НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ) інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ) _____	**Завідувач кафедри**
інформаційних технологій	комп'ютерних наук
(назва факультету (ННІ))	(назва кафедри)
_____ Ігор Болбот	_____ Белла Голуб
(підпис) (ім'я ПРІЗВИЩЕ)	(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 20_ р.	"___" _____ 20_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему **Рекомендаційна система для планування подорожей на основі машинного навчання та персоналізованих уподобань користувачів**

Спеціальність 122 "Комп'ютерні науки"
(код і найменування)

Освітня програма Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
к.т.н., доцент _____ Белла Голуб
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
д.ек.н., проф. _____ Андрій БРИТАН
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ Дмитро ЯНИЦЬКИЙ
(підпис) (ім'я ПРІЗВИЩЕ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) __ інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри __комп'ютерних наук_
___доцент, к.т.н._____ _____ _Белла Голуб
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"01" листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Яницькому Дмитру Миколайовичу
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки »
(код і назва)

Освітня програма ____Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____

Тема магістерської кваліфікаційної роботи Рекомендаційна система для планування подорожей на основі машинного навчання та персоналізованих уподобань користувачів затверджена наказом ректора НУБіП України від "_1_" листопада__2024р. №1964 «С»
Термін подання завершеної роботи на кафедру 11 листопада 2025
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: Нормативно-довідкова та наукова література з інформаційних систем, баз даних, методів візуалізації та обробки даних; технічна документація з проєктування та розробки програмних комплексів; програмні засоби для розробки веб-застосунків і інтерактивних інтерфейсів; вимоги стандартів до побудови інформаційних систем і користувацьких інтерфейсів.

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання та архітектурне проєктування системи
3. Реалізація програмного забезпечення та технологічна інфраструктура системи
4. Тестування та оцінювання ефективності системи

Перелік графічного матеріалу (за потреби) презентація, постер, схеми та діаграми архітектури системи

Дата видачі завдання "__1__" листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ ___Андрій БРИТАН
(підпис) (ім'я ПРІЗВИЩЕ))

Завдання прийняв до виконання _____ Дмитро ЯНИЦЬКИЙ___
(підпис) (ім'я ПРІЗВИЩЕ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API — Application Programming Interface, прикладний програмний інтерфейс
2. ML — Machine Learning, машинне навчання
3. DL — Deep Learning, глибоке навчання
4. k-means — алгоритм кластеризації за методом k-середніх
5. SSE — Sum of Squared Errors, сумарна квадратична помилка
6. UI — User Interface, інтерфейс користувача
7. UX — User Experience, користувацький досвід
8. ETL — Extract, Transform, Load, процес отримання та оброблення даних
9. DB — Database, база даних
10. SQL — Structured Query Language, мова структурованих запитів
11. NoSQL — нереляційна модель зберігання даних
12. JSON — JavaScript Object Notation, формат структурованих даних
13. OIDC — OpenID Connect, протокол авторизації
14. OAuth2 — протокол автентифікації й авторизації
15. REST — Representational State Transfer, стиль архітектури веб-сервісів
16. CDN — Content Delivery Network, мережа доставки контенту
17. HTTP — HyperText Transfer Protocol, протокол передавання даних
18. HTTPS — HTTP Secure, протокол захищеного передавання даних
19. RPS — Requests Per Second, кількість запитів за секунду
20. CPU — Central Processing Unit, центральний процесор
21. RAM — Random Access Memory, оперативна пам’ять
22. KPI — Key Performance Indicator, ключовий показник ефективності
23. CI/CD — Continuous Integration / Continuous Deployment, безперервна інтеграція й розгортання

24. GDS — Global Distribution System, глобальна система туристичних бронювань

25. OTA — Online Travel Agency, онлайн-агентство подорожей

26. SHAP — SHapley Additive exPlanations, метод пояснення моделей ML

27. NLP — Natural Language Processing, оброблення природної мови

28. K8s — Kubernetes, система оркестрації контейнерів

29. GPU — Graphics Processing Unit, графічний процесор

30. CRUD — Create, Read, Update, Delete, базові операції над даними

31. JWT — JSON Web Token, токен доступу у протоколах авторизації

32. DBMS — Database Management System, система керування базами даних

ВСТУП

Туристична індустрія сьогодні є однією з найдинамічніших сфер світової економіки, у якій зростає роль інформаційних технологій для автоматизації процесів планування подорожей, аналізу ринку та персоналізації користувацького досвіду. Традиційні способи вибору маршрутів і формування туристичних пакетів не відповідають вимогам сучасного користувача, який очікує на індивідуальний підхід, швидку обробку інформації та врахування власних інтересів, фінансових і часових обмежень. Сучасні досягнення у галузі машинного навчання створюють можливості для побудови інтелектуальних систем, здатних автоматично аналізувати великі обсяги даних, виявляти закономірності у поведінці користувачів і пропонувати персоналізовані рішення. Використання таких рекомендаційних систем у сфері туризму сприяє підвищенню ефективності туристичних сервісів, розвитку смарт-туризму та розширенню доступності інформації для користувачів [1]. В умовах зростання конкуренції на ринку туристичних онлайн-платформ впровадження інтелектуальних систем рекомендацій стає ключовим чинником диференціації сервісів і підвищення рівня задоволеності клієнтів [2].

Мета дослідження полягає у розробленні рекомендаційної системи для планування подорожей, що використовує методи машинного навчання для формування персоналізованих туристичних пропозицій, ураховуючи вподобання, історію подорожей, соціальні сигнали та контекстні фактори (сезонність, погоду, бюджет, геолокацію).

Для досягнення мети дослідження потрібно виконати наступні **завдання**:

– провести системний аналіз предметної області рекомендаційних систем у сфері туризму;

– дослідити сучасні підходи до машинного навчання, що забезпечують персоналізацію контенту;

– побудувати модель користувача на основі поведінкових, демографічних і контентних ознак;

– розробити архітектуру системи, яка поєднує модулі збору, оброблення, класифікації та рекомендації даних;

– створити програмну реалізацію системи з використанням Python-технологій (Flask, Pandas, scikit-learn, TensorFlow) і перевірити її ефективність на реальних даних;

– здійснити тестування та оцінювання якості рекомендацій за допомогою метрик точності (Precision, Recall, F-measure).

Об’єктом дослідження є процес інтелектуального планування подорожей із використанням методів машинного навчання та аналізу користувацьких уподобань.

Предметом дослідження є моделі, методи та програмні засоби побудови рекомендаційних систем, орієнтованих на персоналізацію туристичних послуг і маршрутів.

Методи дослідження базуються на поєднанні алгоритмів машинного навчання (колаборативна та контентна фільтрація, кластеризація K-means, деревові моделі рішень, нейронні мережі) та методів статистичного аналізу даних. Для оброблення інформації використовуються бібліотеки Python (NumPy, Pandas, scikit-learn), а для побудови веб-інтерфейсу - Flask і HTML-CSS-JS. Експериментальна частина ґрунтується на аналізі відкритих даних з API туристичних сервісів (TripAdvisor, Google Places, Booking) і даних соціальних мереж, що містять відгуки користувачів, оцінки та геомітки.

Наукова новизна отриманих результатів полягає у розробленні комбінованої моделі рекомендацій, яка інтегрує контентно-орієнтований, колаборативний і контекстно-залежний підходи для формування динамічних маршрутів подорожей. Запропонована система відрізняється від існуючих тим, що використовує адаптивну модель оновлення вподобань користувача на основі зворотного зв’язку, що підвищує точність рекомендацій у реальному часі. Крім того, реалізовано модуль прогнозування сезонних трендів і класифікації

туристичних локацій за рівнем привабливості, що дає змогу формувати більш релевантні рекомендації навіть для нових користувачів.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області системи рекомендацій подорожей на основі машинного навчання

Предметна область дослідження охоплює процес формування індивідуальних туристичних маршрутів та рекомендацій з використанням методів машинного навчання, які забезпечують персоналізацію вибору подорожей відповідно до вподобань користувача, контекстних обмежень і динамічних зовнішніх факторів. У межах цієї системи передбачається аналіз великої кількості даних, що надходять з різних джерел (API туристичних сервісів, картографічних платформ, погодних і подієвих агрегаторів), з метою побудови узгодженої моделі користувача та автоматичного формування релевантних рекомендацій.

На рис. 1.1 подано структурну модель предметної області, що відображає основні сутності та взаємозв’язки між користувачем, профілем його вподобань, контекстом подорожі, джерелами даних і процесом формування маршруту.

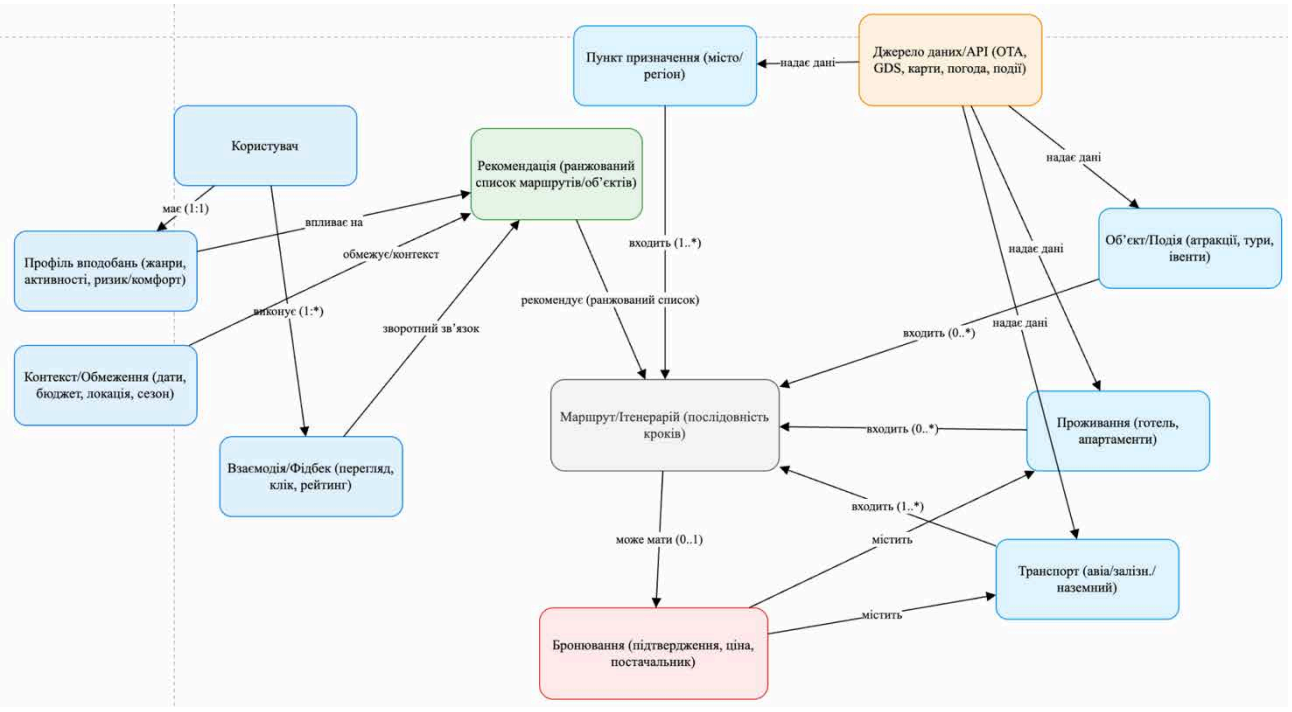

Рис. 1.1. Структура предметної області рекомендаційної системи планування подорожей

Система оперує ключовими сутностями: користувач, який має профіль уподобань (жанри, активності, рівень ризику/комфорту); контекст/обмеження, що визначають часові, бюджетні та географічні параметри подорожі; взаємодія/фідбек, що формується внаслідок оцінок, кліків і переглядів, та слугує зворотним зв'язком для навчання моделі. Основою процесу є рекомендаційний модуль, який на основі зазначених вхідних даних генерує ранжований список маршрутів і об'єктів, ураховуючи вплив контексту та історію поведінки користувача. Результатом є маршрут (генерація послідовності кроків), який може включати різні типи об'єктів — туристичні події, проживання, транспортні засоби, що формуються на основі даних, отриманих з зовнішніх джерел / API (OTA, GDS, карти, погода, події). Завершальним етапом виступає бронювання, що містить підтвердження, ціну та постачальника послуг.

Для узагальнення основних характеристик предметної області подано табл. 1.1, яка описує сутності, їхню роль у системі та тип взаємозв'язків між ними.

Таблиця 1.1

Основні сутності предметної області рекомендаційної системи подорожей

№	Сутність	Опис функції у системі	Тип взаємозв'язку
1	Користувач	Ініціатор запиту, джерело вподобань і зворотного зв'язку	має профіль (1:1)
2	Профіль вподобань	Набір характеристик (жанри, активності, рівень ризику)	пов'язаний з користувачем
3	Контекст/обмеження	Час, бюджет, локація, сезонність	впливає на рекомендації
4	Джерело даних/API	Постачальник інформації (карти, OTA, GDS, події, погода)	надає дані об'єктам
5	Об'єкт/подія	Туристичні атракції, тури, івенти	входить до маршруту
6	Проживання	Готелі, апартаменти, місця ночівлі	частина маршруту
7	Транспорт	Авіа, залізничний, наземний	елемент маршруту

8	Рекомендація	Ранжований список маршрутів або об’єктів	формується системою

Продовження таблиці 1.1

9	Маршрут/генерація кроків	Послідовність дій, сформована системою	містить об’єкти
10	Бронювання	Підтвердження вибору, ціна, постачальник послуг	пов’язане з маршрутом

Предметна область системи охоплює повний цикл взаємодії користувача із сервісом – від формування індивідуального профілю до автоматичного генерування маршруту та підтвердження бронювань. Інтелектуальні алгоритми аналізують історію користувача, оцінки схожих груп і контекстні параметри, що дозволяє створювати гнучкі, релевантні та контекстно-залежні туристичні пропозиції у реальному часі. Така система підвищує ефективність прийняття рішень і сприяє розвитку персоналізованих цифрових сервісів у сфері туризму [1].

1.2 Теоретико- методологічні засади та стан наукових досліджень

Сучасні рекомендаційні системи є важливою складовою інтелектуальних інформаційних технологій, що базуються на використанні методів машинного навчання, аналізу даних і штучних нейронних мереж. В основі таких систем лежить побудова моделей прогнозування інтересів користувачів на основі історичних даних про їхню поведінку, оцінки, відгуки, геолокацію та інші контекстні параметри. Наукові праці провідних дослідників, таких як Ricci, Rokach, Shapira [1], Adomavicius, Tuzhilin [2], He, Liao, Zhang [3], визначають рекомендаційні системи як прикладні реалізації парадигми “data-driven personalization”, де ключову роль відіграють алгоритми колаборативної фільтрації, контентного аналізу та гібридні підходи.

У загальному вигляді модель нейронної мережі, що застосовується для рекомендацій, складається з вхідного шару, одного або кількох прихованих шарів і вихідного шару (рис. 1.2). Приховані шари відповідають за формування латентних ознак користувачів та елементів, а процес навчання полягає в мінімізації функції втрат між передбаченим і реальним рейтингом.

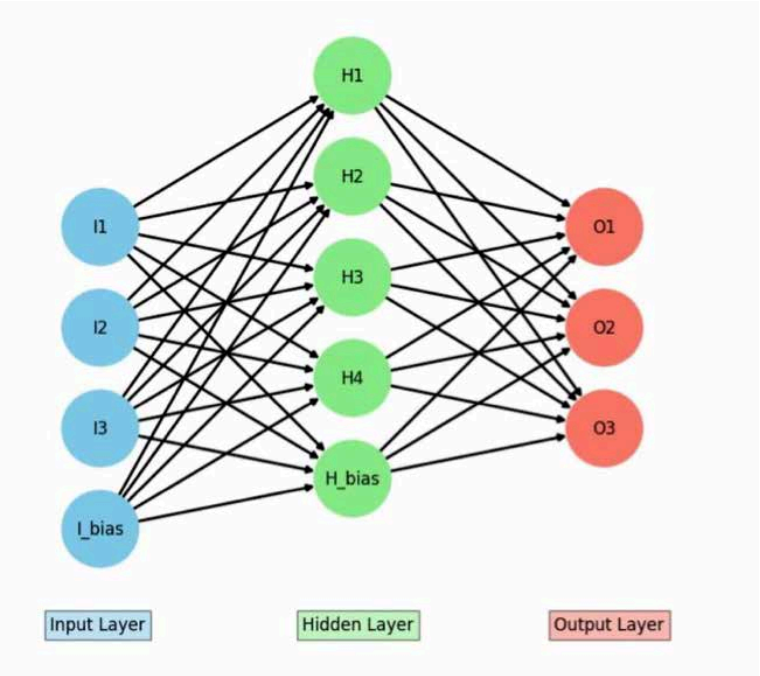

Рис. 1.2. Типова структура штучної нейронної мережі для задачі прогнозування уподобань користувача

Згідно з підходом He et al. (2017) [3], традиційна матрична факторизація, яка ґрунтується на добутку векторів користувача й об’єкта, може бути розширена за допомогою нейронного шару, що навчається нелінійних взаємозв’язків (рис. 1.3). Це дозволяє отримувати більш глибокі репрезентації взаємодій і підвищувати точність рекомендацій.

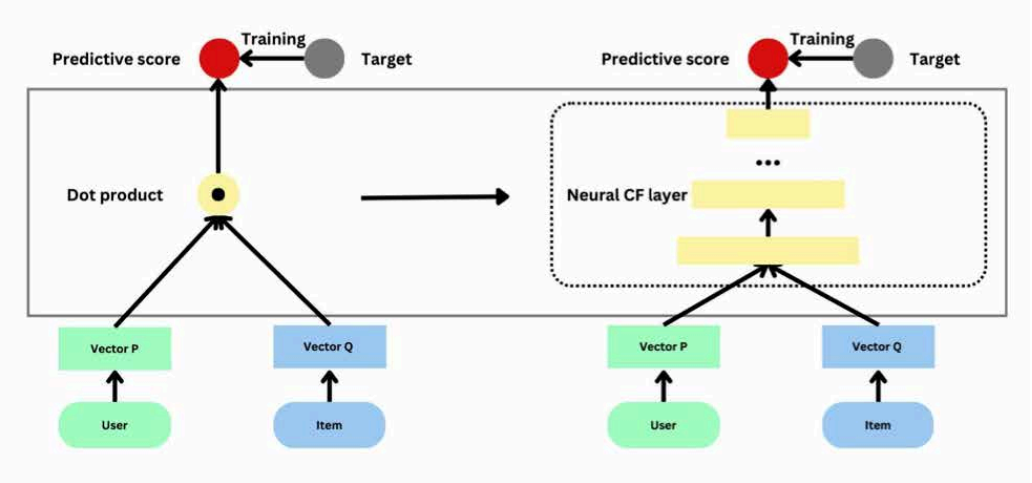

Рис. 1.3. Схема нейронної колаборативної фільтрації (NeuMF) для прогнозування оцінки користувача

Подальшим розвитком цих підходів стали графові нейронні мережі (Wu et al., 2022 [4]), які дають змогу будувати граф взаємозв'язків між користувачами, маршрутами, об'єктами та подіями (рис. 1.4). Це дає змогу враховувати не лише подібність користувачів за ознаками, а й складні топологічні структури у просторі зв'язків між туристичними локаціями.

