Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

з теми:

«Методи та засоби створення вебплатформи подій з підтримкою персоналізованих рекомендацій»

Виконав: здобувач вищої освіти
групи KN1-M24
спеціальності 122 Комп’ютерні науки
Олександр КУЧМІЙ

Керівник:
Володимир ФЕДОРЧУК, доктор технічних наук, професор кафедри комп’ютерних наук

Науковий консультант:
Віталій ІВАНЮК, доктор технічних наук, доцент

Рецензент:
Сергій ОПТАСЮК, кандидат фізико-математичних наук, завідувач кафедри фізики

Кам’янець-Подільський – 2025 р

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ 3
АНОТАЦІЯ 4
ВСТУП 6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1 Огляд ринку платформ для організації міських подій 9
1.2 Аналіз існуючих рішень та конкурентів 11
1.3 Визначення проблем та потреб користувачів 13
1.4 Постановка задачі та формулювання вимог 14
Висновки до розділу 1 16
РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ВЕБ-ПЛАТФОРМИ URBAN GATHER 18
2.1 Визначення функціональних компонентів системи 18
2.2 Проектування бази даних та моделі даних 21
2.3. Архітектура клієнт-серверної взаємодії 24
2.4. Проектування системи персоналізованих рекомендацій 26
2.5 Використання картографічних сервісів 31
Висновки до розділу 2 33
РОЗДІЛ 3. РОЗРОБКА ВЕБ-ПЛАТФОРМИ 34
3.1 Вибір технологічного стеку та інструментів розробки 34
3.2 Розробка серверної частини 36
3.3 Розробка клієнтської частини 36
3.4 Реалізація системи рекомендацій 39
3.5 Інтеграція зовнішніх сервісів 42
ВИСНОВКИ ДО РОЗДІЛУ 3 43
ВИСНОВКИ 45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48
ДОДАТКИ 51
Додаток А. Тестування веб-платформи 51
Додаток Б. Структура бази даних 57
Додаток В. Аналіз веб-платформ для подій 60
Додаток Г. Архітектура клієнт-серверної взаємодії 63
Додаток Д. Інтеграція зовнішніх сервісів 68
Додаток Е. Аналіз трендів розвитку та сучасний 70
Додаток Ж. Серверний модуль вебплатформи 72
Додаток К. Ілюстрації інтерфейсу вебплатформи Urban Gather 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

Urban Gather - власна назва веб-платформи яка розроблялась в межах кваліфікаційної роботи.

CI/CD - Continuous Integration/Continuous Deployment (неперервна інтеграція/неперервне розгортання).

E2E - End-to-End Testing (наскрізне тестування).

RBAC - Role-Based Access Control (контроль доступу на основі ролей).

REST - Representational State Transfer (передача стану уявлення).

SEO - Search Engine Optimization (оптимізація для пошукових систем).

SSR - Server-Side Rendering (відтворення на стороні сервера).

UX - User Experience (користувацький досвід).

АНОТАЦІЯ

Тема роботи: Методи та засоби створення вебплатформи подій з підтримкою персоналізованих рекомендацій.

Актуальність дослідження. Зростання міської культури та цифрова трансформація суспільства в Україні вимагає ефективних технологічних рішень для організації та управління міськими заходами. Організатори заходів, волонтери та споживачі стикаються з труднощами через фрагментованість існуючих платформ і відсутність рішень, спеціалізованих для українського ринку. Координація волонтерського руху та персоналізація користувацького досвіду є особливо важливими питаннями.

Мета дослідження – розробити веб-платформу під назвою Urban Gather яка включає алгоритми машинного навчання для персональних рекомендацій та ефективної організації міських заходів в Україні. Вона повинна бути здатною обслуговувати три основні групи учасників: волонтерів, організаторів і відвідувачів.

Методи дослідження. У роботі використовувалися методи системного аналізу для вивчення предметної області та визначення вимог до системи; методи об’єктно-орієнтованого проектування для створення архітектури платформи; UML-моделювання для візуалізації компонентів системи; алгоритми машинного навчання (колаборативна та контентна фільтрація) для створення гібридної системи рекомендацій; і методи тестування програмного забезпечення, включаючи юніт і інтеграційне тестування, для валідації ефективності.

Наукова новизна. Вперше для українського ринку розроблено комплексну модель веб-платформи, що інтегрує потреби трьох типів користувачів (відвідувачі, організатори, волонтери) в єдиній системі. Запропоновано гібридний підхід до рекомендаційної системи, адаптований до специфіки українського ринку культурних та громадських заходів. Створено інноваційну систему гейміфікації для мотивації волонтерської діяльності, адаптовану до українського контексту громадянського суспільства.

Практична цінність. Розроблена платформа Urban Gather може бути впроваджена для підвищення ефективності організації міських заходів, покращення координації між учасниками та розвитку громадської активності. Система забезпечує персоналізований досвід для користувачів, автоматизацію процесів планування та детальну аналітику ефективності заходів. Платформа сприяє розвитку міської культури, активізації громадянського суспільства та покращенню якості дозвілля городян.

Основні результати. Створено повнофункціональну веб-платформу з 16 функціональними компонентами, базою даних з 42 моделями, понад 80 API ендпоінтами та гібридною системою рекомендацій.

Ключові слова: веб-платформа, міські заходи, система рекомендацій, гейміфікація, волонтерство, персоналізація, гібридна фільтрація, колаборативна фільтрація, контентна фільтрація, Next.js, TypeScript, PostgreSQL, React, цифрова трансформація.

ВСТУП

Актуальність дослідження. В умовах прискорення цифрової трансформації українського суспільства та активізації міської культури виникає потреба у створенні ефективних технологічних рішень для організації та управління міськими заходами. Сучасні тенденції розвитку "розумних міст" та зростання громадської активності в Україні потребують інноваційних підходів до координації культурних, освітніх та соціальних ініціатив. Існує багато перешкод для ефективної співпраці між організаторами, волонтерами та відвідувачами заходів через різноманітність існуючих платформ для організації заходів і відсутність спеціалізованих продуктів, адаптованих до потреб українського ринку.

Особливо важливою проблемою є координація волонтерського руху, який відіграє важливу роль у підтримці міських ініціатив, але потерпає від недостатньої організаційної підтримки та відсутності механізмів мотивації. Недостатньо розроблений досвід персоналізації культурних і громадських заходів, що призводить до зниження залученості громадян і ефективності заходів [1].

Сучасні платформи для організації подій, такі як Eventbrite та Meetup, здебільшого орієнтовані на західні ринки та не враховують правила, культурні традиції та поведінкові патерни українських користувачів. Відсутність інтегрованих рішень, які поєднують потреби різних типів користувачів у єдиній екосистемі, створює додаткові перешкоди для розвитку міських спільнот і культурних проектів.

Мета дослідження – розробити веб-платформу Urban Gather, яка використовує алгоритми машинного навчання, щоб персоналізувати досвід користувачів і ефективно організовувати міські заходи в Україні. Платформа повинна забезпечувати різноманітні функції для трьох основних груп користувачів: відвідувачів, організаторів і волонтерів.

Завдання дослідження. Для досягнення мети було визначено наступні завдання:

1. Дослідити поточний стан ринку українських платформ для організації міських заходів і проаналізувати існуючі рішення.;
2. Проаналізувати потреби різних груп користувачів та визначити функціональні вимоги до системи;
3. Спроєктувати архітектуру веб-платформи враховуючи маштабованість та модульність;
4. Розробити гібридну систему машинного навчання для надання персоналізованого досвіду користування платформою та рекомендацій заходів;
5. Реалізувати веб-платформу Urban Gather з використанням сучасних технологій та фреймворків;
6. Впровадити систему гейміфікації для мотивації волонтерської діяльності;
7. Провести комплексне тестування системи та оцінити її ефективність.

Об'єкт дослідження – процес організації та управління міськими заходами з використанням цифрових технологій.

Предмет дослідження – методи та засоби розробки веб-платформ для координації міських заходів з інтеграцією алгоритмів машинного навчання.

Методи дослідження. Методологія дослідження базується на системному аналізі предметної області та існуючих рішень для організації міських подій, що дозволяє визначити вимоги до системи та застосувати об'єктно-орієнтоване проектування для створення модульної архітектури платформи, а також методів машинного навчання (колаборативної та контентної фільтрації) для розробки гібридної системи рекомендацій та комплексного тестування програмного забезпечення для верифікації якості розробленого рішення.

Наукова новизна одержаних результатів:

- вдосконалено гібридний підхід до рекомендаційної системи, який полягає в динамічній адаптації вагових коефіцієнтів між колаборативною та контентною фільтрацією залежно від рівня активності користувача, що дозволяє забезпечити релевантні рекомендації як для нових користувачів без історії взаємодій, так і для активних користувачів з історією користування платформою. На відміну від

традиційних підходів із статичними коефіцієнтами, запропонований метод автоматично балансує джерела рекомендацій, що вирішує проблему "холодного старту" та суттєво підвищує конверсію та швидкість прийняття рішень користувачами.

Апробація одержаних результатів:

1. Участь у науковій конференції студентів і магістрантів за підсумками НДР у 2024-2025 н.р. (9-10 квітня 2025 р.) [2].

Практичне значення одержаних результатів. Розроблена платформа Urban Gather може бути впроваджена для підвищення ефективності організації міських заходів, покращення координації між учасниками та розвитку громадської активності. Система забезпечує персоналізований досвід для користувачів, автоматизацію процесів планування та детальну аналітику ефективності заходів. Платформа сприяє розвитку міської культури, активізації громадянського суспільства та покращенню якості дозвілля городян. Результати дослідження можуть бути використані органами місцевого самоврядування, громадськими організаціями та комерційними структурами для організації масштабних міських ініціатив.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Цифрова трансформація охоплює всі частини сучасного суспільства, організація міських заходів не є винятком. Кожного дня у великих містах України відбуваються десятки культурних, освітніх, розважальних та соціальних заходів, але інформація про них розгалужена по численним сайтам і соціальним мережам. Це обмежує розвиток міської культури та громадянської активності, перешкоджаючи як відвідувачам, так і організаторам заходів.

У цьому розділі розглянуто предметну область організації міських подій, розглянуто існуючі рішення на глобальному та локальному ринках, визначено основні проблеми та потреби користувачів. Крім того, у цьому розділі визначено вимоги до веб-платформи Urban Gather, мета якої полягає в тому, щоб вирішити ці проблеми, створивши єдиний цифровий простір для всіх учасників екосистеми міських подій.

1.1 Огляд ринку платформ для організації міських подій

Сучасний стан ринку

Ринок платформ для організації міських подій переживає період активного зростання. За даними Grand View Research глобальний ринок event management software оцінювався в $6.4 млрд у 2023 році з прогнозованим щорічним зростанням 11.8% до 2030 року [3]. Пандемія COVID-19 прискорила цифровізацію індустрії подій, створивши попит на комплексні онлайн-рішення для планування, просування та управління заходами. На рисунку 1.1 ми можемо поглянути на динаміку росту ринку платформ для організації подій в світі починаючи з 2018 року з прогнозом до 2030 року.

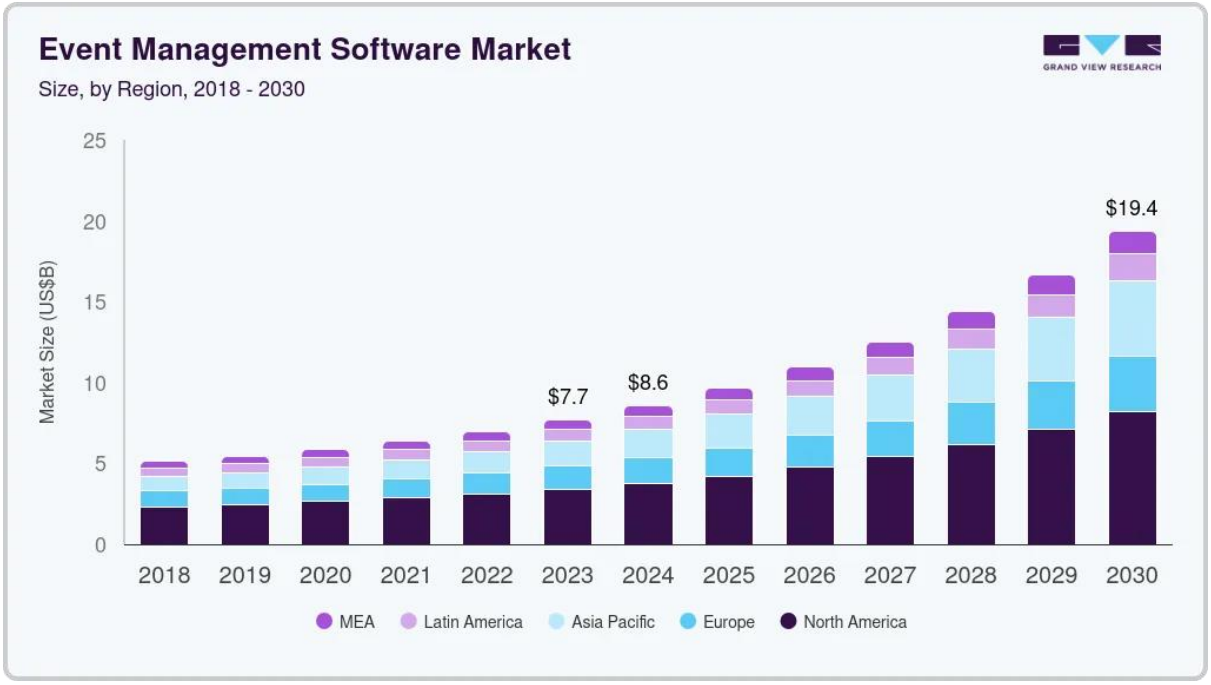


Рисунок 1.1 – Динаміка росту ринку платформ для організації подій 2018-2030

Сучасні тренди включають персоналізацію контенту через алгоритми машинного навчання, інтеграцію з екосистемою міських сервісів, гейміфікацію та програми лояльності, а також інтеграцію волонтерського функціоналу. Детальний аналіз трендів та статистичні дані наведено у **Додатку Е**.

Український ринок має специфічні виклики:

Фрагментація інформації – відсутність єдиної платформи змушує користувачів шукати події через численні Telegram-канали, Instagram-сторінки та Facebook-групи;

Обмежені платіжні можливості – не всі міжнародні платформи підтримують локальні платіжні системи;

Низька цифрова грамотність організаторів – багато невеликих подій організовуються без використання спеціалізованих інструментів;

Відсутність екосистеми – немає інтеграції між платформами подій та іншими міськими сервісами.

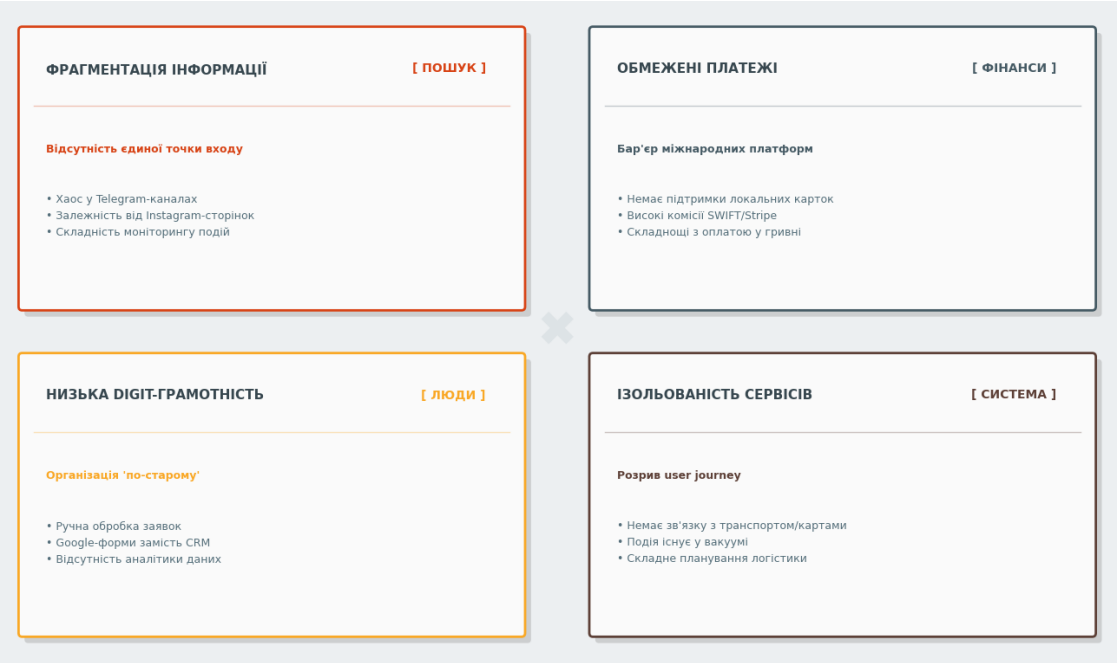


Рисунок 1.4 – Карта проблем українського ринку організації подій

1.2 Аналіз існуючих рішень та конкурентів

На сьогоднішній день існує низка платформ для організації та управління подіями, як міжнародних (Eventbrite, Meetup), так і локальних українських (Concert.ua, Karabas.com). Детальний порівняльний

аналіз цих платформ, включаючи їх функціональні можливості, сильні та слабкі сторони, наведено у **Додатку В**.

