

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Інформаційна система планування туристичних
маршрутів для гідів»**

Виконав студент групи ІПЗ-22-3
ЧОРНОУС Владислав Іванович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
СУГОНЯК Інна Іванівна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЧОРНОУС Владислав Іванович**

Керівник роботи: **ЧИЖМОТРЕ Олексій Володимирович**

Тема роботи: **«Інформаційна система планування туристичних маршрутів для гідів»**,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., № 92/с

Вихідні дані для роботи: **процеси управління туристичною діяльністю, планування маршрутів та координації роботи екскурсководів; методи та інструментальні засоби створення клієнт-серверних веб-систем з використанням платформи .NET 9 та фреймворку React; алгоритми обробки часових колізій розкладу, рольового розмежування доступу та валідації бізнес-правил бронювання.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	01.03.2026р.	01.03.2026р
2	Чижмотря О. В.	01.03.2026р	01.03.2026р
3	Чижмотря О. В.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з інформаційної системи планування маршрутів та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 88 сторінках друкованого тексту. Список використаних джерел містить 31 найменування і займає 3 сторінки. У роботі наведено 49 рисунків та 4 таблиці. Також у роботі є 10 сторінок додатків. Загальний обсяг роботи – 101 сторінка.

Метою роботи є розробка інформаційної системи, що забезпечує автоматизацію процесів планування екскурсійних турів, управління пам'ятками, гідами та бронюваннями, а також надає зручні інструменти календарного планування та рольового доступу для працівників туристичної компанії.

Архітектура системи побудована на клієнт-серверній моделі. Серверна частина реалізована з використанням платформи .NET 9 та ASP.NET Core Web API, що забезпечує надійну обробку бізнес-логіки та взаємодію з базою даних SQL Server. Клієнтська частина розроблена на основі бібліотеки React з використанням мови програмування TypeScript, що забезпечує зручний та інтуїтивно зрозумілий інтерфейс для менеджерів і гідів.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ПЛАНУВАННЯ МАРШРУТІВ, ТУРИСТИЧНА КОМПАНІЯ, ВЕБРОЗРОБКА, REST API, REACT, ASP.NET CORE.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Чорноус В. І.			Інформаційна система планування туристичних маршрутів для гідів Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		Чижмотря О. В.					3	101
						Житомирська політехніка, ІПЗ-22-3		
Рецензент.		Сугоняк І. І.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a route planning information system and an explanatory note. The explanatory note to the thesis contains a text section comprising 88 pages of printed text. The list of references contains 31 entries and spans 3 pages. The thesis includes 49 figures and 4 tables. The thesis also includes 10 pages of appendices. The total length of the thesis is 101 pages.

The objective of the thesis is to develop an information system that automates the processes of planning sightseeing tours, managing attractions, guides, and reservations, and provides convenient tools for calendar planning and role-based access for employees of a travel company.

The system architecture is based on a client-server model. The server-side is implemented using the .NET 9 platform and ASP.NET Core Web API, which ensures reliable business logic processing and interaction with the SQL Server database. The client-side is developed based on the React library using the TypeScript programming language, providing a convenient and intuitive interface for managers and guides.

KEYWORDS: INFORMATION SYSTEM, ROUTE PLANNING, TRAVEL COMPANY, WEB DEVELOPMENT, REST API, REACT, ASP.NET CORE.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ ДО СИСТЕМИ.....	9
1.1. Постановка задачі	9
1.2. Аналіз наявних аналогів програмного комплексу	12
1.3. Вибір архітектури системи	18
1.4. Обґрунтування вибору інструментальних засобів	23
Висновки до розділу 1	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ	27
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	27
2.2. Розробка бази даних системи.....	34
2.3. Проектування та реалізація алгоритмів роботи системи	39
2.4. Реалізація додатку	44
Висновки до розділу 2	56
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ ПЛАНУВАННЯ МАРШРУТІВ ДЛЯ ГІДІВ.....	58
3.1. Порядок встановлення та налаштування параметрів системи.....	58
3.2. Структура інтерфейсу веб-орієнтованої системи.....	61
3.3. Тестування роботи програмного комплексу	78
Висновки до розділу 3	86
ВИСНОВКИ	87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	89
ДОДАТКИ.....	92

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – програмний інтерфейс, що надає набір готових функцій, класів та структур для взаємодії між різними програмними продуктами.

B2B (Business-to-Business) – комерційна модель взаємодії, яка передбачає надання послуг, продуктів або програмного забезпечення іншим компаніям та підприємствам.

DTO (Data Transfer Object) – шаблон проєктування, призначений для транспортування даних між різними архітектурними шарами системи без розкриття внутрішньої бізнес-логіки.

JWT (JSON Web Token) – відкритий стандарт у форматі JSON для безпечної передачі інформації та безстанової автентифікації користувачів між клієнтом і сервером.

REST (Representational State Transfer) – архітектурний стиль взаємодії компонентів вебзастосунку, заснований на використанні стандартних протоколів передачі даних.

SPA (Single Page Application) – односторінковий вебзастосунок, інтерфейс якого завантажується один раз, а вміст оновлюється динамічно за допомогою асинхронних запитів без перезавантаження сторінки.

URL (Uniform Resource Locator) – стандартизована адреса унікального об'єкта або вебсторінки в мережі Інтернет.

СКБД – система керування базами даних, спеціалізований комплекс програмних засобів для створення, ведення, захисту та спільного використання сховищ інформації.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність дослідження зумовлена стрімким розвитком туристичної галузі та зростаючою потребою екскурсійних бюро в автоматизації операційних процесів для підвищення ефективності менеджменту. Сучасні компанії стикаються з необхідністю одночасно керувати великою кількістю турів, об'єктів, розкладом гідів та клієнтськими бронюваннями. Використання застарілих підходів, таких як паперові журнали або локальні електронні таблиці, призводить до втрати часу, помилок через людський фактор, виникнення часових накладок у графіках персоналу та перевищення ліміту вільних місць у групах.

Ефективне моделювання логістичних маршрутів, раціональний розподіл тижневого навантаження між гідами та синхронна обробка великих масивів замовлень у реальному часі стають ключовими факторами конкурентоспроможності підприємств. Відсутність централізованих інструментів автоматичного контролю бізнес-обмежень ускладнює координацію внутрішніх ресурсів та знижує якість обслуговування клієнтів. Це вимагає розробки спеціалізованого програмного забезпечення, здатного надати надійний цифровий простір для взаємодії менеджерів та гідів з будь-яких пристроїв.

Метою випускної кваліфікаційної роботи є розробка архітектури, алгоритмів роботи та програмна реалізація комплексу веб-орієнтованої інформаційної системи планування маршрутів для гідів, що забезпечує комплексну автоматизацію процесів моделювання екскурсійних турів, управління пам'ятками, обліку робочого часу гідів та координації бронювань, а також надає ергономічні інструменти календарного планування, інтерактивного картографічного аналізу та рольового доступу для працівників туристичної компанії.

Для досягнення поставленої мети необхідно виконати аналіз предметної області й наявних програмних аналогів, формалізувати технічні вимоги до системи, а також обґрунтувати вибір клієнт-серверної архітектури та інструментальних засобів розробки. Важливим етапом є проєктування

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

узагальненої структури платформи, варіантів використання, схеми реляційної бази даних із підтримкою мультитенантності та ключових бізнес-алгоритмів контролю часових діапазонів і доступності ресурсів. Практична частина передбачає розробку серверного API та реалізацію клієнтського застосунку. Завершується комплекс задач описом порядку розгортання й конфігурування параметрів системи, а також проведенням ручного функціонального тестування для перевірки коректності бізнес-логіки й надійності засобів валідації вхідних даних.

Об'єктом дослідження є процес планування та управління екскурсійною діяльністю туристичної компанії, що охоплює організацію турів, розподіл гідів та роботу з бронюваннями.

Предметом дослідження є методи та засоби автоматизації процесів планування маршрутів, управління розкладом гідів та обробки бронювань у межах веборієнтованої інформаційної системи. Дослідження також охоплює аналіз підходів до проектування клієнт-серверних застосунків, побудови гнучкої системи рольового доступу та реалізації комплексної бізнес-логіки з автоматичною перевіркою обмежень і валідацією даних на рівні серверної частини.

Практичне значення роботи полягає у створенні повнофункціонального веборієнтованого застосунку, який, за сценарієм, може бути використаний реальною туристичною компанією для автоматизації щоденних операційних процесів. Розроблена система дозволяє менеджерам ефективно планувати екскурсії, керувати розкладом гідів з урахуванням їхнього тижневого навантаження, контролювати доступність місць у запланованих екскурсіях та обробляти бронювання з автоматичною перевіркою бізнес-обмежень. Реалізація системи демонструє практичне застосування сучасних технологій веброзробки, зокрема ASP.NET Core Web API та React, у вирішенні реальних прикладних задач автоматизації бізнес-процесів.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ ДО СИСТЕМИ

1.1. Постановка задачі

Операційна діяльність сучасної екскурсійної компанії супроводжується необхідністю постійного керування багатьма взаємопов'язаними ресурсами. У межах розробки інформаційної системи планування туристичних маршрутів для гідів необхідно вирішити задачі координації графіків роботи гідів, урахування логістичних особливостей маршрутів, контролю наповнюваності екскурсійних груп та оперативної обробки вхідних заявок. Оскільки всі ці елементи тісно пов'язані між собою, будь-яка неточність або зміна в розкладі, наприклад перенесення часу екскурсії чи коригування послідовності відвідування локацій, може порушити узгодженість роботи системи на поточний день.

Проте на практиці більшість фірм малого та середнього бізнесу досі ведуть облік застарілими методами. Інформація про зайнятість гідів та доступність місць фіксується вручну – у паперових журналах, блокнотах або таблицях Excel. Через це виникає проблема людського фактора. Коли обсяг заявок зростає, менеджери починають плутатися в даних, витрачають багато часу на телефонні дзвінки та переписку в месенджерах для узгодження розкладу. В результаті заявки опрацьовуються повільно, інформація втрачається, а сама система втрачає здатність до масштабованого функціонування.

Найбільш критичним наслідком ручного керування є виникнення організаційних «пляшкових шийок» під час координації ресурсів [1]. У сфері туризму такими вузькими місцями зазвичай стають робочий час гідів та ліміти наповнюваності екскурсійних груп. Без централізованої інформаційної системи менеджер може випадково призначити одного гіда на два різні тури, що перетинаються за часом, або сформувати групу з кількістю учасників, що перевищує встановлений ліміт маршруту. Такі проблеми в операційних процесах призводять до деградації сервісу, оскільки виникає необхідність скасовувати тури в останній момент, коригувати вже підтверджені заявки та втрачати

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

репутацію через негативні відгуки. Саме тому впровадження спеціалізованої інформаційної системи планування екскурсійних маршрутів є необхідним кроком.

Ефективна автоматизація операційних процесів вимагає чіткого визначення всіх учасників системи та встановлення меж їхньої взаємодії з програмним комплексом. У розроблюваній системі виділяються два ключові суб'єкти, які мають кардинально різні посадові обов'язки та інформаційні потреби. Головним актором системи виступає менеджер – працівник, який повністю відповідає за організацію туристичної діяльності та має розширені права для управління всіма об'єктами системи. У межах своїх повноважень він наповнює каталоги пам'яток, формує екскурсійні тури, керує календарним розкладом виїздів, призначає гідів та контролює модуль заявок. Другорядним актором є гід, чий інформаційні потреби є обмеженими і мають виключно ознайомчий характер. Йому доступний лише особистий кабінет у вигляді календаря для перегляду власного актуального розкладу та деталей призначених саме йому турів без можливості редагування даних системи. Таке чітке обмеження функціоналу дозволяє гіді фокусуватися на виконанні своїх безпосередніх обов'язків, не відволікаючись на адміністрування системи.

З огляду на такий суттєвий розподіл обов'язків, важливим завданням постає впровадження концепції контролю доступу на основі ролей (RBAC) [2]. У системах такого типу безпека даних є одним із головних пріоритетів, оскільки випадкова або навмисна модифікація інформації сторонніми особами може нашкодити роботі фірми. Реалізація моделі RBAC дозволяє чітко розмежувати права доступу на рівні програмного коду, а система має перевіряти роль користувача при кожному запиті. Гід за жодних умов не повинен отримати доступ до екранів редагування турів, параметрів заявок чи персональних даних клієнтів. Це не лише захищає конфіденційну інформацію підприємства від несанкціонованого перегляду, а й унеможливорює виникнення помилок, коли користувач із обмеженими повноваженнями міг би випадково змінити важливий елемент розкладу або параметри чужого замовлення.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Оскільки програмний комплекс проектується як комерційний продукт для туристичної діяльності, його фундаментальною особливістю є робота в мультитенантному середовищі, де єдина платформа обслуговує багато незалежних екскурсійних компаній одночасно [3]. Спільне використання ресурсів вимагає забезпечення повної логічної ізоляції даних, тому користувачі однієї фірми за жодних умов не повинні мати доступу до розкладів, клієнтських баз, маршрутів чи звітної інформації інших організацій. База даних клієнтів та графіки завантаженості є важливою службовою інформацією кожної компанії, тому будь-який витік або перехресне відображення даних між окремими користувацькими просторами є неприпустимим. Через це система зобов'язана на рівні кожного запиту автоматично ідентифікувати приналежність користувача до конкретної організації та суворо фільтрувати дані, забезпечуючи автономність і безпеку інформаційного простору кожного замовника.

Функціонал системи розподілений на кілька ключових модулів, першим з яких є модуль управління інформаційним контентом. Він забезпечує ведення довідників визначних пам'яток із можливістю прикріплення файлів та точних геолокаційних міток для побудови інтерактивних карт. На основі цих пам'яток менеджери можуть формувати базові маршрути турів, визначати їхню послідовність, тривалість та описові характеристики. Наступним є модуль календарного планування, призначений для створення сітки запланованих екскурсій на конкретні дати. Для менеджерів цей модуль надає інструменти наочної візуалізації розкладу з підтримкою зручних режимів перегляду, тоді як для гідів передбачені адаптовані кабінети для мобільних пристроїв, що дозволяють оперативно переглядати особисту зайнятість на місцях.

Окреме місце в архітектурі посідає модуль взаємодії з клієнтами, який автоматизує процес реєстрації та опрацювання заявок. Його логіка побудована на контролі наповнюваності екскурсійних груп – система в реальному часі зіставляє кількість заявлених місць із гранично допустимими лімітами туру, щоб унеможливити ситуації перевищення місткості. Крім того, модуль автоматично

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

відстежує часові ліміти, блокуючи спроби оформити або скасувати заявку, якщо до початку екскурсії залишилося менше встановленого регламентом часу.

Для гарантування коректності даних програмний комплекс впроваджує набір автоматичних перевірок. Логіка координації графіків персоналу базується на порівнянні часових інтервалів – система миттєво виявляє та блокує будь-які прямі накладання розкладу, не дозволяючи призначити гіда на два тури одночасно. Крім того, алгоритм враховує правило обов’язкових перерв. Програма також самостійно підраховує сумарну тривалість екскурсій кожного гіда в межах поточного тижня і блокує нові призначення, якщо перевищено встановлений внутрішнім регламентом ліміт робочих годин. Паралельно з цим правила валідації діють і в модулі взаємодії з клієнтами – система автоматично перевіряє наявність вільних місць у групах для запобігання надмірному заповненню та відхиляє будь-які спроби створити або відредагувати заявку «минулим числом».

1.2. Аналіз наявних аналогів програмного комплексу

У сучасній туристичній індустрії веб-орієнтовані програмні рішення поступово витісняють ручні способи ведення маршрутної та календарної документації, оскільки саме автоматизоване середовище дозволяє синхронно враховувати послідовність локацій, час виїзду, зайнятість гідів і наповнюваність екскурсійних груп. Для предметної області даної роботи особливо важливими є не стільки комерційні механізми, скільки можливість моделювати екскурсію як впорядкований маршрут із чіткою географічною структурою, контролем часових інтервалів і коректним відображенням локацій у календарі. Саме тому аналіз аналогів доцільно проводити з огляду на те, наскільки вони підтримують маршрутне планування, картографічний супровід та координацію гідів, а не лише функції бронювання чи обліку заявок.

Платформа TourCMS розроблялася як хмарне рішення для туроператорів та організаторів екскурсій, орієнтоване на структуроване описання турів і керування їхньою подальшою організацією (рис. 1.1). В основі архітектурного

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

підходу цієї системи лежить класична монолітна хмарна архітектура з централізованим керуванням даними. Платформа використовує концепцію «все в одному», де ядром системи є механізм бронювання, що взаємодіє з іншими функціональними модулями. Такий підхід забезпечує стабільність під час опрацювання великого обсягу запитів, проте водночас створює щільну внутрішню зв'язаність компонентів, що ускладнює локальне налаштування інтерфейсів і впровадження гнучких правил, орієнтованих саме на маршрутну логіку окремої туристичної компанії.

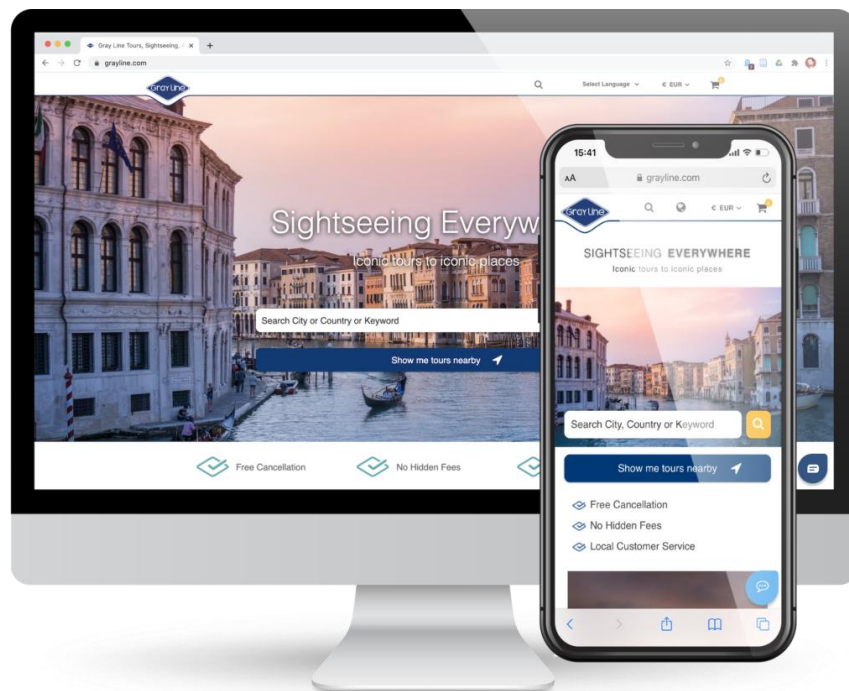


Рис. 1.1. TourCMS

Ключові функціональні можливості TourCMS зосереджені навколо конфігурації складних турів, ведення клієнтських баз та динамічного керування екскурсійними слотами. Система дозволяє детально описувати маршрути, прив'язувати пам'ятки та вести облік призначених гідів у межах створюваних календарних планів [4]. Важливою перевагою платформи є її розвинений модуль дистрибуції, який забезпечує оперативне оновлення інформації про доступність місць у турах та підтримує узгодження між різними каналами поширення маршрутних пропозицій. Менеджери компанії мають доступ до загального розкладу, де можуть коригувати склад груп і відстежувати стан запланованих екскурсій.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Незважаючи на функції TourCMS має суттєві недоліки та обмеження. Інтерфейс системи є застарілим і перевантаженим текстовими таблицями, що ускладнює щоденну роботу порівняно з сучасними інтерактивними календарними рішеннями. Організація внутрішньої логістики та контролю завантаженості персоналу тут реалізована на базовому рівні. Система не має гнучких автоматичних алгоритмів для перевірки складних часових конфліктів у графіках гідів або контролю ліміту їхніх тижневих робочих годин, залишаючи значну частину координації на ручне опрацювання. Крім того, оскільки платформа орієнтована на глобальний ринок і зовнішню дистрибуцію, вона є надмірно громіздкою для локальних компаній, які потребують простого, швидкого та ізольованого інструменту саме для координації внутрішніх маршрутних ресурсів і оперативного менеджменту на місцях.

Платформа Travify є більш наближеним до тематики даної роботи аналогом, оскільки вона орієнтована саме на побудову маршрутних планів, підготовку туристичних itineraries та супровід групових поїздок, а не лише на загальне календарне бронювання (рис. 1.2). На офіційній сторінці продукту зазначено, що система підтримує drag-and-drop конструктор маршрутів, мобільний доступ до поїздки, спільну бібліотеку контенту, роботу з груповими подорожами та імпорт даних від туристичних постачальників, зокрема турів, проживання, трансферів і супровідних матеріалів [5].

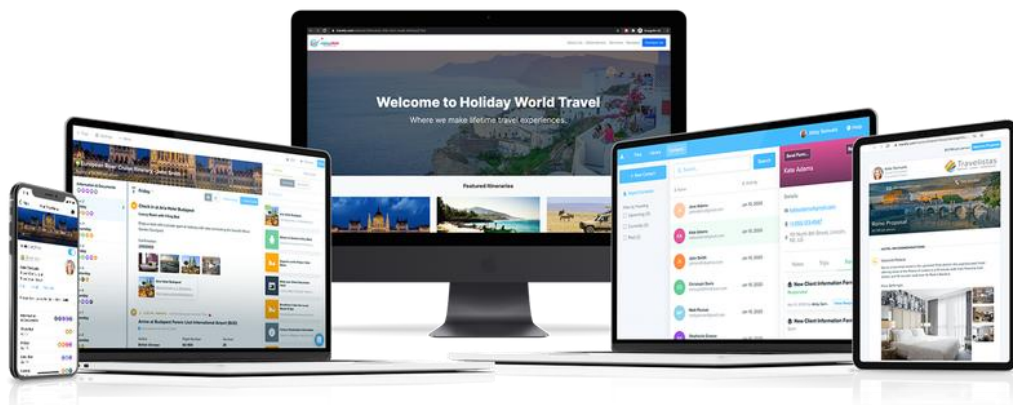


Рис. 1.2. Travify

До переваг Travify слід віднести зручність формування структурованого маршруту, швидке наповнення поїздки довідковими матеріалами та підтримку

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

мобільного перегляду, що робить платформу корисною для візуального представлення екскурсійної програми і координації дій гіда в процесі підготовки маршруту. Важливою перевагою також є можливість повторного використання контенту та централізованого збирання відомостей про елементи поїздки, що спрощує оперативне складання маршруту для клієнта або групи. Разом із тим недоліком Travefy в контексті цієї роботи є те, що система переважно орієнтована на подання готового travel plan і зручність його супроводу, а не на жорстке моделювання екскурсійної послідовності як спрямованого графа локацій із формалізованою логістичною логікою. Крім того, у ній відсутній достатньо глибокий акцент на просторовій валідації порядку пам'яток і на поєднанні маршрутного контенту з календарним контролем зайнятості гідів у межах локальної екскурсійної компанії.

Програмне рішення SimplyBook.me орієнтоване на автоматизацію процесів календарного планування в сервісних організаціях і забезпечує централізоване керування розкладом працівників (рис. 1.3). У межах цієї платформи реалізовано чітке рольове розмежування доступу між адміністраторами, менеджерами та виконавцями послуг.

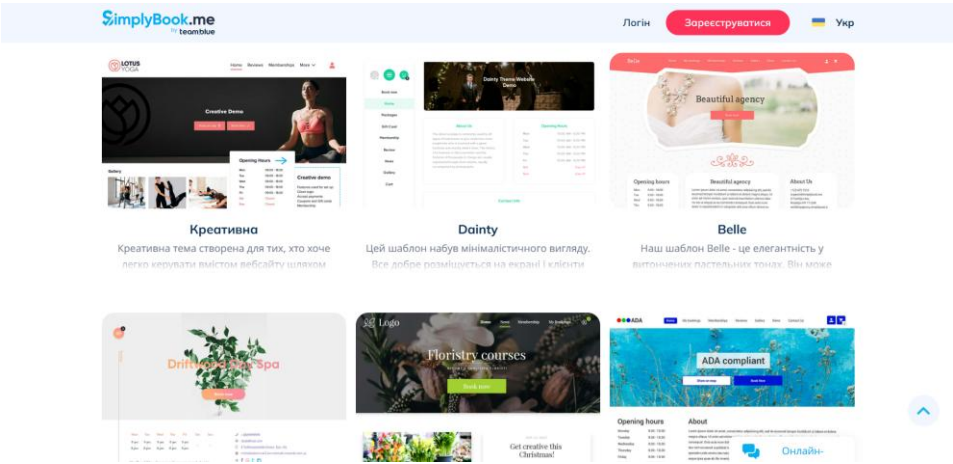


Рис. 1.3. SimplyBook.me

До переваг SimplyBook.me належить наявність механізмів, які дозволяють менеджеру переглядати та координувати загальний розклад, а також контролювати персональні календарі працівників, тоді як користувачі з обмеженими правами можуть працювати лише з власними записами та бачити призначені їм події. Така організація доступу забезпечує захист службової

інформації та підтримує впорядковану взаємодію між різними ролями в межах системи.

Логіка перевірки часових конфліктів у SimplyBook.me базується на динамічному аналізі записів у розкладі в режимі реального часу. Під час створення або редагування події система порівнює новий часовий інтервал із уже наявними записами та у разі виявлення перетину автоматично блокує операцію, сигналізуючи про конфлікт. До переваг також належить можливість задавати індивідуальні параметри робочого часу для кожного працівника, включно з тривалістю робочого дня, перервами та вихідними [6]. Водночас недоліком SimplyBook.me в контексті цієї роботи є її універсальна сервісна орієнтація: система добре підходить для стандартного планування послуг, але не враховує специфіку екскурсійної діяльності, де необхідні складніші правила координації гідів, контроль сумарного тижневого навантаження та узгодження розкладу з маршрутною логістикою. Через це платформа не забезпечує достатнього рівня адаптації до потреб туристичного маршрутного планування.