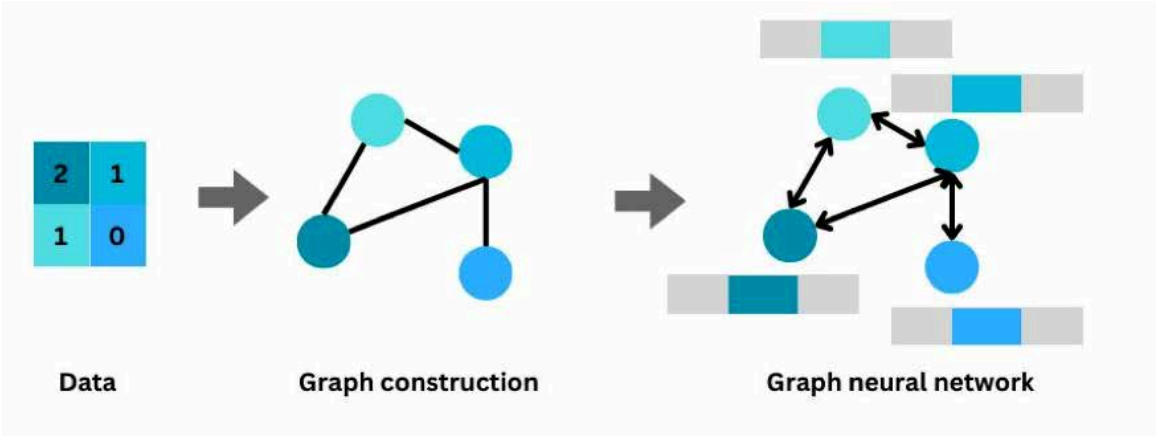

Рис. 1.4. Принцип побудови графової нейронної мережі для оброблення даних рекомендаційної системи

Крім цього, для виділення суттєвих ознак і зменшення розмірності даних у рекомендаційних системах широко застосовуються автоенкодери (Hinton, Salakhutdinov, 2006 [5]), які складаються з двох частин – енкодера та декодера (рис. 1.5). Вони дають змогу відновити основні латентні структури даних,

зменшуючи інформаційний шум і підвищуючи стійкість моделі до нерівномірності даних.

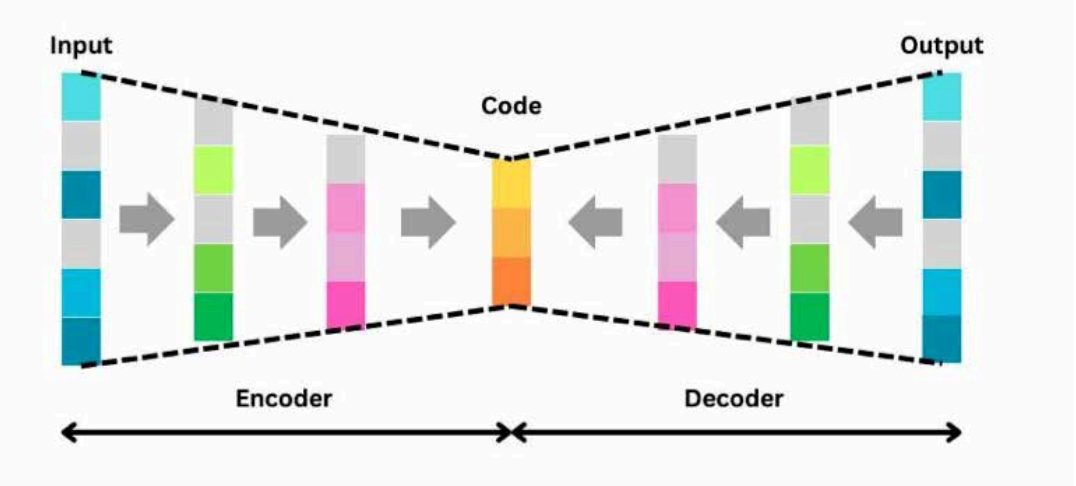

Рис. 1.5. Архітектура автоенкодера для формування латентних представлень користувацьких ознак

Окрему групу методів становлять обмежувальні машини Больцмана (Salakhutdinov & Hinton, 2009 [6]), які використовуються для побудови глибоких моделей колаборативного фільтрування (рис. 1.6). Такі мережі дозволяють виявляти приховані закономірності між користувачами та об’єктами навіть при наявності частково заповнених матриць рейтингів.

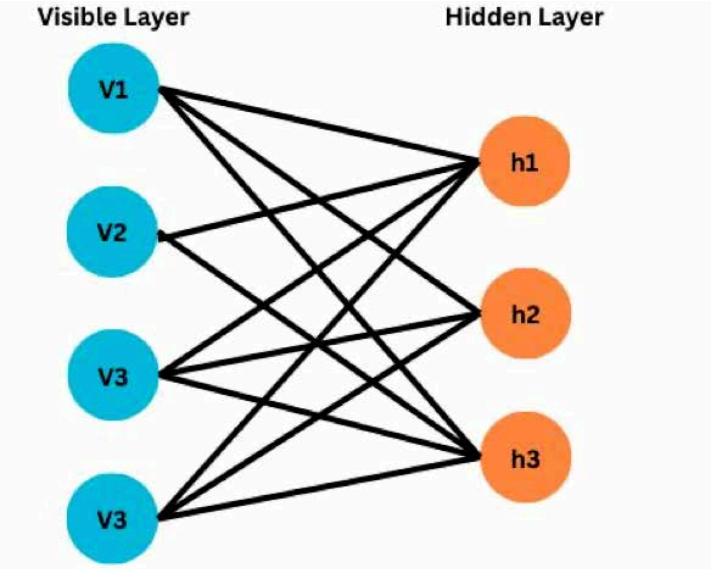

Рис. 1.6. Обмежувальна машина Больцмана (RBM) як модель для глибинного колаборативного фільтрування

Аналіз наукових джерел свідчить, що більшість сучасних систем орієнтовані на однотипні дані (наприклад, рейтинги або кліки), у той час як туризм є багатофакторною доменною областю, де релевантність залежить від часу, погоди, бюджету, місця, уподобань і соціальної поведінки. Отже, постає потреба у розробленні гібридної моделі, здатної інтегрувати кілька типів ознак – поведінкових, контекстних і контентних – у єдину архітектуру машинного навчання.

Результуюча наукова новизна дослідження полягає у створенні комбінованої рекомендаційної системи для планування подорожей, яка поєднує нейронну колаборативну фільтрацію, графові репрезентації та автоенкодерну обробку даних. Це забезпечує динамічне формування персоналізованих туристичних маршрутів, адаптивне до змін контексту користувача та умов середовища, що дозволяє підвищити точність і релевантність рекомендацій у реальному часі.

1.3 Огляд інформаціних джерел та існуючих рішень

На сучасному ринку цифрових сервісів для подорожей існує низка популярних рекомендаційних систем, які застосовують алгоритми машинного навчання, аналіз відгуків користувачів та контентну фільтрацію для персоналізації пропозицій. Їхні функціональні можливості значно різняться за ступенем інтеграції сервісів, обсягом оброблюваних даних і типами рекомендацій - від пошуку готелів до комплексного планування маршрутів.

Одним із найвідоміших прикладів є TripAdvisor, що пропонує рекомендації місць для відвідування, закладів харчування та готелів на основі системи рейтингів, відгуків і категоризації за тематиками (рис. 1.7). TripAdvisor використовує контентно-орієнтовану фільтрацію, алгоритми ранжування за популярністю та аналіз текстових описів для формування індивідуальних рекомендацій.

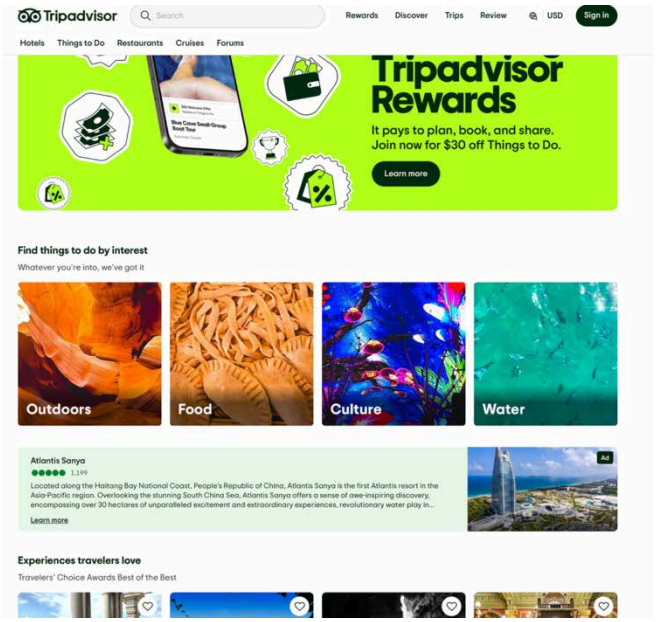

Рис. 1.7. Інтерфейс системи TripAdvisor для пошуку локацій та рекомендацій за інтересами користувача

Інша масштабна платформа - Expedia, яка об’єднує бронювання готелів, квитків, авто та екскурсій у єдиному середовищі (рис. 1.8). Система застосовує гібридну модель рекомендацій, поєднуючи історію переглядів користувача, пошукові фільтри й колаборативну фільтрацію для пропонування найбільш релевантних варіантів подорожей.

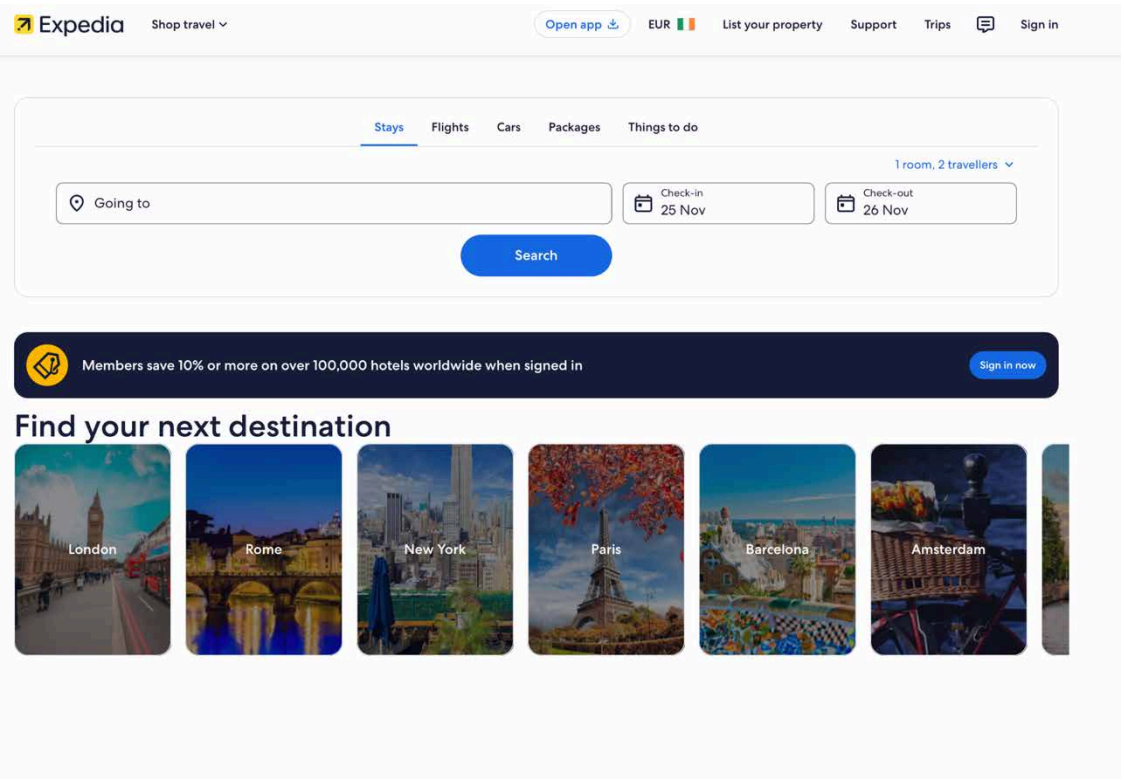

Рис. 1.8. Веб-інтерфейс Expedia із персоналізованими пропозиціями напрямків

Сервіс Booking.com спеціалізується на рекомендації житла, використовуючи дані про попередні бронювання, геолокацію та поведінку користувачів (рис. 1.9). Його рекомендаційна модель включає аналіз трендів попиту, ціноутворення та динамічне ранжування пропозицій на основі машинного навчання.

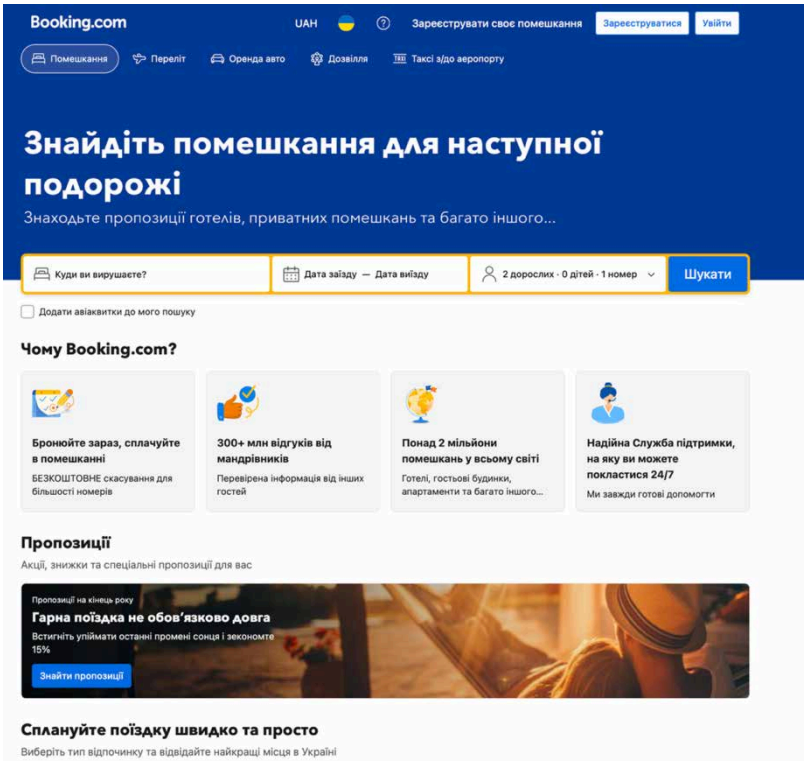

Рис. 1.9. Сторінка Booking.com із параметрами пошуку та фільтрами пропозицій проживання

Платформа Airbnb зосереджується на рекомендаціях приватного житла та унікальних локацій (рис. 1.10). Система поєднує контентний аналіз описів, відгуки користувачів та оцінювання рівня схожості об'єктів, створюючи рекомендації для схожих профілів мандрівників.

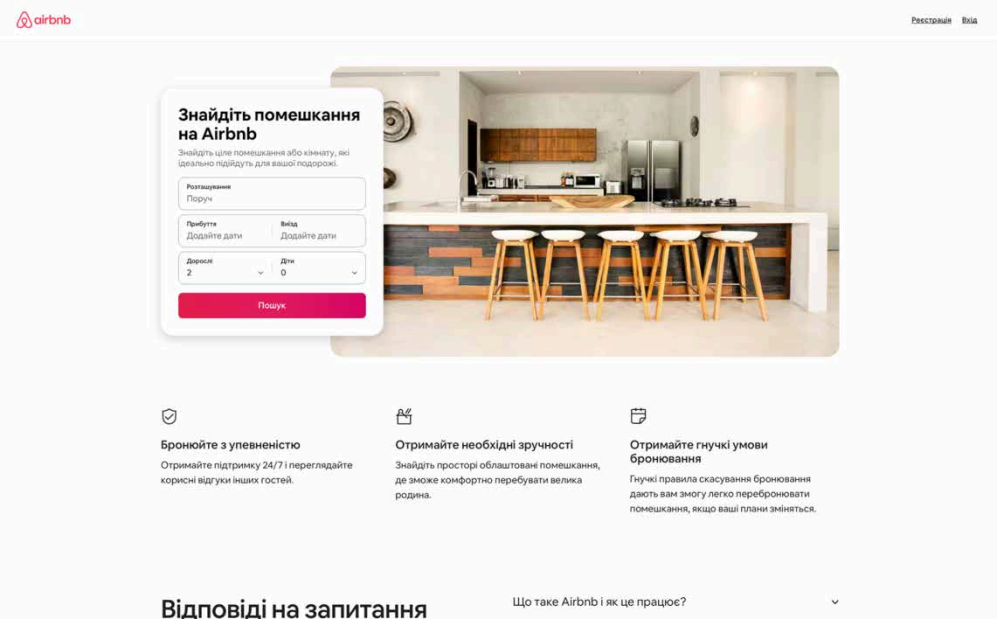

Рис. 1.10. Інтерфейс Airbnb із рекомендаціями житла та фільтрами за зручностями

Сервіс Google Travel використовує комплексну екосистему, що інтегрує карти, бронювання, транспорт і аналітику цін (рис. 1.11). Рекомендаційні алгоритми базуються на аналізі історії пошуків користувача, місця розташування, сезонності та агрегованих даних із Google Maps, Flights і Hotels.

