

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

**«Програмно-інформаційний комплекс для
адміністрування та організації розважальних заходів»**

Виконав студент групи ПЗ-22-2
РУДЕНКО Олексій Іванович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«28» лютого 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Здобувач вищої освіти: **РУДЕНКО Олексій Іванович**

Керівник роботи: **ЛОКТИКОВА Тамара Миколаївна**

Тема роботи: **«Програмно-інформаційний комплекс для адміністрування та організації розважальних заходів»** ,

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

Вихідні дані для роботи: **об'єктом дослідження є процес організації та адміністрування розважальних заходів у програмно-інформаційному комплексі, а предметом дослідження – методи та засоби розробки програмно-інформаційного комплексу для створення, управління та відвідування подій, а також технології реалізації клієнтської, серверної та адміністративної частин вебзастосунку.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М	02.03.2026	02.03.2026
2	Локтікова Т.М	02.03.2026	02.03.2026
3	Локтікова Т.М	02.03.2026	02.03.2026

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-застосунку та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить текстову частину, викладену на 83 сторінках друкованого тексту, 38 ілюстрації, 39 таблиць. Список використаних джерел містить 30 найменувань і займає 2 сторінки. Також у роботі є 19 сторінок додатків. Загальний обсяг роботи складає 102 сторінки.

Мета роботи – розробка програмно-інформаційного комплексу для адміністрування та організації розважальних заходів, що забезпечує створення та керування подіями, їх редагування та пошук, реєстрацію користувачів на події, обробку оплат, а також адміністрування системи через відповідні інструменти керування.

В роботі використано методи веброзробки, проєктування баз даних, побудови REST API, автентифікації користувачів із використанням JWT-токенів, а також розробки адміністративного інтерфейсу для управління даними. Отримані результати включають спроектовану архітектуру програмно-інформаційного комплексу, реалізовану реляційну базу даних для зберігання інформації про користувачів, події, реєстрації, квитки, платежі та адміністративні сутності, а також реалізовані серверні модулі для управління подіями, категоріями, платежами та реєстраціями користувачів.

Рекомендовано впровадження розробленого програмно-інформаційного комплексу для автоматизації процесів організації розважальних заходів, підвищення ефективності взаємодії між організаторами та учасниками, а також спрощення адміністрування платформи та контролю за її роботою.

КЛЮЧОВІ СЛОВА: АВТЕНТИФІКАЦІЯ, АДМІНІСТРУВАННЯ, БАЗА ДАНИХ, КОРИСТУВАЧ, ПОДІЇ, СЕРВЕР, DJANGO, DRF, JWT, PYTHON, REST API.

					ІПЗ.КР.Б – 121 – 26 – ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата								
Розроб.		Руденко О.І.			Програмно-інформаційний комплекс для адміністрування та організації розважальних заходів			Літ.		Арк.	Аркушів	
Перевір.		Локтікова Т.М.								3	102	
Керівник		Локтікова Т.М.						Житомирська політехніка, група ІПЗ-22-2				
Н. контр		Тичина О.П.										
Зав.каф.		Вакалюк Т.А.										

ABSTRACT

The bachelor's qualification work consists of a web application and an explanatory note. The explanatory report contains a textual part presented on 83 pages of printed text, 38 illustrations, and 39 tables. The list of references includes 30 sources and occupies 2 pages. The thesis also includes 19 pages of appendices. The total volume of the work is 102 pages.

The purpose of this work is to develop a software and information system for the administration and organization of entertainment events, which provides functionality for creating and managing events, editing and searching events, registering users for events, processing payments, and administering the system through dedicated management tools.

The work employs methods of web development, database design, REST API development, user authentication using JWT tokens, and the development of an administrative interface for data management. The obtained results include a designed architecture of the software and information system, an implemented relational database for storing information about users, events, registrations, tickets, payments, and administrative entities, as well as implemented server-side modules for managing events, categories, payments, and user registrations.

The developed software and information system is recommended for automating the processes of organizing entertainment events, improving the efficiency of interaction between organizers and participants, and simplifying platform administration and operational control.

KEYWORDS: AUTHENTICATION, ADMINISTRATION, DATABASE, USER, EVENTS, SERVER, DJANGO, DRF, JWT, PYTHON, REST API.