Для систематизації результатів дослідження аналогів та зіставлення ключових параметрів систем було проведено порівняльний аналіз їхніх функціональних характеристик. Порівняння здійснювалося за критеріями, що безпосередньо впливають на ефективність маршрутного планування, картографічної прив'язки пам'яток, календарної координації гідів і контролю часових обмежень, тобто саме тих властивостей, які є визначальними для автоматизації екскурсійної діяльності.

Таблиця 1.1

Результати порівняння аналогів

Критерій порівняння	TourCMS	Travefy	SimplyBook.me
Основна функціональна орієнтація системи	Централізована конфігурація екскурсійних програм, глобальна дистрибуція та облік замовлень.	Візуальне моделювання просторово-часових траєкторій подорожей та побудова інтерактивних ітингів.	Універсальне календарне планування робочого часу та автоматизація диспетчеризації точкових сервісних послуг.

Моделювання лінійної послідовності (маршрутна складова)	Обмежене (пам'ятки прив'язуються до турів, проте сувора валідація лінійної послідовності руху відсутня).	Наявне (маршрут подорожі формується як чітка хронологічна послідовність локацій та проміжних точок прямування).	Відсутнє (система повністю орієнтована на фіксовані у часі процедури без урахування просторового треку руху).
Картографічна та геопросторова підтримка	Базова (відображення точок відвідування без можливості комплексного просторового аналізу логістичних зв'язків).	Часткова (інтеграція з картами для візуалізації ітинерарію, проте відсутня автоматична валідація геопросторових розривів).	Відсутня (географічні координати об'єктів та інтерактивні картографічні рушії не входять до функціоналу платформи).
Автоматичний контроль часових колізій у розкладі	Відсутній (виявлення часових накладок у графіках гідів повністю покладається на ручний контроль адміністратора).	Відсутній (усунення часових конфліктів та накладань у графіках персоналу не є предметною задачею платформи).	Розвинений (реалізовано динамічні алгоритми синхронізації та миттєвого автоматичного блокування часових перетинів).
Контроль наповнюваності екскурсійних груп	Наявний (підтримується кількісний облік квитків та обмеження реєстрації при досягненні ліміту місць).	Обмежений (можливе фіксування списку учасників групової поїздки, але без суворого автоматичного ресурсного контролю).	Наявний (автоматична верифікація кількості людей на спланований час запису з контролем максимального ліміту учасників).
Рольове розмежування доступу	Базове (наявне спільне середовище для менеджерів, проте відсутня ізольована модель кабінетів для гідів).	Обмежене (орієнтація на спільне редагування контенту без суворого рольового відокремлення прав для гідів).	Розвинене (суворе розмежування повноважень між адміністраторами та захищені робочі простори для гідів).

Проведений аналіз дозволяє визначити вектори розвитку, які будуть враховані при побудові архітектури розроблюваного продукту. TourCMS демонструє придатність до конфігурації турів і календарного супроводу маршрутів, Travify – до побудови маршрутів у форматі супроводу поїздок, а SimplyBook.me – до оперативного контролю розкладу і часових конфліктів. Водночас жоден із розглянутих аналогів не поєднує в межах одного рішення локацій, картографічну прив'язку пам'яток, контроль наповнюваності груп і календарне планування гідів, що обґрунтовує доцільність розробки власної інформаційної системи планування туристичних маршрутів для гідів, у якій саме логістична та геопросторова складові є центральними.

1.3. Вибір архітектури системи

При проєктуванні сучасних програмних систем першочерговим завданням є вибір архітектурного підходу, який визначає спосіб організації, розгортання та взаємодії компонентів додатку. Історично базовим рішенням для побудови додатків є традиційна монолітна архітектура. У такому підході всі функціональні модулі системи – інтерфейс користувача, бізнес-логіка, правила валідації та механізми роботи з базою даних – об'єднані в єдиний кодовий простір та виконуються як один неподільний сервіс [7]. Головними перевагами моноліту є простота початкової розробки, швидкий запуск продукту на перших етапах та висока швидкість локального виклику функцій, оскільки всі операції відбуваються всередині одного процесу і не потребують мережових запитів.

Проте для системи планування екскурсійних маршрутів монолітна архітектура створює певні обмеження. Виникає низька гнучкість масштабування – за необхідності збільшити потужність одного логістичного або картографічного модуля доводиться масштабувати весь моноліт повністю, що неефективно використовує обчислювальні ресурси. Крім того, зростає технологічний ризик – критична помилка в коді другорядного модуля може негативно вплинути на доступність сервісу для всіх гідів та диспетчерів, а процес оновлення ускладнюється – навіть для виправлення незначної помилки в інтерфейсі диспетчера необхідне повне зупинення, перезбирання та перезапуск серверного застосунку.

Альтернативою, яка дозволяє зменшити ці ризики, є розподілена клієнт-серверна архітектура. Цей підхід передбачає чітке розділення на клієнтську частину, що відповідає за відображення інтерфейсу та взаємодію з користувачем, і серверну частину, де зосереджена логістична та картографічна обробка даних, включно з алгоритмами валідації послідовності точок маршруту, геопросторовим аналізом та контролем робочого часу гіда. Взаємодія між цими частинами відбувається через мережу за допомогою стандартизованих протоколів та API, що дозволяє незалежно масштабувати та оновлювати

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

компоненти, знижуючи навантаження на серверну інфраструктуру та забезпечуючи високу швидкість відгуку інтерфейсу для гіда та диспетчера [8].

У сучасній інженерії також розглядають мікросервісну архітектуру, де кожна бізнес-функція виноситься в окремий незалежний міні-сервер. Однак для даного проєкту мікросервісний підхід не був прийнятий, оскільки він уводить додаткову складність у налаштуванні мережевої взаємодії, вимагає значних ресурсів для підтримки інфраструктури та є надлишковим щодо початкового етапу розробки веб-орієнтованого комплексу диспетчеризації маршрутів.

Зіставлення архітектурних концепцій показує, що для проєктованої інформаційної системи планування екскурсійних маршрутів оптимальним вибором є саме розподілена клієнт-серверна архітектура. Вона забезпечує необхідний баланс між простотою підтримки та продуктивністю модулів логістики, картографії та диспетчеризації. Відокремлення інтерфейсу користувача від серверної частини дозволяє створити захищене хмарне ядро, яке централізовано обробляє запити від численних точок доступу гідів і диспетчерів [9].

При проєктуванні клієнтської частини веб-додатків розглядають два основні патерни організації інтерфейсу – багатосторінкові (MPA) та односторінкові (SPA) застосунки. Традиційний підхід багатосторінкових додатків базується на класичній схемі, де кожна дія користувача або перехід за посиланням ініціює запит до сервера, який генерує нову HTML-сторінку та повертає її в браузер [10].

Головним архітектурним недоліком MPA для динамічних систем є необхідність повного перезавантаження сторінки при будь-якій взаємодії з інтерфейсом. Це створює надлишковий мережевий трафік, оскільки браузер змушений повторно завантажувати однакові блоки, та суттєво уповільнює відгук системи. Для рішень з інтерактивними елементами, такими як календарі розкладу чи графіки завантаженості персоналу, ручний вибір фільтрів або зміна дати в MPA супроводжувалися б постійними затримками та миготінням екрана,

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

що значно погіршує користувацький досвід і знижує продуктивність роботи оператора.

Сучасною альтернативою є патерн односторінкових додатків, який кардинально змінює логіку взаємодії користувача з вебресурсом. При використанні SPA браузер завантажує з сервера лише одну базову HTML-сторінку разом із необхідними скриптами та стилями, а всі подальші оновлення інтерфейсу відбуваються динамічно без перезавантаження всього екрана [11]. Клієнтська частина самостійно формує інтерфейс, обмінюючись із сервером виключно чистими даними через асинхронні фонові запити до API. Такий підхід забезпечує миттєву реакцію системи на дії користувача – перемикання між місячним та тижневим режимами відображення календаря, відкриття карток турів та додавання нових бронювань відбуваються плавно і без затримок. Вебдодаток працює швидко та ефективно навіть на мобільних пристроях гідів, створюючи відчуття роботи з повноцінним локальним нативним застосунком.

Порівняльний аналіз цих підходів чітко вказує на те, що для реалізації проєктованої CRM-системи концепція МРА є неоптимальною. Вибір архітектури SPA є кращим рішенням, оскільки вона повністю задовольняє вимоги щодо динамічності, швидкодії та зручності використання інтерфейсу.

При розробці серверної частини сучасних інформаційних систем вибір архітектурного патерну визначає гнучкість, швидкість розробки та простоту супроводу програмного коду. У сучасній практиці проєктування виділяють кілька підходів до організації бізнес-логіки, серед яких найбільш відомими є чиста архітектура та шарова архітектура. Концепція чистої архітектури (Clean Architecture) передбачає повну незалежність ядра системи від зовнішніх фреймворків та баз даних, де всі залежності суворо спрямовані всередину [12]. Попри високий рівень ізоляції, такий підхід значно ускладнює початкову структуру коду, вимагає створення великої кількості дублюючих моделей-маперів між шарами та створює надлишкове інфраструктурне навантаження, що не завжди є виправданим для проєктів.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Більш збалансованим та перевіреним часом рішенням для корпоративних систем є класична шарова архітектура (Layered Architecture) [12]. Її суть полягає в послідовному горизонтальному розподілі компонентів додатка на автономні рівні, кожен з яких відповідає за свій етап обробки даних – представлення (Presentation), бізнес-логіку (Business Logic) та доступ до даних (Data Access). Головними перевагами шарової архітектури є висока швидкість написання коду, чітке розмежування обов'язків між класами, легкість масштабування окремих рівнів та простота навігації за структурою папок. Хоча цей підхід і несе в собі теоретичний ризик сильної зв'язності через лінійну залежність шарів згори донизу, цей недолік повністю нівелюється за рахунок проєктування системи через абстрактні інтерфейси, що дозволяє ізолювати компоненти один від одного.

Для надійної та ефективної реалізації бізнес-правил в серверній частині обрано шарову архітектуру. Логіка додатка базується на взаємодії кількох рівнів, де точкою входу є Presentation Layer (шар представлення) у вигляді вебконтролерів, які обробляють HTTP-запити від клієнтського SPA-застосунку та повертають відповіді через RESTful API [13]. Ці запити передаються в Business Logic Layer, представлений сервісами додатка, де зосереджена вся ключова логіка системи – від автоматичного підрахунку робочих годин гіда до координації графіків і перевірки накладання турів. Самі сервіси діють як оркестратори процесів і залишаються повністю відокремленими від мережевих протоколів. Робота з інформацією покладена на Data Access Layer, який інкапсулює взаємодію з базою даних. Репозиторії приховують запити до таблиць і надають вищому шару сервісів лише чисті методи для отримання та збереження сутностей. Таке розділення дозволяє гнучко змінювати структуру бази даних без ризику руйнування головної бізнес-логіки додатка.

При проєктуванні мультиорганізаційних інформаційних систем розглядають три основні стратегії організації баз даних – архітектуру з окремою базою для кожного клієнта, спільну базу з різними логічними схемами та спільну базу зі спільною схемою. Перший підхід гарантує максимальну фізичну ізоляцію

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

даних, проте є фінансово витратним та складним в адмініструванні через необхідність обслуговування великої кількості окремих сховищ. Використання різних схем у межах однієї бази виступає компромісним варіантом, який, однак, має обмежену гнучкість при масштабуванні платформи. Найбільш ефективною для нашого сервісу є стратегія спільної бази та спільної схеми, де дані всіх організацій зберігаються в єдиних таблицях, а логічний розподіл інформації реалізується через обов'язкове введення унікального ідентифікатора компанії для кожного запису [14].

Для реалізації цього проєкту було обрано саме стратегію спільної бази та спільної схеми на основі Microsoft SQL Server з єдиним контекстом. Такий вибір є оптимальним, оскільки він мінімізує витрати на серверні ресурси та спрощує оновлення структури даних за допомогою автоматичного застосування міграцій до єдиної бази при запуску додатка. Безпека та строга ізоляція інформації забезпечуються на рівні кожного запиту – під час автентифікації користувача система зчитує унікальний ідентифікатор companyId із JWT-токена та за допомогою фільтрації на рівні доступу до даних обмежує вибірку. Як наслідок, менеджери та гіді отримують доступ виключно до сутностей своєї організації.

Взаємодія між клієнтською частиною та сервером побудована за принципами архітектурного стилю REST за допомогою стандартних HTTP-методів [13]. Для забезпечення надійного захисту внутрішніх доменних моделей і сутностей бази даних обмін інформацією між архітектурними рівнями відбувається виключно через об'єкти передачі даних. Використання DTO дозволяє повністю відірвати структуру вебсторінок від схеми бази даних, інкапсулюючи лише ті поля, які необхідні для конкретного запиту чи відображення на фронтенді. Використання таких моделей запобігає випадковому витоку конфіденційної інформації, оскільки сутності з бази даних автоматично трансформуються в безпечні DTO перед відправкою клієнту [15].

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4. Обґрунтування вибору інструментальних засобів

Для розробки серверної частини було обрано кросплатформену платформу .NET 9 та фреймворк ASP.NET Core Web API. Такий вибір обґрунтований високими показниками продуктивності модулів компіляції, оптимізованим базовим вебсервером Kestrel та вбудованою підтримкою асинхронності на рівні ключових слів `async/await`, що забезпечує неблокуючу обробку великої кількості одночасних запитів від користувачів. Важливою перевагою платформи є її розвинена екосистема для побудови захищених корпоративних рішень, зокрема наявність гнучкого вбудованого контейнера інверсії керування (IoC) для реалізації патерна впровадження залежностей (Dependency Injection). Це дозволяє легко керувати життєвим циклом створюваних бізнес-сервісів, репозиторіїв та контекстів через конфігурацію режимів `Transient`, `Scoped` та `Singleton`. Окрім цього, модульна архітектура конвеєра обробки HTTP-запитів дозволяє інтегрувати централізовані компоненти для глобального перехоплення винятків, логування операцій, автентифікації користувачів та гнучкого налаштування правил спільного використання ресурсів різних джерел (CORS) [16].

Роль системи керування базами даних виконує Microsoft SQL Server. Реляційна модель цієї СКБД забезпечує надійне збереження структурованих даних із чіткими зв'язками між компаніями, менеджерами, турами та гідами, а суворе дотримання транзакційних принципів ACID гарантує абсолютну цілісність комерційної інформації при одночасному бронюванні місць кількома клієнтами [17].

Взаємодія між шаром бізнес-логіки та базою даних реалізована за допомогою об'єктно-реляційного відображення на базі фреймворку Entity Framework Core. Використання підходу Code-First дозволило проектувати архітектуру таблиць безпосередньо через мову C# за допомогою декларативного опису моделей сутностей, вбудований інструмент міграцій забезпечив автоматичне створення та оновлення схеми бази даних. Особливе місце в архітектурі доступу до даних посідає реалізація концепції мультитенантності для

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

логічного розмежування корпоративних клієнтів. Через перевизначення конфігурацій у методі OnModelCreating контексту бази даних для всіх сутностей може бути впроваджено глобальні фільтри запитів за допомогою конструкції HasQueryFilter. Це гарантує, що будь-яка операція вибірки автоматично містить умову фільтрації за ідентифікатором поточної компанії, повністю ізолюючи дані на рівні інфраструктури. Завдяки інтеграції фреймворку, усі операції пошуку, фільтрації та пагінації даних виконуються на стороні сервера за допомогою об'єктно-орієнтованих запитів LINQ. Це дозволяє повністю абстрагувати код сервісів від написання низькорівневих SQL-скриптів, мінімізує ризик виникнення помилок та суттєво підвищує загальну безпеку роботи з даними [18].

Для реалізації клієнтської частини обрано бібліотеку React, яка базується на концепції компонентного інтерфейсу. Такий підхід дозволяє розділити графічну оболонку додатка на незалежні блоки, що повторно використовуються, спрощуючи підтримку та подальший розвиток коду. Головною технологічною перевагою React є використання механізму віртуального DOM, який мінімізує кількість прямих маніпуляцій з елементами сторінки браузера. Це забезпечує миттєвий рендеринг та плавне перемикавання режимів у складних інтерактивних календарних сітках розкладу екскурсій для менеджерів та гідів [19]. Розробка ведеться з використанням мови програмування TypeScript. Впровадження строгої статичної типізації є повністю виправданим для масштабних інтерфейсів, оскільки воно дозволяє виявляти потенційні помилки ще на етапі компіляції. Використання кастомних інтерфейсів та типів, які чітко повторюють структуру серверних DTO, забезпечує зручне автодоповнення в середовищі розробки, спрощує рефакторинг великих модулів та гарантує точну відповідність обмінних моделей даним, що надходять із серверного API [20].

Візуальна частина та користувацький інтерфейс додатку побудовані на основі популярної бібліотеки готових компонентів Material UI. Вибір цього інструменту дозволив створити лаконічний дизайн у стилі корпоративних CRM-систем, а також гарантує мобільну адаптацію вебсторінок для зручної роботи гідів із особистим графіком безпосередньо на екранах смартфонів [21].

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Для організації внутрішньої навігації та побудови маршрутів без перезавантаження сторінки інтегровано бібліотеку React Router. Окрім забезпечення переходу між сторінками застосунку, цей інструмент виступає елементом системи безпеки клієнтської частини, оскільки дозволяє організувати контроль доступу на основі ролей. Завдяки цьому додаток динамічно обмежує доступ до певних екранів та функцій, повністю приховуючи адміністративні модулі управління турами чи бронюваннями від користувачів із роллю гіда [22].

Валідація та трансформація обмінних моделей у серверній частині автоматизована за допомогою інструментів FluentValidation та AutoMapper. Декларативний підхід FluentValidation забезпечує перевірку об'єктів передачі даних на рівні контролерів перед їх безпосередньою обробкою в шарі бізнес-сервісів. Написання окремих профілів валідації дозволило відокремити бізнес-правила перевірки вхідних параметрів від безпосередньої логіки додатку. Більше того, використання користувацьких правил валідації дало змогу реалізувати перевірки складних логічних умов, наприклад, відповідності формату годин роботи пам'яток або коректності часових діапазонів турів. У разі виявлення помилок валідації додаток автоматично перериває виконання запиту та повертає клієнту структуровану відповідь із кодом 400 Bad Request, що дозволяє повністю відокремити логіку перевірки від основних бізнес-сценаріїв [23].

Для оптимізації конвертації сутностей бази даних у вихідні DTO інтегровано бібліотеку AutoMapper. Вона автоматизує рутинний мапінг об'єктів у сервісах, усуваючи потребу ручного копіювання властивостей. Використання методу ProjectTo всередині асинхронних запитів LINQ дозволяє AutoMapper транслювати структуру вихідного DTO безпосередньо у вибірку SELECT бази даних, завдяки чому з СКБД витягуються лише ті стовпці, що потрібні для відображення, значно економиться оперативна пам'ять та знижується навантаження на мережу [24].

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 1

У першому розділі проведено аналіз предметної області та існуючих аналогів, на основі чого формалізовано вимоги до інформаційної системи планування екскурсійних маршрутів. Для реалізації проєкту обґрунтовано вибір розподіленої клієнт-серверної архітектури з інтерактивним односторінковим інтерфейсом, реалізованим із використанням React, TypeScript та Material UI, та серверним ядром на платформі .NET 9, спроектованим за шаровою моделлю для чіткого поділу Presentation, Business Logic і Data Access рівнів. Організацію збереження інформації вирішено здійснити за стратегією спільної бази даних зі спільною схемою під керуванням Microsoft SQL Server з єдиним контекстом, що дозволяє централізовано підтримувати структуру геопросторових даних, моделей екскурсійних маршрутів і реєстрів груп, водночас реалізуючи логічну ізоляцію через глобальні фільтри за ідентифікатором організації на рівні запитів. Інтеграція безстанової автентифікації на основі JWT, обміну даними через DTO та декларативної валідації запитів за допомогою FluentValidation забезпечила технологічний фундамент для координації роботи гідів і менеджерів, спрощуючи подальший розвиток системи у бік розширення логістичних алгоритмів, контролю ресурсної місткості груп і вдосконалення картографічних сервісів.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Важливим етапом проєктування інформаційної системи планування екскурсійних маршрутів є чітка специфікація її функціоналу. На основі аналізу технічних вимог та проєктованої архітектури додатка визначено, що система має передбачати дві основні автентифіковані ролі користувачів із розмежуванням прав доступу – менеджер (Manager) та гід (Guide). Крім того, у моделі передбачено роль «гість» як окремий, але обмежений тип користувача. Він розглядається як користувач системи, який не має повноцінної автентифікації та для якого на початковому етапі передбачено лише операції створення облікового запису або авторизації на платформі з подальшим переходом у одну з авторизованих ролей.

Для забезпечення систематизації та відображення функціональних вимог до проєктованого програмного комплексу всі обов'язки та доступні сценарії взаємодії розподілено відповідно до ролей. Детальний перелік можливостей, що охоплюють процеси від реєстрації та автентифікації до операційного менеджменту й перегляду особистих графіків, наведено у таблиці 2.1.

Таблиця 2.1

Специфікація функціональних вимог системи за ролями користувачів

Роль користувача	Функціональні вимоги
<i>Неавтентифікований користувач</i>	Реєстрація профілю нового менеджера одночасно зі створенням нової компанії у системі.
	Приєднання менеджера до вже створеної туристичної компанії за її унікальним кодом.
	Налаштування пароля для активації облікового запису гіда.
	Автентифікація в системі за допомогою email/пароля та отримання токена сесії.
<i>Гід</i>	Доступ до персональної головної панелі управління для моніторингу поточного стану роботи.
	Перегляд та облік відпрацьованого часу й моніторинг ліміту робочих годин.
	Перегляд особистого інтерактивного розкладу та календаря призначених екскурсій.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

<i>Гід</i>	Завершення робочої сесії з автоматичним очищенням токена з локального сховища.
	Перегляд повного списку та детальної інформації про закріплені за ним тури.
	Перегляд особистих даних, верифікація ролі та поточних параметрів профілю користувача.
<i>Менеджер</i>	Моніторинг загальної діяльності фірми через головну панель з інтегрованим календарем.
	Повне керування каталогом пам'яток та локацій для екскурсій.
	Повне керування шаблонами екскурсійних турів із налаштуванням зв'язків.
	Управління базою гідів, перегляд їхніх профілів та встановлення лімітів часу.
	Планування розкладу, формування сітки запланованих турів та призначення гідів на дати.
	Керування замовленнями, створення та обробка заявок клієнтів із контролем вільних місць.
	Перегляд зведеної статистики підприємства.
	Вихід з акаунту, перегляд профілю або деактивація акаунта.

Такий розподіл функціональних обов'язків закладає основу для подальшого проектування структури бази даних та інтерфейсу користувача. Розмежування сценаріїв дозволяє забезпечити ізоляцію прав доступу на рівні логіки додатка, гарантуючи, що персонал через відокремлений портал матиме доступ лише до власного робочого розкладу, турів та обліку годин, тоді як менеджер володітиме повним адміністративним інструментарієм для координації діяльності всього туристичного підприємства.

На відміну від функціональних вимог, які визначають безпосередній набір операцій для користувачів, нефункціональні вимоги встановлюють якісні, архітектурні та інфраструктурні обмеження системи. Вони регламентують параметри безпеки, надійності, масштабованості та доступності додатку. Технічну специфікацію нефункціональних вимог до розроблюваного програмного комплексу наведено у таблиці 2.2.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.2

Специфікація нефункціональних вимог

Категорія вимог	Зміст та технічна специфікація вимоги
Продуктивність	Швидке відображення сторінок вебдодатка, зокрема основного робочого простору менеджера. Оптимізація обробки бізнес-сервісів (перевірка часових конфліктів, лімітів годин персоналу) на стороні сервера за рахунок асинхронності операцій.
Надійність	Забезпечення цілодобового режиму роботи платформи з мінімальним часом простою. Безперебійне функціонування серверної частини та реляційного сховища даних за умов постійного доступу.
Безпека	Захист доступу через автентифікацію на основі токенів JWT з HMAC-SHA256 підписом, що передаються в заголовках запитів. Розмежування прав доступу на рівні користувацьких ролей за допомогою ізольованих маршрутів та серверних атрибутів. Передача інформації виключно через захищені канали зв'язку.
Масштабованість	Здатність системи стабільно обробляти лінійне зростання кількості екскурсій, записів у розкладі та бронювань від багатьох компаній одночасно без втрати продуктивності та деградації швидкості обробки транзакцій.
Кросплатформеність та адаптивність	Коректне відображення й стабільна робота інтерфейсу в усіх сучасних веб-оглядачах з підтримкою HTML5. Адаптація UI під різні екрани: широкі таблиці та розклади для десктопних ПК, компактні форми та прихована в бічне меню навігація для мобільних пристроїв.

Важливим компонентом нефункціонального контуру є часові характеристики реакції інтерфейсу та серверної частини на дії менеджерів і гідів. Оскільки архітектура системи розробляється як односторінковий застосунок, що взаємодіє з сервером через асинхронні запити, швидкість обробки інформації варіюється залежно від апаратної платформи. Порівняльні часові обмеження для десктопної та мобільної версій системи деталізовано у таблиці 2.3.