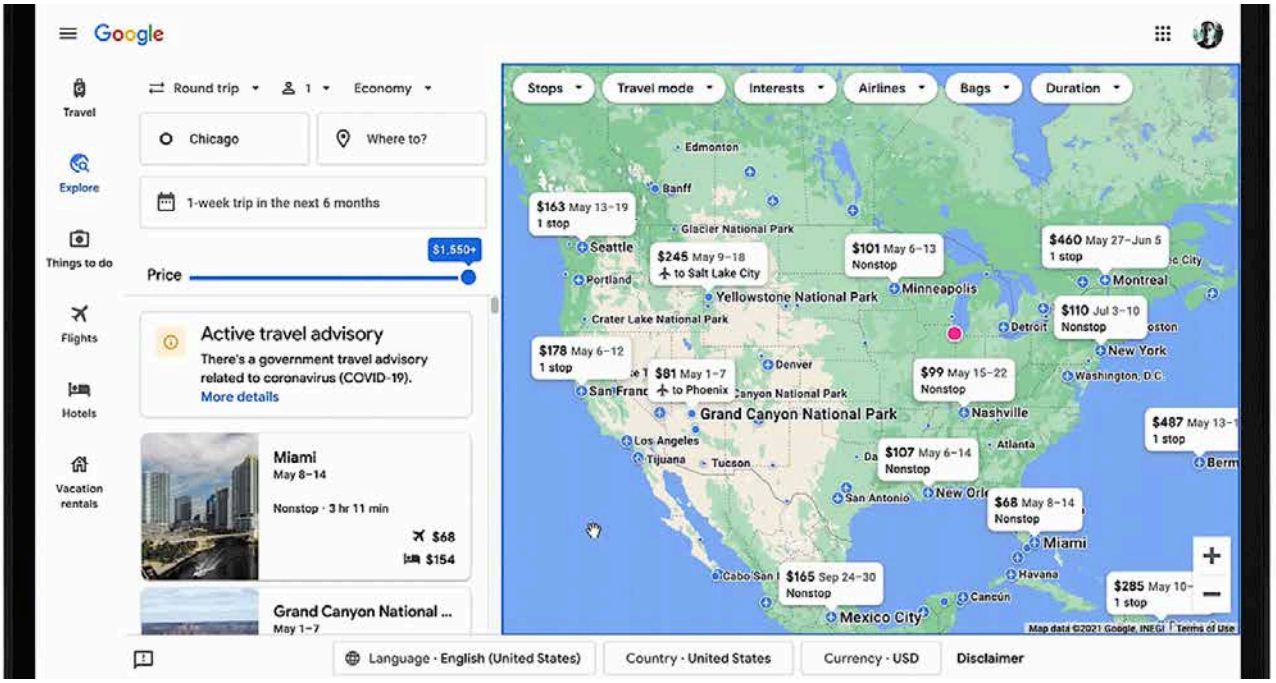

Рис. 1.11. Функціональний інтерфейс Google Travel із інтегрованою картою маршрутів та фільтрами вартості

Для узагальнення проведеного аналізу створено порівняльну таблицю 1.2, у якій наведено основні характеристики існуючих систем та розроблюваної рекомендаційної системи для планування подорожей.

Таблиця 1.2

Порівняння функціональних можливостей існуючих та розроблюваної системи

№	Система	Тип рекомендацій	Особливості реалізації	Обмеження
1	TripAdvisor	Контентно-оріє нтована	Аналіз відгуків, рейтингів, категорій інтересів	Відсутня інтеграція з бронюванням і транспортом
2	Expedia	Гібридна	Об’єднання бронювання готелів, квитків, авто	Менше уваги до персоналізації маршрутів

Продовження таблиці 1.2

3	Booking.com	Колаборативна	Моделі ML для оцінки попиту та ціноутворення	Обмежена контекстна адаптація
4	Airbnb	Контентна + соціальна	Рекомендації на основі відгуків і профілів користувачів	Не формує комплексні маршрути
5	Google Travel	Контекстна + поведінкова	Інтеграція карт, бронювання, аналітики	Залежність від екосистеми Google
6	Розроблювана система	Гібридна ML-архітектура (Graph + CF + Context)	Персоналізоване планування маршрутів із урахуванням контексту (бюджет, сезон, погода, геолокація), інтеграція проживання, подій і транспорту	Розширення бази даних і тестування на реальних користувачах

Узагальнюючи проведений аналіз, можна зазначити, що жодна з наявних платформ не забезпечує повної інтеграції всіх компонентів туристичної подорожі в єдиному інтелектуальному середовищі. Сучасні сервіси здебільшого

фокусуються на окремих аспектах - бронюванні, відгуках або пошуку локацій - без урахування комплексного контексту користувача (погодні умови, сезонність, бюджетні межі, індивідуальні інтереси).

Отже, наукова новизна запропонованої системи полягає у створенні єдиної гібридної рекомендаційної моделі, яка об’єднує алгоритми нейронної та графової колаборативної фільтрації з контекстним модулем адаптації, що забезпечує персоналізоване планування подорожей у реальному часі з урахуванням усіх доменних параметрів.

1.4 Моделювання предметної області

Моделювання предметної області системи рекомендацій подорожей базується на застосуванні формальних методів UML, що забезпечують структуроване представлення взаємодії користувачів, сервісів і зовнішніх компонентів системи. Такий підхід дозволяє визначити логіку функціонування програмного комплексу, інформаційні потоки між підсистемами та взаємозв’язки між об’єктами, які реалізують функціональні вимоги системи.

На рис. 1.12 подано діаграму прецедентів, яка демонструє основних акторів системи - користувача (мандрувальника), адміністратора, постачальників даних, платіжний сервіс і службу автентифікації. Вона відображає типові сценарії взаємодії, зокрема авторизацію користувача, керування профілем вподобань, отримання рекомендацій, створення і збереження маршруту, бронювання послуг проживання та транспорту, а також формування зворотного зв’язку.

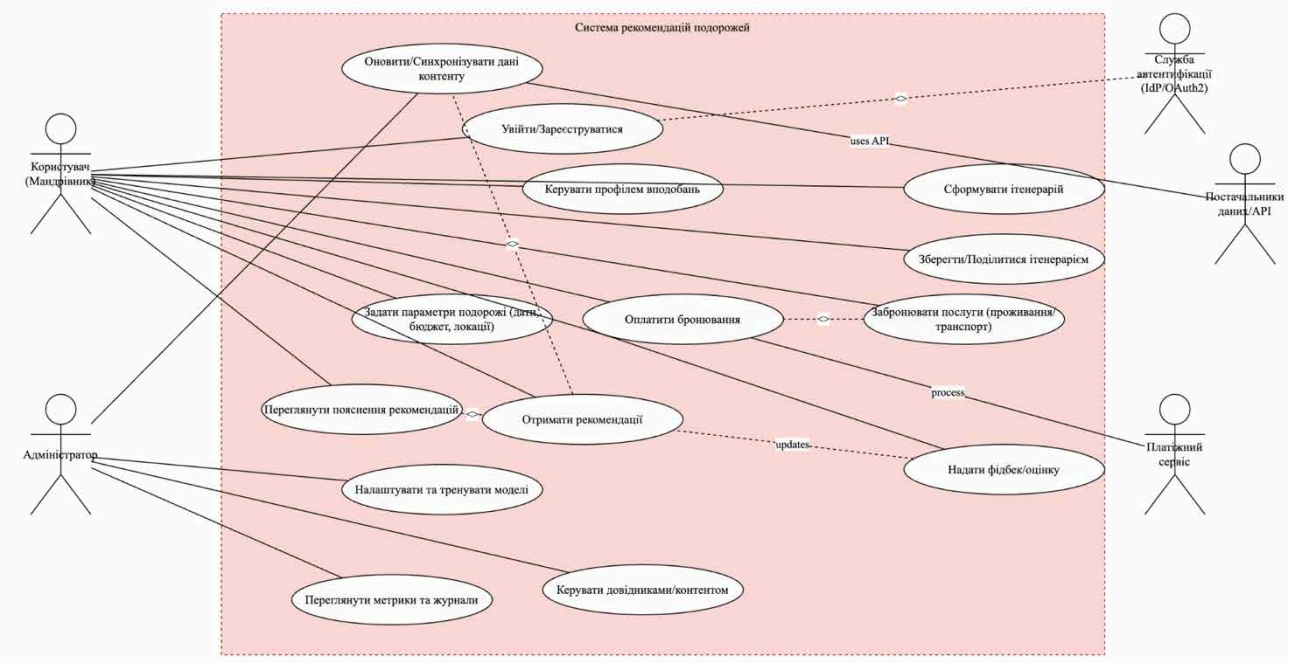

Рис. 1.12. Діаграма прецедентів системи рекомендацій подорожей

Взаємодія між компонентами системи, починаючи з моменту запиту користувача й до підтвердження бронювання, представлена на діаграмі послідовності (рис. 1.13). Користувач через веб-додаток ініціює автентифікацію через службу IdP/OAuth2, після чого сервіс рекомендацій надсилає запит до зовнішніх постачальників даних (API контенту: локації, готелі, транспорт). На основі отриманих даних виконується машинне ранжування маршрутів, формуються пояснення за допомогою алгоритму SHAP, і користувач отримує персоналізований список рекомендацій. У разі вибору варіанту здійснюється інтеграція з сервісом ітенераріїв, бронюванням і платіжним шлюзом.

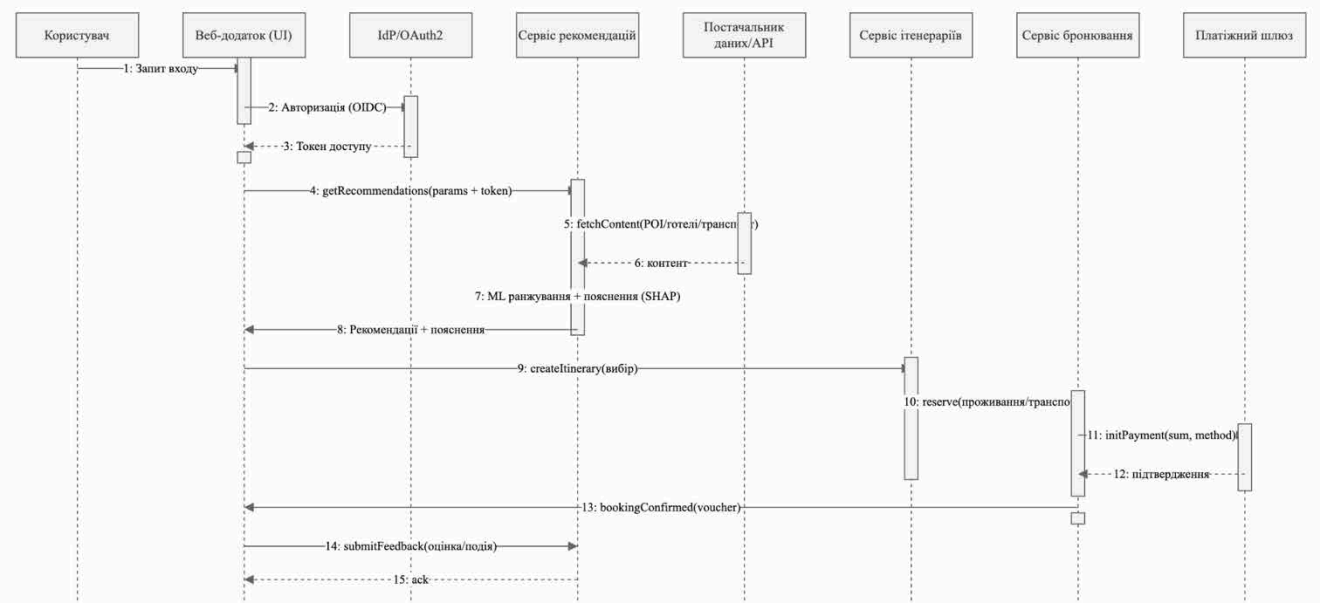

Рис. 1.13. Діаграма послідовності процесу формування та підтвердження подорожі

На діаграмі активності (рис. 1.14) відображено бізнес-процес системи, який включає етапи введення параметрів подорожі, перевірку автентифікації користувача, формування запиту рекомендацій, обчислення ранжованого списку, уточнення фільтрів, вибір маршруту, оплату й формування зворотного зв’язку. Модель охоплює взаємодію між підсистемами: інтерфейсом користувача (UI), сервісом рекомендацій, модулями бронювання та зовнішніми API, що забезпечують узгоджене функціонування компонентів.

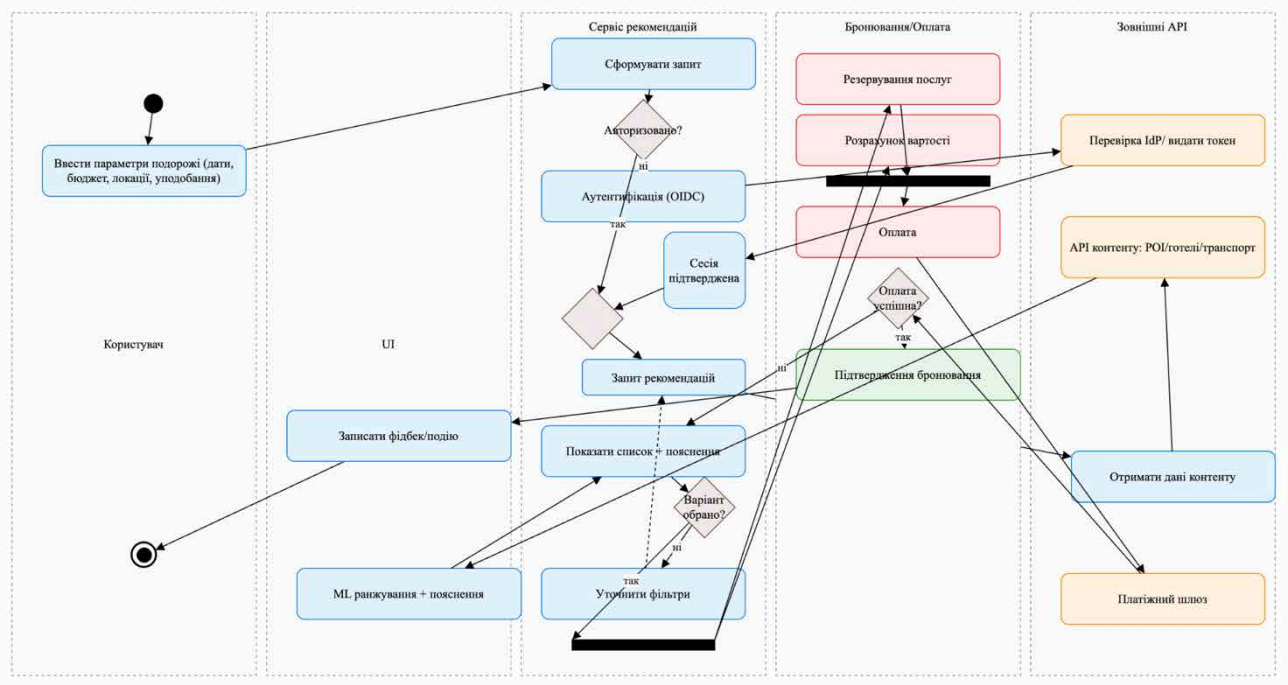

Рис. 1.14. Діаграма активності бізнес-процесу системи рекомендацій подорожей

Моделювання предметної області дозволило визначити основні сценарії взаємодії користувача із системою, логіку передачі даних між підсистемами та точки інтеграції з зовнішніми сервісами. Отримані UML-моделі забезпечують формалізоване уявлення про роботу системи, що є підґрунтям для розроблення програмної архітектури й подальшої реалізації модулів машинного навчання, ітенераріїв та бронювання.

Результуючий висновок: проведене моделювання підтвердило доцільність використання багаторівневої архітектури системи, у якій рекомендаційний модуль виступає ядром, що координує взаємодію між користувацьким інтерфейсом, сервісами ідентифікації, бронюванням і зовнішніми API. Така структурна модель забезпечує масштабованість, модульність і можливість подальшого розширення системи, зокрема шляхом інтеграції нових алгоритмів машинного навчання та джерел даних.

1.5 Аналіз вимог до системи рекомендацій подорожей

Розроблення рекомендаційної системи для планування подорожей на основі машинного навчання вимагає чіткої формалізації вимог, що визначають функціональні можливості, нефункціональні характеристики та вимоги до безпеки. Аналіз вимог проводиться з урахуванням результатів попереднього моделювання, особливостей предметної області та сучасних тенденцій у створенні інтелектуальних інформаційних систем.

Функціональні вимоги описують основні можливості системи, пов’язані з процесами реєстрації користувача, формування рекомендацій, планування маршрутів, бронювання та отримання зворотного зв’язку. Їх наведено у таблиці 1.3.

Таблиця 1.3

Функціональні вимоги до системи рекомендацій подорожей

№	Вимога	Опис
1	Реєстрація та автентифікація користувача	Підтримка входу через OAuth2/OpenID Connect для захищеного доступу до профілю вподобань і даних користувача.
2	Керування профілем вподобань	Збереження інформації про уподобання користувача (жанри, тип активності, бюджет, комфорт, ризик).
3	Формування та відображення рекомендацій	Побудова персоналізованих списків маршрутів, локацій і подій на основі ML-моделей та контекстних факторів.
4	Генерація ітенераріїв	Автоматичне формування послідовності кроків подорожі з урахуванням обмежень і контентних даних.
5	Бронювання послуг	Інтеграція з постачальниками даних (API готелів, транспорту, турів) та виконання резервації через платіжний шлюз.
6	Відображення пояснень рекомендацій	Пояснення результатів ранжування засобами методу SHAP для підвищення прозорості алгоритмів.
7	Надання зворотного зв’язку	Збір оцінок, рецензій і фідбеку для покращення точності майбутніх рекомендацій.

Продовження таблиці 1.3

8	Адміністрування системи	Перегляд метрик, логів і керування контентом адміністратором через окремий інтерфейс.

Нефункціональні вимоги встановлюють показники продуктивності, масштабованості, сумісності та надійності системи. Вони формують основу для забезпечення якості програмного продукту та ефективності роботи системи в умовах великої кількості користувачів. Ці характеристики наведено у таблиці 1.4.

Таблиця 1.4

Нефункціональні вимоги до системи рекомендацій подорожей

№	Параметр	Опис та обмеження
1	Продуктивність	Час відгуку на запит рекомендацій ≤ 2 секунди; оновлення моделі в асинхронному режимі.

2	Масштабованість	Підтримка паралельної роботи ≥ 1000 користувачів через кластеризацію сервісів і контейнеризацію.
3	Надійність та відмовостійкість	Рівень готовності системи ≥ 99,9 %, автоматичне відновлення контейнерів через оркестратор (Docker/Kubernetes).
4	Сумісність	Підтримка REST/GraphQL API для інтеграції з зовнішніми платформами й мобільними застосунками.
5	Модульність	Чітке розділення функціональних модулів (рекомендації, бронювання, платежі) для спрощення супроводу.
6	Зручність користування	Інтуїтивний інтерфейс UI з адаптивним дизайном для настільних і мобільних пристроїв.

Вимоги до безпеки визначають механізми захисту персональних даних, фінансових транзакцій і взаємодії з зовнішніми API. Їх подано у таблиці 1.5.

Таблиця 1.5

Вимоги до забезпечення інформаційної безпеки системи

№	Вимога	Опис механізму захисту
1	Автентифікація та авторизація	Застосування OAuth 2.0 та OpenID Connect для керування сесіями та правами доступу.

Продовження таблиці 1.3

2	Захист транзакцій	Використання TLS 1.3 та шифрування AES-256 для захисту даних під час передачі та платежів.
3	Контроль доступу	Рольова модель RBAC/ABAC для розмежування доступу адміністратора та користувача.
4	Захист даних користувача	Шифрування персональних даних у базі даних та використання анонімізації для аналітики.
5	Аудит та журнали	Збереження журналів входів, бронювань і платежів з контролем інцидентів безпеки.