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНО-ІНФОРМАЦІЙНОГО КОМПЛЕКСУ	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури програмно-інформаційного комплексу	19
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	23
Висновки до розділу 1	26
РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНО-ІНФОРМАЦІЙНОГО КОМПЛЕКСУ	28
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	28
2.2 Розробка бази даних.....	39
2.3 Проєктування та реалізація алгоритмів роботи системи	49
2.4 Реалізація функціоналу програмно-інформаційного комплексу	52
Висновки до розділу 2	55
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ.....	57
3.1 Порядок встановлення та налаштування системи	57
3.2 Структура інтерфейсу та порядок взаємодії із системою	60
3.3 Тестування роботи програмного продукту	76
Висновки до розділу 3	79
ВИСНОВКИ	80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	81
ДОДАТКИ	83

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

СУБД – система управління базами даних

ШІ – штучний інтелект

API – програмний інтерфейс застосунку (Application Programming Interface)

REST API – архітектурний стиль взаємодії клієнта та сервера через HTTP (Representational State Transfer Application Programming Interface)

JWT – токен доступу для автентифікації користувачів (JSON Web Token)

UX – досвід користувача (User Experience)

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol)

HTTPS – захищений протокол HTTP із використанням шифрування

ORM – технологія відображення об'єктів у базу даних (Object-Relational Mapping)

		Руденко О.І.			ІПЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми зумовлена активним розвитком цифрових сервісів у сфері організації подій та зростанням потреб користувачів у швидкому доступі до інформації про розважальні заходи. Сьогодні значна частина реєстрації на події, купівлі квитків та взаємодії між організаторами й відвідувачами здійснюється через веб платформи та мобільні застосунки. У зв'язку з цим виникає потреба у створенні сучасних інформаційних систем, які забезпечують зручне керування подіями, автоматизацію процесу реєстрації учасників, електронні квитки та обробку онлайн оплат.

Особливу роль відіграє використання сучасних вебтехнологій, які дозволяють створювати масштабовані та безпечні системи з підтримкою клієнт серверної взаємодії. Також актуальним є впровадження додаткових можливостей, зокрема системи сповіщень, рекламування подій та генерації контенту за допомогою штучного інтелекту.

Метою випускної кваліфікаційної роботи є розробка програмно інформаційного комплексу для організації та адміністрування розважальних заходів, який забезпечує створення подій, реєстрацію користувачів, придбання електронних квитків, взаємодію між організаторами та учасниками, а також управління рекламними можливостями платформи.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

- 1) провести аналіз предметної області та існуючих платформ для організації подій;
- 2) спроектувати архітектуру програмної системи та структуру бази даних;
- 3) розробити серверну частину системи з REST API для взаємодії з клієнтською частиною;
- 4) реалізувати систему авторизації та автентифікації користувачів;
- 5) створити функціонал роботи з подіями, категоріями, коментарями та відгуками;
- 6) інтегрувати систему онлайн оплат для придбання квитків та реклами подій;

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

7) розробити клієнтську частину застосунку для взаємодії користувачів із системою;

8) реалізувати систему сповіщень та генерації контенту за допомогою штучного інтелекту;

9) провести тестування створеного програмного забезпечення.

Об'єктом дослідження є процес організації та адміністрування розважальних заходів у програмно-інформаційному комплексі.

Предметом дослідження є методи та засоби розробки програмно-інформаційного комплексу для створення, управління та відвідування подій, а також технології реалізації клієнтської, серверної та адміністративної частин вебзастосунку.

За темою випускної кваліфікаційної роботи бакалавра були подані та прийняті до публікації тези: Руденко О.І., Лімінович І.Д., Локтікова Т.М. Розробка серверної частини платформи для управління розважальними заходами. Всеукраїнська науково-практична он-лайн конференція здобувачів вищої освіти і молодих учених, присвячена Дню науки (15 травня 2026 року) [1].

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНО-ІНФОРМАЦІЙНОГО КОМПЛЕКСУ

1.1 Постановка задачі

Основною метою кваліфікаційної роботи є розробка програмно інформаційного комплексу для організації та адміністрування розважальних заходів, який забезпечує зручну та безпечну взаємодію між користувачами та організаторами подій, а також стабільну роботу серверної та клієнтської частин системи. Використання платформи повинно бути інтуїтивно зрозумілим і не викликати труднощів у користувачів.

Розробка системи базується на сучасних підходах до створення вебзастосунків, що дозволяє отримати надійний та масштабований програмний продукт. Усі етапи реалізації виконуються послідовно, починаючи від проєктування інтерфейсу і завершуючи розгортанням системи у хмарному середовищі.

