МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Кодлубовської Анжеліки Олександрівни
(прізвище, ім'я, по батькові здобувача)

на тему Розробка комплексного веб-додатку туристичної фірми з підсистемою адміністрування замовлень та інтерактивної взаємодії з користувачами
(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник к.п.н., доцент _____ Павлиш Т.Г.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 12 червня 2026 р. № 15
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384
Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ ЗДОБУВАЧУ

_____Кодлубовської Анжеліки Олександрівни_____

1. Тема роботи Розробка комплексного веб-додатку туристичної фірми з підсистемою адміністрування замовлень та інтерактивної взаємодії з користувачами

науковий керівник роботи _____ к.п.н., доцент, Павлиш Тетяна Григорівна,
затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 193-ст

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної бакалаврської роботи, об'єкт, предмет та мета дослідження:

Розділ 1 - Теоретичні засади та аналіз предметної області розробки веб-додатків туристичної фірми
Розділ 2 - Проектування архітектури та моделей системи веб-додатку туристичної фірми
Розділ 3 - Програмна реалізація веб-додатку туристичної фірми з підсистемою замовлень та взаємодії з користувачами

Об'єкт дослідження – процеси автоматизації обліку замовлень, адміністрування даних та інтерактивної взаємодії з клієнтами в туристичній індустрії.

Предмет дослідження – методи, технології та програмні засоби розробки клієнт-серверного вебдодатку туристичної фірми на базі фреймворків Flask та React.

Мета кваліфікаційної бакалаврської роботи – Підвищення ефективності роботи менеджерів туристичної агенції та покращення користувацького досвіду клієнтів шляхом проєктування і розробки кастомного вебдодатку з гнучкими інструментами фільтрації, безпечною авторизацією та Особистим кабінетом.

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	
2	Підготовка розділу 2	до 08.05.2026р.	
3	Підготовка розділу 3	до 25.05.2026р.	
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	
5	Отримання відгуку від наукового керівника	04.06.2026р.	
6	Отримання зовнішньої рецензії	05.06.2026р.	
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	
8	Перевірка кваліфікаційної роботи на плагіат	12.06.2026р.	
9	Допуск кафедрою кваліфікаційної роботи до захисту	12.06.2026р.	
10	Підготовка студента до захисту в ЕК	до 19.06.2026р.	

Завдання підготував науковий керівник _____ __Павлиш Т.Г.___
(підпис) (прізвище та ініціали)

Завдання одержав здобувач _____ __Кодлубовська А.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Робота містить 69 сторінок машинописного тексту, 28 рисунків, 1 таблицю, 1 додаток. При підготовці було використано 30 джерел.

Об’єктом дослідження є процеси автоматизації взаємодії з клієнтами та управління контентом у діяльності сучасних туристичних підприємств.

Ціль роботи полягає у теоретичному обґрунтуванні, проєктуванні та програмній реалізації комплексного вебдодатка для туристичної фірми, який забезпечує автоматизовану обробку замовлень та ефективне керування базою турів.

У роботі використано системний та порівняльний аналіз для обґрунтування технологічного стека, методи об’єктно-орієнтованого проєктування для створення архітектури системи та моделювання баз даних, а також методи емпіричного тестування для перевірки працездатності програмного забезпечення. Розробка здійснювалася на базі персонального комп'ютера з використанням сучасного середовища розробки (IDE), серверних технологій Python і Flask/Django та бібліотеки React для фронтенд-частини.

Розроблено оригінальну архітектуру вебдодатка, що інтегрує клієнтський інтерфейс та адміністративну панель у єдину екосистему. Новизна результатів полягає в удосконаленні механізмів динамічного управління контентом та реалізації модуля оперативного опрацювання клієнтських запитів, адаптованого до специфіки малих туристичних агенцій.

Створений програмний продукт базується на клієнт-серверній архітектурі. Серверна частина забезпечує безпеку даних та обробку бізнес-логіки, а клієнтська частина – адаптивність інтерфейсу та високу швидкість відгуку. Система характеризується низькими вимогами до технічного забезпечення користувача, інтуїтивно зрозумілим інтерфейсом адміністратора та можливістю масштабування функціоналу.

Результати роботи були апробовані у межах пілотного проєкту для туристичної агенції «LikiaTours». Розроблений вебдодаток успішно пройшов етап функціонального тестування та готовий до розгортання на хостинг-платформах для реальної експлуатації.

Дане дослідження базується на сучасних концепціях побудови Full-stack додатків та розвиває ідеї цифровізації сфери послуг, що розглядалися в попередніх науково-технічних проєктах кафедри.

Програмний продукт рекомендується до використання малим та середнім туристичним компаніям для оптимізації роботи менеджерів та підвищення лояльності клієнтів.

Галузь туризму та гостинності, а також інформаційні системи управління взаємовідносинами з клієнтами.

Впровадження системи дозволяє знизити витрати часу на обробку одного замовлення на 30–40%, мінімізувати помилки в бронюванні та підвищити прозорість взаємодії між клієнтом і компанією.

Значимість роботи полягає у створенні готового до використання інструменту цифрової трансформації бізнесу, що сприяє підвищенню конкурентоспроможності вітчизняних туристичних операторів.

Висновки та пропозиції. У ході виконання роботи було повністю досягнуто поставленої мети. Подальший розвиток об’єкта розроблення доцільно спрямувати на інтеграцію платіжних шлюзів, впровадження системи автоматичних сповіщень та розширення аналітичного модуля для збору статистики вподобань користувачів.

АВТОМАТИЗАЦІЯ, БАЗА ДАНИХ, ВЕБДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ОНЛАЙН-БРОНЮВАННЯ, ПРОЄКТУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТУРИСТИЧНА АГЕНЦІЯ, FULL-STACK РОЗРОБКА.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних

ЗП – запит користувача

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ТЗ – технічне завдання

API – Application Programming Interface (інтерфейс програмування додатків)

CORS – Cross-Origin Resource Sharing (спільне використання ресурсів різних джерел)

CRUD – Create, Read, Update, Delete (операції створення, читання, оновлення, видалення)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

JS – JavaScript (мова програмування сценаріїв)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

JSX – JavaScript XML (розширення синтаксису JavaScript)

MVC – Model-View-Controller (схема розділення даних додатка)

REST – Representational State Transfer (архітектурний стиль взаємодії компонентів)

SPA – Single Page Application (односторінковий вебдодаток)

SQL – Structured Query Language (мова структурованих запитів)

ВСТУП

Стрімка цифрова трансформація світової економіки та посилення конкуренції на ринку туристичних послуг змушують сучасні компанії шукати інноваційні шляхи оптимізації внутрішніх бізнес-процесів. У сучасних умовах традиційні методи комунікації та ручна обробка замовлень поступово втрачають свою ефективність, що призводить до зниження швидкості обслуговування та підвищення ризику виникнення помилок через людський фактор. Актуальність теми даної роботи зумовлена необхідністю створення високоефективних вебдодатків, які поєднують інтерактивну взаємодію з користувачем та потужний функціонал для автоматизованого керування даними. Це дозволяє туристичним фірмам забезпечувати високий рівень сервісу в режимі реального часу та оперативно адаптуватися до мінливих вимог ринку.

Ступінь вивченості проблеми свідчить про те, що питання впровадження інформаційних технологій у туристичну сферу та розробки систем управління взаємовідносинами з клієнтами (CRM) активно досліджувалися як вітчизняними, так і закордонними науковцями та практиками. Значний внесок у розвиток методології проєктування інформаційних систем та принципів архітектури програмного забезпечення зробили такі провідні фахівці, як Едсгер Дейкстра, Роберт Мартін [1, 2], Мартін Фаулер та Греді Буч. Їхні напрацювання у сфері об’єктно-орієнтованого аналізу та чистих архітектур стали фундаментом для побудови надійних систем. Питаннями розробки високонавантажених вебдодатків та ефективної взаємодії клієнтської і серверної частин займалися такі дослідники, як Рой Філдінг (автор архітектурного стилю REST) та фахівці-практики, що розвивали сучасні фреймворки, зокрема Ден Абрамов (екосистема React) [3, 4] та Армін Ронахер (мікрофреймворки на Python). Серед вітчизняних вчених, які досліджували інформаційні технології в управлінні, слід відзначити праці В. М. Глушкова та О. І. Кульчицького, чиї ідеї щодо автоматизації бізнес-процесів не втрачають актуальності й сьогодні. Попри наявність універсальних рішень, проблема розробки кастомних, легких та гнучких вебдодатків, що

максимально враховують специфіку діяльності локальних туристичних операторів, залишається актуальною і потребує подальшого розроблення.

Специфіка джерельної бази дослідження визначається комплексним підходом до використання інформації. Інформаційну базу роботи склали фундаментальні наукові праці в галузі побудови інформаційних систем, матеріали професійних видань, присвячених цифровізації економіки, а також офіційна технічна документація сучасних інструментів розробки (зокрема Python, Flask/Django та React). Важливе місце в обґрунтуванні розробки посіли аналітичні звіти щодо тенденцій розвитку цифрового туризму та фактичні дані про діяльність сучасних туристичних агенцій, що дозволило виявити ключові недоліки існуючих процесів обробки замовлень.

Об'єктом дослідження є процес автоматизації взаємодії з клієнтами та управління контентом у діяльності туристичних підприємств. Даний процес охоплює всі етапи роботи з клієнтом: від первинного перегляду доступних турів на сайті до фіксації замовлення в адміністративній системі та подальшої комунікації з менеджером.

Предметом дослідження виступають методи, технології та інструментальні засоби проєктування, розробки та тестування сучасного програмного забезпечення, що забезпечує функціонування комплексного вебдодатка для туристичної галузі.

Метою роботи є отримання конкретного програмного результату – теоретичне обґрунтування, архітектурне проєктування та програмна реалізація вебдодатка для туристичної фірми, який забезпечує автоматизовану обробку клієнтських запитів та надає зручний інструментарій для динамічного керування внутрішньою базою турів.

В процесі дослідження для досягнення мети вирішувалися наступні завдання: по-перше, здійснено аналіз сучасного стану та виявлено суть проблемних ситуацій у цифровізації туристичного бізнесу; по-друге, сформовано детальні функціональні та технічні вимоги до системи; по-третє, спроєктовано логічну структуру бази даних та загальну архітектуру програмного

продукту; по-четверте, реалізовано клієнтську та серверну частини додатка із застосуванням сучасних стандартів кодування; і, нарешті, проведено комплексне тестування розробленого рішення на відповідність поставленим вимогам.

Методика дослідження базується на алгоритмі системного підходу. Результати роботи були отримані за допомогою поєднання методів об'єктно-орієнтованого аналізу, моделювання структур даних та емпіричних методів тестування готового програмного коду. Робота виконувалася з використанням сучасних методологій розробки програмного забезпечення, що дозволило забезпечити високу стабільність та масштабованість системи.

Наукова новизна даного дослідження, попри його практичну спрямованість, полягає в удосконаленні підходів до інтеграції модулів швидкого реагування на клієнтські запити та розробці адаптивного інтерфейсу керування контентом у межах єдиної програмної платформи. Новим для автора є застосування комплексного стека технологій для вирішення конкретної прикладної задачі автоматизації туристичного бізнесу.

Практична значущість результатів дослідження полягає у можливості безпосереднього впровадження створеного вебдодатка в реальний сектор туристичних послуг. Розроблене рішення дозволяє менеджерам без спеціальної технічної підготовки гнучко керувати пропозиціями, оперативно реагувати на звернення та знижувати операційні витрати компанії. Крім того, матеріали та архітектурні рішення роботи можуть бути використані як база для подальших розробок у сфері автоматизації малих та середніх підприємств сервісної галузі.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ВЕБ-ДОДАТКІВ ТУРИСТИЧНОЇ ФІРМИ

1.1 Стан та тенденції цифровізації туристичного бізнесу

На сучасному етапі розвитку туристичного сектору спостерігається інтенсивна диджиталізація, яка радикально змінює підходи до клієнтського сервісу та внутрішнього менеджменту. Впровадження інноваційних веб-рішень та автоматизованих комплексів стає критичним фактором виживання на ринку, оскільки вони гарантують оперативність обробки даних та безшовну інтеграцію зовнішніх сервісів. Відтак, ґрунтовне дослідження актуальних тенденцій цифрового ринку є фундаментальним етапом, що передує проєктуванню та розробці якісного програмного продукту для туристичної індустрії [5].

Фундаментом таких змін стає людський капітал та інноваційне мислення, адже будь-яка цифрова система починається з творчого задуму. Як зазначає Натан Мірволд, генеральний директор Intellectual Ventures: “Ми живемо в суспільстві, де технології є дуже важливою частиною бізнесу, нашого повсякденного життя. І всі технології починаються з іскор в чиїйсь голові. Ідея чогось, чого раніше не існувало, але одного разу буде винайдено, може змінити все” [6]. В контексті туризму такою «іскрою» стає перехід від складних паперових архівацій до гнучких веб-платформ, що здатні кардинально змінити досвід взаємодії клієнта з агенцією. Проте, важливо пам’ятати, що попри всю складність алгоритмів, головною цінністю залишається користувач. Це підтверджує думка Роберта Вейда: «Технології, насправді, про людей, а не про апаратне або програмне забезпечення» [7].

Сучасний стан туристичного ринку характеризується глобальним переходом до моделі «Digital First», де цифрові технології стають основним каналом взаємодії між постачальником послуг та споживачем. Традиційні методи продажу турів поступово витісняються онлайн-платформами (OTA –

Online Travel Agencies), такими як Booking.com або Expedia, що задають високі стандарти швидкості та зручності. Така трансформація зумовлена потребою сучасного мандрівника отримати повний цикл обслуговування – від віртуальної екскурсії готелем до миттєвої онлайн-оплати – без необхідності відвідування фізичного офісу.

Однією з провідних тенденцій є впровадження інтелектуальних систем на основі Big Data (великих даних) та штучного інтелекту. Наприклад, сучасні веб-додатки використовують алгоритми машинного навчання для аналізу попередніх пошукових запитів користувача, пропонуючи персоналізовані турпакети, які максимально відповідають його інтересам. Використання інтерактивних чат-ботів та систем підтримки на базі нейромереж дозволяє компаніям забезпечувати консультації в режимі 24/7, що значно підвищує конверсію відвідувачів сайту у реальних покупців [8].

Також значний вплив на галузь мають хмарні технології (Cloud Computing) та концепція «мобільного офісу». Завдяки перенесенню підсистем адміністрування замовлень у хмару, менеджери туристичних фірм отримують можливість керувати бронюваннями, змінювати статуси оплат та актуалізувати ціни в режимі реального часу з будь-якого пристрою. Це забезпечує гнучкість бізнес-процесів та дозволяє компаніям миттєво реагувати на критичні зміни, такі як скасування рейсів або зміна умов готелів, що є життєво важливим в умовах динамічного ринку. Досліджуючи цей аспект, І. В. Токмакова наголошує, що цифровізація сьогодні виступає як нова парадигма управління підприємствами туристичної індустрії, яка докорінно змінює підходи до організації сервісу [9]. Крім того, ефективність функціонування таких систем безпосередньо залежить від специфіки менеджменту та операційної діяльності конкретної туристичної агенції.

У контексті цих трансформацій особливого значення набуває адаптація внутрішнього менеджменту компаній до нових умов. Як зазначають М. В. Новикова та Ю. Ю. Лола, ефективність впровадження будь-яких інформаційних рішень безпосередньо корелює зі специфікою операційної

діяльності та організаційною структурою конкретної туристичної агенції [10]. Традиційні підходи до управління поступаються місцем гнучким автоматизованим стратегіям. Це підтверджують також Г. Антолик та Н. Кампов, підкреслюючи, що сучасні вимоги до роботи менеджера в туристичній фірмі обов'язково включають вміння оперувати цифровими інструментами для координації клієнтських запитів та оптимізації внутрішніх бізнес-процесів [11].

Нарешті, цифровізація сприяє інтеграції систем безпеки та безконтактних технологій. Використання електронних ваучерів, QR-кодів для реєстрації та захищених шлюзів для фінансових транзакцій (таких як Stripe або LiqPay) мінімізує ризики втрати даних та шахрайства. Таким чином, розробка комплексного веб-додатку є не просто технічним завданням, а стратегічним кроком до створення прозорого та надійного цифрового середовища, яке відповідає найвищим очікуванням безпеки та комфорту сучасного клієнта [12].

На думку Грабар Г.А., «завдяки впровадженню цифрових технологій - таких як big data, блокчейн, мобільні застосунки, геоаналітика та онлайн-платформи - туристичні компанії отримують змогу планувати екологічно обґрунтовані маршрути, управляти потоками туристів та підвищувати прозорість у взаємодії з клієнтами. Цифрові інструменти також сприяють інформуванню та вихованню туристів щодо принципів відповідальної поведінки під час подорожей. Наприклад, спеціалізовані мобільні додатки можуть рекомендувати локальні екологічні ініціативи, пропонувати сталі види транспорту або допомагати уникати перевантажених туристичних локацій» [13].

