmitting(false)
            closeAlert()
           });
         }}
        >
         {({handleSubmit, isSubmitting}) =>
          <Form onSubmit={handleSubmit} className='w-100'>
```

```
            <div className="row">
              <label className="col-xl-3" />
              <div className="col-lg-9 col-xl-6">
                <h3 className="kt-section__title kt-section__title-sm">
                  Дані Адміністратора:
                </h3>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Ім'я
              </label>
              <div className="col-10">
                <Field              className="form-control"              name='firstName'
placeholder="Ім'я"/>
                <ErrorMessage name='firstName' render={formErrorMessage}/>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Прізвище
              </label>
              <div className="col-10">
                <Field              className="form-control"              name="lastName"
placeholder="Прізвище"/>
                <ErrorMessage name='lastName' render={formErrorMessage}/>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Email
              </label>
              <div className="col-10">
                <Field     type='email'     className="form-control"     name="email"
placeholder="someone@gmail.com"/>
                <ErrorMessage name='email' render={formErrorMessage}/>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Телефон
              </label>
              <div className="col-10">
                <Field              className="form-control"              name="mobileNo"
placeholder="00XXXXXXXX"/>
```

```
                <ErrorMessage name='mobileNo' render={formErrorMessage}/>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Адреса
              </label>
              <div className="col-10">
                <Field                 className="form-control"                 name="address"
placeholder="Адреса"/>
                <ErrorMessage name='address' render={formErrorMessage}/>
              </div>
            </div>
            <div className="form-group row">
              <label className="col-2 col-form-label">
                Країна
              </label>
              <div className="col-10">
                <InputCountry/>
                <ErrorMessage name='country' render={formErrorMessage}/>
              </div>
            </div>

            <div className="kt-form__actions pt-3" style={{borderTop: '1px solid
#ebedf2'}}>
              <div className="d-flex justify-content-end">
                <button
                  type="submit"
                  className={`btn btn-primary btn-elevate kt-login__btn-primary
${clsx(
                    {
                      "kt-spinner kt-spinner--right kt-spinner--md kt-spinner--light":
loading
                    }
                  )}`}
                  style={loadingButtonStyle}
                  disabled={isSubmitting}
                >
                  Створити Адміністратора
                </button>
              </div>
            </div>
          </Form>
        }
      </Formik>
```

```
        </div>
       </PortletBody>
     </Portlet>
   </div>
  );
};

export default CreateAdmin;
```

Створення ТУРА----------

```
import React, { useState } from "react";
import {
  Portlet,
  PortletBody,
  PortletHeader,
  PortletHeaderToolbar
} from "../../partials/content/Portlet";
import { useHistory } from "react-router-dom";
import { Alert } from "react-bootstrap";
import { ErrorMessage, Field, Form, Formik } from "formik";
import { createWorldTour } from "../../crud/flights.crud";
import { formErrorMessage } from "../errors/FormErrorMessage";
import InputCountry from "../../Components/input/InputCountry";
import Dropzone from "react-dropzone";
import CKEditor from "@ckeditor/ckeditor5-react";
import ClassicEditor from "@ckeditor/ckeditor5-build-classic";
import clsx from "clsx";
import { worldTourCreateValidations } from
"../../../utils/validations/worldTourCreateValidations";

const CreateWorldTour = () => {
  const history = useHistory();
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState({ show: false, message: "" });
  const [success, setSuccess] = useState({ show: false, message: "" });
  const [loadingButtonStyle, setLoadingButtonStyle] = useState({
    paddingRight: "1rem"
  });

  const enableLoading = () => {
```

```
    setLoading(true);
    setLoadingButtonStyle({ paddingRight: "3.5rem" });
  };

  const disableLoading = () => {
    setLoading(false);
    setLoadingButtonStyle({ paddingRight: "1rem" });
  };
  const closeAlert = () => {
    setTimeout(() => {
      setError({ show: false, message: "" });
      setSuccess({ show: false, message: "" });
    }, 3000);
  };
  const onDrop = (acceptedFiles, setFieldValue) => {
    setFieldValue("packageImage", acceptedFiles[0]);
  };
  const onDropReject = files => {
    setError({ show: true, message: "Could not accept this file" });
    setTimeout(() => {
      setError({ show: false, message: "" });
    }, 2000);
  };
  const handleClickRemoveImage = setFieldValue => {
    setFieldValue("packageImage", null);
  };
  return (
    <div className="pb-5">
      <Portlet     className="kt-portlet--height-fluid-half     kt-portlet--border-bottom-
brand">
        <PortletHeader title="Створити світовий тур" />
        <PortletBody>
          <Alert show={success.show} variant="success">
            {success.message}
          </Alert>
          <Alert show={error.show} variant="danger">
            {error.message}
          </Alert>
          <div className="row container">
            <Formik
              initialValues={{
                country: "",
                packageTitle: "",
                packagePrice: "",
                packageDescription: "",
```