В першу чергу передбачається розробка користувацького інтерфейсу. Він включає головну сторінку з переліком подій, сторінки реєстрації та авторизації, сторінку перегляду детальної інформації про подію, а також розділи для організаторів, де вони можуть створювати та керувати подіями. Окремо реалізується функціонал перегляду та фільтрації подій, що дозволяє користувачам швидко знаходити необхідні заходи за різними параметрами, такими як категорія, дата проведення, місце розташування або вартість участі. Також передбачено відображення актуальної інформації про популярні та рекомендовані події.

У серверній частині системи реалізовано основні модулі, які забезпечують повноцінну роботу платформи. До них належать модуль користувачів, модуль подій, модуль реєстрацій, модуль оплат, модуль рекламних можливостей та модуль сповіщень. Кожен з цих модулів виконує окрему роль у системі та взаємодіє з іншими через API.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

Модуль користувачів відповідає за реєстрацію, авторизацію та управління профілями. Він забезпечує розмежування прав доступу між звичайними користувачами та організаторами, а також збереження персональної інформації.

Модуль подій реалізує створення, редагування та видалення подій, а також їх пошук і фільтрацію. Він є основною частиною системи, оскільки саме через нього користувачі взаємодіють із контентом платформи.

Модуль реєстрацій відповідає за запис користувачів на події та збереження відповідних даних у системі. Він забезпечує контроль кількості учасників та доступ до подій.

Модуль оплат забезпечує обробку фінансових операцій, пов'язаних із купівлею квитків та додатковими послугами. Він гарантує коректну взаємодію з платіжними сервісами та збереження інформації про транзакції.

Модуль рекламних можливостей дозволяє організаторам просувати свої події через спеціальні банери та промо блоки, що відображаються на платформі.

Модуль сповіщень відповідає за інформування користувачів про важливі події, такі як нові реєстрації, зміни у подіях або інші системні повідомлення.

Після реалізації основних модулів важливим етапом є забезпечення взаємодії між серверною та клієнтською частинами системи. Для цього використовується REST API [2], який забезпечує обробку запитів та обмін даними між частинами застосунку. Такий підхід дозволяє реалізувати гнучку архітектуру системи, спрощує підтримку програмного продукту та створює можливість для інтеграції з іншими сервісами у майбутньому.

Окремим етапом розробки є тестування програмного продукту. Воно включає перевірку коректності роботи функціональних модулів, тестування взаємодії між компонентами системи та оцінку зручності використання інтерфейсу. Проведення тестування дозволяє виявити та усунути можливі помилки ще до введення системи в експлуатацію.

У результаті виконання роботи створюється повноцінна веб платформа , яка дозволяє користувачам знаходити та відвідувати події, а організаторам ефективно керувати своїми заходами. Система забезпечує зручний інтерфейс,

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

надійну роботу серверної частини та можливість подальшого розширення функціоналу.

1.2 Аналіз аналогів програмного продукту

Розробка програмно-інформаційного комплексу для управління розважальними заходами потребує детального аналізу існуючих рішень на ринку. У сучасних умовах сфера організації подій активно розвивається, що сприяє появі великої кількості онлайн-платформ для створення, управління та перегляду заходів.

Незважаючи на значну кількість готових рішень, багато з них мають певні недоліки. Частина платформ перевантажена функціоналом, що ускладнює їх використання, тоді як інші мають обмежені можливості або недостатньо зручну взаємодію через API. Тому доцільно провести аналіз існуючих систем для визначення їх переваг і недоліків.

Одним із українських аналогів є 2Event [3], що поєднує функції квиткового сервісу та соціальної мережі для нетворкінгу.

На головній сторінці представлено каталог подій (рис. 1.1) у вигляді карток, які містять основну інформацію про захід: назву, дату, місце проведення та вартість квитків. Також передбачено можливість швидкого створення нової події через відповідну кнопку в інтерфейсі.