Таким чином, слід зазначити, що цифрова трансформація туристичної галузі зміщує акцент із простої наявності веб-сайту на створення складних екосистем із глибокою автоматизацією внутрішніх процесів. Сучасні тенденції, такі як інтеграція API-інтерфейсів авіаперевізників, автоматична генерація документів та використання прогресивних веб-додатків (PWA), стають галузевим стандартом. Саме тому розробка програмного продукту, що поєднує в собі інтерактивний інтерфейс для клієнта та потужну підсистему адміністрування «Back-office», є актуальною відповіддю на виклики ринку. Це

дозволяє не лише оптимізувати операційні витрати фірми, а й забезпечити високий рівень сервісу, що є ключовим фактором конкурентоспроможності в умовах сучасної глобалізованої економіки [14].

1.2 Аналіз існуючих інформаційних систем для автоматизації взаємодії клієнтами

Обґрунтований вибір технологічного стеку та архітектури майбутнього веб-додатку неможливий без ретельного дослідження вже існуючих на ринку програмних продуктів. На сьогодні сфера туристичних послуг насичена різноманітними інструментами автоматизації – від універсальних систем управління взаємовідносинами з клієнтами (CRM) до спеціалізованих хмарних сервісів (SaaS), кожен з яких пропонує власні підходи до інтерактивної взаємодії та обробки замовлень.

Метою даного аналізу є виявлення функціональних можливостей та критичних недоліків популярних рішень, що дозволить визначити вільну нішу та сформулювати вимоги до власної розробки. Оцінка проводиться за критеріями гнучкості налаштувань, вартості впровадження, рівня контролю над даними та зручності адміністрування бізнес-процесів. Розгляд існуючих альтернатив дасть змогу довести доцільність створення індивідуального комплексного веб-додатку, який би максимально відповідав специфічним потребам конкретної туристичної агенції, уникаючи при цьому надлишковості та високих експлуатаційних витрат сторонніх платформ. Сучасний інструментарій автоматизації туристичної діяльності за своїми функціональними та архітектурними особливостями логічно розподілити на три основні категорії.

Першу категорію становлять спеціалізовані (нішеві) галузеві CRM-системи, найвідомішими серед яких є «U-ON.Travel» та «Moisoft». Ці платформи розроблені з урахуванням специфіки туристичного бізнесу та містять готові інструменти для інтеграції з провідними туроператорами, модулі автоматичного заповнення документів за паспортними даними та системи контролю за

замовленнями авіаквитків. Як зазначають Г. Антолик та Н. Кампов, такі рішення суттєво оптимізують операційну діяльність менеджерів [15]. Проте серйозним недоліком нішевих систем є жорстка структура інтерфейсу та закрита архітектура. Будь-яка спроба кастомізації, додавання унікальних інтерактивних елементів або зміни бізнес-логіки під індивідуальні вимоги локального агентства є неможливою або вимагає надмірно дорогих доробок з боку вендора.

До другої категорії належать потужні універсальні CRM-платформи, лідерами серед яких є «Bitrix24» та «Salesforce». Проблематику впровадження та адаптації таких масштабних корпоративних систем детально досліджувала Л. Гордєєва-Герасимова, вказуючи на їхню високу функціональність [16]. Вони мають розвинені інструменти для автоматизації маркетингу, інтеграції з IP-телефонією та наскрізної аналітики. Однак їхнім головним недоліком для малого та середнього туристичного бізнесу є значна надлишковість функціоналу. Інтерфейс систем сильно перевантажений опціями, які ніколи не використовуються локальними агенціями, що призводить до тривалого і дорогого навчання персоналу [17]. Крім того, модель щомісячної абонентської плати за кожного користувача створює постійне фінансове навантаження на підприємство.

Третім поширеним підходом є використання хмарних конструкторів сайтів (SaaS), таких як «Wix» чи «Tilda», або робота виключно через готові агрегатори турів. Конструктори дозволяють швидко та з мінімальними витратами створити візуальну вітрину для залучення клієнтів, що підтверджує у своєму дослідженні Г. О. Корогод [18]. Проте такі платформи мають критичні обмеження у реалізації складної серверної логіки, захищеної автентифікації користувачів та прямої взаємодії з гнучкими базами даних. У свою чергу, використання готових агрегаторів позбавляє туристичну компанію брендової ідентичності, змушує сплачувати високі комісійні винагороди та унеможливлює створення унікальних програм лояльності в Особистому кабінеті користувача.

Узагальнені результати порівняльного аналізу зазначених категорій програмного забезпечення та індивідуальної розробки наведено в таблиці 1.1.

Систематизація даних дозволяє наочно оцінити кожне рішення за ключовими техніко-економічними параметрами, що безпосередньо впливають на життєздатність програмного продукту. Таке комплексне зіставлення допомагає визначити оптимальний баланс між початковими інвестиціями у розробку та подальшими витратами на масштабування і підтримку системи.

Таблиця 1.1

Порівняльна характеристика засобів автоматизації туристичної діяльності

Критерій порівняння	Готові CRM-системи	Конструктори (SaaS)	Власний веб-додаток
Функціональність	Надлишкова	Обмежена шаблонами	Повна (під запит)
Можливість кастомізації	Низька (жорстка структура)	Середня	Висока
Вартість володіння	Щомісячна абонентплата	Орендна плата	Одноразова розробка
Контроль над даними	На боці провайдера	На боці платформи	Повний контроль фірми
Інтеграція з БД	Складна настройка	Мінімальна	Пряма взаємодія

Примітка. Джерело: розроблено автором

Результати проведеного порівняльного аналізу, представлені в табл. 1.1, показують, що для досягнення оптимального балансу між функціональністю та витратами найдоцільнішим є розробка індивідуального веб-додатку. Власне рішення дозволяє гнучко налаштувати підсистему адміністрування під конкретні бізнес-процеси компанії, впровадити унікальні інтерактивні елементи та забезпечити повний контроль над безпекою даних. Це створює фундамент для незалежності від сторонніх сервісів і дає можливість масштабувати систему відповідно до зростання потреб бізнесу без додаткових витрат на оновлення підписок.

З огляду на результати проведеного аналізу, для реалізації даного проєкту було обрано стек технологій, що дозволяє поєднати гнучкість індивідуальної розробки з високою швидкістю обробки даних. Використання мови програмування Python та фреймворку Flask на стороні сервера забезпечить надійне адміністрування бази даних та безпеку транзакцій. Для створення інтерактивного та адаптивного інтерфейсу користувача доцільним є використання бібліотеки React, що дозволить реалізувати динамічне оновлення інформації без перезавантаження сторінок. Такий підхід забезпечить створення продукту, який не лише конкуруватиме з існуючими CRM-системами за функціональністю, а й переважатиме їх у зручності використання для специфічних задач конкретної туристичної агенції [19].

1.3 Обґрунтування необхідності розробки комплексної системи з підсистемою адміністрування

Ефективна розробка комплексного веб-додатку вимагає глибокого розуміння внутрішніх механізмів функціонування туристичної агенції та особливостей її взаємодії з контрагентами та клієнтами. Предметною областю дослідження є процес надання туристичних послуг, що охоплює формування актуального каталогу турів, опрацювання вхідних запитів від потенційних мандрівників, бронювання та супровід замовлень до моменту їх завершення. Ключовим завданням на цьому етапі є формалізація бізнес-логіки, яка дозволить перетворити хаотичні операційні процеси на чітку структуру даних майбутньої інформаційної системи.

У межах даного проєкту розглядається діяльність туристичної фірми, що спеціалізується на комплексному обслуговуванні мандрівників. Основними бізнес-процесами, що потребують автоматизації, є ведення бази даних напрямків (країни, готелі, типи відпочинку), реєстрація та ведення профілів користувачів, обробка заявок на бронювання та надання клієнтам миттєвого зворотного зв'язку через особистий кабінет. Впровадження централізованої підсистеми

адміністрування дозволить структурувати роботу менеджера, забезпечуючи зручний контроль за етапами виконання кожного замовлення в режимі реального часу та мінімізуючи ризик втрати інформації.

Для забезпечення стабільного функціонування туристичного веб-додатку було спроектовано архітектуру комплексної системи (рис. 1.1), яка базується на модульній взаємодії декількох ключових підсистем.

Центральне місце в цій структурі посідає підсистема адміністрування, яка виступає фундаментом для управління всіма іншими модулями. Зокрема, вона забезпечує:

- керування користувачами: контроль рівнів доступу (менеджер/клієнт);
- налаштування системи: конфігурація параметрів відображення турів та цін;
- моніторинг та звіти: відстеження активності бронювань у реальному часі;
- резервне копіювання та захист: гарантування цілісності даних про клієнтів та фінансові транзакції.

Інформаційна та операційна підсистеми відповідають за безпосередню взаємодію з базою даних турів, тоді як аналітичний модуль дозволяє оцінювати ефективність продажів. Така архітектура дозволяє реалізувати повноцінний «Back-office», що є критично важливим для автоматизації бізнес-процесів сучасної туристичної компанії. Завдяки такій інтеграції менеджер отримує єдине цифрове середовище для оперативного управління контентом та клієнтськими запитами без залучення розробників. Це дозволяє суттєво скоротити час на обробку замовлень та мінімізувати вплив людського фактору при внесенні змін у базу даних. Крім того, наявність виокремленого аналітичного блоку створює можливості для прогнозування попиту на певні напрямки, що є основою для стратегічного планування діяльності фірми. У підсумку, така багаторівнева структура робить веб-додаток не просто статичним сайтом, а динамічним інструментом для масштабування туристичного бізнесу [20].

Рис. 1.1. Структурна модель комплексної інформаційної системи з інтегрованою підсистемою адміністрування

Основою бізнес-логіки додатка є рольова модель взаємодії, що передбачає два основні сценарії використання. Для клієнта головним процесом є «пошук – вибір – бронювання», де система має забезпечити легкий доступ до інформації та зручну форму зворотного зв'язку. Для адміністратора (менеджера фірми) ключовим процесом є «моніторинг – підтвердження – управління», що включає зміну статусів замовлень (наприклад, «очікує на оплату», «підтверджено», «скасовано») та актуалізацію контенту на сайті. Такий поділ обов'язків дозволяє створити замкнений цикл обслуговування, де кожна дія користувача на фронтенді миттєво відображається в панелі керування на бекенді [21].

Завершальним етапом аналізу предметної області є визначення життєвого циклу замовлення в системі. З моменту, коли користувач натискає кнопку «Забронювати», створюється запис у базі даних із унікальним ідентифікатором, який пов’язує конкретного клієнта з обраним туром. Надалі система повинна автоматично оновлювати стан цього запису залежно від дій менеджера або проведення оплати. Розуміння цієї логіки є критично важливим для наступного етапу проектування – розробки схеми бази даних та алгоритмів взаємодії компонентів веб-додатку, що дозволить уникнути дублювання даних та забезпечити високу швидкість відгуку системи.

1.4 Формування вимог до функціональності та інтерактивних можливостей веб-додатку

На основі проведеного дослідження предметної області та аналізу бізнес-логіки туристичної агенції, наступним критично важливим етапом є детермінація технічних та функціональних вимог до майбутнього програмного продукту. Формування чіткого переліку вимог дозволяє трансформувати бізнес-потреби компанії у конкретні завдання для розробника, що є запорукою створення ефективної та відмовостійкої системи. У межах даного проєкту вимоги класифікуються на функціональні, які описують безпосередні можливості системи для різних типів користувачів, та нефункціональні, що визначають якісні характеристики додатка, такі як продуктивність, безпека та зручність інтерфейсу.

Особливий акцент при проектуванні «LikiaTours» ставиться на забезпеченні високого рівня інтерактивності, що є визначальним фактором для сучасних веб-орієнтованих систем. Це передбачає не лише наявність статичних сторінок з інформацією, а й створення динамічного середовища, яке миттєво реагує на дії користувача та забезпечує безперервний цикл обміну даними між клієнтським інтерфейсом та підсистемою адміністрування. Нижче наведено деталізований опис вимог, розподілений за функціональними блоками системи.

Головною вимогою до клієнтської частини веб-додатку «LikiaTours» є забезпечення максимально простого та безперешкодного доступу до повної інформації про доступні туристичні продукти. Користувацький інтерфейс має бути спроектований таким чином, щоб мінімізувати шлях від ознайомлення з пропозицією до моменту успішного бронювання. Для реалізації цієї мети система повинна підтримувати складну багаторівневу фільтрацію, що дозволяє сортувати доступні тури за широким спектром критеріїв: географічним напрямком (країна, місто, регіон), ціновим діапазоном, зірковістю готелів, тривалістю перебування та типом харчування. Висока швидкість обробки таких

запитів на стороні клієнта є критичною для утримання уваги користувача та забезпечення комфортного перегляду каталогу.

Важливим складником функціональності є механізм безпечної реєстрації та авторизації користувачів. Система має використовувати захищені протоколи передачі даних та сучасні алгоритми шифрування персональної інформації. Процес створення облікового запису повинен бути лаконічним, вимагаючи лише необхідні дані для ідентифікації клієнта та подальшої комунікації. Після авторизації користувач отримує доступ до розширених можливостей системи, зокрема до системи збережених турів (список «бажаного») та персоналізованих пропозицій, що базуються на його попередніх запитах.

Окрему увагу в структурі інтерфейсу приділено реалізації «Особистого кабінету» користувача. Цей модуль має стати центральним хабом для взаємодії клієнта з компанією, де акумулюється вся історія попередніх замовлень, детальні описи заброньованих турів та фінансові документи (рахунки, підтвердження оплати). Ключовою вимогою до кабінету є відображення актуального статусу поточного бронювання в режимі реального часу: від моменту подання заявки («Опрацьовується менеджером») до підтвердження («Бронювання підтверджено») та завершення подорожі. Це забезпечує прозорість процесу надання послуг і значно підвищує рівень довіри користувача до веб-ресурсу.

Інтерфейс оформлення заявки на тур повинен бути реалізований як покроковий майстер (wizard), що веде користувача через етапи вибору параметрів подорожі, введення даних пасажирів та вибору способу оплати. Функціонал має передбачати валідацію введених даних «на льоту», щоб запобігти помилкам при заповненні критично важливої інформації. Крім того, клієнтський інтерфейс повинен включати інтерактивні елементи зворотного зв’язку, такі як форми для залишення відгуків після завершення подорожі та систему онлайн-консультацій, що дозволяє отримати швидку відповідь на запитання без переходу до інших каналів зв’язку. Такий комплексний підхід до функціональності гарантує створення сучасного, клієнтоорієнтованого

середовища, що відповідає високим стандартам сучасної туристичної індустрії [21, 22].

Підсистема адміністрування має надавати менеджеру або адміністратору туристичної агенції повний набір інструментів для комплексного керування контентом та операційною діяльністю. Основною вимогою є реалізація повного циклу CRUD-операцій (Create, Read, Update, Delete) над базою даних туристичних продуктів. Це передбачає можливість оперативного додавання нових турів із завантаженням медіа-контенту, детальним описом маршрутів, готелів та послуг, що включені у вартість. Гнучкість цієї системи повинна дозволяти миттєво редагувати цінову політику, враховуючи сезонні знижки або зміну вартості послуг контрагентів, що є критично важливим для підтримки актуальності пропозицій на ринку.

Важливим компонентом підсистеми є реалізація панелі моніторингу (Dashboard), яка слугує центральним вузлом контролю за всіма бізнес-процесами. Вимогою до цієї панелі є можливість візуалізації поточного стану всіх заявок та зміна статусів замовлень «в один клік». Це дозволяє менеджеру ефективно супроводжувати клієнта на кожному етапі – від отримання первинного запиту до фінального підтвердження та архівування виконаної послуги. Автоматизація зміни статусів має супроводжуватися логуванням дій адміністратора, що дозволяє уникнути помилок та забезпечує прозорість внутрішнього документообігу фірми.

Функціонал управління користувачами передбачає не лише перегляд бази даних клієнтів, а й можливість модерації їхніх облікових записів, перегляд історії взаємодії та призначення індивідуальних привілеїв. Крім того, підсистема адміністрування має включати блоки для генерації звітів про продажі та завантаженість напрямків, що дозволяє керівництву приймати обґрунтовані управлінські рішення. Доступ до цього модуля має бути суворо обмежений багаторівневою системою авторизації та автентифікації. Це гарантує, що конфіденційна комерційна інформація та персональні дані клієнтів

залишатимуться захищеними від несанкціонованого доступу або випадкового витоку даних.

Для підвищення конверсії веб-ресурсу та створення позитивного користувацького досвіду, додаток повинен реалізовувати сценарії миттєвого інтерактивного відгуку на будь-які дії відвідувача. Однією з ключових вимог є впровадження динамічної перевірки валідності даних у формах реєстрації та бронювання. Замість очікування відповіді від сервера після відправки форми, система має сигналізувати про помилки (наприклад, некоректний формат пошти або пропущене обов'язкове поле) у реальному часі за допомогою анімацій або кольорових індикаторів. Це значно знижує когнітивне навантаження на користувача та пришвидшує процес взаємодії з інтерфейсом.