Таблиця 2.3

Вимоги до часових характеристик реакції системи

Тип операції / Сценарій взаємодії	Граничний час (ПК)	Граничний час (Мобільна версія)
Натискання інтерактивних кнопок (збереження створеного туру або реєстрація клієнтського бронювання)	Менше 1 секунди	
Перехід між екранами системи (внутрішня маршрутизація всередині SPA без перезавантаження сторінки)	Менше 1 секунди	

Фільтрація списків даних та виконання пошукових запитів (пошук вільних гідів на обрану дату)	Менше 2 секунд	
Початкове завантаження сторінки інтерактивного календаря розкладу разом із масивом даних від API	До 3 секунд	До 4 секунд
Завантаження статичних файлів із сервера (обробка та відображення зображень локацій і фото персоналу)	Менше 2 секунд	Менше 3 секунд
Пряма взаємодія з динамічним інтерфейсом (плавність прокручування календаря, відкриття бічного меню)	Без видимих затримок	

Зафіксовані часові рамки досягаються за рахунок локальної валідації форм на стороні клієнта перед відправкою HTTP-запитів, використання механізмів віртуального DOM для рендерингу складних сіток графіків та побудови оптимізованої централізованої функції виконання запитів, що забезпечує безшовну роботу користувачів незалежно від типу їхнього робочого пристрою.

Для формалізації вимог та візуалізації сценаріїв взаємодії користувачів із мультитенантною системою розроблено діаграму варіантів використання відповідно до стандарту UML 2.0. Побудову схеми виконано за допомогою інструменту моделювання Mermaid [25]. Модель чітко розмежовує область дії системи на три ізольовані функціональні контури, які охоплюють сторінку авторизації, спеціалізований портал гіда та адміністративну панель менеджера, а також визначає особливості взаємодії з ними трьох типів дійових осіб.

Першим типом користувачів є Гість, що виступає як неавторизований суб'єкт, чия взаємодія з платформою обмежена виключно процесами реєстрації нової компанії, приєднання до існуючої структури за генерованим кодом та проходження первинної автентифікації для входу в систему. Другу групу дійових осіб становить Гід – співробітник, який комунікує з відокремленим індивідуальним порталом для ведення обліку особистого робочого часу, перегляду закріплених за ним турів та моніторингу інтерактивних календарних графіків. Головним керуючим суб'єктом системи є Менеджер, який володіє правами адміністратора та має повний доступ до аналітичних панелей,

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

інструментів операційного планування розкладу, а також до модулів повного управління каталогами пам'яток, шаблонами турів та базою гідів підприємства.

Графічне представлення структури зв'язків між акторами та прецедентами наведено на рисунку 2.1.

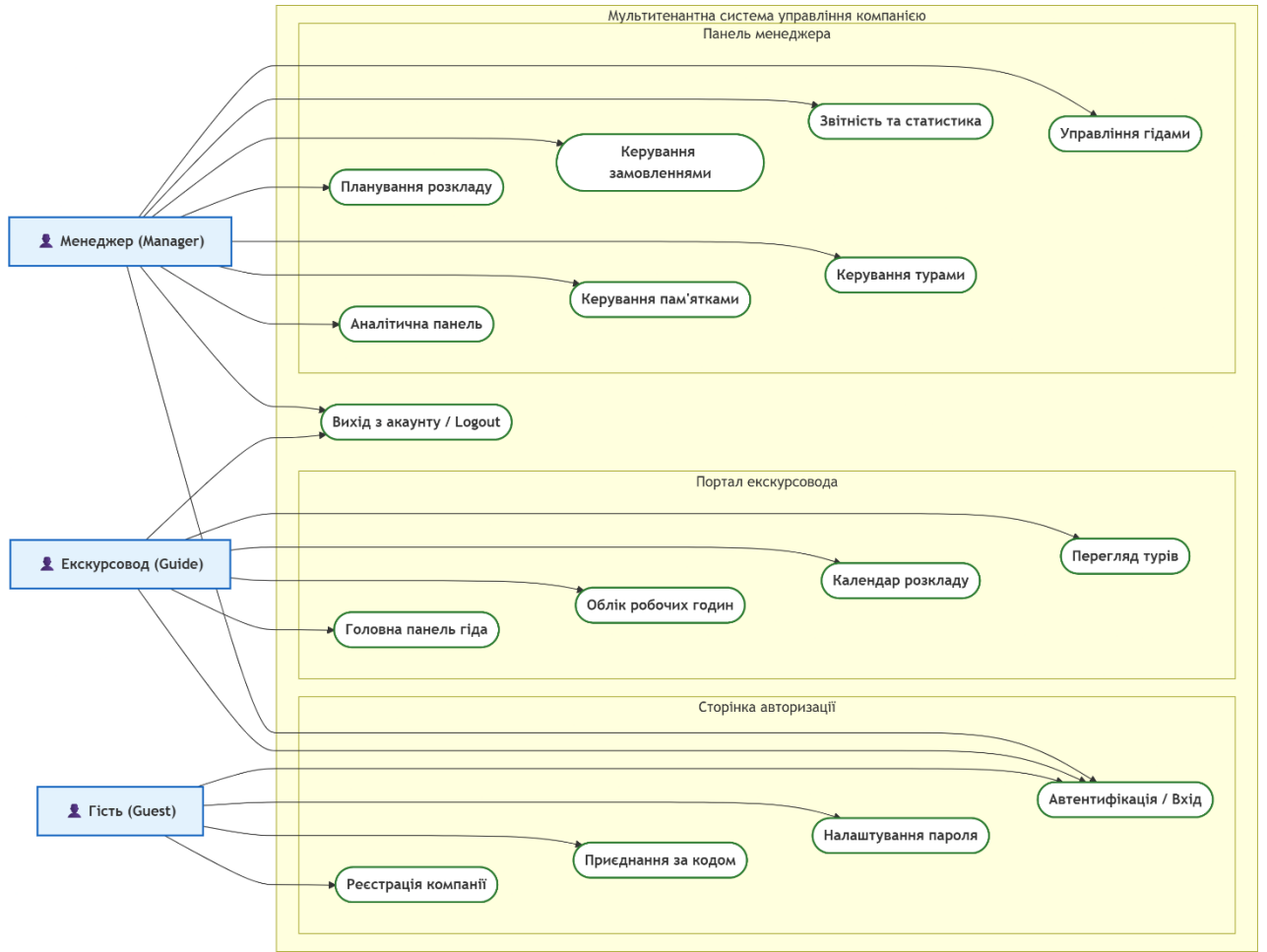


Рис. 2.1. Діаграма варіантів використання

Проектування об'єктно-орієнтованої структури системи базується на принципах розділення відповідальності та типізації доменних моделей додатка. В основі архітектури лежить абстрактний базовий клас BaseEntity, який інкапсулює унікальний ідентифікатор об'єкта Id та ідентифікатор організаційного контексту CompanyId. Цей клас виступає єдиним предком для ключових системних компонентів і забезпечує наскрізну логіку логічного відокремлення даних комерційних фірм. Від нього безпосередньо успадковуються класи токенів безпеки AuthToken та абстрактний клас користувача User. Ієрархія користувачів розширюється за допомогою механізму успадкування, де від базової сутності User, яка містить загальні персональні дані,

походять дві конкретні реалізації об'єктів із власним бізнес-призначенням – менеджер Manager та гід TourGuide. Склад атрибутів сутностей, їхні внутрішні методи та характери зв'язків між ними графічно відображено на діаграмі класів (рис. 2.2).

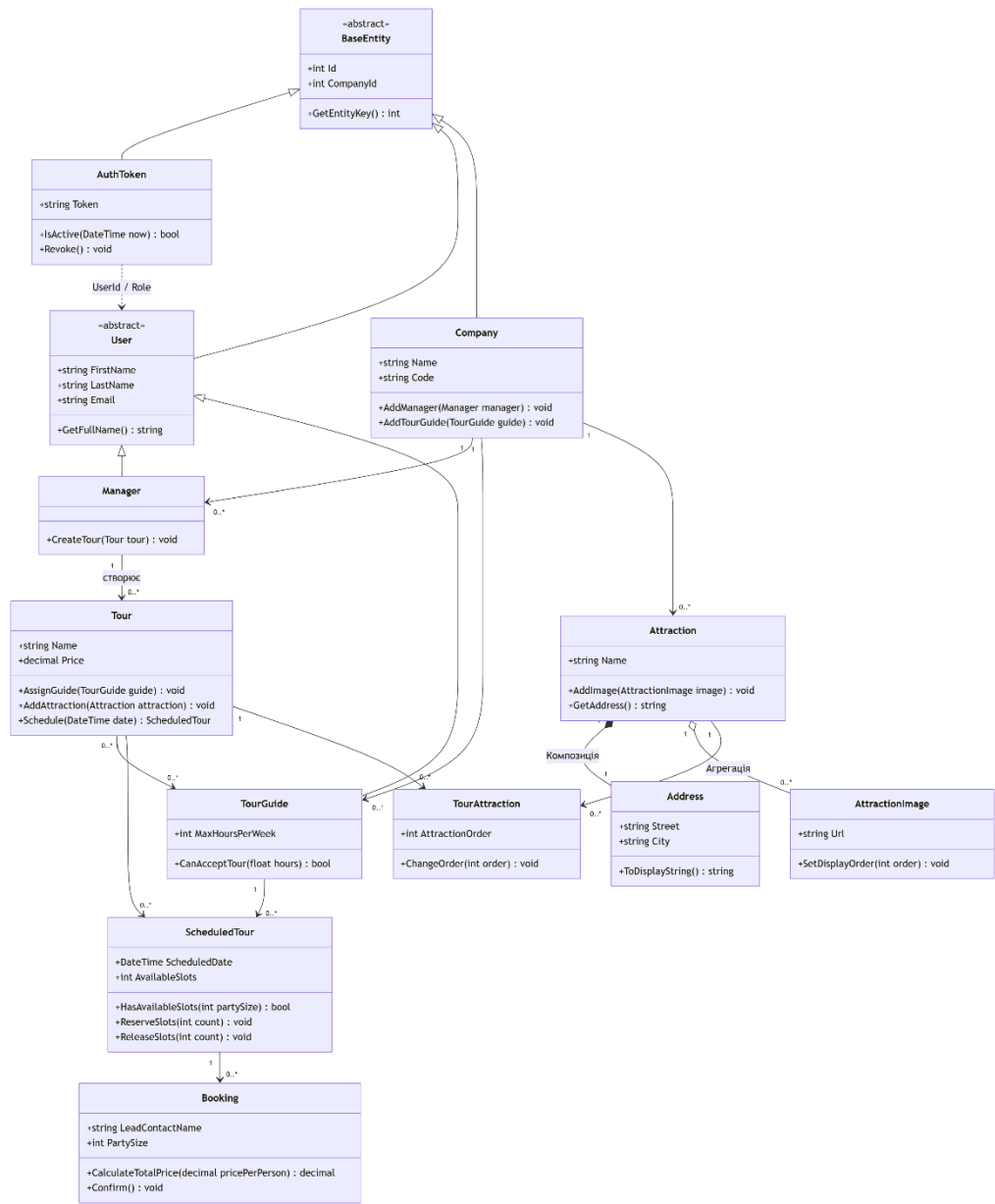


Рис. 2.2. Об'єктно-орієнтована діаграма класів

Організаційний блок системи безпосередньо відповідає за забезпечення бізнес-процесів реєстрації, автентифікації та адміністрування структури підприємства. Сутність Company виступає центральним елементом організації, містить атрибути назви й коду доступу, а також інкапсулює методи для динамічної генерації унікальних шестизначних шифрів приєднання, додавання керівників чи інших працівників. Абстрактний клас User об'єднує дані

користувача, такі як ім'я, прізвище та електронна адреса і надає метод для автоматичного формування повного імені працівника. Сутність Manager розширює базову модель користувача та володіє методом для створення нових екскурсійних турів у системі. Об'єкт гідів TourGuide містить специфічні параметри максимального тижневого навантаження, а його внутрішній метод дозволяє валідувати доступність графіка при спробах призначення на нові дати. Безпека сесій контролюється сутністю AuthToken, яка зберігає текстове значення JWT, статус токена, часові мітки його випуску й ліміту дії, а також надає методи для перевірки активності або примусового відкликання прав доступу при виході з акаунту.

Операційні сутності системи забезпечують моделювання діяльності екскурсійного бюро та облік клієнтських заявок. Шаблон туристичного маршруту описується сутністю Tour, яка фіксує назву та базову вартість, а також володіє методами призначення гідів, прив'язки пам'яток та генерації конкретних часових екземплярів у розкладі. Об'єкти відвідування описуються класом Attraction, який має композиційний зв'язок із сутністю Address для фіксації міста й вулиці, а також агрегує сутності AttractionImage для збереження посилань на медіафайли. Модель TourAttraction – модель екскурсійного маршруту, спрямований граф локацій, у якому поле Order виступає індексом черговості відвідувань і служить лінійним валідатором послідовності руху між вузлами; технічні деталі збереження порядку залишаються у специфікації збереження даних, водночас предметна інтерпретація підкреслює роль TourAttraction у формуванні логістично коректного маршруту. Безпосередній календарний запуск туру в часі контролюється сутністю ScheduledTour, яка містить дату проведення, параметри поточної місткості та механізми верифікації групової місткості, а також методи розрахунку, резервування та звільнення місць. Фінальним елементом цієї операційної структури є сутність Booking, яка фіксує ім'я замовника, розмір групи й підсумкову вартість і володіє методами перевірки коректності параметрів та фінансового розрахунку вартості на основі базового тарифу туру.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Логіка взаємодії елементів об'єктної структури реалізована за допомогою типів зв'язків, що гарантують цілісність даних на доменному рівні. Відношення один-до-багатьох пов'язують сутність компанії з менеджерами, гідами й пам'ятками, а також кожен сутність Tour чи TourGuide із множиною об'єктів запланованих у розкладі екскурсій. Аналогічний тип зв'язку діє між конкретним календарним запуском ScheduledTour та зареєстрованими на нього клієнтськими бронюваннями. Відношення багато-до-багатьох виникають між сутностями турів та гідів для відображення професійних компетенцій персоналу, а також між шаблонами турів та визначними пам'ятками. Останній зв'язок декомпоновано на два послідовні зв'язки один-до-багатьох через проміжну доменну модель TourAttraction, що дозволяє зберігати унікальні атрибути сформованого маршруту, зокрема точний індекс черговості відвідування локацій. Вся логіка взаємодії об'єктів спроектована таким чином, що зміна стану однієї сутності, наприклад створення бронювання, автоматично викликає каскадне виконання методів суміжних сутностей, забезпечуючи динамічний перерахунок доступних місць та запобігаючи виникненню конфліктів у системі.

2.2. Розробка бази даних системи

Обґрунтування вибору реляційного підходу під керуванням СКБД Microsoft SQL Server зумовлене потребою в забезпеченні цілісності, строгої типізації та надійності збереження структурованої інформації. Проектування архітектури реляційного сховища спирається на принципи нормалізації даних, що передбачають послідовне приведення таблиць до третьої нормальної форми (3NF) з метою повної ліквідації надлишковості інформації, запобігання виникненню аномалій модифікації, видалення чи вставки, а також оптимізації виконання декомпонованих SQL-запитів [26]. Платформа Microsoft SQL Server забезпечує повну підтримку транзакційних гарантій, володіє оптимізатором виконання запитів під навантаженням та надає вбудовані декларативні інструменти перевірки обмежень.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Архітектура програмного комплексу як веб-орієнтованого комплексу диспетчеризації маршрутів передбачає стратегію спільної бази даних та спільної схеми з логічною ізоляцією даних окремих замовників на рівні реляційної моделі. Ізоляція реалізується шляхом обов'язкового включення в структури базових сутностей ідентифікатора організації як зовнішнього ключа до всіх таблиць, які містять дані про користувачів, маршрути, розклади та записи координації дій гідів; такий підхід спирається на механізми посилювальної цілісності, де обмеження зовнішніх ключів, реалізовані на рівні СКБД, контролюють зв'язки між записами, запобігають випадковому змішуванню даних різних замовників і є базою для накладання глобальних фільтрів на рівні серверних сервісів, що забезпечує автоматичне обмеження операцій вибірки і модифікації в межах дозволеної контекстної області автентифікованого користувача [27].

Фізична схема даних відображає кінцеву структуру сховища та визначає точні типи даних, системні ідентифікатори та обмеження на рівні бази даних під керуванням. Графічна діаграма, що деталізує повнотекстовий склад полів, первинних, зовнішніх та унікальних ключів кожної таблиці розроблюваної системи, показана на рис. 2.3.

Аналіз логічних зв'язків дозволяє простежити шлях формування цілісності даних між елементами системи. В організаційному блоці зв'язок між таблицями реалізовано за первинними ключами COMPANIES та USERS, які входять як зовнішні ключі до таблиць MANAGERS та TOUR_GUIDES. Це формує відношення один-до-багатьох для гнучкого відображення корпоративних ролей користувачів у межах конкретних підприємств. Крім того, зв'язок таблиці AUTH_TOKENS із базовою таблицею USERS за первинним ключем забезпечує приналежність токенів до відповідних суб'єктів системи.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

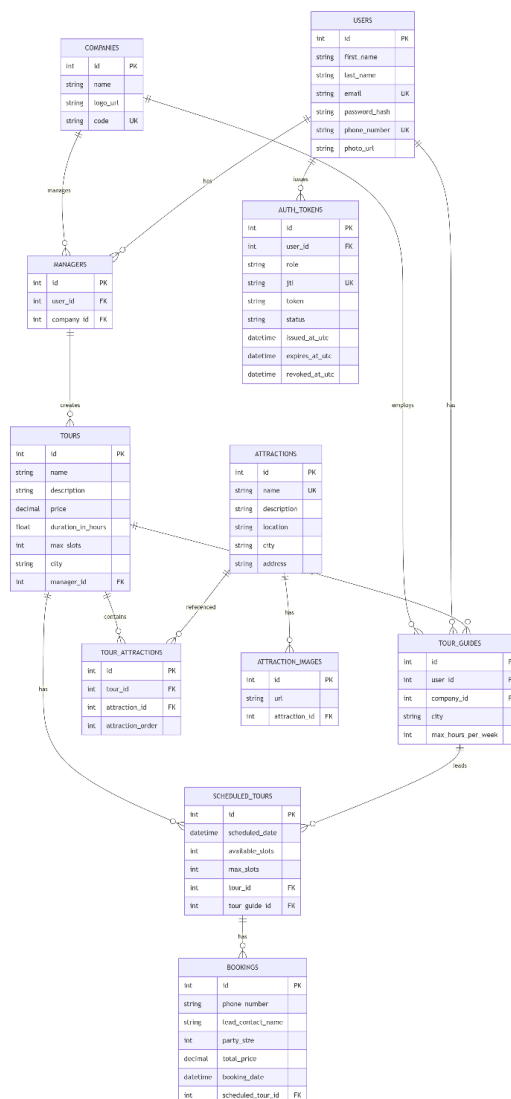


Рис. 2.3. Діаграма бази даних системи

Операційний сегмент бази даних спрямований на фіксацію бізнес-логіки планування та логістичної координації маршрутів, причому таблиця шаблонів маршрутів TOURS пов'язана за первинним ключем із таблицею MANAGERS через зовнішній ключ manager_id, що закріплює відповідальність за створення й оновлення контенту за конкретним керівником. Модель екскурсійного маршруту реалізована не як проста сполучна таблиця, а як семантично насичена таблиця TOUR_ATTRACTIONS, яка конструює спрямований граф локацій та містить зовнішні ключі на відповідні сутності локацій і маршруту разом із полем Order, яке забезпечує лінійну валідацію логістичної послідовності відвідувань; такий підхід дозволяє формалізувати топологію маршруту, виявляти та унеможливлювати просторові і часові розриви між послідовними точками, а також створює фундамент для алгоритмічної оптимізації порядку відвідувань з

урахуванням географічних координат і часових обмежень. Формування розкладу залишається прив'язаним до зв'язків таблиць TOURS та TOUR_GUIDES із таблицею SCHEDULED_TOURS через зовнішні ключі tour_id та tour_guide_id, які визначають момент запуску екскурсії в часі та відповідального гіда, а зв'язок таблиці BOOKINGS із сутністю SCHEDULED_TOURS за зовнішнім ключем scheduled_tour_id прив'язує кожне замовлення до конкретної події в інтерактивному календарі, забезпечуючи цілісність подій і підставу для роботи автоматизованої підсистеми верифікації ресурсної місткості груп і алгоритмів запобігання логістичним колізіям та контролю овертаймів персоналу.

Цілісність у схемі бази даних забезпечується набором декларативних правил каскадної обробки видалень і заборон, що автоматично підтримують консистентність інформаційної системи планування екскурсійних маршрутів при видаленні сутностей. Упроваджено механізми, які при видаленні облікового запису користувача або компанії гарантують послідовне видалення залежних профілів, шаблонів маршрутів та пов'язаних сесійних даних, тоді як для зв'язків, що фіксують відповідальність менеджера за маршрут, передбачено блокування видалення до моменту явної передачі відповідальності. Така конфігурація обмежень мінімізує ризик появи ізольованих або несумісних записів, зберігає цілісність розкладу і бронювань та слугує інженерним захистом логіки маршрутоутворення.

Унікальні обмеження застосовані до структури таблиць як механізм захисту від дублювання інформації та забезпечення бізнес-правил на рівні збереження даних. Впровадження обмежень унікальності UK для полів email та phone_number у таблиці USERS унеможлиблює повторну реєстрацію користувачів із однаковими контактними даними. У таблиці COMPANIES унікальним визначено поле code, що гарантує точність автентифікації та правильність адресації даних під час приєднання нових гідів до конкретної фірми. Для запобігання створенню ідентичних екскурсійних місць обмеження накладено на поле name у таблиці ATTRACTIONS. Також унікальний індекс застосовано до поля jti в таблиці AUTH_TOKENS для виключення дублювання

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

ідентифікаторів сесій. Створення таких обмежень слугує вторинним рівнем захисту.

Оптимізація швидкості виконання пошукових запитів та операцій фільтрації є ключовою вимогою до системи та досягається за рахунок розробки збалансованої стратегії індексації таблиць. Механізм індексування передбачає створення додаткових внутрішніх структур даних, які діють подібно до алфавітного покажчика в книзі, які дозволяють СКБД знаходити потрібні рядки за вказаними вказівниками замість виконання високовартісного повного сканування всієї таблиці [28]. Оскільки створення великої кількості індексів прискорює операції читання (SELECT), але створює додаткове навантаження на дискову підсистему та уповільнює операції модифікації (INSERT, UPDATE, DELETE) через необхідність постійного оновлення індексних дерев, у проекті застосовано строго збалансований підхід.

Окрім стандартних кластеризованих індексів, що формуються для первинних ключів, передбачено створення додаткових некластеризованих індексів на полях, які найчастіше використовуються при відборі та з'єднанні записів із метою оптимізації запитів логістичного та картографічного підсистем інформаційної системи планування екскурсійних маршрутів. Передусім індексу підлягає поле ідентифікатора контексту розгортання (ідентифікатор замовника) у операційних таблицях, оскільки більшість запитів до мультіарендного середовища містять фільтр за цим атрибутом; додаткові індекси застосовано до поля `scheduled_date` таблиці `SCHEDULED TOURS` для прискореного рендерингу розкладу та агрегації подій, а також до полів `tour_id` і `tour_guide_id` для оптимізації сервісних перевірок часових конфліктів у графіках гідів і фонового валідування маршрутної послідовності. Така індексаційна стратегія спрямована на мінімізацію часу відгуку СКБД та запобігання деградації продуктивності веб-орієнтованого комплексу диспетчеризації маршрутів при зростанні обсягів геопросторових і операційних даних.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

2.3. Проектування та реалізація алгоритмів роботи системи

У цьому підрозділі розглядається динамічний аспект функціонування системи, що включає опис узагальнених алгоритмів роботи програмного комплексу та логіку взаємодії його структурних елементів у процесі виконання бізнес-операцій. На відміну від статичних моделей, наведених раніше, даний аналіз дозволяє деталізувати послідовність проходження інформаційних потоків між клієнтською та серверною частинами додатка. Опис логічної взаємодії компонентів дає змогу формалізувати процеси прийняття рішень системою, встановити правила обробки користувацьких запитів та визначити механізми забезпечення цілісності даних при виконанні складних операційних сценаріїв, таких як планування екскурсій та управління бронюваннями.

Функціонування програмного комплексу базується на подійній моделі взаємодії між React-інтерфейсом та багатошаровим серверним API. Будь-яка активність користувача, наприклад планування нової екскурсії, ініціює ланцюжок дій, що починається з локальної перевірки даних у браузері та завершується виконанням бізнес-правил на сервері. Клієнтська частина надсилає HTTP-запити з токенами авторизації, які сервер перевіряє на валідність і зіставляє з базою даних для забезпечення можливості миттєвого відкликання доступу.

Центральне місце в логіці системи займає шар сервісів, де виконується перевірка предметних правил, таких як доступність гідів, відсутність часових накладок у розкладі та дотримання лімітів робочого часу. Після проходження всіх перевірок дані передаються через репозиторії до об'єктно-реляційної моделі для фіксації змін у базі даних. У разі виникнення помилок на будь-якому етапі глобальний обробник винятків повертає структуровану відповідь, яку клієнтський додаток трансформує у повідомлення для користувача.

Завершальним етапом взаємодії є синхронізація стану односторінкового додатка з оновленими даними сервера. Після отримання успішної відповіді інтерфейс автоматично ререндериться, відображаючи актуальний стан календаря або списків бронювань без необхідності повного перезавантаження

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

сторінки. Це забезпечує безперервність робочого процесу, високу швидкість реакції системи та цілісність інформаційних потоків між користувачем, логічними компонентами додатка та фізичним сховищем даних.