Проведений аналіз виявив ключові обмеження існуючих рішень. Міжнародні платформи характеризуються високою комісією за продаж квитків та недостатньою локалізацією для українського ринку. Водночас як міжнародні, так і локальні сервіси не пропонують функцій персоналізованих рекомендацій на основі вподобань користувача, мають обмежені можливості для залучення волонтерів та слабкі інструменти для побудови спільнот навколо подій.

Ці недоліки обґрунтовують доцільність розробки нової платформи Urban Gather, яка враховує специфіку українського ринку та пропонує розширений функціонал для організаторів і відвідувачів.

Порівняльний аналіз функціональності

Проведений порівняльний аналіз ключових функцій існуючих платформ виявив наступні закономірності:

Таблиця 1.1 – Порівняльний аналіз платформ організації подій

Платформа	Тип	Аудиторія	Комісія	Функції	Спеціалізація
Eventbrite	Міжнародна	Комерційні	3.7% + 59₽	Продаж, реєстр.	Конференції
Facebook Events	Соціальна	Особисті	Безкоштовно	Запрошення	Зустрічі
Meetup	Соціальна	Спільноти	$14.99/міс	Групи, зустрічі	Нетворкінг
Concert.ua	Локальна (UA)	Розваги	5-15%	Продаж квитків	Концерти, театр
Karabas.com	Локальна (UA)	Сімейні	5-10%	Каталог, бронь	Дитячі події
Eventmate	Ua / Месенджери	Воркшопи	6%	Квитки в Telegram	Спільноти

Таблиця 1.2 – Матриця функціональності конкурентів

Функція	Eventbrite	Facebook Events	Meetup	Concert.ua	Karabas.com	Eventmate	Urban Gather
Для відвідувачів							
Пошук подій	Так (потужний)	Так (базовий)	Так (за групами)	Так (каталог)	Так (категорії)	Так (тематичний)	Так (ML-рекомендації)
Персоналізація	Так (базові)	Ні	Ні	Ні	Ні	Ні	Так (ML-алгоритми)
Купівля квитків	Так	Ні	Ні	Так	Так	Так	Так
Календар подій	Так	Так	Так	Ні	Ні	Так	Так
Соціальні функції	Частково (обмежені)	Так (потужні)	Так (спільноти)	Ні	Ні	Частково (базові)	Так
Моб. застосунок	Так	Так	Так	Ні	Ні	Так	Так (PWA)
Відгуки та оцінки	Так	Частково (коментарі)	Так	Ні	Ні	Так	Так
Для організаторів							
Створення подій	Так (складне)	Так (просте)	Так (групи)	Так (базове)	Так (базове)	Так (середнє)	Так (інтуїтивне)
Управління квитками	Так	Ні	Ні	Так	Так	Так	Так
Аналітика	Так	Частково (базова)	Так	Ні	Ні	Частково (базова)	Так
Маркетинг	Так	Так (реклама FB)	Частково	Ні	Ні	Частково	Так
Інтеграції	Так	Так (FB еко)	Частково	Ні	Ні	Частково	Так
CRM система	Так	Ні	Ні	Ні	Ні	Ні	Так
Для волонтерів							
Реєстрація	Ні	Ні	Ні	Ні	Ні	Ні	Так
Система завдань	Ні	Ні	Ні	Ні	Ні	Ні	Так
Гейміфікація	Ні	Ні	Ні	Ні	Ні	Ні	Так
Нагороди	Ні	Ні	Ні	Ні	Ні	Ні	Так
Планування змін	Ні	Ні	Ні	Ні	Ні	Ні	Так
Технічні можливості							
API	Так	Так	Так	Ні	Ні	Частково	Так
Локалізація	Так	Так	Так	Частково (UA)	Частково (UA)	Частково (UA)	Так
Безпека даних	Так	Так	Так	Частково	Частково	Частково	Так
Масштабування	Так	Так	Так	Частково	Частково	Частково	Так

Функції для організаторів: Міжнародні платформи пропонують потужні інструменти, але вони складні для малих організаторів. Локальні рішення мають базовий функціонал без можливостей просування.

Волонтерський функціонал: Жодна з проаналізованих платформ не має інтегрованої системи для роботи з волонтерами.

Персоналізація: Тільки Eventbrite має базові рекомендації, інші платформи покладаються на ручний пошук.

Соціальні функції: Meetup лідирує в створенні спільнот, але не підходить для комерційних подій. Інші платформи ігнорують соціальний аспект.

Локалізація: Українські платформи краще адаптовані до місцевих реалій, але програють у функціональності міжнародним аналогам.

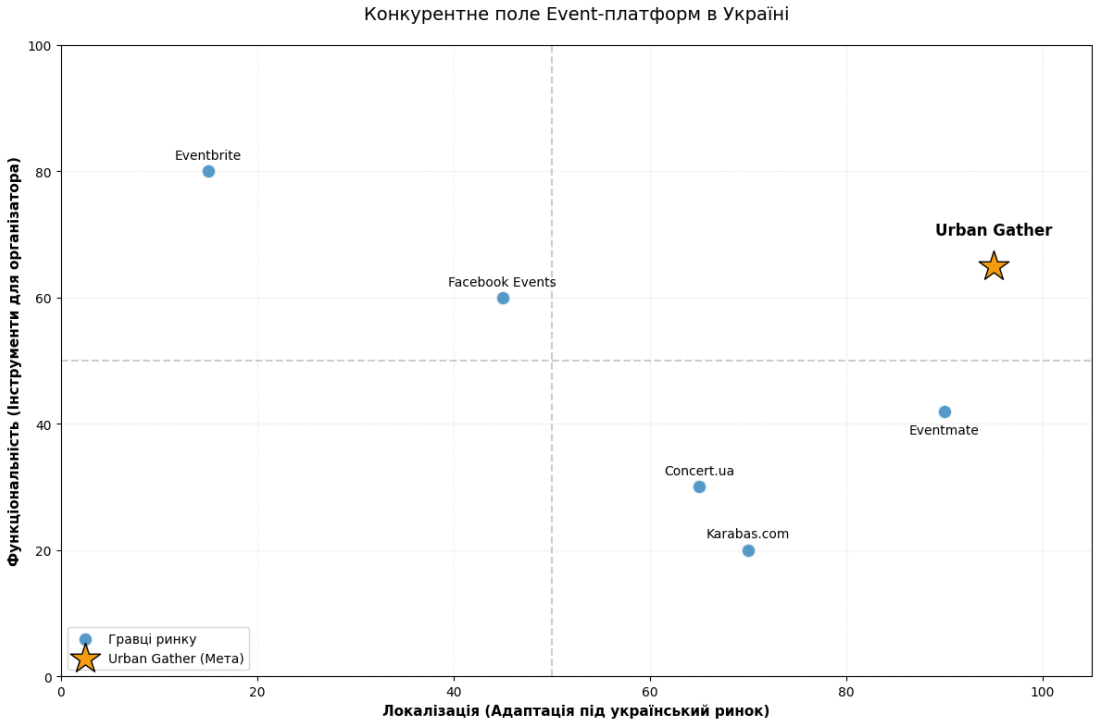


Рисунок 1.5 – Конкурентне позиціонування платформ

1.3 Визначення проблем та потреб користувачів

Детальний аналіз цільової аудиторії Urban Gather виявив три основні групи користувачів з різними потребами та проблемами у сфері управління міськими подіями. **Дослідження користувацького досвіду** проведено через інтерв'ю з 12 респондентами, онлайн-опитування 42 учасників та аналіз існуючих платформ для розуміння поточних больових точок та очікувань користувачів.

Відвідувачі подій (20-45 років) стикаються з ключовими проблемами:

- Розпорошеність інформації: 78% респондентів відчувають труднощі з пошуком релевантних подій через необхідність перевірки множинних джерел

- Відсутність персоналізації: 65% незадоволені загальними рекомендаціями без урахування особистих переваг

- Обмежені соціальні можливості: 52% висловили бажання знайомитися з людьми зі схожими інтересами на подіях

Організатори подій (від індивідуальних ентузіастів до культурних організацій) мають наступні виклики:

- Високі комісії платформ: 84% незадоволені рівнем комісій, особливо для невеликих подій;

- Неефективне просування: 91% відчувають складність досягнення релевантної аудиторії через примітивні алгоритми;

- Обмежена аналітика: 76% потребують детальних інсайтів про аудиторію для стратегічного планування;

- Проблеми з волонтерами: 68% мають труднощі з пошуком та координацією волонтерів.

Волонтери (18-35 років) зазнають наступних проблем:

- Складність пошуку можливостей: 87% не можуть знайти релевантні волонтерські позиції;

- Недостатня мотивація: 73% потребують систем визнання та відстеження особистого внеску;

- Обмежений особистий розвиток: 69% прагнуть отримувати нові навички через волонтерство;

- Слабка соціальна взаємодія: 58% хочуть створювати спільноту з іншими волонтерами.

1.4 Постановка задачі та формулювання вимог

На основі проведеного аналізу існуючих рішень та потреб користувачів сформульовано комплексну задачу розробки платформи Urban Gather, що має об'єднати всі аспекти управління міськими подіями в єдиній екосистемі. Основна мета проєкту полягає у створенні технологічної платформи, яка забезпечить ефективну взаємодію між відвідувачами, організаторами подій та волонтерами

через персоналізовані рекомендації, зручні інструменти управління та соціальні функції.

1.4.1 Постановка основної задачі

Центральна проблема, що вирішується платформою Urban Gather, полягає у фрагментації екосистеми міських подій, де відвідувачі не можуть ефективно знаходити релевантні заходи, організатори стикаються з високими витратами на просування, а волонтери не мають централізованих можливостей для участі у житті міста. Поточні рішення на ринку або зосереджені на окремих аспектах (лише продаж квитків або лише соціальні функції), або не враховують специфіку українського ринку та потреби місцевої аудиторії.

Комплексність задачі вимагає створення багатофункціональної платформи, що одночасно забезпечує персоналізований досвід для відвідувачів через систему рекомендацій, ефективні інструменти для організаторів з детальною аналітикою та мотиваційну систему для волонтерів з елементами геймифікації. Платформа має інтегрувати сучасні технології машинного навчання для рекомендацій, мобільно-орієнтований дизайн для зручності використання та надійну платіжну систему для обробки фінансових транзакцій.

Стратегічна мета розробки включає не лише вирішення поточних проблем користувачів, але й створення умов для розвитку культурної екосистеми міста через підвищення доступності інформації про події, зниження бар'єрів для організації заходів та стимулювання волонтерської активності громадян.

1.4.2 Функціональні вимоги до системи

Підсистема управління користувачами має забезпечувати реєстрацію та автентифікацію з підтримкою різних ролей (відвідувач, організатор, волонтер, адміністратор), управління профілями з налаштуваннями приватності та систему рекомендацій на основі користувацьких переваг. Кожен тип користувача отримує персоналізований інтерфейс з функціями, релевантними для його потреб та цілей використання платформи.

Система управління подіями повинна підтримувати повний життєвий цикл події від створення до завершення, включаючи:

- створення та редагування подій з мультимедійним контентом;
- гнучку систему квитків з різними типами та цінами;
- інтеграцію з картографічними сервісами для локацій;
- систему реєстрації та управління учасниками.

Рекомендаційна система має реалізовувати алгоритми персоналізації на основі історії користувачів, їхніх переваг, геолокації та поведінкових патернів. Система повинна забезпечувати релевантні рекомендації як для нових користувачів (через демографічні дані та популярні події), так і для досвідчених (через аналіз попередніх взаємодій та схожість з іншими користувачами) [4].

Волонтерський модуль має включати каталог можливостей з фільтрацією за навичками та інтересами, систему подання заявок та відбору, інструменти координації діяльності та геймифіковану систему досягнень з публічним визнанням внеску волонтерів у розвиток спільноти.

1.4.3 Нефункціональні вимоги

Вимоги до продуктивності передбачають підтримку до 1000 одночасних користувачів з часом відповіді API менше 500 мілісекунд для стандартних операцій та менше 2 секунд для складних аналітичних запитів. Час завантаження сторінок не повинен перевищувати 3 секунди на стандартному мобільному з'єднанні 3G.

Масштабованість архітектури має дозволяти горизонтальне розширення системи для обробки зростаючого навантаження, модульність компонентів для незалежного розвитку функцій та можливість інтеграції з зовнішніми сервісами без порушення роботи основних функцій.

Висновки до розділу 1

Проведений аналіз предметної області виявив значний потенціал для створення інноваційної платформи Urban Gather. Український ринок має

специфічні особливості, які не враховують існуючі міжнародні рішення, водночас локальні платформи не мають достатнього функціоналу для задоволення потреб усіх груп користувачів.

Ключові висновки:

1. Ринок платформ для організації подій активно зростає (11.8% щорічно), що створює сприятливі умови для запуску нового продукту.

2. Існуючі рішення не забезпечують комплексного підходу до роботи з трьома ключовими групами користувачів: відвідувачами, організаторами та волонтерами.

3. Український ринок має незаповнені ніші в сфері персоналізованих рекомендацій, інтегрованої системи волонтерства та соціальних функцій.

4. Визначені функціональні та нефункціональні вимоги створюють основу для розробки технічної архітектури платформи Urban Gather.

Результати аналізу формують підґрунтя для проектування архітектури системи, що розглядається в наступному розділі.

Стрімкий розвиток інформаційних технологій суттєво трансформував повсякденне життя, зумовивши розширення спектру платформ — мобільних, десктопних, веб-орієнтованих та хмарних рішень.

Для сучасного користувача критично важливим є безперебійний доступ до інформації незалежно від пристрою. Комунікація, електронна комерція та фінансові операції здійснюються переважно через мобільні пристрої, тоді як адміністрування даних залишається зручнішим на стаціонарних системах(ПК та ноутбуки).

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ВЕБ-ПЛАТФОРМИ URBAN GATHER

У цьому розділі представлено проектування архітектури веб-платформи Urban Gather на основі аналізу вимог і потреб користувачів, проведеного в першому розділі. Розділ містить визначення функціональних компонентів системи, проектування структури бази даних, архітектуру взаємодії клієнт-сервер, рекомендації щодо системи персоналізації та інтеграцію з зовнішніми сервісами відповідно до принципів системного аналізу та проектування програмних систем.

2.1 Визначення функціональних компонентів системи

2.1.1 Декомпозиція системи на модулі

Відповідно до методології структурного аналізу та принципів модульного проектування, система Urban Gather декомпонована на функціональні модулі з високим рівнем зв'язності всередині модулів та низьким рівнем зв'язування між модулями.

Система включає наступні основні функціональні модулі:

Модуль автентифікації та управління користувачами забезпечує повний життєвий цикл користувачів системи, включаючи процеси реєстрації, автентифікації через власну реалізацію сесійного механізму, управління профілями та налаштуваннями користувачів. Модуль підтримує багаторольову систему з чотирма рівнями доступу: відвідувач (VISITOR), організатор (ORGANIZER), волонтер (VOLUNTEER) та адміністратор (ADMIN).

Модуль управління подіями реалізує основну бізнес-логіку системи, забезпечуючи створення, редагування, публікацію та управління життєвим циклом міських подій. Модуль включає систему станів події (чернетка,

опублікована, скасована), управління вмістимістю, категоризацію та зв'язок з квитковою системою.

Модуль пошуку та фільтрації відповідає за реалізацію алгоритмів пошуку з підтримкою повнотекстового індексування, багатокритеріальної фільтрації за категоріями, локаціями, датами, ціновим діапазоном та іншими параметрами, а також інтелектуальне ранжування результатів.

Модуль персоналізованих рекомендацій використовує гібридні алгоритми машинного навчання для генерації індивідуальних рекомендацій подій на основі аналізу переваг користувача, історії взаємодій, соціального графу та контекстних факторів. Включає підсистеми збору поведінкових даних та багатофакторного оцінювання релевантності.

Модуль квитків та реєстрацій управляє процесами реєстрації користувачів на події, включаючи багаторівневу систему квитків (безкоштовні, платні, VIP), контроль доступності місць, генерацію унікальних кодів квитків та інтеграцію з платіжними системами.

Модуль управління локаціями забезпечує роботу з геопросторовими даними, включаючи зберігання географічних координат, адресної інформації, характеристик місць проведення, а також інтеграцію з картографічними сервісами для візуалізації та навігації.

Модуль волонтерства та гейміфікації реалізує спеціалізовану функціональність для координації волонтерської діяльності, включаючи систему подачі заявок, розподіл завдань, відстеження відпрацьованих годин, нарахування балів та систему досягнень для мотивації учасників.

Модуль адміністрування надає інструменти для управління платформою, включаючи модерацію контенту, управління користувачами та їх ролями, перегляд системної аналітики, налаштування параметрів системи та моніторинг активності [5].

2.1.2 Визначення взаємозв'язків між компонентами

Архітектура взаємодії між модулями базується на принципах слабкого зв'язування та високої когезії, що забезпечує можливість незалежного розвитку та тестування окремих компонентів системи [6].

Типи взаємозв'язків:

- **Синхронні**: валідація автентифікації, перевірка доступності місць
- **Асинхронні**: оновлення рекомендацій, нарахування балів гейміфікації
- **Подійно-орієнтовані**: реагування на створення подій, реєстрації

Система побудована за шаровою архітектурою: презентаційний, прикладний, доменний та інфраструктурний шари.