Інтерактивність також має проявлятися через систему автоматичних сповіщень. Після кожної значущої події в системі – будь то реєстрація, успішне бронювання або зміна статусу туру менеджером – користувач повинен отримувати миттєвий зворотний зв'язок. Це реалізується через інтеграцію з сервісами Email-розсилок або внутрішні push-сповіщення в особистому кабінеті. Такі механізми підтверджують успішність дій клієнта та підтримують відчуття контролю над ситуацією (рис.1.2), що є надзвичайно важливим у сфері фінансових транзакцій та планування подорожей.

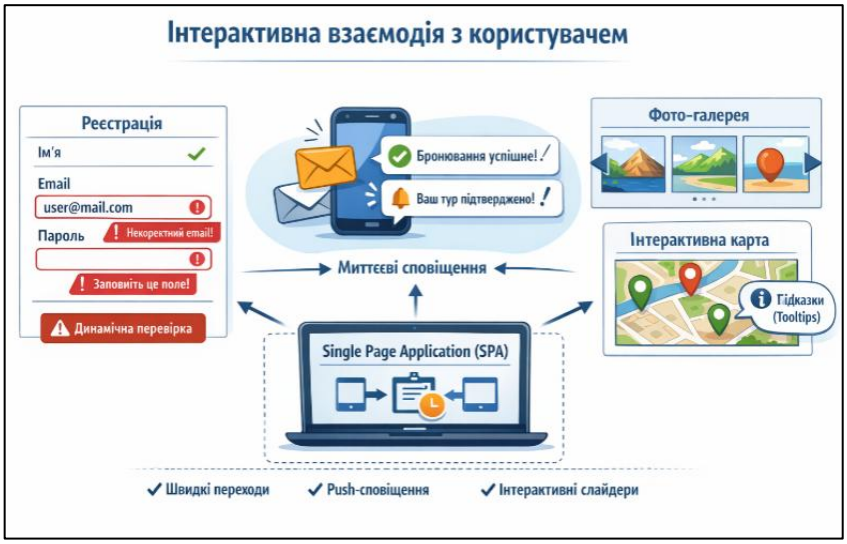

Рис. 1.2 Схема інтерактивної взаємодії користувача з елементами веб-додатку

Окрему увагу слід приділити впровадженню інтерактивних графічних елементів, що спрощують процес вибору туру. Вимогою є використання динамічних слайдерів для перегляду галерей, інтерактивних карт із позначенням локацій готелів, а також спливаючих підказок (tooltips), які роз’яснюють складні терміни або умови надання послуг. Вся взаємодія повинна будуватися на засадах архітектури Single Page Application (SPA), що мінімізує кількість повних перезавантажень сторінок. Це забезпечує плавність переходів, схожу на роботу нативного мобільного додатка, та дозволяє зберігати стан інтерфейсу (наприклад, позицію прокрутки або відкриті фільтри) протягом всієї сесії користувача.

Нефункціональні вимоги до нашого веб-додатку визначають експлуатаційні характеристики системи, які безпосередньо впливають на її стабільність, захищеність та життєздатність у довгостроковій перспективі. Однією з пріоритетних вимог є висока продуктивність та швидкість відгуку інтерфейсу. Система має відповідати стандартам високої доступності, забезпечуючи час завантаження критично важливих сторінок (головна сторінка, результати пошуку турів) не більше ніж 2 секунди при середньому навантаженні. Для досягнення таких показників архітектура повинна передбачати оптимізацію запитів до бази даних, кешування статичних ресурсів та мінімізацію обсягу передаваних даних між сервером і клієнтом. Висока швидкість роботи є не лише технічним параметром, а й чинником утримання користувачів, оскільки затримки у роботі туристичних порталів часто призводять до відмови від бронювання.

Безпека даних клієнтів та конфіденційність комерційної інформації є фундаментальною вимогою, враховуючи роботу з персональними даними та фінансовими транзакціями. Веб-додаток має гарантувати захист від найбільш розповсюджених типів веб-вразливостей, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF). На рівні серверної логіки обов'язковим є використання сучасних методів шифрування та хешування

паролів (наприклад, за допомогою алгоритму bcrypt) замість збереження їх у відкритому вигляді. Крім того, взаємодія між клієнтом і сервером повинна відбуватися виключно через захищений протокол HTTPS з використанням SSL/TLS сертифікатів, що унеможливлює перехоплення даних зловмисниками під час їх передачі [23].

Вимога до масштабованості архітектури передбачає здатність системи до зростання без необхідності докорінної зміни її ядра. Програмний код має бути спроектований за модульним принципом, що дозволить у майбутньому легко інтегрувати нові функціональні блоки, такі як додаткові платіжні шлюзи, модулі для автоматичної генерації авіаквитків або складні аналітичні інструменти на основі штучного інтелекту. Це досягається шляхом чіткого розділення зон відповідальності (Separation of Concerns) та використання API-орієнтованого підходу. Масштабованість також стосується здатності бази даних обробляти зростаючу кількість записів про тури та клієнтів без суттєвої втрати швидкодії, що забезпечується правильною індексацією та структуризацією таблиць.

Окремим аспектом нефункціональних вимог є відмовостійкість та надійність збереження даних. Система повинна передбачати механізми регулярного створення резервних копій бази даних, щоб у разі технічного збою була можливість оперативного відновлення працездатності сервісу з мінімальними втратами інформації. Крім того, програмне забезпечення має бути кросбраузерним та стабільно працювати у різних середовищах виконання, забезпечуючи ідентичний рівень функціональності для всіх категорій користувачів незалежно від їхнього програмного забезпечення. Такий комплексний підхід до нефункціональних вимог гарантує, що розроблений веб-додаток буде не лише зручним, а й надійним інструментом для ведення реального бізнесу.

Дизайн веб-додатку «LikiaTours» повинен бути повністю адаптивним (Responsive Web Design), що є обов'язковою умовою для сучасних сервісів. Це забезпечує коректне та візуально привабливе відображення контенту на широкому спектрі пристроїв – від широкоформатних моніторів настільних

комп'ютерів до екранів смартфонів та планшетів з різною роздільною здатністю. Адаптація не повинна обмежуватися лише масштабуванням елементів; вона має передбачати зміну логіки взаємодії, наприклад, трансформацію горизонтальних навігаційних меню у компактні «бургер-меню» для мобільних версій, а також оптимізацію розмірів кнопок та полів введення для зручного керування за допомогою сенсорних екранів.

Основною вимогою до проектування користувацького досвіду (UX) є суворе дотримання принципу «трьох кліків». Згідно з цим правилом, архітектура навігації має бути вибудована таким чином, щоб будь-яка стратегічно важлива інформація – чи то детальний опис туру, чи то форма миттєвого бронювання – була доступна користувачеві за мінімальну кількість свідомих дій. Це досягається шляхом використання логічної ієрархії сторінок, наскрізного пошуку та інтуїтивно зрозумілих навігаційних ланцюжків («хлібних крихт»). Зменшення когнітивного навантаження на відвідувача дозволяє утримувати його увагу на головній меті візиту, не відволікаючи на пошук потрібних розділів у заплутаній структурі сайту.

Візуальний стиль додатка (UI) повинен бути лаконічним, професійним та максимально зосередженим на контенті. Оскільки туристична галузь базується на візуальному сприйнятті емоцій, основний акцент у дизайні слід зробити на високоякісних фотографіях напрямків та готелів. Колірна палітра інтерфейсу має бути стриманою, щоб не відволікати від зображень, а типографіка – чіткою та читабельною на будь-якому фоні. Використання достатнього обсягу «білого простору» (negative space) дозволить структурувати інформацію, роблячи описи турів та цінові пропозиції легкими для сприйняття навіть при швидкому перегляді сторінки [12].

Додатковою вимогою до інтерфейсу є забезпечення інклюзивності та високого рівня доступності (Accessibility). Веб-додаток має підтримувати високий контраст між текстом та фоном, а також коректно оброблятися програмами зчитування екрана, що важливо для користувачів із певними обмеженнями зору. Окрім візуальної складової, інтерфейс повинен надавати

чіткий зворотний зв’язок на дії користувача: зміна кольору кнопки при наведенні курсору, індикатори завантаження під час очікування відповіді від сервера та зрозумілі повідомлення про успішність виконання операцій. Такий комплексний підхід до UI/UX проектування дозволить створити продукт, який буде не лише функціональним, а й викликатиме позитивний емоційний відгук у клієнтів, стимулюючи їх до повторного використання сервісу.

Висновки розділу 1

У результаті проведеного дослідження теоретичних засад та детального аналізу предметної області було сформовано науково-практичну базу для подальшої розробки проєкту. Вивчення стану та сучасних тенденцій цифровізації показало, що ринок вимагає високої швидкості обробки запитів та персоналізованого підходу, а ключовими факторами успіху є використання хмарних сервісів та інтерактивних вебплатформ. Проведений порівняльний аналіз існуючих рішень для автоматизації взаємодії з клієнтами продемонстрував, що готові комерційні системи часто мають надлишковий функціонал, високу вартість або ж складно адаптуються під специфічні бізнес-процеси. Це дозволило обґрунтувати необхідність розробки власної комплексної системи, де наявність спеціалізованої підсистеми адміністрування забезпечить гнучке керування контентом, контроль замовлень та якісну обробку запитів користувачів. На основі проведеного аналізу було також детально сформовано вимоги до функціональності та інтерактивних можливостей майбутнього вебдодатка, визначено перелік ключових модулів та критеріїв до інтерфейсу. Таким чином, отримані результати аналізу підтверджують актуальність розробки та є основою для подальшого проєктування архітектури системи й бази даних.

РОЗДІЛ 2
ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛЕЙ СИСТЕМИ ВЕБ-ДОДАТКУ ТУРИСТИЧНОЇ ФІМИ

2.1 Моделювання бізнес-процесів обробки замовлень

У процесі проєктування сучасних інформаційних систем критично важливим етапом є формалізація та структурування логіки їх функціонування. Для вебдодатка LikiaTours ключовим аспектом життєвого циклу є саме процедура оформлення та супроводження клієнтських запитів. Коректна побудова алгоритмів на цьому етапі дозволяє мінімізувати помилки під час розробки програмного коду та забезпечує високу швидкість обробки даних на сервері.

Логічна модель бізнес-процесів нашого веб-додатку розроблена для візуалізації взаємодії користувачів із системою бронювання турів. Основною метою моделювання є чітке визначення меж системи, ролей користувачів та ключових функціональних можливостей, які забезпечує додаток. Для високорівневого відображення функціональних вимог до системи було застосовано діаграму прецедентів (UML Use Case Diagram), яка представлена на рис. 2.1.

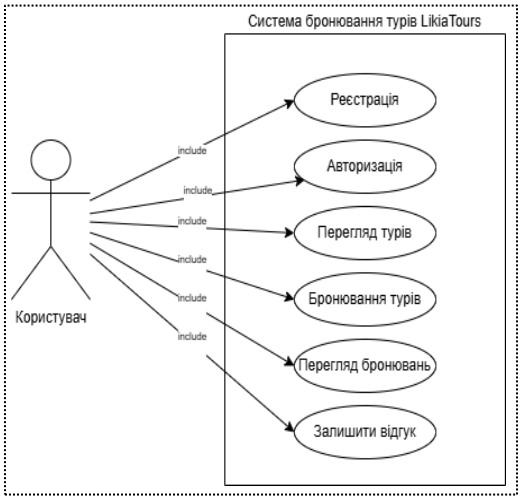

Рис. 2.1. Діаграма прецедентів для веб-додатку

Дана діаграма є інструментом для опису вимог до системи і визначає взаємодію між користувачем та вебдодатком. Наведену діаграму можна інтерпретувати наступним чином:

а) актори (Actors). Основним актором системи є Користувач. Це роль, яка об'єднує всіх відвідувачів сайту, які пройшли авторизацію;

б) прецеденти (Use Cases). Діаграма виділяє шість ключових бізнес-процесів, які доступні авторизованому користувачеві:

- реєстрація та авторизація: базові процеси для отримання доступу до персоналізованого функціоналу;
- перегляд турів: процес ознайомлення з каталогом доступних туристичних пропозицій;
- бронювання турів: ключовий бізнес-процес, який ініціює створення замовлення на тур;

– перегляд бронювань: надання користувачеві доступу до історії його замовлень та їх поточних статусів;

– залишити відгук: механізм зворотного зв'язку, який дозволяє користувачеві оцінити тур після його завершення.

Відносини <<include>> на діаграмі вказують на те, що базові прецеденти (наприклад, «Бронювання турів») обов'язково включають в себе виконання включених прецедентів (у цьому випадку, логіка системи вимагає, щоб користувач був авторизований для здійснення бронювання). Таким чином, діаграма прецедентів фіксує «що» система повинна робити. Для детального розуміння «як» саме протікає ключовий процес — обробка замовлення — необхідно розробити діаграму діяльності [23, 24].

Для деталізації логіки процесу обробки замовлення (бронювання туру) було розроблено модель міжрівневої взаємодії компонентів (рис. 2.2). Модель відображає розподіл обов'язків між клієнтською частиною, сервером та базою даних:

– рівень Користувача (Frontend): відповідає за безпосередню взаємодію з людиною. Він реалізує методи вибору туру, введення даних, підтвердження оплати та отримання фінального статусу від сервера;

– рівень Сервера Flask (Backend): виступає посередником та містить основну бізнес-логіку. Він приймає запити від клієнта (обробитиЗапит()), проводить валідацію введених даних, формує запис для бази даних та повертає результат користувачеві;

– рівень Бази Даних (SQLite): відповідає за збереження та цілісність інформації. На цьому рівні перевіряється наявність вільних місць на обраний тур, здійснюється безпосередній запис броні в таблицю та оновлюється статус самого туру.

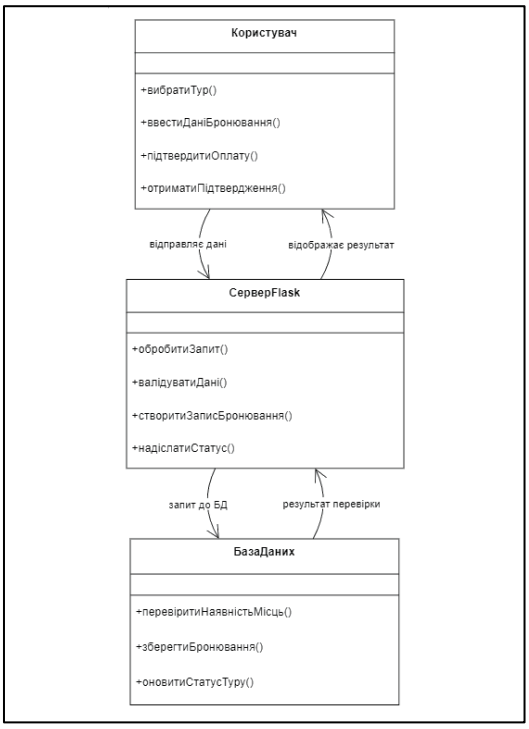

Рис. 2.2. Логічна модель взаємодії рівнів системи при обробці бронювання

Підсумовуючи результати моделювання, можна стверджувати, що побудовані діаграми дозволили детально декомпозувати процес обробки замовлень на окремі атомарні операції. Визначений розподіл обов'язків між клієнтом, сервером Flask та базою даних SQLite закладає надійний фундамент для подальшої технічної реалізації системи. Сформована логічна модель дозволяє перейти до безпосереднього обґрунтування архітектурного стеку та взаємодії компонентів у наступному розділі.

2.2 Проектування архітектури системи та обґрунтування вибору full-stack технологій

Після визначення логічної моделі процесів нашого вебдодатка та ключових прецедентів використання, необхідно спроєктувати архітектуру системи та обґрунтувати вибір технологічного стеку. Правильно обрана архітектура забезпечує масштабованість, швидкість розробки та простоту підтримки

програмного продукту. Для реалізації сервісу LikiaTours було обрано класичну трирівневу архітектуру (клієнт – сервер – база даних), детальна схема якої наведена на рис. 2.4.