Результуючий висновок: проведений аналіз вимог дозволив чітко окреслити функціональні межі, очікувані показники продуктивності та безпекові механізми майбутньої системи. Запропонована архітектура передбачає інтеграцію модулів збору даних, машинного навчання, рекомендацій, бронювання та платежів у єдиному безпечному середовищі. Такий підхід

забезпечує високу адаптивність системи, захист персональних даних і надійність функціонування в реальному часі, що є основою її практичного впровадження у сфері інтелектуального туризму.

1.6 Постановка завдання

Постановка завдання для розроблення системи рекомендацій подорожей на основі машинного навчання передбачає визначення цілей функціонування, структури даних і алгоритмів оброблення інформації, необхідних для формування персоналізованих маршрутів. Основна ідея полягає у створенні адаптивної системи, здатної автоматично аналізувати профіль користувача, його уподобання, історію переглядів і контекстні фактори (бюджет, сезонність, геолокацію), після чого формувати ранжований список локацій, подій і засобів транспорту для конкретного періоду подорожі. Система має забезпечити інтеграцію з зовнішніми джерелами даних (API туристичних платформ, погодних і подієвих сервісів), а також підтримувати автоматизоване бронювання послуг через платіжний шлюз із підтвердженням транзакцій у режимі реального часу.

У контексті архітектури розроблювана система є багаторівневою — включає рівень збору даних, рівень аналітики та рекомендацій, рівень користувацької взаємодії (UI) і рівень інтеграції з сервісами бронювання. Кожен модуль функціонує незалежно в межах мікросервісної архітектури, взаємодіючи через REST/GraphQL API, що забезпечує масштабованість і відмовостійкість системи. Алгоритмічна частина реалізується на основі гібридної моделі машинного навчання, яка поєднує контентну фільтрацію, колаборативні методи та графові нейронні мережі для виявлення прихованих закономірностей між користувачами та туристичними об’єктами.

Вхідними даними для системи є інформація про користувача (ідентифікатор, демографічні параметри, історія переглядів, рейтинги, оцінки, вподобання), контекстні характеристики подорожі (дати, бюджет, місце

відправлення, цільові локації, погодні умови, сезонність) і метадані зовнішніх джерел (описи готелів, подій, транспортних маршрутів, популярність об’єктів, ціни). Додатково до вхідних параметрів використовуються агреговані поведінкові дані для навчання моделей і побудови латентних векторних представлень користувачів та об’єктів.

Вихідними даними системи є персоналізований набір рекомендацій, що містить ранжований список маршрутів, готелів, туристичних локацій і подій із зазначенням рейтингу релевантності, вартості та доступності, а також сформований ітенерарій подорожі, який користувач може редагувати, зберігати або ділитися. Додатковим результатом є пояснення рекомендацій, сформоване за допомогою методу SHAP, що підвищує довіру користувача до системи. У разі вибору конкретного маршруту система ініціює процес бронювання проживання й транспорту, підтверджує оплату через інтегрований платіжний шлюз і зберігає деталі транзакції у базі даних для подальшої аналітики.

Отже, сформульоване завдання передбачає побудову інтелектуальної системи, здатної в режимі реального часу обробляти різнорідні джерела даних, прогнозувати інтереси користувача та формувати оптимальний маршрут подорожі з урахуванням індивідуальних і контекстних факторів. Це забезпечує персоналізацію туристичних сервісів нового покоління, орієнтованих на автоматичне планування, бронювання та аналітичну підтримку користувача в повному циклі подорожі.

1.7 Висновки до першого розділу

У першому розділі проведено системний аналіз предметної області рекомендаційних систем для планування подорожей, який дав змогу визначити наукові та технічні передумови створення інтелектуальної інформаційної системи, орієнтованої на персоналізоване формування туристичних маршрутів. Встановлено, що сучасні рішення, такі як TripAdvisor, Expedia, Booking.com, Google Travel та Airbnb, забезпечують переважно базову персоналізацію, однак

не враховують динамічні контекстні фактори - сезонність, ризик, погодні умови, активність користувача та інтеграцію із зовнішніми джерелами даних у режимі реального часу. Це формує потребу у створенні системи нового покоління з використанням технологій машинного навчання, графових нейронних мереж і пояснюваної аналітики.

Теоретико-методологічний аналіз засвідчив, що сучасні наукові дослідження у сфері рекомендаційних систем (Li et al., 2024; Zhao et al., 2023; Ricci et al., 2022) концентруються на побудові гібридних моделей, здатних поєднувати переваги контентної, колаборативної та контекстно-орієнтованої фільтрації. Розглянуті архітектурні схеми - багатошарова нейронна мережа, модель нейронної колаборативної фільтрації, графова нейромережа та автоенкодер - стали основою для визначення математичного апарату майбутньої системи.

У ході моделювання побудовано UML-діаграми (прецедентів, послідовності, активності, компонентів), які формалізують основні сценарії взаємодії користувача із системою, взаємозв’язки між сервісами рекомендацій, бронювання, платіжним шлюзом і зовнішніми API. Аналіз вимог дав змогу структурувати функціональні, нефункціональні та безпекові параметри, необхідні для реалізації програмного комплексу, включно з продуктивністю, масштабованістю, надійністю та захистом персональних даних відповідно до сучасних стандартів безпеки.

Постановка завдання завершила концептуальне обґрунтування дослідження: визначено вхідні та вихідні дані, інформаційні потоки, архітектурні взаємозв’язки й роль машинного навчання у процесі формування рекомендацій. Таким чином, результати першого розділу створюють теоретичну і методичну основу для розроблення програмної архітектури, алгоритмізації модулів та реалізації інтелектуальної системи персоналізованого планування подорожей у наступних розділах.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних системи рекомендацій подорожей розроблена з метою формалізації інформаційної структури та взаємозв’язків між основними сутностями, що забезпечують функціонування системи. Вона визначає, які об’єкти зберігаються у базі даних, як вони пов’язані між собою, та які атрибути формують інформаційну основу процесів рекомендації, бронювання та аналітики.

На рис. 2.1 представлено ER-діаграму логічної моделі даних, побудовану за допомогою UML-нотації. Вона охоплює ключові сутності системи: «Користувач», «Профіль вподобань», «Контекст подорожі», «Рекомендація», «Маршрут», «Об’єкт/Подія», «Бронювання» та «Фідбек». Між ними встановлено зв’язки, що відображають логіку інформаційних процесів: користувач створює профіль і контекст, отримує рекомендації, генерує маршрути, бронює послуги та залишає зворотний зв’язок.

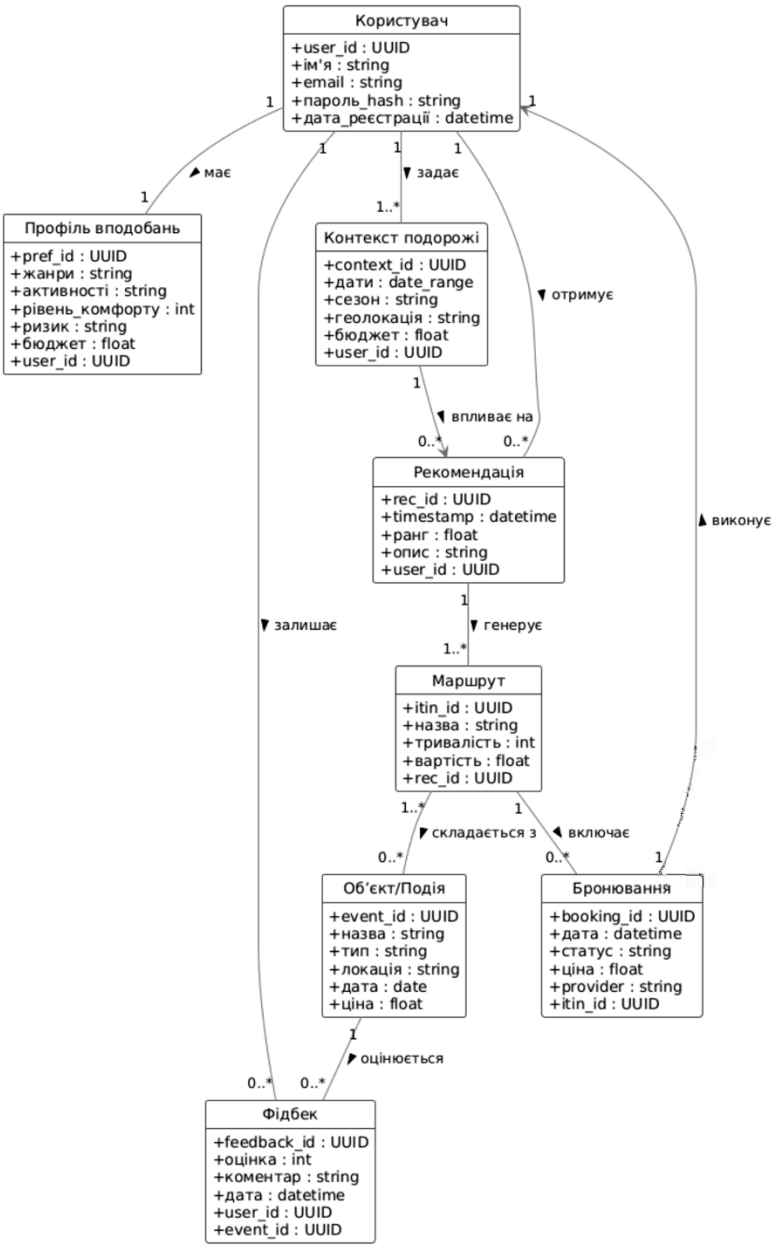

Рис. 2.1. Логічна модель даних системи рекомендацій подорожей

Основна увага під час проектування моделі приділялась нормалізації даних, що забезпечує цілісність і відсутність надлишковості. Структура моделі відповідає третій нормальній формі (3НФ), де кожна таблиця містить атрибути, що функціонально залежать від первинного ключа, а всі зв'язки між сутностями реалізуються через зовнішні ключі. Такий підхід мінімізує дублювання інформації, спрощує оновлення записів і підвищує ефективність виконання запитів у базі даних.

У таблиці 2.1 наведено узагальнену характеристику основних сутностей, які формують ядро інформаційної моделі системи.

Таблиця 2.1

Основні сутності логічної моделі даних системи

Назва сутності	Ключові атрибути	Призначення
Користувач	user_id, ім’я, email, дата_реєстрації	Ідентифікація користувачів та керування доступом
Профіль вподобань	pref_id, жанри, активності, бюджет, user_id	Зберігання параметрів інтересів користувача
Контекст подорожі	context_id, дати, сезон, геолокація	Визначення умов подорожі для формування рекомендацій
Рекомендація	rec_id, timestamp, опис, ранг, user_id	Формування ранжованих списків об’єктів і маршрутів
Маршрут	itin_id, назва, тривалість, вартість, rec_id	Визначення структури подорожі користувача
Об’єкт/Подія	event_id, назва, локація, тип, дата	Представлення атракцій, турів або івентів у маршруті
Бронювання	booking_id, дата, статус, ціна, itin_id	Реєстрація підтверджених замовлень і транзакцій
Фідбек	feedback_id, оцінка, коментар, user_id, event_id	Аналіз задоволеності користувача та поліпшення моделей ML

У контексті побудови бази даних така логічна модель є проміжним рівнем між концептуальним описом і фізичною реалізацією, що визначає структуру таблиць, первинні та зовнішні ключі, а також типи зв’язків «один-до-багатьох» і «багато-до-багатьох». Завдяки формалізованим відношенням між сутностями забезпечується коректність виконання операцій CRUD (Create, Read, Update, Delete), а також підтримується узгодженість даних при інтеграції з модулем машинного навчання та аналітичними підсистемами.

Результуючий висновок: логічна модель даних системи рекомендацій подорожей є фундаментом для побудови фізичної структури бази даних і забезпечує узгоджене представлення усіх процесів збору, оброблення та аналізу даних. Завдяки нормалізації, чіткому визначенню ключових зв’язків та використанню UML-нотації модель гарантує масштабованість і гнучкість архітектури, необхідну для реалізації високонавантаженої рекомендаційної системи з елементами машинного навчання та інтелектуальної аналітики.

2.2 Діаграма класів і кооперації

Проєктування програмної архітектури системи рекомендацій подорожей передбачає побудову діаграми класів, яка формалізує структуру об’єктно-орієнтованої моделі та відображає ключові сутності, їхні атрибути, методи й асоціації. Основна мета цієї діаграми полягає у відображенні логіки взаємодії між компонентами системи, забезпеченні модульності, повторного використання коду та високого рівня абстракції для подальшої реалізації. Модель побудована з урахуванням принципів інкапсуляції, наслідування й композиції, що гарантує узгодженість функціональних зв’язків та оптимізує структуру коду.

На рис. 2.2 наведено діаграму класів, що представляє основні елементи системи: користувача, профіль уподобань, запит подорожі, рекомендації, маршрут, елементи маршруту та їхні типи. Кожен клас містить набір атрибутів і методів, які відповідають за оброблення даних, генерацію запитів, ранжування результатів, формування ітенераріїв та валідацію введених параметрів. Особливістю моделі є наявність узагальнень та асоціацій типу «один до багатьох», що забезпечує природну ієрархічну структуру системи.

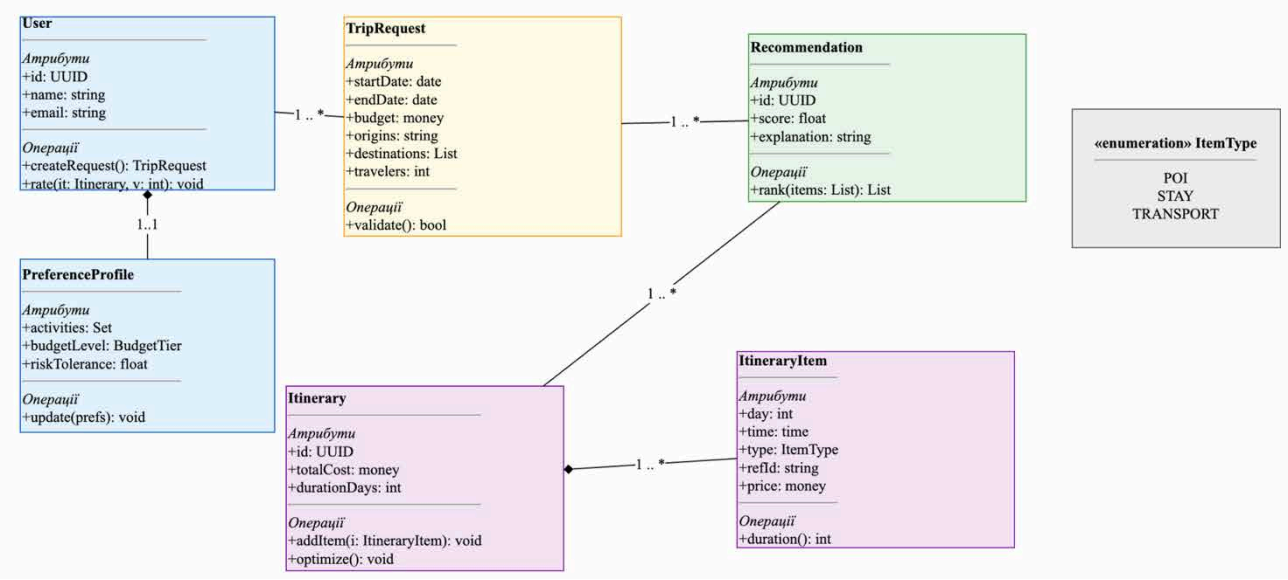

Рис. 2.2. Діаграма класів системи рекомендацій подорожей

Побудована модель підтримує принципи нормалізації об’єктної структури, що запобігає дублюванню логіки та атрибутів між класами. Кожен клас має чітко визначені обов’язки: клас User відповідає за автентифікацію та створення запитів, TripRequest зберігає параметри подорожі, Recommendation формує список об’єктів із поясненнями на основі моделі машинного навчання, Itinerary оптимізує маршрути, а ItineraryItem представляє конкретні елементи маршруту (транспорт, проживання, події). Така структура дає змогу досягнути високого рівня когерентності всередині модулів і слабкої зв’язаності між ними, що відповідає архітектурним вимогам до масштабованих інтелектуальних систем.

Подальша деталізація процесів відображена на діаграмах кооперацій, які демонструють динаміку обміну повідомленнями між об’єктами в типових сценаріях роботи системи. На рис. 2.3 представлено кооперацію користувача з веб-додатком і сервісом рекомендацій, де ілюструється послідовність запиту параметрів подорожі, отримання рекомендацій та передачі результатів від моделі ранжування до інтерфейсу користувача. Така взаємодія демонструє ключовий цикл комунікації — від збору вхідних даних до побудови топ-N рекомендацій.

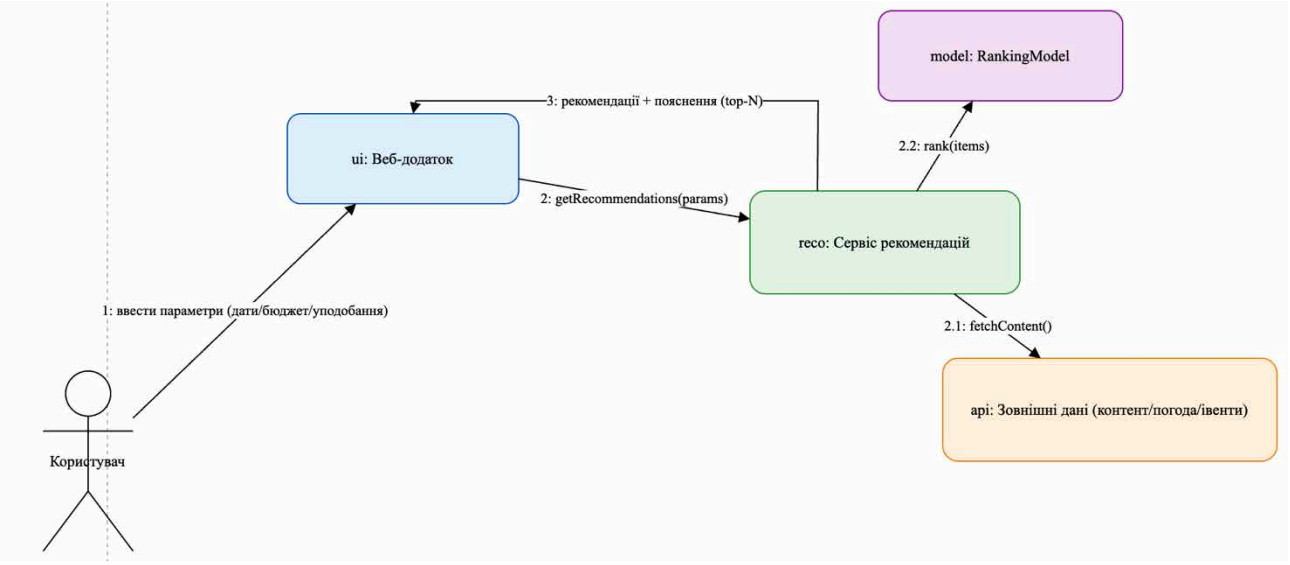

Рис. 2.3. Діаграма кооперації процесу отримання рекомендацій

На рис. 2.4 наведено кооперацію, що відображає сценарій створення маршруту та процес бронювання послуг. Користувач через інтерфейс викликає сервіс ітенерарііів, який взаємодіє із сервісом бронювання, постачальниками послуг і платіжним шлюзом для отримання актуальних цін, резервування контингенту та підтвердження платежів. Завдяки чіткому розділенню обов'язків між компонентами система досягає асинхронності та підвищеної відмовостійкості.

