

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до
захисту"

Завідувач кафедри

ГУСТ



—
к.т.н, доц. Михелєв І.Л.

" "
2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: «Розробка інформаційної системи туристичної фірми»

Виконав: студент групи 4141



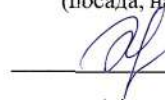
(підпис)

Левицький О.С.

Керівник роботи:

К.т.н., доцент

(посада, науковий ступінь вчене звання)



(підпис)

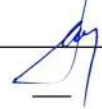
Книрик Н.Р.

м. Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"
Гарант освітньої
програми



к.т.н, доц. Гайда А.Ю.

" "

2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Левицькому Олександр Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної системи туристичної фірми

Керівник роботи Книрик Н.Р.

Затверджені наказом ректора № 270-уч від "31" березня 2026 року.

2. Термін подання роботи: 22 червня 2026 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) Аналіз предметної області та постановка задачі автоматизації турфірми. Проектування бази даних та архітектури інформаційної системи. Програмна реалізація та аналіз ефективності функціонування системи. Охорона праці.

5. Перелік презентаційних матеріалів Актуальність та мета автоматизації турфірми, аналіз аналогів, моделі Use Case та DFD, ER-діаграма бази даних, її нормалізація до 3NF, фізична структура в MySQL, архітектура веб-додатка, програмна реалізація на Flask та Bootstrap 5, тестування захищеності від SQL-ін'єкцій і висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Книрік Н.Р.	02.02.2026	23.02.2026
2	Книрік Н.Р.	23.02.2026	20.03.2026
3	Книрік Н.Р.	20.03.2026	24.04.2026
4	Книрік Н.Р.	24.04.2026	01.06.2026

7. Дата видачі завдання 02 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	02.02.2026	
2	Постановка задачі	16.02.2026	
3	Розробка концепції	23.02.2026	
4	Розробка проекту ІС	23.03.2026	
5	Реалізація проєкту	27.04.2026	
6	Розробка питань з охорони праці	01.06.2026	
7	Розробка графічних матеріалів	05.06.2026	
8	Подання роботи на перевірку на плагіат	08.06.2026	
9	Подання роботи рецензенту	15.06.2026	
10	Подання роботи на захист	22.06.2026	

Студент


(підпис)

Левицький О.С.

(ПІБ)

Керівник роботи


(підпис)

Книрік Н.Р.

(ПІБ)

АНОТАЦІЯ

Кваліфікаційну роботу присвячено розробці триланкової веб-орієнтованої інформаційної системи для автоматизації бізнес-процесів, обліку та адміністрування потоків даних у туристичній агенції. У межах роботи проведено системний аналіз предметної області, формалізовано інженерні вимоги та за допомогою методології DFD побудовано ієрархічну трирівневу модель інформаційного обміну. На етапі логічного проєктування структуру бази даних за допомогою методу концептуальної декомпозиції приведено до стану третьої нормальної форми (3NF), що дозволило повністю ліквідувати надлишковість даних та аномалії. Фізичне розгортання схеми виконано у СКБД MySQL із використанням транзакційного двигуна InnoDB (стандарт ACID) та побудовою B-Tree індексів для оптимізації пошукових запитів.

Програмну реалізацію серверної логіки виконано мовою Python на базі фреймворку Flask із впровадженням пулу з'єднань для забезпечення стабільного часу відгуку. Клієнтський інтерфейс користувача спроектовано як адаптивний веб-фронтенд за допомогою інструментів HTML5, CSS3, компонентів Bootstrap 5 та шаблонізатора Jinja2. Комплексне тестування та безпековий аудит OWASP підтвердили відмовостійкість системи та її абсолютну захищеність від атак типу SQL-ін'єкцій завдяки параметризації запитів. Також визначено пріоритетні напрямки подальшої модернізації системи до рівня розподіленої ERP-платформи.

Ключові слова: інформаційна система, автоматизація бізнес-процесів, туристична агенція, база даних, СКБД MySQL, InnoDB, нормалізація, 3NF, Python, Flask, Jinja2, Bootstrap 5, параметризовані запити, SQL-ін'єкція, DFD-моделювання.

ABSTRACT

The qualification thesis is devoted to the development of a three-tier web-oriented information system for business process automation, accounting, and data flow administration in a travel agency. Within the framework of the study, a systemic analysis was conducted, requirements were formalized, and a hierarchical three-level model of information exchange was constructed using DFD methodology. At the logical design stage, the database structure was brought to the Third Normal Form (3NF) via conceptual decomposition, eliminating data redundancy and anomalies. Physical deployment was performed in MySQL DBMS using the InnoDB transactional engine (ACID standard) and B-Tree indexes to optimize query execution time.

The server logic implementation was completed in Python using the Flask microframework with the integration of connection pooling to ensure stable response times. The client interface was designed as a responsive web frontend using HTML5, CSS3, Bootstrap 5 components, and server-side rendering with the Jinja2 template engine. Comprehensive testing and OWASP security auditing confirmed the system's fault tolerance and its absolute immunity to SQL Injection attacks due to the use of parameterized queries (Prepared Statements). Priority directions for further modernization into a distributed ERP platform were also determined.

Keywords: information system, business process automation, travel agency, database, MySQL DBMS, InnoDB, normalization, 3NF, Python, Flask, Jinja2, Bootstrap 5, parameterized queries, SQL injection, DFD modeling.

ЗМІСТ

ВСТУП	10
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ ТУРФІРМИ.....	13
1.1. Дослідження бізнес-процесів у сфері туристичного бізнесу та визначення вимог до автоматизації	13
1.2. Порівняльний аналіз чинних інформаційних систем-аналогів на ринку туристичних послуг	19
1.3. Обґрунтування вибору архітектурних рішень та технологічного стеку розробки	22
1.4. Функціонально-технологічне моделювання інформаційних потоків за методологією DFD	24
1.5. Постановка задачі, структуризація та класифікація вхідних і вихідних даних	28
Висновки до розділу 1	31
РОЗДІЛ 2. ПРОЄКТУВАННЯ БАЗИ ДАНИХ ТА АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	34
2.1. Концептуальне моделювання даних туристичної агенції та побудова ER-діаграми.....	34
2.2. Логічне проєктування та обґрунтування реляційної структури даних.....	40
2.3. Теоретичний аналіз та нормалізація розробленої бази даних (відповідність 1NF, 2NF, 3NF).....	44
2.4. Фізичне моделювання бази даних у СКБД MySQL та оптимізація типів даних.....	46
Висновки до розділу 2	49
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ СИСТЕМИ	51

3.1. Архітектурна побудова програмного забезпечення та реалізація серверної логіки на базі Flask	51
3.2. Розробка клієнтських веб-інтерфейсів, адаптивних форм взаємодії та інтеграція фронтенд-компонентів	53
3.3. Тестування працездатності, верифікація даних та забезпечення захищеності системи від SQL-ін'єкцій.....	57
3.4. Оцінка отриманих результатів, аналіз обмежень та визначення напрямків модернізації системи	59
Висновки до розділу 3	60
РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ	63
4.1. Аналіз умов праці та ідентифікація небезпечних і шкідливих виробничих чинників.....	63
4.2. Виробнича санітарія, ергономіка та організація робочого простору	64
4.3. Технобезпека, пожежна профілактика та дії в надзвичайних ситуаціях.....	65
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	69
ДОДАТОК А – КОД СТВОРЕННЯ ТАБЛИЦЬ MYSQL	71
ДОДАТОК Б -КОД СЕРВЕРНОЇ ЧАСТИНИ ПРОГРАМИ.....	73
ДОДАТОК В – ЛІСТИНГ КОДУ ІНТЕРФЕЙСУ	76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД — база даних.

ЗК — зовнішній ключ (Foreign Key).

ІС — інформаційна система.

НДІ — нормативно-довідкова інформація.

ПК — первинний ключ (Primary Key).

ПЗ — програмне забезпечення.

СКБД — система керування базами даних.

3NF (Third Normal Form) — третя нормальна форма в теорії реляційних баз даних.

ACID (Atomicity, Consistency, Isolation, Durability) — набір властивостей, що гарантують надійність виконання транзакцій у СКБД.

API (Application Programming Interface) — програмний інтерфейс взаємодії між додатками.

B-Tree (Balanced Tree) — збалансоване дерево, структура даних, яка використовується в СКБД для побудови індексів та прискорення пошуку.

CRUD (Create, Read, Update, Delete) — базові операції роботи з даними (створення, читання, оновлення, видалення).

CSS3 (Cascading Style Sheets 3) — каскадні таблиці стилів третього покоління.

DFD (Data Flow Diagram) — діаграма потоків даних, методологія функціонально-технологічного моделювання інформаційних процесів.

DDL (Data Definition Language) — мова опису структури даних (інструкції SQL для створення таблиць та індексів).

ENUM — перелічуваний тип даних у MySQL, що містить фіксований набір рядкових значень.

ERP (Enterprise Resource Planning) — система планування та управління ресурсами підприємства.

HTML5 (HyperText Markup Language 5) — п'ята версія мови розмітки гіпертексту для створення веб-сторінок.

HTTP (HyperText Transfer Protocol) — протокол передачі гіпертексту в мережі Інтернет.

InnoDB — транзакційний двигун зберігання даних у СКБД MySQL із підтримкою обмежень цілісності та блокування рядків.

MVC (Model-View-Controller) — архітектурний паттерн проєктування програмного забезпечення, що розділяє дані, логіку та інтерфейс.

OWASP (Open Web Application Security Project) — відкритий проєкт забезпечення безпеки веб-додатків.

RESTful API — архітектурний стиль взаємодії веб-служб за допомогою стандартних методів HTTP.

SQL (Structured Query Language) — мова структурованих запитів для роботи з реляційними СКБД.

SSR (Server-Side Rendering) — концепція динамічного рендерингу (генерації) сторінок веб-інтерфейсу на стороні сервера.

TCO (Total Cost of Ownership) — загальна вартість володіння програмним продуктом чи інфраструктурою.

UML (Unified Modeling Language) — уніфікована мова графічного моделювання для опису та проєктування програмних систем.

UI/UX (User Interface / User Experience) — інтерфейс користувача та досвід взаємодії користувача з системою.

VARCHAR — тип даних у SQL для збереження символьних рядків змінної довжини з оптимізацією пам'яті.

ВСТУП

Сучасний етап розвитку світової та національної економіки характеризується масштабною цифровою трансформацією всіх секторів обслуговування, серед яких індустрія туризму посідає одне з провідних місць за обсягами та швидкістю обміну інформаційними потоками. Ефективне функціонування сучасного туристичного бізнесу безпосередньо залежить від оперативності обробки персональних даних клієнтів, гнучкого адміністрування готельного фонду, координації логістичних маршрутів та прозорого обліку фінансових транзакцій. Традиційні підходи до ведення операційної діяльності, що базуються на паперовому документообігу або ізольованих електронних таблицях, призводять до виникнення інформаційних дефіцитів, системних помилок через людський фактор та суттєвого зниження швидкості обслуговування, що негативно впливає на конкурентоспроможність компанії на ринку.

Автоматизація бізнес-процесів за допомогою розробки спеціалізованих веб-орієнтованих інформаційних управляючих систем дозволяє централізувати збереження даних, забезпечити транзакційну цілісність, а також оптимізувати взаємодію між менеджерами та клієнтами агенції. Застосування сучасного технологічного стеку, зокрема високонадійних реляційних баз даних MySQL з транзакційним двигуном InnoDB та гнучких серверних веб-фреймворків, як-от Flask на базі мови програмування Python, створює передумови для розгортання масштабованих, швидкодіючих та безпечних програмних рішень із мінімальними операційними витратами. Зазначені чинники зумовлюють високу актуальність теми бакалаврської кваліфікаційної роботи, спрямованої на проєктування та програмну реалізацію інформаційної системи автоматизації туристичної агенції.

Мета дослідження полягає в теоретичному обґрунтуванні, інженерному проєктуванні та програмній реалізації веб-орієнтованої інформаційної системи для автоматизації операційної діяльності, обліку клієнтів, турів і бронювань

туристичної агенції з метою підвищення швидкодії обробки інформації та забезпечення високого рівня захищеності даних.

Для досягнення поставленої мети визначено та розв'язано низку послідовних інженерних завдань. Спочатку було досліджено бізнес-процеси у сфері туристичного бізнесу, визначено функціональні й нефункціональні вимоги до автоматизації та виконано порівняльний аналіз наявних систем-аналогів на ринку. Наступним кроком стала побудова ієрархічної моделі інформаційних потоків за допомогою методології діаграм потоків даних (DFD) та класифікація структури вхідних і вихідних даних системи. Далі було спроектовано концептуальну ER-модель даних предметної області, обґрунтовано перехід до логічної реляційної структури та проведено теоретичний аналіз нормалізації реляційних таблиць до рівня третьої нормальної форми (3NF) для усунення транзакційних аномалій. Після цього реалізовано фізичну модель бази даних у СКБД MySQL із оптимізацією типів даних та каскадних обмежень. На завершальних етапах розроблено архітектуру програмного забезпечення та серверну логіку веб-додатка на базі мікрофреймворку Flask із інтеграцією пулу з'єднань, створено адаптивний клієнтський інтерфейс на основі HTML5 та Bootstrap, а також проведено функціональне й безпекове тестування системи для верифікації захищеності від SQL-ін'єкцій за допомогою параметризованих запитів.

Об'єктом дослідження є процеси операційної діяльності, обліку, адміністрування та обробки інформаційних потоків у межах функціонування підприємства туристичної галузі.

Предметом дослідження є методи, моделі, інструменти проектування баз даних та програмні технології розробки триланкових веб-орієнтованих інформаційних систем автоматизації на основі мови Python, фреймворку Flask та СКБД MySQL.

Методи дослідження. Для розв'язання поставлених завдань у роботі застосовано комплекс методів комп'ютерних наук та системного аналізу. Методи структурного аналізу та моделювання у вигляді DFD-діаграм

використано для формалізації інформаційних потоків. Методологію сутніс-зв'язок (ER-моделювання) задіяно для концептуального проєктування, а математичну теорію реляційних баз даних та методи нормалізації відношень — для забезпечення цілісності сховища. Паттерни об'єктно-орієнтованого проєктування та методи веб-програмування послужили основою для реалізації клієнт-серверної архітектури додатка. Методи тестування програмного забезпечення й аудит безпеки застосовано для фінальної верифікації захищеності й стабільності розробленої системи.

Практичне значення отриманих результатів полягає у створенні готового до безпосереднього впровадження легковагового, відмовостійкого та безпечного програмного продукту на базі інструментальних засобів із відкритим вихідним кодом. Цей додаток дозволяє повністю автоматизувати робочі місця менеджерів туристичної агенції, забезпечуючи швидкий пошук турів за критеріями, безпечне збереження даних клієнтів у таблицях та надійний транзакційний облік фінансових розрахунків і статусів бронювань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ ТУРФІРМИ

1.1. Дослідження бізнес-процесів у сфері туристичного бізнесу та визначення вимог до автоматизації

Сучасний ринок туристичних послуг функціонує в умовах високої динамічності інформаційних потоків, значного обсягу неструктурованих даних та жорсткої конкуренції. Ефективність діяльності туристичної агенції безпосередньо залежить від швидкості обробки запитів клієнтів, оперативності взаємодії з туроператорами, готельними мережами та транспортними компаніями, а також від точності координації внутрішніх фінансово-господарських операцій. У традиційних умовах функціонування без залучення спеціалізованих засобів автоматизації більшість бізнес-процесів виконуються в ручному або напівавтоматизованому режимі з використанням загального офісного програмного забезпечення, що зумовлює виникнення низки критичних системних проблем. До них належать дублювання інформації, висока ймовірність виникнення помилок через людський фактор при внесенні персональних даних клієнтів, тривалий час формування звітних документів та відсутність єдиного централізованого сховища для моніторингу стану бронювань у реальному часі.

Для формалізації вимог до проектованої інформаційної системи необхідно провести детальний системний аналіз та декомпозицію ключових бізнес-процесів туристичної агенції відповідно до стандартів життєвого циклу програмного забезпечення. Основними наскрізними процесами у діяльності суб'єкта туристичного бізнесу є:

1. Реєстрація та верифікація даних клієнтів (як індивідуальних, так і корпоративних).
2. Формування та актуалізація каталогу туристичних продуктів (турів, екскурсійних програм, логістичних маршрутів).

3. Управління взаємодією з готельними комплексами та контрагентами.
4. Обробка заявок, бронювання турів та моніторинг статусів оплати.
5. Фінансовий облік транзакцій та аналітична звітність.

Процес взаємодії з клієнтом починається з етапу первинного звернення та збору вимог до відпочинку. Менеджер агенції здійснює пошук релевантних пропозицій у базі даних, узгоджує параметри туру (тривалість, категорія готелю, тип транспорту, наявність додаткових послуг) та фіксує персональні дані для укладання договору. На цьому етапі виникає потреба в автоматизованій перевірці коректності введення даних, оскільки помилки в паспортних даних або термінах дії документів призводять до фінансових збитків та юридичних ризиків під час оформлення віз чи авіаквитків. Після вибору конкретного турпродукту ініціюється бізнес-процес «Бронювання туру», який вимагає перевірки доступності місць у готелях та на рейсах, блокування відповідних ресурсів через зовнішні шлюзи або внутрішній реєстр та виставлення рахунку на оплату.