```
    packageImage: null
  }}
  validate={worldTourCreateValidations}
  onSubmit={(values, { setStatus, setSubmitting, resetForm }) => {
    enableLoading();
    const formData = new FormData();
    formData.append("file", values.packageImage);
    formData.append(
      "details",
      JSON.stringify({ ...values, packageImage: undefined })
    );
    createWorldTour(formData)
      .then(res => {
        if (!res.data.success) {
          disableLoading();
          setError({ show: true, message: res.data.message });
          closeAlert();
        } else {
          disableLoading();
          setSuccess({ show: true, message: res.data.message });
          closeAlert();
          setTimeout(() => {
            history.push("/world-tour");
          }, 2000);
        }
        setSubmitting(false);
      })
      .catch(error => {
        disableLoading();
        setError({ show: true, message: "Could not create Tour!" });
        setSubmitting(false);
        closeAlert();
      });
  }}
>
  {({
    handleSubmit,
    isSubmitting,
    values,
    setFieldValue,
    errors
  }) => (
    <Form onSubmit={handleSubmit} className="w-100">
      <div className="row">
        <label className="col-xl-3" />
```

```
  <div className="col-lg-9 col-xl-6">
    <h3 className="kt-section__title kt-section__title-sm">
      Вкажіть необхідну інформацію про тур:
    </h3>
  </div>
</div>
<div className="form-group row">
  <label className="col-2 col-form-label">Країна</label>
  <div className="col-10">
    <InputCountry />
    <ErrorMessage name="country" render={formErrorMessage} />
  </div>
</div>
<div className="form-group row">
  <label className="col-2 col-form-label">
    Назва туру
  </label>
  <div className="col-10">
    <Field
      className="form-control"
      name="packageTitle"
      placeholder="Назва туру"
    />
    <ErrorMessage
      name="packageTitle"
      render={formErrorMessage}
    />
  </div>
</div>
<div className="form-group row">
  <label className="col-2 col-form-label">
    Ціна туру
  </label>
  <div className="col-10">
    <Field
      className="form-control"
      name="packagePrice"
      type="number"
      placeholder="Ціна туру"
    />
    <ErrorMessage
      name="packagePrice"
      render={formErrorMessage}
    />
  </div>
```

```
</div>
<div className="form-group row">
  <label className="col-2 col-form-label">
    Опис туру
  </label>
  <div className="col-10">
    <CKEditor
      editor={ClassicEditor}
      onChange={(event, editor) => {
        setFieldValue("packageDescription", editor.getData());
      }}
      config={{
        ckfinder: {
          uploadUrl:
            "/api/flights/world-tour/editorImage/images"
        },
        toolbar: {
          items: [
            // "heading",
            // "|",
            // "bold",
            // "italic",
            // "|",
            // "bulletedList",
            // "numberedList",
            // "indent",
            // "|",
            // "undo",
            // "redo"
          ]
        }
      }}
    />
    <ErrorMessage
      name="packageDescription"
      render={formErrorMessage}
    />
  </div>
</div>
<div className="form-group row">
  <label className="col-2 col-form-label">Додати зображення</label>
  <div className="col-10">
    <div>
      <Dropzone
        onDrop={acceptedFiles =>
```

```
                onDrop(acceptedFiles, setFieldValue)
              }
              accept="image/*"
              onDropRejected={onDropReject}
              multiple={false}
            >
              {({ getRootProps, getInputProps, isDragActive }) =>
                getRootProps &&
                getInputProps && (
                  <section>
                    <div
                      {...getRootProps({
                        className: `base-style ${
                          isDragActive ? "active-style" : ""
                        }`
                      })}
                    >
                      <input {...getInputProps()} />
                      {isDragActive ? (
                        <span>Перетягніть файл сюди ...</span>
                      ) : (
                        <span>
                          Перетягніть сюди зображення або натисніть, щоб вибрати файл
                        </span>
                      )}
                    </div>
                  </section>
                )
              }
            </Dropzone>
          </div>
          <ErrorMessage
            name="packageImage"
            render={formErrorMessage}
          />
          {values.packageImage && (
            <div className="pdf-uploaded">
              <img
                src={URL.createObjectURL(values.packageImage)}
                alt="packageImage"
                width={150}
              />
              <div
                className="fa fa-minus-circle"
```

```
                    onClick={() =>
                      handleClickRemoveImage(setFieldValue)
                    }
                   />
                  </div>
                 )}
                </div>
               </div>

               <div
                className="kt-form__actions pt-3"
                style={{ borderTop: "1px solid #ebedf2" }}
               >
                <div className="d-flex justify-content-end">
                 <button
                  type="submit"
                  className={`btn btn-primary btn-elevate kt-login__btn-primary
${clsx(
                   {
                    "kt-spinner kt-spinner--right kt-spinner--md kt-spinner--light":
loading
                   }
                  )}`}
                  style={loadingButtonStyle}
                  disabled={isSubmitting}
                 >
                  Створити тур
                 </button>
                </div>
               </div>
              </Form>
             )}
            </Formik>
           </div>
          </PortletBody>
         </Portlet>
        </div>
       );
};

export default CreateWorldTour;
```