Шарова архітектура системи:

1. **Презентаційний шар (Presentation Layer)** - користувацький інтерфейс та компоненти візуалізації
2. **Прикладний шар (Application Layer)** - координація бізнес-процесів та обробка запитів
3. **Доменний шар (Domain Layer)** - бізнес-логіка та правила предметної області
4. **Інфраструктурний шар (Infrastructure Layer)** - робота з базами даних, зовнішніми API та файловими сховищами [7].

На рисунку 2.1 можемо наглядно подивитись на схему роботи шарів в гексагональній архітектурі.

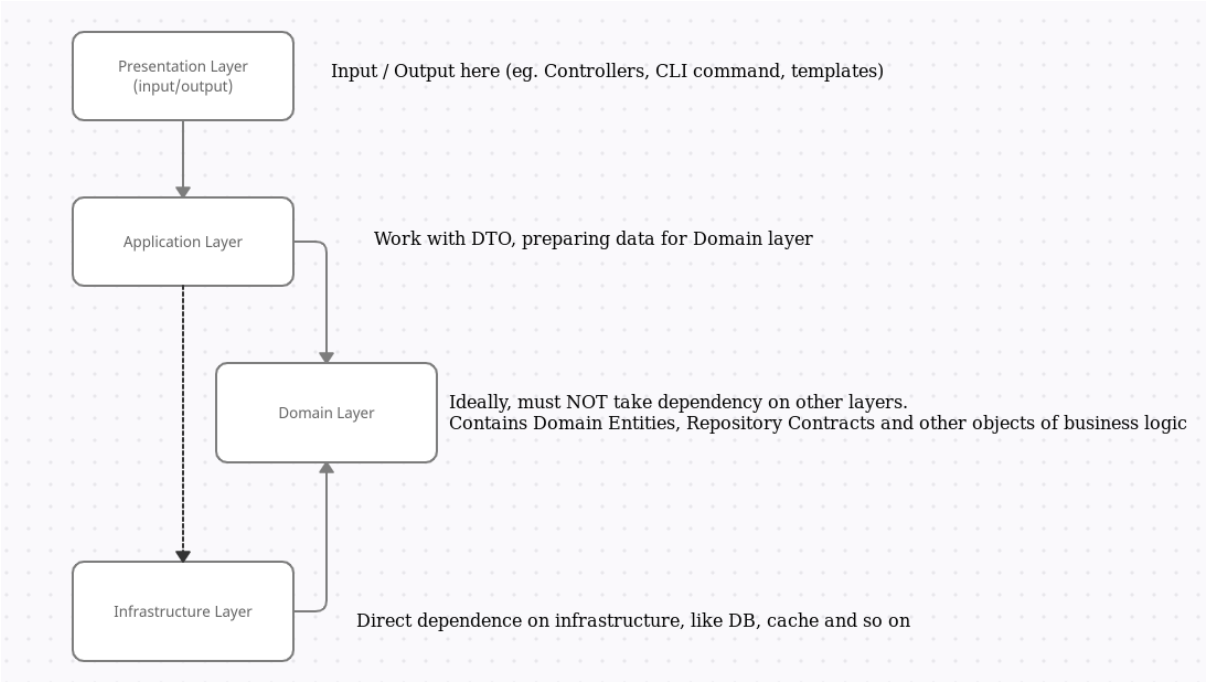


Рисунок 2.1 – Діаграма взаємодії між функціональними модулями

2.2 Проектування бази даних та моделі даних

Концептуальна модель даних Urban Gather розроблена на основі принципів Domain-Driven Design та відображає основні сутності предметної області організації міських подій, їх атрибути та взаємозв'язки [8].

Основні сутності системи:

Сутність **User** є базовою для системи автентифікації та авторизації. Включає такі поля ідентифікації (id, email, username), безпеки (password — в хешованому вигляді за допомогою bcrypt), рольової моделі (Role: VISITOR, ORGANIZER, VOLUNTEER, ADMIN), гейміфікації (points, volunteerPoints, visitorLevel, volunteerLevel, streak) та профілю (bio, avatar, phone, telegramNickname, socialLinks). Зв'язки реалізовано через відношення "один-до-багатьох" із сутностями Event, Registration, Achievement та "багато-до-багатьох" — з подіями через збереження та волонтерство.

Сутність **Event** є центральною в нашій доменній моделі та містить основні атрибути події (title, description, date, endDate), налаштування важливі для участі(maxAttendees, currentAttendees, maxVolunteers, capacity, enableWaitingList), фінансові показники та параметри(price, paymentRequired,

priceInKopecks, refundPolicy) та метадані (status: UPCOMING/ONGOING/CANCELLED/COMPLETED, type: MUSIC/SPORTS/EDUCATION тощо, tags, schedule). Зовнішні ключі пов'язують подію з організатором (organizerId), локацією (locationId) та медіафайлами (bannerAssetId, thumbnailAssetId).

Сутність **Location** описує місця проведення та включає геодані (координати у форматі JSON {lat, lng}), характеристики (name, address, description, facilities, accessibility) та модераційні атрибути (isPublic, isApproved). Зв'язок з Asset забезпечує зберігання фотогалереї локації.

Сутність Registration фіксує участь користувачів у подіях. Статусна модель (RegistrationStatus: PENDING, CONFIRMED, CANCELLED, WAITLISTED) дозволяє керувати чергою очікування через поле waitlistPosition. Унікальний індекс [userId, eventId] запобігає повторним реєстраціям.

Проектування бази даних платформи базувалося на ретельному аналізі предметної області та виявлених функціональних вимог. В результаті концептуального моделювання було ідентифіковано ключові сутності системи: користувачі (User), події (Event), локації (Location), реєстрації (Registration), категорії (Category) та їх взаємозв'язки.

На рисунку 2.2 представлено ER-діаграму, що відображає логічну структуру бази даних. Схема включає 27 взаємопов'язаних таблиць, які забезпечують цілісність даних та підтримку всіх бізнес-процесів платформи.

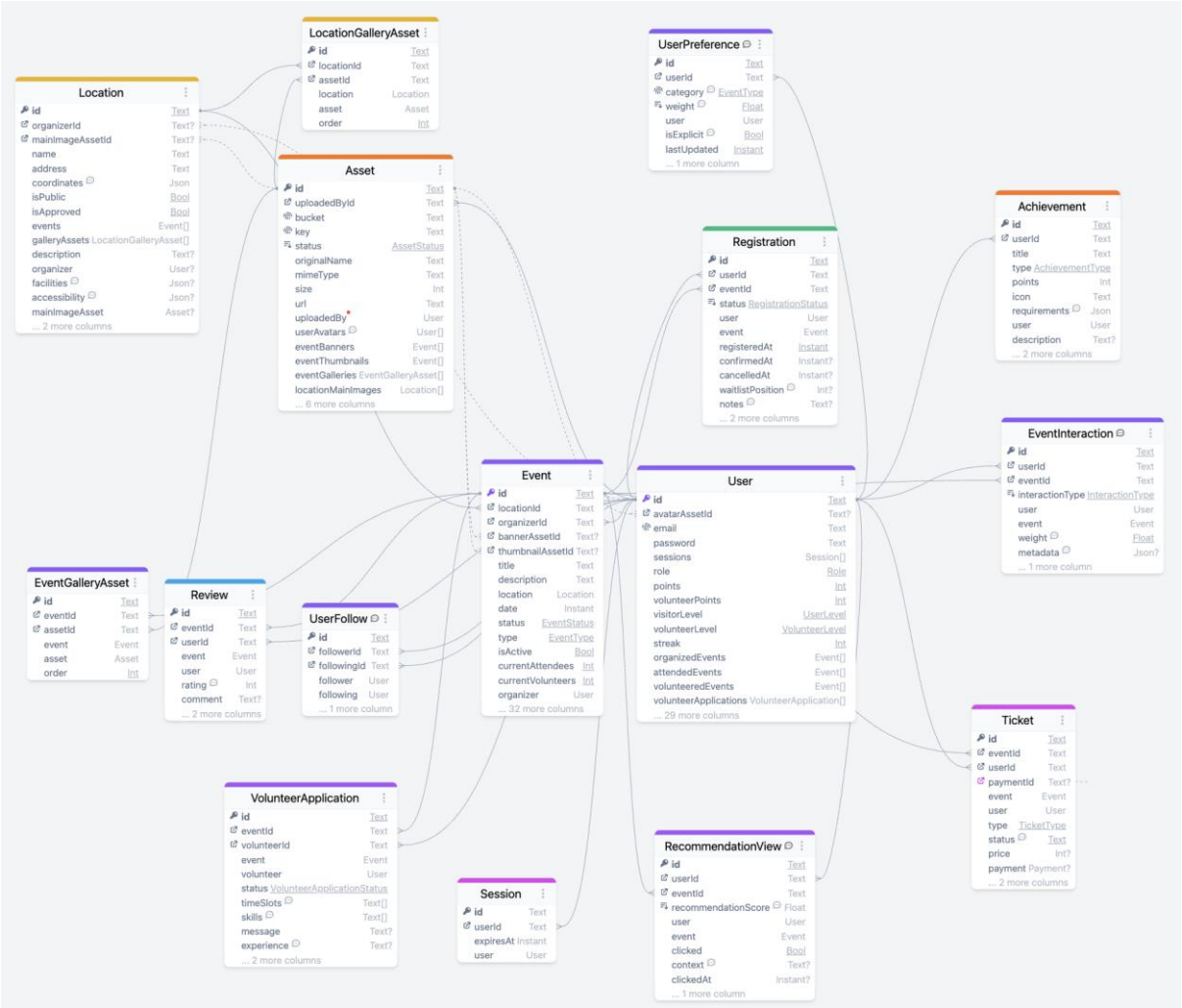


Рисунок 2.2 – Концептуальна ER-діаграма предметної області Urban Gather

Для забезпечення високої продуктивності при очікуваних навантаженнях було застосовано комплекс оптимізаційних стратегій: денормалізацію критичних шляхів запитів, композитне індексування для частих операцій пошуку та часткові індекси для фільтрації за статусами [9].

Повна специфікація структури бази даних, включаючи детальний опис усіх таблиць, їх атрибутів, типів даних, обмежень цілісності та стратегій індексування, наведена у **Додатку Б**. Винесення цього матеріалу обумовлено його довідковим характером: специфікація бази даних є технічною документацією, необхідною для супроводу та розвитку системи, проте надмірно деталізованою для основного тексту наукової роботи, де важливішим є обґрунтування прийнятих архітектурних рішень.

2.3. Архітектура клієнт-серверної взаємодії

Архітектура клієнт-серверної взаємодії платформи Urban Gather базується на сучасних веб-технологіях та патернах, що забезпечують оптимальний баланс між продуктивністю, масштабованістю та досвідом розробки. Система побудована з використанням гібридного підходу, що поєднує переваги серверного рендерингу, клієнтської інтерактивності та ефективної комунікації між компонентами.

Гібридна архітектура на базі Next.js App Router

Для побудови платформи Urban Gather обрано інноваційну гібридну архітектуру, що базується на сучасному фреймворку Next.js з використанням нового App Router. Цей підхід органічно поєднує кілька потужних технологій для створення оптимального користувацького досвіду.

Основою архітектури є серверний рендеринг, який забезпечує швидке завантаження сторінок та повну видимість контенту для пошукових систем. React Server Components дозволяють виконувати складні обчислення безпосередньо на сервері, зменшуючи навантаження на пристрої користувачів [10].

Водночас Client Components забезпечують багату інтерактивність там, де це необхідно – у формах, динамічних елементах та реального часу оновленнях. Унікальною особливістю є використання Server Actions, які дозволяють виконувати операції з базою даних прямо з компонентів без створення додаткових API endpoints [11].

Такий вибір архітектури продиктований кількома ключовими факторами. З точки зору продуктивності, Server Components значно зменшують розмір JavaScript коду, що завантажується в браузер, та прискорюють початкове відображення сторінки [12]. Для SEO-оптимізації серверний рендеринг гарантує, що всі пошукові системи можуть повноцінно проіндексувати контент платформи. Досвід розробки покращується завдяки єдиній кодовій базі з повною типобезпекою та можливості прямого доступу до даних без складних API шарів

[13]. Нарешті, модульна структура забезпечує легке масштабування та додавання нової функціональності без перебудови існуючої системи.

Структура клієнтської частини

Клієнтська частина платформи організована за принципом функціональної архітектури, де кожен компонент має чітко визначену роль та місце в системі. Основною папкою є `src/app`, яка містить всі сторінки додатку, організовані згідно з новим App Router підходом Next.js. Публічні сторінки автентифікації, такі як реєстрація та вхід, розміщені в окремій групі `(auth)`, що дозволяє застосовувати спеціальні налаштування лише до цих сторінок. Захищені сторінки, доступні лише авторизованим користувачам, знаходяться в групі `(authenticated)`, що автоматично перевіряє права доступу. Адміністративна панель виділена в окрему секцію `admin` з додатковими рівнями захисту.

Функціональні модулі системи зосереджені в папці `features`, де кожен напрямок діяльності має власну область. Модуль автентифікації містить всю логіку роботи з користувацькими акаунтами, управління подіями зібране в одному місці для зручності підтримки, система рекомендацій ізольована в окремий модуль, а волонтерські функції також мають власний простір. Спільні компоненти інтерфейсу, які використовуються по всій платформі, розміщені в папці `components`, тоді як утиліти та допоміжні сервіси знаходяться в папці `lib`.

Одним з ключових рішень в архітектурі є розумний розподіл між серверними та клієнтськими компонентами. За замовчуванням всі компоненти виконуються на сервері, що забезпечує оптимальну продуктивність та SEO. До серверних компонентів належать сторінки та їх макети, компоненти, які потребують прямого доступу до бази даних, статичні елементи інтерфейсу без інтерактивності, та компоненти, що обробляють великі обсяги даних.

Клієнтські компоненти використовуються лише там, де це дійсно необхідно, та позначаються спеціальною директивою "use client". До них належать всі форми та інтерактивні елементи, які потребують миттєвої реакції на дії користувача. Компоненти, що використовують React hooks для управління станом або ефектами, також виконуються на клієнті. Елементи з обробниками

подій, анімаціями та оновленнями в реальному часі потребують клієнтського виконання для забезпечення плавної взаємодії з користувачем.

Взаємодія між клієнтською та серверною частинами платформи реалізована через комбінований підхід, що поєднує інноваційні Server Actions Next.js для внутрішніх операцій та традиційний RESTful API для зовнішніх інтеграцій. Server Actions дозволяють виконувати серверний код безпосередньо з компонентів без створення окремих API endpoints, що значно спрощує архітектуру та забезпечує повну типобезпеку [14].

Управління станом базується на багаторівневій стратегії: React Query (TanStack Query) для серверного стану з автоматичним кешуванням та ревалідацією, Zustand для глобального клієнтського стану, та React Context для локального стану функціональних областей. URL-стан через бібліотеку nuqs забезпечує можливість збереження фільтрів пошуку як закладок [15].

Безпека системи забезпечується на кількох рівнях: валідація вхідних даних через схеми Zod, обмеження на рівні бази даних Prisma, захист від XSS та CSRF атак, rate limiting для чутливих операцій.

Детальний опис проектування API endpoints, специфікації Server Actions, патернів взаємодії клієнт-сервер, стратегій оптимізації продуктивності (code splitting, lazy loading, кешування) та механізмів безпеки наведено у **Додатку Ж**. Винесення цього матеріалу обумовлено його технічно-імплементаційним характером: специфікації API та патерни управління станом є важливими для розробників системи, проте надмірно деталізованими для основного тексту, де акцент зроблено на архітектурних рішеннях та обґрунтуванні вибору технологій.

2.4. Проектування системи персоналізованих рекомендацій

Система персоналізованих рекомендацій є ключовим компонентом платформи Urban Gather, що забезпечує користувачам релевантні пропозиції подій на основі їх вподобань, поведінки та контексту. Проектування системи базується на гібридному підході, що поєднує методи контентної фільтрації, елементи колаборативної фільтрації та контекстні фактори.

2.4.1 Архітектура та алгоритми рекомендаційної системи

Загальна архітектура системи рекомендацій

Рекомендаційна система Urban Gather спроектована як інтелектуальний механізм, що вивчає поведінку кожного користувача та пропонує йому найбільш релевантні події. Архітектура побудована навколо п'яти ключових компонентів, кожен з яких відповідає за конкретний аспект роботи системи.

Сервіс збору даних працює як невидимий спостерігач, який фіксує всі дії користувачів на платформі. Він відстежує, які події люди переглядають, на які реєструються, що зберігають у закладках та з чим взаємодіють. Ця інформація стає основою для розуміння індивідуальних вподобань кожної людини [16].

Сервіс побудови профілів аналізує зібрані дані та створює унікальний "цифровий портрет" кожного користувача. Він визначає улюблені категорії подій, часові вподобання, географічні преференції та інші характеристики, що допомагають зрозуміти, що саме цікавить конкретну людину.

Рекомендаційний движок є мозком всієї системи, який використовує профіль користувача для пошуку найкращих подій. Він оцінює кожну доступну подію з точки зору її відповідності інтересам користувача та формує персональний список рекомендацій.

Сервіс кешування забезпечує швидкість роботи системи, зберігаючи готові рекомендації для миттєвого доступу. Це дозволяє користувачам отримувати персональні пропозиції без затримок.