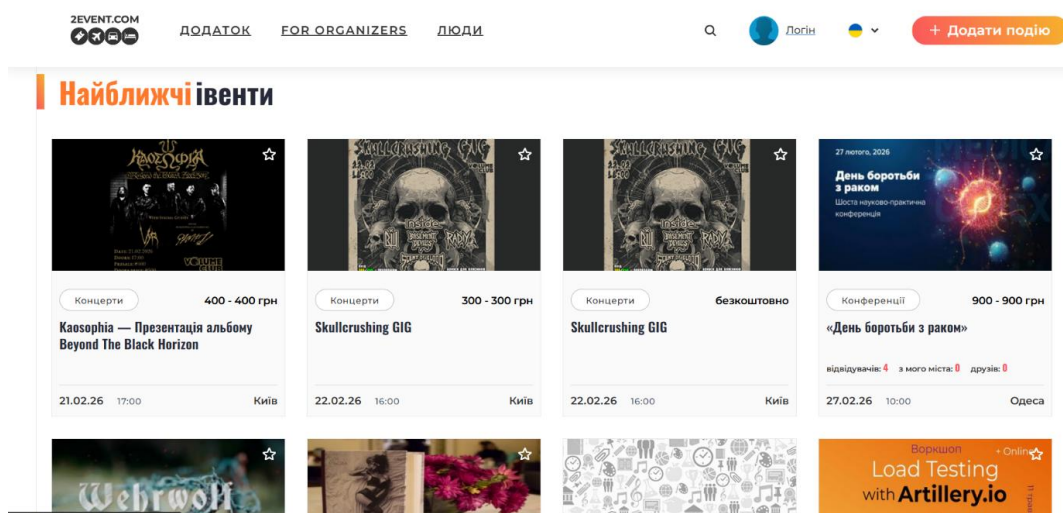


Рис. 1.1. Головна сторінка 2Event

		Руденко О.І.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Після вибору події користувач переходить на сторінку її детального перегляду (рис. 1.2), де відображаються назва, опис, дата, місце проведення та інша інформація, необхідна для ознайомлення із заходом і прийняття рішення щодо участі.

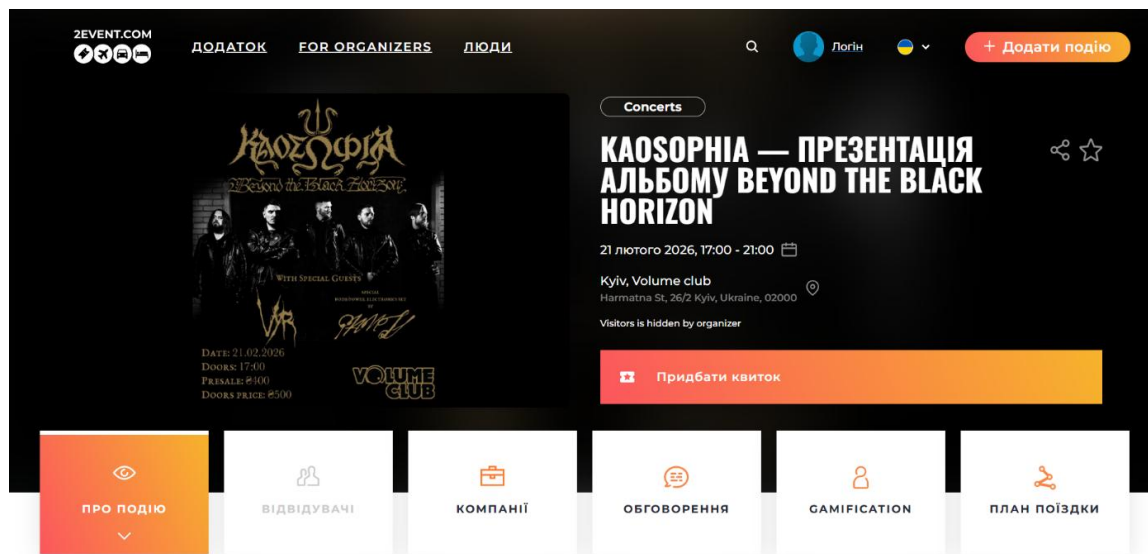


Рис. 1.2. Сторінка детальної інформації про захід

При реєстрації пропонується екран вибору інтересів (рис. 1.3). Система збирає дані про вподобання користувача (бізнес, ІТ, спорт тощо) для подальшого формування персоналізованих рекомендацій у стрічці подій.

What are your interests?