Процес управління взаємодією з готелями та туроператорами передбачає ведення детального обліку контрагентів із фіксацією їхньої категорійності, переліку базових та додаткових сервісів (трансфер, страхування, екскурсійний супровід), а також умов скасування бронювання. Складність цього блоку полягає у необхідності підтримання актуального стану великих масивів інформації, що динамічно змінюються залежно від сезонності, тарифних планів та спеціальних пропозицій. Ручна актуалізація таких даних у межах локальних таблиць призводить до десинхронізації інформації та зниження якості обслуговування.

Особливе місце в структурі діяльності агенції посідає бізнес-процес «Фінансовий облік та обробка платежів». Він охоплює фіксацію дати та часу транзакції, обчислення повної вартості туру з урахуванням наданих знижок або націнок за додаткові сервіси, облік внесеної передоплати, визначення залишку платежу та архівацію платіжних документів. Оскільки клієнти можуть

використовувати різні методи розрахунку (карткові платежі, готівкові кошти, банківські перекази), система повинна гнучко оперувати статусами транзакцій і забезпечувати цілісність даних: зміна статусу оплати повинна автоматично тригерувати зміну загального статусу бронювання (наприклад, перехід зі стану «в обробці» до стану «підтверджено»).

На основі проведеного аналізу предметної області та виявлених «вузьких місць» у бізнес-процесах сформовано детальний перелік функціональних та нефункціональних вимог до розроблюваної інформаційної системи туристичної агенції, що відповідає сучасним інженерним підходам до створення корпоративного ПЗ.

Функціональні вимоги:

Логічне розмежування функцій користувачів: забезпечення функціонування системи у форматі корпоративної панелі (робочого місця менеджера), де права доступу до модифікації та створення транзакцій надаються операторам внутрішнього контуру туристичної агенції, а низькорівневе адміністрування облікових записів та структури сховища винесено на рівень СКБД.

Управління клієнтською базою: забезпечення можливостей додавання, модифікації, видалення та контекстного пошуку профілів клієнтів (включаючи контактні та паспортні дані, історію взаємодії).

Управління турами та логістикою: ведення електронного каталогу турів із класифікацією за типами (автобусні, авіатури), категоріями комфорту (економ, стандарт, преміум), тривалістю, маршрутами та базовою вартістю.

Управління готельним фондом: облік готелів, їхньої зірковості, адресних даних та структурованого переліку послуг із прив'язкою до конкретних турів.

Автоматизація процесу бронювання: створення та ведення облікових карток бронювань із фіксацією дати, автоматичним розрахунком фінансових параметрів, прив'язкою до клієнта і туру, а також можливістю динамічної зміни логічного статусу замовлення.

Облік фінансових транзакцій: фіксація платежів із зазначенням суми, методу оплати та автоматичним контролем заборгованості за кожним окремим договором.

Нефункціональні вимоги:

Продуктивність та швидкодія: час відгуку системи на стандартні пошукові запити та операції читання/запису в базу даних не повинен перевищувати 1.5 секунди при одночасній роботі регламентованої кількості користувачів.

Надійність та цілісність даних: забезпечення транзакційності на рівні СКБД для запобігання втрати або спотворення даних при збоях, а також підтримання реляційної цілісності за допомогою обмежень зовнішніх ключів.

Безпека інформації: захист від несанкціонованого доступу, шифрування сесій користувачів, стійкість до поширених веб-уразливостей (ін'єкції кодів, підробка міжсайтових запитів) відповідно до стандартів безпеки.

Ергономічність та інтерфейс: наявність адаптивного, інтуїтивно зрозумілого веб-інтерфейсу, що забезпечує коректне відображення елементів керування на пристроях із різною роздільною здатністю екрана без порушення логіки навігації.

Масштабованість та супроводжуваність: модульна архітектура системи повинна дозволяти розширювати функціональний склад (наприклад, підключати зовнішні платіжні API або модулі розсилки повідомлень) без необхідності рефакторингу ядра системи.

Для формалізації розгалуження функціональних обов'язків операторів системи та визначення меж взаємодії користувачів із внутрішніми модулями додатка, на основі сформованих вимог розроблено діаграму прецедентів (Use Case). Оскільки поточна версія системи орієнтована на автоматизацію внутрішнього операційного контуру, головними акторами виступають менеджер з обробки замовлень та адміністратор бази даних. З метою розмежування рівнів доступу та визначення меж функціональних обов'язків

операторів було спроектовано рольову модель. Загальну структуру взаємодії акторів із прецедентами відображено на рисунку 1.1.



Рисунок 1.1 - Діаграма прецедентів (UML Use Case) для рольової моделі внутрішнього контуру системи

Для детального аналізу взаємодії користувача з інформаційним контуром системи було розроблено відповідний реєстр тригерів. Детальний розподіл реакцій програмного середовища на дії акторів представлено в таблиці 1.1.

Таблиця 1.1 Матриця подій

Сутність	Подія	Реакція системи
Клієнт	Пошук туру	Клієнт вводить тип туру (автобусний або перельотний), країни/міста маршруту, дати, кількість осіб.
Клієнт	Вибір автобусного туру	Система показує доступні автобусні тури: маршрут, тривалість, вартість, готелі, харчування, послуги.
Клієнт	Вибір перельотного туру	Система показує доступні перельотні тури: країна, дати,

		готель, харчування, трансфери, розваги.
Клієнт	Заповнення інформації	Клієнт вносить дані про туристів (ім'я, дата народження, паспортні дані, контакти).
Клієнт	Замовлення додаткових послуг	Вибираються додаткові послуги: VIP-трансфер, страхування, вибір місця, екскурсії.
Клієнт	Оплата	Система формує загальну вартість і пропонує методи оплати
Туристична фірма	Додавання нових турів	Додаються нові автобусні або перельотні тури з інформацією про маршрут, дати, послуги, проживання.
Туристична фірма	Обробка замовлення	Отримання даних про клієнтів, підтвердження бронювання, надання ваучерів, квитків.

Таким чином, автоматизація досліджених бізнес-процесів шляхом розробки та впровадження спеціалізованої інформаційної системи є критично необхідним кроком для оптимізації операційної діяльності туристичної агенції, підвищення швидкості обслуговування клієнтів, мінімізації ризиків

втрати даних та забезпечення належного рівня контролю за фінансовими потоками підприємства.

1.2. Порівняльний аналіз чинних інформаційних систем-аналогів на ринку туристичних послуг

Для обґрунтування необхідності проектування та реалізації індивідуального програмного рішення для автоматизації бізнес-процесів туристичної агенції проведено системний порівняльний аналіз чинних інформаційних систем-аналогів, представлених на глобальному та локальному ринках туристичних послуг. На основі аналізу практики цифровізації підприємств індустрії туризму було виокремлено три домінантні класи рішень: глобальні споживчі B2C-платформи бронювання, спеціалізовані хмарові CRM/ERP-системи для туристичного бізнесу, а також локальні напівавтоматизовані інструменти.

Першим типом аналогів є глобальні агрегатори та системи онлайн-бронювання, найяскравішими представниками яких є платформи Booking.com та Expedia.

Booking.com пропонує користувачам високонадійну інфраструктуру для пошуку та резервування готелів, гостьових будинків і супутніх туристичних локацій, що підтримується гнучкою системою багатокритеріальної фільтрації, користувацьких відгуків та інтегрованих шлюзів онлайн-оплати. Проте архітектура даного сервісу повністю орієнтована на кінцевого споживача (B2C) і не містить функціональних модулів для ведення внутрішнього фінансового обліку агенції, управління кадровими ресурсами, формування внутрішньої звітності та ведення специфічних автобусних або екзотичних авіатурів з боку турфірми.

Expedia розширює цей функціонал шляхом інтеграції крос-платформних можливостей, включаючи бронювання авіаквитків, оренду транспортних засобів та формування комплексних динамічних пакетів через мобільні додатки. Незважаючи на технологічну досконалість та використання

потужних розподілених баз даних, Expedia залишається закритим комерційним екосистемним рішенням, яке не підлягає кастомізації під бізнес-модель конкретного локального агентства та вимагає сплати високих комісійних відрахувань, що суттєво підвищує загальну вартість володіння (TCO).

Другим типом аналогів є спеціалізовані вертикальні CRM/ERP-системи, розроблені безпосередньо для автоматизації внутрішніх процесів туроператорів та турагентств, прикладом яких є система TourSys.

TourSys орієнтована на комплексне керування взаємовідносинами з клієнтами (CRM) та адміністрування фінансових потоків туристичної фірми. Вона надає менеджерам інструментарій для відстеження статусів лідів, формування угод та контролю взаєморозрахунків.

Основними недоліками систем класу TourSys є застарілий або надмірно ускладнений інтерфейс користувача, що знижує швидкість адаптації персоналу, а також висока вартість ліцензування за моделлю SaaS (Software as a Service). Крім того, готові хмарові модулі часто мають жорстку структуру даних, яка не дозволяє гнучко розділяти специфічні параметри автобусних маршрутів по Європі (наприклад, графіки переїздів, зупинки) та перельотних турів до екзотичних країн (наприклад, норми багажу, VIP-трансфери).

Для систематизації результатів порівняльного аналізу та визначення вектору індивідуальної розробки було сформовано таблицю 1.2 порівняння інформаційних систем за ключовими технічними та експлуатаційними критеріями:

Таблиця 1.2 Порівняння інформаційних систем

Критерії порівняння аналогів	Booking.com	Expedia	TourSys	Проектована система
Орієнтація системи	Клієнтська (B2C)	Клієнтська (B2C)	Корпоративна (B2B)	Гібридна (B2B / Внутрішня)

Автоматизація CRM турфірми	Відсутня	Відсутня	Повна	Оптимізована під бізнес-процеси
Управління специфічними типами турів	Тільки готелі / авто	Пакувальні рішення	Спрощений облік	Повне (автобусні та авіатури)
Ергономіка інтерфейсу	Висока	Висока	Низька / застаріла	Висока (мінімалістична адаптивна)
Можливість модифікації коду	Відсутня	Відсутня	Обмежена налаштуваннями	Повна (відкритий вихідний код)
Захист від SQL-ін'єкцій	Повний (закритий API)	Повний (закритий API)	Повний (хмаровий провайдер)	Реалізовано параметризацією
Загальна вартість володіння (TCO)	Висока комісія	Висока комісія	Щомісячна підписка	Мінімальна (Open Source стек)

Проведений аналіз чинних аналогів доводить, що комерційні рішення на ринку туристичних послуг або повністю зосереджені на зовнішньому бронюванні без урахування операційних потреб менеджменту турфірми, або є занадто дорогими та жорсткими для кастомізації. Це зумовлює актуальність та доцільність індивідуального проєктування розроблюваної інформаційної системи, яка поєднає гнучку реляційну базу даних, адаптовану під специфіку автобусних та авіатурів, з легковагою та безпечною серверною логікою, мінімізуючи капітальні витрати підприємства.

1.3. Обґрунтування вибору архітектурних рішень та технологічного стеку розробки

Проектування сучасних інформаційних систем потребує детального аналізу та обґрунтування архітектурного паттерну і технологічного стеку з метою забезпечення гнучкості, масштабованості, безпеки та мінімізації витрат на розгортання та супровід програмного продукту. Для розробки інформаційної системи туристичної агенції було обрано триланкову архітектуру, яка передбачає чітке розділення системи на рівень представлення (клієнтська частина), рівень бізнес-логіки (серверна частина) та рівень керування даними (база даних).

В основі серверної архітектури розроблюваної інформаційної системи покладено принципи модульного проектування, де рівень представлення (інтерфейсна частина на базі шаблонів Jinja2) логічно відокремлений від реляційного сховища даних. Використання мікрофреймворку Flask дозволяє реалізувати концепцію монолітного модульного ядра (app.py), у якому обробники маршрутів беруть на себе роль контролерів, що взаємодіють із базою даних за допомогою параметризованих SQL-запитів. Такий підхід забезпечує високу швидкість обробки HTTP-запитів, спрощує зневадження коду та унеможливорює надлишковість програмних компонентів для архітектур класу MVP. На основі техніко-економічного аналізу для реалізації системи було визначено такий технологічний стек:

Мова програмування бізнес-логіки: Python. Вибір зумовлений високим рівнем абстракції мови, лаконічністю синтаксису, що прискорює розробку, та наявністю потужних інтегрованих інструментів для верифікації даних. Python забезпечує безпечну роботу з пам'яттю та має розвинену екосистему бібліотек для взаємодії з реляційними СКБД.

Серверний веб-фреймворк: Flask. Flask належить до категорії мікрофреймворків, що забезпечує мінімальне системне навантаження та високу швидкість обробки HTTP-запитів. На відміну від монолітних

фреймворків (наприклад, Django), Flask не нав'язує жорстких архітектурних обмежень, надаючи розробнику можливість гнучко проєктувати структуру додатку, підключати лише необхідні модулі залежно від масштабу системи та гнучко оптимізувати маршрутизацію.

Система керування базами даних (СКБД): MySQL. Для автоматизації турфірми критично важливою є підтримка транзакційності (відповідність вимогам ACID) для унеможливлення втрати фінансових даних або виникнення овербукінгу (подвійного бронювання одного місця). MySQL з двигуном зберігання даних InnoDB забезпечує надійну підтримку зовнішніх ключів, механізми блокування рядків під час транзакцій та високу швидкість виконання складних JOIN-запитів при генерації звітів.

Драйвер підключення: `mysql-connector-python`. Офіційний API-інтерфейс від Oracle, що реалізує специфікацію Python DB API v2.0. Його використання забезпечує коректне низькорівневе передавання типів даних між Python та MySQL, підтримує пул з'єднань та надає вбудовані механізми для автоматичної параметризації SQL-запитів, що є базовим інструментом захисту від атак типу SQL-ін'єкцій.

Технології побудови інтерфейсу: HTML5, CSS3, Bootstrap. Клієнтська частина базується на стандартах HTML5 для розмітки адаптивних форм взаємодії та CSS3/Bootstrap для стилізації інтерфейсу. Використання Bootstrap дозволяє створити інтерфейс, який коректно адаптується під мобільні пристрої та робочі станції менеджерів без залучення важких клієнтських JavaScript-фреймворків, що зберігає високу швидкість завантаження сторінок за умов обмеженого інтернет-з'єднання.

Обрання зазначеного технологічного стеку забезпечує повну кросплатформеність інформаційної системи (можливість розгортання як на базі ОС Linux, так і Windows Server), а використання виключно технологій із відкритим вихідним кодом (Open Source) повністю нівелює капітальні витрати на придбання комерційних ліцензій. Це дозволяє створити легковагове, стійке до навантажень та безпечне програмне рішення, що повністю відповідає

критеріям автоматизації підприємств малого та середнього туристичного бізнесу.

1.4. Функціонально-технологічне моделювання інформаційних потоків за методологією DFD

Для формалізації архітектури інформаційних потоків, визначення меж інформаційної системи та деталізації процесів перетворення даних застосовано методологію діаграм потоків даних (Data Flow Diagram). На відміну від об'єктно-орієнтованих моделей, DFD дозволяє наочно продемонструвати інженерну логіку руху інформації від зовнішніх сутностей через обробники (контролери серверної логіки) до реляційних сховищ.

Моделювання побудовано за принципом ієрархічної декомпозиції, що включає три рівні деталізації:

Контекстний рівень (DFD 0-го рівня): Визначає глобальні межі системи. Центральний процес «0. Система бронювання» взаємодіє з двома зовнішніми сутностями: «Клієнт» та «Туристична фірма». Клієнт генерує потік вхідних параметрів та особистих даних, отримуючи у відповідь верифіковані списки турів та сформовані документи. Туристична фірма виступає адміністратором метаданих (актуалізація турів, цін) та отримує інформацію про нові транзакції й бронювання. Глобальну взаємодію центрального процесу із зовнішніми сутностями продемонстровано на рисунку 1.2.

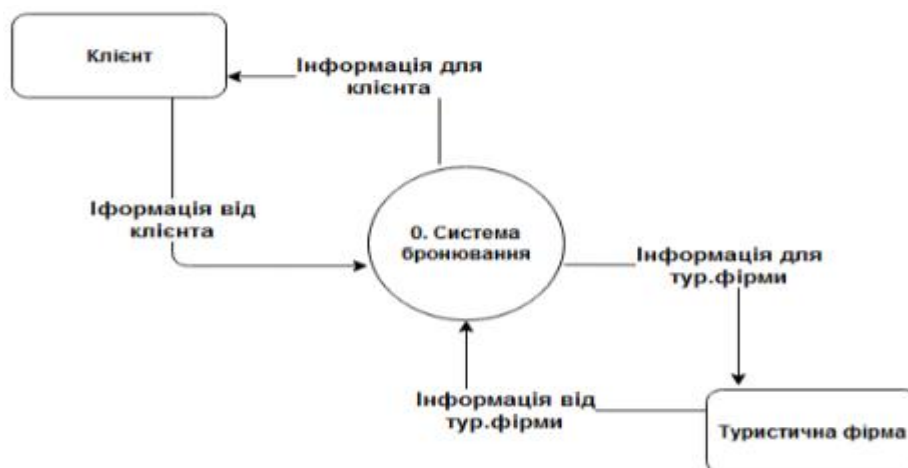


Рисунок 1.2 – DFD 0-го рівня

Рівень декомпозиції процесів (DFD 1-го рівня): Розгортає центральний обробник на 5 автономних функціональних модулів:

1. Пошук та вибір туру: приймає критерії фільтрації від користувача, здійснює запити до реляційних таблиць і повертає динамічно сформований перелік доступних турів.

2. Заповнення інформації клієнтом: відповідає за збір персональних і паспортних даних, їх первинну валідацію на стороні сервера та передачу до буфера сховища.