Ефективність описаної клієнт-серверної взаємодії та коректність відображення даних в інтерфейсі напряду залежать від чіткості виконання внутрішніх алгоритмів системи. Для формалізації цих процесів та забезпечення надійної роботи програмного комплексу було деталізовано ключові сценарії, що охоплюють безпеку доступу та логіку операційної діяльності. Першочерговим серед них є механізм ідентифікації користувача, який виступає фундаментом для всіх подальших транзакцій у системі.

Алгоритм авторизації та розмежування прав базується на перевірці автентичності користувача та наступній видачі токена доступу. Процес розпочинається з передачі облікових даних на сервер. Система проводить пошук відповідного запису в базі даних. У разі успішного збігу генерується унікальний токен, куди записуються ідентифікатор користувача, його роль та код компанії. Водночас запис про виданий токен фіксується у сховищі для контролю сесій, що дозволяє системі при кожному наступному запиті миттєво визначати права доступу користувача до конкретних операційних модулів.

Алгоритм створення бронювання зосереджений на забезпеченні актуальності даних про заповненість груп та запобіганні помилок переповнення. При отриманні заявки від користувача система звертається до конкретного запису в розкладі та проводить порівняльний аналіз кількості вже зарезервованих місць із максимальною місткістю екскурсії. Якщо запит не перевищує доступний ліміт, система виконує атомарне оновлення лічильника вільних місць та фіксує контактні дані замовника. У разі дефіциту місць користувач отримує миттєве сповіщення про неможливість операції.

Алгоритм планування екскурсії є найбільш складним процесом, оскільки він забезпечує цілісність усього операційного розкладу компанії. Система виконує перевірку, яка включає контроль існування обраного маршруту та доступності персоналу. Ключовим аспектом є аналіз часових накладок – система

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

перевіряє, щоб один і той самий маршрут не дублювався у конфліктний час, а гід не мав призначень на два різні заходи одночасно. Окремо розраховується сумарне тижневе навантаження співробітника, щоб не перевищити встановлену норму годин. Алгоритм також враховує технологічні перерви між поїздками, необхідні для відпочинку або переміщення між локаціями. Лише за умови успішного проходження всіх етапів перевірки система дозволяє зафіксувати нову подію в загальному календарі.

Послідовність дій при плануванні екскурсії представлена на діаграмі активності (рис. 2.4).

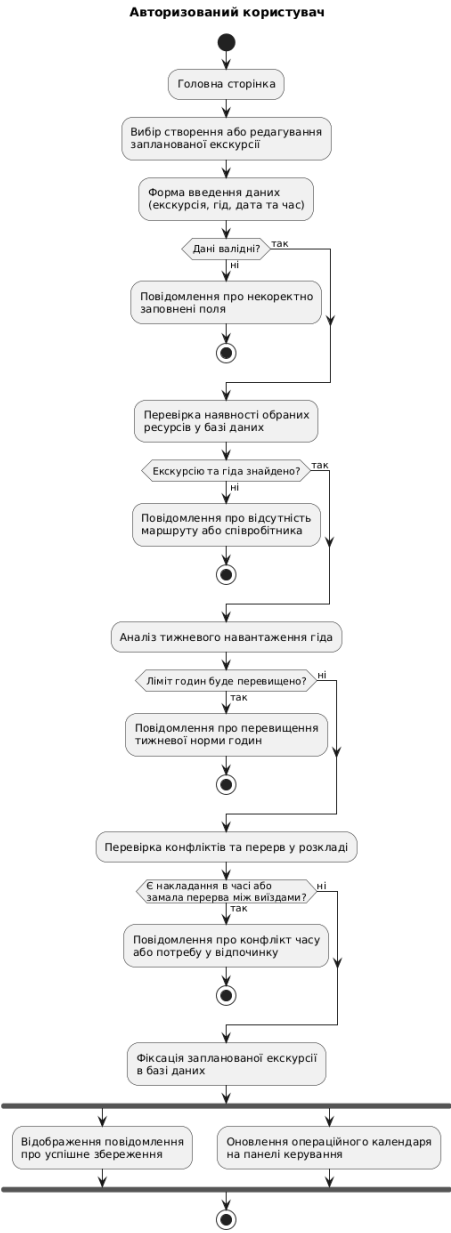


Рис. 2.4. Діаграма активності планування екскурсії

Логічним продовженням опису алгоритмів функціонування є аналіз сценаріїв взаємодії компонентів системи в часі. Окреслена раніше клієнт-серверна архітектура знаходить своє практичне втілення у процесах завантаження даних та керування контентом. Зокрема, при роботі з головною панеллю керування система реалізує сценарій послідовного завантаження операційних даних, де клієнтська частина ініціює запити до API для отримання агрегованої інформації про розклад та бронювання. У відповідь сервер проводить вибірку даних із бази, враховуючи часові межі та ідентифікатор компанії, що дозволяє динамічно формувати календарну сітку та графіки завантаженості без переривання роботи користувача з інтерфейсом.

Більш складним і важливим для системи є процес керування контентом, що охоплює створення та редагування екскурсій або локацій. Цей сценарій демонструє інтеграцію фронтенду, бекенду та БД. Взаємодія починається з передачі даних у форматі DTO або FormData, після чого серверний шар автентифікації перевіряє роль користувача. Лише за умови наявності прав менеджера запит передається до сервісного рівня, де виконується перевірка унікальності назв, існування пов'язаних об'єктів та обробка статичних файлів через спеціалізований сервіс файлового сховища. При оновленні існуючих записів система додатково контролює бізнес-обмеження, наприклад, забороняючи радикальні зміни у маршрутах, на які вже призначені майбутні виїзди.

Завершується цей цикл обробкою результату на боці фронтенду. У разі успішного проходження всіх етапів на сервері, клієнтський додаток отримує статус підтвердження та оновлений об'єкт, що тригерить ререндеринг сторінки та відображення актуальних даних користувачеві. Якщо ж на будь-якому рівні виникає помилка бізнес-правил або валідації, сервісний шар генерує виняток, який трансформується в структуровану відповідь з описом проблеми. Механізм обробки виняткових ситуацій у системі реалізований через впровадження глобального обробника на боці сервера, що дозволяє централізовано перехоплювати помилки бізнес-логіки або валідації та трансформувати їх у

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

структуровані відповіді уніфікованого формату. Це гарантує, що будь-який збій повертає клієнту чіткий опис проблеми, запобігаючи непередбачуваній поведінці програмного комплексу та забезпечуючи стабільність серверної частини.

На рівні клієнтського додатка синхронізація стану здійснюється шляхом аналізу відповідей сервера та миттєвого оновлення локальних даних без необхідності перезавантаження сторінки. У разі отримання повідомлення про помилку інтерфейс динамічно відображає відповідні сповіщення. Таке поєднання серверної обробки винятків та реактивного керування станом фронтенду дозволяє створити відмовостійке середовище, де користувач завжди отримує актуальний зворотний зв'язок щодо результатів своїх дій. Графічне представлення взаємодії компонентів під час керування контентом наведено на діаграмі послідовності (рис. 2.5).

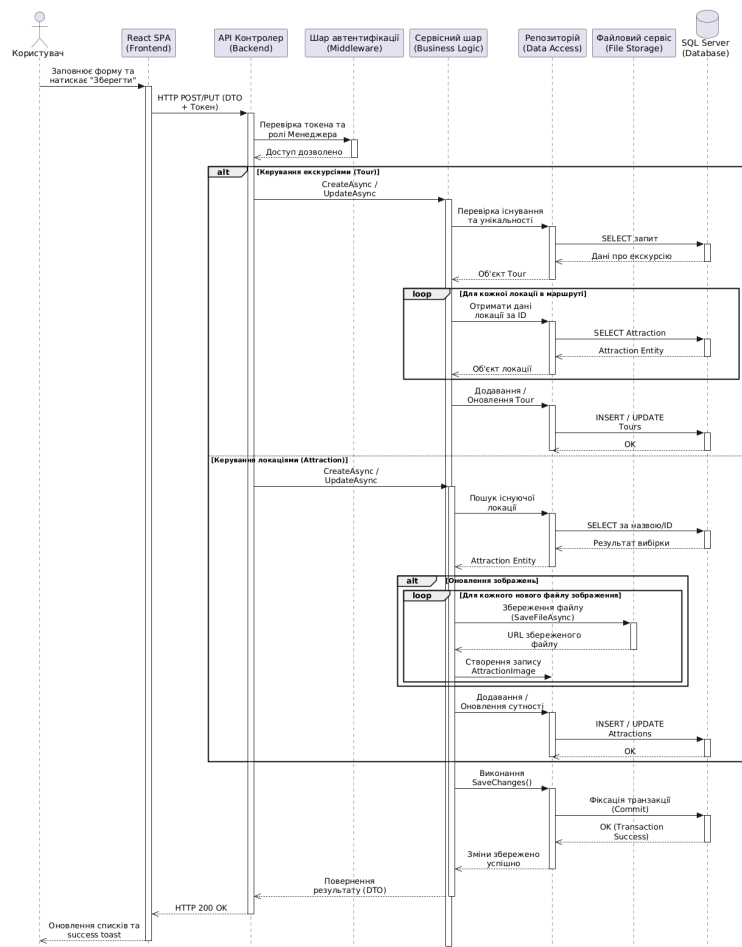


Рис. 2.5. Діаграма послідовності взаємодії компонентів під час керування контентом

Таким чином, деталізація узагальнених алгоритмів функціонування та моделювання процесів взаємодії компонентів дозволили формалізувати динамічний аспект роботи системи. Описана модель асинхронної взаємодії між клієнтською частиною та багатошаровим сервером забезпечує швидку обробку інформаційних потоків. Впровадження декларативних правил на бекенді, локальних перевірок на фронтенді та централізованого перехоплення помилок гарантує відмовостійкість додатка й захист від порушень бізнес-логіки предметної області.

2.4. Реалізація додатку

Процес розробки програмного комплексу базується на використанні стеку технологій, що забезпечує продуктивність, масштабованість та безпеку даних. Основна логіка системи зосереджена у серверній частині, яка виступає центральним вузлом обробки інформації та координації взаємодії між клієнтським інтерфейсом і базою даних.

Серверна частина програмного комплексу реалізована у вигляді ASP.NET Core Web API застосунку на платформі .NET 9. Архітектура побудована за багатошаровим принципом, що забезпечує чіткий розподіл відповідальностей. Контролери відповідають за обробку вхідних HTTP-запитів, сервіси реалізують бізнес-логіку, а репозиторії інкапсулюють доступ до бази даних через Entity Framework Core. Наведений нижче фрагмент контролера ілюструє такий розподіл – уся предметна логіка делегована сервісному шару, що робить код контролера лаконічним та легким для супроводу (повний лістинг модуля наведено у Додатку В).

```
[Route("api/[controller]")]
[ApiController]
[Authorize(Roles = "Manager")]
public class ToursController : ControllerBase
{
    private readonly ITourService _service;
    public ToursController(ITourService service) { _service =
service; }
```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        [HttpPost]
        public async Task<IActionResult> Create([FromBody]
CreateTourDto dto)
        {
            await _service.CreateAsync(dto);
            return Ok();
        }
    }

```

Структура проєкту організована за функціональним принципом – папка Services містить реалізацію правил, Repositories забезпечують ізоляцію роботи з БД, а Data/Configurations відповідають за налаштування зв'язків сутностей за допомогою Fluent API. Для забезпечення слабкого зв'язку між компонентами архітектури всі контракти винесені в директорію Interfaces. Обмін даними між клієнтом та сервером формалізовано за допомогою об'єктів передачі даних у папці DTOs, які за допомогою AutoMapper автоматично відображаються на доменні моделі. Попередня перевірка бізнес-правил DTO реалізована через FluentValidation у папці Validators, а централізований перехоплювач винятків ізолювано в Middleware.

Зв'язування створених інтерфейсів із їхніми конкретними реалізаціями, а також керування життєвим циклом об'єктів виконується у головній точці входу додатка – файлі Program.cs. Для компонентів доменного рівня визначено життєвий цикл Scoped за допомогою методу розширення AddScoped. Вибір саме цього режиму зумовлене потребою в ізоляції бізнес-операцій у межах одного запиту. Використання життєвого циклу Singleton для доменних служб є неприпустимим, оскільки контекст Entity Framework Core за своєю природою не є безпечним для багатопотокового доступу, і його спільне використання різними користувачами призвело б до взаємного блокування транзакцій та витоку конфіденційних даних. З іншого боку, режим Transient руйнував би архітектурний патерн Unit of Work, оскільки створював би окремі, не пов'язані між собою екземпляри TMSDbContext для кожної ін'єктованої залежності під час обробки однієї операції. В нашому випадку, специфікація Scoped гарантує, що новий екземпляр класу створюється індивідуально для кожного вхідного HTTP-

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

запиту, дозволяє всім сервісам у межах однієї сесії працювати з єдиним станом відстеження сутностей і утилізується після завершення запиту, запобігаючи витоку пам'яті [29].

```
// Реєстрація інфраструктури доступу до даних (Repositories)
builder.Services.AddScoped<ITourRepository, TourRepository>();
builder.Services.AddScoped<IScheduledTourRepository,
ScheduledTourRepository>();

// Реєстрація компонентів бізнес-логіки (Services)
builder.Services.AddScoped<ITourService, TourService>();
builder.Services.AddScoped<IScheduledTourService,
ScheduledTourService>();
```

Одним із найбільш технічно складних елементів системи є сервіс планування екскурсій `ScheduledTourService`. Його реалізація вимагає врахування множини обмежень – від перевірки існування об'єктів до аналізу часових накладок у розкладі гіда. Ключовим методом є `EnsureGuideAvailability`, який перевіряє доступність фахівця в обраному проміжку часу та дотримання обов'язкових технологічних перерв. Наведений у Додатку А фрагмент коду демонструє внутрішню логіку цього процесу.

Детальний технічний аналіз наведеного алгоритму дозволяє виділити три рівні контролю операційної діяльності підприємства. Перший рівень реалізується через виклик допоміжного методу `EnsureGuideWeeklyHours`, який здійснює агрегацію сумарної тривалості всіх затверджених виїздів гіда за поточний тиждень. Отримане значення сумується з тривалістю нового заходу `durationInHours` і порівнюється з установленим лімітом `MaxHoursPerWeek`, що зберігається в профілі працівника. У разі перевищення норми годин алгоритм миттєво перериває транзакцію.

Другий рівень забезпечує захист від часових накладок у межах одного робочого дня. Для цього виконується ітераційний перебір колекції `ScheduledTours` гіда, відфільтрованої за поточною датою `newTourStart.Date`. Перевірка перетину інтервалів виконується так – якщо початок існуючої екскурсії менший за час завершення нової, а початок нової менший за час

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

завершення існуючої (`otherStart < newTourEnd && newTourStart < otherEnd`), система фіксує конфлікт графіків та генерує виняток `BadRequestException`.

Третім рівнем бізнес-логіки є розрахунок обов'язкових технологічних перерв. Величину необхідного інтервалу відпочинку `requiredBreak` адаптовано до умов поїздок – якщо працівник послідовно проводить один і той самий маршрут (`scheduledTour.TourId == tourId`), час на переміщення між локаціями не потрібен, і значення встановлюється в `TimeSpan.Zero`. Якщо ж менеджер призначає гіда на інший тип туру, алгоритм автоматично вимагає наявності 30-хвилинної технічної перерви для відпочинку та зміни локації. Перевірка `newTourStart - otherEnd < requiredBreak` гарантує, що між завершенням попереднього заходу та початком наступного буде забезпечено достатній часовий проміжок, захищаючи систему від логічних помилок планування розкладу.

Для забезпечення високої стійкості до некоректних дій користувача та чистоти коду в сервісному шарі системи реалізовано перевірку вхідних даних за допомогою бібліотеки `FluentValidation`. Усі правила валідації ізольовані в окремих класах-валідаторах всередині директорії `Validators/`, що дозволяє повністю відокремити логіку перевірки від самих об'єктів передачі даних та контролерів. Валідація інтегрована в конфігурацію конвеєра `ASP.NET Core` і запускається автоматично перед тим, як запит потрапляє до відповідного методу контролера. Це гарантує, що до бізнес-сервісів надходять виключно валідні моделі, які відповідають обмеженням щодо обов'язковості полів, діапазонів чисел та довжини рядків. Приклад програмної реалізації декларативного валідатора для створення запису на екскурсію наведено нижче:

```
public class BookingDtoValidator<T> : AbstractValidator<T> where
T : BookingDtoRequestBase
{
    public BookingDtoValidator()
    {
        RuleFor(x => x.PhoneNumber)
            .NotEmpty()
            .MaximumLength(20)
            .Matches(@"^\+?[0-9\-\ ]+$")
            .WithMessage("Номер телефону може містити лише
цифри, пробіли, дефіси або необов'язковий символ + на початку.");
    }
}
```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        RuleFor(x => x.LeadContactName)
            .NotEmpty()
            .MaxLength(200);

        RuleFor(x => x.PartySize)
            .GreaterThan(0);

        RuleFor(x => x.ScheduledTourId)
            .GreaterThan(0);
    }
}

```

Важливим аспектом архітектури є ізоляція внутрішніх сутностей бази даних від зовнішнього клієнтського інтерфейсу. Для автоматизації процесу трансформації даних застосовано бібліотеку AutoMapper. Усі правила відображення об'єктів сконфігуровані у профілях мапінгу в папці MapperProfiles/. В операціях читання та повернення окремих сутностей трансформація виконується безпосередньо в сервісах через інтерфейс IMapper методом Map. Проте для оптимізації роботи з великими масивами даних та списками впроваджено метод розширення ProjectTo. Цей підхід дозволяє інтегрувати AutoMapper безпосередньо у LINQ-запити до бази даних через Entity Framework Core, завдяки чому SQL-сервер формує вибірку лише для тих стовпців, які оголошені в кінцевому DTO, значно знижуючи навантаження на мережу та оперативну пам'ять сховища.

```

public class BookingProfile : Profile
{
    public BookingProfile()
    {
        CreateMap<Booking, BookingListDto>()
            .ForMember(dest => dest.ScheduledTourDate, opt =>
opt.MapFrom(src => src.ScheduledTour.ScheduledDate))
            .ForMember(dest => dest.TourName, opt =>
opt.MapFrom(src => src.ScheduledTour.Tour.Name));

        //мапінги інших DTO
    }
}

```

Глобальна обробка помилок побудована на базі інтерфейсу IExceptionHandler, представленого в платформі .NET. Глобальний обробник

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

GlobalExceptionHandler реєструється в конвеєрі Program.cs через стандартне розширення app.UseExceptionHandler [30]. Усі винятки бізнес-логіки (наприклад, BadRequestException) або специфічні помилки FluentValidation (ValidationException), що виникають під час виконання транзакцій, автоматично перехоплюються цим шаром. Обробник логує деталі збою через сервіс ILogger і формує для клієнта уніфіковану JSON-відповідь. Такий підхід гарантує, що вебдодаток ніколи не поверне необроблену системну помилку та надає фронтенду чіткий і зрозумілий опис проблеми для виведення сповіщень користувачеві. Лістинг пов'язаного коду наведений у Додатку Б.

Для забезпечення повноцінного інформаційного супроводу об'єктів системи (логотипів підприємств, фотографій гідів та зображень туристичних локацій) виникає потреба в ефективному управлінні статичним контентом. Оскільки пряме збереження бінарних даних у таблицях реляційного сховища призводить до суттєвого розширення бази даних та деградації її продуктивності, у проекті реалізовано стратегію роздільного збереження. Вона передбачає фізичне розміщення медіафайлів на дисковому просторі сервера з фіксацією лише відносних текстових URL-шляхів у відповідних реляційних таблицях.

Приймання графічних файлів від клієнтського застосунку здійснюється асинхронно через HTTP POST/PUT запити за допомогою кодування типу multipart/form-data (FormData), що дозволяє ефективно передавати великі бінарні потоки даних неблокуючим способом. Центральним інфраструктурним компонентом, який інкапсулює всю логіку обробки, валідації та очищення медіаресурсів, є сервіс FileStorageService. Він приймає вхідні об'єкти типу IFormFile, перевіряє їх на відповідність дозволенным MIME-типам і лімітам розміру, а потім виконує асинхронний запис у вебкорінь додатка.

```
public async Task<string> SaveFileAsync(IFormFile file,
string folder, int entityId, int fileOrder, CancellationToken
cancellationToken = default)
{
    var root = _environment.WebRootPath;
    if (string.IsNullOrEmpty(root))
    {
```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        root = Path.Combine(_environment.ContentRootPath,
"wwwroot");
    }

    var folderPath = Path.Combine(root, "images", folder);
    Directory.CreateDirectory(folderPath);

    var extension = Path.GetExtension(file.FileName);
    var fileName = file.FileName;
    var fullPath = Path.Combine(folderPath, fileName);

    await using var stream = new FileStream(fullPath,
    FileMode.Create);
    await file.CopyToAsync(stream, cancellationToken);

    return $"/images/{folder}/{fileName}";
}

```

Фізична структура збереження типізована та організована всередині директорії `wwwroot/images/`, де для кожної бізнес-сутності створено окрему ізольовану підпапку за категоріями – `companies` (для фірмових логотипів), `tourguides` (для персональних фотографій гідів) та `attractions` (для демонстраційних картинок пам'яток). Щоб виключити ризик переписування файлів із однаковими назвами при одночасному завантаженні різними користувачами, сервіс реалізує стратегію динамічного формування імен за шаблоном, який поєднує категорію, унікальний ідентифікатор об'єкта та системний глобальний маркер `Guid`.

Синхронізація з реляційною базою даних відбувається на етапі успішного завершення файлової транзакції. Метод `SaveFileAsync` повертає нормалізований відносний шлях у вигляді рядка (наприклад, `/images/attractions/uid.jpg`), який передається в доменну модель через архітектурний маппінг `DTO` і фіксується у `SQL Server` у полі `photo_url` або `url` відповідної таблиці. Такий підхід дозволяє вбудованому `middleware` статичних файлів (`app.UseStaticFiles()`) повертати медіаконтент за прямим `HTTP`-запитом від `React SPA` без залучення логіки контролерів. Крім того, у сервісі реалізовано принцип каскадного очищення – при видаленні пам'ятки, профілю чи компанії, `FileStorageService` автоматично знаходить на диску пов'язані файли за збереженими `URL`-адресами та

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

безповоротно видаляє їх, запобігаючи накопиченню застарілих та невикористовуваних даних.

```
public async Task<bool> DeleteAsync(int id)
{
    var attraction = await _repository.GetByIdAsync(id);
    if (attraction is null)
    {
        return false;
    }

    if (attraction.Tours.Any(t => t.Tour.ScheduledTours.Any(st
=> st.ScheduledDate.Date > DateTime.Today)))
    {
        throw new BadRequestException("Цю пам'ятку не можна
видалити, оскільки вона входить до запланованих турів.");
    }

    foreach (var image in attraction.Images)
    {
        await _fileStorageService.DeleteFileAsync(image.Url);
    }

    _repository.Delete(attraction);
    await _repository.SaveChangesAsync();
    return true;
}
```

Безпека системи та захист даних реалізовані на основі безстанового протоколу JWT Bearer, що дозволяє відмовитися від збереження стану сесій на сервері. При успішній автентифікації генерується токен із шифруванням підпису за алгоритмом HMAC-SHA256. Маркер містить службові твердження (Claims) – ідентифікатор користувача (sub), роль (role), ID компанії для мультитенантності (companyId) та унікальний ідентифікатор сесії (jti) з терміном дії 24 години [31].

Для відкликання токенів впроваджено механізм подвійного контролю сесій. Окрім стандартної криптографічної перевірки підпису та терміну дії засобами JWT Middleware, реалізовано динамічну верифікацію статусу маркера в базі даних через подію OnTokenValidated. Під час кожного HTTP-запиту система витягує з токена унікальний jti й перевіряє його стан за таблицею AuthTokens у SQL Server. Якщо запис не знайдено або маркер має статус Revoked чи Expired, запит відхиляється з поверненням статус-коду 401 Unauthorized.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Програмна реалізація логату базується на концепції «м'якого» анулювання сесії на стороні сервера. При виході користувача сервіс AuthService знаходить відповідний запис у таблиці AuthTokens за його jti, змінює статус на Revoked та фіксує мітку часу в полі RevokedAtUtc. Одночасно на стороні фронтенду функція clearAuthSession() повністю видаляє об'єкт сесії з сховища localStorage (ключ tms.auth.session). Така схема унеможливорює повторне використання перехопленого токена злоумисниками, оскільки внутрішня перевірка надійно заблокує авторизацію на рівні бази даних.