Виберіть не менше 5 категорій, які відповідають самим цікавим подіям для вас

Бізнес

Tech & IT

Спорт

Сім'я

Концерти

Нічне життя

Культура

Туризм

Розвиток

Відпочинок

Виставки

Фестивалі

Розваги

Соціальні події

Освіта

Тренінги

Прес-конференція

Наука

Медицина

Blockchain

Психологія

Юриспруденція

Їжа та напої

Фотографії

Домашні тварини

Мода

Мистецтво

Діти

Політика

Бухгалтерія

Релігія

Для організаторів подій

B2B-Networking

18+

Рис. 1.3. Модуль персоналізації при реєстрації

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

На рис. 1.4 зображено форму, яка дозволяє будь-якому зареєстрованому користувачу перейти до ролі організатора. Функціонал реалізовано через розширений набір полів введення (inputs) та контролерів

Рис. 1.4. Інтерфейс створення події

Для зручності користувачів на сайті є блок "Переглядали раніше" (рис. 1.5). Він автоматично зберігає історію переглядів, щоб можна було швидко повернутися до цікавої події, не шукаючи її заново.

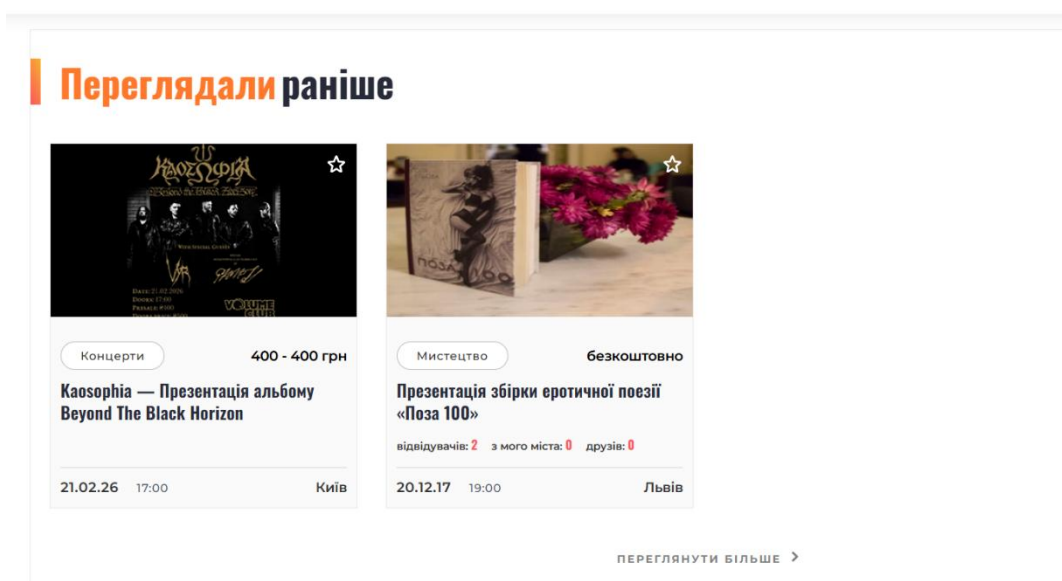


Рис. 1.5. Блок ретаргетингу «Переглядали раніше»

		Руденко О.І.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Другим аналогом розглянемо Eventbrite [4], який є світовим лідером серед сервісів для івентів. Окрім стандартних функцій, він використовує сучасні технології для автоматизації роботи організатора.

На відміну від українських сервісів, Eventbrite пропонує функцію «Create an event with AI» (рис. 1.6). Користувачу достатньо ввести назву та дату, а штучний інтелект сам генерує опис, підбирає категорію та теги. Це значно економить час організатора.

Create an event with AI

Answer a few questions about your event and our AI creation tool will use internal data to build an event page. You can still [create an event without AI](#).

What's the name of your event?

This will be your event's title. Your title will be used to help create your event's summary, description, category, and tags – so be specific!

Event title *

Reading test event

When does your event start and end?

Date

03/28/2026

Start time

10:00 AM

End time

12:00 PM

Where is it located?

Venue

Online event

To be announced

Рис. 1.6. Інтерфейс створення події за допомогою ШІ

Платформа Eventbrite має вбудований аналітичний модуль, що дозволяє організаторам переглядати статистику продажу квитків та відвідуваності заходів. Даний інструмент призначений для оцінювання ефективності подій.

Під час проведених огляду й аналізу встановлено, що за відсутності проданих квитків статистичні дані не відображаються. У такому випадку система

показує повідомлення (рис. 1.7): «Nothing to report. Your reports will be available after you sell some tickets», що означає доступність звітів лише після появи фактичних даних про продаж.



Nothing to report

Your reports will be available after you sell
some tickets.