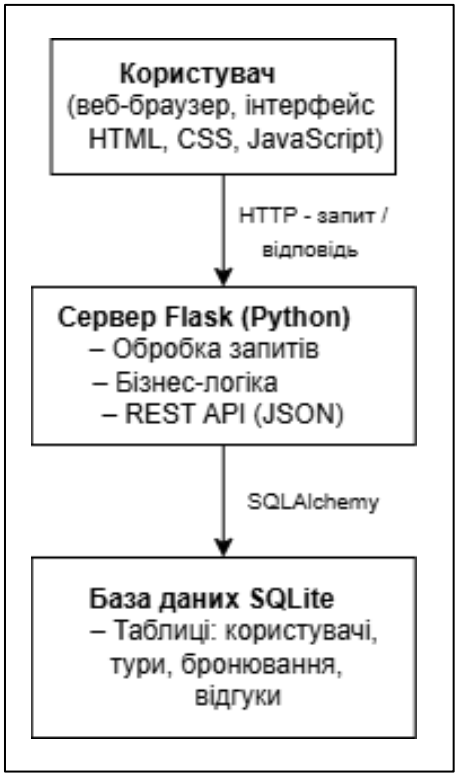

Рис. 2.4 Архітектура веб-додатку LikiaTours

Відповідно до представленої архітектурної схеми, взаємодія в системі відбувається на трьох основних рівнях:

– клієнтський рівень (Frontend) є точкою входу для користувача. Веб браузер завантажує інтерфейс, де реалізована динамічна взаємодія (інтерактивні форми, кнопки, перемикачі). Використання JavaScript та компонентного підходу дозволяє реалізувати концепцію SPA (Single Page Application), що забезпечує швидке завантаження сторінок без повного перезавантаження сайту, покращуючи користувацький досвід (UX);

– серверний рівень (Backend). Flask – це легковагий WSGI-мікрофреймворк, який ідеально підходить для швидкої розробки гнучких вебдодатків. Він не перевантажений зайвим функціоналом «із коробки», що дозволяє розробнику самостійно обирати потрібні бібліотеки. Серверний рівень на базі Flask

відповідає за обробку HTTP-запитів, реалізацію бізнес-логіки (валідація даних, розрахунки) та роботу REST API, яке повертає клієнту дані у форматі JSON;

– рівень збереження даних (Database). Для збереження інформації про користувачів, доступні тури, створені бронювання та відгуки було обрано реляційну базу даних SQLite. Оскільки це локальна вбудована база даних, вона не потребує окремого сервера для адміністрування, що спрощує процес розгортання та тестування на етапі розробки (MVP). Зв'язок між сервером Flask та базою даних реалізовано за допомогою технології SQLAlchemy (Object-Relational Mapping). ORM дозволяє працювати з таблицями БД як зі звичайними об'єктами (класами) у коді Python, що значно прискорює написання запитів та захищає систему від SQL-ін'єкцій.

Для деталізації внутрішньої структури програмного комплексу та інтерфейсів взаємодії, на основі обраного технологічного стеку, було розроблено діаграму компонентів (UML Component Diagram), яка представлена на рис. 2.5. Ця діаграма дозволяє візуалізувати логічні модулі системи як автономні програмні сутності. Такий підхід дозволяє наочно продемонструвати принцип інкапсуляції бізнес-логіки та слабкої пов'язаності (loose coupling) між різними частинами системи. Завдяки чітко визначеним інтерфейсам взаємодії («гніздам» та «кулькам» на схемі), кожен компонент може розроблятися, тестуватися та модернізуватися незалежно від інших. Це суттєво спрощує подальше масштабування веб-додатка та полегшує пошук і усунення потенційних помилок у коді [25, 26].

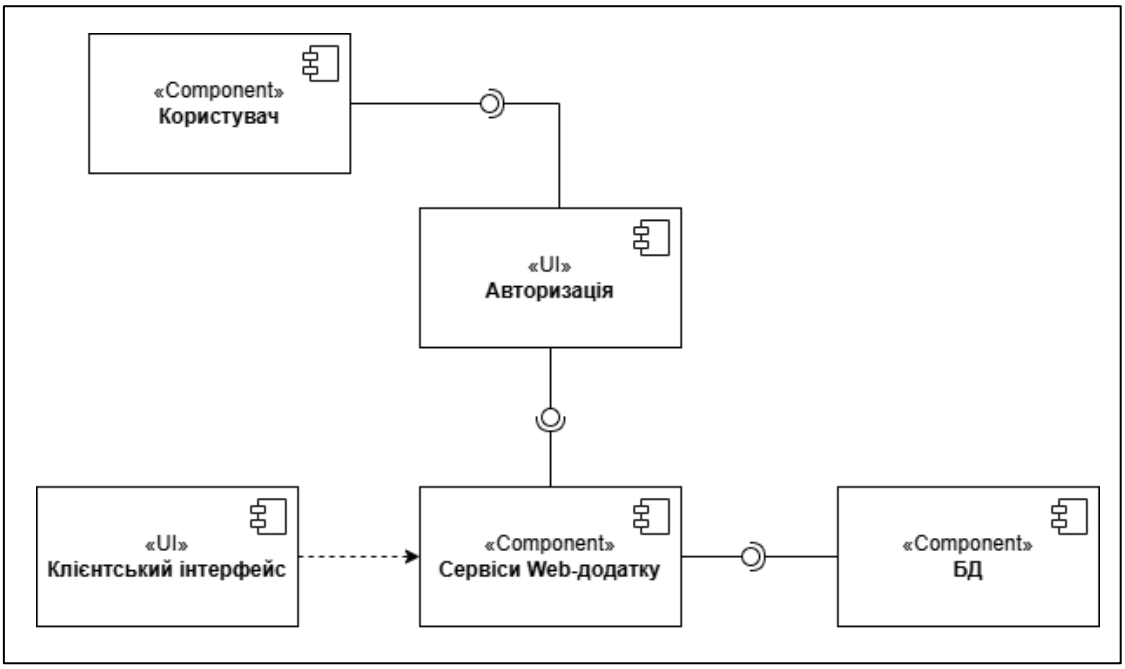

Рис. 2.5. Діаграма компонентів програмного комплексу веб-додатку

Відповідно до представленої діаграми, система складається з трьох ключових логічних компонентів:

– компонент «Користувач»: модуль, який виконується на стороні клієнта. Він реалізує інтерфейси для відображення турів, форм бронювання та відгуків. Компонент взаємодіє з сервером через визначений інтерфейс, використовуючи, наприклад, API. Усередині цього компонента виділено підмодуль «UI» Авторизація, який відповідає за безпеку та доступ до персоналізованих функцій;

– компонент «Сервіси Web-додатку»: центральний модуль системи, який містить всю бізнес-логіку. Він приймає запити від клієнта, проводить валідацію даних, взаємодіє з базою даних та повертає результат. Він також взаємодіє з підмодулем «UI» Авторизація для перевірки прав доступу. Також на діаграмі відображено «UI» Клієнтський інтерфейс, який логічно пов'язаний з основним сервісним компонентом;

– компонент «База даних»: модуль збереження даних, який забезпечує цілісність та персистентність інформації. Він надає інтерфейс для виконання CRUD-операцій (створення, читання, оновлення, видалення) над даними.

Підсумовуючи результати проектування архітектури, можна стверджувати, що поєднання логічного (діаграма компонентів) та технічного (схема архітектури) підходів дозволило створити цілісне та несуперечливе бачення майбутньої системи. Обраний Fullstack-стек на базі Flask та SQLite є оптимальним балансом між швидкістю розгортання прототипу та надійністю збереження даних клієнтів. Спроектована трирівнева архітектура забезпечує чітке розмежування сфер відповідальності: користувацький інтерфейс ізольований від бізнес-логіки, а остання, у свою чергу, не залежить від прямих низькорівневих запитів до сховища. Це закладає надійну основу для наступного етапу проектування – детального опису структури бази даних та забезпечення цілісності інформації.

2.3 Проектування структури бази даних та забезпечення цілісності інформації

Надійність та коректність роботи будь-якого веб-додатка напряму залежить від спроектованої моделі даних. На етапі розробки сервісу LikiaTours було проведено аналіз інформаційних потоків та виділено ключові сутності, необхідні для повноцінного функціонування системи. Для відображення структури бази даних та зв'язків між таблицями було використано діаграму класів (UML Class Diagram), оскільки обрана технологія ORM SQLAlchemy дозволяє автоматично проектувати класи коду Python у реляційні таблиці (рис.2.6). Такий підхід дозволяє абстрагуватися від низькорівневих SQL-запитів і зосередитися на роботі з об'єктами бізнес-логіки безпосередньо в коді бекенду. Це не лише пришвидшує процес розробки та зменшує кількість потенційних синтаксичних помилок, але й забезпечує високу гнучкість при необхідності масштабування структури даних. Крім того, використання ORM виступає додатковим шаром безпеки, автоматично екрануючи дані та захищаючи базу даних від поширених вразливостей.

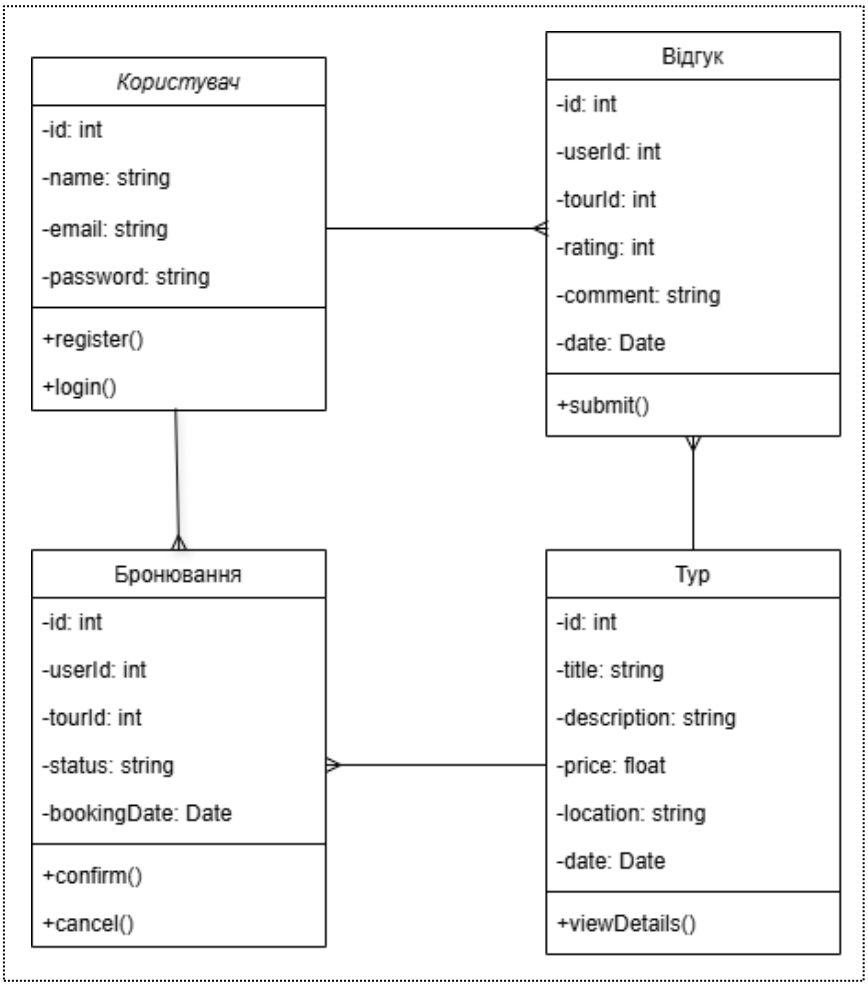

Рис. 2.6 Діаграма класів веб-додатку

Спроектована реляційна схема, відображена на діаграмі, базується на чотирьох фундаментальних сутностях, що мають між собою стійкі зв'язки:

– користувач (User): зберігає облікові дані клієнтів. Сюди входять унікальний ідентифікатор id (Primary Key), ім'я користувача name, електронна пошта email та захешований пароль password;

– тур (Tour): містить інформацію про доступні туристичні путівки. Поля включають id (Primary Key), назву туру title, детальний опис description, вартість price, географічне розташування location та дату проведення date;

– бронювання (Booking): центральна таблиця транзакцій. Вона пов'язує користувача з обраним туром через зовнішні ключі userId та tourId (Foreign Keys). Також містить статус замовлення status та дату створення запису bookingDate;

– відгук (Review): таблиця для збереження зворотного зв'язку. Вона прив'язана до конкретного користувача та конкретного туру, містить оцінку rating та текстовий коментар comment.

Для того, щоб інформація в системі не дублювалася, а між таблицями не виникало «битих» посилань (наприклад, коли видалили тур, а бронь на нього залишилася), у проекті LikiaTours реалізовано наступні правила забезпечення цілісності даних:

– сутнісна цілісність (Entity Integrity): кожна таблиця має свій первинний ключ (id), який гарантує унікальність кожного окремого запису і не може бути порожнім (NOT NULL);

– посилальна цілісність (Referential Integrity): забезпечується через використання зовнішніх ключів (Foreign Keys). Система фізично не дозволить створити відгук чи бронювання на тур, якого не існує в базі даних, або від імені неіснуючого користувача;

– каскадні операції (Cascade Constraints): на рівні ORM SQLAlchemy налаштовано правила видалення. Наприклад, у разі видалення профілю користувача, всі його особисті відгуки та неактивні бронювання автоматично анулюються, щоб не засмічувати базу даних неактуальною інформацією;

– валідація типів даних: кожне поле має жорстко визначений тип даних (int, string, float, Date), що унеможливлює потрапляння некоректних значень (наприклад, тексту в поле вартості туру).

Спроектована реляційна модель повністю покриває бізнес-вимоги до нашого веб-додатка. Використання підходу ORM у поєднанні із суворими обмеженнями цілісності дозволяє гарантувати безпеку даних та високу швидкість виконання пошукових запитів. Це створює надійний бекенд-каркас, що дозволяє перейти до фінального етапу проектування – розробки алгоритмів інтерактивної взаємодії.

2.4 Проектування алгоритмів інтерактивної взаємодії та зворотного зв’язку користувача

Після визначення архітектури та структури збереження даних, необхідно деталізувати динаміку роботи системи. Інтерактивна взаємодія у веб-додатку LikiaTours базується на клієнт-серверній архітектурі, де кожна дія користувача породжує ланцюжок подій між інтерфейсом, контролерами бекенду та базою даних. Для опису цих процесів у часі було застосовано діаграми послідовності (UML Sequence Diagram), які дозволяють наочно продемонструвати обмін повідомленнями між об'єктами системи. Першим критично важливим алгоритмом інтерактивної взаємодії є процес створення нового облікового запису користувача (рис.2.7).

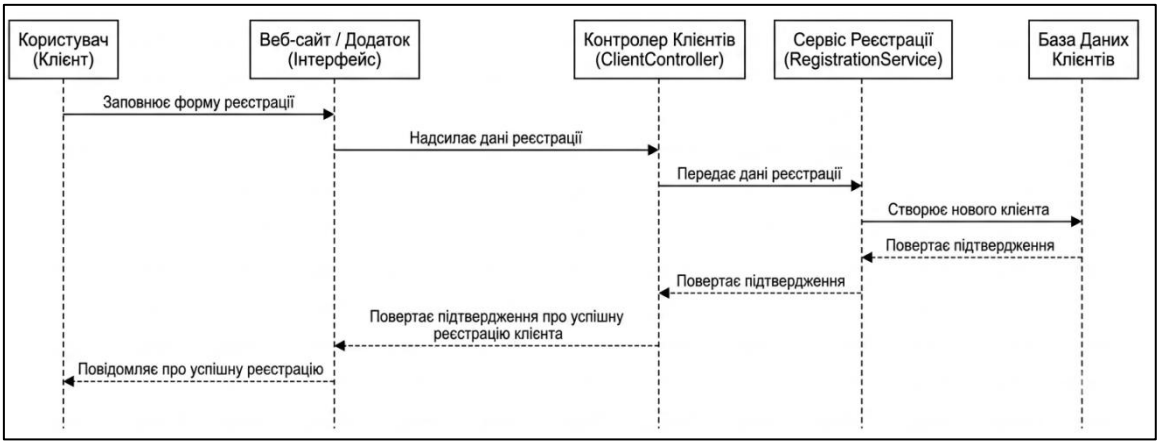

Рис. 2.7 Діаграма послідовності: Реєстрація користувача

Згідно з побудованою моделлю, алгоритм реєстрації складається з наступних послідовних кроків:

– користувач заповнює форму реєстрації на фронтенді (вводить ім'я, email, пароль) та ініціює відправку даних;

– клієнтський інтерфейс транслює ці дані на контролер сервера (UserController);

– контролер передає параметри на рівень сервісів (UserService) для виконання бізнес-логіки (перевірка на унікальність пошти, хешування пароля);

– сервіс формує команду на створення нового об'єкта та надсилає запит до бази даних;

– база даних фіксує новий запис і повертає підтвердження успішної операції назад по ланцюжку аж до інтерфейсу, який інформує користувача про успішну реєстрацію.

Не менш важливим є алгоритм автентифікації користувача для отримання доступу до закритих функцій системи (наприклад, бронювання) (2.8).