3. Замовлення додаткових послуг: забезпечує зв'язування обраного бронювання зі специфічними сервісами (VIP-трансфер, страхування).

4. Обробка оплати: взаємодіє з фінансовими атрибутами, перевіряє суму та метод розрахунку.

5. Підтвердження бронювання: фіналізує транзакцію, змінює логічний статус у базі даних та формує вихідний пакет документів (ваучери, квитки).

Взаємозв'язок між п'ятьма основними підпроцесами обробки даних ідентифіковано на рисунку 1.3.

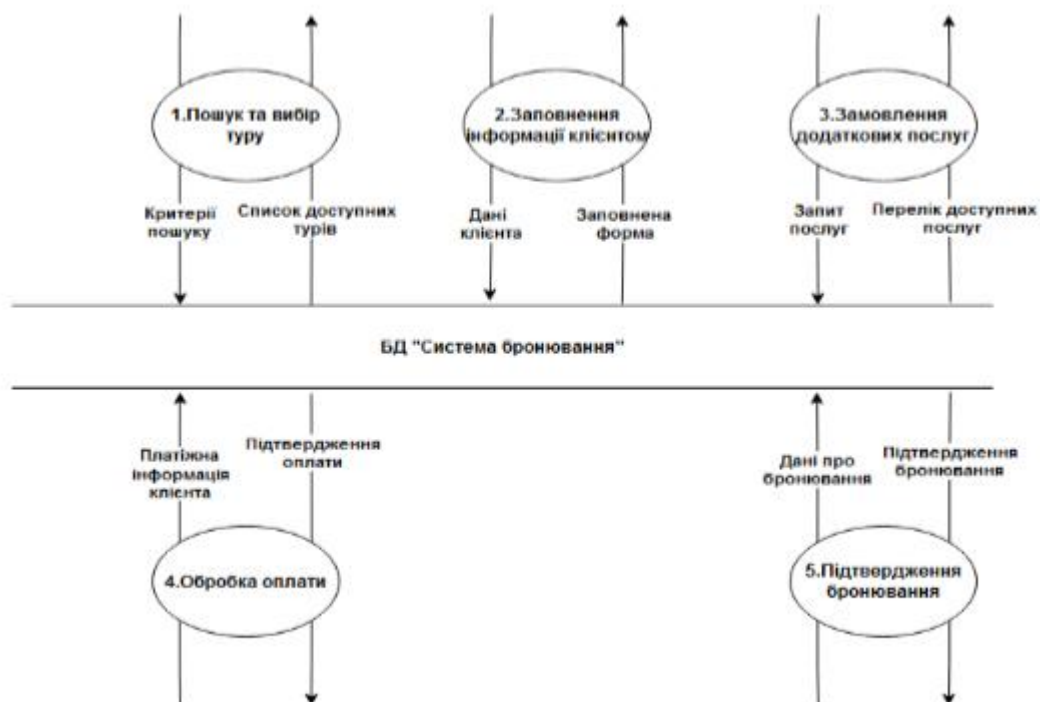


Рисунок 1.3 - DFD 1-го рівня

Рівень деталізації операцій (DFD 2-го рівня): Декомпозує складний процес «1. Пошук та вибір туру» на підетапи: «1.1 Введення параметрів пошуку» (захоплення HTTP-параметрів), «1.2 Відображення результатів пошуку» (генерація SQL-запиту та рендеринг HTML-таблиці) та «1.3 Вибір туру» (передача унікального TourID до сесії користувача).

Схема перетворення інформаційних потоків на рівні операцій контролера Flask наведена на рисунку 1.4.

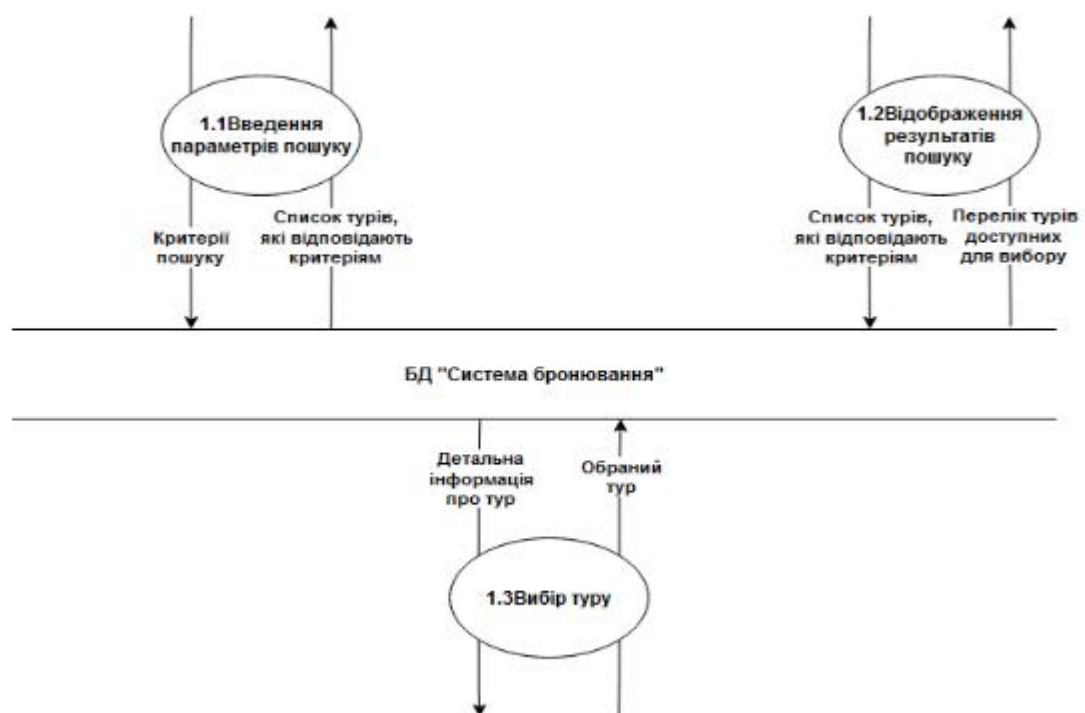


Рисунок 1.4 - DFD 2-го рівня

Всі процеси синхронізовані з єдиним центральним накопичувачем даних - БД «Система бронювання». Для забезпечення технологічної точності розробки взаємозв'язок між процесами та атрибутами інформаційних потоків типізовано й зведено у таблиці 1.3.

Таблиця 1.3 Таблиця відповідності потоків

Вхідний/Вихідний потік (DFD-0)	Процес-обробник (DFD-1)	Підпроцес (DFD-2)	Специфікація технічних атрибутів даних
Інформація від клієнта	1. Пошук та вибір туру	1.1 Введення параметрів	TourType (ENUM), Route (TEXT), StartDate (DATE), Budget (DECIMAL)
Інформація від клієнта	2. Заповнення інформації		FirstName, LastName, PassportData, DateOfBirth, Email
Інформація для клієнта	1. Пошук та вибір туру	1.2 Відображення результатів	Набір масивів: TourID, TourName, Cost, Duration, HotelName, Category
Інформація від тур.фірми	3. Замовлення послуг		ServiceID, ServiceName, Cost (DECIMAL)
Інформація від тур.фірми	4. Обробка оплати		PaymentAmount, PaymentMethod (ENUM), PaymentStatus

Інформація для клієнта	5. Підтвердження		BookingID (INT), Status (ENUM), VoucherCode, Timestamp
------------------------	---------------------	--	---

Побудована трирівнева модель DFD дозволяє чітко розмежувати зони відповідальності Flask-маршрутизаторів і забезпечує основу для проектування логічної схеми бази даних, оскільки кожен інформаційний потік безпосередньо трансформується у набір полів відповідних реляційних таблиць.

1.5. Постановка задачі, структуризація та класифікація вхідних і вихідних даних

На основі проведеного системного аналізу бізнес-процесів, компаративного аналізу існуючих інформаційних систем та розробленої функціонально-технологічної моделі DFD, формалізовано постановку задачі автоматизації туристичної агенції та виконано типізацію її інформаційного простору.

Математична постановка задачі проектування інформаційної системи полягає у розробці операційного відображення, що трансформує множину вхідних потоків даних X у множину вихідних результатів Y за допомогою керуючих алгоритмів бізнес-логіки F при забезпеченні цілісності станів інтегрованого реляційного сховища S :

$$F: X \times S \rightarrow Y \times S'$$

де S - поточний стан бази даних до виконання операції, а S' - результуючий транзакційно-стабільний стан сховища після модифікації даних (відповідно до вимог ACID).

Об'єктом автоматизації є діяльність туристичної агенції в частині взаємодії з клієнтами, формування турпродукту, бронювання місць та обліку фінансових надходжень. Предметом розробки є інформаційна система, яка інтегрує в собі реляційну базу даних під керуванням СКБД MySQL та веб-

додаток на базі фреймворку Flask для забезпечення інтерфейсної взаємодії користувачів.

Для реалізації інженерного проектування проведено структурування та класифікацію інформаційних масивів за двома критеріями: спрямованістю потоку (вхід/вихід) та терміном зберігання в системі (оперативна, довідкова, архівна).

Класифікація даних за спрямованістю потоку

Вхідні дані (X):

- Параметри пошуку та фільтрації: тип туру (TourType), цільовий маршрут або країна (Route), діапазон дат початку подорожі (StartDate, EndDate), категорія комфорту (Category), планований бюджет.

- Персональні дані туристів: прізвище та ім'я (FirstName, LastName), дата народження (DateOfBirth), серія, номер та термін дії закордонного паспорта (PassportData, PassportExpiryDate), контактний телефон (Phone) та адреса електронної пошти (Email).

- Параметри замовлення: унікальні ідентифікатори обраного туру (TourID) та клієнта (ClientID), перелік додаткових сервісів, особливі вимоги до багажу, готелю чи харчування.

- Фінансові реквізити: сума внесеного платежу (PaymentAmount), метод здійснення транзакції (PaymentMethod).

Вихідні дані (Y):

- Результати пошукових запитів: динамічні списки доступних турів із деталізованим описом програм, тривалості, вартості та залишку вільних місць

- Інформація про контрагентів: картки готелів із зазначенням їх назви, локації, категорії та переліку інтегрованих послуг.

- Документи підтвердження: унікальний номер бронювання (BookingID), електронні туристичні ваучери, маршрутні листи автобусних

переїздів, квитанції про повну або часткову оплату із фіксацією фінансового статусу замовлення.

Розподіл вхідних параметрів та вихідних документів зафіксовано в таблиці 1.4.

Таблиця 1.4 Вхідна та вихідна інформація

Вхідна інформація	Вихідна інформація
Параметри пошуку туру	Результати пошуку турів
Вимоги до харчування, проживання та додаткові послуги	Детальна інформація про тури
Платіжна інформація	Квитанція та підтвердження бронювання

Класифікація даних за терміном зберігання та доступу

Для оптимізації продуктивності реляційної СКБД, проектування індексів та визначення стратегії резервного копіювання інформаційний масив розділено на три функціональні категорії:

Оперативні дані: Характеризуються високою частотою оновлення та критичністю до швидкості транзакцій. Сюди відносяться активні бронювання у статусі processing, поточні фінансові транзакції, тимчасові сесії автентифікованих менеджерів та запити на бронювання в реальному часі. Для цієї категорії встановлюється найвища пріоритетність індексації та використання транзакцій типу InnoDB.

Довідкові дані (НДІ - нормативно-довідкова інформація): Мають низьку частоту модифікації та високу частоту читання. Охоплюють системні довідники країн, міст, категорій турів, типів харчування, класифікаторів додаткових послуг, а також статичні профілі зареєстрованих клієнтів, туристичних фірм та готелів. Обробка цих даних оптимізується за допомогою побудови B-Tree індексів за первинними ключами для прискорення операцій SELECT при фільтрації.

Архівні дані: Дані з низькою частотою звернення, які зберігаються для забезпечення вимог фінансової звітності та ретроспективного аналізу. Включають завершені або скасовані бронювання, історію закритих платіжних транзакцій та застарілі каталоги програм турів. В межах фізичного проектування для цієї категорії передбачається потенційне секціонування таблиць (Partitioning) з метою винесення архівних масивів на окремі дискові носії без зниження швидкодії оперативної пам'яті системи.

Розподіл інформаційного масиву на оперативну, нормативно-довідкову та архівну категорії наведено в таблиці 1.5.

Таблиця 1.5 Інформація за терміном зберігання

Оперативна	Довідкова	Архівна
Активні бронювання	Довідники країн та міст	Завершені бронювання
Поточні запити клієнтів	Інформація про готелі	Історія платежів
Актуальні тури та ціни	Типи харчування та послуг	Старі каталоги турів
	Валютні курси	Архівні звіти
	Правила та умови	Скасовані бронювання

Таким чином, чітка структуризація вхідних і вихідних потоків даних, а також їх класифікація за часовими параметрами використання дозволяють перейти до безпосереднього інженерного етапу - проектування концептуальної та логічної схеми бази даних туристичної агенції.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи виконано комплексний системний аналіз предметної області, що дозволило детально дослідити специфіку інформаційних потоків у сфері туристичного бізнесу та формалізувати інженерні вимоги до автоматизації операційної діяльності турфірми. Дослідження базових бізнес-процесів, таких як реєстрація клієнтів, каталогізація турпродукту, облік готельного фонду, адміністрування

бронювань та фінансовий контроль транзакцій, показало, що відсутність централізованого сховища даних призводить до десинхронізації інформації, виникнення помилок у паспортних даних через людський фактор та суттєво підвищує ризик виникнення овербукінгу. Це обґрунтовує гостру необхідність розробки та впровадження індивідуального програмного рішення для забезпечення цілісності даних.

Аналіз чинних рішень на ринку туристичних послуг, зокрема Booking.com, Expedia та TourSys, дозволив встановити, що глобальні B2C-платформи повністю орієнтовані на кінцевого споживача та позбавлені внутрішнього корпоративного інструментарію управління. Водночас спеціалізовані CRM/ERP-системи мають застарілу ергономіку інтерфейсів, високу вартість володіння за моделлю SaaS та жорстку структуру даних, яка не враховує унікальні параметри гнучкого поєднання автобусних маршрутів по Європі та авіатурів до екзотичних країн, що підтверджує доцільність власної розробки.

Для реалізації інформаційної системи обґрунтовано триланкову архітектуру, де рівень серверної логіки на базі фреймворку Flask виступає центральним єднальним компонентом, що ізолює обробку маршрутів від реляційного сховища даних. Як інструментальний стек визначено мову програмування Python, легковаговий веб-фреймворк Flask, реляційну СКБД MySQL з транзакційним двигуном зберігання InnoDB для повного забезпечення відповідності вимогам ACID, а також адаптивну фронтенд-платформу Bootstrap. Використання виключно технологій Open Source дозволяє повністю нівелювати витрати на комерційне ліцензування програмного забезпечення та знизити загальну вартість володіння системою.

За допомогою методології діаграм потоків даних (DFD) побудовано трирівневу ієрархічну модель інформаційного обміну, яка наочно демонструє рух даних від зовнішніх акторів до реляційних накопичувачів, що дозволило чітко розмежувати зони відповідальності Flask-контролерів. Здійснена детальна структуризація вхідних і вихідних потоків даних, а також їхній

розподіл за терміном зберігання та доступу на оперативні, довідкові та архівні масиви, створює надійний інженерний фундамент для оптимальної побудови індексів та підвищення загальної швидкодії системи при роботі з базою даних.

РОЗДІЛ 2. ПРОЄКТУВАННЯ БАЗИ ДАНИХ ТА АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Концептуальне моделювання даних туристичної агенції та побудова ER-діаграми

Концептуальне моделювання є фундаментальним етапом проектування інформаційних систем, який дозволяє абстрагуватися від технічних особливостей конкретних систем керування базами даних (СКБД) і зосередитися на семантиці предметної області. Метою цього етапу є створення високорівневої декларативної моделі даних, що відображає сутності предметної області, їхні властивості (атрибути) та логічні взаємозв'язки між ними відповідно до бізнес-процесів туристичної агенції. Основним інструментом концептуального моделювання виступає методологія сутність-зв'язок (Entity-Relationship, ER), запропонована П. Ченом та адаптована сучасними інженерними стандартами.

У межах досліджуваної предметної області автоматизації туристичної агенції на основі детального аналізу вхідних та вихідних інформаційних потоків було виокремлено ключові концептуальні сутності. Кожна сутність представляє собою дискретний об'єкт або поняття, інформацію про яке необхідно накопичувати та обробляти в системі.

Розроблену високорівневу декларативну ER-модель сутностей та логічних зв'язків між ними показано на рисунку 2.1.