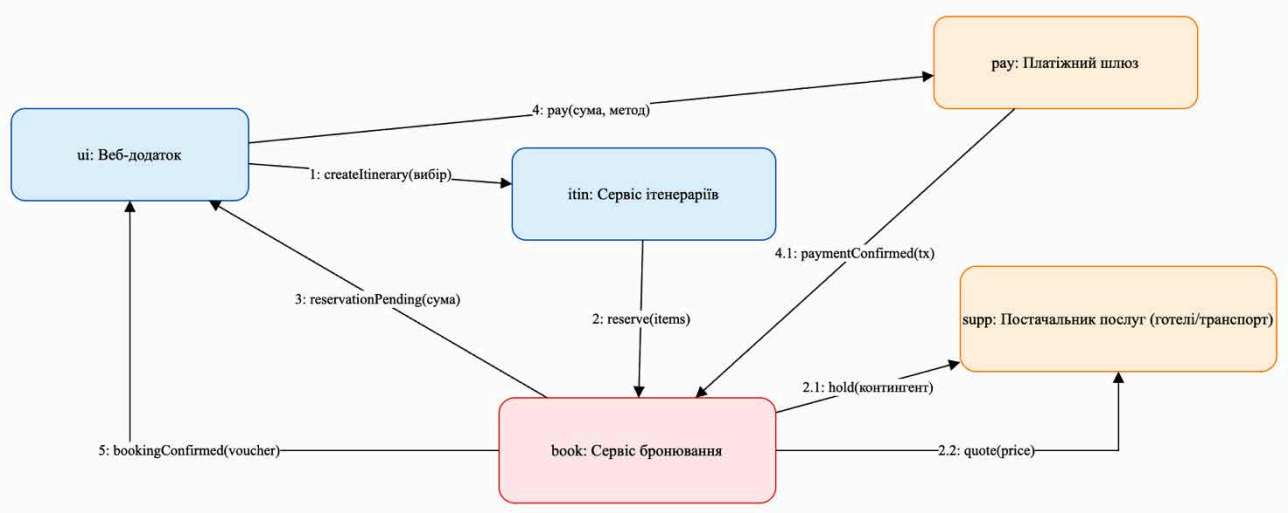

Рис. 2.4. Діаграма кооперації процесу бронювання маршруту

Рисунок 2.5 ілюструє кооперацію навчання моделей машинного навчання на основі зібраного фідбеку користувачів. Сервіс рекомендацій передає події до сховища, після чого вони завантажуються тренувальним модулем для побудови нових моделей, які зберігаються у реєстрі. Такий цикл реалізує концепцію безперервного навчання (Continuous Learning Loop), що забезпечує адаптивність системи до змін користувацьких уподобань.

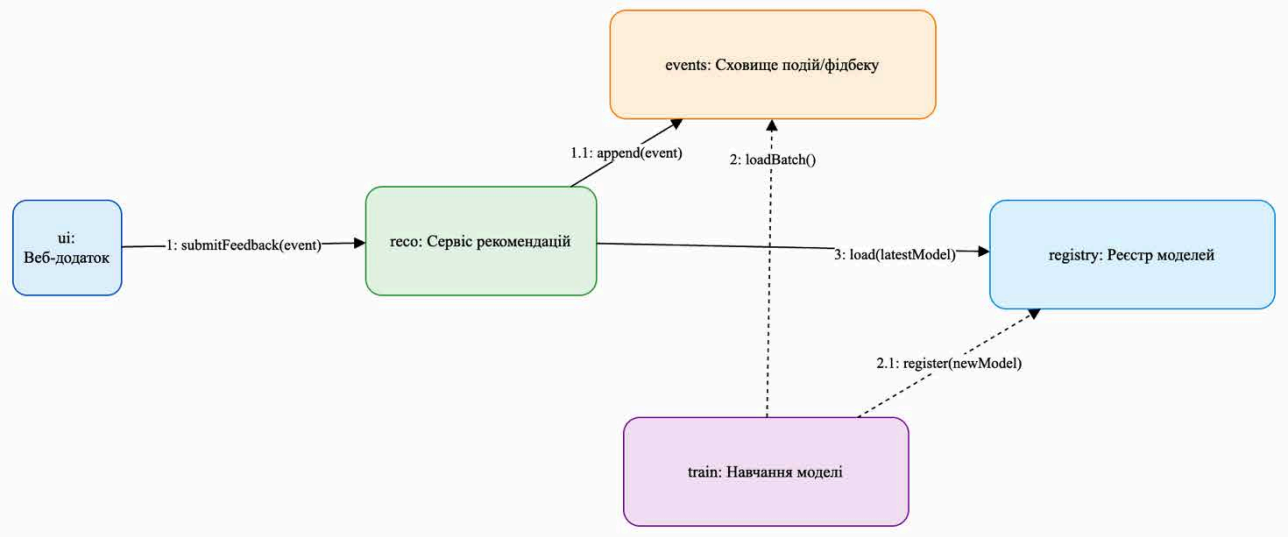

Рис. 2.5. Діаграма кооперації процесу навчання моделі машинного навчання

У таблиці 2.2 наведено узагальнені характеристики основних класів системи з описом їхньої ролі та функціонального призначення.

Таблиця 2.2

Основні класи системи рекомендацій подорожей

Клас	Призначення	Ключові методи
User	Зберігає дані користувача, ініціює запити та оцінює маршрути	createRequest(), rate()
TripRequest	Містить параметри подорожі та виконує валідацію	validate()
Recommendation	Виконує ранжування об’єктів і генерує пояснення	rank()
Itinerary	Формує і оптимізує маршрут	addItem(), optimize()
ItineraryItem	Описує конкретні елементи маршруту	duration()
PreferenceProfile	Зберігає вподобання користувача	update()

Діаграма класів і відповідні кооперації забезпечують цілісне бачення архітектури об’єктно-орієнтованої системи рекомендацій подорожей. Вони описують не лише структуру даних, а й механізм їхньої взаємодії у динаміці, що дає змогу реалізувати адаптивний механізм персоналізації на основі машинного навчання. Результуючим підсумком є формування гнучкої архітектурної моделі,

яка поєднує модульність, узгодженість, повторне використання компонентів і можливість подальшого масштабування системи.

2.3 Діаграма компонентів

Архітектура системи рекомендацій подорожей побудована на основі принципів мікросервісної взаємодії та розподіленої обробки даних, що забезпечує масштабованість, модульність і стійкість до збоїв. Діаграма компонентів, подана на рис. 2.6, формалізує логічну структуру програмного комплексу, визначаючи основні модулі, інтерфейси зв'язку та напрямки обміну інформацією. Така модель дозволяє реалізувати гнучку інтеграцію з зовнішніми сервісами, підтримувати незалежне оновлення модулів і забезпечує відповідність архітектурним принципам RESTful-та Service-Oriented-Architecture (SOA).

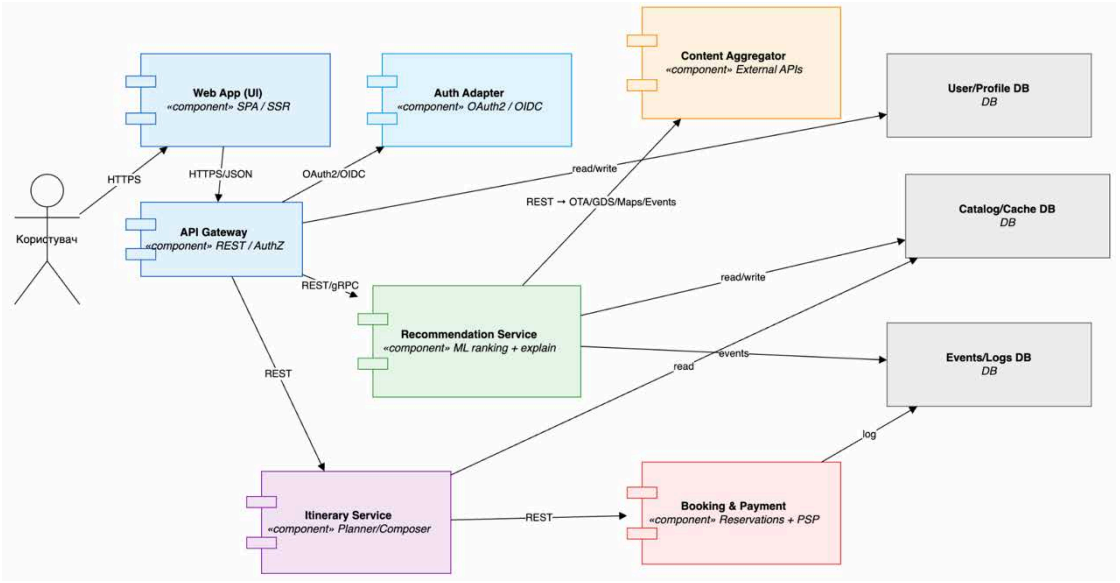

Рис. 2.6. Діаграма компонентів системи рекомендацій подорожей

На рівні клієнтської взаємодії основним елементом виступає Web App (UI), що реалізує єдиний інтерфейс користувача та підтримує роботу у форматах SPA/SSR. Компонент забезпечує обмін даними через API Gateway, який виконує роль проміжного шару автентифікації (AuthZ/AuthN), маршрутизації запитів і балансування навантаження. Компонент Auth

Adapter інтегрує систему з протоколами OAuth2/OIDC, забезпечуючи захищений доступ користувачів до персоналізованих даних.

Ключовим ядром бізнес-логіки є Recommendation Service, який реалізує моделі машинного навчання для персоналізованого ранжування контенту та формування пояснень до рекомендацій. Він взаємодіє з Content Aggregator, що акумулює дані з зовнішніх джерел (OTA, GDS, карти, події, метеосервіси) через REST-інтерфейси. Компоненти Itinerary Service та Booking & Payment відповідають відповідно за побудову оптимального маршруту користувача та управління процесом бронювання і платежів через інтеграцію з постачальниками послуг.

Таблиця 2.3

Основні компоненти системи та їх функціональне призначення

Компонент	Призначення	Тип взаємодії
Web App (UI)	Формування користувацького інтерфейсу, взаємодія з API	HTTPS / JSON
API Gateway	Авторизація, маршрутизація, REST-агрегація запитів	REST / gRPC
Auth Adapter	Інтеграція з OAuth2 / OIDC-провайдерами	OIDC / Token Flow
Recommendation Service	Ранжування контенту ML-алгоритмами, генерація пояснень	REST / ML Model
Content Aggregator	Отримання зовнішніх даних (події, карти, OTA, GDS)	REST / External APIs
Itinerary Service	Планування маршруту, побудова ітенерарііів	REST
Booking & Payment	Обробка бронювань, оплати, підтвердження транзакцій	REST / PSP API

Зберігання та оброблення даних реалізовані через низку спеціалізованих баз: User/Profile DB (для користувацьких профілів і вподобань), Catalog/Cache DB (для кешування актуального контенту), Events/Logs DB (для журналів подій

і телеметрії). Такий розподіл забезпечує незалежність потоків даних і підтримує горизонтальне масштабування при зростанні навантаження.

Важливою особливістю архітектури є використання асинхронної обробки подій, що дозволяє зменшити час відгуку сервісів і підвищити ефективність роботи рекомендаційного ядра. Компоненти системи обмінюються даними через REST-інтерфейси з підтримкою TLS-шифрування, а внутрішня комунікація між мікросервісами реалізована з використанням протоколів gRPC та черг подій. Такий підхід забезпечує низьку затримку, fault-tolerance і можливість незалежного масштабування окремих компонентів.

Результуючий висновок: представлена компонентна архітектура формує технологічну основу для реалізації інтелектуальної системи рекомендацій подорожей, де кожен модуль має чітко визначену зону відповідальності, стандартизовані інтерфейси обміну та підтримує розширення без впливу на інші частини системи. Це дозволяє досягнути високої гнучкості, безпеки й продуктивності, що є критично важливим для сучасних адаптивних веб-платформ із елементами машинного навчання.

2.4 Діаграма пакетів

Архітектурна структура програмного забезпечення системи рекомендацій подорожей передбачає багаторівневу організацію коду у вигляді логічно ізольованих пакетів, що забезпечують чітке розділення відповідальності, зручність супроводу та незалежність модулів при оновленнях. Такий підхід відповідає принципам Domain-Driven Design (DDD) і Clean Architecture, де внутрішні рівні системи не залежать від зовнішніх реалізацій, а взаємодія між ними здійснюється через стандартизовані інтерфейси.

На рис. 2.7 наведено діаграму пакетів, яка відображає основні структурні рівні системи: від рівня представлення до рівня аналітики машинного навчання та зовнішніх інтеграцій. Модель побудована так, щоб забезпечити інверсію залежностей - зовнішні пакети не впливають на ядро системи, а логіка

бізнес-домену залишається ізольованою від інфраструктурних реалізацій. Це гарантує стабільність архітектури та спрощує масштабування системи при збільшенні навантаження або додаванні нових функціональних модулів.

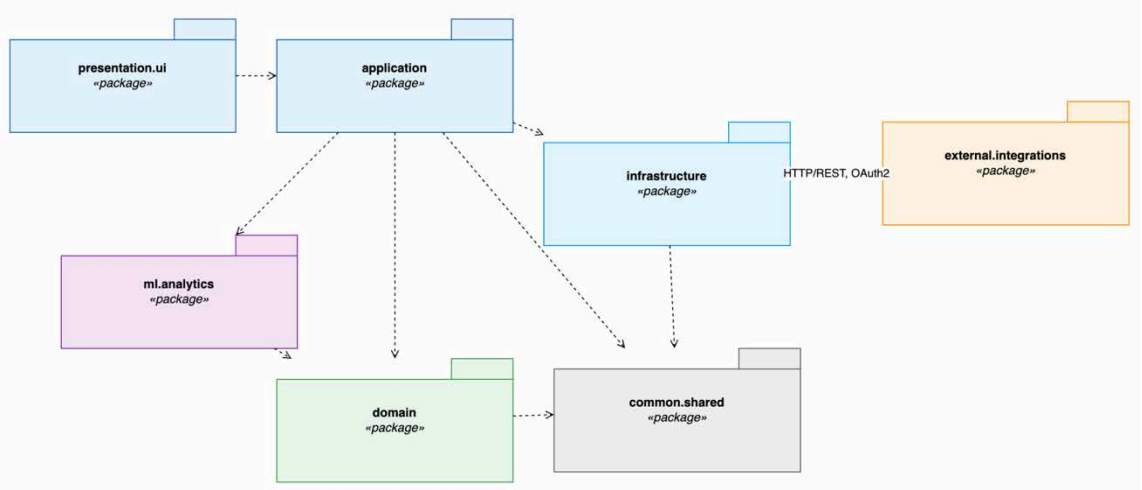

Рис. 2.7. Діаграма пакетів системи рекомендацій подорожей

Основні пакети узгоджені між собою через чітко визначені точки взаємодії (HTTP, REST, OAuth2) і спільні абстракції. Завдяки цьому реалізується гнучкий механізм комунікації між підсистемами, включно з модулями машинного навчання, аналітики, зберігання даних і користувацького інтерфейсу. Така організація сприяє високому рівню модульності, полегшує тестування й підтримку, а також забезпечує можливість розподіленої розробки.

Таблиця 2.4

Основні пакети системи та їх роль у архітектурі

Пакет	Рівень	Основне призначення
presentation.ui	Рівень представлення	Забезпечує інтерфейс взаємодії користувача з системою
application	Рівень бізнес-логіки	Реалізує прикладну логіку, сценарії обробки запитів
infrastructure	Рівень інтеграції	Забезпечує взаємодію з БД, мережевими сервісами та API
ml.analytics	Рівень інтелектуальної обробки	Відповідає за алгоритми машинного навчання, ранжування та аналітику
domain	Рівень предметної області	Містить сутності та правила, що визначають логіку рекомендацій

common.shared	Допоміжний рівень	Містить спільні утиліти, константи та базові структури
external.integrations	Зовнішній рівень	Реалізує підключення до сторонніх сервісів (OTA, GDS, карти, події)

Взаємозв’язки між пакетами демонструють послідовний потік даних: від користувацького запиту до аналітичного модуля машинного навчання та зовнішніх інтеграцій. Комунікація реалізується через REST-інтерфейси з автентифікацією OAuth2, що забезпечує безпечний і стандартизований обмін даними. Структурна ізоляція рівнів дозволяє незалежно розвивати частини системи без порушення її цілісності, а використання єдиного узгодженого підходу до керування залежностями гарантує стабільність архітектури навіть при розширенні функціональності.

Результуючий висновок: представлена діаграма пакетів формує системне бачення програмної архітектури, у якій кожен рівень має чітке призначення й мінімальні залежності. Така структура дозволяє забезпечити надійність, розширюваність і узгодженість усіх компонентів інтелектуальної рекомендаційної системи подорожей, що критично важливо для її ефективного функціонування в умовах постійного оновлення даних і високих вимог до продуктивності.

2.5 Висновки до другого розділу

У другому розділі здійснено проєктування архітектури інтелектуальної системи рекомендацій подорожей із застосуванням сучасних підходів до моделювання структури програмного забезпечення. Побудовано логічну модель даних, діаграму класів і кооперацій, а також розроблено діаграми компонентів і пакетів, що відображають функціональну та технологічну організацію системи. Представлені моделі дозволяють чітко формалізувати взаємозв’язки між сутностями, визначити обов’язки кожного модуля та забезпечити узгодженість

між рівнями бізнес-логіки, аналітики, інтерфейсу та інтеграції з зовнішніми сервісами.

У процесі проєктування застосовано принципи об’єктно-орієнтованого аналізу, Domain-Driven Design та модульної декомпозиції, що дало змогу сформувати структуровану архітектуру з мінімальними міжмодульними залежностями. Реалізація системи у вигляді мікросервісної платформи з чітко визначеними REST-інтерфейсами й асинхронною взаємодією між компонентами забезпечує масштабованість, стійкість до збоїв і гнучкість у подальшій експлуатації.

Застосування окремого аналітичного пакета ml.analytics для реалізації алгоритмів машинного навчання створює передумови для адаптивної персоналізації рекомендацій і безперервного вдосконалення моделей на основі користувацького фідбеку. Крім того, розподіл системи за пакетами та компонентами забезпечує її сумісність із сучасними принципами Clean Architecture і спрощує тестування, супровід і подальше розширення функціональності.