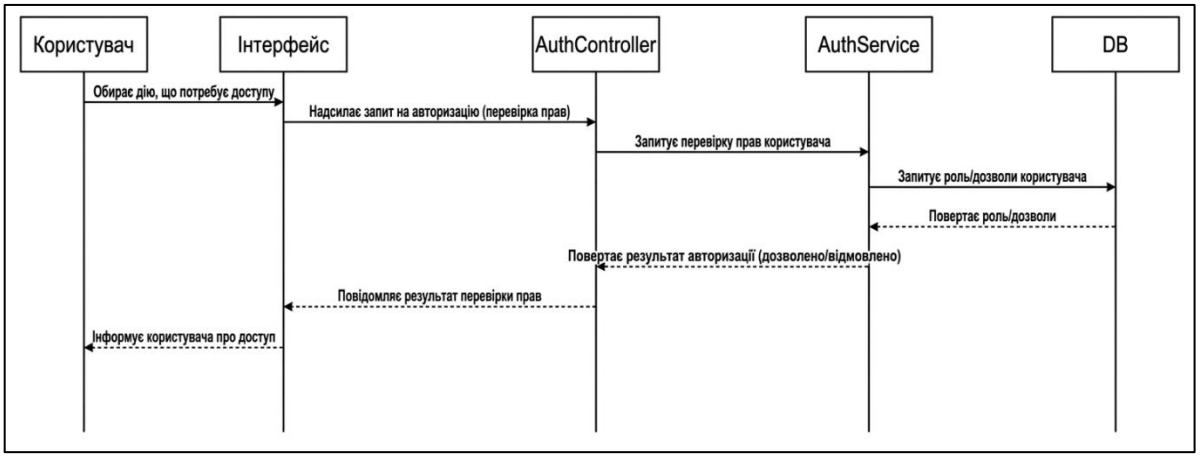

Рис. 2.8 Діаграма послідовності: Авторизація користувача

Алгоритм інтерактивної перевірки облікових даних виконується за такою схемою:

– користувач вводить свій email та пароль у форму входу;

– інтерфейс формує запит на автентифікацію і передає його спеціалізованому контролеру AuthController;

– контролер делегує перевірку сервісу авторизації (AuthService);

– сервіс робить вибірку з бази даних (DB) для перевірки існування користувача з таким email та порівнює хеші паролів;

– результат перевірки повертається назад. Якщо дані правильні – користувач отримує токен сесії (або cookie) і перенаправляється в особистий кабінет, про що його сповіщає інтерфейс.

Описані алгоритми взаємодії демонструють чітке розділення зон відповідальності між компонентами системи. Використання діаграм послідовності дозволило зафіксувати точний порядок виклику методів та передачі параметрів, що значно спрощує подальший етап безпосереднього програмування коду на Python та React. Таким чином, у Розділі 2 було повністю виконано проектування веб-додатка LikiaTours: від абстрактних бізнес-процесів до конкретних алгоритмів та структури бази даних, що дозволяє перейти до практичної реалізації системи.

Висновки до розділу 2

У ході виконання другого розділу було проведено повний цикл проектування архітектури та інформаційних моделей веб-орієнтованої системи для туристичної фірми. Основним етапом роботи стало обґрунтування вибору клієнт-серверної архітектури, яка дозволяє розділити логіку обробки даних на стороні сервера та візуалізацію інтерфейсу на стороні клієнта. Такий підхід забезпечує необхідну гнучкість системи, спрощує її подальше масштабування та дозволяє інтегрувати сучасні безпекові протоколи для захисту даних користувачів. Проектування функціональної структури системи дозволило виділити ключові модулі, серед яких особливе місце посідає інтегрована система керування контентом та підсистема обробки клієнтських запитів. Завдяки розробці діаграм прецедентів та сценаріїв взаємодії було детально описано рольові моделі доступу та алгоритми роботи основних сервісів, що закладає фундамент для створення інтуїтивно зрозумілого інтерфейсу. Побудована логічна схема взаємодії компонентів підтвердила ефективність обраного стеку технологій для реалізації поставлених завдань.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ТУРИСТИЧНОЇ ФІРМИ З ПІДСИСТЕМОЮ ЗАМОВЛЕНЬ ТА ВЗАЄМОДІЇ З КОРИСТУВАЧАМИ

3.1 Реалізація клієнтської частини та інтерфейсу користувача

На основі проведеного аналізу предметної області та розроблених архітектурних рішень було здійснено програмну реалізацію вебзастосунку для туристичної фірми. Основне призначення розробленого програмного продукту полягає в автоматизації обліку бронювань, реєстрації клієнтів та наданні актуальної інформації про тури. Технологічний стек системи базується на поєднанні бібліотеки React для побудови динамічного фронтенду та фреймворку Flask для реалізації серверної логіки.

Вибір React як основного інструменту розробки клієнтської частини обумовлений необхідністю створення гнучкого, компонентно-орієнтованого інтерфейсу з високою продуктивністю [27]. Під час проектування було дотримано сучасних принципів UX/UI-дизайну, що забезпечує інтуїтивно зрозумілу взаємодію користувача з такими функціями, як авторизація, пошук турів, формування заявок на бронювання та подання відгуків.

Для розробки фронтенд-складової було обрано бібліотеку JavaScript – React (рис. 3.1). Дана технологія дозволяє реалізувати концепцію Single Page Application (SPA), де оновлення контенту відбувається динамічно без повного перезавантаження сторінок. Це суттєво підвищує швидкість відгуку інтерфейсу та покращує загальний користувацький досвід.

Компонентний підхід React дозволяє декомпозувати інтерфейс на незалежні блоки (кнопки, форми, навігаційні панелі), що спрощує їх повторне використання та тестування. Важливою перевагою є механізм Virtual DOM, який мінімізує кількість операцій з реальним деревом об’єктів документа, забезпечуючи високу швидкість рендерингу навіть на пристроях з обмеженими ресурсами. Також було забезпечено ефективну інтеграцію з бекенд-частиною на

Flask через REST API, що дозволило чітко розмежувати зони відповідальності між клієнтом та сервером [3].

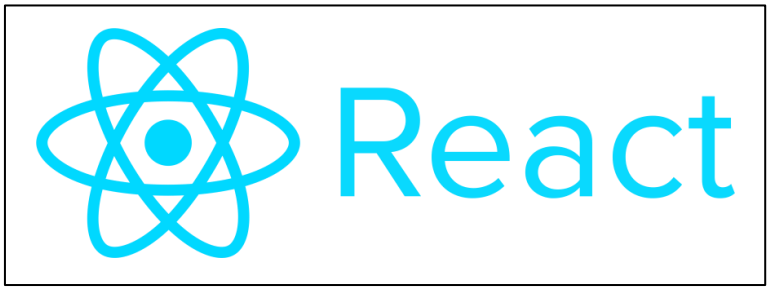

Рис. 3.1 Логотип бібліотеки React

Використання декларативного підходу у розробці інтерфейсу дозволило зосередитися на описі того, як має виглядати кожен стан системи, тоді як React автоматично бере на себе завдання з ефективного оновлення та рендерингу компонентів при зміні даних. Це значно спростило реалізацію динамічних форм, таких як система фільтрації турів за категоріями та ціною, де інтерфейс миттєво реагує на введення користувача [28].

Структура інтерфейсу була спроектована з урахуванням сценаріїв поведінки користувача. Для забезпечення логічної навігації застосунок було розділено на декілька функціональних зон:

– головна сторінка: виконує роль презентаційного майданчика, що містить основні пропозиції та навігаційні точки входу до інших розділів системи (рис. 3.2);

– каталог турів та детальні сторінки: реалізують відображення списку актуальних напрямків з можливістю переходу до детального опису кожної пропозиції, де вказані умови, вартість та візуальні матеріали (рис. 3.3, рис.3.4);

– інформаційні та інтерактивні розділи: включають сторінку FAQ для підтримки користувачів, розділ з контактними даними та модуль відгуків, що сприяє підвищенню лояльності клієнтів.

Завдяки впровадженню адаптивної верстки, інтерфейс зберігає функціональність та візуальну цілісність як на десктопних моніторах, так і на мобільних пристроях.

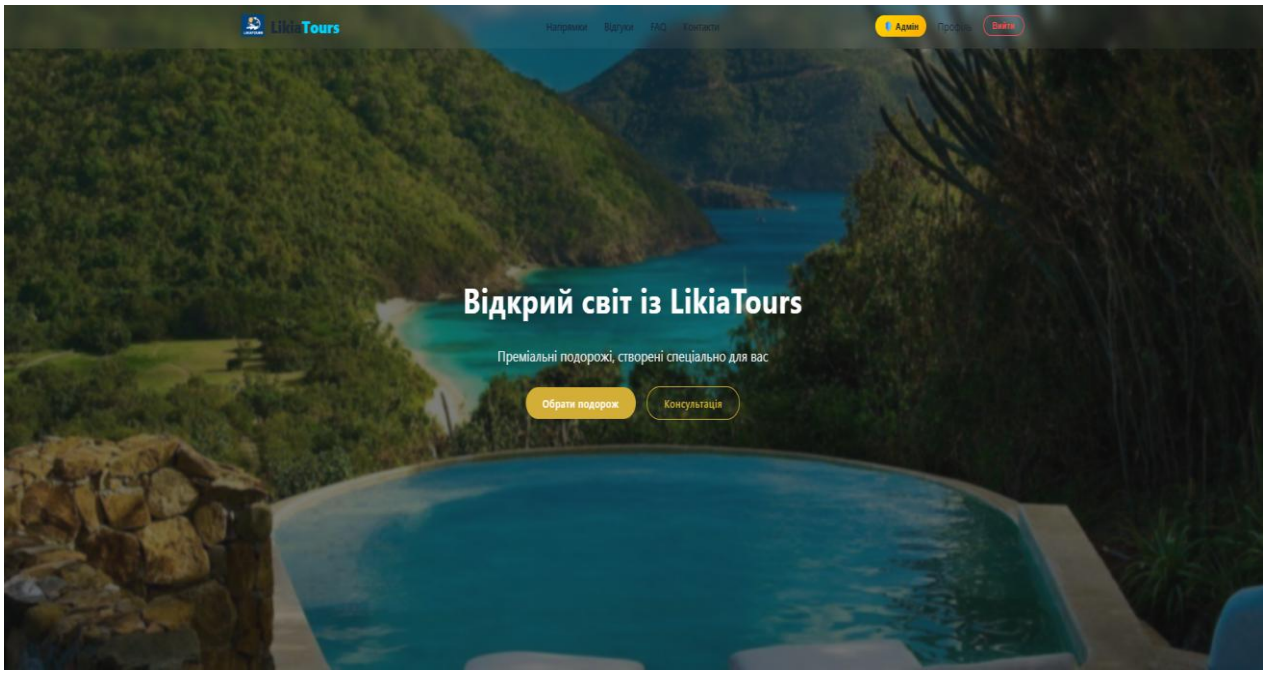

Рис. 3.2 Головна сторінка сайту

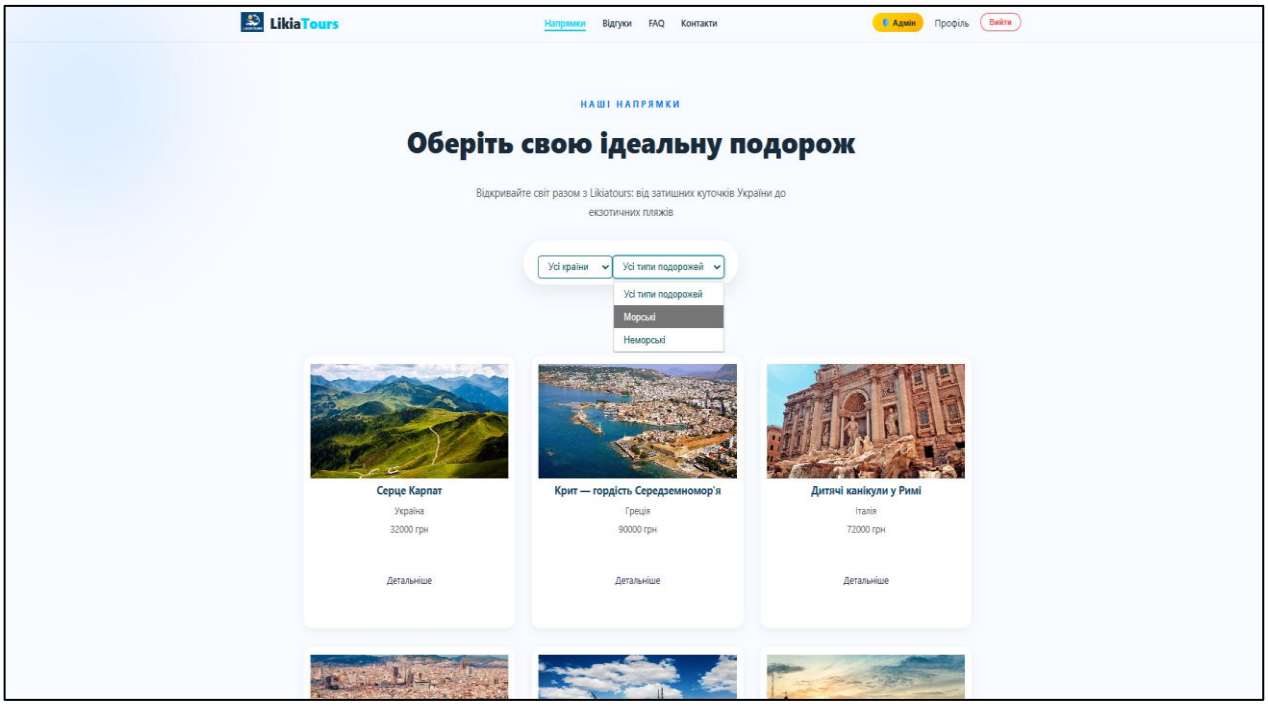

Рис. 3.3 Сторінка каталогу турів

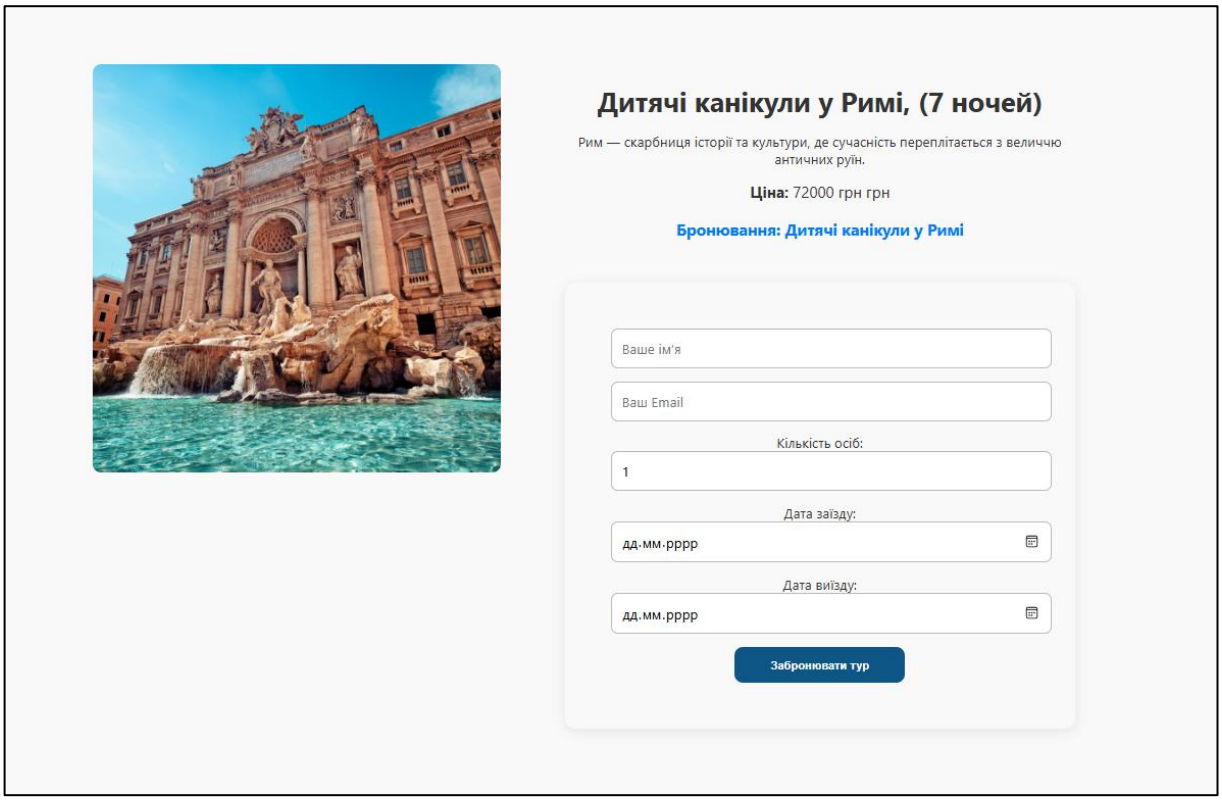

Рис. 3.4 Сторінка детального опису туру

Окремим функціональним блоком системи є особистий кабінет користувача та модулі автентифікації. Реалізація форм реєстрації та входу в систему базується на принципах валідації даних у реальному часі, що дозволяє користувачеві миттєво отримувати підказки щодо коректності заповнення полів. Після успішної авторизації клієнт отримує доступ до розширеного функціоналу, зокрема до перегляду історії власних бронювань та можливості редагування персональних даних. Це забезпечує персоналізований досвід взаємодії та підвищує рівень безпеки інформаційної системи.

Система навігації вебзастосунку була спроектована як дворівнева структура, що складається з глобального меню (Header) та допоміжної панелі (Footer). Глобальне меню забезпечує постійний доступ до ключових розділів сайту незалежно від поточної позиції користувача, що мінімізує кількість переходів для досягнення цілі. Додатково впроваджено механізм «хлібних крихт» або динамічних посилань, які допомагають користувачеві легко

орієнтуватися в ієрархії сторінок під час перегляду конкретних турів чи інформаційних блоків.