Аналітичний сервіс постійно відстежує, наскільки точними виявилися рекомендації, та допомагає покращувати роботу всієї системи на основі реальної поведінки користувачів.

Гібридний алгоритм рекомендацій

Система використовує багатофакторний алгоритм оцінювання, що враховує різні аспекти релевантності події для користувача. Загальна оцінка розраховується як зважена сума факторів:

Score = Σ(wi × fi)

Де wi - вага фактора, fi - нормалізоване значення фактора (0-1).

Основні фактори оцінювання:

1. **Відповідність категоріям (вага 3.0)**
   - Порівняння категорії події з явними перевагами користувача.
   - Врахування ваги переваги для кожної категорії.
   - Аналіз історичних взаємодій з подіями цієї категорії.
2. **Схожість тегів (вага 2.0)**
   - Порівняння тегів події з тегами подій, що сподобались користувачу.
   - Використання TF-IDF для визначення важливості тегів.
   - Косинусна міра подібності між векторами тегів.
3. **Географічна близькість (вага 1.5)**
   - Розрахунок відстані між локацією користувача та місцем події.
   - Експоненційне зменшення оцінки з відстанню.
   - Максимальний радіус пошуку - 25 км.
4. **Популярність події (вага 1.0)**
   - Нормалізована кількість реєстрацій.
   - Кількість збережень події іншими користувачами.
   - Загальна кількість позитивних взаємодій.
5. **Історія взаємодій (вага 2.5)**
   - Аналіз попередніх дій користувача з подібними подіями.
   - Врахування типу взаємодії (перегляд - 0.1, збереження - 0.8, реєстрація - 1.0).
   - Часове згасання важливості старих взаємодій.
6. **Організатор події (вага 1.0)**
   - Перевірка підписки користувача на організатора.
   - Історія відвідування подій цього організатора.
   - Рейтинг організатора в системі.
7. **Актуальність за часом (вага 0.5)**
   - Пріоритет подіям, що відбудуться найближчим часом.
   - Врахування дня тижня та часу проведення.
   - Фільтрація подій, що вже розпочалися.

2.4.2 Збір та обробка даних користувачів

Типи даних для побудови профілю користувача

Система збирає чотири типи даних для персоналізації:

Експліцитні дані – свідомий вибір користувача: обрані категорії при реєстрації, налаштування переваг, оцінки та відгуки про події, збереження в улюблене, підписки на організаторів [17].

Імпліцитні дані – автоматичне відстеження поведінки: історія переглядів, час на сторінці події, послідовність переходів, кліки по елементах, патерни пошукових запитів [18].

Контекстні дані – обставини користування: геолокація, час та день активності, тип пристрою, джерело переходу на подію.

Соціальні дані – мережеві взаємодії: підписки на організаторів, спільні події з друзями, поширення в соцмережах, запрошення інших користувачів.

Модель взаємодій користувача з подією

Кожна дія користувача на платформі ретельно документується системою для подальшого аналізу та покращення рекомендацій. Система фіксує шість основних типів взаємодії: простий перегляд сторінки події, збереження в закладки, реєстрацію на подію, фактичне відвідування заходу, поширення інформації про подію та залишення відгуку після відвідування.

Для кожної взаємодії зберігається точна часова мітка, що дозволяє аналізувати поведінкові патерни та виявляти зміни в інтересах користувача з часом. Контекст взаємодії показує, як саме користувач потрапив на сторінку події – через головну сторінку, пошук, рекомендації чи соціальні мережі. Тривалість взаємодії, особливо для переглядів, вказує на рівень зацікавленості. Результат взаємодії фіксує, чи призвела дія до досягнення цільової мети, що допомагає оцінити ефективність інтерфейсу та рекомендацій.

Агрегація та нормалізація даних

Перетворення сирих даних про поведінку користувачів у корисну інформацію для рекомендацій відбувається у чотири етапи.

Очищення даних усуває шум та некоректну інформацію. Система

автоматично видаляє аномальні взаємодії – надто короткі перегляди сторінок, що свідчать про випадкові кліки або технічні проблеми. Фільтрується трафік від ботів та сканерів, усуваються дублікати взаємодій через технічні збої або повторні завантаження.

Агрегація об'єднує дані користувача в узагальнені показники. Система підраховує взаємодії з подіями кожної категорії, аналізує часові патерни активності – найактивніші дні тижня та години доби, розраховує середні показники різних типів взаємодій для розуміння загальної поведінкової моделі [19].

Визначення вподобань конкретизує інтереси користувача. Для кожної категорії розраховується вага інтересу, ідентифікуються найчастіше вживані теги в подіях, що привернули увагу, визначаються географічні преференції – улюблені райони міста та максимальна комфортна відстань до події.

Нормалізація забезпечує коректну роботу алгоритму рекомендацій. Всі оцінки приводяться до діапазону 0-1 для порівняння різних факторів, ваги типів взаємодій стандартизуються для справедливого врахування кожної поведінки, проводиться балансування щоб жоден фактор не домінував [20].

2.4.3. Оптимізація продуктивності та якості рекомендацій

Стратегії кешування

Для швидкості роботи рекомендаційної системи платформа використовує багаторівневе кешування.

Кеш рекомендацій зберігає готові списки за алгоритмом LRU з обмеженням 1000 записів. Час життя – 5 хвилин, при зміні профілю відбувається негайна інвалідація.

Кеш профілів містить агреговані дані про вподобання у компактному вигляді. Оновлення раз на годину або при значних змінах у поведінці, що дозволяє швидко розраховувати рекомендації без повторного аналізу історії.

Кеш популярності зберігає показники для всіх активних подій з оновленням кожні 15 хвилин для відображення поточних трендів.

Метрики оцінки якості

Click-Through Rate – відсоток рекомендацій, що призвели до переходу. Аналізується окремо для різних контекстів показу [21].

Conversion Rate – відсоток рекомендацій, що призвели до реєстрації на подію. Аналізується за типами подій для оптимізації алгоритму.

Engagement Metrics – середній час перегляду, кількість взаємодій, частота повернення користувачів.

Diversity Score – різноманітність категорій, баланс між популярними та нішевими заходами, географічне розмаїття для запобігання "інформаційним бульбашкам".

Інтеграція з іншими компонентами системи

Система рекомендацій тісно інтегрована з іншими модулями платформи:

- **Модуль пошуку**: використання історії пошуків для уточнення профілю
- **Модуль аналітики**: детальні звіти про ефективність рекомендацій

Проектування системи персоналізованих рекомендацій Urban Gather забезпечує баланс між точністю рекомендацій, продуктивністю системи та простотою підтримки. Гібридний підхід дозволяє ефективно комбінувати різні сигнали для створення релевантних пропозицій, а модульна архітектура забезпечує можливість подальшого вдосконалення алгоритмів без значних змін у системі.

2.5 Використання картографічних сервісів

Інтеграція з картографічними сервісами забезпечує роботу з геопросторовими даними та надає користувачам повноцінні навігаційні функції.

Основу підсистеми для картографії складає Google Maps Platform API який використовується у двох ключових напрямках. Це реалізовано через Maps JavaScript API, що забезпечує роботу інтерактивних карт на веб платформі. Саме цей інтерфейс дозволяє відображати локації подій користувачам безпосередньо на карті, застосовувати кластеризацію маркерів при роботі з великою кількістю об'єктів, а також формувати інтерактивні інформаційні вікна з деталями кожної

події. Додатково підтримується налаштування користувацьких стилів карт для відповідності візуальному брендингу платформи [22].

Другий напрямок представлений Geocoding API, який забезпечує повний цикл роботи з адресною інформацією. Цей сервіс виконує перетворення текстових адрес у географічні координати та підтримує зворотне геокодування для отримання адреси за заданими координатами. Крім того, API забезпечує валідацію та стандартизацію введених користувачами адрес, а також автоматичне визначення адміністративних районів для коректної класифікації локацій подій [23].

Висновки до розділу 2

У другому розділі виконано комплексне проектування архітектури веб-платформи Urban Gather відповідно до сучасних принципів системного аналізу та програмної інженерії.

Здійснено декомпозицію системи на вісім функціональних модулів із забезпеченням високої когезії та слабкого зв'язування: автентифікації, управління подіями, пошуку та фільтрації, персоналізованих рекомендацій, квитків та реєстрацій, управління локаціями, волонтерства з гейміфікацією та адміністрування. Визначено типи взаємозв'язків між компонентами — синхронні, асинхронні та подійно-орієнтовані.

Спроектовано гібридну клієнт-серверну архітектуру на базі Next.js App Router, що поєднує Server Components для оптимальної продуктивності та SEO з Client Components для інтерактивності. Реалізовано двоїстий підхід до серверної логіки через Server Actions для внутрішніх операцій та RESTful API для зовнішніх інтеграцій.

Розроблено архітектуру рекомендаційної системи з гібридним алгоритмом багатофакторного оцінювання, що враховує відповідність категоріям, схожість тегів, географічну близькість, популярність та історію взаємодій. Передбачено збір експліцитних, імпліцитних, контекстних та соціальних даних для побудови профілів користувачів.

Визначено стратегії оптимізації продуктивності, включаючи багаторівневе кешування, code splitting та lazy loading. Спроектовано підсистему безпеки з валідацією через Zod, session-based автентифікацією та захистом від типових веб-вразливостей.

Описано інтеграцію з картографічними сервісами Google Maps Platform для роботи з геопросторовими даними та навігаційними функціями.

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ПЛАТФОРМИ

3.1 Вибір технологічного стеку та інструментів розробки

Технологічний стек Urban Gather обраний з урахуванням потреб українського ринку та необхідності створення надійної платформи управління подіями. Вибір технологій базується на критеріях продуктивності, масштабованості, наявності кваліфікованих розробників в Україні та підтримки локальних сервісів.

3.1.1 Основні технології розробки

Next.js 15 з App Router обрано як основний каркас для створення веб-додатку з серверним рендерингом та оптимізацією швидкості завантаження. Використання React 19 з новими серверними компонентами дозволяє зменшити розмір JavaScript та покращити показники завантаження сторінок. TypeScript забезпечує статичну типізацію, що зменшує кількість помилок під час розробки та спрощує командну роботу [24].

Tailwind CSS використовується для стилізації з власною системою дизайну на базі ShadCN UI компонентів, побудованих на Radix UI основі. Такий підхід забезпечує доступність інтерфейсу, консистентний дизайн та швидку розробку нових компонентів. Lucide React надає векторні іконки з оптимізованим завантаженням.

Управління станом реалізовано через Zustand для глобального стану додатку та TanStack React Query v5 для серверних даних з автоматичним кешуванням. React Hook Form обробляє форми з валідацією через Zod схеми, забезпечуючи безпеку типів між клієнтом та сервером.

3.1.2 Серверна архітектура та база даних

PostgreSQL 15 служить основною базою даних з Prisma ORM для типобезпечного доступу до даних. Схема бази включає понад 25 моделей з

комплексними зв'язками для повного покриття функціоналу платформи. Redis використовується для кешування сесій користувачів та часто запитуваних даних.

Автентифікація реалізована через власну систему на основі сесій з безпечним управлінням токенами доступу. Система підтримує чотири ролі користувачів (відвідувач, організатор, волонтер, адміністратор) з детальними правами доступу до різних функцій платформи.

MinIO забезпечує сховище файлів з сумісністю S3 для зберігання зображень подій та користувацьких файлів. Система автоматично оптимізує зображення та генерує різні розміри для адаптивного дизайну [25].

3.1.3 Спеціалізовані інтеграції

Google Maps API надає інтерактивні карти з можливістю відображення локацій подій, групування маркерів у щільно населених районах та автоматичне заповнення адрес. Інтеграція оптимізована для мінімізації витрат через кешування популярних локацій.

3.1.4 Тестування та забезпечення якості

Система тестування включає Jest для модульних тестів (151 тестових файлів), Playwright для наскрізного тестування (12 тестових файлів) та Cypress для додаткового покриття критичних сценаріїв користувача. Testing Library забезпечує якісне тестування React компонентів з фокусом на поведінку користувачів.

Автоматизовані тести інтегровані в процес розробки для підтримки стабільності кодової бази та швидкого виявлення проблем при внесенні змін.

3.1.5 Розгортання та інфраструктура

Docker з конфігурацією docker-compose забезпечує консистентне середовище розробки та виробництва. Nginx служить зворотним проксі та сервером статичних файлів з автоматичним управлінням SSL сертифікатами через Certbot.

Контейнеризована архітектура включає окремі сервіси для веб-додатку, бази даних PostgreSQL, кешу Redis та файлового сховища MinIO. Така структура дозволяє незалежне масштабування компонентів та спрощує технічне обслуговування системи.

3.2 Розробка серверної частини

Серверна частина організована за доменним принципом з модулями для автентифікації, подій, користувачів та адміністрування.

API структура: `/api/auth/*` для автентифікації, `/api/events/*` для управління подіями, `/api/recommendations/*` для персоналізованих пропозицій. Валідація через Zod, централізована обробка помилок.

Безпека: автентифікація на основі сесій з RSA-підписом токенів, RBAC з чотирма рівнями доступу, rate limiting для захисту від DDoS.

Рекомендації: движок інтегрований через API з кешуванням у Redis. Активні користувачі отримують свіжіші рекомендації, нові – трендові події.

Інтеграції: Google Maps для геокодування, MinIO для файлів з автоматичною генерацією мініатюр.

Детальний опис організації API endpoints, схеми Prisma, механізмів автентифікації та безпеки, а також інтеграції з зовнішніми сервісами наведено у **Додатку Г**.

3.3 Розробка клієнтської частини

Клієнтська частина Urban Gather розроблена з використанням React 19 та Next.js 15 App Router для створення швидкого, адаптивного та доступного інтерфейсу. Архітектура клієнтської частини базується на компонентному підході з широким використанням серверних компонентів для оптимізації продуктивності та покращення індексації пошуковими системами.

Візуальне оформлення інтерфейсу базується на сучасних принципах UX/UI дизайну з використанням компонентної бібліотеки ShadCN UI та утилітарного CSS-фреймворку Tailwind. Головна сторінка включає

персоналізовану стрічку рекомендацій, інтерактивну карту подій та зручну навігацію за категоріями. Детальні знімки екрану всіх основних розділів інтерфейсу, включаючи головну сторінку, каталог подій, профіль користувача, панель організатора та адміністративний інтерфейс, наведено у **Додатку К**.

3.3.1 Компонентна архітектура та система дизайну

Система дизайну побудована на ShadCN UI компонентах з налаштуванням через Tailwind CSS та власні токени дизайну. Компоненти організовані за принципами атомарного дизайну з базовими примітивами інтерфейсу (Button, Input, Modal), складними компонентами (EventCard, UserProfile) та макетами сторінок.

Серверні компоненти React використовуються для статичного контенту та компонентів з великою кількістю даних, що зменшує розмір пакета JavaScript та покращує початкове завантаження сторінки. Клієнтські компоненти застосовуються тільки для інтерактивних елементів (форми, модальні вікна, оновлення в реальному часі), забезпечуючи оптимальний баланс між продуктивністю та користувацьким досвідом.

Доступність забезпечується через примітиви Radix UI з вбудованими атрибутами ARIA, навігацією з клавіатури та підтримкою зчитувачів екрана. Всі інтерактивні елементи мають правильне управління фокусом та контрастність кольорів відповідно до рекомендацій доступності веб-контенту.

Іконографія реалізована через Lucide React з імпортами, що дозволяють виключати невикористані іконки для мінімізації розміру пакета. Векторні іконки автоматично адаптуються до розміру та кольорової теми без додаткових ресурсів.

3.3.2 Управління станом та потік даних

Сховища Zustand управляють глобальним станом додатку з окремими сховищами для автентифікації, гейміфікації та стану інтерфейсу. Сховища

зберігаються в локальному сховищі браузера з автоматичним відновленням для безперервного користувацького досвіду між сесіями.

TanStack React Query v5 обробляє всі взаємодії з сервером з автоматичним кешуванням, фоновими оновленнями та оптимістичним інтерфейсом для покращення відчутної швидкодії. Стратегія анулювання запитів забезпечує актуальність даних при операціях створення, оновлення та видалення.

Управління формами реалізовано через React Hook Form з валідацією Zod для типобезпечної обробки форм. Схеми валідації спільні між клієнтом та сервером забезпечують консистентність та безпеку. Валідація в реальному часі надає негайний зворотний зв'язок користувачам.

Межі помилок створені на різних рівнях додатку для м'якої обробки помилок з резервним інтерфейсом та автоматичним звітуванням про помилки. Користувачі отримують зрозумілі повідомлення про помилки українською мовою з можливістю відновних дій.

3.3.3 Адаптивний дизайн та мобільна оптимізація

Підхід "mobile first" з точками зламу Tailwind CSS забезпечує оптимальний досвід на всіх пристроях. Макет автоматично адаптується від одноколонкового мобільного до багатоколонкового настільного з плавними переходами між точками зламу.

Інтерфейс, орієнтований на дотики включає достатньо великі цілі для дотиків (мінімум 44px), правильну обробку жестів та тактильний зворотний зв'язок на підтримуваних пристроях. Навігація через проведення пальцем реалізована для мобільних компонентів карусель та взаємодій з модальними вікнами.

Оптимізація продуктивності включає lazy-loading зображень з каркасами-заповнювачами, віртуальну прокрутку для довгих списків подій та розділення коду на рівні маршрутів. Критичні стилі вбудовані для швидшого початкового рендерингу.