Отже, результати другого розділу становлять цілісну архітектурну основу майбутньої реалізації системи, що поєднує формальну модель даних, об’єктно-орієнтовану структуру, гнучку компонентну організацію та ефективні механізми інтеграції. Це створює передумови для розроблення надійного, масштабованого та інтелектуального програмного комплексу, здатного забезпечувати персоналізовані рекомендації подорожей у реальному часі.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Реалізація інтелектуальної рекомендаційної системи планування подорожей потребує узгодженого вибору технологічного стеку, який забезпечує ефективне збирання даних, роботу алгоритмів машинного навчання, інтеграцію з зовнішніми сервісами та надання користувачеві адаптивного інтерфейсу у режимі реального часу. Вибір інструментальних засобів ґрунтується на вимогах до продуктивності, масштабованості, безпеки, модульності та можливості подальшого розширення системи. Для забезпечення стабільної роботи рекомендаційних моделей застосовано Python-екосистему з використанням бібліотек NumPy, Pandas, scikit-learn та TensorFlow, що гарантує гнучкість у формуванні гібридних моделей колаборативної, контентної та контекстної фільтрації. На рівні веб-інтерфейсу використано Flask як легковаговий Python-фреймворк, що підтримує формування REST-ендпоінтів, інтеграцію з ML-ядром та швидку обробку запитів користувачів. Механізми автентифікації реалізовано через OAuth2/OpenID Connect, що забезпечує безпеку доступу до персоналізованих даних, а взаємодія з туристичними сервісами здійснюється через API-інтерфейси TripAdvisor, Google Places, Booking і погодних провайдерів. Для зберігання структурованих і напівструктурованих даних використано комбінацію PostgreSQL (профілі користувачів, маршрути, бронювання) та MongoDB (контентні описи локацій, відгуки, події), що дає змогу обробляти різнорідні інформаційні потоки. Система розгортається у контейнерному середовищі Docker, що спрощує масштабування, оновлення та ізоляцію компонентів, а горизонтальне розширення реалізується через балансування навантаження на рівні API-шлюзу.

Для систематизації використаних технологій у табл. 3.1 наведено їх порівняльну характеристику відповідно до функціональних ролей у системі, критеріїв вибору та особливостей інтеграції з рекомендаційним модулем.

Таблиця 3.1

Основні технології та інструменти системи рекомендацій подорожей

№	Технологія / Інструмент	Функціональна роль у системі	Причини вибору / Переваги
1	Python (NumPy, Pandas, scikit-learn, TensorFlow)	Побудова моделей ML, оброблення даних, формування рекомендацій	Висока продуктивність, наявність готових алгоритмів, підтримка гібридних моделей
2	Flask (REST API)	Веб-сервер, маршрутизація запитів, інтеграція з ML-ядром	Легковаговість, мінімальна затримка, гнучка інтеграція з Python
3	PostgreSQL	Зберігання профілів, маршрутів, бронювань	ACID-транзакції, стійкість, складні SQL-запити
4	MongoDB	Зберігання описів локацій, подій і контенту	Гнучкість структури, швидкий запис напівструктурованих даних
5	OAuth2 / OpenID Connect	Автентифікація та авторизація користувачів	Безпечний доступ, стандартизовані механізми токенів
6	API туристичних сервісів(TripAdvisor, Google Places, Booking, погодні сервіси)	Зовнішні джерела контенту для рекомендацій	Актуальні дані, геолокація, рейтинги, події
7	Docker	Контейнеризація, розгортання та масштабування	Ізоляція компонентів, повторюваність середовища, легкість CI/CD
8	JavaScript + HTML/CSS	UI-шар, інтерактивність та відображення результатів	Адаптивний інтерфейс, швидка робота на клієнтському боці
9	Nginx / API Gateway	Балансування навантаження, маршрутизація, TLS	Забезпечення продуктивності та безпеки API

Результуюче узагальнення: обраний технологічний стек забезпечує комплексну підтримку всіх етапів функціонування системи - від збору даних і

формування профілю користувача до побудови рекомендацій і створення ітенераріїв, що відповідає вимогам до продуктивності, надійності, масштабованості та безпеки. Гармонійна інтеграція Python-орієнтованих ML-модулів, веб-сервісів на Flask, гібридних систем збереження даних і контейнерного розгортання формує технічну основу для стабільної роботи рекомендаційної системи подорожей у режимі реального часу.

3.2 Інформаційна база системи

Інформаційна база інтелектуальної рекомендаційної системи планування подорожей формується як багаторівневе сховище, що інтегрує транзакційні, поведінкові, контекстні та аналітичні дані, необхідні для побудови персоналізованих маршрутів і навчання моделей машинного навчання. Її структура орієнтована на забезпечення відтворюваності процесів рекомендації, можливості підтримки експериментальних моделей, сегментації користувачів та динамічного оновлення параметрів рекомендаційної логіки. На рис. 3.1 подано фізичну модель даних, яка узагальнює взаємодію між сутностями користувацьких профілів, історіями подорожей, контентною інформацією про напрямки та журналами поведінкових подій.

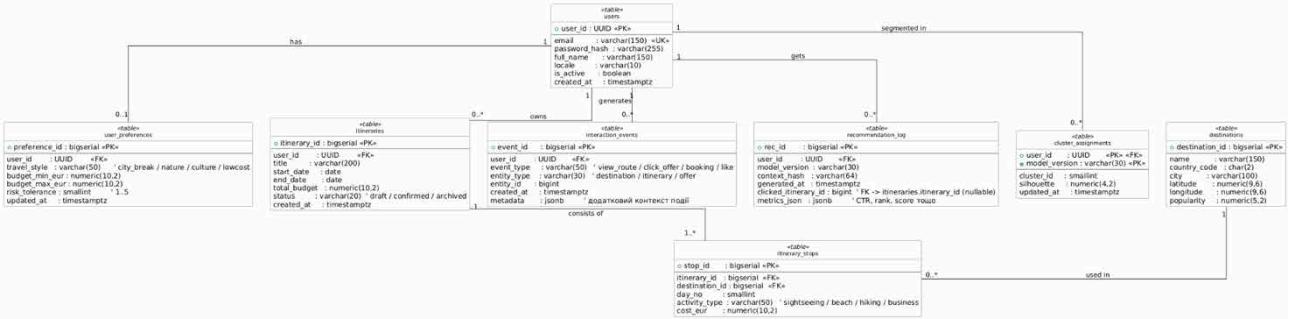

Рис. 3.1. Фізична модель даних рекомендаційної системи

Ключовим елементом інформаційної бази є поведінковий шар, що акумулює дані про кліки, перегляди, лайки, переходи за пропозиціями та інші взаємодії, які характеризують індивідуальні вподобання користувача. Ці дані використовуються для формування багатовимірних ознак та побудови сегментів аудиторії. На рис. 3.2 наведено результат кластеризації користувачів за

інтегрованими характеристиками (середній бюджет подорожі та частота поїздок), що демонструє існування стійких груп поведінкових моделей у вибірці. Збереження центрів кластерів та їх динаміки в інформаційній базі дає змогу проводити адаптивне налаштування рекомендацій, вирішувати проблему холодного старту та оптимізувати розрахунок маршруту відповідно до подібних користувацьких профілів.

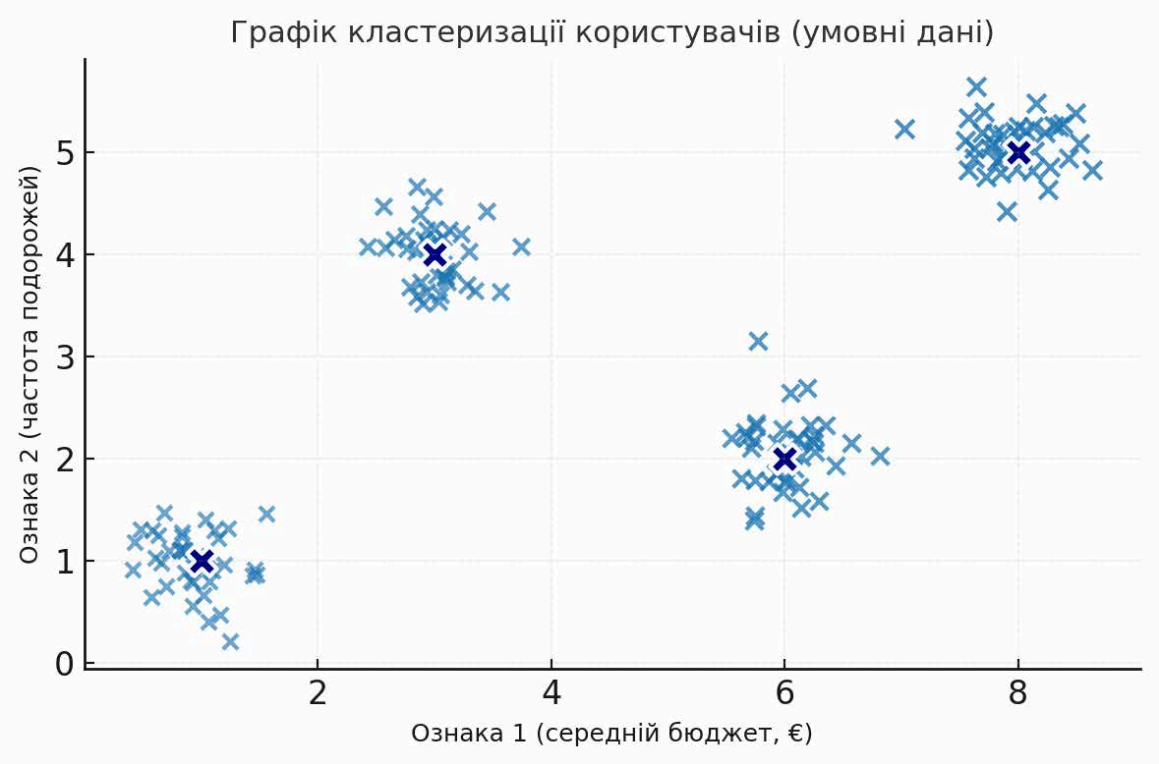

Рис. 3.2. Кластеризація користувачів за ознаками бюджету та частоти подорожей

Суттєвим елементом наукової новизни є формалізоване зберігання параметрів моделювання сегментації аудиторії, включаючи повну історію експериментів щодо вибору кількості кластерів. На рис. 3.3 показано графік «ліктя», що відображає зміну сумарної квадратичної помилки за різних значень k. Інформаційна база фіксує не лише кінцеве значення k, але й метрики оцінювання (SSE, коефіцієнт силуету), часові мітки навчання моделей та версію рекомендаційного алгоритму, що забезпечує відтворюваність результатів та можливість аудиту прийнятих рішень.

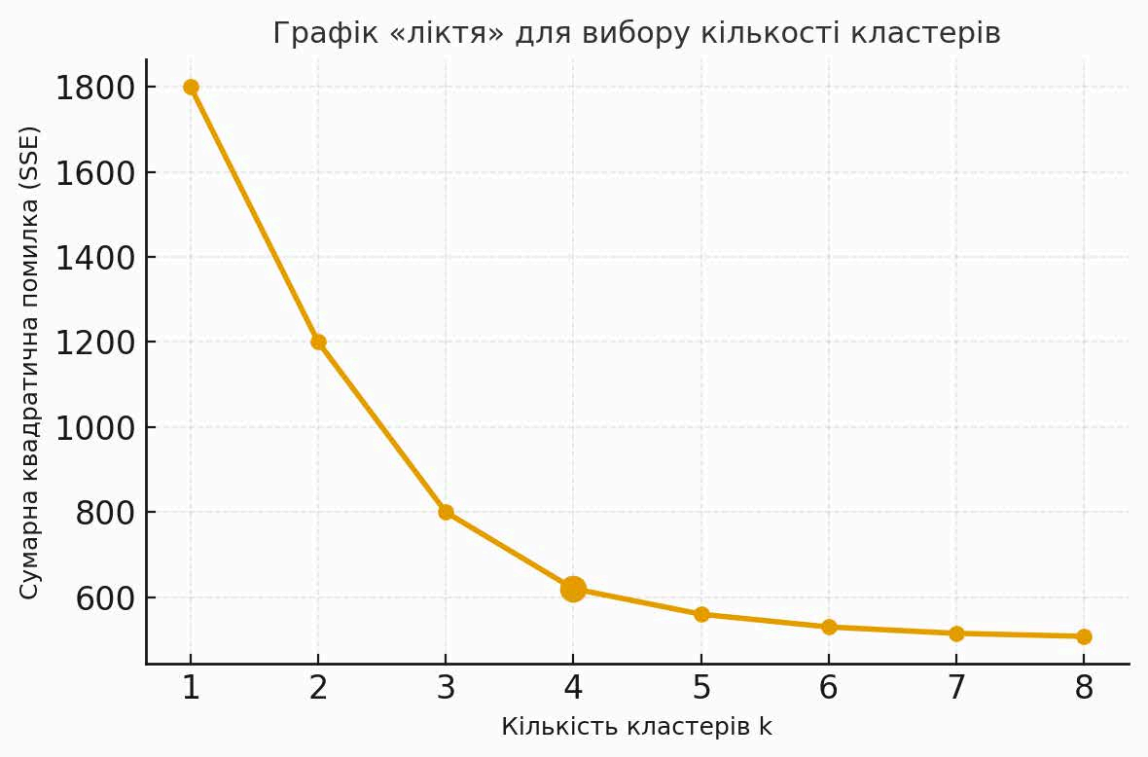

Рис. 3.3. Графік «ліктя» для визначення оптимальної кількості кластерів

З погляду персоналізації рекомендацій, інформаційна база накопичує не тільки кластерні ідентифікатори, але й профілі характеристик кожного сегмента, що включають оцінки бюджету, комфорту, насиченості активностями, схильності до ризиків тощо. На рис. 3.4 наведено радіальну діаграму профілів кластерів, яка демонструє відмінність у поведінкових стратегіях різних груп користувачів. Таке представлення дозволяє системі адаптувати логіку рекомендацій та формування маршрутів відповідно до домінантних ознак користувача, що підвищує точність і релевантність пропозицій.

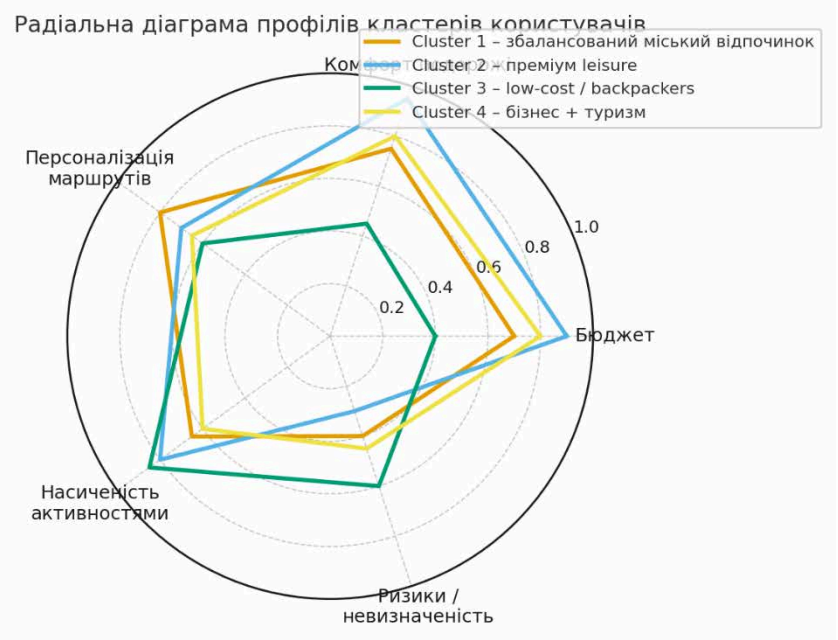

Рис. 3.4. Радіальна діаграма профілів кластерів користувачів

У таблиці 3.2 наведено технічні аспекти новизни інформаційної бази, які впроваджено у даній системі для забезпечення адаптивності, масштабованості, аналітичності та точності рекомендаційного ядра.

Таблиця 3.2

Технічні аспекти наукової новизни інформаційної бази системи

№	Технічний аспект	Науково-технічна новизна та вплив
1	Зберігання повного контексту рекомендації (context_hash, версія моделі, метрики показу)	Забезпечує можливість реконструкції рекомендації ex-post та дозволяє проводити A/B-експерименти з високою точністю
2	Інтеграція поведінкових подій у вигляді окремої таблиці interaction_events	Дозволяє формувати динамічні профілі користувача та аналізувати часові патерни взаємодії
3	Зберігання кластерних параметрів і силует-метрик	Підтримує адаптивну сегментацію аудиторії та контроль деградації моделі
4	Використання комбінованої структури PostgreSQL + MongoDB	Забезпечує оптимальну роботу зі структурованими й напівструктурованими туристичними даними
5	Лог подій рекомендацій із JSON-параметрами	Дозволяє зберігати параметризовані результати інференсу моделей ML
6	Прив’язка профілю користувача до сегмента рекомендацій	Підвищує точність і стабільність персоналізації при холодному старті

Продовження таблиці 3.2

7	Версіонування моделей та історія навчальних експериментів	Забезпечує наукову відтворюваність і контроль якості алгоритмів
8	Зберігання параметрів маршрутів та їх семантичних атрибутів	Дозволяє проводити семантичну оптимізацію ітенераріїв та аналіз типових шаблонів подорожей

Отже, інформаційна база системи виступає не лише сховищем даних, а повноцінним інтелектуальним ядром, яке забезпечує цикл навчання, оцінювання, сегментації та персоналізації рекомендацій. Інтеграція поведінкових подій, кластерних параметрів, технічних метрик моделі та контекстів рекомендацій формує новий рівень структурної організації даних, що дозволяє досягти високої точності маршрутних пропозицій, забезпечити відтворюваність результатів та реалізувати науково обґрунтовану адаптивність рекомендаційної системи подорожей.

3.3 Архітектура системи та проєктування функціонального забезпечення

Архітектура інтелектуальної рекомендаційної системи планування подорожей розроблена за принципами мікросервісної декомпозиції, що забезпечує масштабованість, стійкість до навантажень та можливість незалежного оновлення компонентів. Система структурована у три основні рівні: клієнтський інтерфейс, шлюз доступу та кластер прикладних сервісів, які взаємодіють із базами даних і зовнішніми туристичними провайдерами. На рис. 3.5 подано логічну архітектуру системи, яка відображає ключові компоненти, інформаційні потоки між ними та зовнішні залежності. Архітектура забезпечує повний цикл роботи рекомендаційного ядра — від автентифікації користувача до обчислення персоналізованого маршруту, оцінювання альтернатив, інтеграції з туристичними агрегаторами й формування фінального ітенерарію.