Важливою складовою інтерфейсу є інтерактивні елементи зворотного зв'язку та системні сповіщення. Для покращення UX-показників було розроблено систему модальних вікон та спливаючих повідомлень (toasts), які інформують користувача про статус виконання його запитів: успішне бронювання туру, надсилання відгуку або помилку при авторизації. Такий візуальний супровід робить систему передбачуваною та зрозумілою, зменшуючи когнітивне навантаження на клієнта під час роботи з вебзастосунком.

В основу архітектури клієнтської частини покладено ієрархію компонентів. Кожен елемент сторінки є ізольованою логічною одиницею, що володіє власним станом (state) та отримує вхідні дані через властивості (props). Це дозволило реалізувати принцип односпрямованого потоку даних, що робить поведінку інтерфейсу передбачуваною. Наскрізні елементи, такі як Header, Footer та SocialBar, забезпечують єдину стилістику всього проекту. Спеціалізовані компоненти TourCard та BookingForm відповідають за відображення динамічних даних та обробку вводу користувача. Організація файлової структури за компонентним принципом (рис. 3.6) значно спрощує подальше масштабування системи.

Важливою особливістю реалізованої архітектури є розмежування компонентів на «розумні» (контейнери) та «презентаційні». Презентаційні компоненти, такі як TourCard або SocialBar, фокусуються виключно на візуальному відображенні отриманих через props даних і не містять складної бізнес-логіки. У свою чергу, компоненти-контейнери відповідають за взаємодію з API на базі Flask, обробку станів завантаження та передачу результатів запитів дочірнім елементам. Такий підхід значно полегшує тестування окремих частин інтерфейсу та дозволяє швидко змінювати дизайн без втручання в логіку обробки даних [12].

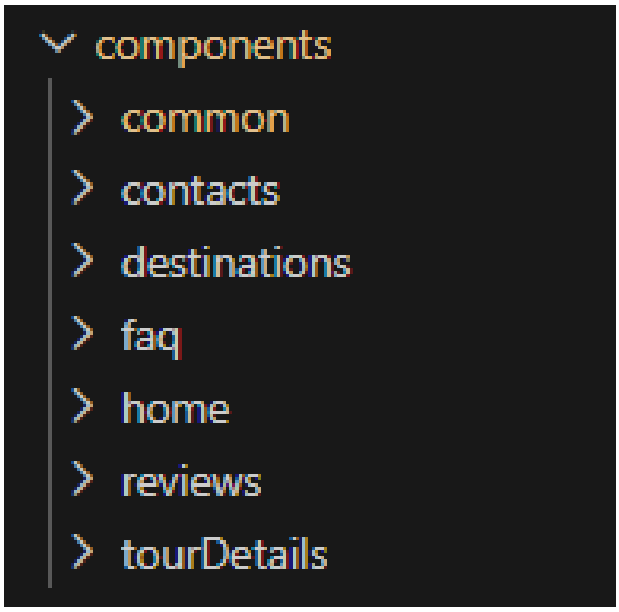

Рис. 3.6 Папка Components

Для забезпечення динамічності інтерфейсу було використано механізм «підняття стану» (lifting state up). Це дозволило синхронізувати роботу декількох компонентів, що знаходяться на одному рівні ієрархії. Наприклад, стан обраного туру в списку TourList може передаватися до батьківського компонента, а потім транслюватися у форму бронювання BookingForm. Це гарантує цілісність відображуваної інформації та виключає конфлікти даних при одночасній взаємодії користувача з різними частинами сторінки.

Навігація всередині SPA реалізована за допомогою бібліотеки React Router. Це дало змогу створити систему динамічних маршрутів, де кожна URL-адреса відповідає певному стану інтерфейсу без перезавантаження сторінки.

Використання параметризованих шляхів (наприклад, /tours/:id) дозволяє створювати універсальні шаблони сторінок, які динамічно завантажують дані про конкретний тур на основі ідентифікатора з адресної стрічки (рис. 3.7). Такий підхід забезпечує зручність копіювання посилань на конкретні пропозиції та покращує індексацію сайту.

```jsx
<Routes>
  <Route path="/" element={<HomePage />} />
  <Route path="/destinations" element={<DestinationsPage />} />
  <Route path="/tour/:id" element={<TourDetailsPage />} />
  <Route path="/faq" element={<FAQPage />} />
  <Route path="/reviews" element={<ReviewsPage />} />
  <Route path="/contacts" element={<ContactsPage />} />
  <Route path="/register" element={<Register />} />
  <Route path="/login" element={<Login />} />
  <Route path="book" element={<Booking />} />
  <Route path="/review-form" element={<ReviewForm />} />
  <Route path="/auth" element={<AuthPage />} />
  <Route
    path="/profile"
    element={
      <PrivateRoute>
        <ProfilePage />
      </PrivateRoute>
    }
  />
</Routes>
```

Рис. 3.7 Реалізація маршрутизації

Важливим аспектом реалізації маршрутизації стало впровадження захищених маршрутів (Private Routes). Це дозволило розмежувати доступ до різних частин застосунку залежно від статусу авторизації користувача. Наприклад, доступ до сторінки особистих бронювань або панелі адміністратора надається лише після успішної перевірки токена доступу. У разі спроби неавторизованого входу система автоматично перенаправляє користувача на сторінку логіна, що значно підвищує рівень захисту персональних даних та запобігає несанкціонованому доступу до функцій керування контентом [29, 30].

Крім того, використання бібліотеки React Router дозволило реалізувати обробку помилок навігації на клієнтському рівні. Було створено спеціальний маршрут-заглушку («Not Found»), який активується у разі введення некоректної URL-адреси. Це забезпечує безперервність користувацького досвіду, оскільки замість стандартної помилки браузера користувач бачить стилізовану сторінку з можливістю швидкого повернення до головного розділу сайту. Такий підхід робить навігацію всередині застосунку більш стабільною та передбачуваною.

Візуальна ідентичність проекту реалізована за допомогою каскадних таблиць стилів (CSS). Для забезпечення консистентності дизайну було

впроваджено використання CSS-змінних (variables.css), що дозволяють централізовано керувати палітрою кольорів та типографікою.

Кожен компонент має індивідуальні стилі, що виключає конфлікти між елементами. Особливу увагу приділено мікровзаємодіям: ефектам наведення (hover), плавності переходів (transitions) та валідації форм. Це створює відчуття "живого" інтерфейсу та надає користувачеві зворотний зв'язок на кожну його дію (рис. 3.8, 3.9). Адаптивність реалізована через гнучкі сітки та медіа-запити, що гарантує коректне відображення блоків на екранах будь-якої роздільної здатності.

```
html,
body,
#root {
  height: 100%;
  margin: 0;
}

.App {
  min-height: 100%;
  display: flex;
  flex-direction: column;
}

main {
  flex: 1;
}

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
```

Рис.3.8 Файл App.css (частина 1)

```css
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

Рис.3.9 Файл App.css (частина 2)

Важливим етапом стилізації стала розробка адаптивної системи відступів та типографіки, що базується на використанні відносних одиниць вимірювання (rem, em, %). Це дозволило досягти високої гнучкості інтерфейсу: при зміні базового розміру шрифту або масштабу екрана всі елементи системи пропорційно підлаштовуються, зберігаючи візуальний баланс. Використання CSS-змінних для управління цими параметрами дозволило створити цілісну дизайн-систему, де зміна одного значення у файлі variables.css автоматично оновлює вигляд всього вебзастосунку, забезпечуючи легкість підтримки коду.

Для реалізації складної структури сторінок, таких як сітка карток турів та макети форм, було використано сучасні модулі Flexbox та CSS Grid. Ці інструменти дозволили створити стійкі до зміни контенту макети, які автоматично перегруповуються залежно від ширини в’юпорту. Наприклад, список турів, що відображається у три колонки на десктопних моніторах, плавно трансформується у вертикальний список на смартфонах, не втрачаючи при

цьому читабельності та зручності натискання інтерактивних елементів. Такий підхід мінімізує необхідність використання великої кількості медіа-запитів, роблячи код більш чистим та зрозумілим.

Особливий акцент було зроблено на доступності та візуальній ієрархії контенту. Шляхом правильного налаштування контрастності кольорів, вибору чітких шрифтових пар та використання тіньових ефектів (box-shadow) вдалося виділити головні елементи взаємодії, такі як кнопки заклику до дії (CTA) та форми бронювання. Крім того, впровадження плавних анімацій при завантаженні контенту та переході між станами компонентів не лише покращує естетичне сприйняття проекту, а й допомагає користувачеві краще зрозуміти логіку роботи системи, створюючи відчуття цілісного та завершеного програмного продукту.

На завершальному етапі стилізації було проведено кросбраузерне тестування, яке підтвердило ідентичність відображення інтерфейсу в усіх популярних сучасних браузерах. Оптимізація CSS-коду, зокрема мінімізація використання застарілих властивостей та перевірка на швидкість завантаження стилів, дозволила досягти високих показників продуктивності клієнтської частини. У результаті поєднання продуманої архітектури стилів та сучасних методів верстки забезпечило створення привабливого, швидкого та функціонального інтерфейсу, що повністю відповідає вимогам до сучасних вебзастосунків у туристичній сфері.

3.2 Розробка серверної логіки та адміністративної підсистеми керування замовленнями

Серверна частина інформаційної системи виступає сполучною ланкою між інтерфейсом користувача та базою даних, забезпечуючи реалізацію ключових бізнес-процесів туристичної фірми. Бізнес-логіка розробленого вебзастосунку базується на мікрофреймворку Flask, що дозволило створити гнучку структуру обробки запитів та забезпечити високу швидкість обміну даними [31, 32].

Основна увага при розробці бекенд-складової була приділена автоматизації циклу обробки замовлень, безпеці авторизації та ефективному управлінню контентом через підсистему Back-office.

Взаємодія між клієнтською частиною на React та сервером реалізована через RESTful API. Сервер приймає HTTP-запити, виконує необхідні обчислення, взаємодіє з базою даних та повертає відповідь у структурованому форматі JSON (рис. 3.10). Такий підхід дозволяє повністю відокремити логіку представлення від логіки обробки даних, що полегшує тестування та подальшу підтримку системи.

```
export async function registerUser(username, password, email) {
  const res = await fetch("http://localhost:5000/register", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, password, email }),
  });
  return await res.json();
}

export async function loginUser(username, password) {
  const res = await fetch("http://localhost:5000/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, password }),
  });
  return res.json();
}