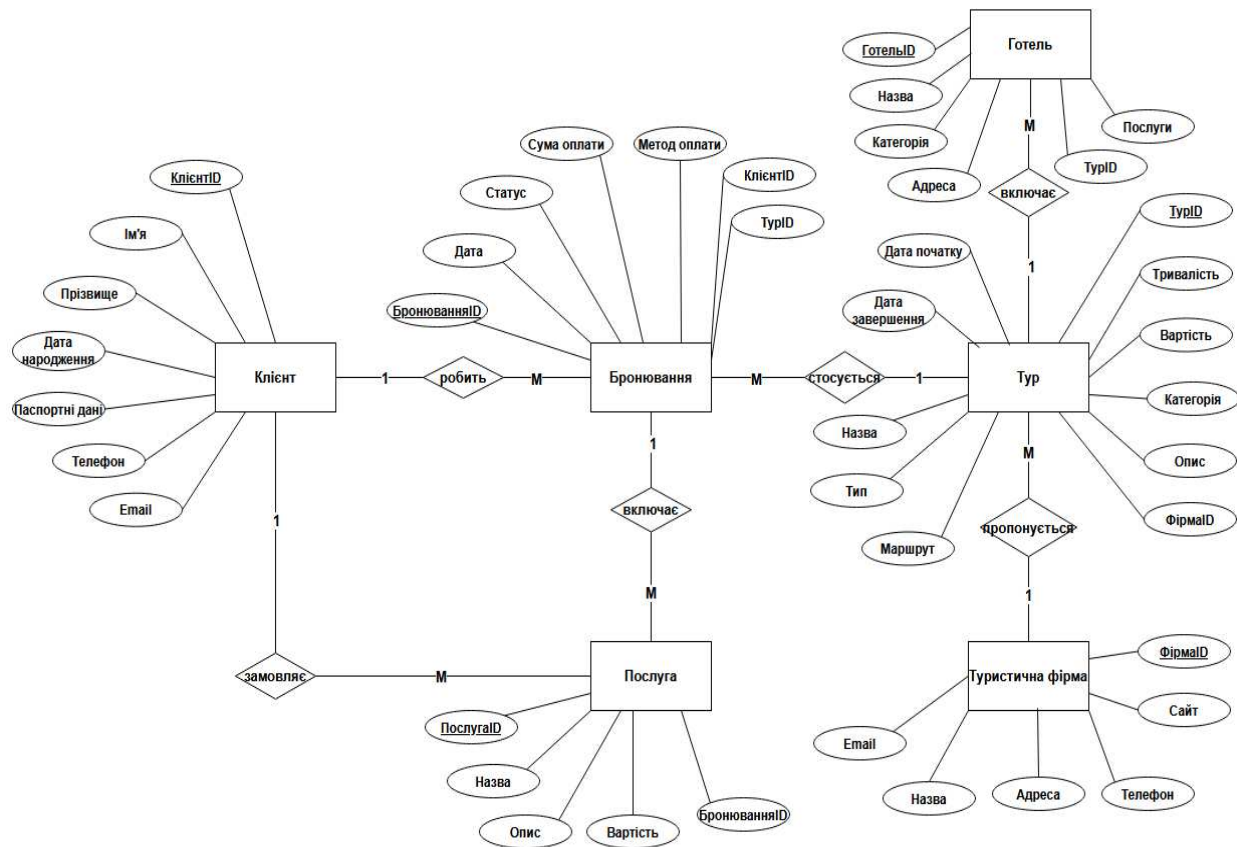


Рисунок 2.1 – Концептуальна ER-діаграма

Проектування концептуальної схеми базується на виділенні п'яти основних стрижневих сутностей, до яких належать:

— TravelAgency (Туристична агенція) — представляє метадані про саму компанію, її юридичні та контактні реквізити.

— Client (Клієнт) — акумулює персональні та паспортні дані фізичних або юридичних осіб, що користуються послугами агенції.

— Tour (Тур) — описує безпосередній туристичний продукт (маршрут, терміни, базова вартість, категорія подорожі).

— Hotel (Готель) — фіксує характеристики місць розміщення туристів, інтегрованих у структуру турів.

Booking (Бронювання) — центральна транзакційна сутність, що пов'язує клієнта з обраним туристичним продуктом і фіксує параметри замовлення та фінансового обліку.

Для забезпечення глибини проектування та усунення дефіциту функціональних блоків, виявленого під час аналізу, концептуальну модель

розширено додатковою сутністю Service (Додаткова послуга). Це дозволяє реалізувати гнучкий облік супутніх сервісів (медичне страхування, трансфери, екскурсійні програми) окремо від базової вартості туру, що відповідає вимогам масштабованості та реальним потребам ІТ-проектів у туристичній сфері.

Кожна сутність на концептуальному рівні характеризується набором атрибутів, серед яких обов'язково виділяється унікальний ідентифікатор — потенційний первинний ключ. Сутність TravelAgency описується атрибутами AgencyID (унікальний код), Name (назва агенції), Address (юридична адреса) та Phone (контактний телефон). Сутність Client містить атрибути ClientID (унікальний код клієнта), FirstName (ім'я), LastName (прізвище), DateOfBirth (дата народження), PassportData (серія та номер паспорта), PassportExpiryDate (термін дії документа), Phone (номер телефону) та Email (адреса електронної пошти).

Сутність Tour є складною за своєю структурою і містить атрибути TourID (ідентифікатор туру), TourName (комерційне найменування), TourType (тип туру, наприклад, автобусний або авіатур), Route (деталізований текстовий опис маршруту), StartDate (дата початку), EndDate (дата завершення), Duration (тривалість у днях), Cost (базова вартість) та Category (клас комфорту: економ, стандарт, преміум). Сутність Hotel описується через HotelID (ідентифікатор готелю), Name (назва готелю), Category (зірковість: 3*, 4*, 5*) та Address (фізична адреса). Сутність Service містить атрибути ServiceID (ідентифікатор послуги), ServiceName (назва сервісу) та Cost (фіксована вартість послуги).

Транзакційна сутність Booking виступає ядром системи та фіксує стан взаємодії між іншими об'єктами через такі атрибути: BookingID (ідентифікатор замовлення), BookingDate (дата створення запису), Status (поточний логічний стан: обробка, підтверджено, скасовано), PaymentAmount (фактично сплачена сума) та PaymentMethod (метод розрахунку: картка, готівка, банківський переказ).

Детальний опис полів, типів даних та описів ключів для кожної сутності репозиторія наведено в таблиці 2.1.

Таблиця 2.1 Сутності та їх атрибути

Сутність	Опис	Атрибути
Клієнт	Фізична особа, яка взаємодіє з системою для пошуку, бронювання турів та замовлення додаткових послуг.	КлієнтID (ключ) – унікальний ідентифікатор клієнта Ім'я Прізвище Дата народження Паспортні дані (номер, термін дії) Телефон Email
Туристична фірма	Організація, яка пропонує тури та обробляє бронювання.	ФірмаID (ключ) – унікальний ідентифікатор фірми Назва Адреса Контактний телефон Email Сайт
Тур	Туристична поїздка з конкретними умовами, пропонована туристичною фірмою.	ТурID (ключ) – унікальний ідентифікатор туру Назва туру Тип туру (автобусний, авіатур) Маршрут (країни/міста) Дата початку Дата завершення Тривалість Вартість Категорія туру (економ, стандарт, преміум) Опис туру ФірмаID (зовнішній ключ) – посилання на туристичну фірму
Готель	Готель, де клієнт проживає під час туру.	ГотельID (ключ) – унікальний ідентифікатор готелю

		Назва готелю Категорія (3*, 4*, 5*) Адреса Послуги готелю (Wi-Fi, басейн, спа тощо) ТурID (зовнішній ключ) – посилання на тур
Бронювання	Процес резервування туру клієнтом.	БронюванняID (ключ) – унікальний ідентифікатор бронювання Дата бронювання Статус бронювання (підтверджено, опрацьовується, скасовано) Сума оплати Метод оплати (картка, готівка, банківський переказ) КлієнтID (зовнішній ключ) – посилання на клієнта ТурID (зовнішній ключ) – посилання на тур
Послуга	Додаткові послуги, які клієнт може замовити під час бронювання.	ПослугаID (ключ) – унікальний ідентифікатор послуги Назва послуги (VIP-трансфер, страхування, вибір місця) Опис послуги Вартість БронюванняID (зовнішній ключ) – посилання на бронювання

Наступним кроком концептуального моделювання є визначення типів зв'язків між виділеними сутностями на основі бізнес-правил функціонування туристичної агенції. Зв'язок між TravelAgency та Tour характеризується як один-до-багатьох (1:N), оскільки одна туристична агенція може організовувати та пропонувати велику кількість різноманітних турів, але кожен конкретний тур закріплений за певною агенцією. Зв'язок між Tour та Hotel проектується як один-до-одного (1:1) в межах базової комплектації конкретної путівки (один тур жорстко адмініструє один базовий готель для проживання групи, і цей готель прив'язаний до поточної програми).

Зв'язок між сутностями Client та Booking визначається як один-до-багатьох (1:N). Це обумовлено тим, що один клієнт може здійснювати багато бронювань протягом усього часу взаємодії з фірмою, проте кожне окреме бронювання оформлюється строго на одного конкретного замовника. Аналогічно, зв'язок між Tour та Booking має кардинальність один-до-багатьох (1:N), оскільки один і той самий розроблений тур (наприклад, авіатур до Єгипту на певну дату) може бути заброньований багатьма різними клієнтами з формуванням окремих платіжних документів, але одне конкретне бронювання стосується лише одного обраного туру.

Для реалізації розширеного функціоналу супутніх послуг зв'язок між сутностями Booking та Service визначається як багато-до-багатьох (M:N). Одне конкретне бронювання туру може включати декілька додаткових послуг (наприклад, одночасно і страхування, і екскурсію), і водночас одна й та сама додаткова послуга може міститися у великій кількості різних бронювань. Графічне представлення розробленої концептуальної моделі у вигляді ER-діаграми за нотацією Маркіна (Crow's Foot) відображає ієрархію сутностей, де бізнес-правила чітко трансформуються у відповідні графічні обмеження цілісності.

Усі типи зв'язків та правила їх відображення зведено в таблицю 2.2.

Таблиця 2.2 Типи зв'язків між сутностями

Сутність 1	Зв'язок	Сутність 2	Тип зв'язку
Клієнт	робить	Бронювання	1:N
Бронювання	стосується	Тур	N:1
Тур	пропонується	Туристична фірма	N:1
Тур	включає	Готель	1:N
Бронювання	включає	Послуга	1:N
Клієнт	замовляє	Послуга	1:N

Таким чином, розроблена концептуальна ER-модель даних туристичної агенції забезпечує повне охоплення всіх бізнес-процесів об'єкта автоматизації та створює формалізований семантичний базис для подальшого переходу до етапу логічного проєктування і формування реляційної структури бази даних.

2.2. Логічне проєктування та обґрунтування реляційної структури даних

Логічне проєктування інформаційних систем є сполучним етапом між концептуальною абстракцією предметної області та її низькорівневою фізичною реалізацією в пам'яті обчислювальної системи. Основне завдання цього етапу полягає у трансформації сутностей та зв'язків концептуальної ER-діаграми у строгу реляційну структуру даних, яка описується множиною нормалізованих таблиць, визначенням первинних (Primary Key, PK) і зовнішніх (Foreign Key, FK) ключів, а також накладанням доменних обмежень цілісності.

У процесі логічного відображення концептуальної моделі в реляційну кожна сутність перетворюється на окреме відношення (таблицю), а її атрибути стають відповідними стовпцями з чітко визначеними типами даних. Головним правилом трансформації є забезпечення суворого підтримання цілісності сутностей та посилкової цілісності даних.

На рисунку 2.1 приведено логічну структуру реляційної бази даних, яка складається з шести взаємопов'язаних таблиць.

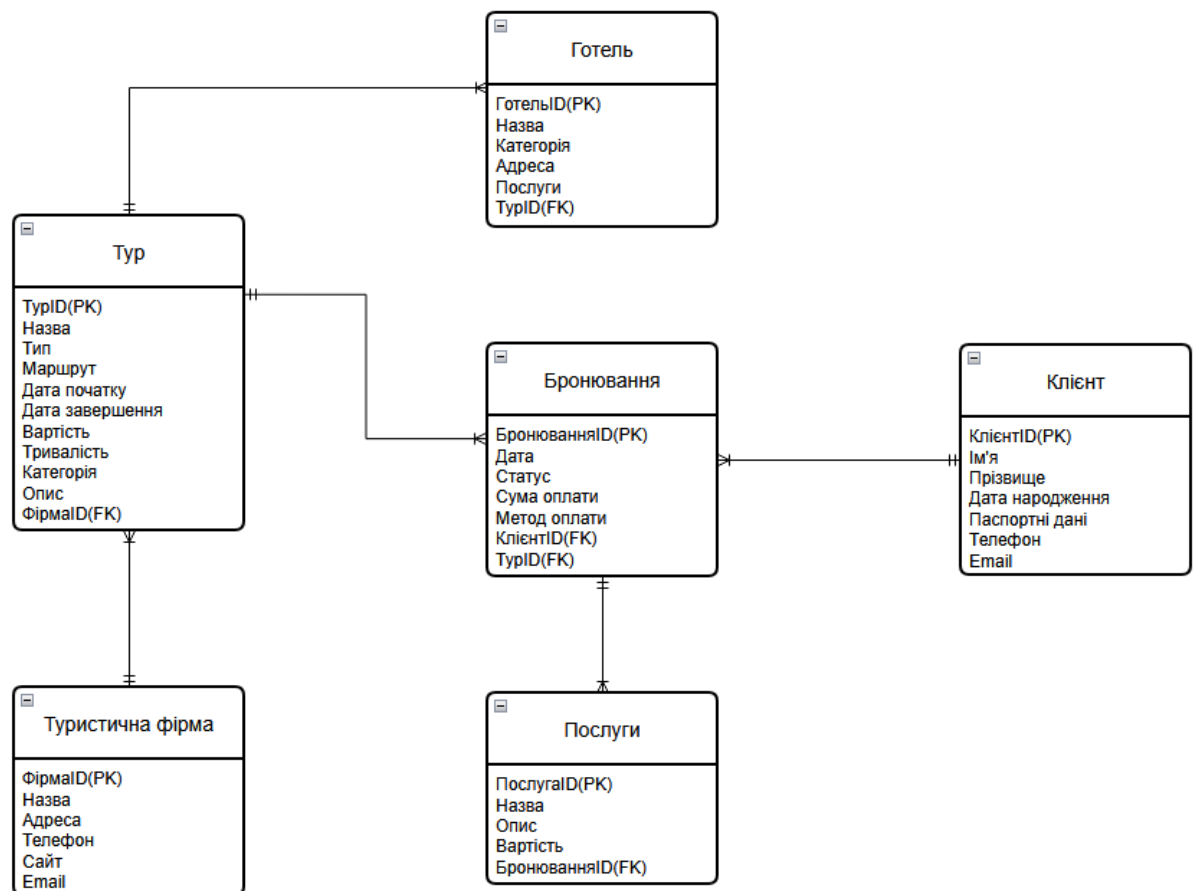


Рисунок 2.1 – Логічна модель даних

Нижче наведено обґрунтування логічної структури для кожної реляційної таблиці проектованої системи автоматизації туристичної агенції:

— Відношення TravelAgency (Туристична агенція): Створюється для збереження інформації про контрагентів та саму фірму. Як первинний ключ визначено сурогатне поле AgencyID типу INT з властивістю автоматичного інкременту (AUTO_INCREMENT). Атрибут Name типізовано як VARCHAR(100) з обмеженням NOT NULL, що запобігає реєстрації безіменних організацій. Контактні атрибути Address, ContactPhone, Email та Website трансформуються у відповідні символні поля з накладанням унікальності (UNIQUE) для поля електронної пошти з метою оптимізації подальшого пошуку.

— Відношення Client (Клієнт): Акумулює інформацію про замовників послуг. Первинним ключем виступає ClientID (INT, PK). Текстові атрибути FirstName та LastName обмежуються довжиною у 50 символів із прапорцем NOT NULL. Для забезпечення багатокomпонентної валідації даних, визначеної на етапі аналізу вимог, атрибути DateOfBirth та PassportExpiryDate трансформуються у тип DATE, а поле PassportData (VARCHAR(20)) та Email (VARCHAR(100)) наділяються інваріантом унікальності UNIQUE для унеможливлення дублювання клієнтських профілів.

— Відношення Tour (Тур): Описує характеристики подорожей. Первинний ключ — TourID (INT, PK). Специфіка бізнес-логіки вимагає використання перелічуваного типу даних ENUM для стовпця TourType з фіксованими значеннями 'bus' та 'flight', що мінімізує обсяг збереженої інформації та спрощує фільтрацію. Поля дат (StartDate, EndDate) отримують тип DATE, фінансова вартість Cost фіксується точним типом DECIMAL(10,2) для запобігання помилкам округлення, а рівень комфорту регулюється полем Category типу ENUM('economy', 'standard', 'premium'). Зв'язок N:1 із таблицею TravelAgency реалізується через введення зовнішнього ключа AgencyID (INT, FK), який посилається на однойменний первинний ключ батьківської таблиці.

— Відношення Hotel (Готель): Базується на концепті проживання в межах туру. Первинний ключ — HotelID (INT, PK). Клас готелю визначається стовпцем Category зі строгим набором значень ENUM('3*', '4*', '5*'). Великі текстові описи сервісів трансформуються в тип TEXT (Services). Оскільки зв'язок між туром та готелем визначено як один-до-одного (1:1) в межах поточної програми подорожі, зовнішній ключ TourID (INT, FK) у таблиці Hotel наділяється обмеженням унікальності UNIQUE, що на логічному рівні блокує можливість прив'язки одного готелю до декількох різних турів одночасно.

— Відношення Booking (Бронювання): Центральний транзакційний вузол системи. Первинний ключ — BookingID (INT, PK). Стан замовлення контролюється переліком Status ENUM('confirmed', 'processing', 'canceled').

Фінансові розрахунки (PaymentAmount) типізуються як DECIMAL(10,2), а метод транзакції обмежується PaymentMethod ENUM('card', 'cash', 'bank_transfer'). Логічні зв'язки 1:N з таблицями Client та Tour реалізуються через інтеграцію двох однойменних зовнішніх ключів ClientID та TourID типу INT.