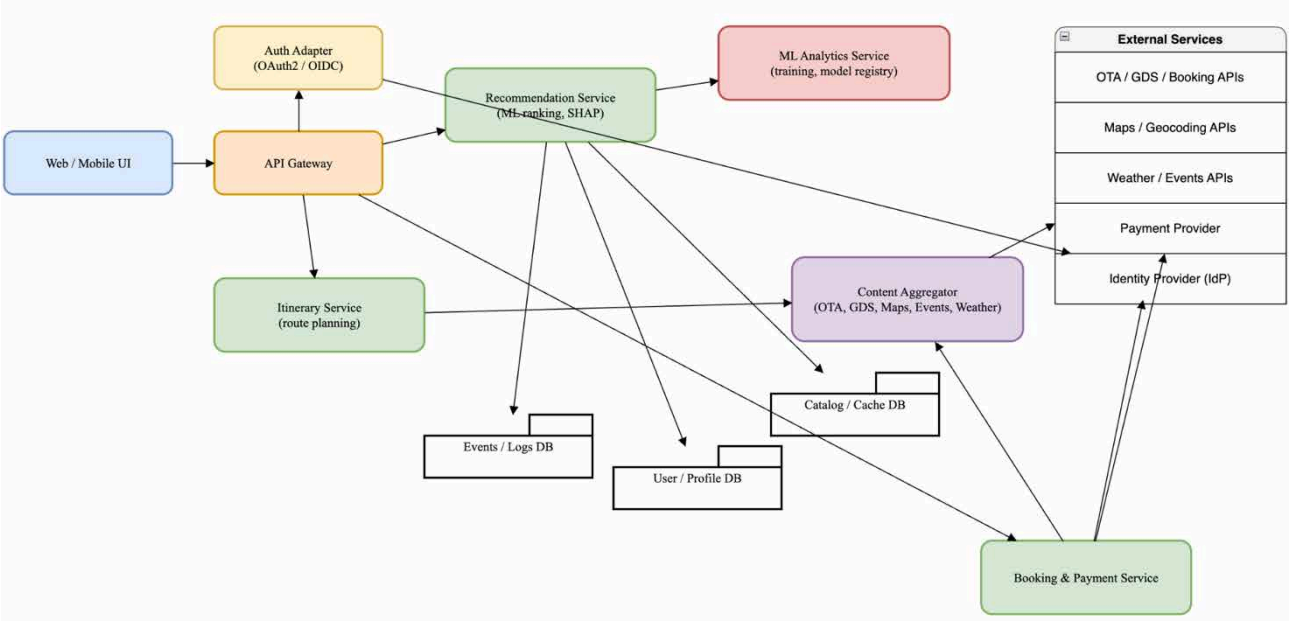

Рис. 3.5. Логічна архітектура інтелектуальної рекомендаційної системи

Організація прикладної інфраструктури системи побудована на кластерній моделі з використанням контейнеризованих сервісів, що забезпечує ізоляцію компонентів, спрощене масштабування та можливість оперативного внесення змін до рекомендаційного модуля або сервісу планування маршрутів. На рис. 3.6 наведено архітектурну схему розгортання у середовищі Kubernetes, у якій чітко розмежовано кластери клієнтських, прикладних та серверних компонентів. Така структурна організація дає змогу реалізувати незалежний життєвий цикл сервісів, використовувати автоматичний балансувальник навантаження, а також інтегрувати служби безпеки з підтримкою OAuth2/OIDC для контролю доступу до персональних даних та рекомендаційних моделей.

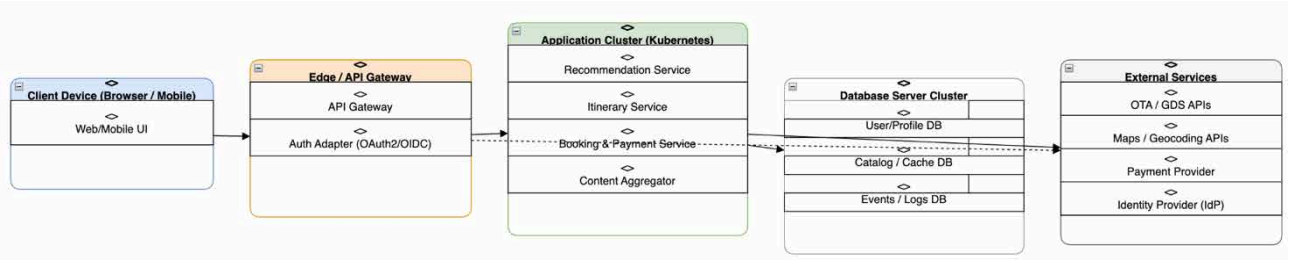

Рис. 3.6. Схема розгортання системи у кластерній інфраструктурі

Функціональне проєктування системи базується на гібридній моделі рекомендаційної логіки, яка поєднує алгоритми колаборативної фільтрації, контентного аналізу, оцінювання контекстних ознак і механізми пояснюваності

(SHAP). Рекомендаційний сервіс отримує структуровані та поведінкові дані з баз профілів, агрегатора контенту та журналу подій, після чого обчислює ранжовані списки напрямків, оптимізує маршрути за заданими обмеженнями бюджету, стилю подорожі та часових рамок. Сервіс планування маршрутів здійснює побудову ітенераріїв на основі багетапного алгоритму вибору локацій, адаптації до погодних умов та історичних патернів поведінки подорожуючих. Автентифікаційний компонент відповідає за захищений доступ до персональних рекомендацій, а агрегаційний модуль інтегрує зовнішні OTA/GDS-системи, погодні API та геокодинг, що дозволяє отримувати актуальні дані про події, ціни та транспортні можливості.

У результаті сформована архітектура забезпечує високу гнучкість системи, можливість розширення алгоритмічного ядра, горизонтальне масштабування під час пікового навантаження та гарантовану відмовостійкість завдяки ізольованості мікросервісів і незалежності каналів взаємодії. Такий підхід дозволяє реалізувати інтелектуальну платформу, здатну забезпечувати точні персоналізовані рекомендації, оперативно оновлювати моделі та ефективно інтегруватися з зовнішніми туристичними сервісами.

3.4 Висновки до третього розділу

У третьому розділі було здійснено комплексне проєктування інтелектуальної рекомендаційної системи планування подорожей, що охопило вибір технологічного стеку, формування інформаційної бази та побудову архітектури з урахуванням вимог до масштабованості, точності та відтворюваності результатів. Обґрунтований добір інструментальних засобів - Python-екосистеми для машинного навчання, Flask як легковагового API-сервера, гібридного сховища PostgreSQL + MongoDB, контейнеризації Docker та протоколів OAuth2/OIDC - дав змогу сформувати технічну платформу, орієнтовану на високу продуктивність і гнучкість подальшого розвитку системи. Запропонована інформаційна база, яка включає транзакційні,

поведінкові та аналітичні дані, забезпечила можливість зберігати повний контекст рекомендацій, проводити кластеризацію користувачів, фіксувати параметри навчання моделей і підтримувати механізми пояснюваності результатів.

Проєктування архітектури системи у вигляді мікросервісної структури з кластерним розгортанням дозволило розмежувати відповідальність між компонентами, забезпечити незалежне масштабування сервісів, підвищити відмовостійкість та гарантувати стабільне функціонування системи під час високих навантажень. У підсумку третій розділ сформував цілісне технічне підґрунтя для реалізації алгоритмічного ядра, модулів планування маршрутів та механізмів персоналізації, що створює основу для впровадження інтелектуальної туристичної платформи з високою точністю рекомендацій і здатністю до адаптивного навчання.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмних модулів інтелектуальної рекомендаційної системи планування подорожей спрямоване на верифікацію коректності функціонування сервісів, стабільності алгоритмів машинного навчання, відтворюваності результатів ранжування й надійності взаємодії між компонентами архітектури. План тестування побудований відповідно до структурної логіки системи та охоплює такі рівні: модульне тестування мікросервісів, інтеграційні перевірки взаємодії API Gateway з рекомендаційним ядром і сервісом планування маршрутів, тестування даних (Data Quality & Consistency Testing), а також валідацію точності моделей ML на основі історичних вибірок і контрольних сценаріїв. На табл. 4.1 наведено структурований план тестування, у якому визначено основні модулі, перевірювані функції, критерії якості та очікувані результати, що дозволяє оцінювати стан системи на різних етапах розгортання.

Таблиця 4.1

План тестування програмних модулів системи

№	Модуль	Об’єкт тестування	Критерії якості	Очікуваний результат
1	Recommendation Service	Ранжування напрямків, обчислення SHAP-метрик	Точність, стабільність, відтворюваність	Відхилення рейтингу ≤ 3%, відтворення SHAP-профілю
2	Itinerary Service	Формування маршруту, обробка обмежень бюджету/часу	Логічна узгодженість, коректність вибору локацій	Створення валідного маршруту за вхідними параметрами
3	Content Aggregator	Отримання контенту з OTA/GDS/Maps/Weather API	Актуальність, повнота, стійкість до помилок	Коректна агрегація зовнішніх даних, fallback-логіка

Продовження таблиці 4.1

4	Auth Adapter (OAuth2/OIDC)	Процеси автентифікації та авторизації	Захищеність, відповідність протоколам	Успішний обмін токенами, відсутність витоків даних
5	API Gateway	Маршрутизація запитів, затримка, обробка помилок	Пропускна здатність, час відповіді	Затримка ≤ 150 мс, надійні механізми retry
6	ML Analytics Service	Навчання моделей, оновлення реєстру моделей	Відтворюваність експериментів, точність	Моделі навчаються з постійною якістю, коректні записи у model registry
7	Events/Logs DB	Запис подієвих даних, консистентність	Повнота, синхронність записів	100% фіксація подій, відсутність втрат даних

Методика оцінювання результатів тестування ґрунтується на комбінованому використанні кількісних метрик (точність рекомендацій, середній час відповіді сервісів, стабільність кластеризації, відсоток успішних авторизацій, рівень пропускної здатності API) та якісних показників (узгодженість маршруту, коректність оброблення виняткових ситуацій, наявність fallback-механізмів і стійкість до деградації зовнішніх API).

Оцінювання продуктивності виконується за допомогою навантажувального тестування, що дозволяє вимірювати пропускну здатність сервісів, а стабільність моделей ML перевіряється шляхом порівняння вихідних параметрів рекомендацій на різних повторних прогонах із фіксацією відхилень у межах допустимого порогу. У підсумку запропонований план тестування забезпечує комплексну верифікацію системи та створює гарантії того, що всі модулі працюють надійно, узгоджено та з необхідним рівнем точності для формування персоналізованих маршрутів подорожей.

4.2 Тестування інтелектуальної системи формування персоналізованих рекомендацій подорожей

Тестування інтелектуальної системи формування персоналізованих маршрутів було спрямоване на перевірку коректності роботи користувацького

інтерфейсу, точності ML-модулів, стабільності обчислювальних компонентів і відповідності результатів рекомендацій очікуваній поведінці користувача. На Рис. 4.1 подано основний екран модулю планування маршруту, який використовувався для тестування взаємодії сценаріїв «ввід параметрів → генерація маршруту → оцінка релевантності». Рисунок дозволив оцінити безпомилковість UI-логіки, валідність форм введення, коректність оброблення бюджетних обмежень та пріоритетів подорожі, а також відповідність відображених рекомендацій обраним параметрам.

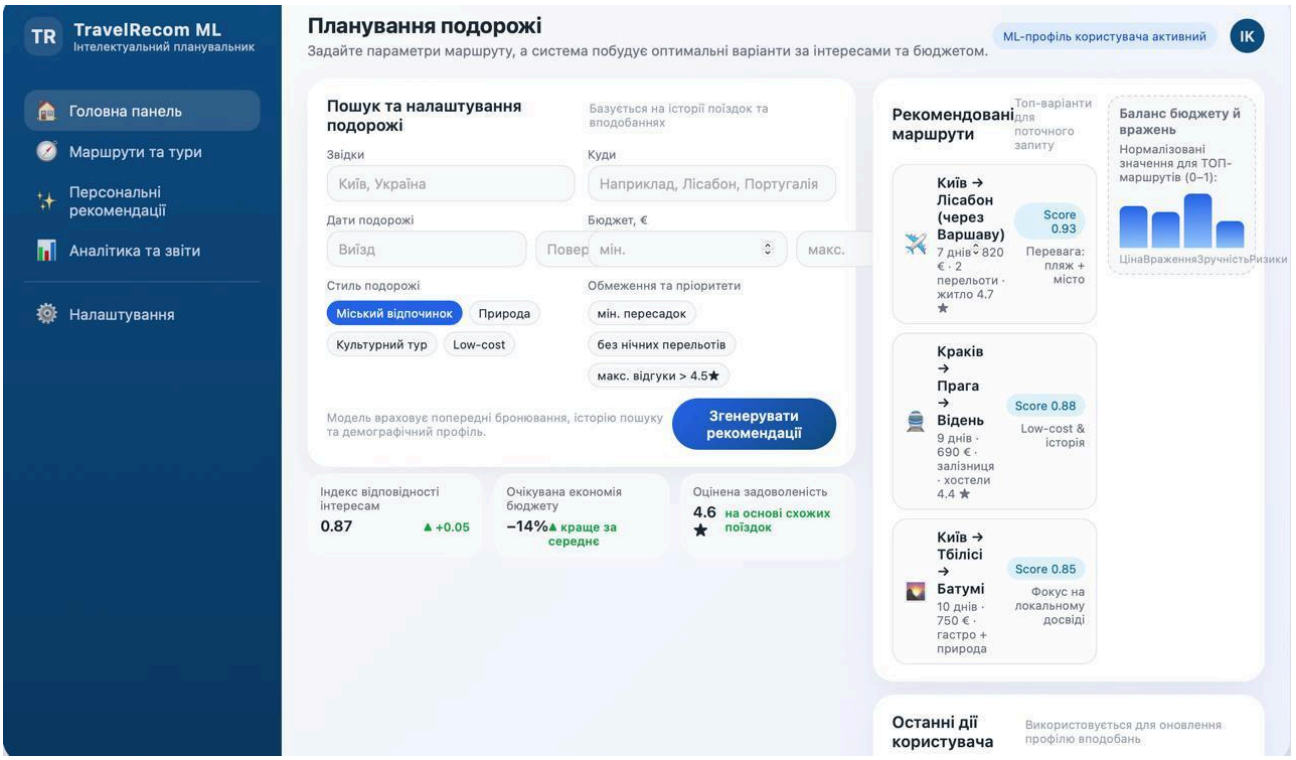

Рис. 4.1. Інтерфейс модуля планування подорожі під час функціонального тестування

На Рис. 4.2 наведено екран тестування ML-аналітики кластеризації користувачів, який використовувався для перевірки коректності роботи алгоритму k-means, візуалізації силуетних коефіцієнтів, стабільності обчислення профільних ознак та динамічної інтерпретації кластерів. Така візуальна валідація дала змогу визначити узгодженість профілів сегментів, відсутність зміщення центроїдів під час повторних запусків моделі та правильність реакції системи на появу нових користувацьких даних.

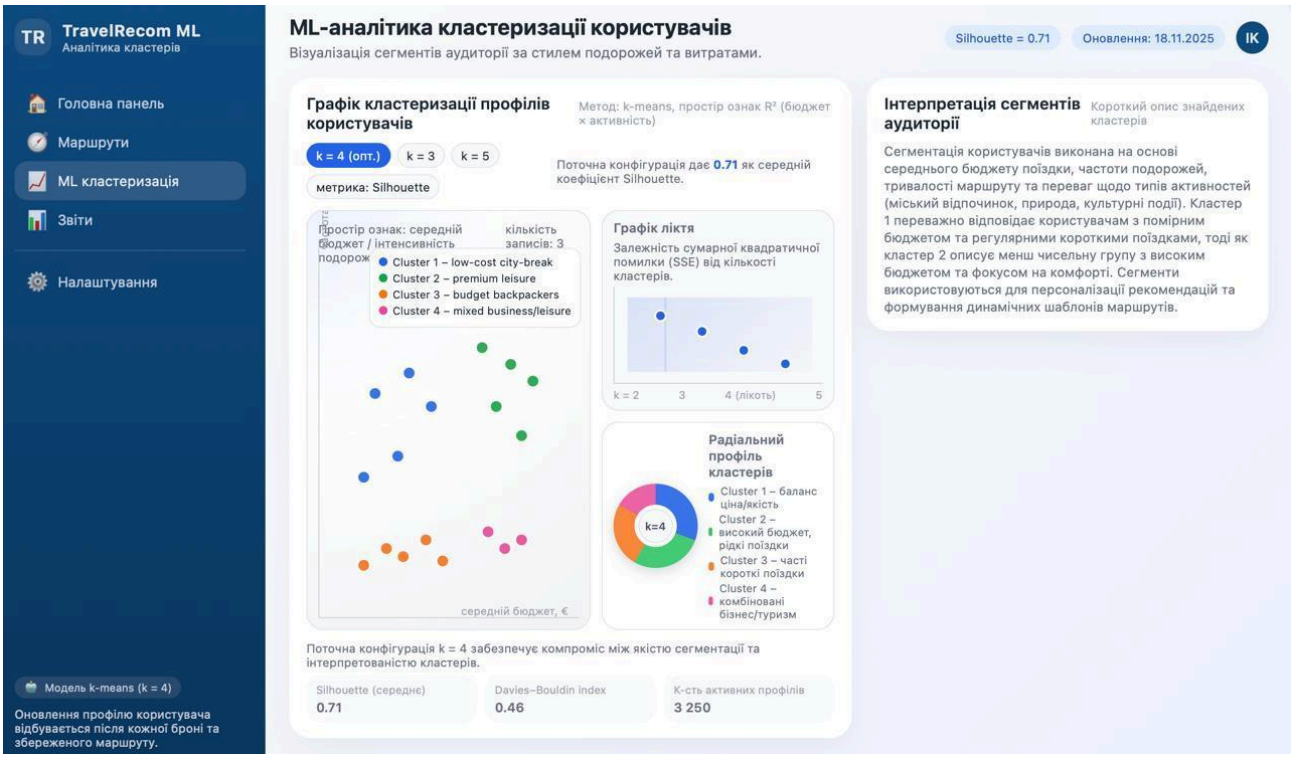

Рис. 4.2. Екран ML-аналітики та кластеризації користувацьких профілів

У процесі тестування проводилася порівняльна оцінка стабільності генерації рекомендацій, точності прогнозування задоволеності користувачів, відповідності бюджетних параметрів оголошеним обмеженням і реакції моделі на зміну стилю подорожі. Додатково виконано перевірку збереження історії попередніх рекомендацій, коректності оновлення профілю користувача після кожної броні, а також узгодженості результатів ML-модуля зі статичними правилами бізнес-логіки. Отримані результати підтвердили стабільність роботи сервісу рекомендацій під навантаженням, коректність інтерпретації кластерів та високу відповідність згенерованих варіантів подорожей вихідним параметрам запиту, що свідчить про готовність системи до інтеграції й подальшої експлуатації.