export function logoutUser() {
  localStorage.removeItem("token");
  localStorage.removeItem("user");
  window.location.reload();
}
```

Рис. 3.10 Реалізація API-запитів на стороні клієнта (api.js)

Одним із центральних алгоритмів системи є механізм бронювання турів. При надсиланні запиту на бронювання серверна логіка виконує декілька послідовних операцій:

– перевірка статусу автентифікації користувача;

– валідація вхідних даних (наявність туру, коректність дати та контактної інформації);

– синхронний запис транзакції в базу даних з прив'язкою замовлення до конкретного профілю;

– повернення статусу успішності операції для візуального підтвердження на фронтенді.

Безпека користувацьких даних забезпечується багаторівневою системою реєстрації та авторизації (рис. 3.11, рис.12). При створенні нового облікового запису сервер виконує хешування паролів, що унеможливлює їх зберігання у відкритому вигляді. Процес авторизації побудований на використанні токенів доступу (Access Tokens), які після перевірки облікових даних передаються клієнту та зберігаються в localStorage для підтримки активної сесії.

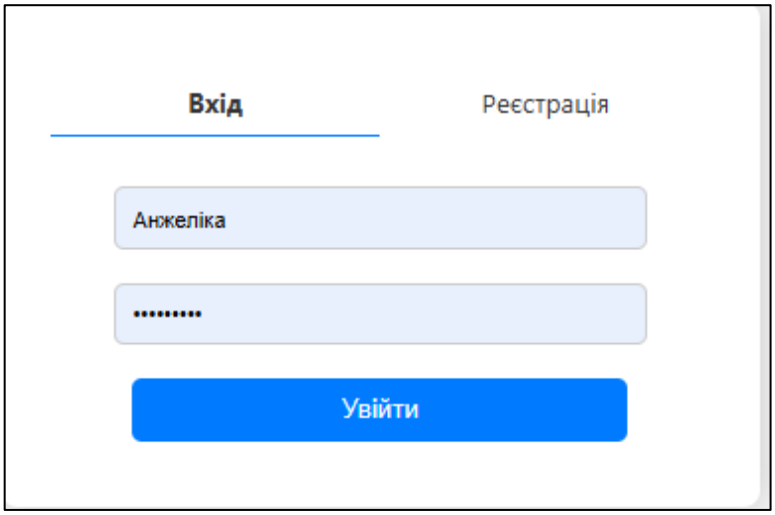

Рис. 3.11 Форма реєстрації та авторизації користувачів

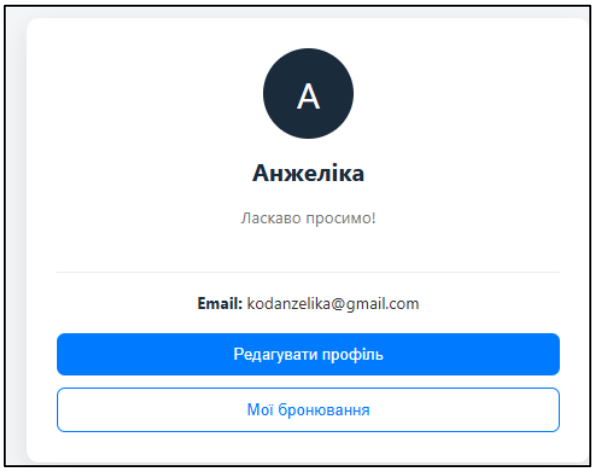

Рис. 3.12 Профіль користувача

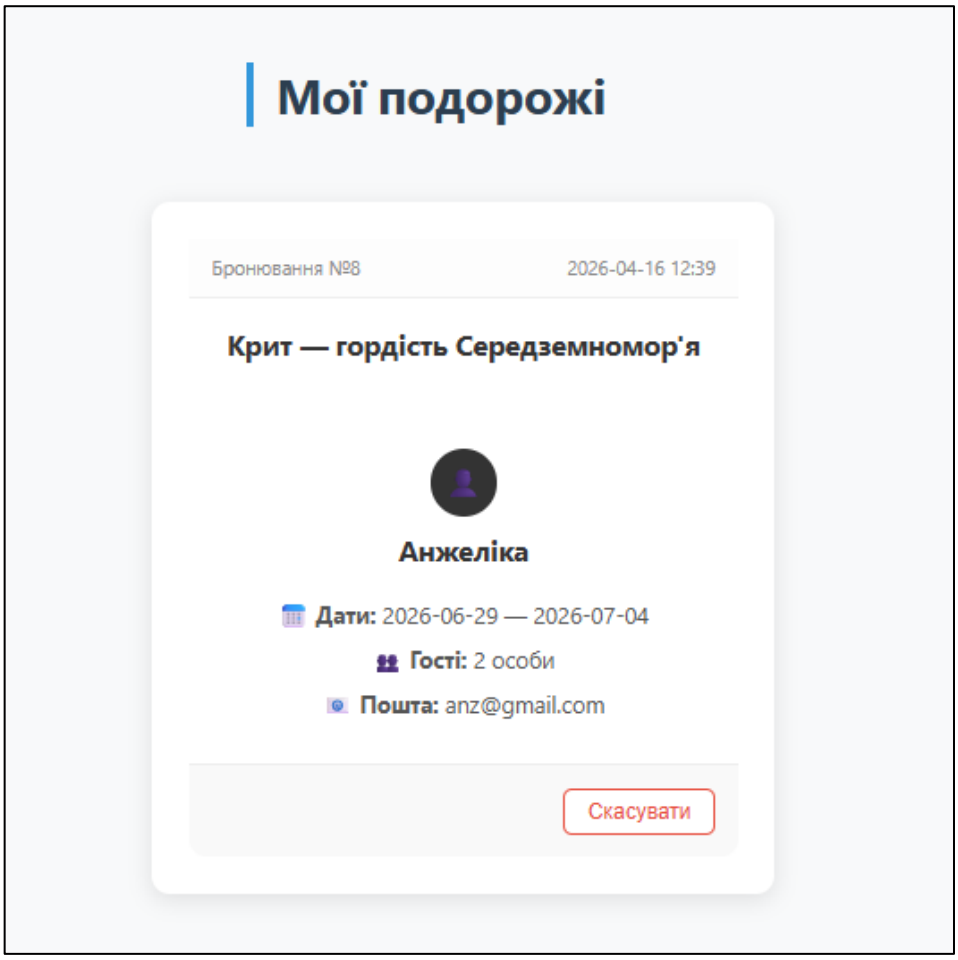

Рис. 3.13 Профіль користувача з історією бронювань

Наявність дійсного токена є обов’язковою умовою для доступу до приватних методів API, таких як перегляд історії замовлень у профілі клієнта (рис. 3.12) або публікація відгуків. Якщо токен відсутній або термін його дії вичерпано, система автоматично блокує доступ до захищених ресурсів, перенаправляючи користувача на сторінку входу. Такий механізм гарантує цілісність даних та захищає систему від несанкціонованого втручання.

Окрему увагу в межах розділу приділено підсистемі керування замовленнями (Back-office), яка призначена для адміністраторів туристичної фірми. Ця частина системи дозволяє здійснювати повний контроль над життєвим циклом туристичного продукту та клієнтськими запитами.Функціонал адмін-панелі включає:

– керування каталогом: додавання нових турів, редагування існуючих описів, оновлення цін та керування галереями зображень;

- моніторинг замовлень: перегляд актуальних заявок на бронювання у реальному часі з можливістю зміни їх статусу (наприклад, "підтверджено", "очікує оплати", "скасовано");
- модерація відгуків: перевірка клієнтських коментарів на відповідність правилам спільноти перед їх публікацією на сайті, що дозволяє підтримувати високу якість контенту.

Логіка Back-office інтегрована з алгоритмами фільтрації та ранжування. Наприклад, при зміні статусу туру на "архівний" у панелі адміністратора, серверна частина автоматично виключає його з результатів пошуку на клієнтській стороні. Це забезпечує актуальність інформації для кінцевого споживача без необхідності ручного оновлення програмного коду сторінок [33].

Для забезпечення надійності системи було реалізовано централізований механізм обробки помилок. У разі виникнення збоїв у базі даних або некоректних запитів від клієнта, сервер генерує відповідні коди статусів (400, 401, 404, 500) із пояснювальними повідомленнями. Це дозволяє фронтенду коректно реагувати на непередбачувані ситуації, виводячи користувачеві інформативні сповіщення замість припинення роботи застосунку. Також було проведено оптимізацію взаємодії з базою даних через механізм фільтрації на рівні сервера. Замість завантаження всього масиву турів у пам'ять браузера, клієнт надсилає запит із конкретними параметрами (категорія, ціновий діапазон), а сервер виконує вибірку лише необхідних записів. Це суттєво зменшує навантаження на мережевий канал та пришвидшує роботу системи на мобільних пристроях.

3.3 Реалізація інструментів інтерактивної взаємодії з користувачем (системи сповіщень, фільтрація даних)

В межах розробки системи особлива увага була приділена створенню інтерактивного інтерфейсу, який забезпечує безперервний діалог між користувачем та програмним комплексом. Основними інструментами взаємодії

стали динамічні системи фільтрації, механізми миттєвих сповіщень та інтерактивні форми з валідацією даних. Система динамічної фільтрації була реалізована для забезпечення швидкого доступу до цільової інформації без необхідності повного оновлення сторінки, що значно знижує навантаження на сервер та покращує показники продуктивності інтерфейсу. Механізми миттєвих сповіщень інтегровані для надання користувачу актуального зворотного зв'язку щодо статусу виконання операцій (успішне збереження, виникнення помилок або попередження), що робить роботу з програмою прогнозованою та зрозумілою. Інтерактивні форми з вбудованою валідацією забезпечують перевірку коректності введених даних у реальному часі безпосередньо на стороні клієнта [34]. Це дозволяє виявляти помилки на етапі введення, надаючи користувачу візуальні підказки та запобігаючи передачі некоректних запитів до бази даних. Впровадження цих інструментів у сукупності дозволило створити гнучке та адаптивне середовище для ефективної взаємодії користувача з функціональними модулями системи.

Для підвищення ефективності роботи з інформаційними масивами реалізовано модуль динамічної фільтрації. Користувач має можливість миттєво відсіювати зайву інформацію за декількома параметрами (наприклад, країна, вид подорожі) (рис.3.14). Технічно це реалізовано шляхом передачі стейту (state) фільтрів до API-запиту. Це дозволяє системі динамічно оновлювати контент на сторінці без її повного перезавантаження, що мінімізує час очікування та зменшує навантаження на сервер. Алгоритм обробки запиту передбачає формування динамічного SQL-запиту на стороні бекенд-частини застосунку, де кожен обраний користувачем параметр стає частиною умови фільтрації. Використання асинхронних запитів дозволяє підтримувати високу чуйність інтерфейсу, оскільки оновлюється лише контейнер із цільовими даними, а не вся структура DOM-дерева. Такий підхід забезпечує плавну взаємодію користувача з каталогом, дозволяючи швидко знаходити релевантні пропозиції навіть за умови значного обсягу записів у базі даних.

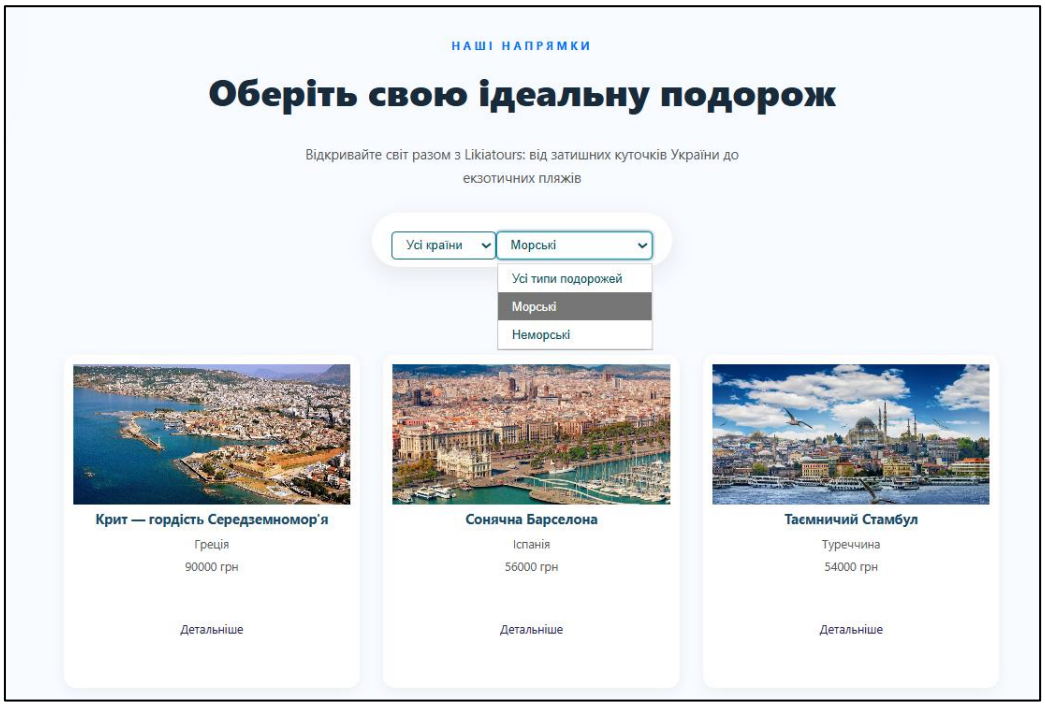

Рис. 3.14 Інтерфейс користувача з активованими фільтрами за країною та видом подорожі

Для інформування користувача про результати його дій реалізовано систему інтерактивних сповіщень (Toast-повідомлення). Система автоматично генерує відгук на кожну ключову подію:

– інформаційні сповіщення (наприклад, підтвердження відправки форми (рис.3.15));

– попередження (повідомлення про незаповнені обов'язкові поля);

– повідомлення про помилки (відсутність з’єднання з базою даних або помилка авторизації). Це дозволяє користувачу миттєво розуміти стан системи та коректність своїх маніпуляцій.

Важливою частиною взаємодії є механізм "живої" валідації введених даних. Замість очікування відповіді від сервера після натискання кнопки «Надіслати», система перевіряє коректність введення (наприклад, формат email або довжину пароля) безпосередньо під час заповнення полів. Це реалізовано за допомогою обробників подій на стороні фронтенду, що дозволяє виявляти помилки на ранньому етапі та надавати користувачу підказки щодо їх виправлення. Зокрема, перевірка форматів даних здійснюється за допомогою

регулярних виразів (RegEx), що гарантує відповідність введеної інформації встановленим стандартам ще до моменту формування HTTP-запиту.

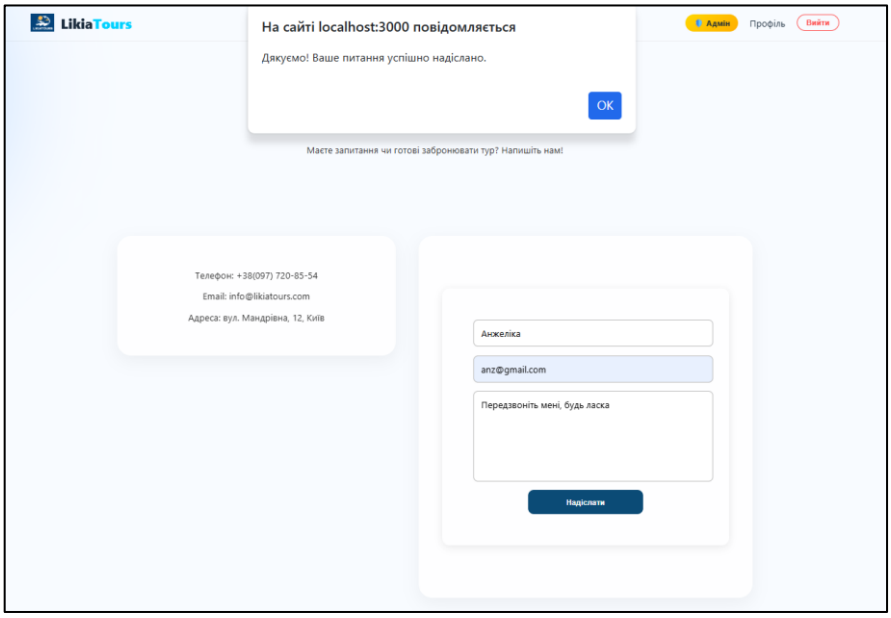

Рис. 3.15. Приклад візуалізації Toast-сповіщення при успішній взаємодії з системою

Такий підхід дозволяє суттєво розвантажити серверний компонент системи, оскільки він отримує лише заздалегідь валідовані та очищені дані. Крім того, миттєва візуальна індикація (наприклад, зміна кольору меж поля або виведення повідомлень-підказок) знижує когнітивне навантаження на користувача та запобігає роздратуванню від повторного заповнення всієї форми після невдалої відправки. Це сприяє підвищенню загального рівня довіри до інтерфейсу та робить процес взаємодії з інформаційною системою більш безперервним та інтуїтивно зрозумілим.

3.4 Організація безпечної авторизації та розмежування прав доступу

Організація безпечної авторизації та чітке розмежування прав доступу є фундаментальним аспектом розробки інформаційної системи, що забезпечує захист персональних даних та стабільність бізнес-процесів. У межах даного

проєкту було впроваджено систему автентифікації, засновану на рольовій моделі доступу (RBAC), що дозволяє гнучко керувати можливостями різних категорій користувачів. Процес входу до системи реалізовано з дотриманням сучасних стандартів безпеки, де паролі проходять процедуру хешування перед збереженням у базі даних, що унеможливлює їх компрометацію у відкритому вигляді. Сесії взаємодії захищаються за допомогою шифрованих токенів, які гарантують ідентифікацію користувача протягом усього часу роботи з вебзастосунком та запобігають несанкціонованому перехопленню прав доступу [35].

Особливу увагу приділено розробці захищеної адміністративної панелі, яка виступає центральним вузлом управління контентом та користувачами (рис. 3.16). Доступ до цього модуля суворо обмежений і надається виключно обліковим записам із привілеями адміністратора.

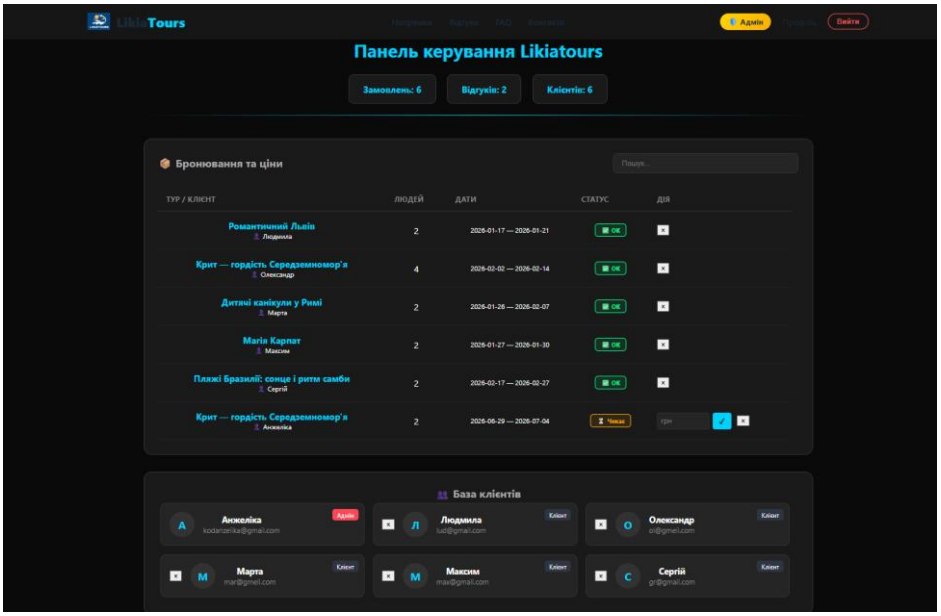

Рис. 3.16 Адміністративна панель сайту

Інтерфейс адміністративної частини реалізовано з урахуванням потреби у швидкій обробці інформації, забезпечуючи повний цикл операцій над даними: від створення та редагування описів подорожей до модерації вхідних заявок і керування станом клієнтської бази. Візуальна логіка адмін-панелі відокремлена

від користувацької частини, що дозволяє уникнути випадкових змін у структурі даних та забезпечує високу ергономічність при виконанні рутинних операцій.

Важливою складовою екосистеми є інтеграція з сервісами електронної пошти, що виступає інструментом додаткової верифікації та зворотного зв'язку. Взаємодія з поштою реалізована на основі протоколу SMTP, що дозволяє системі автоматично генерувати та відправляти персоналізовані листи при настанні певних подій. Наприклад, після успішного бронювання користувач отримує деталізоване повідомлення на свою електронну адресу (рис. 3.17). Використання професійно оформлених HTML-шаблонів для листів не лише покращує візуальне сприйняття сервісу, але й підтверджує надійність системи, створюючи для користувача прозоре та безпечне інформаційне середовище. У сукупності ці заходи забезпечують комплексний захист системи та високу якість інтерактивної взаємодії.

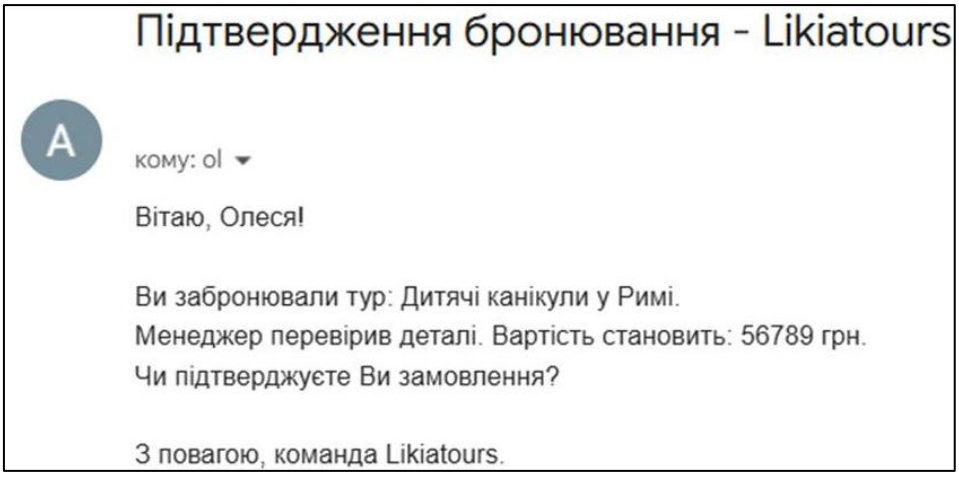

Рис. 3.17 Отриманий лист в поштовій скриньці, що підтверджує взаємодію системи з користувачем

Окрім розмежування прав на рівні інтерфейсу, особлива увага була приділена безпеці на рівні обробки запитів до бази даних. Для запобігання поширеним вразливостям, таким як SQL-ін'єкції, у системі застосовано метод параметризованих запитів та використання об'єктно-реляційного відображення (ORM). Це гарантує, що будь-які дані, введені адміністратором або

користувачем, проходять сувору фільтрацію перед виконанням на стороні сервера. Такий підхід створює додатковий бар'єр захисту, що є критично важливим для адміністративної панелі, де зосереджені інструменти повного контролю над контентом та фінансовою інформацією (якщо передбачено платежі).

Додатково реалізовано механізм контролю за станом сесій, який автоматично завершує активний доступ після певного періоду бездіяльності користувача. Це мінімізує ризики, пов'язані із залишенням відкритого доступу до адміністративної панелі на публічних або спільних пристроях. Валідація прав доступу відбувається при кожному зверненні до захищеного маршруту (endpoint), що унеможливлює обхід системи авторизації шляхом прямого введення URL-адреси адміністративного розділу в браузері. Кожна спроба несанкціонованого доступу фіксується системою, що в перспективі дозволяє проводити аудит безпеки [36, 37].

Ефективність впровадженої інтеграції з поштовими сервісами також проявляється у функціоналі відновлення доступу. У разі втрати пароля користувач може скористатися захищеним алгоритмом скидання, що передбачає генерацію унікального тимчасового посилання, яке надсилається на верифіковану електронну адресу. Це не лише спрощує експлуатацію системи, але й відповідає сучасним стандартам самообслуговування користувачів, знімаючи навантаження з технічної підтримки адміністратора. Таким чином, розроблена архітектура безпеки є комплексною, охоплюючи як внутрішню логіку збереження даних, так і зовнішні канали взаємодії.

Висновки розділу 3

У третьому розділі було проведено детальну розробку та опис програмної реалізації інформаційної системи, що базується на сучасному технологічному стеку та відповідає вимогам до продуктивності й безпеки. У ході виконання практичної частини роботи було реалізовано інтерактивний інтерфейс

користувача на основі компонентного підходу, що дозволило впровадити інструменти динамічної фільтрації та асинхронного оновлення контенту. Застосування таких рішень забезпечило високу швидкість відгуку системи та значне покращення користувацького досвіду завдяки мінімізації навантаження на серверний компонент. Важливою складовою реалізації стало створення багаторівневої системи зворотного зв’язку, яка включає механізми миттєвих сповіщень та валідацію введених даних у реальному часі, що дозволило мінімізувати ризики виникнення помилок та зробити процес взаємодії з системою більш інтуїтивним.

Паралельно з інтерфейсною частиною було побудовано надійну модель безпеки, засновану на рольовому розмежуванні прав доступу між користувачами та адміністраторами. Використання сучасних методів хешування паролів та механізмів контролю сесій гарантує високий рівень захисту персональної інформації від несанкціонованого доступу. Особливу увагу було приділено розробці функціональної адміністративної панелі, яка забезпечує повний контроль над контентом та бізнес-логікою застосунку. Завдяки інтеграції з поштовими сервісами через протокол SMTP було автоматизовано процеси верифікації та інформування клієнтів, що значно підвищило автономність системи. Таким чином, розроблений програмний продукт є технічно завершеним, має високий ступінь захищеності та повністю готовий до експлуатації у межах поставлених завдань.

ВИСНОВКИ

У даній роботі проведено комплексне дослідження, результатом якого стала розробка та реалізація сучасної інформаційної системи, що повністю відповідає актуальному рівню наукових і технічних знань у галузі розробки вебзастосунків. Одержані результати підтверджують ефективність обраного технологічного стека та методологій проєктування, забезпечуючи високу продуктивність, масштабованість та надійність програмного продукту. Наукова та науково-технічна значимість роботи полягає в удосконаленні підходів до створення інтерактивних інтерфейсів та впровадженні багаторівневої системи безпеки, що дозволяє мінімізувати ризики виникнення вразливостей і забезпечити конфіденційність даних користувачів. Розроблена методика інтеграції динамічної фільтрації та систем миттєвого інформування у поєднанні з автоматизацією поштових розсилань дозволила створити гнучке середовище для ефективної взаємодії користувача з функціональними модулями системи.

Практична цінність результатів дослідження підтверджується створенням функціонального прототипу, який може бути впроваджений у діяльність організацій відповідного спрямування (наприклад, у сфері туризму чи послуг) як інструмент автоматизації бізнес-процесів та підвищення якості обслуговування клієнтів. Соціально-економічний ефект від використання розробленої системи полягає в оптимізації витрат робочого часу персоналу та підвищенні лояльності користувачів через надання зручного цифрового сервісу. Доцільність подальших досліджень за цією тематикою зумовлена стрімким розвитком вебтехнологій та потенціалом впровадження методів інтелектуального аналізу даних для персоналізації контенту. На основі проведеної роботи рекомендується подальше вдосконалення об'єкта дослідження через розширення функціональних можливостей адміністративної панелі та інтеграцію додаткових аналітичних модулів. Обґрунтованим є проведення подальших дослідно-технологічних робіт для масштабування системи та створення на її базі комерційного продукту, адаптованого до потреб сучасного ринку інформаційних послуг.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dijkstra E. W. A discipline of programming. Englewood Cliffs: Prentice-Hall, 1976. 217 p.

2. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 p.

3. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Diss. ... Doctor of Philosophy. Irvine: University of California, 2000. 162 p.

4. Abramov D. Live React: Hot Reloading with Time Travel : conference paper. 2015. P. 1–5.

5. Пєчкурова Е. Огляд сучасних фреймворків для web розробки сервісу «органайзер студента». 2025. 15 с.

6. Myhrvold N. Strategic Value of Innovation and Technological Sparks. Intellectual Ventures Insights. 2014. URL: https://www.intellectualventures.com (дата звернення: 19.05.2026).

7. Wade, Imogen Sophie Kristin. Igniting technological modernization through science towns and technology parks: the case of Russia. Diss. UCL (University College London), 2020.

8. Теницька А. С. Клієнтська частина веб-сайту за допомогою JavaScript : навчальний посібник. 2020. С. 6–7.

9. Токмакова І. В. Цифровізація як нова парадигма управління підприємствами туристичної індустрії. Вісник економіки транспорту і промисловості. 2022. № 78-79. С. 5–7.

10. Новикова М. В., Лола Ю. Ю. Особливості менеджменту діяльності туристичних фірм. Географія та туризм. 2021. № 11. С. 47–54.

11. Антолик Г., Кампов Н. Особливості роботи менеджера в туристичній фірмі : методичні матеріали. 2022. С. 3–7.

12. Петруша К. В. Веб-студія з дослідженням процесу створення шаблонів адаптивних веб-сайтів. 2021. 52 с.

13. Гарбар Г. А. Основні тенденції та особливості впливу цифрових технологій на трансформацію туристичних послуг. Modern Economics. 2022. URL: https://modecon.mnau.edu.ua/main-trends-and-features-of/

14. Пахомова В., Тодоров В. Дослідження архітектури розподіленої бази даних в системах управління закупівель B2B компаній. Вісник Херсонського національного технічного університету. 2025. № 2.2 (93). С. 281–287.

15. Гордєєва-Герасимова Л. Впровадження CRM-системи на підприємстві. Herald of Khmelnytskyi National University. Economic Sciences. 2022. Vol. 312, № 6 (2). С. 1–5.

16. Корогод Г. О., Мосійчук Д. І. Дослідження структури організації веб-сайту під час просування в мережі. Інформаційні технології в науці, виробництві та підприємництві. 2023. С. 1–5.

17. Рудовіл Є. А. Розробка веб-додатку для управління фінансами з використанням Python і Flask. 2024. 45 с.

18. Пономарьова О. А., Гузь Г. М., Гузь Д. В. Розробка системи реєстрації та авторизації користувачів у клієнт-серверному web-додатку. 2025. С. 393–398.

19. Легка Х. А. Тестування та розробка frontend-частини медіа-сховища з використанням Nuxt. 2024. 59 с.

20. Безверхий О., Куценко О. Ефективність застосування бібліотеки React. Інформаційні технології та суспільство. 2022. С. 13–19.

21. Терещенко О. О. Дослідження методів аналізу даних вебсайтів (на прикладі сайту по продажу відеоігор). 2025. С. 24–37.

22. Гнатишин М. А. Аналіз стану прогресивних технологій для створення веб-застосунків. 2023. 24 с.

23. Лукашук М. Особливості дизайну при створенні веб-сервісів та мобільних застосунків з врахуванням інклюзивності. Collection of scientific papers «ΛΌГОΣ». Zürich, 2023. С. 1–2.

24. Стадник П. О. Проєктування та розробка веб-сайту «BeCalm». 2022. 37 с.

25. Ткачов В. Аналіз JavaScript-фреймворків для розроблення вебзастосунків. Integration of science as a mechanism of effective development: матеріали XI Міжнар. наук.-практ. конф. 2023. 467 с.

26. Бондар Д. С. Проєктування та розробка веб-сайту «Посібник з HTML5». 2022. 41 с.

27. Гарькава В. Ф. Тренди розвитку менеджменту та бізнес-технологій в умовах формування сучасної української економіки. Академічні візії. 2023. № 16. С. 2–3.

28. Дмитришин Б. В., Титаренко А. В. Використання сучасних інформаційних технологій у туристичній галузі. Наукові праці КНТУ. 2020. № 27. С. 278–283.

29. Зуєва Н. Ю. Розробка серверної частини мобільного застосунку для спільноти книголюбів. 2025. С. 23–24.

30. Комлева Н. О., Нікітченко М. І. Структурне подання UML-метамоделі у форматі JSON. Інформатика та математичні методи в моделюванні. 2025. № 15.2. С. 1–3.

31. Лунак О. Розробка веб-застосунку для збору та опрацювання даних із використанням сучасних інформаційних технологій. Матеріали VIII Міжнародної студентської НТК. 2025. С. 171–172.

32. Дегтяренко О. В. Етапи та особливості розробки сучасного дизайну веб-сайту : наукове видання. 2023. С. 85–90.

33. Монастирський М. О. Розробка сервісу-портфоліо для розробників ПЗ з використанням Flask : бакалаврська робота. ТНТУ ім. І. Пулюя. 2025. 15 с.

34. Палєга Р. В., Карпенко Н. В., Герасимов В. В. Особливості авторизації за допомогою jwt токенів. 2024. С. 208–211.

35. Ушакова І. О., Конюхов Є. Є. Аналіз фреймворків для розроблення веб-застосунків : тези доповідей. 2020. 6 с.

36. Шапошнікова О., Степанов Г. Використання технології React.js для створення сучасних веб-застосунків: науковий звіт. 2024. С. 1–3.

37. Шевченко І. П. Створення веб-застосунку засобами React у середовищі Visual Studio Code: методичні матеріали. 2023. 28 с.

ДОДАТКИ

ДОДАТОК А

ПРОГРАМНА ДОКУМЕНТАЦІЯ

А.1. Технічне завдання на розробку вебдодатка для туристичної фірми

1. Найменування та галузь розробки

Об’єктом розробки є вебдодаток для автоматизації діяльності туристичної агенції «LikiaTours». Програмний продукт призначений для оптимізації процесів взаємодії з клієнтами та внутрішнього управління турами.

2. Призначення розробки

Система розроблена для забезпечення можливості онлайн-перегляду актуальних пропозицій клієнтами, оформлення заявок на бронювання та надання адміністратору (менеджеру) зручного інтерфейсу для оперативного керування контентом.

3. Технічні вимоги до програмного забезпечення

- Серверна частина: мова програмування Python, фреймворк Flask.
- Клієнтська частина: бібліотека React.js, HTML5, CSS3.
- База даних: реляційна СУБД (SQLite).
- Архітектура**:** Single Page Application (SPA) з RESTful API взаємодією.

4. Функціональні вимоги

- Відображення каталогу доступних турів з детальною інформацією.
- Система фільтрації та пошуку турів за параметрами.
- Форма зворотного зв’язку та оформлення замовлення.
- Адміністративна панель з функціями CRUD (створення, читання, оновлення, видалення) для управління турами.
- Адаптивність інтерфейсу для коректної роботи на десктопних та мобільних пристроях.

А.2. Посилання на віддалений репозиторій

Повний вихідний код проєкту, структура файлів та історія розробки розміщені у відкритому репозиторії на платформі GitHub за посиланням: https://github.com/AnzhelikaKodlubovska/LikiaTours_myproject

А.3. Лістинг програмного коду основних модулів

Нижче наведено фрагменти коду ключових компонентів системи:

1. Файл серверної частини (Backend – models.py)