```
public async Task LogoutCurrentTokenAsync()
{
    var token = await
_authTokenRepository.GetByJtiAsync(_currentUserService.Jti)
    ?? throw new BadRequestException("Токен не
знайдено.");

    if (token.Status == AuthTokenStatus.Active)
    {
        token.Status = AuthTokenStatus.Revoked;
        token.RevokedAtUtc = DateTime.UtcNow;
        await _authTokenRepository.SaveChangesAsync();
    }
}
```

Клієнтська частина програмного комплексу розроблена як односторінковий застосунок з використанням бібліотеки React та мови програмування TypeScript. Архітектура фронтенд-частини базується на модульному принципі Feature-based. Уся ключова функціональність розподілена за предметними областями всередині директорії src/features/ (автентифікація auth, управління турами tours, гідями guides, пам'ятками attractions та бронюваннями bookings), де кожен модуль інкапсулює власні компоненти, кастомні хуки, сервіси запитів та файли локальної валідації. Спільна інфраструктурна логіка, загальні типи та перевикористовувані компоненти інтерфейсу винесені в окремий шар src/shared/.

Побудова інтерфейсу базується на використанні перевикористовуваних компонентів та механізму React Hooks для кращого управління станом і життєвим циклом елементів. Глобальний стан авторизації реалізований через

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

AuthContext, який дозволяє поширювати дані про поточну сесію на всі рівні ієрархії компонентів. Реалізацію провайдера контексту наведено у Додатку В.

Використання хука useМемо у провайдері забезпечує стабільність об'єкта контексту і запобігає зайвим ререндерам дочірніх компонентів при зміні інших станів, тоді як useEffect синхронізує стан детального профілю користувача з актуальною сесією шляхом виклику сервера. Безстрокове збереження сесії користувача після закриття вкладки або браузера забезпечується за рахунок серіалізації об'єкта авторизації у безстрокове сховище localStorage під унікальним текстовим ключем tms.auth.session.

Для надійного розмежування прав доступу користувачів та захисту приватних екранів панелі керування від неавторизованих запитів розроблено Guard-компонент RequireAuthenticated. Цей компонент виступає захисним фільтром на рівні клієнтської маршрутизації в App.tsx. Він динамічно аналізує поточний стан сесії – якщо користувач не автентифікований, система перенаправляє його на сторінку входу, а у разі встановленого прапорця managerOnly компонент перевіряє роль користувача і блокує доступ гідів до адміністративних функцій менеджера.

```

const RequireAuthenticated:
React.FC<React.PropsWithChildren<GuardProps>> = ({
  sessionRole,
  managerOnly,
}) => {
  const { session } = useAuth();

  if (!session) {
    return <Navigate to="/auth/login" replace />;
  }

  if (managerOnly && session.role !== 'Manager') {
    return <Navigate to="/guide/dashboard" replace />;
  }

  if (sessionRole && session.role !== sessionRole) {
    return <Navigate to={session.role === 'Manager' ?
'/dashboard' : '/guide/dashboard'} replace />;
  }

  return <Outlet />;
};

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижотря О. В.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Надійність та безпека взаємодії клієнтської частини з сервером забезпечуються за рахунок повної централізації всіх HTTP-запитів. Замість використання розрізнених викликів у кожному окремому компоненті, у системі реалізовано єдиний транспортний інфраструктурний шар на основі функції `apiFetch`, розміщеної в директорії `src/shared/api/http.ts`. Ця функція розширює стандартний інтерфейс `fetch`, автоматично перехоплюючи кожен вихідний запит для динамічного додавання заголовка автентифікації `Authorization: Bearer {token}` у разі наявності активної сесії користувача.

Важливою архітектурною особливістю клієнтського HTTP-додатка є автоматичне відстеження та обробка статус-коду `401 Unauthorized`. У разі отримання такої відповіді від серверного API, `apiFetch` ініціює очищення локального сховища за допомогою функції `clearAuthSession()`, скидає стан авторизації та генерує глобальну подію для безпечного перенаправлення користувача на сторінку входу.

```
export const apiFetch = async (
  input: RequestInfo | URL,
  init?: RequestInit,
): Promise<Response> => {
  const headers = new Headers(init?.headers ?? undefined);
  const token = getAuthToken();

  if (token && !headers.has('Authorization')) {
    headers.set('Authorization', `Bearer ${token}`);
  }

  const response = await fetch(input, {
    ...init,
    headers,
  });

  if (response.status === 401) {
    clearAuthSession();

    if (typeof window !== 'undefined') {
      window.dispatchEvent(new Event(AUTH_UNAUTHORIZED_EVENT));
    }
  }

  return response;
};
```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

Обробка та типізація помилок, що повертаються сервером, реалізована за допомогою утиліти `buildApiError()`. Ця функція здійснює послідовний аналіз JSON-структури об'єкта помилки, намагаючись витягти текстовий опис проблеми з полів `detail`, `message` або `title`. У разі перехоплення комплексних помилок `FluentValidation`, утиліта автоматично розбирає масив `errors` та витягує перше доступне повідомлення, повертаючи типізований об'єкт `Error` з описом збою для кінцевого користувача.

Візуальне відображення результатів операцій та виняткових ситуацій покладено на перевикористовуваний компонент `FeedbackToast`. Залежно від переданого контексту, компонент динамічно рендерить сповіщення одного з чотирьох зафіксованих типів (`success`, `error`, `info`, `warning`), яке плавно з'являється у верхній центральній частині екрана та автоматично приховується за 4 секунди. Лістинг пов'язаного коду наведено у Додатку Г.

Складна логіка користувацького інтерфейсу, пов'язана з управлінням модальними вікнами створення та модифікації доменних об'єктів, повністю винесена в кастомні React-хуки `useTourDialogForm` та `useAttractionDialogForm`. Ці хуки контролюють внутрішній стан форм, відстежують редагування полів користувачем та управляють асинхронним підвантаженням списків, які надходять від сервера у вигляді пагінованого об'єкта `PagedResult`. Компоненти таблиць `DataTable` використовують структуру `PagedResult` для автоматичного прорахунку кількості сторінок та рендерингу рядків.

Хук `useAttractionDialogForm` безпосередньо інтегрований із картографічним компонентом `LocationMap.tsx` на базі бібліотеки `Leaflet`, що дозволяє користувачеві встановлювати точні геокоординати пам'ятки кліком по карті. У свою чергу, хук управління формами маршрутів `useTourDialogForm` реалізує механізми динамічного додавання й видалення локацій, дебаунс-пошук пам'яток, а також підтримує функціонал перетягування елементів для оперативного редагування черговості відвідування об'єктів гідом у межах екскурсії.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Для оптимізації мережевого трафіку та забезпечення миттєвого зворотного зв'язку в UI, до моменту пакування даних у FormData та відправки HTTP-запиту, на стороні фронтенду виконується валідація форм. Спеціалізовані файли перевірки (наприклад, tourValidation.ts, attractionValidation.ts) контролюють обов'язковість полів, максимальну довжину текстових рядків, числові межі та дозволені параметри завантажуваних зображень.

```
export const validateTourCollections = (
  selectedGuideCount: number,
  hasEmptyAttractionRow: boolean,
): Pick<TourFieldErrors, 'tourGuidesId' | 'attractions'> => {
  const errors: Pick<TourFieldErrors, 'tourGuidesId' |
'attractions'> = {};

  if (!selectedGuideCount) {
    errors.tourGuidesId = 'Потрібно призначити хоча б одного
гiда.';
  }

  if (hasEmptyAttractionRow) {
    errors.attractions = "Кожен рядок пам'яток повинен містити
вибрану пам'ятку.";
  }

  return errors;
};
```

У разі виявлення помилок хуки миттєво блокують виконання асинхронних сервісних промісів і заповнюють локальний масив fieldErrors для підсвічування некоректних полів безпосередньо в інтерфейсі форми. Тільки після успішного проходження клієнтських валідаторів сервіс ініціює відправку даних до API.

Висновки до розділу 2

У другому розділі здійснено проектування веб-орієнтованої системи, яке забезпечило послідовний перехід від аналізу користувацьких вимог до безпосередньої програмної реалізації комплексу. У межах роботи опрацьовано як статичні, так і динамічні аспекти платформи – визначено функціональні обмеження користувачів, розроблено схему реляційної бази даних із логічною ізоляцією даних різних компаній, а також формалізовано ключові бізнес-

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

алгоритми та сценарії клієнт-серверної взаємодії. Завершальним етапом проектування став комплексний опис архітектурної реалізації додатку, де поєднання тришарового серверного API та модульного компонентного фронтенду дозволило втілити закладені інфраструктурні рішення щодо безпеки сесій, валідації та централізованої обробки винятків.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ ПЛАНУВАННЯ МАРШРУТІВ ДЛЯ ГІДІВ

3.1. Порядок встановлення та налаштування параметрів системи

Розгортання та запуск веб-орієнтованої системи в локальному станції є стандартним етапом життєвого циклу розробки для локального функціонального тестування та демонстрації працездатності всіх модулів програмного комплексу. Для забезпечення стабільного функціонування триланкової архітектури додатку, яка одночасно поєднує роботу клієнтського інтерфейсу, серверного асинхронного API та реляційної системи керування базами даних, апаратні ресурси локального комп'ютера розробника або оператора повинні відповідати певним технічним критеріям.

До мінімальних та рекомендованих системних вимог щодо локальної робочої станції відноситься комплекс апаратних характеристик, які забезпечують життєздатність усіх рівнів платформи. Стосовно центрального процесора, для обробки паралельної компіляції та швидкого відгуку серверів необхідна наявність мінімум 4-ядерного чипа з тактовою частотою від 2.0 ГГц, хоча рекомендованим є 6-ядерний або вище процесор архітектури x64 рівня. Обсяг оперативної пам'яті визначає загальну стабільність розгорнутих служб, тому її мінімальний поріг становить 8 ГБ, оскільки одночасний запуск СКБД та середовищ виконання створює суттєве оперативне навантаження, тоді як для безперешкодного функціонування інструментів розробки, IDE та браузерних сесій настійно рекомендовано використовувати 16 ГБ. Наостанок, дискова підсистема вимагає наявності мінімум 5 ГБ вільного простору на накопичувачі, де для прискорення операцій введення-виведення під час важких транзакцій до бази даних та забезпечення гарячого релодингу інтерфейсу рекомендовано використовувати швидкісний твердотільний накопичувач з інтерфейсом NVMe або SATA.

Базовий стек системного та інфраструктурного програмного забезпечення, необхідний для розгортання платформи, включає середовище виконання .NET 9

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

SDK, платформу Node.js версії v18 або вище разом із пакетним менеджером NPM, а також СКБД Microsoft SQL Server (версії Express, Developer або LocalDB). Кінцевий доступ до графічного інтерфейсу системи здійснюється через будь-який сучасний веббраузер із підтримкою стандарту HTML5 та виконання сценаріїв JavaScript (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).

Конфігурація програмного комплексу з точки зору засобів зберігання та обробки даних у межах локального хоста описується за допомогою схеми ресурсів системи, виконаної у вигляді діаграми розгортання. Ця модель графічно відображає розподіл обчислювальних запусків на логічні рівні, які фізично функціонують у межах одного комп'ютера розробника, взаємодіючи між собою через локальні мережеві порти (рис. 3.1).

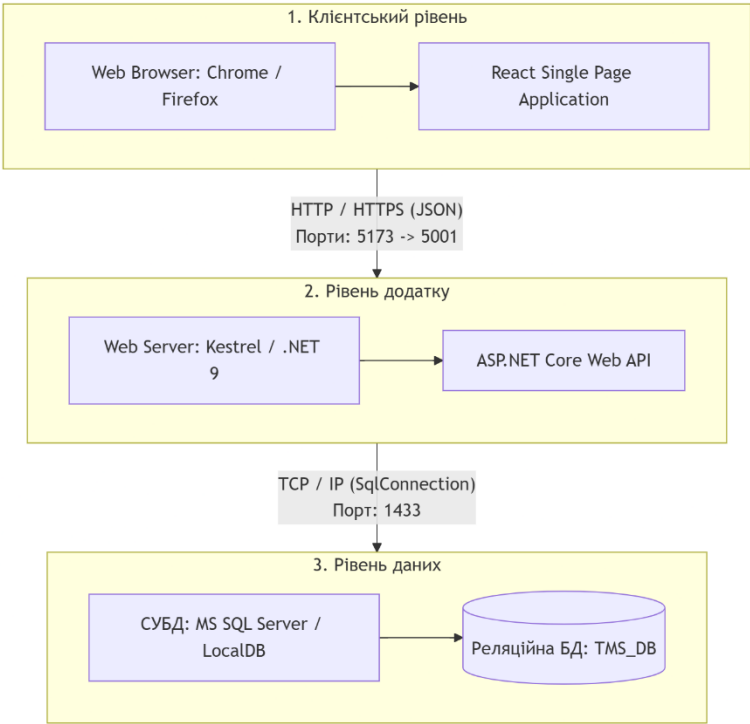


Рис. 3.1. Діаграма розгортання системи

Архітектура системи базується на трирівневій моделі, що розподіляє обчислювальні ресурси між клієнтським рівнем, рівнем додатка та рівнем даних. Клієнтський рівень функціонує у середовищі веббраузера у вигляді односторінкового застосунку React SPA на базі TypeScript (локальний порт 5173), виконуючи рендеринг графічних елементів та реактивне управління

станом інтерфейсу. Рівень додатка функціонує під керуванням вебсервера Kestrel у платформі .NET 9 (порти 5000/5001) і виступає центральним ядром системи, забезпечуючи маршрутизацію API, аутентифікацію за допомогою JWT-токенів, валідацію моделей та координацію транзакцій. Рівень даних реалізовано на базі СКБД MS SQL Server (порт 1433), яка забезпечує фізичне збереження інформації, підтримує реляційні обмеження цілісності та оптимізує виконання пошукових запитів.

Міжрівнева комунікація організована за допомогою стандартизованих протоколів. Взаємодія між клієнтом та додатком побудована на базі архітектурного стилю REST API за допомогою асинхронного обміну JSON-повідомленнями через протоколи HTTP/HTTPS. Зв'язок між рівнем додатка та базою даних реалізовано по протоколу TCP/IP через підключення типу SqlConnection, де об'єктно-реляційний мапер Entity Framework Core трансліює логічні запити в оптимізовані SQL-команди. Така конфігурація локального середовища структурно відтворює топологію майбутнього хмарного розгортання системи.

Розгортання серверної частини системи розпочинається з процедури клонування або імпорту вихідного коду проєкту до локального середовища розробки. Наступним обов'язковим етапом є конфігурація параметрів файла appsettings.json, у якому специфікується локальний рядок підключення до реляційної бази даних SQL Server. У цьому ж документі налаштовуються ключові компоненти безпеки, що включають встановлення секретного ключа для шифрування JWT-токенів, а також чітке визначення видавця та безпосереднього споживача цих токенів.

Формування та актуалізація фізичної структури бази даних здійснюється через інтерфейс командного рядка за допомогою CLI-команд міграцій Entity Framework Core, зокрема шляхом виконання директиви dotnet ef database update. З метою забезпечення працездатності платформи під час її першого старту інтегровано алгоритм первинного автоматичного наповнення сховища тестовими матеріалами. Після завершення всіх підготовчих етапів ініціалізація

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

локального сервера додатків виконується за допомогою стандартної команди `dotnet run` або функціональних засобів сучасних середовищ розробки, серед яких Visual Studio та Rider.

Ініціалізація середовища клієнтського інтерфейсу розпочинається зі встановлення всіх необхідних проєктних залежностей, зафіксованих у `package.json`, за допомогою пакетного менеджера шляхом виконання команди `npm install`. Наступним кроком є створення та налаштування конфігураційного файлу локальних змінних оточення `.env.local` або відповідного інфраструктурного документа. У цьому файлі визначається базова адреса локального серверного API за допомогою змінної `VITE_API_BASE_URL`, що дозволяє транспортному клієнту коректно ідентифікувати напрямок для відправлення асинхронних запитів.

Процес компіляції та запуску локального сервера розробки реалізується через введення команди `npm run dev`, яка запускає середовище збирання та забезпечує автоматичне оновлення компонентів додатка. Після успішної ініціалізації інструментів розробника здійснюється первинне підключення користувача до системи через веббраузер за локальною адресою хоста на порті 5173. Цей етап завершує розгортання фронтенд-частини та надає повноцінний доступ до користувацького інтерфейсу для подальшої взаємодії з платформою.

3.2. Структура інтерфейсу веб-орієнтованої системи

Графічний інтерфейс користувача розпочинається з форми автентифікації. Візуальний інтерфейс форми автентифікації розроблено у мінімалістичному стилі з використанням світлої палітри кольорів та чітким акцентом на центральних елементах керування. У верхній частині блоку розташовано головний заголовок форми та підзаголовок. Безпосередньо під ними розміщено навігаційну панель горизонтальних посилань для перемикання між режимами входу, реєстрації нової компанії, приєднання до існуючої структури за унікальним кодом та переходу до процедури первинного налаштування пароля облікового запису гіда.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Центральну область форми займають текстові поля введення із маркуванням обов'язковості заповнення. Перше інтерактивне поле призначене для вказання ідентифікатора користувача, яким може виступати адреса електронної пошти або номер мобільного телефону, а друге поле слугує для введення пароля із реалізацією функції автоматичного маскування символів крапками. Завершальним елементом інтерфейсу виступає прямокутна кнопка, яка забезпечує ініціалізацію процесу входу та відправлення внесених даних на серверний рівень додатка для подальшої перевірки (рис. 3.2).

Рис. 3.2. Форма автентифікації користувача

Після успішного проходження автентифікації менеджер переходить до головного вікна програми, структура якого розподілена на ліве бічне меню та основну робочу область із верхньою інформаційною панеллю. Бічна навігаційна панель містить назву компанії з можливістю згортання елемента, а також послідовний перелік розділів для переходу до керування розкладом, пам'ятками, гідами, екскурсіями, бронюваннями та статистикою. У нижній частині цього блока закріплені кнопки доступу до персонального профілю та виходу з облікового запису (рис. 3.3).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

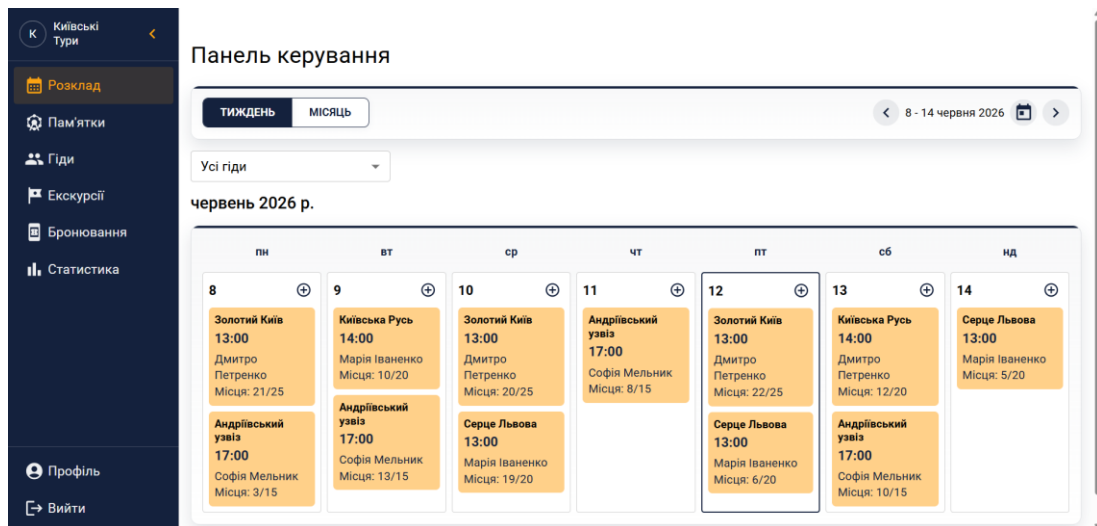


Рис. 3.3. Головна сторінка додатку

За замовчуванням на ПК меню завжди відкрите, проте за бажанням користувач може мінімізувати його, щоб мати більше робочого місця (рис. 3.4). Поточний вибір записується у локальне сховище, щоб при перезавантаженні сторінки стан меню запам'ятовувався.

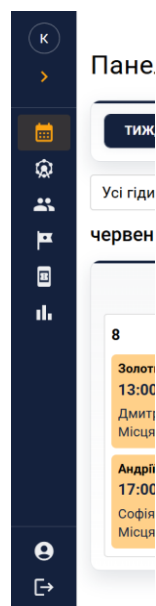


Рис. 3.4. Згорнуте меню на ПК

Візуальна адаптація інтерфейсу користувача під різні типи мобільних пристроїв та планшетів забезпечує збереження повноцінного функціоналу системи при обмеженій площі екрана. У мобільних конфігураціях та на планшетах постійне відображення бічного меню вимикається для максимального розширення площі робочого простору (рис. 3.5).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

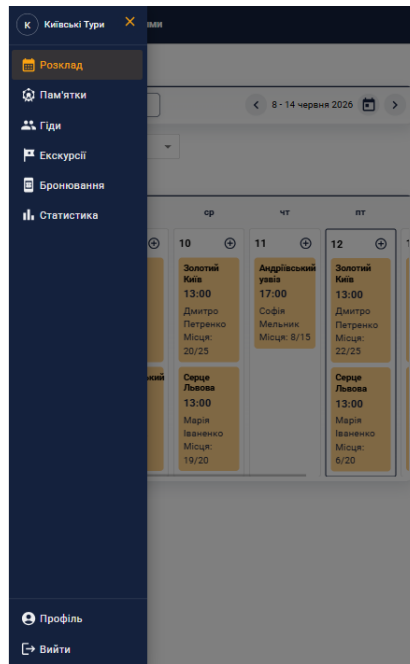


Рис. 3.5. Адаптація робочого простору на планшеті

Перехід між розділами платформи у такому разі здійснюється за допомогою висувного компонента типу drawer, який накладається поверх затіненої робочої області при його виклику. Мобільна версія навігації повністю відтворює склад головного меню та додатково обладнана графічною кнопкою закриття у верхній зоні для оперативного повернення до перегляду інформаційних віджетів.

На мобільних пристроях, на відміну від планшетів, меню розгортається на весь екран, щоб забезпечити доступність усіх опцій і максимальну комфортність роботи з системою (рис. 3.6).

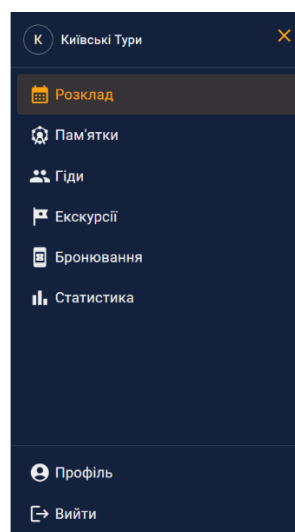


Рис. 3.6. Меню на мобільному пристрої

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Головна панель керування менеджера призначена для моніторингу, планування та координації екскурсійної діяльності. Основну частину робочого простору цього додатка займає інтерактивний календар, який у тижневому режимі відображення детально структурує поточні заходи за відповідними стовпцями від понеділка до неділі. Верхня зона інтерфейсу містить інформацію про поточний часовий проміжок, навігаційні кнопки для перемикання між періодами та графічні елементи швидкого додавання нових подій у сітку розкладу за допомогою інтегрованої кнопки зі знаком плюс (рис. 3.7).

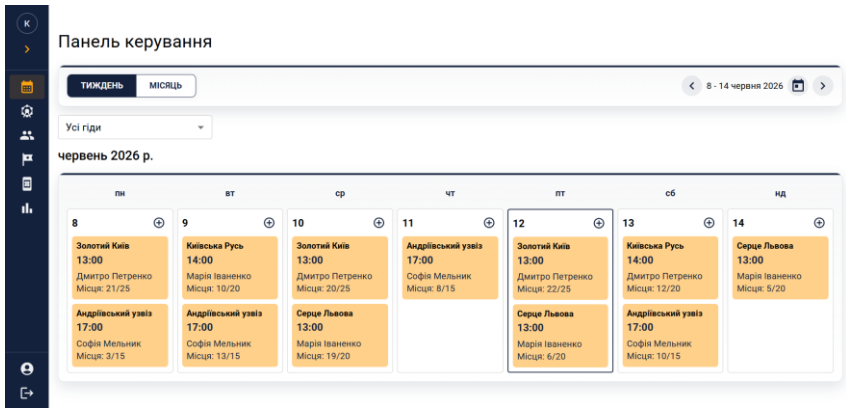


Рис. 3.7. Інтерфейс календаря у тижневому форматі відображення

Для довгострокового планування та аналізу загального навантаження компанії передбачено можливість перемикання інтерфейсу у місячний формат відображення за допомогою відповідної кнопки у верхньому блоці. У цьому режимі календар трансформується у масштабну сітку днів поточного місяця, де заплановані тури подаються у вигляді компактних горизонтальних інформаційних ліній із зазначенням часу старту та скороченої назви маршруту (рис. 3.8).

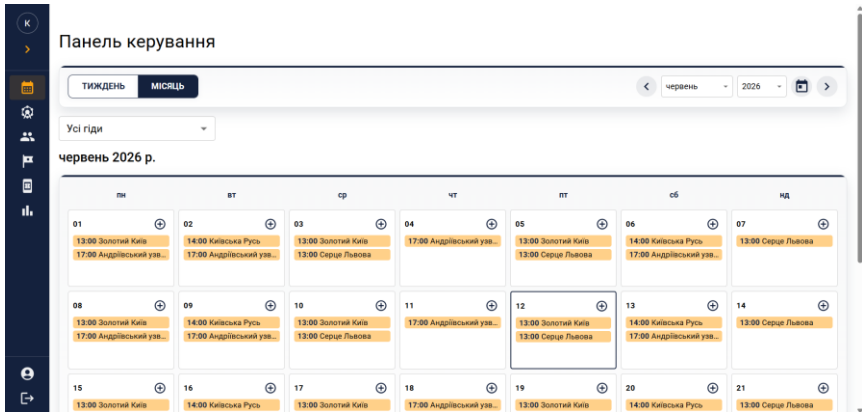


Рис. 3.8. Календар менеджера у місячному режимі відображення розкладу

Оптимізація відображення даних забезпечується за допомогою інструментів швидкої фільтрації. Спеціальний випадаючий список дозволяє менеджеру активувати інтерактивне меню із переліком усіх зареєстрованих у системі гідів компанії для аналізу графіка конкретного працівника (рис. 3.9).