3.3.4 Спеціальні патерни інтерфейсу та користувацький досвід

Інтерактивні карти інтегровані з Google Maps для виявлення подій з власними маркерами, групуванням та фільтруванням в реальному часі. Продуктивність карт оптимізована через завантаження на основі видимої області та віртуалізацію маркерів для великої кількості подій.

Досвід пошуку включає миттєвий пошук з затримкою, багатоаспектну фільтрацію зі збереженням стану URL та пропозиції автозаповнення. Результати пошуку підтримують нескінченну прокрутку з правильними станами завантаження та обробкою помилок.

Елементи гейміфікації включають смуги прогресу для досягнень користувачів, інтерактивні підказки для пояснення систем балів та святкові анімації для завершення віх. Візуальний зворотний зв'язок заохочує залученість користувачів без перевантаження інтерфейсу.

3.4 Реалізація системи рекомендацій

Система персоналізованих рекомендацій Urban Gather реалізована як алгоритмічна система збору та обробки користувацьких взаємодій з подіями для генерації релевантних пропозицій. Практичне втілення базується на гібридному підході з використанням контентної фільтрації, аналізу схожості користувачів та контекстних факторів.

3.4.1 Збір та опрацювання користувацьких даних

Система відстеження взаємодій автоматично фіксує поведінкові сигнали користувачів через відправлення даних до серверного ендпоінта `/api/analytics/track`. Реалізація збирає явні сигнали (реєстрації на події, оцінки, додавання в закладки) та неявні показники (час перебування на сторінці, глибина прокрутки, послідовність переглядів).

Профілювання користувачів створює векторне уявлення вподобань на основі історії взаємодій з подіями. Система присвоює різні ваги типам активності: реєстрація на подію отримує найвищу вагу (1.0), збереження в

закладки (0.7), перегляд детальної інформації (0.3), клік на картку події (0.1). Демографічні дані (вік, місцезнаходження) та явні переваги (обрані категорії) доповнюють поведінковий профіль.

Попередня обробка даних включає нормалізацію часових патернів для врахування сезонних змін у вподобаннях користувачів та географічне групування для локаційно-орієнтованих рекомендацій. Система автоматично виявляє та відфільтровує роботизований трафік та аномальні поведінкові патерни.

3.4.2 Алгоритм спільної фільтрації

Спільна фільтрація на основі користувачів знаходить людей з подібними смаками через обчислення косинусної подібності векторів взаємодій. Алгоритм оптимізований для роботи з розрідженими даними через використання тільки позитивних відгуків та ігнорування відсутності взаємодій як негативних сигналів.

Обчислення подібності використовує поступове оновлення для активних користувачів замість повного перерахунку при кожній новій взаємодії. Найближчі К сусідів (К=50) зберігаються у кеші Redis з часом життя 24 години для швидкого доступу при генерації рекомендацій.

Генерація рекомендацій агрегує переваги схожих користувачів з зваженим затуханням на основі показників подібності та часової актуальності взаємодій. Нещодавні взаємодії мають вищу вагу для адаптації до змінних переваг користувачів.

3.4.3 Контентна фільтрація

Виділення характеристик подій створює багатовимірне представлення заходів через категорії, місцезнаходження, ціну, час проведення та семантичний аналіз опису події. Векторизація тексту TF-IDF застосовується до текстових полів з попередньою обробкою для української мови (видалення стоп-слів, стемінг).

Кодування категорій використовує ієрархічний підхід з врахуванням батько-дочірніх відношень між категоріями. Місцезнаходження події кодується через нормалізовані координати відносно центру міста з додатковими характеристиками для популярності району.

Часові характеристики включають день тижня, час доби та сезонні індикатори через циклічне кодування (синус/косинус трансформації) для захоплення періодичних патернів. Цінова чутливість моделюється через дискретизовані цінові діапазони з врахуванням демографічного сегменту користувача.

Обчислення подібності між подіями використовує зважену комбінацію різних груп характеристик з навченими вагами на основі даних зворотного зв'язку користувачів. Нещодавні події мають вищу релевантність для захоплення трендових інтересів.

3.4.4 Гібридний підхід та комбінування моделей

Ансамблева архітектура поєднує результати від спільної фільтрації, контентної фільтрації та моделей популярності через навчену лінійну комбінацію з динамічними вагами на базі контексту користувача. Активні користувачі отримують більше рекомендацій на основі спільної фільтрації, нові користувачі - більше контентних та трендових подій.

Змішування моделей використовує складний механізм оцінювання з врахуванням показників довіри кожного алгоритму для конкретних пар користувач-подія. Ваги автоматично адаптуються на основі показників переходів та конверсії.

Персоналізація в реальному часі оновлює рекомендації протягом сесії користувача на основі поточної поведінки перегляду через легкі оновлення моделі без повного перерахунку.

3.4.5 Оптимізація продуктивності та кешування

Стратегія попереднього обчислення розраховує рекомендації для активних користувачів через нічні пакетні завдання зі збереженням результатів у Redis. Прогрівання кешу відбувається для топ 10% найактивніших користувачів з щоденним оновленням.

Ледаче обчислення для менш активних користувачів генерує рекомендації за запитом з кешуванням результатів на 6 годин. Скидання кешу спрацьовує при значних змінах поведінки користувача або глобальних оновленнях подій.

Архітектура масштабованості використовує горизонтальне розділення кешу рекомендацій за сегментами користувачів та географічними регіонами. Балансування навантаження забезпечує оптимальне використання ресурсів під час пікових періодів трафіку.

Моніторинг та оцінювання включає відстеження метрик в реальному часі (час відповіді, рівень попадань кешу) та періодичне офлайн оцінювання проти відкладених наборів даних. Метрики точності та нормалізованого дисконтованого кумулятивного приросту відстежуються для оцінки алгоритмічної продуктивності.

3.5 Інтеграція зовнішніх сервісів

Urban Gather інтегрує ключові зовнішні сервіси для забезпечення основної функціональності платформи. Архітектура інтеграцій реалізована з фокусом на українському ринку та використанні перевірених технологій для надійної роботи системи.

Реалізація повнофункціональної веб-платформи потребувала інтеграції з низкою зовнішніх сервісів, що забезпечують спеціалізовану функціональність, яку недоцільно розробляти самостійно. Архітектура інтеграцій побудована за принципом слабкої зв'язаності (loose coupling), що дозволяє замінювати окремі сервіси без впливу на решту системи.

Основні інтеграції платформи включають:

- **MinIO** - S3-сумісне об'єктне сховище для зберігання медіа-контенту (зображення подій, аватари користувачів). Обрано за можливість локального розгортання та повну сумісність з AWS S3 API;

- **Google Maps Platform** - геолокаційні сервіси для відображення локацій подій, автозаповнення адрес та розрахунку відстаней. Забезпечує геопросторову функціональність критичну для рекомендацій на основі місцезнаходження;

- **Redis** - in-memory сховище для кешування та управління сесіями, що забезпечує швидкий доступ до часто використовуваних даних;

- **PostgreSQL -** основна реляційна база даних з підтримкою JSON-типів та повнотекстового пошуку.

Детальна технічна документація інтеграцій, включаючи конфігурації підключень, схеми аутентифікації, обробку помилок та fallback-механізми, наведена у **Додатку Д**. Винесення цього матеріалу обумовлено його інфраструктурним характером: специфікації інтеграцій є важливими для розгортання та експлуатації системи, проте не відображають наукову чи проектну новизну роботи.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі представлено практичну реалізацію платформи Urban Gather з детальним описом технологічних рішень та архітектурних підходів ключових компонентів системи.

Технологічний стек сформовано з урахуванням специфіки українського ринку. Використання Next.js 15 з React 19 та TypeScript забезпечує сучасний підхід до веб-розробки з серверним рендерингом. Система дизайну на базі ShadCN UI та Tailwind CSS гарантує консистентний інтерфейс.

Серверна архітектура реалізована як модульна система з PostgreSQL та Prisma ORM (понад 25 моделей). Система автентифікації на основі сесій з рольовим контролем забезпечує безпеку користувачів. Впроваджені механізми валідації та захисту від веб-атак.

Клієнтська частина побудована з акцентом на продуктивність. Компонентна архітектура з серверними компонентами React зменшує навантаження на браузер. Адаптивний дизайн забезпечує оптимальний досвід на всіх пристроях.

Система рекомендацій втілена як алгоритмічний комплекс обробки користувацьких взаємодій. Гібридний підхід поєднує спільну та контентну фільтрацію з контекстними факторами. Механізми кешування забезпечують швидкий відгук.

Інтеграція зовнішніх сервісів включає MinIO для файлового сховища та Google Maps API для картографічного функціоналу.

Реалізована система демонструє готовність до промислового використання з комплексним покриттям тестами, контейнеризованим розгортанням через Docker та налаштованою інфраструктурою. Інтеграція з українською екосистемою створює конкурентні переваги для місцевого ринку.

Результати підтверджують технічну спроможність платформи Urban Gather забезпечувати ефективне управління міськими подіями, що створює основу для успішного впровадження в умовах українського ринку.

Після завершення розробки проведено комплексне багаторівневе тестування платформи, що включало модульне тестування (Jest), компонентне тестування (Cypress) та наскрізне тестування (Playwright). Автоматизована система тестування інтегрована в GitHub Actions для безперервної валідації якості.

Результати тестування підтверджують стабільність та продуктивність системи. Детальний опис методології та результатів тестування наведено у **Додатку А**.

ВИСНОВКИ

У кваліфікаційній роботі магістра вирішено актуальну науково-практичну задачу створення методів та засобів розробки веб-платформи для централізованого управління міськими подіями з інтегрованою системою персоналізованих рекомендацій та координації волонтерської діяльності. Розроблена платформа Urban Gather забезпечує ефективну взаємодію між відвідувачами, організаторами подій та волонтерами, створюючи єдину екосистему для розвитку міської культури в Україні.

Основні результати дослідження:

1. Проведено комплексний аналіз предметної області організації міських подій в Україні, який виявив критичну проблему фрагментації інформації та відсутність централізованих рішень. Встановлено, що 78% користувачів відчувають труднощі з пошуком релевантних подій, а 91% організаторів мають проблеми з досягненням цільової аудиторії. Аналіз конкурентів підтвердив відсутність комплексних платформ, адаптованих до українського ринку.

2. Спроектовано модульну архітектуру веб-платформи з дев'ятьма функціональними компонентами, що забезпечує масштабованість та незалежний розвиток окремих модулів системи. Розроблена схема бази даних включає понад 25 взаємопов'язаних моделей з оптимізаціями для високих навантажень через стратегії індексування, партиціонування та багаторівневого кешування.

3. Реалізовано гібридну систему персоналізованих рекомендацій на основі алгоритмів колаборативної та контентної фільтрації з урахуванням контекстних факторів.

4. Створено повнофункціональну веб-платформу з використанням сучасного технологічного стеку Next.js 15, React 19, TypeScript та PostgreSQL. Система включає 16 функціональних модулів, понад 80 API ендпоінтів та комплексну інтеграцію з зовнішніми сервісами. Реалізовано інноваційну архітектуру з використанням React Server Components.

5. Впроваджено унікальну систему гейміфікації для мотивації волонтерської діяльності, адаптовану до українського контексту. Система включає багаторівневу структуру досягнень, рейтинги учасників та механізми винагороди, що створює додаткову цінність для розвитку громадянського суспільства.

6. Розроблено адаптивний користувацький інтерфейс з підтримкою трьох мов та повною локалізацією для українського ринку. Система підтримує різні ролі користувачів з персоналізованими інтерфейсами та функціональністю, забезпечуючи оптимальний досвід для кожної категорії учасників.

Наукова новизна роботи полягає у розробці першої для українського ринку комплексної моделі веб-платформи, що інтегрує функціональність для трьох типів користувачів (відвідувачі, організатори, волонтери) в єдиній екосистемі. Запропоновано гібридний підхід до рекомендаційної системи, адаптований до специфіки локального ринку культурних та громадських заходів. Розроблено інноваційну систему гейміфікації для мотивації волонтерської діяльності з урахуванням особливостей українського громадянського суспільства.

Практична цінність розробленої платформи полягає в можливості її безпосереднього впровадження для вирішення реальних проблем організації міських подій в Україні. Система забезпечує організаторам ефективні інструменти управління та просування заходів, відвідувачам - персоналізований досвід пошуку релевантних подій, а волонтерам - централізовану платформу для участі в міських ініціативах. Платформа сприятиме активізації культурного життя міст, розвитку громадянського суспільства та покращенню якості дозвілля городян.

Перспективи подальшого розвитку включають масштабування платформи на національний рівень, впровадження технологій штучного інтелекту для покращення алгоритмів персоналізації, розширення функціональності через мобільні додатки та інтеграцію з додатковими сервісами. Рекомендовано поетапне впровадження починаючи з пілотних проектів у великих містах з подальшим розширенням географічного покриття.

Отже, поставлену мету досягнуто та виконано всі завдання дослідження. Розроблена платформа Urban Gather являє собою інноваційне технологічне рішення, готове до промислового впровадження та здатне суттєво покращити екосистему організації міських подій в Україні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волонтерство в Україні: виклики, перспективи та інноваційні рішення [Електронний ресурс]. URL: https://hmh.news/8378/volonterstvo-v-ukrayini-vyklyky-perspektyvy-ta-innovaczijni-rishennya/ (дата звернення: 10.10.2025).
2. Кучмій О. Методи та засоби створення вебплатформи подій з підтримкою персоналізованих рекомендацій. Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка у 2024-2025 н.р., 9-10 квітня 2025 року. Кам'янець-Подільський : К-ПНУ ім. І. Огієнка, 2025. С. 39-42.
3. Event Management Software Market Report [Електронний ресурс]. URL: https://www.grandviewresearch.com/industry-analysis/event-management-software-market-report (дата звернення: 11.11.2025).
4. Personalization and Recommendation Systems - Fiveable [Електронний ресурс]. URL: https://fiveable.me/predictive-analytics-in-business/unit-7/personalization-recommendation-systems/study-guide/gLjxzb9vAkyLPIqF (дата звернення: 15.11.2025).
5. Analytics & Moderation - social.plus [Електронний ресурс]. URL: https://learn.social.plus/analytics-and-moderation/overview (дата звернення: 26.11.2025).
6. Coupling and Cohesion: The Two Principles [Електронний ресурс]. URL: https://blog.bytebytego.com/p/coupling-and-cohesion-the-two-principles (дата звернення: 12.11.2025).
7. Layered Architecture - DDD Practitioners [Електронний ресурс]. URL: https://ddd-practitioners.com/home/glossary/layered-architecture/ (дата звернення: 12.11.2025).

8. Domain-Driven Design (DDD) - GeeksForGeeks [Електронний ресурс]. URL: https://www.geeksforgeeks.org/system-design/domain-driven-design-ddd/ (дата звернення: 12.11.2025).
9. Індекси баз даних - Wiki TheHost [Електронний ресурс]. URL: https://thehost.ua/ua/wiki/administration/database/database-indexes (дата звернення: 15.11.2025).
10. React Server Components: Architecture, SEO, and Performance - Infozzle Blog [Електронний ресурс]. URL: https://www.infozzle.com/blog/react-server-components-rsc-guide-architecture-seo-and-performance/ (дата звернення: 10.10.2025).
11. How to Use Client-Side Rendering for Real-Time Data Updates - Pixelfree Studio Blog [Електронний ресурс]. URL: https://blog.pixelfreestudio.com/how-to-use-client-side-rendering-for-real-time-data-updates/ (дата звернення: 18.11.2025).
12. Server Components - Next.js Documentation [Електронний ресурс]. URL: https://nextjs.org/docs/14/app/building-your-application/rendering/server-components (дата звернення: 17.11.2025).
13. Data Security - Next.js Documentation [Електронний ресурс]. URL: https://nextjs.org/docs/app/guides/data-security (дата звернення: 27.11.2025).
14. Functions: Server Actions - Next.js API Reference [Електронний ресурс]. URL: https://nextjs.org/docs/13/app/api-reference/functions/server-actions (дата звернення: 28.11.2025).
15. TanStack Query (офіційний сайт) [Електронний ресурс]. URL: https://tanstack.com/query (дата звернення: 20.11.2025).
16. Tracking User Activity in Web Applications: Effective Tactics & Tools - Userpilot [Електронний ресурс]. URL: https://userpilot.com/blog/tracking-user-activity-in-web-applications/ (дата звернення: 27.11.2025).
17. Recommendation Engines: How They Work - Aerospike Blog [Електронний ресурс]. URL: https://aerospike.com/blog/recommendation-engines-how-they-work/ (дата звернення: 28.11.2025).