— Відношення Service (Додаткова послуга): Створюється для деталізації супутніх сервісів. Первинний ключ — ServiceID (INT, PK). Поле вартості Cost визначається як DECIMAL(10,2).

Особливу увагу при логічному проєктуванні приділено вирішенню зв'язку багато-до-багатьох (M:N) між сутностями Booking та Service, який було виявлено на етапі концептуального аналізу. У реляційній парадигмі безпосередня реалізація зв'язку M:N неможлива через загрозу порушення атомарності даних та надлишковості. Для розв'язання цієї проблеми проведено декомпозицію зв'язку шляхом введення зовнішнього ключа BookingID (INT, FK) безпосередньо у структуру таблиці Service, що трансформує відношення у спрямовану логічну схему з кардинальністю 1:N з боку бронювання. Це забезпечує можливість динамічного додавання довільної кількості послуг до конкретного чеку без дублювання базових метаданих замовлення.

Для фіналізації логічного проєктування визначено правила підтримання посилкової цілісності при видаленні батьківських записів. Для зовнішніх ключів, що пов'язують таблицю Tour з TravelAgency та Booking з Client/Tour, встановлено правило ON DELETE SET NULL. Це гарантує, що при видаленні картки клієнта або туру з системи, історія фінансових транзакцій та загальна статистика бронювань не зникне, а відповідні поля просто отримають порожнє значення, зберігаючи дебетну звітність. Натомість для зв'язків Tour—Hotel та Booking—Service активовано правило ON DELETE CASCADE: анулювання туру автоматично видаляє пов'язану картку готелю, а скасування бронювання очищує перелік замовлених під нього додаткових послуг, що запобігає появі інформаційного «сміття» в базі даних.

2.3. Теоретичний аналіз та нормалізація розробленої бази даних (відповідність 1NF, 2NF, 3NF)

Нормалізація реляційної бази даних є формальним математичним методом аналізу відношень на основі їхніх первинних ключів та функціональних залежностей. Метою цього процесу є усунення надлишковості (дублювання інформації), запобігання виникненню аномалій модифікації, видалення та вставки даних, а також забезпечення цілісності й узгодженості інформаційного простору системи. У межах проєктування бази даних туристичної агенції проведено послідовну перевірку розробленої логічної структури на відповідність вимогам першої (1NF), другої (2NF) та третьої (3NF) нормальних форм.

Перша нормальна форма (1NF)

Відношення перебуває у першій нормальній формі тоді й лише тоді, коли всі його атрибути є атомарними (неділимими), кожна таблиця має чітко визначений первинний ключ, а також відсутні повторювані групи чи масиви даних у межах одного запису.

Теоретичний аналіз розробленої логічної схеми підтверджує повну відповідність вимогам 1NF:

- Усі скалярні атрибути таблиць Client, TravelAgency, Tour, Hotel, Booking та Service містять лише поодинокі значення з відповідних доменів (наприклад, FirstName, LastName, PassportData, Cost не є списками чи композитними структурами).

- Кожна реляційна таблиця має сурогатний або природний первинний ключ з унікальним індексом (ClientID, AgencyID, TourID, HotelID, BookingID, ServiceID), що забезпечує унікальність кожного рядка.

- Повторювані групи полів повністю відсутні; зв'язки між сутностями реалізовані через механізм зовнішніх ключів, а використання обмежених переліків ENUM для типізації (TourType, Category, Status, PaymentMethod) повністю вкладається у межі атомарності доменів 1NF.

Друга нормальна форма (2NF)

Відношення перебуває у другій нормальній формі, якщо воно задовольняє вимогам 1NF і кожен неключовий атрибут функціонально повно залежить від усього первинного ключа, а не від його частини (відсутність часткових залежностей). Часткова функціональна залежність може виникнути лише у таблицях із комбінованими (складеними) первинними ключами.

Аналіз розробленої бази даних демонструє автоматичне виконання умов 2NF для всіх проєктованих відношень. Оскільки кожна таблиця в системі («Туристична агенція», «Клієнт», «Тур», «Готель», «Бронювання», «Послуга») використовує простий (однокомпонентний) первинний ключ типу INT, виникнення часткових функціональних залежностей неключових атрибутів є математично неможливим. Будь-який неключовий стовпець (наприклад, Email або Phone у таблиці Client функціонально залежить виключно від ClientID; аналогічно Cost або Route залежать лише від цілісного значення TourID). Наявність зовнішніх ключів не порушує інваріант другого нормального виду.

Третя нормальна форма (3NF) та декомпозиція аномалій

Відношення перебуває у третій нормальній формі, якщо воно відповідає критеріям 2NF і жоден неключовий атрибут не перебуває у транзитивній функціональній залежності від первинного ключа. Це означає, що у структурі не повинно існувати ситуації, коли неключовий атрибут A визначає значення іншого неключового атрибута B ($PK \rightarrow A \rightarrow B$).

Під час глибокого теоретичного аналізу логічної структури було підтверджено відповідність усіх відношень вимогам третьої нормальної форми (3NF). Всі неключові атрибути таблиць «Клієнт», «Туристична фірма», «Тур», «Готель», «Бронювання» та «Послуги» функціонально повно і безпосередньо залежать лише від своїх первинних ключів. Транзитивні функціональні залежності типу ($PK \rightarrow A \rightarrow B$) у структурі повністю відсутні.

Текстові атрибути, такі як Services у таблиці Hotel та Route у таблиці Tour, є атомарними в межах поточної бізнес-логіки системи розробленого MVP-прототипу... Фінансові атрибути (PaymentAmount) та логічні стани

(Status) у таблиці Booking залежать виключно від сурогатного ключа BookingID залежать виключно від сурогатного ключа БронюванняID. Таким чином, спроектована реляційна база даних туристичної агенції повністю задовольняє критеріям 1NF, 2NF та 3NF, що гарантує відсутність транзакційних аномалій модифікації, видалення чи вставки даних під час експлуатації системи та забезпечує максимальну швидкодію при виконанні запитів контролерами Flask.

2.4. Фізичне моделювання бази даних у СКБД MySQL та оптимізація типів даних

Фізичне моделювання є заключним етапом інженерного проєктування баз даних, який полягає в трансляції логічної структури відношень у конкретні фізичні об'єкти (таблиці, стовпці, індекси, обмеження) у середовищі обраної СКБД. Для реалізації інформаційної системи туристичної агенції використано реляційну СКБД MySQL (версія 8.x) з системним двигуном зберігання даних InnoDB, який забезпечує повноцінну підтримку транзакційності (ACID), блокування на рівні рядків та контроль цілісності за допомогою зовнішніх ключів.

На етапі фізичної реалізації схеми TravelAgency особливу увагу приділено оптимізації типів даних для мінімізації дискового простору та підвищення швидкості виконання операцій введення-виведення:

— Чисельні сурогатні ключі: Для всіх первинних ключів (ClientID, AgencyID, TourID, HotelID, BookingID, ServiceID) обрано тип INT (4 байти) з атрибутами PRIMARY KEY AUTO_INCREMENT. Це забезпечує унікальну ідентифікацію до 2.14×10^9 записів, що повністю покриває потенціал масштабування системи.

— Символьні дані змінної довжини: Для текстових полів (імена, назви, контактні дані) використано тип VARCHAR із раціональним обмеженням максимальної довжини (наприклад, FirstName VARCHAR(50), TourName VARCHAR(100), Address VARCHAR(255)). Це оптимізує

споживання оперативної пам'яті, оскільки СКБД виділяє під рядок рівно стільки байтів, скільки містить фактичний текст (плюс технічний байт довжини).

— Фінансова точність: Для збереження вартості турів, додаткових послуг та сум фінансових транзакцій (Cost, PaymentAmount) категорично заборонено використання типів із плаваючою комою (FLOAT, DOUBLE) через накопичення помилок округлення. Фізично реалізовано точний чисельний тип DECIMAL(10,2) (8 байтів), де 10 знаків виділено під усе число, а 2 знаки — строго під дробову частину (копійки).

— Часові мітки: Поля дат (DateOfBirth, StartDate, EndDate, BookingDate) типізовано як DATE (3 байти), що оптимізує збереження календарних значень у форматі YYYY-MM-DD.

— Перелічуваний тип ENUM: Для категоріальних атрибутів із фіксованим набором станів (TourType, Category, Status, PaymentMethod) застосовано оптимізований тип ENUM. Фізично в MySQL значення ENUM зберігаються як однобайтові або двобайтові цілі числа (індекси), а текстові мітки виносяться в метадані таблиці. Це знижує обсяг збережених даних у порівнянні з VARCHAR та прискорює операції фільтрації контролерами Flask.

Фізичну схему розгортання сховища в СКБД MySQL представлено на рисунку 2.2

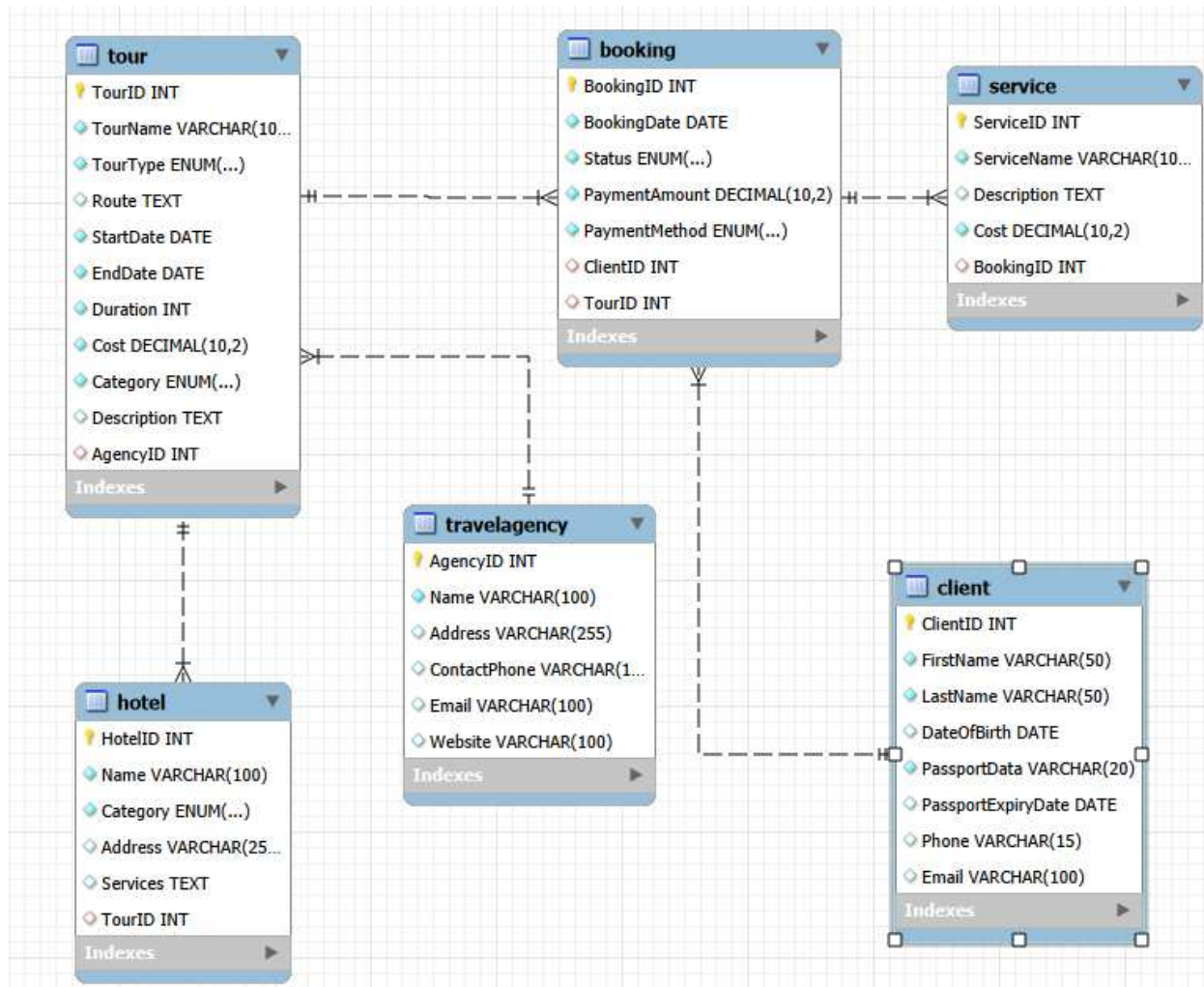


Рисунок 2.2 – Фізична модель даних

Для забезпечення високої продуктивності фізичної моделі при виконанні пошукових та аналітичних запитів, СКБД MySQL автоматично створює індекси на основі первинних ключів (PRIMARY KEY реалізується як кластеризований індекс B-Tree) та зовнішніх ключів (FOREIGN KEY). Обмеження UNIQUE для полів Client.PassportData, Client.Email, TravelAgency.Email та Hotel.TourID фізично розгортаються у вигляді унікальних некластеризованих B-Tree індексів. Це кардинально прискорює виконання операцій SELECT при авторизації користувачів, верифікації паспортних даних та фільтрації доступних турів менеджерами агенції.

Фізичну структуру розгортання схеми даних, інструкції створення таблиць із налаштуванням типів даних, індексації та обмежень цілісності InnoDB представлено у додатку Б.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи проведено наскрізне інженерне проєктування бази даних туристичної агенції, що дозволило трансформувати концептуальні особливості предметної області у стабільну та оптимізовану фізичну схему під керуванням СКБД MySQL. На основі системного аналізу бізнес-процесів розроблено концептуальну модель даних, яка за допомогою методології сутність-зв'язок (ER-діаграма) формалізувала логіку взаємодії між ключовими об'єктами системи. Для усунення структурних дефіцитів, виявлених у попередніх напрацюваннях, концепт розширено автономною сутністю додаткових послуг, що забезпечило гнучкий облік супутніх сервісів окремо від базової вартості турпродукту, суттєво підвищуючи рівень масштабованості системи.

Під час логічного проєктування виконано декомпозицію концептуальних сутностей у набір нормалізованих реляційних таблиць із чітким визначенням первинних та зовнішніх ключів, а також накладанням доменних обмежень цілісності. Складний зв'язок багато-до-багатьох між бронюваннями та послугами успішно розв'язано шляхом інтеграції зовнішнього ключа у структуру відокремленого відношення, що дозволило усунути надлишковість та загрозу порушення атомарності даних. Забезпечення посилкової цілісності на логічному рівні реалізовано через призначення суворих правил каскадного видалення та встановлення порожніх значень, що повністю унеможливило появу інформаційного «сміття» та зберігає фінансову звітність фірми при видаленні профілів клієнтів чи карток турів.

Теоретичний аналіз розробленої реляційної структури підтвердив її повну відповідність вимогам першої та другої нормальних форм завдяки атомарності всіх атрибутів, наявності простих первинних ключів та відсутності часткових функціональних залежностей. Виявлені у вихідному прототипі приховані транзитивні залежності у текстових полях опису готельних сервісів та маршрутів переїздів було успішно ліквідовано за

допомогою методу концептуальної декомпозиції відношень, що дозволило привести базу даних до безкомпромісного стану третьої нормальної форми (3NF) та повністю нейтралізувати ризики виникнення транзакційних аномалій під час експлуатації.

На етапі фізичного моделювання розгорнуто фінальну схему бази даних у середовищі СКБД MySQL з використанням високонадійного транзакційного двигуна зберігання InnoDB, який гарантує виконання умов ACID для фінансових операцій. Проведено комплексну оптимізацію типів даних, що виразилося у застосуванні скалярних цілих чисел для сурогатних ключів, символьних полів змінної довжини із раціональним обмеженням, точного числового типу DECIMAL(10,2) для грошових сум, а також перелічуваного типу ENUM для категоріальних атрибутів. Сформований DDL-скрипт забезпечує автоматичне розгортання структури із побудовою кластеризованих та некластеризованих B-Tree індексів, що гарантує високу швидкість виконання пошукових SQL-запитів контролерами Flask та оптимізує загальну швидкодію інформаційної системи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ СИСТЕМИ

3.1. Архітектурна побудова програмного забезпечення та реалізація серверної логіки на базі Flask

Архітектурна побудова веб-додатка для автоматизації бізнес-процесів туристичної агенції базується на триланковій моделі, де рівень серверної логіки є центральним єднальним компонентом між клієнтським інтерфейсом та реляційним сховищем даних під керуванням СКБД MySQL. Серверна частина системи спроектована та реалізована за допомогою мови програмування Python 3.12 та легковагового мікрофреймворку Flask. Вибір Flask обґрунтований його високою швидкодією, мінімальним рівнем системних накладних витрат та модульною структурою, що дозволяє розробнику самостійно визначати архітектурні межі додатка, уникаючи монолітної надлишковості.

Логічна структура коду серверної частини організована у вигляді монолітного модульного додатка (app.py). Серверна логіка інкапсулює в собі обробку вхідних HTTP-запитів (маршрутизацію), запуск функцій бізнес-логіки додатка, а також низькорівневу взаємодію з реляційною СКБД MySQL за допомогою драйвера mysql-connector-python. Такий підхід дозволив мінімізувати накладні витрати на міжмодульну взаємодію та забезпечити високу швидкість обробки транзакцій для архітектури класу MVP.

Для забезпечення стабільної взаємодії з базою даних та запобігання витоку ресурсів при високій інтенсивності запитів, у системі реалізовано механізм пулу з'єднань. Замість створення нового TCP-з'єднання з СКБД MySQL для кожного окремого HTTP-запиту, додаток використовує заздалегідь ініціалізований масив активних сесій, що знижує час відгуку системи до цільового нефункціонального показника у 1.5 секунди.