Панель керування

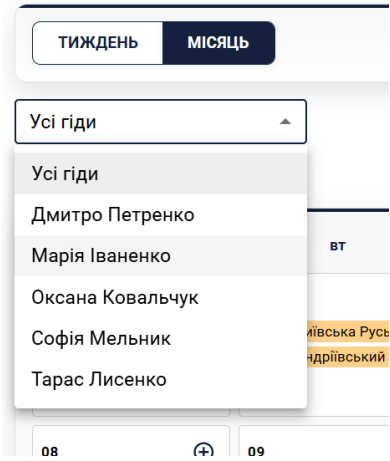


Рис. 3.9. Випадаючий список для фільтрації розкладу за гідом

Після вибору конкретного гідів інтерфейс панелі керування динамічно оновлює стан календарної сітки, залишаючи виключно ті заходи, які закріплені за обраною особою. Для зручності взаємодії поруч із фільтром з'являється додаткова текстова кнопка очищення параметрів (рис. 3.10).

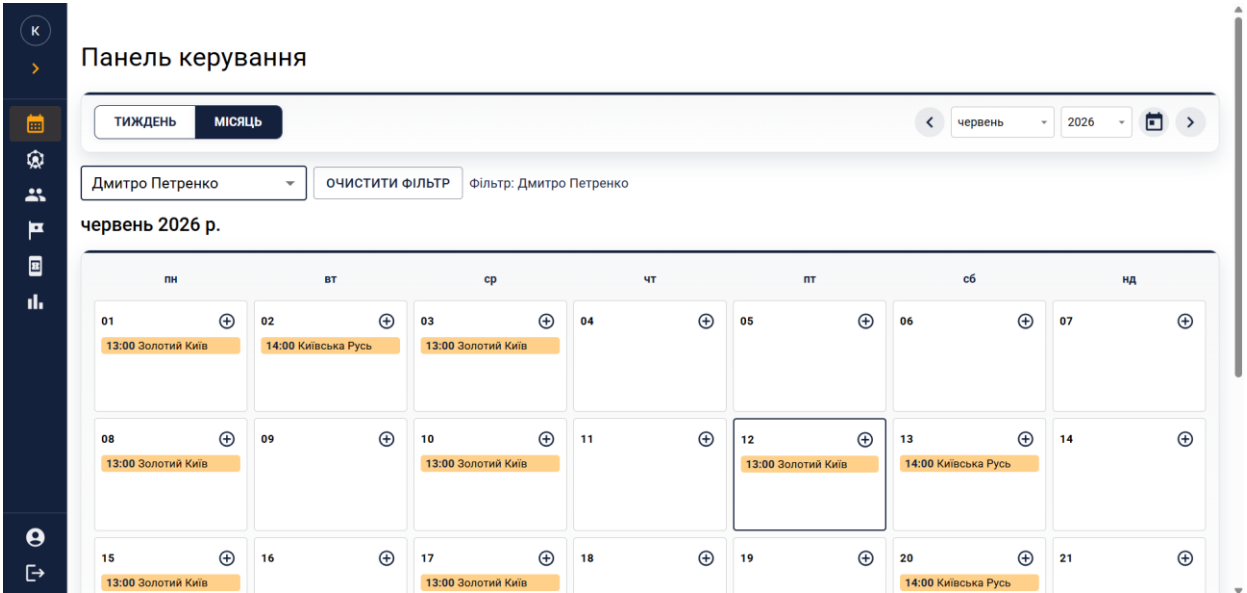


Рис. 3.10. Відображення відфільтрованого розкладу із кнопкою скидання параметрів

При виборі конкретного заходу в календарі над затіненою робочою областю розгортається модальне вікно деталей запланованої екскурсії. У верхній частині цього блоку міститься назва маршруту разом із графічними іконками для редагування або видалення запису, а також кнопка закриття діалогового вікна. Текстова частина інтерфейсу відображає ключові параметри події, серед яких дата й час проведення, ім'я закріпленого гіда, поточний рівень заповнення доступних місць, вартість участі, тривалість у годинах та опис екскурсійної програми. Через наявність активних клієнтських заявок система автоматично виводить попередження помаранчевого кольору про обмеження функцій редагування та повне блокування можливості видалення цієї події. Знизу вікна виступають дві кнопки, призначені для додавання нового або перегляду вже наявних бронювань для цього туру (рис. 3.11).

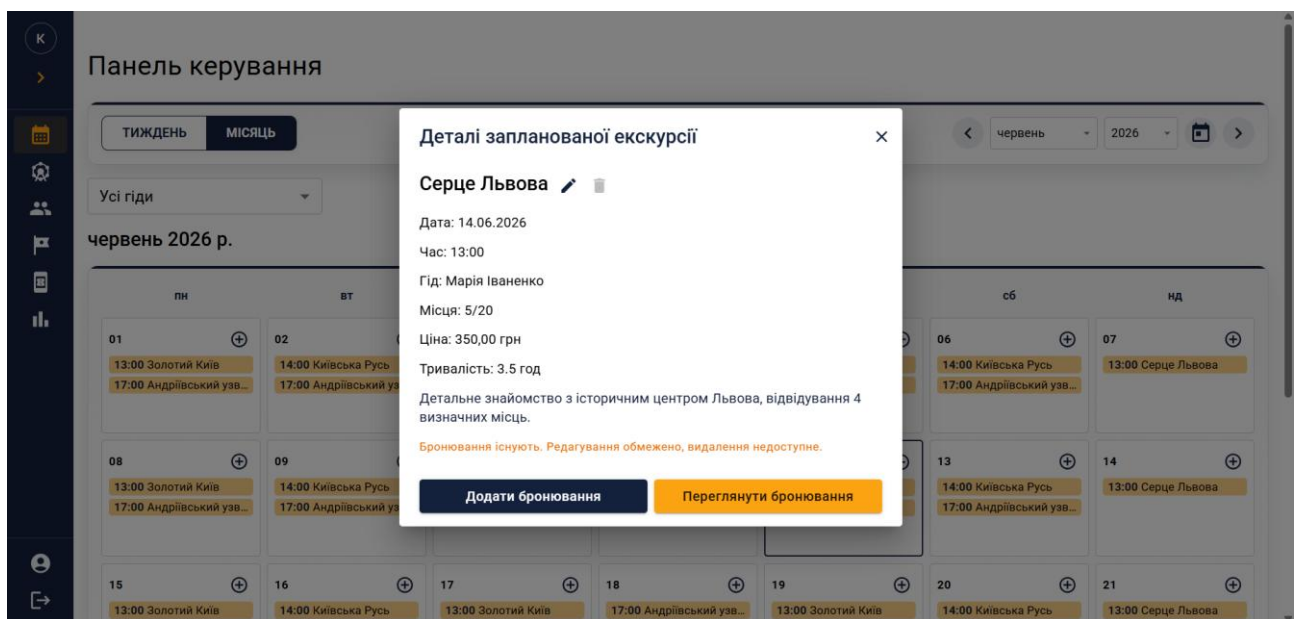


Рис. 3.11. Картка запланованої екскурсії

Графічний інтерфейс підсистем для управління екскурсійною діяльністю та об'єктами інтересу має уніфіковану структуру. Основні екранні форми цих модулів реалізовані у вигляді інтерактивних журналів, де дані структуровано розподілені за відповідними колонками, такими як фотокартка об'єкта, його офіційна назва, графік роботи та точне місцезнаходження. У верхній частині робочої області логічно розміщено рядок глобального текстового пошуку, інструменти швидкої фільтрації та акцентну функціональну кнопку для

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

ініціалізації створення нового запису. Правий стовпець таблиці відведено під блок дій, який містить графічні елементи для оперативного переходу до редагування або безпосереднього видалення вибраного інформаційного рядка (рис. 3.12).

Фото	Назва	Години роботи	Місцезнаходження	Дії
	Андріївська церква	10:00-18:00	Андріївський узвіз, 23, Київ	
	Батьківщина-Мати	10:00-18:00	вулиця Лаврська, 27, Київ	
	Високий Замок	24/7	вулиця Замкова, Львів	
	Золоті Ворота	10:00-18:00	Володимирська вулиця, 40А, Київ	
	Киево-Печерська лавра	09:00-18:00	вулиця Лаврська, 15, Київ	
	Латинська катедра	09:00-18:00	площа Катедральна, 1, Львів	

Рис. 3.12.Список пам'яток у вигляді таблиці

Для забезпечення високої швидкості завантаження та зручності у табличний компонент додано механізми пагінації та динамічного сортування рядових записів. Блок пагінації розташовано по центру в нижній частині таблиці, що дозволяє менеджеру здійснювати послідовний або прямий перехід між ними (рис. 3.13).

	Високий Замок	24/7	вулиця Замкова, Львів	
	Золоті Ворота	10:00-18:00	Володимирська вулиця, 40А, Київ	
	Киево-Печерська лавра	09:00-18:00	вулиця Лаврська, 15, Київ	
	Латинська катедра	09:00-18:00	площа Катедральна, 1, Львів	
	Майдан Незалежності	24/7	Майдан Незалежності, Київ	
	Михайлівський Золотоверхий монастир	08:00-19:00	Трьохсвятительська вулиця, 8, Київ	
	Оперний театр Львова	10:00-19:00	проспект Свободи, 28, Львів	
	Площа Ринок	24/7	площа Ринок, 1, Львів	

< 1 2 >

Рис. 3.13.Пагінація

Сортування активується кліком миші безпосередньо на заголовок відповідного стовпця таблиці. При взаємодії з назвою колонки система автоматично перебудовує список об'єктів за зростанням або спаданням, відображаючи поточний напрямок сортування за допомогою графічного індикатора у вигляді стрілки поруч із текстом (рис. 3.14).

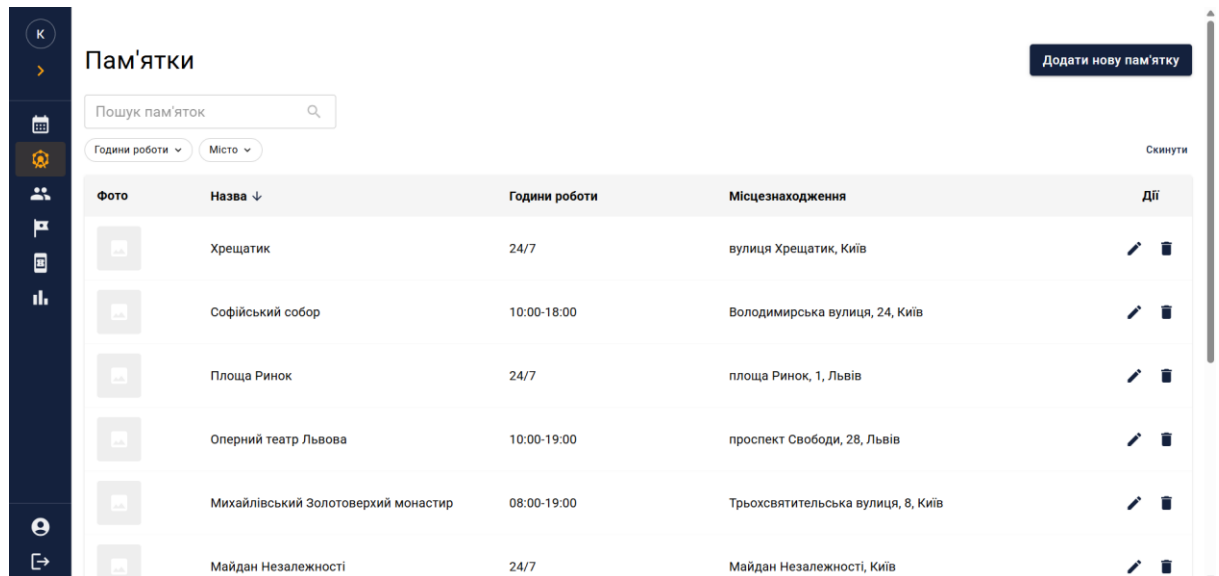


Рис. 3.14. Сортування

Інструмент фільтрації підтримує можливість одночасного накладання кількох незалежних умов. Усі активовані критерії пошуку динамічно трансформуються у візуальні графічні теги з індивідуальними кнопками скасування, які розміщуються безпосередньо над табличною частиною (рис. 3.15).

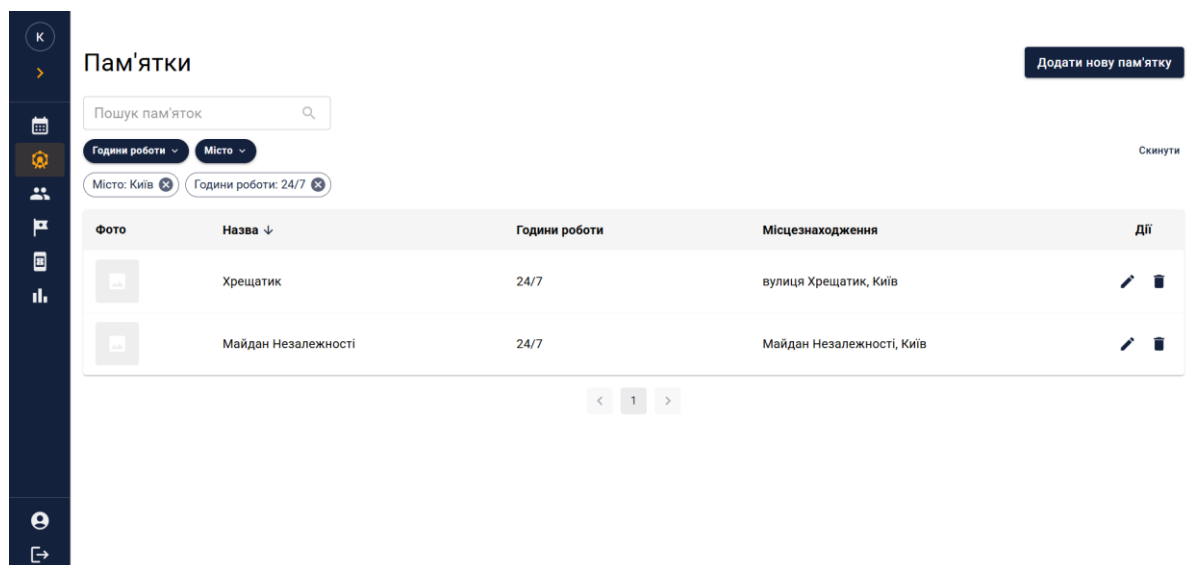


Рис. 3.15. Фільтрація

Процедура внесення нових об'єктів або редагування наявних маршрутів реалізована через розгортання модальних діалогових вікон, які накладаються поверх основного інтерфейсу. Верхня частина такої форми містить послідовні текстові поля для введення назви, детального опису та очікуваного формату робочих годин. Нижче інтегровано інтерактивну карту на базі картографічного рушія Leaflet, яка слугує основним інструментом для швидкого визначення просторових географічних координат об'єкта (рис. 3.16).

Рис. 3.16. Форма додавання/редагування пам'яток

Взаємодія з картою побудована за принципом подвійного зв'язку, де клік миші у будь-якій точці карти автоматично встановлює синій маркер ініціалізації та самостійно заповнює текстове поле адреси відповідними даними з бази геокодингу. Це дозволяє уникнути механічних помилок при ручному введенні довготи чи широти локації оператором (рис. 3.17).

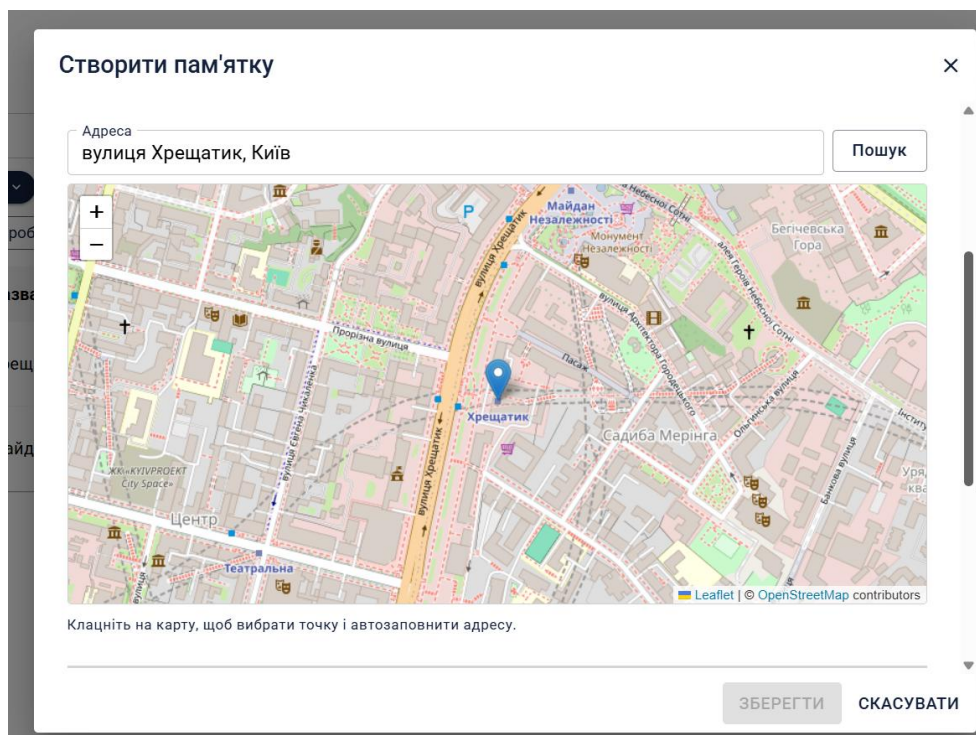


Рис. 3.17. Карта для додавання адреси

Останній блок діалогового вікна відведений під мультимедійне наповнення, що дозволяє прикріпити до чотирьох графічних файлів зображень обмеженого розміру. Цей компонент містить виділені слоти для попереднього перегляду завантажених фотокарток із можливістю їх індивідуального видалення через іконку (рис. 3.18).

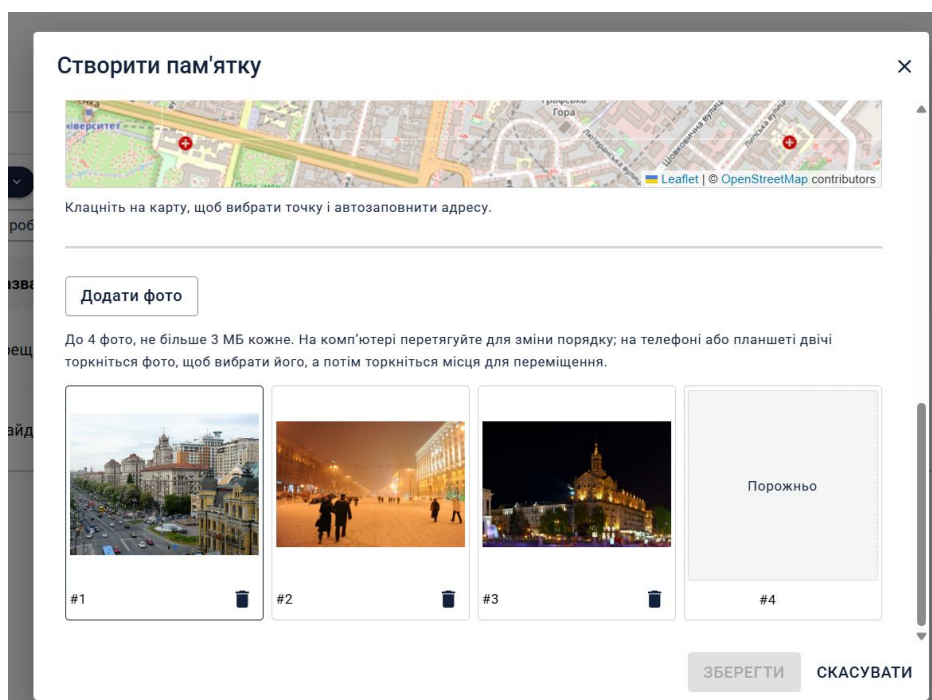


Рис. 3.18. Блок додавання фото

		Чорноус В. І.		
		Чижмотря О. В.		
Змн.	Арк.	№ докум.	Підпис	Дата

Інтерфейс модулю статистики надає менеджеру інструменти для аналізу діяльності компанії за довільно обраний проміжок часу за допомогою календарних віджетів встановлення меж періоду. Головне вікно цього модуля розділене на три тематичні вкладки, перша з яких відображає узагальнений огляд ключових показників ефективності. На цій сторінці сформовано шість інформаційних карток зі зведеними числовими даними щодо загальної кількості створених бронювань, проведених екскурсій, середнього розміру екскурсійної групи, відсотка заповнюваності місць, сумарної чисельності гостей та обсягу доступних вільних слотів. У нижній частині вікна огляду розміщено аналітичні плашки лідерів, які демонструють найбільш затребуваний екскурсійний маршрут, найактивнішого співробітника та найпопулярніше місто серед клієнтів (рис. 3.19).

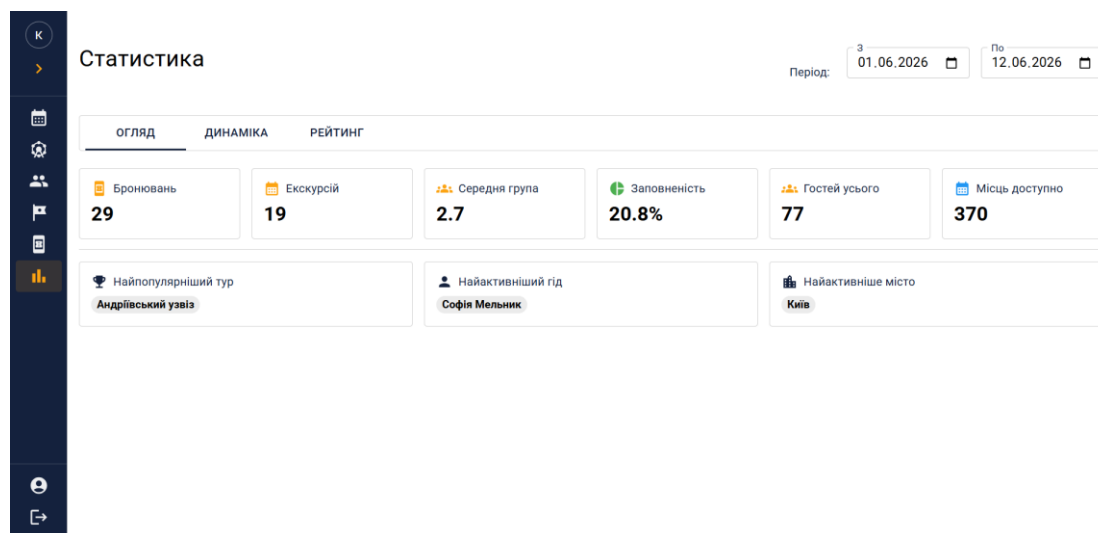


Рис. 3.19. Головна сторінка модулю статистики

Друга вкладка присвячена візуалізації операційних процесів у часовому вимірі за допомогою інтерактивного лінійного графіка динаміки замовлень. Графік підтримує функцію групування даних за днями, тижнями або місяцями через відповідний випадаючий список у верхній зоні робочої області. Візуальне відображення реалізовано за допомогою кольорових ліній, де суцільна помаранчева лінія відображає кількість бронювань, синя показує проведені екскурсії, а пунктирна зелена лінія демонструє зміну рівня заповнюваності місць у відсотках відносно правої вертикальної шкали. При наведенні курсора миші на

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

будь-яку точку часової осі система розгортає інформаційне вікно підказки з точними кількісними показниками для обраної дати(рис. 3.20).



Рис. 3.20. Сторінка динаміки бронювань

Останній блок представлений вкладкою рейтингів, яка дозволяє побачити успішність окремих складових бізнесу за допомогою вкладених розділів для турів, гідів та міст компанії. Інформація у цьому вікні структурується у вигляді таблиці, де кожен рядок відповідає певному об'єкту та містить статистичні метрики щодо кількості замовлень, фактично виконаних виїздів, загальної чисельності залучених туристів та середнього відсотка заповнюваності груп. Нижній сегмент табличної частини обладнаний стандартними інструментами пагінації для зручного перемикання між сторінками сформованого списку лідерів (рис. 3.21).

К

>

Рис. 3.21.Сторінка рейтингів

Персональний кабінет гіда відкривається на головній сторінці, яка сфокусована на відображенні інформації про поточну зайнятість та індивідуальні показники ефективності. У верхній частині інтерфейсу розташовано інформаційне сповіщення про найближчий запланований захід із зазначенням назви туру та точного часу його початку. Середню зону екрана займають інформаційні картки зведеної статистики за поточний місяць, які демонструють кількість завершених та майбутніх екскурсій, а також сумарну кількість обслугованих клієнтів. Нижня частина під блок відпрацьованого часу протягом поточного тижня. Цей елемент реалізовано у вигляді лінійного індикатора прогресу, який демонструє співвідношення фактично набраних годин до максимально дозволеного ліміту тижневого навантаження. Додатково тут виводиться точна кількість запланованих виїздів та розрахований залишок доступного робочого часу (рис. 3.22).