Рис. 1.7. Інтерфейс аналітичного модуля Eventbrite за відсутності проданих квитків

Таким чином, аналітичний модуль є функціонально повноцінним, однак його можливості розкриваються лише за наявності фактичних даних про продаж квитків. Отримана інформація може використовуватися організаторами для оцінювання ефективності проведених заходів, аналізу активності аудиторії та прийняття подальших управлінських рішень. Це підтверджує орієнтацію сервісу на комерційне використання та реальні події з активною аудиторією.

Окрему увагу в Eventbrite приділено модулю налаштування квитків (рис. 1.8). Платформа дозволяє організатору обрати тип квитка залежно від формату заходу та бізнес-моделі події. Передбачено створення платних, безкоштовних квитків, а також квитків формату «Donation», що дає змогу учасникам самостійно визначати суму внеску.

		Руденко О.І.			ІПЗ.КР.Б – 121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Create tickets

Choose a ticket type or build a section with multiple ticket types.



Paid
Create a ticket that people have to pay for.

>



Free
Create a ticket that no one has to pay for.

>



Donation
Let people pay any amount for their ticket.

>

Create a section if you want to sell multiple ticket types that share the same inventory. (ex. Floor, Mezzanine, etc.)

Create a section

Рис. 1.8. Вибір типу квитків під час створення події в Eventbrite

Наступним проаналізованим сервісом є Meetup [5] — міжнародна платформа для організації зустрічей за інтересами. Основною особливістю сервісу є те, що базовою структурною одиницею системи виступає спільнота користувачів, а не окрема подія.

Кожна група об'єднує користувачів за спільними інтересами та може регулярно організовувати заходи. На сторінці спільноти відображається її назва, рейтинг, кількість учасників, місце розташування, опис діяльності, організатори, а також перелік минулих і майбутніх подій (рис. 1.9). Користувач має можливість приєднатися до групи, після чого отримує сповіщення про нові заходи та може брати участь в обговореннях.

На відміну від Eventbrite, де акцент зроблено переважно на створенні окремих подій, Meetup орієнтований на довготривалу взаємодію користувачів у межах спільнот. Таким чином, платформа виконує не лише функцію реєстрації на заходи, а й роль соціальної мережі для професійних та тематичних груп.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

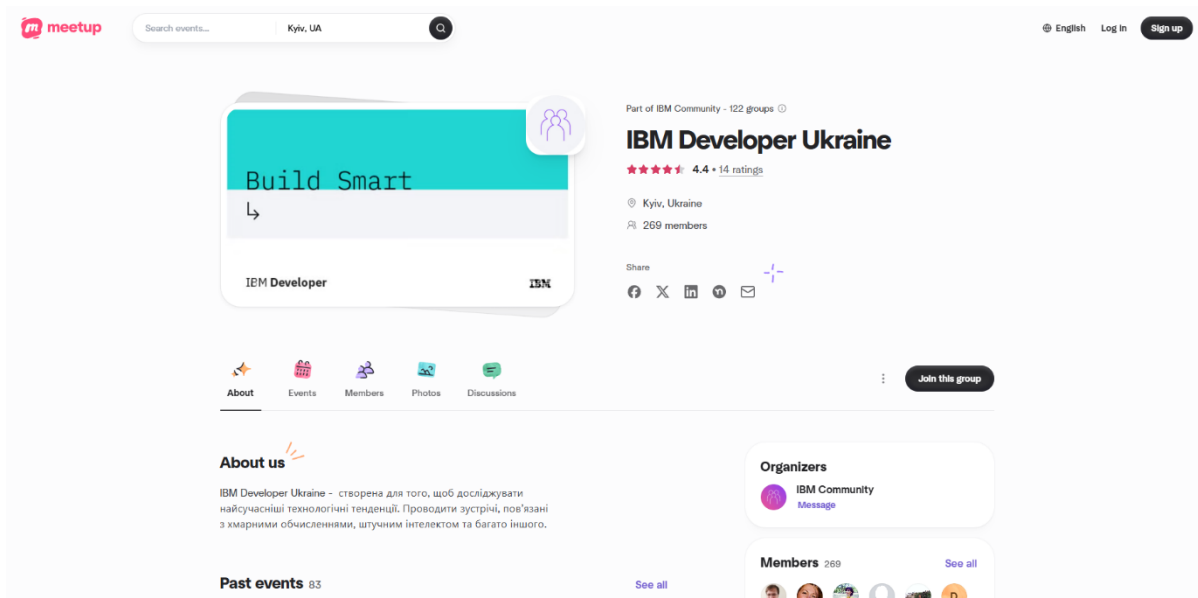


Рис. 1.9. Сторінка спільноти в сервісі Meetup

У вкладці Discussions реалізовано можливість обговорення тем, пов'язаних із діяльністю групи (рис. 1.10). Учасники можуть ставити запитання, ініціювати дискусії та обмінюватися досвідом. Такий функціонал підсилює соціальну складову сервісу, оскільки комунікація між учасниками відбувається постійно, а не лише під час проведення заходів.