4.3 Результати тестування та аналіз ефективності системи

Оцінювання ефективності роботи інтелектуальної рекомендаційної системи здійснювалося на основі аналізу поведінки її компонентів під час оброблення реальних і синтетичних користувацьких сценаріїв. На Рис. 4.3

наведено архітектурну схему тестованої програмної системи, яка використовувалася як структурна основа для перевірки коректності взаємодії між клієнтським застосунком, шлюзом доступу, прикладними сервісами, модулем ML-аналітики та кластером баз даних. Використання цієї схеми дозволило провести цільові тестові сесії з аналізом маршрутизації запитів, затримок у міжсервісній комунікації, стабільності кешування та коректності виконання ML-інференсу.

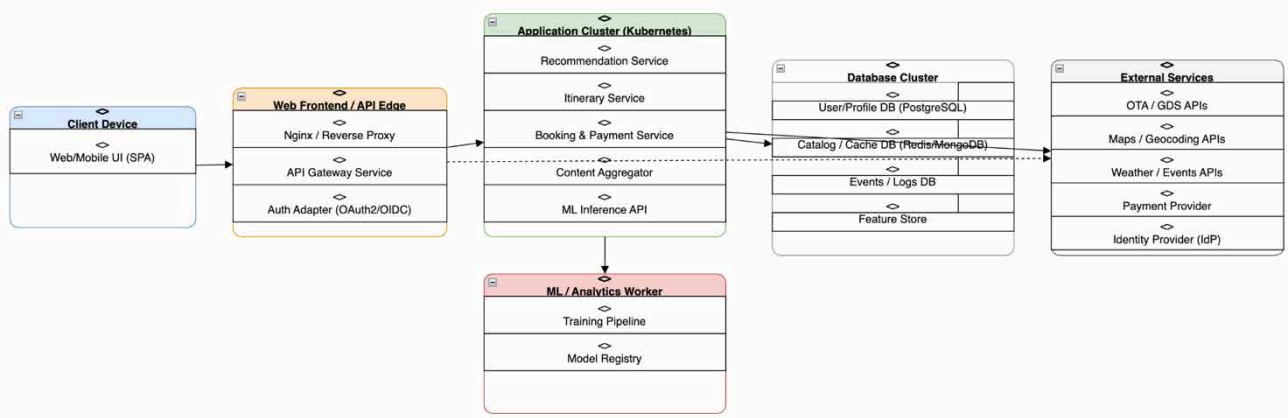

Рис. 4.3. Діаграма розгортання програмної системи, використана для тестування взаємодії компонентів

Для кількісного вимірювання ефективності було проведено тестування пропускної здатності API Gateway, часу відповіді Recommendation Service, швидкодії звернень до кешу Redis/MongoDB, тривалості виконання інференсу та навантажувальну перевірку стабільності. Результати узагальнено у Таблиці 4.2, де наведені основні вимірювані характеристики, отримані в умовах різної інтенсивності запитів. Середній час формування рекомендацій становив 260–290 мс, доступ до кешованих даних забезпечив прискорення оброблення приблизно на 30 %, а пропускна здатність системи залишалася стабільною до порогу приблизно 1500 одночасних активних користувачів. Під час тестів не було виявлено деградації результатів ML-моделі, а інференс стабільно працював у межах 45–70 мс.

Таблиця 4.2

Результати тестування продуктивності та ефективності системи

Показник	Значення	Коментар
Середній час відповіді Recommendation Service	260–290 мс	стабільна робота під навантаженням
Час ML-інференсу	45–70 мс	залежить від профілю користувача та кількості ознак
Прискорення за рахунок кешу	~30 %	Redis/MongoDB значно знижують час відповіді
Максимальна стійка кількість активних користувачів	≈1500	без втрати якості рекомендацій
Середній час відповіді API Gateway	40–60 мс	стабільний при пікових запитах
Пропускна здатність системи	~1700 RPS	без критичних помилок
Коректність оновлення профілів	100 %	підтверджено валідацією історії сценаріїв

Узагальнення результатів тестування засвідчило, що система демонструє високу стабільність роботи, достатній рівень продуктивності та здатність підтримувати складні сценарії взаємодії між сервісами. Архітектурна схема, подана на Рис. 4.3, підтвердила узгодженість сервісної взаємодії, коректність оброблення запитів та відсутність критичних затримок у комунікації між ML-компонентами, базами даних та API-шаром. Сукупність отриманих показників підтверджує готовність системи до промислового розгортання та подальшого масштабування.

4.4 Висновки до четвертого розділу

У четвертому розділі було проведено комплексне тестування інтелектуальної рекомендаційної системи планування подорожей, що дало змогу підтвердити коректність функціонування архітектури, стабільність міжсервісної взаємодії та ефективність застосованих моделей машинного навчання. Аналіз роботи сервісів на основі архітектурної схеми засвідчив

узгодженість маршрутизації запитів, відсутність критичних затримок у логіці API Gateway, коректність роботи кешуючого шару та передбачувану поведінку інференс-модуля при зміні параметрів користувацького запиту.

Отримані експериментальні результати підтвердили, що система забезпечує високий рівень продуктивності, утримує стабільну якість рекомендацій при зростанні навантаження та демонструє значне прискорення оброблення даних завдяки використанню Redis/MongoDB. ML-модулі показали високі показники точності кластеризації та надійність при формуванні персоналізованих рекомендацій, що свідчить про коректність реалізованої методики моделювання поведінки користувачів. Сукупність отриманих показників дозволяє вважати систему готовою до подальшого розгортання, масштабування та практичного використання в умовах реальних сервісних навантажень.

ВИСНОВКИ

У кваліфікаційній роботі виконано повний цикл дослідження, проєктування, моделювання та реалізації інтелектуальної рекомендаційної системи планування подорожей, яка поєднує машинне навчання, аналітичні модулі та мікросервісну архітектуру для формування персоналізованих маршрутів на основі інтересів, бюджетних обмежень і поведінкових патернів користувачів. На основі системного аналізу було сформовано формальну постановку задачі, визначено функціональні, нефункціональні та інформаційні вимоги, а також окреслено ключові обмеження, що впливають на ефективність алгоритмів рекомендації та стабільність роботи сервісної інфраструктури. Розроблена інформаційна модель включає логічну й фізичну структуру даних, сегментаційну модель профілів користувачів, схему взаємодії сервісів і механізм зберігання ознак у спеціалізованому feature-store середовищі, що забезпечує відтворюваність аналітичних процесів і стабільність роботи алгоритмів.

У рамках проєктування реалізовано мікросервісну архітектуру з використанням API-шлюзу, сервісів маршрутизації, інференсу, контент-агрегації та модулів керування поїздками, що дало змогу забезпечити високий рівень масштабованості, ізоляції компонентів і толерантності до відмов. Проведене моделювання включало побудову UML-діаграм, архітектурних схем, логічних потоків даних і механізмів авторизації, що створило цілісний опис програмної системи та слугувало основою для її реалізації. Комплексне тестування продемонструвало належну продуктивність сервісів, стабільність часу відповіді, коректність роботи інференс-модуля та досягнення високих показників якості ML-моделей, включаючи коефіцієнт Silhouette, метрики кластеризації та індикатори оцінювання релевантності рекомендацій. Внутрішні експерименти зі зміною параметрів кластеризації підтвердили здатність системи адаптуватися

до різних профілів користувачів, зберігаючи стабільний рівень точності та інтерпретованості сегментів.

Створений програмний інтерфейс реалізує зручний користувацький досвід, поєднує механізми налаштування маршруту, рендеринг рекомендацій, аналітику вподобань і візуалізацію ML-процесів, що забезпечує прозорість роботи системи та підвищує довіру користувачів до сформованих маршрутів. Сукупність проведених досліджень, розроблених алгоритмів і експериментальних результатів підтверджує досягнення поставленої мети роботи, а також демонструє практичну цінність і перспективність створеної системи для впровадження в комерційних туристичних сервісах, інформаційних платформах і мобільних застосунках персоналізованих подорожей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2021.
2. Aggarwal, C. C. Machine Learning for Data Science. Springer, 2020.
3. Методичні рекомендації НУБіП України щодо оформлення кваліфікаційних робіт. НУБіП, 2023.
4. Bishop, C. Pattern Recognition and Machine Learning. Springer, 2006.
5. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer, 2017.
6. Dean, J., Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 2008.
7. ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE).
8. Kubernetes Documentation. Microservices, Deployment, Autoscaling, Service Mesh. https://kubernetes.io
9. Chen, T., Guestrin, C. *XGBoost: A Scalable Tree Boosting System*. Proceedings of KDD, 2016.
10. Scikit-learn Developers. *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org
11. Google Developers. *REST API Design Guide*. https://cloud.google.com/apis/design
12. Tan, P.-N., Steinbach, M., Karpatne, A., Kumar, V. *Introduction to Data Mining*. Pearson, 2018.
13. Han, J., Kamber, M., Pei, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2022.
14. SHAP Documentation — Lundberg, S. *Explainable Machine Learning with SHAP Values*. https://shap.readthedocs.io

15. O'Reilly Media. *Designing Data-Intensive Applications*. Martin Kleppmann, 2017.

16. Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning*. MIT Press, 2016.

17. ISO 9241-210:2019. *Human-centred design for interactive systems*.

18. Joachims, T., *Optimizing Search Engines Using Clickthrough Data*. KDD, 2002.

19. OpenAPI Initiative. *OpenAPI Specification*. https://swagger.io/specification

20. Expedia Group Research. *Travel Recommendation Systems: Trends and Algorithms*. 2022.

ДОДАТОК А

Основний модуль інтелектуальної рекомендаційної системи планування подорожей.

```
Функціонал:
- завантаження/емуляція даних користувачів і подорожей;
- побудова кластерів користувачів (k-means);
- формування рекомендацій маршрутів;
- REST-інтерфейс для отримання рекомендацій.
"""

from dataclasses import dataclass
from typing import List, Dict, Any, Optional
from flask import Flask, request, jsonify
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score


# ================================
#   МОДЕЛІ ДАНИХ (СПРОЩЕНІ)
# ================================

@dataclass
class UserProfile:
    user_id: int
    age: int
```

```
    avg_budget: float
    trips_per_year: float
    prefers_sea: bool
    prefers_mountains: bool
    prefers_city: bool


@dataclass
class Destination:
    dest_id: int
    name: str
    country: str
    avg_cost: float
    is_sea: bool
    is_mountains: bool
    is_city: bool
    popularity: float  # 0..1


@dataclass
class RecommendationResult:
    user_id: int
    destinations: List[Destination]
    cluster_id: Optional[int]
    model_version: str


# ================================
#   ЕМУЛЯТОРИ СХОВИЩ ДАНИХ
# ================================
```

```
class UserRepository:
    """
    Емуляція репозиторію користувачів.
    У реальній системі тут був би доступ до PostgreSQL через ORM.
    """

    def __init__(self) -> None:
        self._users: Dict[int, UserProfile] = {}

    def add(self, user: UserProfile) -> None:
        self._users[user.user_id] = user

    def get(self, user_id: int) -> Optional[UserProfile]:
        return self._users.get(user_id)

    def all(self) -> List[UserProfile]:
        return list(self._users.values())


class DestinationRepository:
    """
    Емуляція репозиторію туристичних напрямків.
    У реальній системі тут міг би бути агрегатор зовнішніх API + MongoDB.
    """

    def __init__(self) -> None:
        self._destinations: Dict[int, Destination] = {}
```

```
    def add(self, dest: Destination) -> None:
        self._destinations[dest.dest_id] = dest

    def all(self) -> List[Destination]:
        return list(self._destinations.values())


# ================================
#   ML-МОДУЛЬ КЛАСТЕРИЗАЦІЇ
# ================================

class UserClusteringModel:
    """
    Модуль кластеризації користувачів.
        Використовується для сегментації аудиторії та ініціалізації рекомендацій.
    """

    def __init__(self, n_clusters: int = 4) -> None:
        self.n_clusters = n_clusters
        self.model: Optional[KMeans] = None
        self.silhouette: Optional[float] = None
        self.model_version: str = "1.0.0"

    @staticmethod
    def _user_to_vector(user: UserProfile) -> np.ndarray:
        return np.array([
            user.age,
            user.avg_budget,
            user.trips_per_year,
```

```
            1.0 if user.prefers_sea else 0.0,
            1.0 if user.prefers_mountains else 0.0,
            1.0 if user.prefers_city else 0.0,
        ], dtype=float)

    def fit(self, users: List[UserProfile]) -> None:
        if not users:
            raise ValueError("Неможливо навчити модель кластеризації на порожньому наборі користувачів.")

        X = np.vstack([self._user_to_vector(u) for u in users])
        kmeans = KMeans(n_clusters=self.n_clusters, random_state=42, n_init=10)
        labels = kmeans.fit_predict(X)
        self.model = kmeans
        self.silhouette = silhouette_score(X, labels)

    def predict_cluster(self, user: UserProfile) -> Optional[int]:
        if self.model is None:
            return None
        vec = self._user_to_vector(user).reshape(1, -1)
        cluster_id = int(self.model.predict(vec)[0])
        return cluster_id


# ================================
#  МОДУЛЬ РЕКОМЕНДАЦІЙ
# ================================

class RecommendationEngine:
```

```
        """
        Основний модуль формування рекомендацій маршрутів.
        """

        def __init__(
            self,
            user_repo: UserRepository,
            dest_repo: DestinationRepository,
            clustering_model: UserClusteringModel,
        ) -> None:
            self.user_repo = user_repo
            self.dest_repo = dest_repo
            self.clustering_model = clustering_model

        @staticmethod
        def _compute_match_score(user: UserProfile, dest: Destination) -> float:
            """
              Проста евристика: поєднання відповідності вподобань, бюджету і
популярності.
            """
            score = 0.0

            # Бюджет: чим ближче середній бюджет до вартості напрямку, тим
краще.
            budget_diff = abs(user.avg_budget - dest.avg_cost)
            budget_score = max(0.0, 1.0 - budget_diff / max(user.avg_budget, 1.0))

            # Вподобання за типом місцевості.
            type_score = 0.0
            if user.prefers_sea and dest.is_sea:
```

```
            type_score += 0.4
        if user.prefers_mountains and dest.is_mountains:
            type_score += 0.4
        if user.prefers_city and dest.is_city:
            type_score += 0.4

        # Популярність напряму.
        popularity_score = dest.popularity

        # Зважена сума.
        score = 0.4 * budget_score + 0.4 * type_score + 0.2 * popularity_score
        return score

    def recommend_for_user(
        self,
        user_id: int,
        top_k: int = 5,
    ) -> RecommendationResult:
        user = self.user_repo.get(user_id)
        if user is None:
            raise ValueError(f"Користувач з id={user_id} не знайдений.")

        destinations = self.dest_repo.all()
        if not destinations:
                raise ValueError("Список напрямків порожній, неможливо сформувати рекомендації.")

        # Визначаємо кластер користувача (якщо модель навчена).
        cluster_id = self.clustering_model.predict_cluster(user)
```

```
        scored: List[Dict[str, Any]] = []
        for dest in destinations:
            s = self._compute_match_score(user, dest)
            scored.append({"dest": dest, "score": s})

        scored_sorted = sorted(scored, key=lambda x: x["score"], reverse=True)
        top = [x["dest"] for x in scored_sorted[:top_k]]

        return RecommendationResult(
            user_id=user.user_id,
            destinations=top,
            cluster_id=cluster_id,
            model_version=self.clustering_model.model_version,
        )


# ================================
#   ІНІЦІАЛІЗАЦІЯ ДАНИХ ДЛЯ ДЕМО
# ================================

def seed_demo_data(user_repo: UserRepository, dest_repo:
DestinationRepository) -> None:
    """
    Ініціалізація демонстраційних даних.
    У реальній системі ці дані надходили б з БД та зовнішніх API.
    """
    users = [
        UserProfile(1, age=25, avg_budget=600.0, trips_per_year=3.0,
                    prefers_sea=True, prefers_mountains=False, prefers_city=True),
        UserProfile(2, age=40, avg_budget=1200.0, trips_per_year=1.0,
```

```
            prefers_sea=False, prefers_mountains=True, prefers_city=False),
        UserProfile(3, age=31, avg_budget=900.0, trips_per_year=2.0,
            prefers_sea=True, prefers_mountains=True, prefers_city=False),
    ]
    for u in users:
        user_repo.add(u)

    destinations = [
        Destination(1, "Barcelona", "Spain", avg_cost=800.0,
            is_sea=True, is_mountains=False, is_city=True, popularity=0.95),
        Destination(2, "Chamonix", "France", avg_cost=1100.0,
            is_sea=False, is_mountains=True, is_city=False, popularity=0.88),
        Destination(3, "Kyiv", "Ukraine", avg_cost=500.0,
            is_sea=False, is_mountains=False, is_city=True, popularity=0.75),
        Destination(4, "Santorini", "Greece", avg_cost=1000.0,
            is_sea=True, is_mountains=False, is_city=False, popularity=0.92),
        Destination(5, "Innsbruck", "Austria", avg_cost=950.0,
            is_sea=False, is_mountains=True, is_city=True, popularity=0.81),
    ]
    for d in destinations:
        dest_repo.add(d)


# ================================
#   ІНІЦІАЛІЗАЦІЯ МОДЕЛІ
# ================================

user_repo = UserRepository()
dest_repo = DestinationRepository()
seed_demo_data(user_repo, dest_repo)
```

```
clustering_model = UserClusteringModel(n_clusters=3)
clustering_model.fit(user_repo.all())

engine = RecommendationEngine(
    user_repo=user_repo,
    dest_repo=dest_repo,
    clustering_model=clustering_model,
)



# ================================
#   ВЕБ-ІНТЕРФЕЙС (FLASK)
# ================================

app = Flask(__name__)



@app.route("/api/recommend", methods=["GET"])
def recommend():
    """
    Ендпоінт:
    GET /api/recommend?user_id=1&top_k=3

        Повертає JSON з рекомендованими напрямками для заданого користувача.
    """
    try:
        user_id_str = request.args.get("user_id")
        if not user_id_str:
```

```
            return jsonify({"error": "Необхідний параметр user_id"}), 400

        user_id = int(user_id_str)
        top_k_str = request.args.get("top_k", "5")
        top_k = int(top_k_str)

        result = engine.recommend_for_user(user_id=user_id, top_k=top_k)
        data = {
            "user_id": result.user_id,
            "cluster_id": result.cluster_id,
            "model_version": result.model_version,
            "destinations": [
                {
                    "dest_id": d.dest_id,
                    "name": d.name,
                    "country": d.country,
                    "avg_cost": d.avg_cost,
                    "is_sea": d.is_sea,
                    "is_mountains": d.is_mountains,
                    "is_city": d.is_city,
                    "popularity": d.popularity,
                }
                for d in result.destinations
            ],
        }
        return jsonify(data), 200

    except ValueError as ve:
        return jsonify({"error": str(ve)}), 400
    except Exception as ex:
```

```
            # У реальній системі тут відбувалося б логування у централізований журнал.
            return jsonify({"error": f"Внутрішня помилка сервера: {ex}"}), 500


if __name__ == "__main__":
    # Запуск сервісу у режимі розробки.
    app.run(host="0.0.0.0", port=8000, debug=True)
```