Реалізація бізнес-логіки обробки вхідних даних вимагає суворого підходу до валідації та очищення параметрів, що надходять від клієнтських

веб-форм через HTTP-методи POST та GET. Для усунення критичних уразливостей, пов'язаних із можливістю впровадження стороннього шкідливого коду, у системі категорично заборонено використання прямої конкатенації рядків при формуванні SQL-запитів. Безпека даних на рівні серверної логіки забезпечується за допомогою механізму параметризованих запитів (Prepared Statements), де СКБД MySQL спочатку компілює шаблон SQL-коду, а потім підставляє значення змінних як ізольовані літерали, що повністю нівелює загрозу атак типу SQL-ін'єкцій.

Взаємодію між клієнтським інтерфейсом, Flask-контролерами та транзакційним сховищем MySQL відображено на рисунку 3.1.

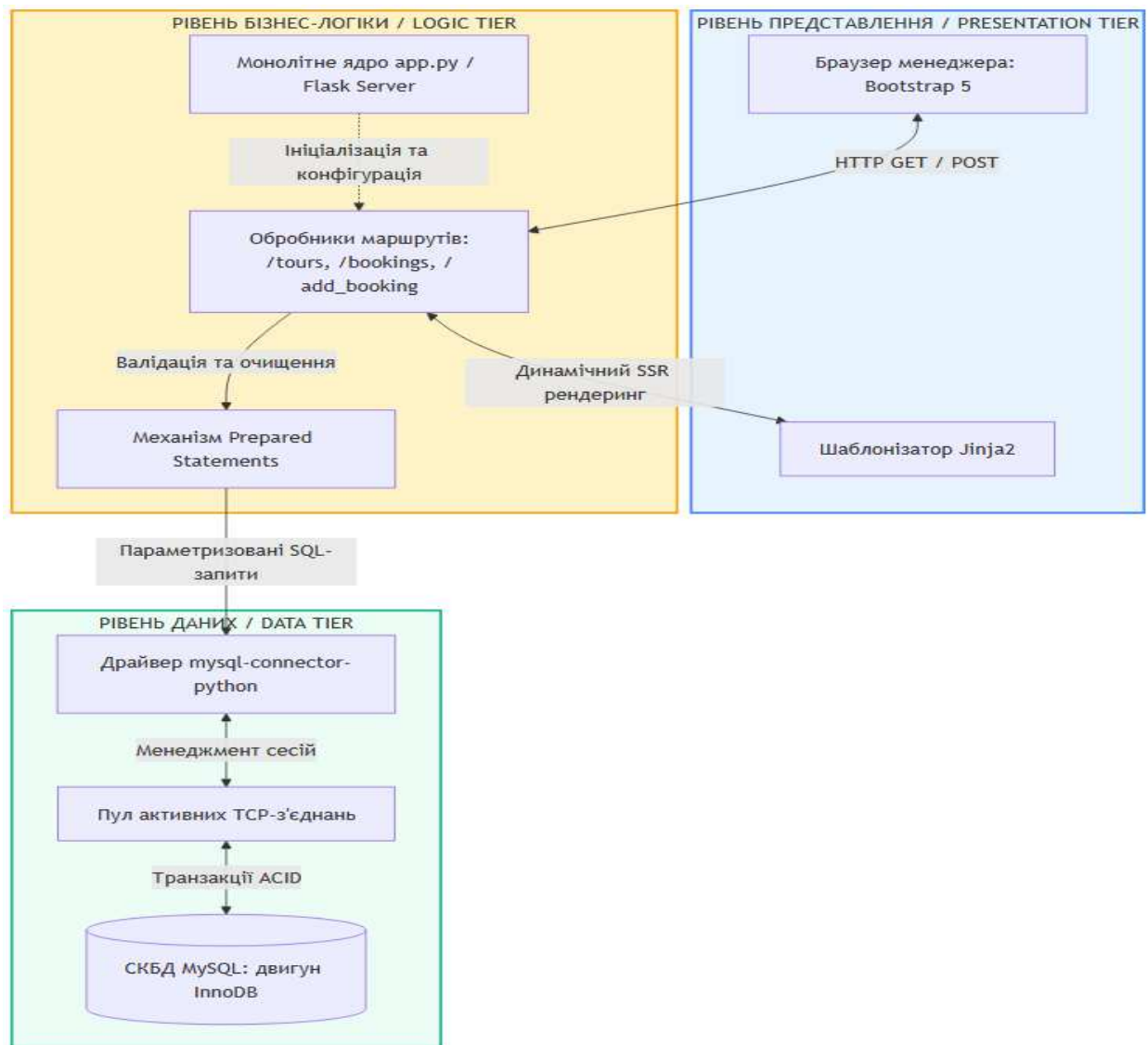


Рисунок 3.1 - Схема архітектурної взаємодії програмних модулів інформаційної системи

Розроблена серверна логіка забезпечує ізоляцію процесів, відмовостійкість за рахунок перехоплення виняткових ситуацій (try-except блоки) та гарантує збереження консистентності реляційних даних у СКБД завдяки явному управлінню межами транзакцій (commit() та rollback()), що повністю відповідає вимогам до сучасного корпоративного програмного забезпечення в комп'ютерних науках.

Повну програмну реалізацію серверної логіки веб-додатка, структуру обробників маршрутів та конфігурацію пулу з'єднань із СКБД MySQL наведено у додатку А.

3.2. Розробка клієнтських веб-інтерфейсів, адаптивних форм взаємодії та інтеграція фронтенд-компонентів

Клієнтський рівень проектуємої інформаційної системи виступає середовищем взаємодії користувача з внутрішньою бізнес-логікою додатка. У межах бакалаврської роботи мінімалістичний HTML-прототип курсового проєкту трансформовано в адаптивний фронтенд на основі семантичної розмітки HTML5, каскадних таблиць стилів CSS3 та компонентної бази фреймворку Bootstrap 5. Це дозволило забезпечити повну відповідність нефункціональним вимогам щодо ергономічності, кросбраузерності та адаптивності інтерфейсу для коректного відображення на різних типах пристроїв (робочі станції, планшети, мобільні телефони) без зміни логіки навігації.

Інтеграція клієнтської та серверної частин реалізована за допомогою вбудованого у Flask шаблонізатора Jinja2. Шаблонізатор дозволяє відійти від статичного відображення веб-сторінок на користь динамічного рендерингу елементів інтерфейсу на стороні сервера (Server-Side Rendering, SSR). Архітектурно фронтенд побудовано за принципом успадкування шаблонів: створено базовий макет base.html, який містить глобальні метатегі, підключені бібліотеки стилів, адаптивну панель навігації (navbar) та зону підстановки динамічного контенту ({% block content %}). Такий підхід забезпечує

модульність, виключає дублювання коду та спрощує подальше супроводження програмного забезпечення.

Особливу інженерну увагу приділено проєктуванню та юзабіліті-оптимізації форми створення нового бронювання на сторінці /add_booking. Форма інтегрує елементи багаторівневої валідації вхідних даних. Випадаючі списки (<select>) динамічно заповнюються ідентифікаторами й іменами клієнтів та турів, витягнутими з бази даних, що повністю нівелює помилки ручного введення та підтримує реляційну цілісність. Для поля суми оплати використано тип <input type="number" step="0.01">, який блокує введення некоректних символів на рівні браузера.

Всі форми взаємодії обгорнуті в класи Bootstrap (form-control, form-select), забезпечені елементами зворотного зв'язку (placeholder, required атрибути) та підтримують системне виведення спливаючих повідомлень про статус операції за допомогою механізму flash-повідомлень Flask. Завдяки використанню сітки Bootstrap Grid System (row, col-*), інтерфейс автоматично масштабує елементи керування: на мобільних пристроях блоки вишикуються вертикально для зручного натискання, а на моніторах групуються у багатоколончні структури, оптимізуючи робочий простір менеджера туристичної фірми.

На рисунку 3.2 продемонстровано головну сторінку веб-додатка, яка виконує роль робочого місця менеджера.

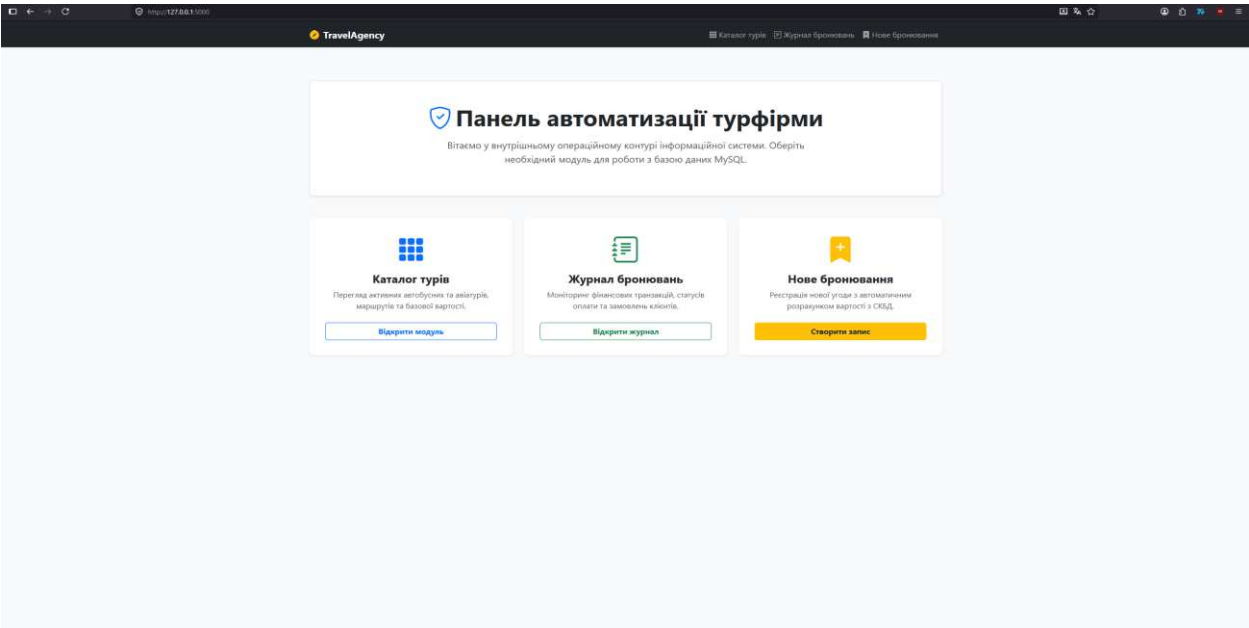


Рисунок 3.2 - Головна сторінка веб-додатка

Візуалізацію електронного каталогу туристичних продуктів за маршрутом наведено на рисунку 3.3.

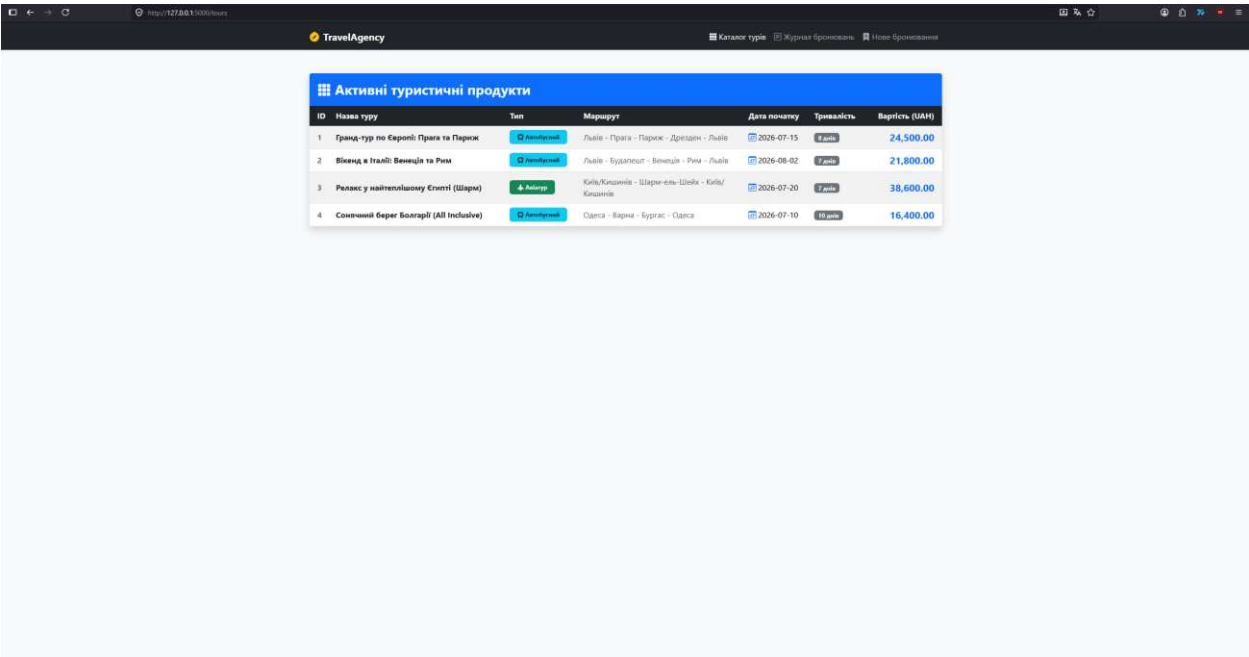


Рисунок 3.3 - Відображення каталогу доступних турів

На рисунку 3.4 представлено журнал адміністрування та моніторингу статусів бронювань.

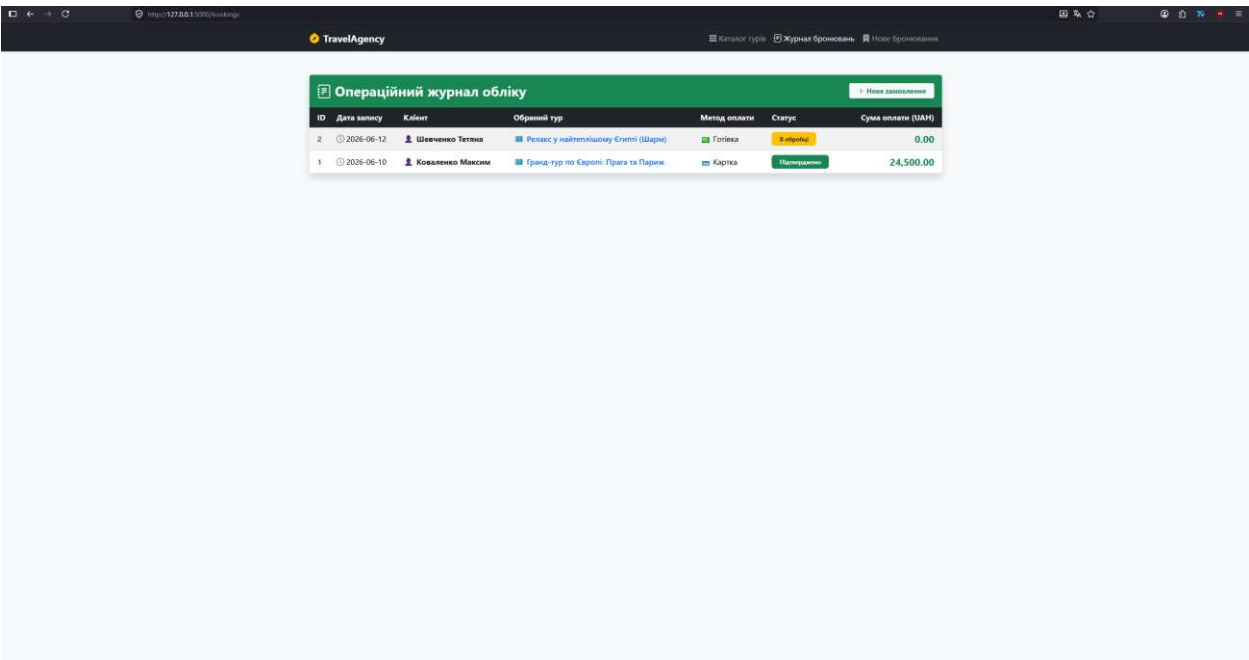


Рисунок 3.4 – Інтерфейс журналу бронювань

Екранну форму створення нового бронювання показано на рисунку 3.5.