```
# Опис структури таблиць застосунку в базі даних
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True,
nullable=False)
    password = db.Column(db.String(120), nullable=False)
    email = db.Column(db.String(120), unique=True,
nullable=False)
    bookings = db.relationship('Booking', backref='user',
lazy=True)
    reviews = db.relationship('Review', backref='user',
lazy=True)
    is_admin = db.Column(db.Boolean, default=False)

class Booking(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    tour_name = db.Column(db.String(120), nullable=True)
    guest_name = db.Column(db.String(100), nullable=False)
    guest_email = db.Column(db.String(100), nullable=False)
    persons = db.Column(db.Integer, nullable=False)
    check_in = db.Column(db.String(20), nullable=False)
    check_out = db.Column(db.String(20), nullable=False)
    status = db.Column(db.String(20), default='Pending')
    booking_date = db.Column(db.DateTime,
default=datetime.utcnow)
    price = db.Column(db.String(50), nullable=True)

class Review(db.Model):
    __tablename__ = 'reviews'
    id = db.Column(db.Integer, primary_key=True)
```

```
        user_id = db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
        content = db.Column(db.Text, nullable=False)
        city = db.Column(db.String(100))
        created_at = db.Column(db.DateTime,
default=datetime.utcnow)

    class ContactQuestion(db.Model):
        id = db.Column(db.Integer, primary_key=True)
        name = db.Column(db.String(100), nullable=False)
        email = db.Column(db.String(120), nullable=False)
        message = db.Column(db.Text, nullable=False)
        is_answered = db.Column(db.Boolean, default=False)
        created_at = db.Column(db.DateTime,
default=db.func.current_timestamp())

        def to_dict(self):
            return {
                "id": self.id,
                "name": self.name,
                "email": self.email,
                "message": self.message,
                "is_answered": self.is_answered,
                "date": self.created_at.strftime("%d.%m.%Y
%H:%M")
            }
```

2. Файл клієнтської частини (Frontend –BookingForm.jsx)

```
    # Програмна реалізація клієнтської форми бронювання та
інтеграція з API

    import React, { useState } from "react";
    import { API_BASE } from "../../constants";
    import "../../styles/ContactForm.css";

    export default function BookingForm({ tourName }) {
      const [name, setName] = useState("");
      const [email, setEmail] = useState("");
      const [persons, setPersons] = useState(1);
      const [checkIn, setCheckIn] = useState("");
      const [checkOut, setCheckOut] = useState("");
      const [loading, setLoading] = useState(false);

      const handleSubmit = async (e) => {
        e.preventDefault();

        const token = localStorage.getItem("token");

        // 1. Фільтрація на фронтенді
        if (!token || token === "null" || token === "undefined")
{
```

```
            alert(
              "На жаль, Ви не можете забронювати цей тур. Будь
ласка, зареєструйтесь.",
            );
            return;
          }

          setLoading(true);

          const bookingData = {
            guest_name: name,
            guest_email: email,
            tour_name: tourName || "Назва не визначена",
            persons: parseInt(persons),
            check_in: checkIn,
            check_out: checkOut,
          };

          try {
            const res = await fetch(`${API_BASE}/book`, {
              method: "POST",
              headers: {
                "Content-Type": "application/json",
                Authorization: `Bearer ${token}`,
              },
              body: JSON.stringify(bookingData),
            });

            const result = await res.json();

            if (res.ok) {
              alert(`Успіх! Ви забронювали тур: ${tourName}`);
              setName("");
              setEmail("");
              setPersons(1);
              setCheckIn("");
              setCheckOut("");
            } else {
              // 2. Обробка помилки, якщо вона все ж прийшла з
бекенду
              if (result.error &&
result.error.includes("segments")) {
                alert(
                  "На жаль, Ви не можете забронювати цей тур. Будь
ласка, зареєструйтесь.",
                );
              } else {
                alert(`Помилка: ${result.error || "Не вдалося
забронювати"}`);
              }
            }
          } catch (err) {
            console.error("Помилка при відправці:", err);
```

```
          alert("Сервер не відповідає. Перевірте з'єднання.");
        } finally {
          setLoading(false);
        }
      };

      return (
        <div className="booking-form-wrapper">
          <h3
            style={{ textAlign: "center", color: "#007bff",
marginBottom: "20px" }}
          >
            Бронювання: {tourName}
          </h3>

          <form className="contact-form" onSubmit={handleSubmit}>
            <div className="form-group">
              <input
                type="text"
                value={name}
                onChange={(e) => setName(e.target.value)}
                placeholder="Ваше ім’я"
                required
              />
            </div>

            <div className="form-group">
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                placeholder="Ваш Email"
                required
              />
            </div>

            <div className="form-group">
              <label>Кількість осіб:</label>
              <input
                type="number"
                min="1"
                value={persons}
                onChange={(e) => setPersons(e.target.value)}
                required
              />
            </div>

            <div className="form-group">
              <label>Дата заїзду:</label>
              <input
                type="date"
                value={checkIn}
                onChange={(e) => setCheckIn(e.target.value)}
```

```
                required
              />
            </div>

            <div className="form-group">
              <label>Дата виїзду:</label>
              <input
                type="date"
                value={checkOut}
                onChange={(e) => setCheckOut(e.target.value)}
                required
              />
            </div>

            <button type="submit" className="submit-btn"
disabled={loading}>
              {loading ? "Зачекайте..." : "Забронювати тур"}
            </button>
          </form>
        </div>
      );
    }
```

ЗГОДА здобувачки вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату
та розміщення в Репозитарії Університету

Я, **Кодлубовська Анжеліка Олександрівна**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

«Розробка комплексного веб-додатку туристичної фірми з підсистемою адміністрування замовлень та інтерактивної взаємодії з користувачами»

виконана самостійно та не містить академічного плагіату. Я не надавала і не одержувала недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформована, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлена.

15.06.26 р  Кодлубовська А. О.