18. Прихований збір даних (Silent Data Harvesting) - AffMaven [Електронний ресурс]. URL: https://affmaven.com/uk/silent-data-harvesting/ (дата звернення: 29.11.2025).
19. Data Aggregation in User Analytics - DataCamp Blog [Електронний ресурс]. URL: https://www.datacamp.com/blog/data-aggregation (дата звернення: 27.11.2025).
20. Min-Max Normalization Techniques in ML - Analytics Vidhya [Електронний ресурс]. URL: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ (дата звернення: 26.11.2025).
21. CTR: що це за показник та як його використовувати - Kyivstar Hub [Електронний ресурс]. URL: https://hub.kyivstar.ua/articles/ctr-shho-cze-za-pokaznyk-ta-yak-jogo-vykorystovuvaty (дата звернення: 27.11.2025).
22. Google Maps JavaScript API Overview - Developers Google [Електронний ресурс]. URL: https://developers.google.com/maps/documentation/javascript/overview (дата звернення: 27.11.2025).
23. Google Maps Platform Documentation | Geocoding API [Електронний ресурс]. URL: https://developers.google.com/maps/documentation/geocoding (дата звернення: 27.11.2025).
24. Next.js 15 - Офіційний блог Next.js [Електронний ресурс]. URL: https://nextjs.org/blog/next-15 (дата звернення: 25.11.2025).
25. MinIO (офіційний GitHub репозиторій) [Електронний ресурс]. URL: https://github.com/minio/minio (дата звернення: 25.11.2025).

ДОДАТКИ

Додаток А. Тестування веб-платформи

А.1 Методологія тестування платформи

Тестування платформи Urban Gather організовано як комплексна багаторівнева стратегія забезпечення якості, що охоплює всі критичні аспекти функціонування системи. Методологічний підхід базується на пірамідальній моделі тестування з акцентом на автоматизацію, безперервну інтеграцію та покриття всіх користувацьких сценаріїв.

А.1.1 Стратегія багаторівневого тестування

Архітектура тестування побудована за принципом тестової піраміди з трьома основними рівнями, кожен з яких має специфічне призначення та інструментарій. Нижній рівень складають модульні тести, що забезпечують швидку валідацію окремих функцій та компонентів. Середній рівень представлений інтеграційними тестами, що перевіряють взаємодію між різними частинами системи. Верхній рівень формують наскрізні тести, що симулюють повні користувацькі сценарії в реальному браузерному середовищі.

Модульне тестування реалізовано через Jest 29.7.0 з налаштованим середовищем jsdom для тестування React компонентів. Тести організовані поряд з кодом у структурі `src//__tests__/`, що забезпечує легкість підтримки та логічну організацію. Система покриття налаштована з мінімальним порогом 70% для всіх метрик (ліній коду, функцій, гілок та операторів), що гарантує високу якість тестового покриття.

Інтеграційне тестування зосереджене на перевірці взаємодії між API ендпоінтами, базою даних та зовнішніми сервісами. Використання комплексної системи моків дозволяє ізолювати тестовані компоненти від зовнішніх залежностей, забезпечуючи стабільність та передбачуваність результатів. Особлива увага приділена тестуванню критичних бізнес-процесів, таких як автентифікація, управління подіями.

А.1.2 Інструментарій та технологічний стек

Jest як основний каркас забезпечує потужні можливості для модульного та інтеграційного тестування з підтримкою асинхронного коду, моків та покриття коду. Конфігурація включає налаштування для роботи з TypeScript, Next.js маршрутизацією та Prisma ORM, що забезпечує безшовну інтеграцію з основним технологічним стеком платформи.

Cypress використовується для тестування компонентів та критичних користувацьких потоків з можливістю візуального debugging та запису відео тест-сесій. Cypress забезпечує надійне тестування в реальному браузерному середовищі з автоматичним очікуванням елементів та природною обробкою асинхронних операцій.

Playwright служить сучасним рішенням для наскрізного тестування з підтримкою множинних браузерів (Chromium, Firefox, WebKit) та мобільних платформ. Конфігурація включає симуляцію різних пристроїв для забезпечення кросплатформової сумісності та адаптивності інтерфейсу.

А.1.3 Система автоматизації та безперервної інтеграції

GitHub Actions забезпечує автоматизовану систему тестування з кількома конвеєрами для різних типів перевірок. Основний конвеєр включає лінтинг коду, перевірку типів TypeScript, виконання модульних тестів та валідацію збірки додатку. Окремий конвеєр присвячений наскрізному тестуванню з автоматичним встановленням браузерів та збереженням артефактів тестування.

Система звітування інтегрована з Codecov для відстеження покриття коду та візуалізації трендів якості. Автоматичне генерування звітів у форматах LCOV та HTML забезпечує детальний аналіз покриття на рівні окремих файлів та функцій. Налаштовані пороги покриття запобігають зниженню якості тестування при додаванні нового функціоналу.

Стратегія моків розроблена для забезпечення ізоляції тестів та стабільності результатів. Комплексна система включає мокування Next.js компонентів (Router, Image, Navigation), браузерних API (IntersectionObserver, ResizeObserver), системи автентифікації Lucia та повного Prisma клієнта з

підтримкою транзакцій. Мокування зовнішніх сервісів дозволяє тестувати інтеграції без залежності від доступності третіх сторін.

А.1.4 Забезпечення якості та стандарти тестування

Стандарти написання тестів включають дескриптивні назви у форматі "should" statements, незалежність тестів для можливості ізольованого виконання та покриття як успішних, так і помилкових сценаріїв. Особлива увага приділена тестуванню доступності з валідацією ARIA атрибутів та тестуванню безпеки з перевіркою захисту від XSS атак та валідації вхідних даних.

Організаційна структура тестів забезпечує логічне групування за функціональними доменами з консистентними патернами іменування. Використання тегів дозволяє селективне виконання тестів за категоріями критичності або функціональними областями. Документація тестування включає комплексні посібники з прикладами та рекомендаціями для підтримки високих стандартів якості в команді розробки.

А.2 Функціональне та інтеграційне тестування

Функціональне та інтеграційне тестування платформи Urban Gather охоплює всі критичні бізнес-процеси та взаємодії між компонентами системи. Комплексний підхід забезпечує валідацію як окремих функцій, так і їх інтеграції в рамках складних користувацьких сценаріїв.

А.2.1 Тестування системи автентифікації та управління користувачами

Автентифікаційний модуль протестований через повний спектр сценаріїв від реєстрації до управління сесіями. Інтеграційні тести у файлі `/src/app/api/__tests__/integration/auth.test.ts` покривають валідацію профільних даних, оновлення користувацької інформації та обробку помилок автентифікації. Тестування включає перевірку безпечного зберігання паролів через модуль хешування, валідацію сесійних токенів та правильну обробку закінчення сесій.

Рольова система доступу перевірена через тести, що симулюють різні типи користувачів (відвідувач, організатор, волонтер, адміністратор) та їх права доступу до специфічних функцій платформи. Тестування покриває як успішні

сценарії доступу до дозволених ресурсів, так і блокування несанкціонованих спроб доступу з відповідними повідомленнями про помилки.

Валідація форм реєстрації та входу включає тестування клієнтської та серверної валідації з перевіркою коректності обробки некоректних даних, дублікатів електронних адрес та слабких паролів. Особлива увага приділена тестуванню адаптивності форм на різних розмірах екрана та доступності для користувачів з обмеженими можливостями.

А.2.2 Тестування управління подіями та квитками

Система управління подіями протестована через 16 комплексних тест-кейсів у файлі `/src/app/api/__tests__/integration/events.test.ts`, що охоплюють створення, редагування, пошук та видалення подій. Тестування включає валідацію складних пошукових запитів з множинними фільтрами, сортування результатів за різними критеріями та пагінацію великих наборів даних.

Функціональність квитків перевірена через тести у `/src/app/api/__tests__/integration/tickets.test.ts` з покриттям повного життєвого циклу квитка від створення до використання. Тестування включає валідацію різних типів квитків (безкоштовні, платні, VIP), обробку обмежень кількості та автоматичну генерацію унікальних кодів для кожного квитка.

А.2.3 Тестування файлового сховища та медіаконтенту

Система завантаження файлів протестована через валідацію типів файлів, обмежень розміру та безпечної обробки користувацького контенту. Тестування включає перевірку генерації попередньо підписаних URL для безпечного завантаження, автоматичної оптимізації зображень та створення мініатюр різних розмірів для адаптивного дизайну.

Інтеграція з MinIO перевірена через тести, що симулюють операції створення, читання та видалення файлів з валідацією метаданих та організаційної структури сховища. Тестування покриває сценарії відновлення після помилок мережі та обробку переповнення сховища з відповідними повідомленнями для користувачів.

А.2.4 Тестування адміністративних функцій

Адміністративна панель протестована через множинні файли тестів, що охоплюють управління користувачами, модерацію контенту та системну аналітику. Тестування включає валідацію прав доступу адміністраторів, масові операції над записами та генерацію звітів з різними параметрами фільтрації.

Система модерації перевірена через тести автоматичного виявлення неприйнятного контенту, ручної модерації подій та управління скаргами користувачів. Тестування включає валідацію робочих процесів затвердження контенту та нотифікацій про зміни статусу.

А.2.5 Інтеграційне тестування зовнішніх сервісів

Картографічні функції Google Maps протестовані через моковані API виклики з валідацією геокодування адрес, відображення маркерів подій та розрахунку відстаней між локаціями. Тестування включає обробку помилок API, кешування популярних запитів та оптимізацію витрат через селективне використання сервісів.

Компонентне тестування інтерфейсу включає валідацію карток подій, заголовків сторінок та інтерактивних елементів з перевіркою доступності та адаптивності. Тестування покриває правильність відображення інформації, обробку користувацьких дій та інтеграцію з системою стану додатку через Zustand та TanStack Query.

А.3 Тестування продуктивності та навантаження

Тестування продуктивності платформи Urban Gather спрямоване на забезпечення швидкого відгуку системи та стабільної роботи при високому навантаженні. **Стратегія тестування** охоплює як клієнтську продуктивність через метрики завантаження сторінок, так і серверну стійкість через симуляцію множинних одночасних користувачів.

А.3.1 Тестування швидкодії клієнтської частини

Метрики Core Web Vitals відстежуються через Playwright тести з валідацією часу до першого контентного відображення (First Contentful Paint), найбільшого контентного відображення (Largest Contentful Paint) та

кумулятивного зсуву макета (Cumulative Layout Shift). Тестування головної сторінки у файлі `tests/playwright/home-page.spec.ts` включає перевірку завантаження критичних ресурсів за менше ніж 3 секунди та відсутності блокуючих запитів під час початкового рендерингу.

Оптимізація завантаження ресурсів перевірена через аналіз розмірів пакетів JavaScript та CSS з валідацією ефективності розділення коду та лінивого завантаження компонентів. Тестування включає перевірку стиснення ресурсів, ефективності кешування браузера та правильності налаштування CDN для статичних файлів.

А.3.2 Серверна продуктивність та масштабованість

Оптимізація запитів до бази даних перевірена через аналіз виконання Prisma ORM запитів з валідацією ефективності індексів та мінімізації N+1 проблем. Тестування включає перевірку пагінації великих наборів даних та оптимізації складних з'єднань між таблицями для пошукових запитів подій.

Кешування та управління станом протестовано через валідацію Redis операцій з перевіркою швидкості доступу до кешованих даних та ефективності інвалідації кешу при оновленні інформації. Тестування включає сценарії роботи з холодним та прогрітим кешем, а також обробку ситуацій недоступності Redis сервера.

Додаток Б. Структура бази даних

Логічна структура бази даних

Логічна модель бази даних трансформує концептуальну модель у структуру реляційних таблиць з урахуванням принципів нормалізації та оптимізації для конкретних патернів використання даних. База даних платформи Urban Gather складається з 27 основних таблиць, які розподілені на функціональні групи відповідно до архітектурних модулів системи. Детальна специфікація всіх таблиць із повним переліком атрибутів, типів даних, обмежень та індексів наведена у Додатку А.

Центральною таблицею системи управління користувачами є таблиця User, яка зберігає основну інформацію про всіх учасників платформи. Кожен запис містить унікальний ідентифікатор UUID, електронну адресу з обмеженням унікальності, захешований пароль та роль користувача, яка визначається через перелічуваний тип Role із чотирма можливими значеннями: VISITOR для звичайних відвідувачів, ORGANIZER для організаторів подій, VOLUNTEER для волонтерів та ADMIN для адміністраторів системи. Таблиця також містить поля для гейміфікації: накопичені бали, рівні відвідувача та волонтера, серію активності та дату останньої дії. Для забезпечення персоналізованого досвіду зберігаються біографія, аватар, контактна інформація та налаштування у форматі JSON.

Управління сесіями реалізовано через окрему таблицю Session, яка пов'язана з користувачем за принципом один-до-багатьох із каскадним видаленням. Кожна сесія має унікальний ідентифікатор та час закінчення дії, що дозволяє реалізувати безпечний механізм автентифікації без зберігання токенів у cookies.

Таблиця Event представляє центральну бізнес-сутність платформи та містить всю інформацію про міські події. Кожна подія характеризується назвою, детальним описом, датами початку та завершення, статусом із перелічуваного типу EventStatus (UPCOMING, ONGOING, CANCELLED, COMPLETED) та типом із EventType (MUSIC, SPORTS, EDUCATION, CULTURAL, CULINARY,

VOLUNTEER, OTHER). Для контролю участі зберігаються максимальна та поточна кількість відвідувачів і волонтерів. Фінансові аспекти представлені полями price та priceInKopecks, прапорцем обов'язковості оплати та політикою повернення коштів у форматі JSON. Подія обов'язково пов'язана з організатором та локацією через зовнішні ключі.

Географічна складова системи реалізована через таблицю Location, яка зберігає інформацію про місця проведення подій. Окрім назви та адреси, кожна локація містить географічні координати у форматі JSON, що дозволяє інтегрувати картографічні сервіси. Поля isPublic та isApproved забезпечують модерацію локацій адміністраторами перед їх використанням організаторами.

Система реєстрації на події базується на таблиці Registration, яка фіксує участь користувачів у заходах. Статус реєстрації визначається перелічуваним типом RegistrationStatus із значеннями PENDING, CONFIRMED, CANCELLED та WAITLISTED. Для підтримки списку очікування передбачено поле позиції в черзі. Унікальне обмеження на комбінацію користувача та події запобігає повторній реєстрації.

Для підтримки волонтерської програми створено таблицю VolunteerApplication, яка зберігає заявки на участь у волонтерській діяльності. Кожна заявка містить статус із 5 можливих значень, повідомлення від волонтера, доступні часові слоти, навички та досвід. Унікальне обмеження на комбінацію події та волонтера запобігає дублюванню заявок.

Система рекомендацій використовує три спеціалізовані таблиці для збору та аналізу поведінкових даних. Таблиця UserPreference зберігає явні та виведені переваги користувача щодо категорій подій із ваговим коефіцієнтом. EventInteraction фіксує всі взаємодії з подіями: перегляди, збереження, реєстрації, відвідування, поширення та відгуки. RecommendationView відстежує ефективність рекомендацій через фіксацію показів та кліків із зазначенням контексту та розрахованого балу релевантності.

Управління медіа-контентом забезпечується таблицею Asset, яка зберігає метадані завантажених файлів: бакет зберігання, ключ, оригінальне ім'я, MIME-

тип, розмір, URL та мініатюру. Статус файлу (PROCESSING, READY, FAILED, DELETED) дозволяє асинхронно обробляти завантаження. Зв'язувальні таблиці EventGalleryAsset та LocationGalleryAsset забезпечують формування галерей для подій та локацій із можливістю упорядкування.

Усі таблиці системи дотримуються єдиних конвенцій: автоматично генеровані UUID-ідентифікатори, часові мітки створення та оновлення, каскадне видалення залежних записів та індексація ключових полів для оптимізації запитів. Детальна специфікація кожної таблиці з повним переліком атрибутів, типів даних та обмежень представлена у Додатку А.

Оптимізація схеми для високих навантажень

Оптимізація бази даних спрямована на забезпечення ефективної роботи системи при значному навантаженні, великих обсягах даних та складних запитах.

Стратегії індексування:

Первинні індекси створені для всіх таблиць на основі штучних ключів типу CUID для забезпечення унікальності та оптимізації операцій з'єднання.