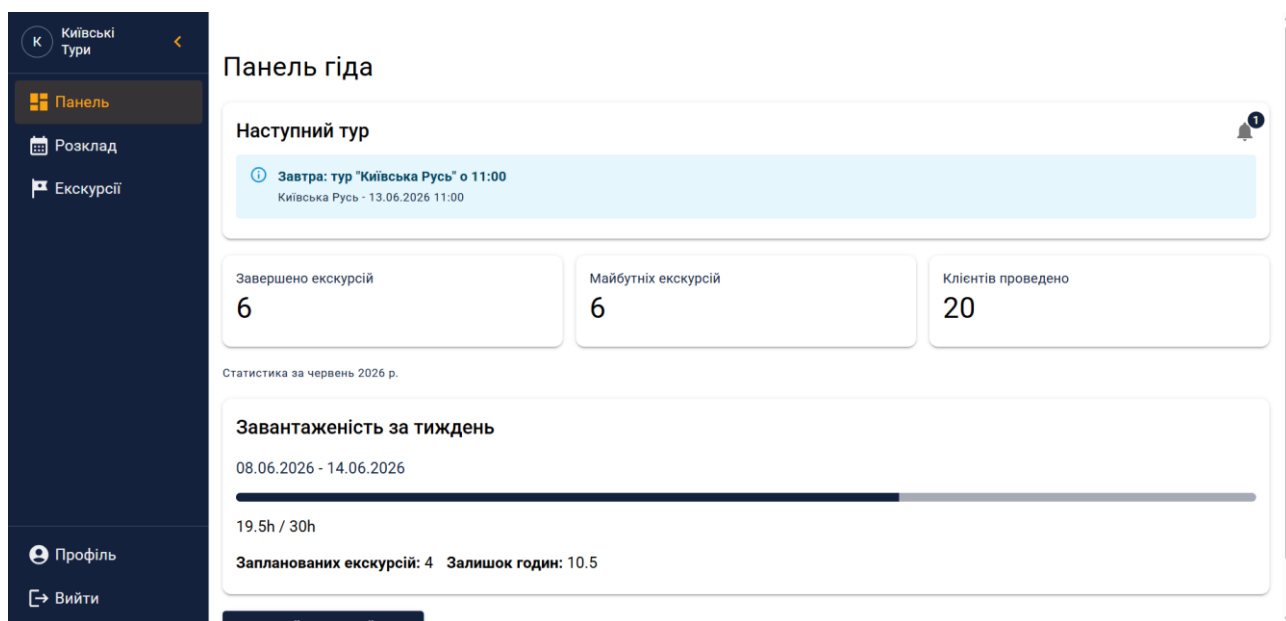


Рис. 3.22. Головна сторінка кабінету гіда

Для детального огляду графіка екскурсій у кабінет додано календар зайнятості гіда у тижневому та місячному форматах. У тижневій сітці кожна особиста екскурсія виділяється окремою інформаційною плашкою жовто-оранжевого кольору, де зазначається назва маршруту, точний час початку та поточна заповненість групи, тоді як дні без активних завдань маркуються повідомленням про відсутність запланованих заходів (рис. 3.23).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

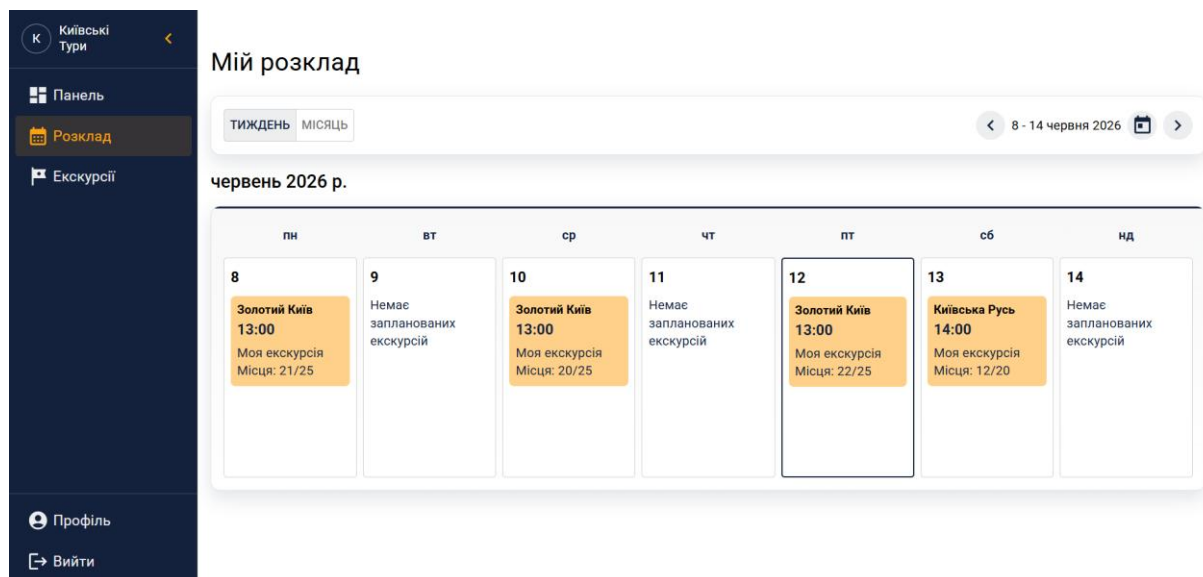


Рис. 3.23. Розклад гідів

При переході до деталей конкретного заходу гід отримує доступ до повної інформаційної картки екскурсії. Верхня частина вікна відображає вартість, тривалість, місто проведення, точний час початку та поточний стан завантаженості екскурсійної групи (рис. 3.24).

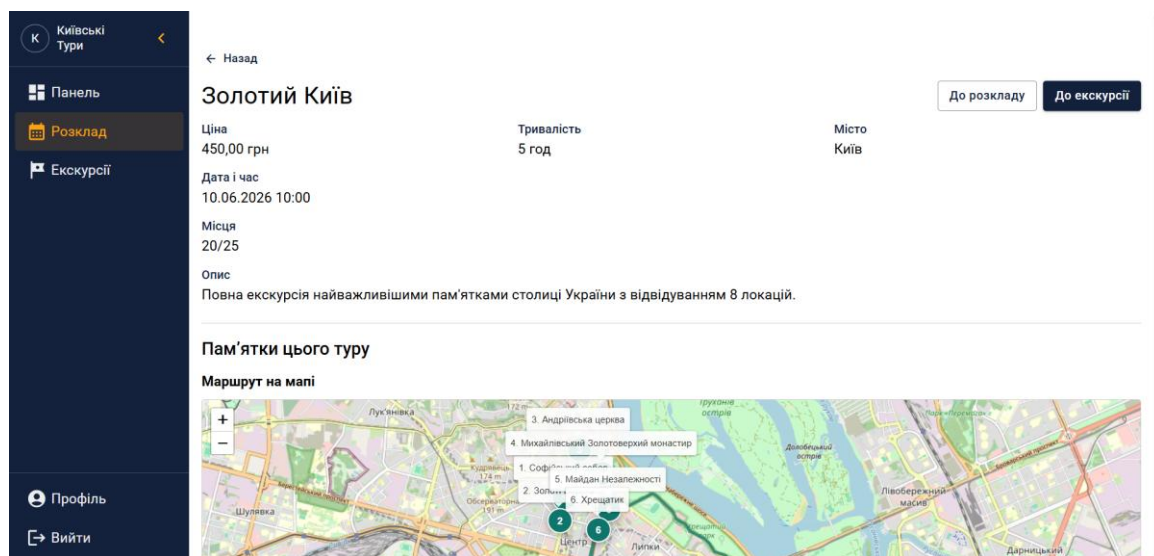


Рис. 3.24. Огляд детальної інформації про екскурсію

Важливим елементом інтерфейсу виступає інтерактивний блок маршруту на мапі, де за допомогою пронумерованих зелених маркерів та сполучних ліній візуалізується точна географічна послідовність переміщення між об'єктами. Під картою розташовано текстову специфікацію маршруту, де у вигляді таблиці наведено порядок відвідування визначних місць, їхні назви, короткі описи та адреси (рис. 3.25).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

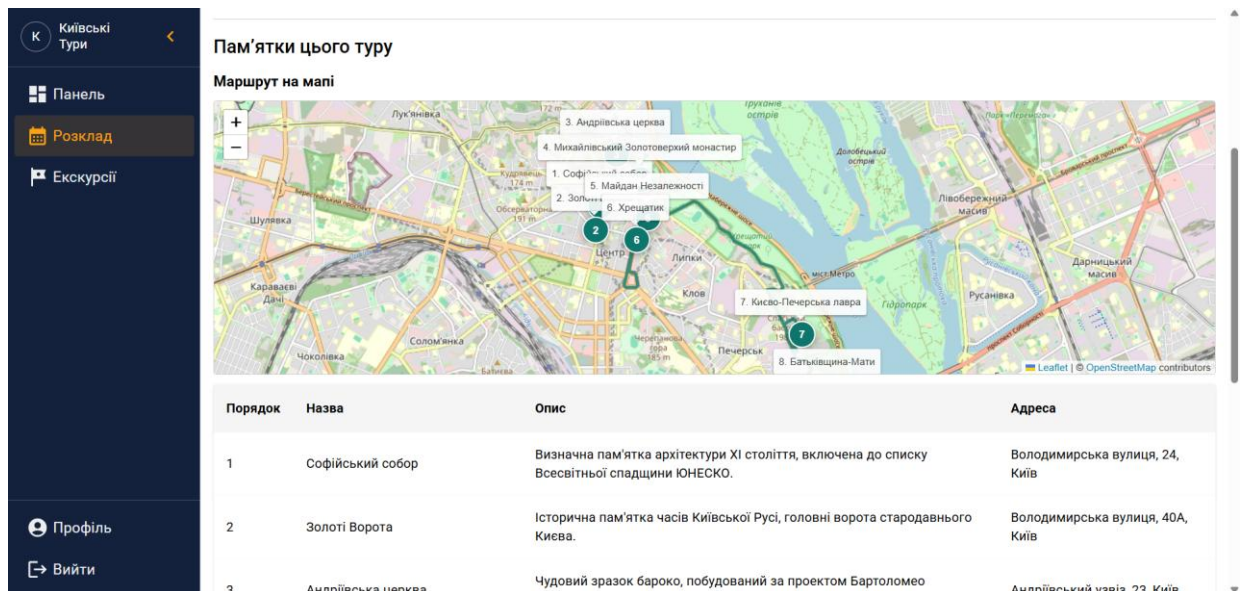


Рис. 3.25. Інформація про маршрут екскурсії

Нижній сегмент цієї сторінки відведено під журнал клієнтських заявок, який відображає список підтверджених бронювань на цю дату із зазначенням імен контактних осіб, номерів телефонів, кількості людей у групі та фінансового статусу операції (рис. 3.26).

2	Золоті Ворота	Історична пам'ятка часів Київської Русі, головні ворота стародавнього Києва.	Володимирська вулиця, 40А, Київ
3	Андріївська церква	Чудовий зразок бароко, побудований за проектом Бартоломео Растреллі.	Андріївський узвіз, 23, Київ
4	Михайлівський Золотоверхий монастир	Діючий чоловічий монастир з блакитними фасадами та золотими банями.	Трьохсвятительська вулиця, 8, Київ
5	Майдан Незалежності	Центральна площа Києва, символ незалежності України.	Майдан Незалежності, Київ
6	Хрещатик	Головна вулиця Києва, центральний бульвар столиці.	вулиця Хрещатик, Київ
7	Києво-Печерська лавра	Один із найдавніших монастирів Київської Русі, внесений до списку ЮНЕСКО.	вулиця Лаврська, 15, Київ
8	Батьківщина-Мати	Монументальна скульптура на правому березі Дніпра, символ перемоги.	вулиця Лаврська, 27, Київ

Бронювання для цього запланованого туру						
Контакт	Телефон	Email	Кількість осіб	Статус	Сума	Дата бронювання
Віктор Гнатюк	+380502323028	-	5	Confirmed	2 250,00 грн	07.06.2026 10:00

Рис. 3.26. Інформацію про бронювання міць екскурсії

Завершальним компонентом кабінету є розділ особистих екскурсій, призначений для ознайомлення з повним переліком закріплених за цим гідом турів. Інтерфейс виконано у вигляді компактного реєстру з функцією швидкого текстового пошуку за назвою або містом, де у табличному вигляді відображаються базові параметри кожного доступного маршруту (рис. 3.27).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

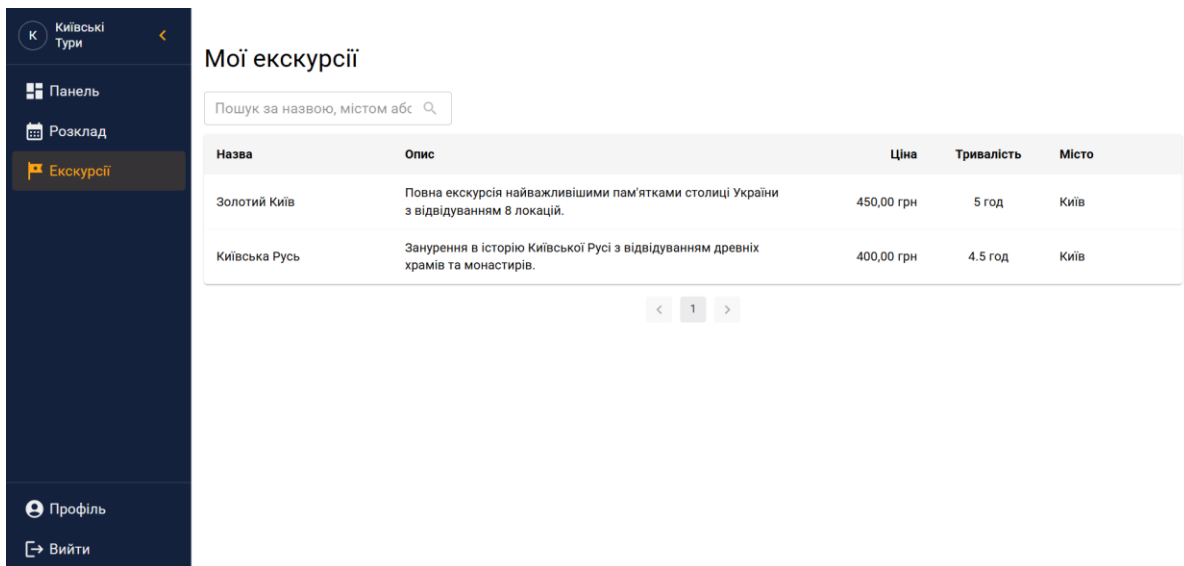


Рис. 3.27. Список екскурсій гіда

Для забезпечення зворотного зв'язку з користувачем у платформі реалізовано механізм асинхронних спливаючих повідомлень, які з'являються у верхній центральній частині екрана поверх основного вмісту сторінки. Ці сповіщення призначені для інформування про результати виконання поточних операцій, включаючи успішне збереження чи оновлення даних, системні попередження або критичні збої при взаємодії з бекендом. У разі безпомилкового завершення транзакції візуальний елемент відображається у вигляді компактної горизонтальної плашки зеленого кольору (рис. 3.28).

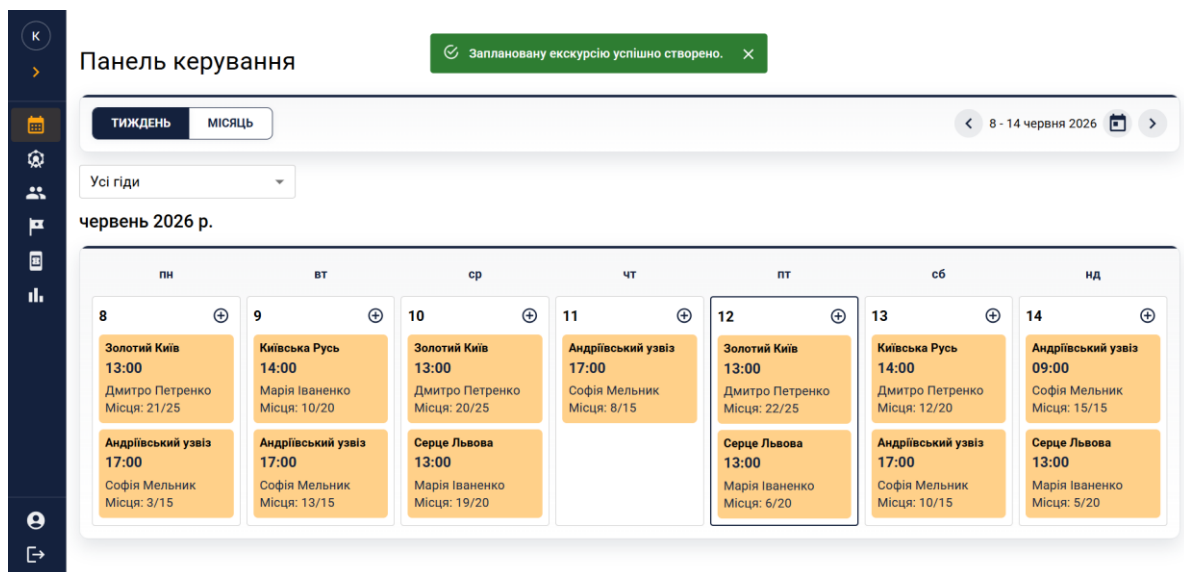


Рис. 3.28. Сповіщення про успішність виконання операції

Захист від випадкового або помилкового видалення записів забезпечується через модальне вікно попереднього підтвердження. При ініціалізації процедури

		Чорноус В. І.			ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

видалення певного ресурсу інтерфейс програми автоматично затінює та блокує робочого простору та повністю фокусує увагу користувача на діалоговому блоці (рис. 3.29).

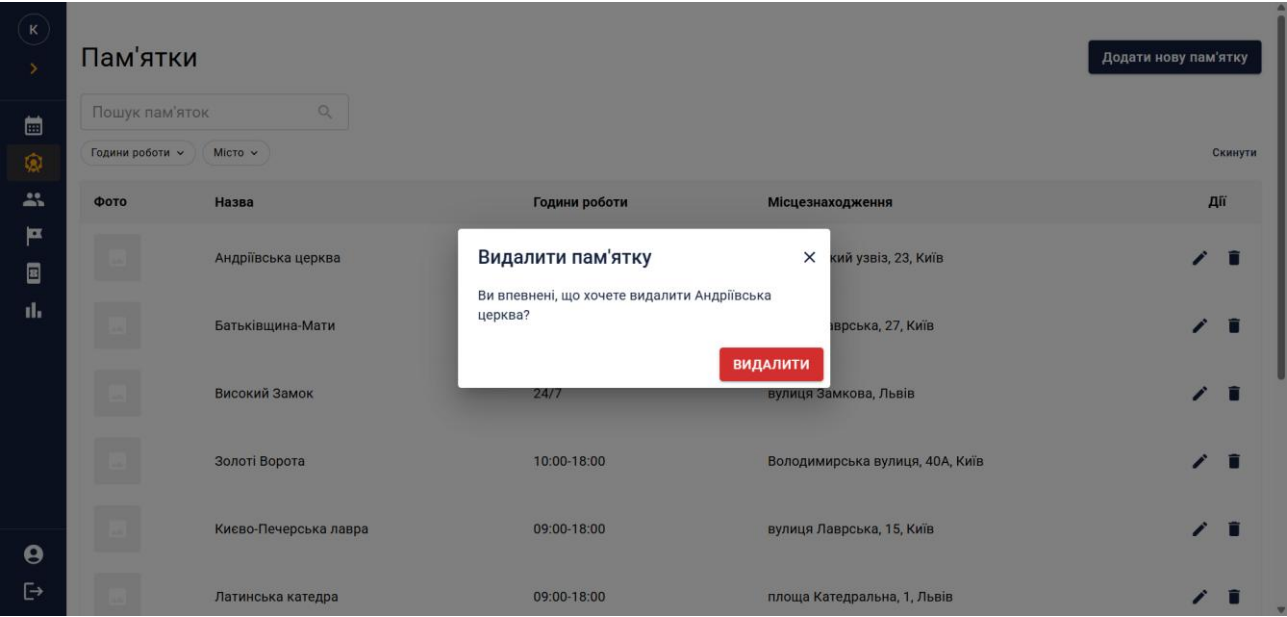


Рис. 3.29. Підтвердження видалення

Отже, розроблений графічний інтерфейс забезпечує зручний, цілісний та адаптивний робочий простір для всіх категорій користувачів системи. Завдяки впровадженню уніфікованих таблиць, інтерактивного календаря розкладу та картографічного модуля стає можливим ефективно планувати й моніторити екскурсійної діяльності в реальному часі. Адаптивність дизайну у поєднанні з компонентами миттєвого зворотного зв'язку гарантує стабільну та інтуїтивно зрозумілу роботу додатка на будь-яких пристроях.

3.3. Тестування роботи програмного комплексу

Тестування роботи розробленого програмного комплексу передбачає випробування його модулів з позиції кінцевого користувача для підтвердження відповідності реалізованого функціоналу заявленим вимогам. Основним інструментом перевірки виступає ручне функціональне тестування за методом чорної скриньки, що дозволяє змоделювати реальні сценарії взаємодії користувача з інтерфейсом додатка. Процес верифікації охоплює оцінку коректності обробки користувацьких дій, аналіз стійкості системи при

виникненні помилок введення, а також підтвердження надійності закладених бізнес-обмежень та механізмів захисту інформації.

Перевірка системи розпочинається з аналізу сценаріїв автентифікації користувачів, де досліджується реакція інтерфейсу на некоректні дії. При спробі входу з помилковим паролем або неіснуючим ідентифікатором електронної пошти клієнтська частина додатка успішно перехоплює відповідь сервера та виводить над формою попереджувальний банер червоного кольору про невірні облікових дані (рис. 3.30).

Увійти
Вхід для менеджерів та гідів.

[Увійти](#) [Зареєструвати нову компанію](#) [Приєднатися до компанії](#) [Налаштування гіда](#)

❗ Невірні облікові дані.

Електронна пошта або телефон *
olena.manager@kyivtours.test

Пароль *
.....

УВІЙТИ

Рис. 3.30. Неправильні дані

У разі успішного проходження перевірки та введення правильних параметрів система автоматично зчитує роль користувача із захищеного токена сесії та здійснює перенаправлення. Для облікового запису з повноваженнями менеджера програма відкриває повноцінний адміністративний робочий простір із розгорнутим бічним меню, що містить усі інструменти для координації екскурсійної діяльності та аналітики.

У межах тестування додаткових сценаріїв авторизації перевірено форму приєднання менеджера до чинної структури за допомогою унікального коду. При введенні невалідного цифрового ідентифікатора компанії система блокує реєстрацію на рівні бекенду та повертає текстове сповіщення про недійсний код,

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

що унеможливорює несанкціоноване підключення сторонніх осіб до корпоративного сховища (рис. 3.31).

Увійти
Реєстрація менеджера для приєднання до існуючої компанії за кодом.

[Увійти](#) [Зареєструвати нову компанію](#) [Приєднатися до компанії](#) [Налаштування гіда](#)

❗ Недійсний код компанії.

Ім'я *
тест

Прізвище *
тест

Електронна пошта *
test@gmail.com

Номер телефону *
0123456789

Пароль *

Код компанії *
987321

[ПРИЄДНАТИСЯ ДО ІСНУЮЧОЇ КОМПАНІЇ](#)

Рис. 3.31. Недійсний код компанії

Аналогічний механізм контролю бізнес-правил зафіксовано під час випробування форми первинного налаштування доступу для гідів. Спроба повторного встановлення пароля гідом для облікового запису, який уже пройшов процедуру активації, успішно блокується сервером із виведенням повідомлення про те, що пароль для цього акаунту вже налаштовано (рис. 3.32).

Увійти
Гіди, створені менеджерами, встановлюють тут свій пароль.

[Увійти](#) [Зареєструвати нову компанію](#) [Приєднатися до компанії](#) [Налаштування гіда](#)

❗ Пароль для цього акаунту вже встановлено.

Електронна пошта або телефон гіда *
dmytro.guide@kyivtours.test

Новий пароль *

[ВСТАНОВИТИ ПАРОЛЬ](#)

Рис. 3.32. Помилка верифікації при повторному налаштування пароля

Окремим етапом тестування виступає верифікація захисту внутрішніх сторінок від несанкціонованого доступу через введення URL-адреси в рядок веббраузера (рис. 3.33).

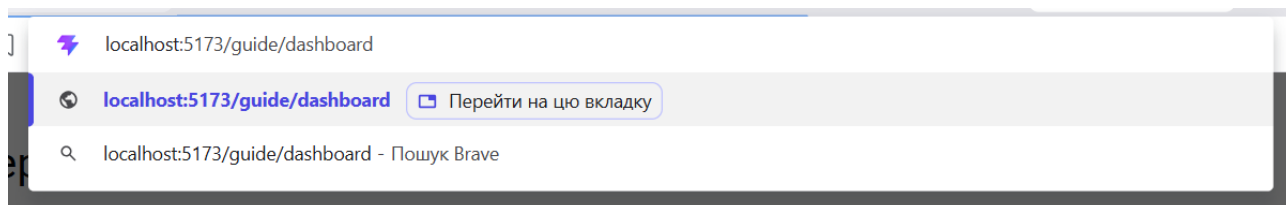


Рис. 3.33. Спроба доступу панелі гіда з акаунту менеджера

Спроба неавторизованої особи або користувача з недостатніми правами відкрити захищений маршрут блокується вбудованим компонентом контролю доступу, який перериває операцію та миттєво повертає клієнта на вихідну сторінку авторизації (рис. 3.34).

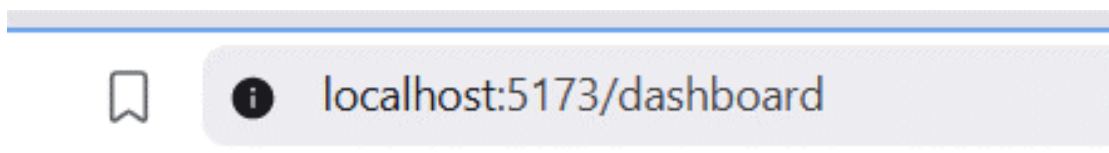


Рис. 3.34. Автоматичне переведення користувача на доступну йому сторінку

Важливо зазначити, що аналогічний захисний механізм автоматично спрацьовує при закінченні терміну дії JWT-токена, коли система самостійно завершує активну сесію та розлогінює користувача, а створення окремого рисунка для цього процесу є недоцільним через повну ідентичність фінального екрана стандартній формі входу.