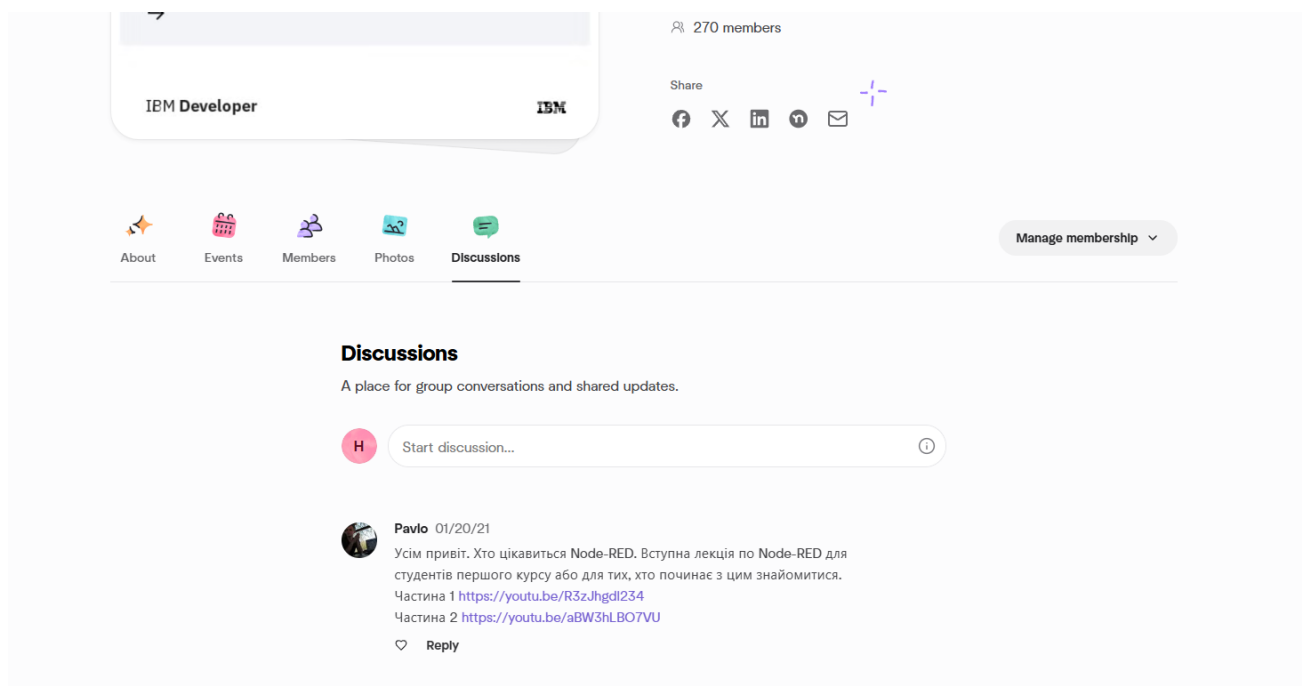


Рис. 1.10. Розділ Discussions у спільноті Meetup

Після детального розгляду функціональних можливостей досліджуваних платформ доцільно узагальнити результати у вигляді табл. 1.1.

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1

Порівняльний аналіз платформ

Критерій	2Event	Eventbrite	Meetup
Створення подій	✓	✓	✓
Пошук та фільтрація подій	✓	✓ (розширений)	✓
Реєстрація користувачів	✓	✓	✓
Ролі користувачів (учасник / організатор / адмін)	частково	✓	частково
Управління подіями (CRUD)	✓	✓ (розширене)	частково
Система квитків	базова	розвинена	відсутня / обмежена
Онлайн оплата	частково	✓	відсутня
Аналітика для організаторів	базова	розвинена	базова
Рекламні можливості подій	✓	✓	відсутні
Персоналізація рекомендацій	✓	✓	✓ (через групи)
Соціальна взаємодія між користувачами	частково	мінімальна	розвинена
Система сповіщень	✓	✓	✓

API для інтеграції	обмежене	✓	обмежене
Рівень автоматизації процесів	середній	високий	середній
Зручність інтерфейсу	середня	складна для нових користувачів	проста

Порівняння розглянутих рішень показує, що кожна платформа має свою спеціалізацію. Одні сервіси більше орієнтовані на соціальну взаємодію, інші на продаж квитків або організацію бізнес подій. Водночас не існує універсального рішення, яке б поєднувало зручний інтерфейс, розвинену систему подій, автоматизацію процесів, рекламні можливості та сучасні інтеграції в межах однієї системи.