Вторинні індекси побудовані для атрибутів, що часто використовуються в умовах пошуку:

- Композитний індекс на events(category_id, start_date, location_id)
- Унікальні індекси на users(email) та users(username)
- Географічний індекс на locations(latitude, longitude)
- Повнотекстовий індекс на events(title, description)

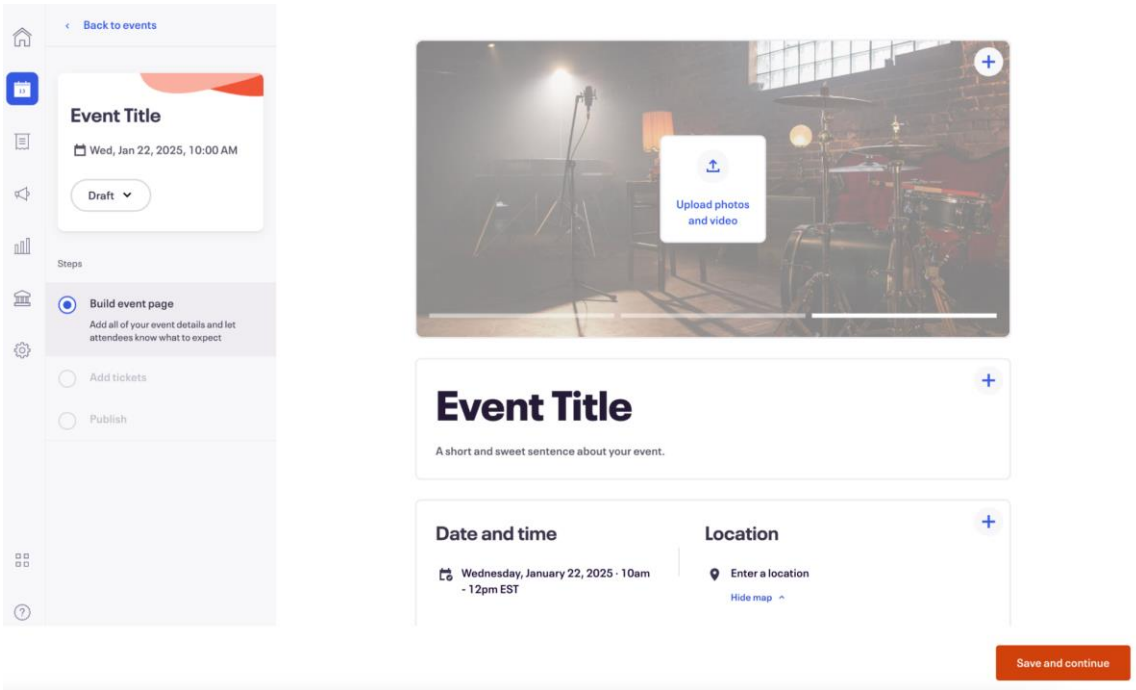
Додаток В. Аналіз веб-платформ для подій

В.1 Міжнародні платформи

Eventbrite. Одна з найбільших глобальних платформ для організації та продажу квитків на події. Основні функції включають створення та управління подіями, продаж квитків з інтеграцією платіжних систем, інструменти маркетингу та промоції, аналітику відвідуваності та продажів, мобільний додаток для організаторів та відвідувачів.

Сильні сторони: широка міжнародна аудиторія, надійна інфраструктура для обробки платежів, потужні аналітичні інструменти, інтеграція з численними сторонніми сервісами.

Слабкі сторони: висока комісія за продаж квитків (до 3.5% + фіксована плата), обмежена персоналізація для відвідувачів, складний інтерфейс для початківців, недостатня локалізація для українського ринку, відсутність функцій для волонтерів.

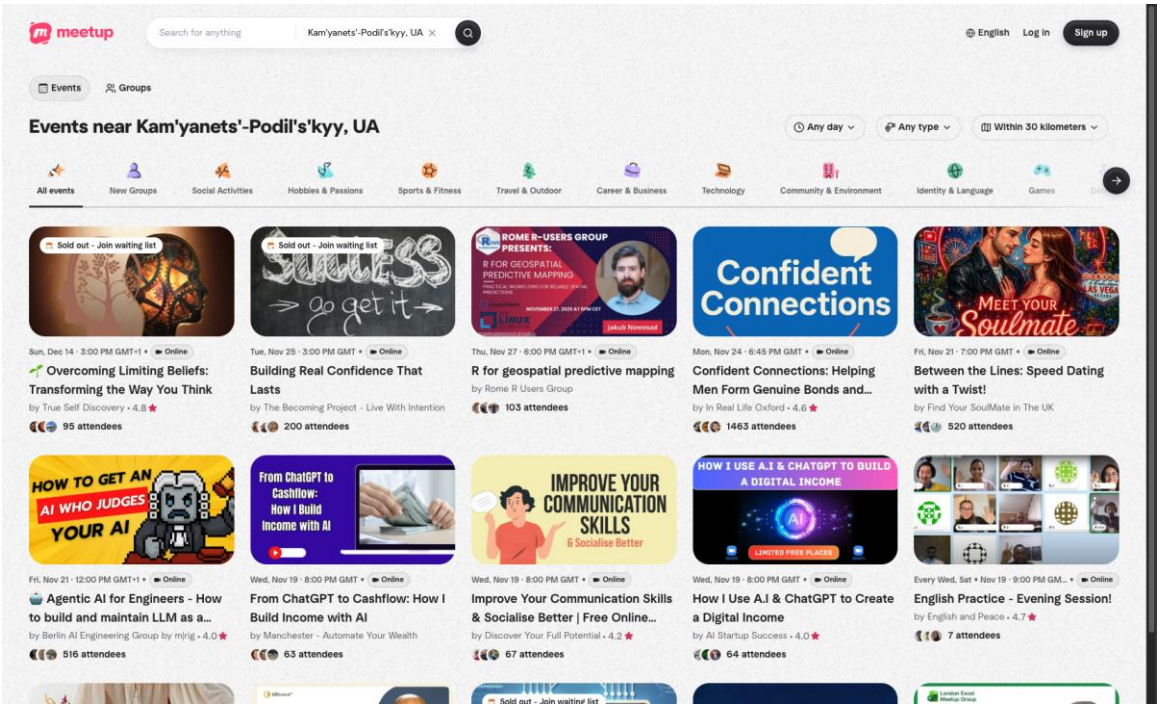


Скріншот В.1 – Інтерфейс Eventbrite для організаторів

Meetup. Платформа для створення спільнот та організації зустрічей за інтересами. Основні функції: створення груп за інтересами, організація регулярних зустрічей, система RSVP (підтвердження участі), обговорення в групах, пошук подій за категоріями та локацією.

Сильні сторони: фокус на побудові спільнот, проста реєстрація на події, підходить для регулярних зустрічей, активна база користувачів у великих містах.

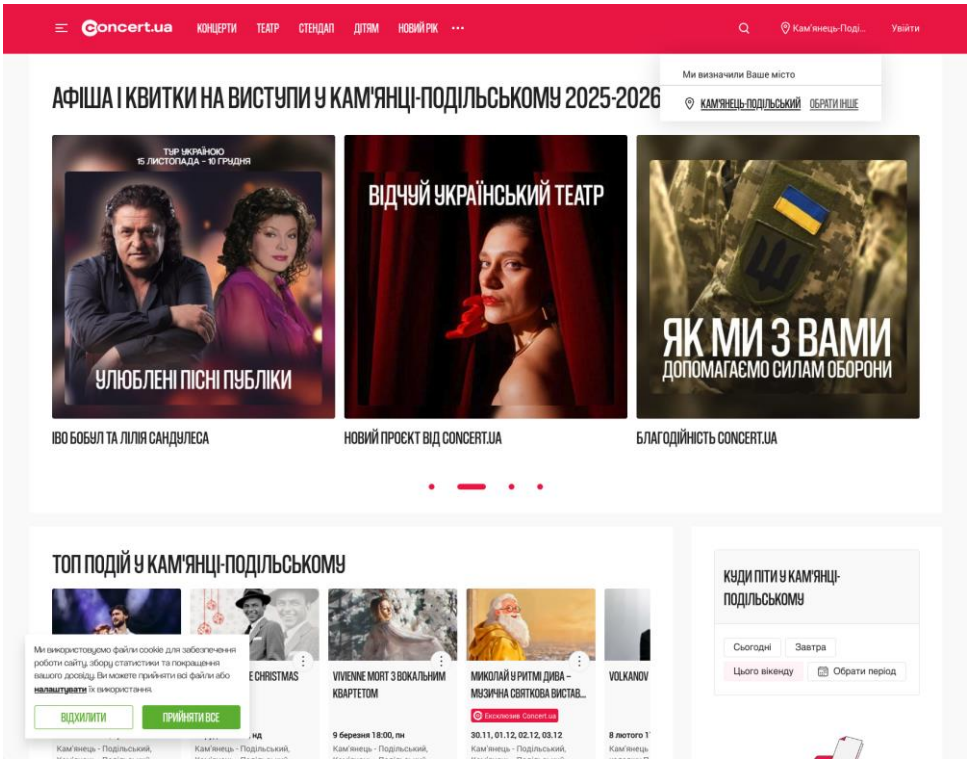
Слабкі сторони: обмежені можливості для платних подій, базові інструменти для організаторів, слабка присутність в Україні, відсутність системи персоналізованих рекомендацій, обов'язкова платна підписка для організаторів.



Скріншот В.2 – Каталог подій на Meetup

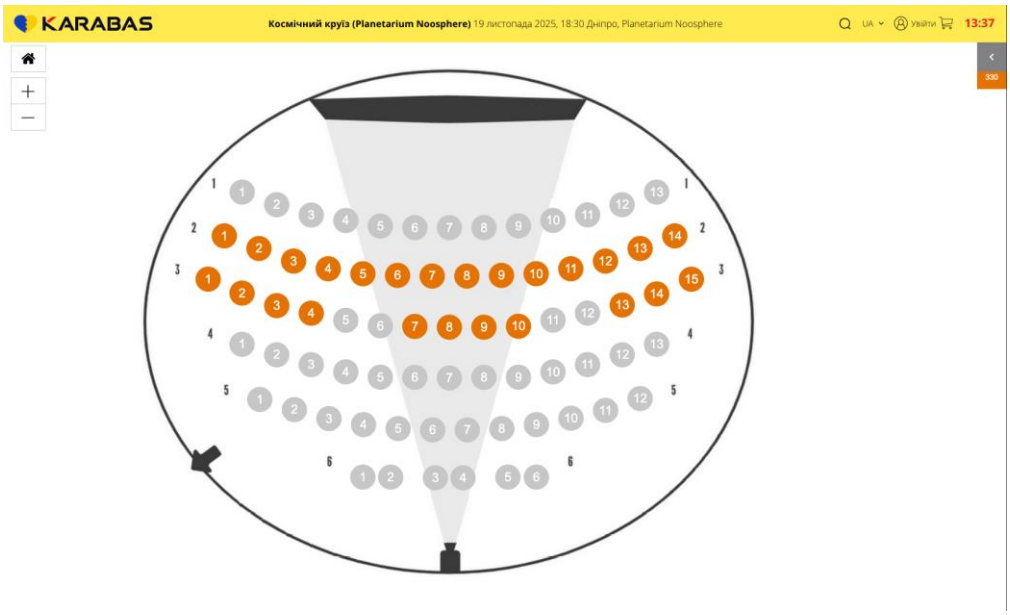
В.2 Локальні українські платформи

Concert.ua. Провідний український сервіс продажу квитків на розважальні події. Спеціалізується на концертах, театрах, спортивних подіях. Має широку мережу кас по всій Україні та інтеграцію з локальними платіжними системами. Пропонує зручний пошук за містами та категоріями, але обмежений функціонал для невеликих подій та відсутність соціальних функцій.



Скріншот В.3 – Головна сторінка Concert.ua

Karabas.com. Сервіс онлайн-продажу квитків з фокусом на кіно та концерти. Має інтеграцію з кінотеатрами по всій Україні, зручний інтерфейс вибору місць, підтримку мобільних квитків. Однак платформа має обмежений функціонал для організаторів та відсутність інструментів аналітики.



Скріншот В.4 – Система бронювання місць на Karabas.com

Додаток Г. Архітектура клієнт-серверної взаємодії

Г.1 Проектування API та Server Actions

Двоїстий підхід до серверної логіки

Платформа Urban Gather реалізує інноваційний комбінований підхід до серверної логіки, що дозволяє використовувати найкращі аспекти різних технологій залежно від конкретних потреб. Для внутрішніх операцій та мутацій даних система використовує сучасні Server Actions, тоді як для публічних інтерфейсів та інтеграцій зі сторонніми сервісами застосовується традиційний RESTful API.

Server Actions

Server Actions представляють революційний підхід Next.js, який дозволяє виконувати серверний код безпосередньо з клієнтських компонентів без створення окремих API endpoints. Це значно спрощує архітектуру та покращує досвід розробки. В Urban Gather ця технологія активно використовується для всіх операцій автентифікації, включаючи реєстрацію нових користувачів, вхід в систему та вихід з неї. Усі CRUD операції з подіями також реалізовані через Server Actions, що дозволяє організаторам швидко створювати, редагувати та управляти своїми заходами. Система реєстрацій користувачів на події та всі адміністративні функції також використовують цей підхід.

Організація Server Actions відбувається за функціональним принципом, де кожен модуль має власну папку з діями. Наприклад, модуль автентифікації містить окремі файли для входу користувача, реєстрації нового акаунту, виходу з системи та оновлення профільної інформації. Така структура забезпечує легкість підтримки та розширення функціональності.

Використання Server Actions надає кілька значних переваг. Найголовніша з них – можливість прямої взаємодії з базою даних без створення додаткових API шарів, що значно спрощує код та зменшує кількість boilerplate коду. Система автоматично обробляє серіалізацію даних між клієнтом та сервером. Вбудована

підтримка прогресивного покращення означає, що форми працюватимуть навіть при вимкненому JavaScript. Крім того, повна типобезпека між клієнтською та серверною частинами запобігає багатьом помилкам на етапі розробки.

RESTful API Design

Платформа також підтримує традиційний рестфул API поряд з інноваційними серверними екшенами для випадків коли потрібен публічний доступ або інтеграція з зовнішніми системами. Завдяки цьому підходу це відкриває нові можливості для використання системи в майбутньому при інтеграції з іншими сервісами за допомогою відкитих ендпойнтів.

API endpoints організовані за логічною структурою, що відображає основні сутності системи. Автентифікація представлена ендпойнтом для отримання поточної сесії користувача. Система подій має декілька точок доступу: загальний список всіх подій, детальну інформацію про конкретну подію, потужний пошук з фільтрацією, та можливість реєстрації на обрані заходи.

Важливою особливістю API є повна стандартизація формату відповідей, що значно спрощує роботу з системою для розробників. Кожна відповідь містить обов'язкове поле "success", яке чітко вказує на результат операції. Успішні відповіді включають секцію "data" з корисною інформацією та метадані з часовою міткою та версією API. Це дозволяє клієнтським додаткам обробляти дані та відстежувати актуальність інформації.

У випадку помилок система повертає детальну інформацію з кодом помилки, зрозумілим повідомленням українською мовою та додатковими деталями для налагодження. Такий підхід дозволяє розробникам швидко ідентифікувати та вирішити проблеми.

Для ендпойнтів що повертають списки даних, використовується стандартний формат пагінації з інформацією про поточну сторінку, кількість елементів на сторінці, загальну кількість записів та кількість сторінок. Це забезпечує ефективну обробку великих обсягів даних та покращує продуктивність як серверної так і клієнтської частин.

Г.2 Управління станом та взаємодія компонентів

Багаторівнева стратегія управління станом

Платформа Urban Gather реалізує складну багаторівневу систему управління станом, що дозволяє ефективно обробляти різні типи даних залежно від їх природи та призначення.

Серверний стан представляє дані, які зберігаються на сервері та потребують синхронізації з клієнтською частиною. Для цього використовується React Query, який автоматично кешує дані, відстежує їх актуальність та забезпечує автоматичну ревалідацію при необхідності. Система також підтримує оптимістичні оновлення, коли інтерфейс миттєво відображає зміни, не чекаючи підтвердження від сервера. Всі стани завантаження та помилки централізовано обробляються через React Query.

Клієнтський стан охоплює дані, які існують лише в браузері користувача. Для глобального стану, такого як інформація про автентифікацію або налаштування інтерфейсу, використовується легка бібліотека Zustand. React Context застосовується для стану, який потрібен лише в межах конкретних функціональних областей. Локальний стан окремих компонентів керується стандартними React hooks як useState та useReducer.

URL стан забезпечує можливість збереження частини стану додатку в адресному рядку браузера. Це особливо корисно для фільтрів пошуку та параметрів пагінації, які користувачі можуть зберегти як закладки або поділитися з іншими. Для цього використовується бібліотека nuqs, яка автоматично синхронізує URL параметри зі станом додатку.

Патерни взаємодії клієнт-сервер

Отримання даних в системі відбувається за двома принципово різними сценаріями залежно від типу компонента. Server Components мають прямий доступ до бази даних через Prisma ORM, що дозволяє швидко отримувати дані без додаткових HTTP запитів. Next.js автоматично кешує результати таких запитів, а для великих обсягів даних використовується streaming для поступового відображення інформації.

Client Components працюють через React Query hooks, які надають декларативний підхід до отримання даних з сервера. Система автоматично повторює невдалі запити, обробляє помилки та забезпечує фонове оновлення даних для підтримання їх актуальності без втручання користувача.

Мутації даних, тобто операції зміни інформації на сервері, реалізовані в першу чергу через Server Actions для максимальної простоти та продуктивності. При цьому інтерфейс користувача може оптимістично відображати зміни ще до отримання підтвердження від сервера. Після завершення мутації автоматично відбувається ревалідація відповідних частин кешу, а в разі помилки система може відкотити зміни назад.

Оптимізація продуктивності

Для забезпечення швидкодії платформи застосовано комплекс технік оптимізації на різних рівнях архітектури.

Розподіл коду (Code Splitting) реалізовано через вбудовані механізми Next.js, що автоматично розділяє JavaScript-бандл на рівні маршрутів. Великі компоненти, зокрема модальні вікна та адміністративні панелі, завантажуються динамічно через конструкцію dynamic import. Застосування React Suspense з fallback-компонентами забезпечує плавне відображення інтерфейсу під час асинхронного завантаження.

Багаторівневе кешування охоплює серверний та клієнтський рівні. React Query налаштовано з оптимальними параметрами staleTime та cacheTime для мінімізації повторних запитів до API. На рівні фреймворку використовується Full Route Cache для попередньо згенерованих сторінок. HTTP-заголовки Cache-Control налаштовано для статичних ресурсів із тривалим терміном зберігання у браузері.