Перевірка стійкості форм до помилок введення з боку користувача та верифікація механізмів локальної валідації клієнтської частини додатка виконується безпосередньо у процесі взаємодії з інтерфейсом. При спробі зберегти форму створення нової пам'ятки з порожніми обов'язковими текстовими полями назви або опису вбудовані валідатори блокують відправлення запиту на сервер. Інтерфейс програми підсвічує невалідні текстові зони червоним кольором, виводить під ними відповідні підказки про обов'язковість заповнення та повністю деактивує головну функціональну кнопку збереження (рис. 3.35).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

Створити пам'ятку

×

Назва

Введіть назву пам'ятки

Назва обов'язкова.

Опис

Введіть опис пам'ятки

Опис обов'язковий.

Години роботи

напр., 9:00-23:00

Години роботи обов'язкові.

Адреса

Пошук

+

−

Гончарі-Кожум'яки

Кудрявець

Кудрявець 174 m

Старокиївська гора 185 m

Нижня станція

Верхня станція

Старий Київ

Молодітське заті

ЗБЕРЕГТИ

СКАСУВАТИ

Рис. 3.35. Помилки валідації при пустих обов'язкових полях

Контроль дотримання форматів і масок даних так само відстежується під час заповнення специфічних полів. Введення довільного тексту або некоректного набору символів замість регламентованого часового шаблону 09:00-23:00 у полі годин роботи пам'ятки призводить до спрацьовування захисного алгоритму клієнтської частини. Система маркує відповідну зону форми червоним кольором і виводить чітке текстове роз'яснення щодо правильного вигляду очікуваного часового проміжку, не дозволяючи оператору завершити операцію до виправлення помилки (рис. 3.36).

Години роботи

9-23

Формат годин роботи має бути: 9:00-23:00.

Рис. 3.36. Неправильний формат тексту

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

Аналогічно аналізу піддається робота бізнес-обмежень інтерфейсу при спробі перевищення допустимих лімітів завантаження медіафайлів. Коли користувач намагається прикріпити до форми пам'ятки понад чотири фотокартки або додати файл, розмір якого перевищує встановлену межу, додаток автоматично перериває процес обробки даних. Інтерфейс залишає слоти для зображень порожніми та виводить текстове попередження про порушення правил завантаження контенту (рис. 3.37).

Додати фото

До 4 фото, не більше 3 МБ кожне. На комп'ютері перетягуйте для зміни порядку; на телефоні або планшеті двічі торкніться фото, щоб вибрати його, а потім торкніться місця для переміщення.

Кожне фото повинно бути розміром не більше 3 МБ, а загальна кількість не може перевищувати 4 фото.

Порожньо Порожньо Порожньо Порожньо

#1 #2 #3 #4

ЗБЕРЕГТИ СКАСУВАТИ

Рис. 3.37. Завантаження занадто великого файлу

Для всіх інших екранних форм системи, включаючи модулі створення екскурсійних турів, менеджменту гідів та оформлення бронювань, діють такі ж уніфіковані принципи перевірки, колірної підсвічування помилок та блокування дій користувача, що забезпечує надійний захист від некоректного введення даних на клієнтському рівні.

Контроль за дотриманням лімітів під час бронювання місць у системі перевіряється безпосередньо через екранну форму оформлення нової заявки. Коли менеджер намагається створити бронювання, у якому вказана кількість учасників перевищує обсяг вільних місць на обрану дату, система підсвічує поле введення червоним кольором і виводить під ним текстове застереження про те, що чисельність групи не може бути більшою за кількість доступних слотів. Крім того, інтерфейс повністю блокує відправлення невалідного запиту, запобігаючи виникненню помилки перевищення квот на рівні бази даних (рис. 3.38).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

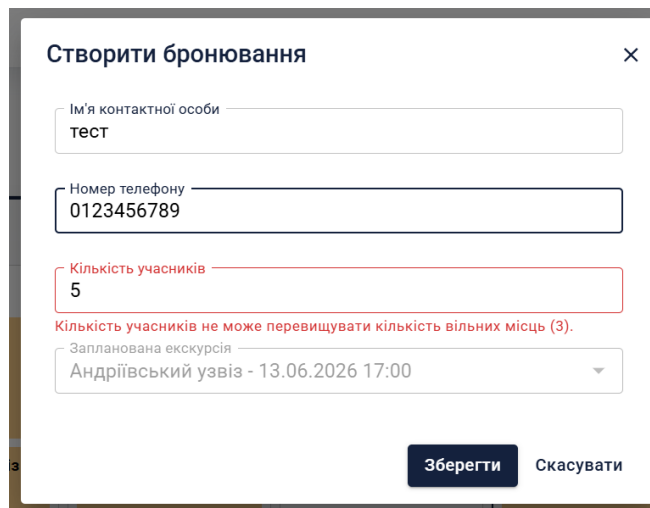


Рис. 3.38. Перевищення кількості учасників бронювання

Аналогічні захисні інструменти діють у системі під час спроби редагування або видалення запланованих екскурсій, які вже мають бронювання. При відкритті вікна редагування запланованої екскурсії із наявними клієнтськими заявками додаток виводить інформаційний банер блакитного кольору, який сповіщає про повне блокування зміни ключових полів дати, часу та закріпленого гіда. На рівні інтерфейсу дозволяється лише збільшення максимальної кількості місць для розширення групи, а всі інші параметри стають недоступними для редагування з метою захисту уже зареєстрованих туристів (рис. 3.39).

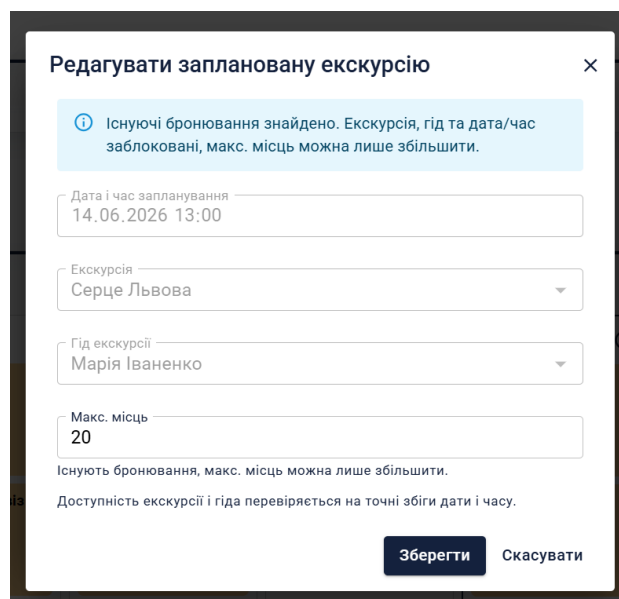


Рис. 3.39. Спроба редагування запланованої екскурсії

		Чорноус В. І.			ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

Якщо менеджер намагається видалити таку екскурсію через картку деталей у календарі, іконка стає неактивною, а при спробі взаємодії з нею додаток розгортає текстову підказку з формулюванням про неможливість видалення об'єкта за наявності бронювань. Додатково у нижній частині форми виводиться акцентне помаранчеве текстове попередження (рис. 3.40).

Рис. 3.40. Спроба видалення запланованої екскурсії

Контроль навантаження гідів виконується у процесі планування нових турів менеджером. Коли в системі створюється або редагується екскурсійний тур із закріпленням за конкретним гідом, алгоритм автоматично підраховує сумарну тривалість його робочих годин на поточний тиждень та порівнює її із гранично встановленим параметром. У разі, якщо новий розклад намагається перевищити максимальну кількість годин, система показує відповідне попередження. Таке сповіщення безпосередньо у формі сигналізує менеджеру про виникнення овертайму і допомагає коригувати графік та розподіляти завдання між іншими гідами без порушення встановлених нормативів праці (рис. 3.41).

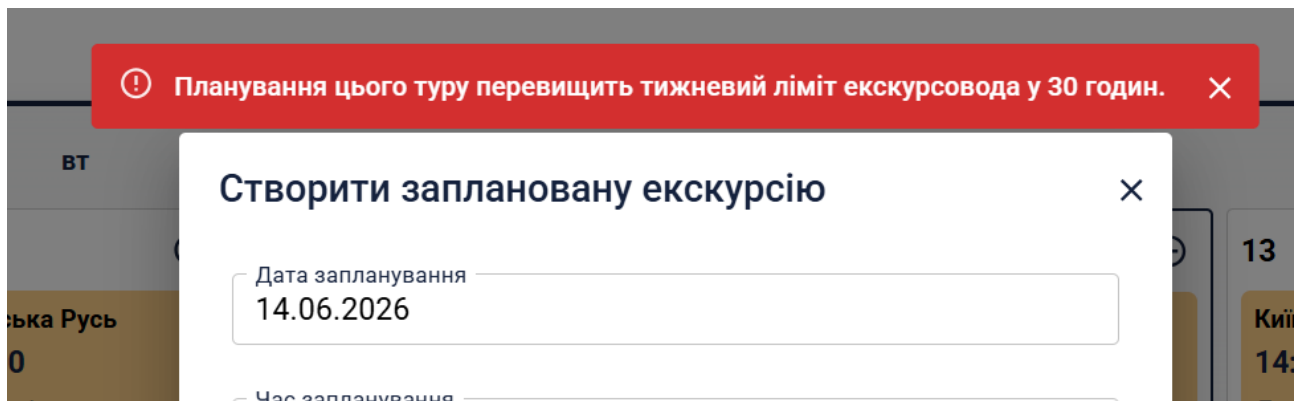


Рис. 3.41. Перевищення максимальної кількості годин гіда

Таким чином, проведене ручне тестування підтвердило працездатність розробленого програмного комплексу та його стійкість до помилок користувача. Надійне функціонування механізмів локальної валідації у поєднанні із контролем бізнес-обмежень на рівні сервера гарантує захист бази даних від некоректного введення інформації та запобігає виникненню конфліктів у розкладі. Візуальне підсвічування невалідних полів, інформаційні банери та динамічні індикатори навантаження забезпечують інтуїтивно зрозумілий зворотний зв'язок.

Висновки до розділу 3

У третьому розділі було успішно вирішено завдання практичного впровадження, візуалізації та верифікації працездатності розробленого програмного комплексу. Сформований порядок встановлення та налаштування параметрів системи забезпечує розгортання всіх її архітектурних ланок і стабільну конфігурацію середовища для подальшої експлуатації платформи. Спроектована структура графічного інтерфейсу утворює цілісний та адаптивний робочий простір. Результати проведеного функціонального тестування підтвердили надійність закладеної бізнес-логіки та стійкість системи до помилок користувача.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

ВИСНОВКИ

У випускній кваліфікаційній роботі відповідно до поставленої мети розроблено архітектуру, алгоритми функціонування та здійснено програмну реалізацію комплексу веб-орієнтованої інформаційної системи планування маршрутів для гідів. Основний результат дослідження полягає у створенні масштабованої платформи, яка забезпечує автоматизацію процесів моделювання екскурсійних турів, координації внутрішніх людських ресурсів компанії та оперативного обліку клієнтських замовлень в єдиному цифровому просторі з гарантованою логічною ізоляцією даних незалежних орендарів.

Оцінюючи ступінь виконання завдань, можна стверджувати, що всі поставлені науково-технічні задачі вирішено в повному обсязі, а реалізований функціонал системи демонструє відповідність технічним вимогам. У межах створеного програмного комплексу успішно втілено підсистему на основі спільної реляційної бази даних Microsoft SQL Server із автоматичним накладанням глобальних фільтрів через Entity Framework Core, безстанову автентифікацію користувачів за допомогою JWT-токенів із розмежуванням рольових моделей менеджера та гіда, а також гнучкі модулі ведення журналів. Реалізовано інтерактивний календар розкладу з тижневою та місячною сітками планування, картографічний модуль на базі Leaflet для просторової прив'язки локацій та автоматичного геокодування адрес, підсистему формування маршрутів із підтримкою динамічної зміни черговості об'єктів за принципом drag-and-drop, а також функціональний модуль аналітики та статистики з інтерактивними графіками динаміки операційної діяльності й таблицями рейтингів успішності.

Впроваджені в шарову серверну архітектуру бізнес-алгоритми автоматичної перевірки часових конфліктів у графіках гідів, контролю ліміту вільних місць при оформленні бронювань та моніторингу граничного тижневого навантаження персоналу продемонстрували безвідмовність роботи. Результати проведеного ручного функціонального тестування за методом чорної скриньки підтвердили високу стійкість інтерактивних екранних форм до некоректних дій

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

оператора, правильність спрацьовування механізмів локальної валідації та декларативних правил FluentValidation на бекенді, що повністю доводить надійність та інженерну зрілість створеного коду.

Практична цінність одержаних результатів полягає у створенні повнофункціонального, кросплатформеного веб-застосунку, готового до безпосереднього впровадження в повсякденну діяльність туристичних агенцій. Застосування розробленої платформи дозволяє підприємствам ліквідувати ризики виникнення накладок у розкладах, мінімізувати час обробки клієнтських заявок, оптимізувати використання робочого часу гідів та виключити вплив людського фактора при веденні обліку.

Напрямами подальшого вдосконалення та розвитку розробленого програмного комплексу є інтеграція інтелектуальних модулів автоматичної оптимізації та побудови найкоротших екскурсійних маршрутів. Перспективним є також розширення функціоналу системи в частині впровадження автоматизованої підсистеми сповіщень клієнтів і персоналу через популярні месенджери, а також розробка мобільного застосунку для гідів з метою забезпечення комфортного доступу до карт та індивідуальних графіків у польових умовах.

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sigsworth W. What Is a Bottleneck in Business? The ultimate guide to detect and fix them every time. Pipedrive. URL: <https://www.pipedrive.com/en/blog/what-is-a-bottleneck-in-business> (дата звернення: 09.06.2026).
2. Lindemulder G., Kosinski M. What is role-based access control (RBAC)? IBM Think. URL: <https://www.ibm.com/think/topics/rbac> (дата звернення: 09.06.2026).
3. Seetharamu G. N. Complete guide to multi-tenant architecture. Medium. URL: <https://medium.com/@seetharamugn/complete-guide-to-multi-tenant-architecture-d69b24b518d6> (дата звернення: 09.06.2026).
4. TourCMS Features. TourCMS. URL: <https://www.tourcms.com/features/> (дата звернення: 09.06.2026).
5. Travefy Itinerary Builder & Travel Itinerary Software. Travefy. URL: <https://travefy.com/products/itinerary> (дата звернення: 09.06.2026).
6. Booking Website Features. SimplyBook.me. URL: <https://simplybook.me/uk/booking-system-features#booking-website> (дата звернення: 09.06.2026).
7. Gos K., Zabierowski W. The Comparison of Microservice and Monolithic Architecture. 2020 IEEE 16th International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) : Conference Proceedings (Lviv, 14–17 May 2020). IEEE, 2020. P. 150–153. DOI: 10.1109/MEMSTECH49584.2020.9109514.
8. Nyabuto G. M., Mony V., Mbugua S. Architectural Review of Client-Server Models. International Journal of Scientific Research & Engineering Trends. 2024. Vol. 10, Issue 1. ISSN 2395-566X.
9. Muditha K. Building the Future: Choosing Between Client-Server, Monolithic, and Microservices. Medium. URL: <https://medium.com/@kulanthamuditha66/building-the-future-choosing-between-client-server-monolithic-and-microservices-2a194677f8eb> (дата звернення: 10.06.2026).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

10. Multipage Application (MPA). Sanity. URL: <https://www.sanity.io/glossary/multipage-application> (дата звернення: 10.06.2026).
11. What Is a Single-Page Application (SPA)? Software Mind. URL: <https://softwaremind.com/blog/what-is-a-single-page-application-spa/> (дата звернення: 09.06.2026).
12. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.
13. What is REST. REST API Tutorial. URL: <https://restfulapi.net/> (дата звернення: 09.06.2026).
14. Multi-Tenant Database Architecture Patterns Explained. Bytebase. URL: <https://www.bytebase.com/blog/multi-tenant-database-architecture-patterns-explained/> (дата звернення: 10.06.2026).
15. Toumi A. Data Transfer Objects (DTOs): A Comprehensive Guide. Medium. URL: <https://abderahmanetoumi.medium.com/data-transfer-objects-dtos-a-comprehensive-guide-2d00e8fa2ec3> (дата звернення: 10.06.2026).
16. What Is ASP.NET Core And Advantages Of Using ASP.NET Core. C# Corner. URL: <https://www.c-sharpcorner.com/article/what-is-asp-net-core-and-advantages-of-using-asp-net-core-how-to-setup-asp-net/> (дата звернення: 10.06.2026).
17. Pijacek R. Microsoft SQL Server Pros and Cons. LearnSQL. URL: <https://learnsql.com/blog/microsoft-sql-server-pros-and-cons/> (дата звернення: 10.06.2026).
18. Roth D., Anderson R., Latham L. Entity Framework Core Overview. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 10.06.2026).
19. React Documentation. React. URL: <https://react.dev/> (дата звернення: 10.06.2026).
20. TypeScript Documentation. TypeScript. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 10.06.2026).
21. Material UI Getting Started. MUI. URL: <https://mui.com/material-ui/getting-started/> (дата звернення: 10.06.2026).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

22. React Router Documentation. React Router. URL: <https://reactrouter.com/home> (дата звернення: 10.06.2026).
23. Jeremy S. FluentValidation Documentation. FluentValidation. URL: <https://docs.fluentvalidation.net/en/latest/> (дата звернення: 09.06.2026).
24. Bogard J. AutoMapper Documentation. AutoMapper. URL: <https://docs.automapper.org/en/stable/> (дата звернення: 09.06.2026).
25. Mermaid Live Editor. URL: <https://mermaid.live/> (дата звернення: 11.06.2026).
26. Kolade C. Database Normalization: A Complete Guide to 1NF, 2NF, 3NF with Table Examples. freeCodeCamp. 2024. URL: <https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/> (дата звернення: 11.06.2026).
27. Acceldata. Referential Integrity: Why It's Vital for Databases. Acceldata Blog. 2023. URL: <https://www.acceldata.io/blog/referential-integrity-why-its-vital-for-databases> (дата звернення: 11.06.2026).
28. Atlassian. How SQL Indexing Works. Atlassian Data Guide. URL: <https://www.atlassian.com/data/sql/how-indexing-works> (дата звернення: 11.06.2026).
29. Microsoft. Dependency injection - Service lifetimes - .NET. Microsoft Learn. URL: <https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection/service-lifetimes> (дата звернення: 11.06.2026).
30. Mukesh M. Global Exception Handling in ASP.NET Core. Code with Mukesh. 2023. URL: <https://codewithmukesh.com/blog/global-exception-handling-in-aspnet-core/> (дата звернення: 11.06.2026).
31. Okta. What Is Token-Based Authentication? Okta Identity 101. URL: <https://www.okta.com/identity-101/what-is-token-based-authentication/> (дата звернення: 11.06.2026).

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

Приклад методу service ScheduledTourService

```

private void EnsureGuideAvailability(TourGuide guide,
DateTime newTourStart, float durationInHours, int
excludeScheduledTourId, int tourId)
{
    EnsureGuideWeeklyHours(guide, newTourStart,
durationInHours, excludeScheduledTourId);

    var sameTourBreak = TimeSpan.Zero;
    var differentTourBreak = TimeSpan.FromMinutes(30);

    var newTourEnd = newTourStart.AddHours(durationInHours);

    foreach (var scheduledTour in guide.ScheduledTours
        .Where(t => t.Id != excludeScheduledTourId &&
t.ScheduledDate.Date == newTourStart.Date))
    {
        var otherStart = scheduledTour.ScheduledDate;
        var otherEnd =
otherStart.AddHours(scheduledTour.Tour.DurationInHours);

        var requiredBreak = scheduledTour.TourId == tourId ?
sameTourBreak : differentTourBreak;

        if (otherStart < newTourEnd && newTourStart <
otherEnd)
        {
            throw new BadRequestException("У цей часовий
проміжок вже заплановано тур для цього екскурсовода");
        }

        if (otherEnd <= newTourStart)
        {

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмоторя О. В.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        var gap = newTourStart - otherEnd;
        if (gap < requiredBreak)
        {
            throw new BadRequestException("Екскурсоводу
потрібна більша перерва перед цим туром");
        }
    }
    else if (otherStart >= newTourEnd)
    {
        var gap = otherStart - newTourEnd;
        if (gap < requiredBreak)
        {
            throw new BadRequestException("Екскурсоводу
потрібна більша перерва після цього туру");
        }
    }
}
}

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

Реєстрація обробника помилок у файлі Program.cs

```

app.UseExceptionHandler(exceptionHandlerApp =>
{
    exceptionHandlerApp.Run(async context =>
    {
        var feature =
context.Features.Get<IExceptionHandlerFeature>();
        if (feature?.Error is null)
        {
            context.Response.StatusCode =
StatusCodes.Status500InternalServerError;
            return;
        }

        var handler =
context.RequestServices.GetRequiredService<IExceptionHandler>();
        await handler.TryHandleAsync(context, feature.Error,
context.RequestAborted);
    });
});

```

Реалізація класу обробника помилок

```

public class
GlobalExceptionHandler(ILogger<GlobalExceptionHandler> logger) :
IExceptionHandler
{
    private readonly ILogger<GlobalExceptionHandler> _logger =
logger;

    public async ValueTask<bool> TryHandleAsync(HttpContext
httpContext, Exception exception, CancellationToken
cancellation_token)
    {

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _logger.LogError(
            exception, "Exception occurred: {Message}",
            exception.Message);

        var problemDetails = exception switch
        {
            ValidationException validationException => new
ProblemDetails
            {
                Status = StatusCodes.Status400BadRequest,
                Title = "Помилка валідації",
                Detail = validationException.Message
            },
            BadRequestException badRequestException => new
ProblemDetails
            {
                Status = StatusCodes.Status400BadRequest,
                Title = "Некоректний запит",
                Detail = badRequestException.Message
            },
            _ => new ProblemDetails
            {
                Status
                =
                StatusCodes.Status500InternalServerError,
                Title = "Помилка сервера",
                Detail = "Виникла непередбачена помилка."
            }
        };

        httpContext.Response.StatusCode
        =
        problemDetails.Status!.Value;

        await httpContext.Response
            .WriteAsJsonAsync(problemDetails,
                cancellationToken);

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```
        return true;
    }
}
```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		97

AuthProvider

```

export const AuthProvider: React.FC<React.PropsWithChildren> = ({
  children }) => {
  const [session, setSession] = useState<AuthSession | null>(() =>
    getAuthSession());
  const [profile, setProfile] = useState<AuthMeDto | null>(null);
  const [isProfileLoading, setIsProfileLoading] = useState(false);

  const logout = useCallback(() => {
    if (session) {
      void authService.logout().catch(() => {
        // Local logout must still proceed even if revoke endpoint
        fails.
      });
    }

    clearAuthSession();
    setSession(null);
    setProfile(null);
    setIsProfileLoading(false);
  }, [session]);

  const refreshProfile = useCallback(async () => {
    if (!session) {
      setProfile(null);
      setIsProfileLoading(false);
      return;
    }

    setIsProfileLoading(true);

    try {
      const currentUser = await authService.getMe();
      setProfile(currentUser);
    } catch {
      logout();
    } finally {
      setIsProfileLoading(false);
    }
  }, [logout, session]);

  const setSessionFromAuth = useCallback(
    async (nextSession: AuthSession) => {
      saveAuthSession(nextSession);
      setSession(nextSession);
      setIsProfileLoading(true);

      try {

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        const currentUser = await authService.getMe();
        setProfile(currentUser);
    } catch {
        logout();
        throw new Error('Failed to load profile after
authentication');
    } finally {
        setIsProfileLoading(false);
    }
},
[logout],
);

useEffect(() => {
    if (!session) {
        setProfile(null);
        setIsProfileLoading(false);
        return;
    }

    void refreshProfile();
}, [session, refreshProfile]);

useEffect(() => {
    if (typeof window === 'undefined') {
        return;
    }

    const handleUnauthorized = () => {
        logout();
    };

    window.addEventListener(AUTH_UNAUTHORIZED_EVENT,
handleUnauthorized);

    return () => {
        window.removeEventListener(AUTH_UNAUTHORIZED_EVENT,
handleUnauthorized);
    };
}, [logout]);

const value = useMemo<AuthContextValue>(
    () => ({
        session,
        profile,
        isProfileLoading,
        setSessionFromAuth,
        refreshProfile,
        logout,
    }),

```

```

        [session, profile, isProfileLoading, setSessionFromAuth,
refreshProfile, logout],
    );

    return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
};

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент FeedbackToast

```

export type FeedbackSeverity = 'success' | 'error' | 'info' |
'warning';

interface FeedbackToastProps {
  open: boolean;
  message: string;
  severity?: FeedbackSeverity;
  autoHideDuration?: number;
  onClose: () => void;
}

const FeedbackToast: React.FC<FeedbackToastProps> = ({
  open,
  message,
  severity = 'info',
  autoHideDuration = 4000,
  onClose,
}) => {
  return (
    <Snackbar
      open={open}
      autoHideDuration={autoHideDuration}
      onClose={(_, reason) => {
        if (reason === 'clickaway') {
          return;
        }
        onClose();
      }}
      anchorOrigin={{ vertical: 'top', horizontal: 'center' }}
      sx={{ mt: 1 }}
    >
      <Alert
        onClose={onClose}
        severity={severity}
        variant="filled"
        sx={{
          width: '100%',
          minWidth: { xs: 280, sm: 360 },
          '& .MuiAlert-icon': {
            color: 'inherit',
          },
        }}
      >
        {message}
      </Alert>
    </Snackbar>
  );
};

```

		Чорноус В. І.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Чижмотря О. В.				101
Змн.	Арк.	№ докум.	Підпис	Дата		