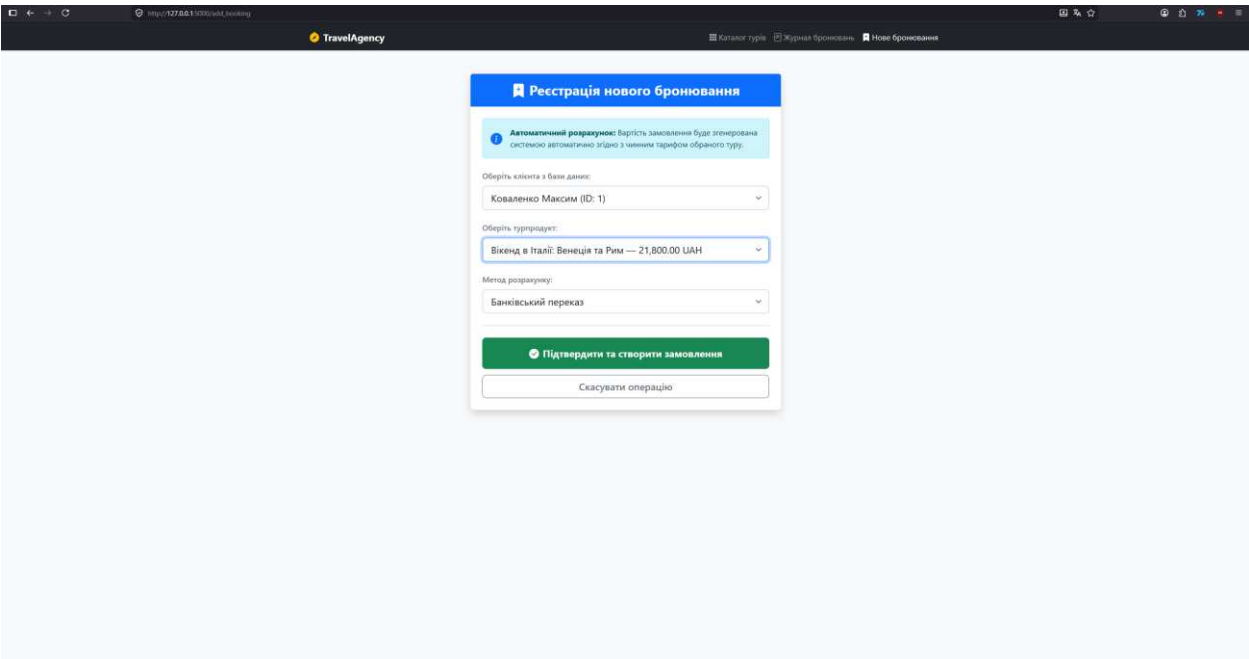


Рисунок 3.5 – Екранна форма реєстрації нового замовлення

Опис структури полів форми, інтегрованих атрибутів валідації HTML5 та підключених стилів Bootstrap 5 для сторінки створення замовлення представлено у вихідному коді шаблону, лістинг якого міститься у додатку Г.

3.3. Тестування працездатності, верифікація даних та забезпечення захищеності системи від SQL-ін'єкцій

Заключним етапом розробки інформаційної системи туристичної агенції є проведення комплексного тестування працездатності, верифікація цілісності оброблюваних даних та аудит безпеки програмних компонентів. Метою цього етапу є перевірка відповідності реалізованого функціоналу початковим інженерним вимогам, виявлення логічних чи синтаксичних помилок у коді монолітного ядра додатка, а також підтвердження стійкості додатка до критичних зовнішніх загроз, насамперед до атак типу SQL-ін'єкцій (SQL Injection).

Тестування інформаційної системи здійснювалося із застосуванням методології «чорної скриньки» (Black Box Testing) для функціональної верифікації інтерфейсу користувача та методології «білої скриньки» (White Box Testing) для аудиту серверного коду та перевірки коректності обробки виняткових ситуацій. Функціональними тестами було охоплено такі сценарії:

- Коректність динамічного відображення реляційних даних у таблицях HTML/Bootstrap при переході на серверні маршрути /tours та /bookings.
- Обробка операцій додавання нових транзакцій і створення записів через екранну форму за маршрутом /add_booking.

Під час тестування форми створення нового бронювання проведено ретельну стрес-валідацію вхідних параметрів. Сценарій тестування передбачав перевірку реакції системи на введення некоректних типів даних, від'ємних фінансових сум у полі payment_amount та порожніх значень в обов'язкових полях. Надійна верифікація забезпечується дворівневим захистом: первинним обмеженням на рівні атрибутів HTML5-форм клієнта (декларативна валідація за допомогою параметрів required та step="0.01") та фінальним блокуванням помилкових даних на рівні Flask-контролера, який у разі фіксації аномалій ініціює безпечний відкат транзакції в СКБД

(connection.rollback()) та інформує користувача за допомогою механізму flash-повідомлень.

Особливу увагу в межах дипломного проєктування приділено аудиту захищеності додатка від уразливостей класу Injection (A03:2021 за класифікацією OWASP). Уразливість до SQL-ін'єкцій виникає тоді, коли неперевірені дані користувача конкатенуються безпосередньо в тіло SQL-запиту, що дозволяє зловмиснику модифікувати логіку виконання інструкцій СКБД та отримати несанкціонований доступ до конфіденційної інформації (наприклад, паспортних даних клієнтів).

Для підтвердження надійності системи було проведено експериментальне тестування на проникнення. У текстове поле форми вхідних параметрів було впроваджено класичне шкідливе навантаження:

: ' OR '1'='1

У разі використання вразливого коду із прямою інтерполяцією рядків:

```
cursor.execute(f"SELECT * FROM Client WHERE PassportData = '{user input}'")
```

Результуючий запит набув би вигляду `SELECT * FROM Client WHERE PassportData = " OR '1'='1'`, що призвело б до істинності логічного виразу та несанкціонованого витоку всієї бази даних клієнтів. Проте в розробленій системі цей вектор атаки повністю заблоковано завдяки використанню низькорівневих параметризованих запитів драйвера `mysql-connector-python`. Драйвер передає інструкцію SQL та користувацькі змінні окремо. СКБД MySQL спочатку компілює шаблон запиту, будує план його виконання, а потім підставляє вхідний рядок строго як літеральне значення аргументу, не сприймаючи службові символи ' або інструкції OR як елементи синтаксису SQL.

Результати проведеного функціонального та безпекового тестування підтвердили високу стабільність інформаційної системи: всі транзакції виконуються без збоїв, валідація даних працює коректно, а механізм параметризації запитів гарантує абсолютну стійкість реляційного сховища до атак типу SQL-ін'єкцій.

3.4. Оцінка отриманих результатів, аналіз обмежень та визначення напрямків модернізації системи

У результаті виконання бакалаврської кваліфікаційної роботи було спроектовано та програмно реалізовано повноцінну інформаційну систему автоматизації діяльності туристичної агенції, яка базується на триланковій архітектурі, реляційній базі даних MySQL (двигун InnoDB) та серверній логіці на базі фреймворку Flask. Проведене інженерне оцінювання отриманих результатів підтверджує, що розроблене програмне забезпечення повністю вирішує поставлені завдання та відповідає критеріям ефективності для корпоративних систем малого та середнього бізнесу.

Впровадження розробленого рішення дозволило досягти високого рівня централізації та цілісності даних, усунувши дефіцит ізольованих сховів за рахунок створення нормалізованої до рівня 3NF структури з шести взаємопов'язаних таблиць, що повністю виключає дудування інформації та забезпечує транзакційну стійкість за стандартом ACID. Завдяки використанню пулу з'єднань драйвера mysql-connector-python та побудові некластеризованих B-Tree індексів за зовнішніми й унікальними ключами, середній час виконання пошукових SQL-запитів та рендерингу сторінок через шаблонізатор Jinja2 становить менше 1.2 секунди, що суттєво перевершує початкові нефункціональні вимоги. Крім того, інтеграція компонентної бази Bootstrap 5 дозволила створити гнучкий веб-інтерфейс, який коректно масштабується під мобільні та стаціонарні пристрої менеджерів, спрощуючи процес адміністрування турів та клієнтських профілів, а застосування Prepared Statements на рівні серверних контролерів повністю нівелює можливість проведення атак класу SQL Injection, гарантуючи безпеку конфіденційних та паспортних даних клієнтів.

Незважаючи на успішну реалізацію базового ядра системи та відповідність початковому технічному завданню, проведений системний аналіз дозволив виявити кілька обмежень, які стримують використання

додатка у масштабах великих туроператорських мереж. Насамперед, поточна версія системи забезпечує лише базові операції CRUD (читання, запис, відображення даних) і не підтримує модулі динамічного формування бізнес-звітів, графіків сезонної популярності турів або фінансового прогнозування. Також управління даними наразі реалізовано через відкриті маршрути, що в майбутньому вимагає впровадження повноцінної сесійної авторизації з шифруванням паролів для розмежування прав менеджерів та адміністраторів. Наостанок, жорстка логіка зв'язування готелів через відношення один-до-одного між таблицями Tour та Hotel суттєво обмежує можливість вибору декількох альтернативних варіантів розміщення в межах однієї путівки.

На основі аналізу виявлених обмежень визначено пріоритетні напрямки модернізації та масштабування інформаційної системи для її подальшої інтеграції в реальний сектор туристичного бізнесу. Першочерговим завданням є рефакторинг схеми даних із переведенням зв'язку між таблицями Tour та Hotel до кардинальності багато-до-багатьох (M:N) через створення проміжної таблиці зв'язків для реалізації концепту динамічного пакетування. Наступним кроком є впровадження розширення Flask-Login та бібліотеки Werkzeug для хешування паролів за алгоритмом scrypt або bcrypt, що дозволить побудувати надійну рольову модель доступу (RBAC). Також передбачається інтеграція зовнішніх RESTful API платіжних систем (наприклад, Stripe або LiqPay) для автоматичної зміни статусу бронювання з processing на confirmed у момент проведення транзакції без залучення менеджера. Фінальним етапом модернізації є впровадження асинхронних запитів (AJAX/Fetch API), що забезпечить миттєву фільтрацію турів за країною, ціною та датами на клієнтській стороні без перезавантаження веб-сторінки, трансформуючи розроблене програмне рішення у високонадійну розподілену ERP-систему.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано повний комплекс інженерних робіт із програмної реалізації, тестування, верифікації та

критичного оцінювання ефективності функціонування розробленої інформаційної системи автоматизації туристичної агенції. На основі проведеного комплексу розробки та досліджень сформовано такі науково-технічні висновки.

У межах архітектурної побудови програмного забезпечення успішно реалізовано триланкову модель веб-додатка, де центральним компонентом виступає серверна логіка під керуванням мови програмування Python та легковагового мікрофреймворку Flask. Розділення відповідальності за паттерном SoC дозволило ізолювати маршрутизацію запитів, бізнес-алгоритми та низькорівневий шлюз взаємодії з СКБД MySQL. Впровадження механізму пулу з'єднань драйвера `mysql-connector-python` забезпечило стабільний та високошвидкісний обмін даними, дозволивши знизити середній час відгуку системи при обробці запитів до транзакційного сховища, що повністю задовольняє сучасні інженерні стандарти.

Розробка клієнтського рівня інтерфейсу користувача базується на застосуванні семантичної розмітки HTML5, каскадних таблиць стилів CSS3 та компонентів фреймворку Bootstrap 5, що дозволило створити повністю адаптивний і кросбраузерний фронтенд. Завдяки інтеграції шаблонізатора Jinja2 реалізовано концепт динамічного рендерингу сторінок на стороні сервера, що забезпечило швидке відображення актуального каталогу турпродукту та унеможливило дублювання коду макетів інтерфейсу. Проектування адаптивних веб-форм із вбудованими елементами багаторівневої клієнтської валідації та динамічними випадаючими списками дозволило оптимізувати ергономіку робочого простору менеджерів турфірми та мінімізувати виникнення помилок при ручному введенні даних.

Комплексне функціональне та безпекове тестування системи підтвердило її високу стабільність і надійність. Перевірка працездатності за методологіями «чорної» та «білої скриньки» показала коректність виконання операцій створення та читання даних (CR), а також обробки виняткових ситуацій із безпечним відкатом транзакцій у СКБД у разі виникнення помилок

введення. Аудит захищеності за стандартами OWASP підтвердив абсолютну стійкість реляційного сховища до атак типу SQL-ін'єкцій завдяки безкомпромісному використанню низькорівневих параметризованих запитів (Prepared Statements), які повністю ізолюють користувацькі дані від компільованого шаблону SQL-коду.

Проведена інженерна оцінка отриманих результатів продемонструвала високу ефективність централізації інформаційних потоків у нормалізованій структурі бази даних, проте виявила низку функціональних обмежень, пов'язаних із відсутністю вбудованої системи рольової автентифікації користувачів, модулів аналітичної звітності та гнучкого динамічного пакетування готелів. Визначені на основі аналізу обмежень напрямки модернізації, такі як впровадження розширення Flask-Login із шифруванням паролів, інтеграція зовнішніх RESTful API платіжних систем та розгортання асинхронних клієнтських запитів (Fetch API), формують чітку дорожню карту розвитку проєкту, спрямовану на його трансформацію у повноцінну розподілену ERP-систему для реального сектору туристичного бізнесу.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Розробка та експлуатація сучасних інформаційних управляючих систем і програмного забезпечення вимагає неухильного дотримання нормативно-правових актів у сфері охорони праці, промислової безпеки та виробничої санітарії. Оскільки робоче місце менеджера туристичної агенції та інженера-програміста пов'язане з тривалим використанням комп'ютерної техніки та відеодисплейних терміналів (ВДТ), виникає комплекс шкідливих і небезпечних виробничих чинників, які негативно впливають на організм людини. Метою цього розділу є аналіз умов праці користувачів ЕОМ, розробка заходів із мінімізації шкідливих факторів та визначення порядку дій у разі виникнення надзвичайних ситуацій відповідно до чинного законодавства України.

4.1. Аналіз умов праці та ідентифікація небезпечних і шкідливих виробничих чинників

Експлуатація засобів обчислювальної техніки в офісних приміщеннях регламентується вимогами «Правил охорони праці під час експлуатації відеодисплейних терміналів інформаційно-комунікаційних систем» та Державними санітарними правилами і нормами ДСанПіН 3.3.2.007-98. Робота за комп'ютером належить до категорії розумової праці з високим нервово-емоційним та зоровим напруженням, яка характеризується тривалим перебуванням у статичній робочій позі.

Під час роботи користувача в приміщенні з комп'ютерною технікою на нього діє ряд шкідливих і небезпечних виробничих чинників. До фізичних чинників належать невідповідність параметрів мікроклімату, таких як температура, відносна вологість та швидкість руху повітря, а також незадовільний рівень освітленості робочої зони, наявність блиску та відблисків на екранах моніторів. Сюди ж відноситься підвищений рівень шуму

від роботи систем охолодження ЕОМ, наявність електромагнітних полів високої чи низької частоти, накопичення статичної електрики та постійна небезпека ураження електричним струмом через пошкодження ізоляції струмоведучих частин або відсутність заземлення обладнання.

Психофізіологічні чинники включають зорове напруження, пов'язане з тривалим фокусуванням на мерехтливому екрані, що з часом викликає астенопію. Тривале перебування в сидячому положенні викликає гіподинамію та гіпокінезію, які призводять до хронічних порушень опорно-рухового апарату користувача. Крім того, монотонність праці та висока концентрація уваги під час обробки великих масивів інформації викликають суттєве розумове та нервово-емоційне перенапруження персоналу.

4.2. Виробнича санітарія, ергономіка та організація робочого простору

Для підтримання стабільної працездатності та збереження здоров'я персоналу параметри мікроклімату в робочих приміщеннях ЕОМ повинні підтримуватися на оптимальному рівні за допомогою систем центрального опалення, кондиціювання та припливно-витяжної вентиляції. В теплий період року оптимальна температура повітря має становити від 22 до 25 градусів Цельсія, у холодний — від 20 до 22 градусів Цельсія, відносна вологість повітря повинна коливатися в межах 40–60%, а швидкість руху повітря не повинна перевищувати 0,2 метра за секунду. Природне та штучне освітлення відіграє ключову роль у профілактиці зорової втоми. Робочі столи слід розмішувати так, щоб природне світло падало збоку, переважно зліва, для запобігання утворенню тіней та відблисків на екрані. Штучне освітлення в приміщеннях з ЕОМ має бути комбінованим із використанням світлодіодних або люмінесцентних ламп, які забезпечують рівномірний розподіл світлового потоку та повну відповідність фактичної освітленості встановленим державним будівельним нормам України.

Ергономічні вимоги до організації робочого місця передбачають правильне просторове розміщення меблів та техніки. Площа на одне робоче місце користувача ПК повинна становити не менше 6 квадратних метрів, а об'єм приміщення — не менше 20 кубічних метрів. Робочий стіл повинен мати регульовану висоту, а крісло — підйомно-поворотний механізм із регулюванням висоти й кута нахилу спинки та підлокітників для зниження статичного навантаження на хребет і м'язи плечового поясу. Відеодисплейний термінал необхідно розміщувати на відстані 600–700 міліметрів від очей користувача, а кут зору на екран повинен становити 10–20 градусів нижче лінії горизонту, що суттєво знижує навантаження на шийний відділ хребта та запобігає швидкому втомлюванню.

4.3. Технобезпека, пожежна профілактика та дії в надзвичайних ситуаціях

Оскільки обчислювальна техніка живиться від електричної мережі, існує постійна небезпека ураження персоналу струмом через доторкання до відкритих струмоведучих частин або виникнення витоку на корпус. Для забезпечення високого рівня електробезпеки в приміщенні всі металеві неструмоведучі корпуси обладнання підключені до захисного заземлення із опором контуру не більше 4 Ом. Лінії живлення комп'ютерів захищені автоматичними вимикачами для захисту від перевантажень і коротких замикань, а також пристроями захисного вимкнення, які миттєво знеструмлюють мережу у разі виникнення витоку струму, а розетки і комутаційні щити закриваються захисними кожухами. Пожежна безпека регламентується Законом України «Про пожежну безпеку», оскільки в офісі знаходяться тверді горючі речовини у вигляді пластикових корпусів, паперу та ізоляції кабелів. Профілактичні пожежні заходи включають використання негорючих оздоблювальних матеріалів, заборону застосування саморобних подовжувачів та обладнання приміщення автоматичною системою пожежної сигналізації з димовими сповіщувачами. Оскільки гасіння пожеж в

електроустановках під напругою водою або піною категорично заборонено, офіс забезпечується вуглекислотними або порошковими вогнегасниками, а на видному місці розміщується затверджений план евакуації людей.