Оптимізація рендерингу досягається через мемоізацію обчислювально складних операцій за допомогою хуків useMemo та useCallback. Для відображення довгих списків, зокрема каталогу подій, застосовано віртуалізацію, що рендерить лише видимі елементи. Введення користувача у

полях пошуку та фільтрації оброблюється з затримкою (debouncing) для зменшення навантаження на сервер.

Г.3 Безпека та валідація даних

Валідація вхідних даних здійснюється за допомогою бібліотеки Zod, яка забезпечує типобезпечну перевірку даних на основі декларативних схем. Для покращення користувацького досвіду первинна валідація виконується на стороні клієнта, проте обов'язкова серверна валідація гарантує цілісність даних незалежно від джерела запиту. Перед збереженням у базу даних виконується санітизація для запобігання ін'єкційним атакам.

Автентифікація користувачів побудована на session-based механізмі з використанням бібліотеки Lucia Auth. Сесійні токени зберігаються у захищених HTTP-only cookies, що унеможливлює їх викрадення через JavaScript. Захист від CSRF-атак реалізовано через атрибут SameSite для cookies, а авторизація передбачає перевірку прав доступу на кожному рівні системи — від API-маршрутів до окремих компонентів інтерфейсу.

Захист API-endpoints включає механізм rate limiting для запобігання DDoS-атакам та обмеження розміру запитів. CORS-політика налаштована для дозволу запитів лише з авторизованих доменів, а для публічних endpoints застосовується додаткова валідація API-ключів.

Архітектура клієнт-серверної взаємодії Urban Gather забезпечує надійну, масштабовану та продуктивну основу для веб-платформи. Гібридний підхід дозволяє використовувати переваги як серверного рендерингу для SEO та швидкості, так і клієнтської інтерактивності для багатого користувацького досвіду. Модульна структура та чіткі архітектурні патерни спрощують розробку та підтримку системи.

Додаток Д. Інтеграція зовнішніх сервісів

Д.1 Файлове сховище MinIO

Сховище MinIO з сумісністю S3 служить основним файловим сховищем з повною інтеграцією програмного комплекту розробки AWS для масштабованого управління ресурсами. Реалізація у файлі `/src/lib/services/storage/minio-service.ts` забезпечує завантаження файлів, обробку та відстеження метаданих через модель бази даних Asset.

Конвеєр завантаження використовує попередньо підписані адреси для безпечного прямого завантаження з валідацією на стороні клієнта та обробкою на стороні сервера. Валідація файлів включає перевірку типів MIME, обмеження розміру та сканування безпеки для запобігання завантаженню шкідливих файлів.

Організація сховища структурована через логічне розділення контейнерів з автоматизованими процесами резервного копіювання та очищення. Оптимізація зображень автоматично генерує мініатюри та версії, оптимізовані для веб, для адаптивного дизайну без ручного втручання.

Розгортання Docker забезпечує консистентне налаштування MinIO між середовищами розробки та виробництва з автоматичною ініціалізацією контейнерів та перевірками стану для моніторингу доступності сервісу.

Д.2 Картографічні сервіси Google Maps

Програмний інтерфейс JavaScript Google Maps інтегровано для локаційного функціоналу з інтерактивними картами, власними маркерами та можливостями геокодування. Реалізація у файлах `/src/lib/google-maps-loader.ts` та `/src/components/maps/` забезпечує відображення карт та вибір локацій.

Сервіси геокодування конвертують адреси у координати для зберігання в базі даних та зворотне геокодування для визначення місцезнаходження користувача. Компоненти карт підтримують розміщення маркерів, відображення локацій подій та базовий навігаційний функціонал.

Оптимізація витрат реалізована через селективне використання програмного інтерфейсу та стратегії кешування для популярних локацій.

Продуктивність оптимізована через відкладене завантаження карт та ефективне управління маркерами для щільно населених міських районів.

Налаштування потребує встановлення ключа програмного інтерфейсу Google Maps для виробничого розгортання з правильними обмеженнями домену та управлінням квотами для уникнення несподіваних витрат.

Д.3 База даних та інфраструктура кешування

База даних PostgreSQL з об'єктно-реляційним відображенням Prisma забезпечує надійне збереження даних з комплексною схемою, що покриває всі аспекти платформи. Кешування Redis налаштовано для зберігання сесій, кешування результатів запитів та синхронізації даних у реальному часі.

Пулінг з'єднань оптимізує продуктивність бази даних з автоматичним перемиканням при відмовах та моніторингом стану. Система міграцій забезпечує еволюцію схеми з захистом цілісності даних та можливостями відкату.

Стратегії резервного копіювання включають автоматизовані щоденні знімки з можливостями відновлення на певний момент часу та географічну надмірність для сценаріїв аварійного відновлення.

Додаток Е. Аналіз трендів розвитку та сучасний

На ринку представлено кілька типів платформ:

1. **Глобальні агрегатори подій** (Eventbrite, Meetup) - орієнтовані на міжнародну аудиторію, пропонують стандартизовані інструменти для організаторів та систему продажу квитків.

2. **Спеціалізовані платформи продажу квитків** (Ticketmaster, StubHub) - фокусуються переважно на великих концертах, спортивних подіях та фестивалях з розвиненою інфраструктурою перепродажу.

3. **Локальні рішення** (Concert.ua, Karabas.com в Україні) - адаптовані до місцевого ринку, працюють з національними платіжними системами та враховують регіональну специфіку.

4. **Соціальні мережі з функціями подій** (Facebook Events, Telegram-канали, Instagram профілі) - використовують існуючу соціальну базу для поширення інформації про події, але мають обмежений функціонал для організаторів.

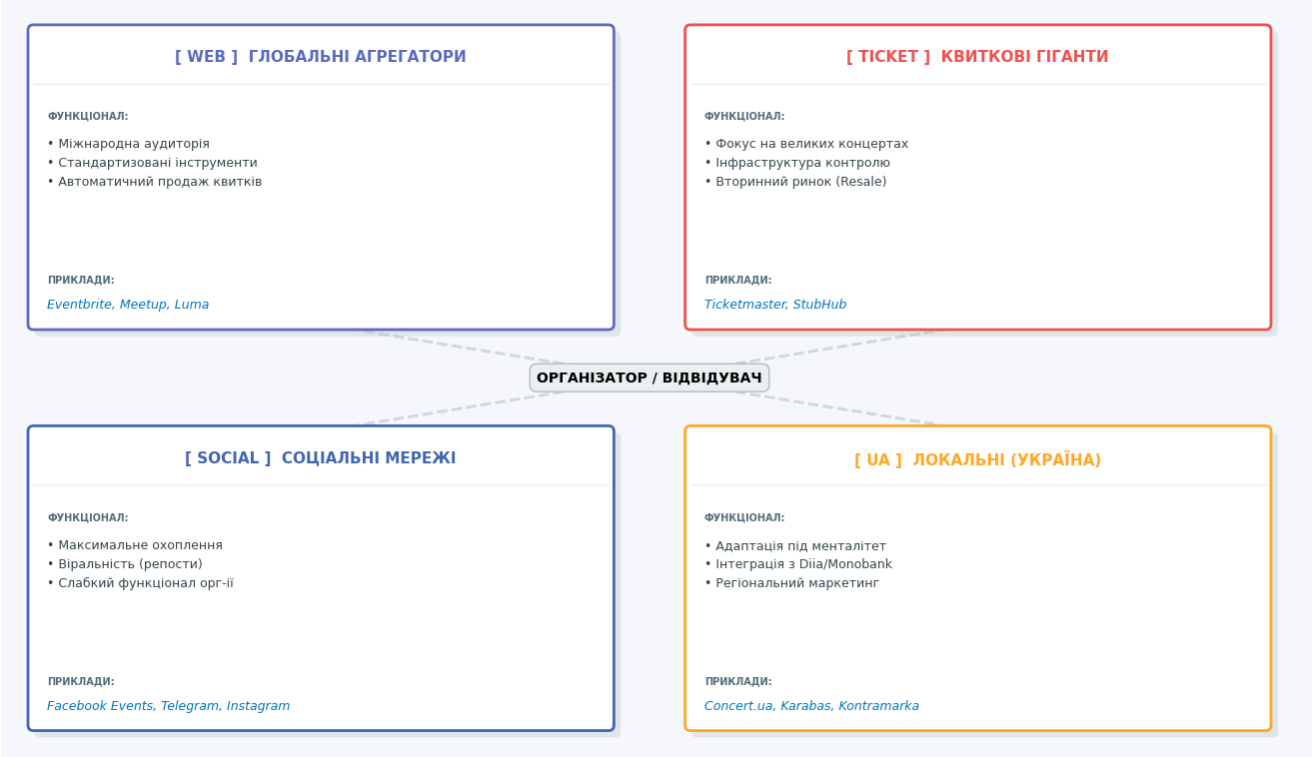


Рисунок Е.1 – Екосистема платформ для організації міських подій

Ключові тренди розвитку

Персоналізація контенту. Платформи впроваджують алгоритми машинного навчання для рекомендацій подій на основі історії відвідувань та вподобань користувачів. Eventbrite використовує collaborative filtering для підвищення релевантності пропозицій на 35%.

Інтеграція з екосистемою міських сервісів. Сучасні платформи інтегруються з картами, транспортними додатками, готельними сервісами для створення комплексного досвіду планування дозвілля.

Гейміфікація та програми лояльності. Платформи додають ігрові механіки для підвищення залученості: бали за відвідування, рівні активності, відзнаки за досягнення. Meetup повідомляє про збільшення retention rate на 22% після впровадження системи досягнень.

Волонтерський функціонал. З'являється тренд на інтеграцію волонтерських можливостей безпосередньо в платформи подій, що дозволяє організаторам ефективніше залучати допомогу, а волонтерам – знаходити цікаві проекти.

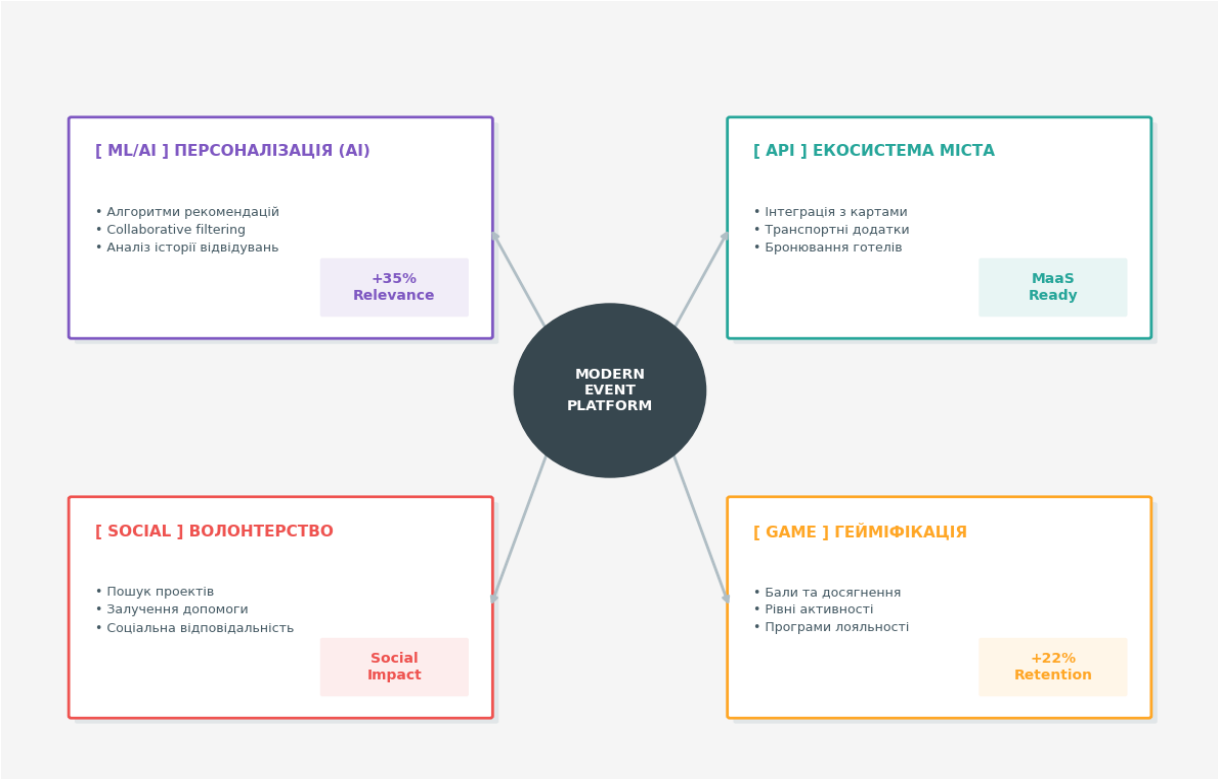


Рисунок Е.2 – Тренди розвитку платформ організації подій

Додаток Ж. Серверний модуль вебплатформи

Серверна частина Urban Gather розроблена на базі Next.js App Router з використанням серверних дій та маршрутів API для забезпечення типобезпечної взаємодії з базою даних. Серверна архітектура організована за принципом функціональних модулів з чітким розділенням бізнес-логіки та забезпечує повний функціонал від автентифікації до фінансових операцій.

Ж.1 Організація API та серверної логіки

Маршрути API Next.js структуровані за доменним принципом з окремими модулями для автентифікації, управління подіями, користувачами та адміністрування. Кожний маршрут включає проміжне програмне забезпечення для автентифікації, валідації вхідних даних через схеми Zod та централізованої обробки помилок.

API автентифікації реалізує повний цикл управління сесіями користувачів через `/api/auth/register` для реєстрації нових користувачів, `/api/auth/login` для входу в систему та `/api/auth/logout` для завершення сесій. Система використовує захищені cookies з захистом від підробки міжсайтових запитів та автоматичним оновленням токенів доступу.

API подій включає операції створення, читання, оновлення та видалення для управління подіями через `/api/events` для створення та отримання списку подій, `/api/events/[id]` для роботи з окремими подіями та `/api/events/[id]/register` для реєстрації учасників. Валідація даних забезпечується через типи, згенеровані Prisma, та схеми Zod.

Ж.2 База даних та моделі даних

Схема Prisma включає понад 25 моделей з комплексними зв'язками, що покривають всю функціональність платформи. **Модель користувача** підтримує рольову систему доступу з переліченими значеннями VISITOR, ORGANIZER, VOLUNTEER, ADMIN та включає профільну інформацію, налаштування та дані гейміфікації.

Модель події зберігає повну інформацію про заходи включно з локацією, організатором, датами проведення, ціною та статусом. Зв'язки з моделями

Location, User та EventRegistration забезпечують цілісність даних та ефективні запити. Поля часових міток автоматично відстежують створення та оновлення записів.

Ж.3 Автентифікація та безпека

Автентифікація на основі сесій реалізована через власну систему з безпечною генерацією токенів та автоматичною ротацією для запобігання перехопленню сесій. Токени підписуються через алгоритм RSA з регулярною зміною ключів підпису.

Контроль доступу на основі ролей забезпечує деталізовані права доступу через функції проміжного програмного забезпечення, що перевіряють права користувача на кожному кінцевому пункті API. Адміністративні функції доступні тільки для ролі ADMIN, організаційні можливості - для ORGANIZER та вище.

Валідація даних здійснюється на кількох рівнях: схеми Zod для вхідних параметрів API, обмеження Prisma на рівні бази даних та бізнес-правила в сервісному шарі. Атаки міжсайтового скриптингу та впровадження SQL запобігаються через параметризовані запити та очищення користувацького вводу.

Обмеження швидкості реалізовано для захисту від атак відмови у обслуговуванні з різними лімітами для автентифікованих та анонімних користувачів. Чутливі операції (реєстрація, скидання пароля) мають додаткові обмеження з тимчасовою блокацією при перевищенні лімітів.

Ж.4 Система рекомендацій та алгоритми

Движок рекомендацій інтегрований у серверну частину через `/api/recommendations/events` для персоналізованих пропозицій подій та `/api/recommendations/similar/[id]` для пошуку схожих заходів. Алгоритм враховує історію користувача, геолокацію, часові переваги та подібність з іншими користувачами.

Конвеєр обробки даних обробляє поведінку користувачів через фонові завдання з періодичним перерахунком рекомендаційних моделей. Система

збирає неявний зворотний зв'язок (переглянуті події, час на сторінці, реєстрації) для покращення якості прогнозів без додавання навантаження на користувачів.

Стратегія кешування використовує Redis для зберігання попередньо розрахованих рекомендацій з часом життя на основі активності користувача. Активні користувачі отримують свіжіші рекомендації, тоді як нові користувачі використовують загальні трендові події з довшим кешем.

Ж.5 Шар інтеграції та зовнішні сервіси

Шар інтеграції сервісів абстрагує взаємодію з зовнішніми API через спеціалізовані клієнти з логікою повторних спроб та механізмами стійкості при недоступності сервісів. Кожна інтеграція має резервні механізми для забезпечення стабільності системи.

Інтеграція Google Maps забезпечує геокодування, зворотне геокодування та автозаповнення місць з кешуванням результатів для популярних запитів. Координація обмежень швидкості з квотами Google запобігає перевищенню лімітів та додатковим витратам.

Файлове сховище через MinIO забезпечує сумісний з S3 API для завантаження та зберігання зображень подій з автоматичною генерацією мініатюр та оптимізацією для різних розмірів екранів. Попередньо підписані URL забезпечують безпечне завантаження без розкриття облікових даних сховища.