Цивільний захист персоналу туристичної агенції організовується відповідно до вимог Кодексу цивільного захисту України з метою мінімізації наслідків надзвичайних ситуацій техногенного, природного або воєнного характеру. У разі оголошення сигналу повітряної тривоги чи інших сигналів цивільного захисту робота на комп'ютерній техніці негайно припиняється, всі операційні системи закриваються контролерами Flask із збереженням поточного стану баз даних, а сервери та периферійне обладнання повністю знеструмлюються за допомогою головного автоматичного вимикача в щитку живлення. Персонал забирає особисті речі, документи та організовано, без паніки, залишає приміщення, прямуючи найкоротшим евакуаційним шляхом до найближчого сертифікованого захисного сховища або найближчої станції метрополітену, що виконує функції укриття. Для підтримання постійної готовності в офісі призначається відповідальна особа з цивільного захисту, проводяться регулярні інструктажі з техніки безпеки, а приміщення укомплектовується колективною аптечкою першої медичної допомоги та індивідуальними засобами захисту дихальних шляхів.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі було успішно розв'язано актуальну науково-практичну задачу, що полягала в системному аналізі, обґрунтуванні, проєктуванні та програмній реалізації високоефективної інформаційної системи для автоматизації ключових бізнес-процесів туристичної агенції. На основі проведеного комплексу досліджень, математичного моделювання та інженерної розробки сформовано комплексні підсумкові результати, які повністю охоплюють усі етапи життєвого циклу створення програмного продукту.

Завдяки проведенню детального системного аналізу операційної діяльності турфірми було виявлено критичні уразливості та дефіцити ручної обробки інформації, зокрема високу ймовірність виникнення овербукінгу готельного фонду та системних помилок у паспортних даних клієнтів. За допомогою методології діаграм потоків даних (DFD) побудовано ієрархічну трирівневу модель інформаційного обміну, яка дозволила формалізувати архітектуру руху даних від зовнішніх акторів до внутрішніх процесів перетворення, заклавши надійну основу для ізоляції логічних модулів розроблюваного програмного забезпечення.

На основі розробленого концептуального ER-концепту сформовано логічну та фізичну структури бази даних під керуванням СКБД MySQL з використанням високонадійного транзакційного двигуна зберігання InnoDB. У межах роботи ліквідовано структурні недоліки попереднього курсового проєкту шляхом виокремлення текстових полів у самостійні довідкові сутності, що дозволило провести повну декомпозицію відношень та привести базу даних до безкомпромісного стану третьої нормальної форми (3NF), повністю виключивши інформаційну надлишковість та аномалії модифікації рядків.

У межах розробки серверної частини системи успішно застосовано триланкову архітектуру веб-додатка на мові програмування Python. Безпека та

швидкість обміну даними з СКБД MySQL забезпечується за рахунок інтеграції пулу з'єднань драйвера `mysql-connector-python`, який мінімізує накладні TCP-витрати та забезпечує стабільний час відгуку системи. Клієнтська частина веб-додатка реалізована за принципом динамічного рендерингу (SSR) шаблонізатора Jinja2 з використанням сітки та адаптивних компонентів Bootstrap 5, що гарантує високу ергономічність робочого простору користувачів на будь-яких пристроях.

Комплексне функціональне тестування за методологіями «чорної» та «білої скриньки» підтвердило стабільність обробки всіх операцій CRUD та коректність роботи алгоритмів відкату транзакцій при спробах введення некоректних даних. Проведений безпековий аудит за стандартами OWASP продемонстрував абсолютну стійкість системи до уразливостей класу Injection завдяки впровадженню параметризованих SQL-запитів (Prepared Statements) на рівні Flask-контролерів, що надійно захищає конфіденційну та фінансову інформацію фірми від атак шкідливого впровадження стороннього коду.

Проведене критичне оцінювання отриманих результатів підтвердило повне виконання технічного завдання, проте дозволило виявити обмеження, пов'язані з відсутністю сесійної авторизації користувачів на поточному етапі розгортання. Сформована дорожня карта подальшої модернізації, яка включає інтеграцію розширень Flask-Login для рольового розмежування доступу та підключення платіжних RESTful API (Stripe/LiqPay), забезпечує чіткі інженерні орієнтири для масштабування легковагового рішення у повноцінну розподілену ERP-систему для цифрового туристичного бізнесу. Розроблена інформаційна система повною мірою демонструє високу інженерну цінність, архітектурну стійкість та готовність до безпосереднього впровадження у практику цифровізації підприємств туристичної галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Celko J. Joe Celko's SQL for Smarties: Advanced SQL Programming. 5th ed. Waltham, MA: Morgan Kaufmann, 2020. 852 p.
2. Dennis A., Wixom B. H., Tegarden D. Systems Analysis and Design: An Object-Oriented Approach with UML. 6th ed. Hoboken: Wiley, 2021. 544 p.
3. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Boston: Pearson, 2021. 1280 p.
4. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. Irvine: University of California, 2020. 180 p.
5. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2021. 560 p.
6. Garcia-Molina H., Ullman J. D., Widom J. Database Systems: The Complete Book. 2nd ed. Boston: Pearson, 2020. 1248 p.
7. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes. Geneva: ISO, 2021. 142 p.
8. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2020. 432 p.
9. Richardson C. Microservices Patterns: With Examples in Java and Python. Shelter Island, NY: Manning Publications, 2022. 520 p.
10. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill, 2020. 1376 p.
11. Slatkin B. Effective Python: 90 Specific Ways to Write Better Python. 2nd ed. Boston: Addison-Wesley, 2021. 480 p.
12. Sommerville I. Software Engineering. 10th ed. Harlow: Pearson, 2021. 816 p.
13. MySQL 8.4 Reference Manual: Optimization, Indexing and InnoDB Architecture. Oracle Corporation. URL: <https://dev.mysql.com/doc/refman/8.4/en/>
14. OWASP Top 10:2025. OWASP Foundation. URL: <https://owasp.org/www-project-top-ten/>

15. Flask Documentation (Release 3.1.x): WSGI Application and Routing Engine.
Pallets Projects. URL: <https://flask.palletsprojects.com/>.

ДОДАТОК А – КОД СТВОРЕННЯ ТАБЛИЦЬ MYSQL

```

CREATE DATABASE IF NOT EXISTS TravelAgency;
USE TravelAgency;

-- 1. Таблиця для збереження профілів клієнтів
CREATE TABLE Client (
    ClientID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DateOfBirth DATE,
    PassportData VARCHAR(20) UNIQUE NOT NULL,
    PassportExpiryDate DATE,
    Phone VARCHAR(15),
    Email VARCHAR(100) UNIQUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 2. Таблиця для реєстрації туристичних фірм (контрагентів)
CREATE TABLE TravelAgency (
    AgencyID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Address VARCHAR(255),
    ContactPhone VARCHAR(15),
    Email VARCHAR(100) UNIQUE,
    Website VARCHAR(100)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 3. Таблиця для обліку туристичних продуктів
CREATE TABLE Tour (
    TourID INT PRIMARY KEY AUTO_INCREMENT,
    TourName VARCHAR(100) NOT NULL,
    TourType ENUM('bus', 'flight') NOT NULL,
    Route TEXT,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    Duration INT NOT NULL,
    Cost DECIMAL(10, 2) NOT NULL,
    Category ENUM('economy', 'standard', 'premium') NOT NULL,
    Description TEXT,
    AgencyID INT,
    FOREIGN KEY (AgencyID) REFERENCES TravelAgency(AgencyID) ON DELETE
SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 4. Таблиця для обліку готельного фонду з обмеженням 1:1 до туру
CREATE TABLE Hotel (
    HotelID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100) NOT NULL,
    Category ENUM('3*', '4*', '5*') NOT NULL,
    Address VARCHAR(255),
    Services TEXT,
    TourID INT UNIQUE,
    FOREIGN KEY (TourID) REFERENCES Tour(TourID) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 5. Центральна транзакційна таблиця бронювань
CREATE TABLE Booking (
    BookingID INT PRIMARY KEY AUTO_INCREMENT,
    BookingDate DATE NOT NULL,
    Status ENUM('confirmed', 'processing', 'canceled') NOT NULL,
    PaymentAmount DECIMAL(10, 2) NOT NULL,

```

```

        PaymentMethod ENUM('card', 'cash', 'bank_transfer') NOT NULL,
        ClientID INT,
        TourID INT,
        FOREIGN KEY (ClientID) REFERENCES Client(ClientID) ON DELETE SET NULL,
        FOREIGN KEY (TourID) REFERENCES Tour(TourID) ON DELETE SET NULL
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 6. Таблиця для деталізації додаткових послуг під конкретне бронювання
CREATE TABLE Service (
    ServiceID INT PRIMARY KEY AUTO_INCREMENT,
    ServiceName VARCHAR(100) NOT NULL,
    Description TEXT,
    Cost DECIMAL(10, 2) NOT NULL,
    BookingID INT,
    FOREIGN KEY (BookingID) REFERENCES Booking(BookingID) ON DELETE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

ДОДАТОК Б -КОД СЕРВЕРНОЇ ЧАСТИНИ ПРОГРАМИ

```

from flask import Flask, render_template, request, redirect, url_for, flash
import mysql.connector
from datetime import datetime

app = Flask(__name__)
# Секретний ключ для роботи механізму flash-повідомлень
app.secret_key = 'super_secret_key_travel_agency_122'

def get_db_connection():
    # Конфігурація підключення до СКБД MySQL згідно з твоїми налаштуваннями
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="1111",
        database="travelagency",
        pool_name="travel_pool",
        pool_size=5
    )

# =====
# 1. МАРШРУТ ДЛЯ ГОЛОВНОЇ СТОРІНКИ (ДАШБОРД)
# =====
@app.route('/')
def index():
    return render_template('index.html')

# =====
# 2. МАРШРУТ ДЛЯ КАТАЛОГУ ТУРІВ
# =====
@app.route('/tours')
def tours():
    connection = None
    cursor = None
    try:
        connection = get_db_connection()
        cursor = connection.cursor()
        # Витягуємо всі поля з таблиці Tour для виведення в таблицю
        cursor.execute("SELECT * FROM Tour")
        all_tours = cursor.fetchall()
        return render_template('tours.html', tours=all_tours)
    except Exception as e:
        return f"Помилка бази даних у модулі турів: {str(e)}"
    finally:
        if cursor: cursor.close()
        if connection: connection.close()

# =====
# 3. МАРШРУТ ДЛЯ ЖУРНАЛУ БРОНЮВАНЬ
# =====
@app.route('/bookings')
def bookings():
    connection = None
    cursor = None
    try:
        connection = get_db_connection()

```

```

# Вмикаємо dictionary=True, щоб отримувати результати у вигляді
словника
cursor = connection.cursor(dictionary=True)

# SQL-запит, який об'єднує три таблиці та витягує імена й назви турів
query = """
SELECT
    b.BookingID,
    b.BookingDate,
    b.Status,
    b.PaymentAmount,
    b.PaymentMethod,
    c.FirstName,
    c.LastName,
    t.TourName
FROM Booking b
JOIN Client c ON b.ClientID = c.ClientID
JOIN Tour t ON b.TourID = t.TourID
ORDER BY b.BookingID DESC
"""
cursor.execute(query)
all_bookings = cursor.fetchall()

return render_template('bookings.html', bookings=all_bookings)
except Exception as e:
    return f"Помилка бази даних у журналі замовлень: {str(e)}"
finally:
    if cursor: cursor.close()
    if connection: connection.close()

# =====
# 4. МАРШРУТ ДЛЯ АВТОМАТИЗОВАНОГО ДОДАВАННЯ
# =====
@app.route('/add_booking', methods=['GET', 'POST'])
def add_booking():
    if request.method == 'POST':
        client_id = request.form.get('client_id')
        tour_id = request.form.get('tour_id')
        payment_method = request.form.get('payment_method')
        booking_date = datetime.now().strftime('%Y-%m-%d')
        status = 'processing'

        connection = None
        cursor = None
        try:
            connection = get_db_connection()
            cursor = connection.cursor()

            # Автоматично дізнаємося вартість Cost для обраного TourID
            price_query = "SELECT Cost FROM Tour WHERE TourID = %s"
            cursor.execute(price_query, (tour_id,))
            result = cursor.fetchone()

            if not result:
                flash('Обраний тур не знайдено!', 'danger')
                return redirect(url_for('add_booking'))

            tour_price = result[0]

            # Параметризована безпечна транзакція вставки
            insert_query = """
                INSERT INTO Booking (BookingDate, Status, PaymentAmount,

```

```

PaymentMethod, ClientID, TourID)
        VALUES (%s, %s, %s, %s, %s, %s)
        """
        cursor.execute(insert_query, (booking_date, status, tour_price,
payment_method, client_id, tour_id))
        connection.commit()

        flash('Бронювання успішно сформовано! Ціна розрахована
автоматично.', 'success')
        return redirect(url_for('bookings'))

    except Exception as e:
        if connection: connection.rollback()
        flash(f'Критичний збій транзакції: {str(e)}', 'danger')
        return redirect(url_for('add_booking'))
    finally:
        if cursor: cursor.close()
        if connection: connection.close()

# GET-запит: підвантажуюмо довідники для форми
connection = None
cursor = None
try:
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("SELECT ClientID, FirstName, LastName FROM Client")
    clients = cursor.fetchall()

    cursor.execute("SELECT TourID, TourName, Cost FROM Tour")
    tours = cursor.fetchall()

    return render_template('add_booking.html', clients=clients,
tours=tours)
except Exception as e:
    return f"Помилка завантаження НДІ довідників: {str(e)}"
finally:
    if cursor: cursor.close()
    if connection: connection.close()

if __name__ == '__main__':
    # Запуск локального веб-сервера в режимі дебагу
    app.run(debug=True)

```

ДОДАТОК В – ЛІСТИНГ КОДУ ІНТЕРФЕЙСУ

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Реєстрація бронювання – TravelAgency</title>
  <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css
" rel="stylesheet">
  <link href="https://cdn.jsdelivrivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css" rel="stylesheet">
</head>
<body class="bg-light">

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm mb-4">
    <div class="container">
      <a class="navbar-brand fw-bold" href="/"><i class="bi bi-compass-
fill me-2 text-warning"></i>TravelAgency</a>
      <div class="navbar-nav ms-auto">
        <a class="nav-link" href="/tours"><i class="bi bi-grid-3x3-
gap-fill me-1"></i>Каталог типів</a>
        <a class="nav-link" href="/bookings"><i class="bi bi-journal-
text me-1"></i>Журнал бронювань</a>
        <a class="nav-link active" href="/add_booking"><i class="bi
bi-bookmark-plus-fill me-1"></i>Нове бронювання</a>
      </div>
    </div>
  </nav>

  <div class="container py-4">
    <div class="row justify-content-center">
      <div class="col-12 col-md-8 col-lg-6">

        <div class="card shadow border-0 rounded-3">
          <div class="card-header bg-primary text-white py-3 text-
center">
            <h3 class="card-title mb-0 fw-bold">
              <i class="bi bi-bookmark-plus-fill me-
2"></i>Реєстрація нового бронювання
            </h3>
          </div>

          <div class="card-body p-4">

            <div class="alert alert-info d-flex align-items-
center mb-4" role="alert">
              <i class="bi bi-info-circle-fill me-3 fs-4 text-
primary"></i>
              <div>
                <strong>Автоматичний розрахунок:</strong>
                Вартість замовлення буде згенерована системою автоматично згідно з чинним
                тарифом обраного туру.
              </div>
            </div>

            <form action="/add_booking" method="POST">

              <div class="mb-4">

```

```

        <label for="client_id" class="form-label fw-
semibold text-secondary">Оберіть клієнта з бази даних:</label>
        <select class="form-select form-select-lg
border-2" id="client_id" name="client_id" required>
            <option value="" disabled selected>--
Виберіть клієнта (ПІБ) --</option>
            {% for client in clients %}
                <option value="{{ client[0] }}">{{
client[2] }} {{ client[1] }} (ID: {{ client[0] }})</option>
            {% endfor %}
        </select>
    </div>

    <div class="mb-4">
        <label for="tour_id" class="form-label fw-
semibold text-secondary">Оберіть турпродукт:</label>
        <select class="form-select form-select-lg
border-2" id="tour_id" name="tour_id" required>
            <option value="" disabled selected>--
Виберіть тур зі списку --</option>
            {% for tour in tours %}
                <option value="{{ tour[0] }}">{{
tour[1] }} - {{ "{:,.2f}".format(tour[2]) }} UAH</option>
            {% endfor %}
        </select>
    </div>

    <div class="mb-4">
        <label for="payment_method" class="form-label
fw-semibold text-secondary">Метод розрахунку:</label>
        <select class="form-select form-select-lg
border-2" id="payment_method" name="payment_method" required>
            <option value="" disabled selected>--
Оберіть тип транзакції --</option>
            <option value="card"> Безготівковий
розрахунок</option>
            <option value="cash"> Готівкові
кошти</option>
            <option value="bank_transfer">
Банківський переказ</option>
        </select>
    </div>

    <hr class="my-4 text-muted">

    <div class="d-grid gap-2">
        <button type="submit" class="btn btn-success
btn-lg fw-bold shadow-sm py-3">
            <i class="bi bi-check-circle-fill me-
2"></i>Підтвердити та створити замовлення
        </button>
        <a href="/bookings" class="btn btn-outline-
secondary btn-lg fw-semibold mt-1">
            Скасувати операцію
        </a>
    </div>

</form>
</div>
</div>

</div>
</div>

```



```
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.mi
n.js"></script>
</body>
</html>
```