Таким чином, можна зробити висновок, що існуючі аналоги не повністю задовольняють усі потреби користувачів та організаторів подій. Це створює передумови для розробки нової платформи, яка об'єднує основні функції аналогів та доповнює їх розширеними можливостями. Зокрема, у системі передбачено гнучке управління подіями, систему онлайн оплат, рекламні механізми, персоналізовані сповіщення та використання сучасних технологій для автоматизації створення контенту.

Розроблювана система у даному контексті розглядається як більш комплексне рішення, яке спрямоване на усунення обмежень існуючих платформ та забезпечення зручної взаємодії між усіма учасниками процесу організації заходів.

1.3 Вибір архітектури програмно-інформаційного комплексу

Вибір архітектури системи є одним із ключових етапів проєктування, оскільки саме він визначає стабільність роботи системи, можливості

		Руденко О.І.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

масштабування та зручність подальшого розвитку проєкту. Для платформи, яка передбачає взаємодію великої кількості користувачів, організаторів та адміністратора, важливо забезпечити чітку структуру системи та ефективний обмін даними між її компонентами.

У межах розробки було обрано клієнт-серверну архітектуру з використанням підходу REST API. Така модель передбачає розділення системи на дві основні частини: клієнтську, яка відповідає за відображення інтерфейсу користувача, та серверну, яка виконує всю бізнес-логіку, обробку даних і взаємодію з базою даних. Такий поділ дозволяє зробити систему більш гнучкою, спрощує масштабування та дає можливість незалежно розвивати окремі частини застосунку.

Серверна частина забезпечує централізовану обробку даних та координує роботу всіх функціональних модулів системи. До її основних завдань належать управління користувачами, подіями, платежами, сповіщеннями та іншими процесами, необхідними для функціонування платформи. Завдяки централізованій обробці інформації забезпечується цілісність даних та контроль доступу до ресурсів системи.

Особлива увага приділена безпеці доступу до функціональних можливостей платформи. Для цього використовується механізм автентифікації користувачів та розмежування прав доступу. Такий підхід дозволяє контролювати дії різних категорій користувачів та обмежувати доступ до службових функцій системи.

Клієнтська частина системи реалізується як окремий вебзастосунок, що забезпечує взаємодію користувача з платформою. Через нього користувач може переглядати перелік подій, здійснювати пошук та фільтрацію заходів, проходити реєстрацію, керувати власним профілем і взаємодіяти з іншими можливостями системи. Усі необхідні дані отримуються із серверної частини через прикладний програмний інтерфейс.

Взаємодія між компонентами системи здійснюється шляхом надсилання та обробки HTTP-запитів. Обмін інформацією між клієнтською та серверною

		Руденко О.І.			ІПЗ.КР.Б – 121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

частинами відбувається за допомогою формату JSON, що забезпечує сумісність між різними програмними платформами та спрощує інтеграцію із зовнішніми сервісами. Для передачі файлів використовується формат multipart/form-data, що дозволяє завантажувати зображення подій та аватари користувачів. Такий підхід забезпечує незалежність окремих частин проєкту та дозволяє в майбутньому розширювати функціональність без суттєвих змін у вже реалізованих модулях.

Для підвищення зручності супроводу система побудована за модульним принципом. Окремі модулі відповідають за роботу з користувачами, подіями, платежами, сповіщеннями та рекламними матеріалами. Завдяки цьому кожна функціональна частина може розвиватися незалежно від інших, що значно спрощує підтримку програмного забезпечення.

Однією з особливостей архітектури системи є використання сервісного шару (Service Layer), який забезпечує відокремлення бізнес-логіки від рівня обробки HTTP-запитів. Усі ключові процеси, пов'язані з реєстрацією користувачів на події, обробкою платежів, формуванням статистики, створенням електронних квитків та надсиланням сповіщень, реалізовані у спеціалізованих сервісних модулях. Рівень представлення відповідає лише за приймання та обробку запитів, тоді як основні правила роботи системи зосереджені в сервісах. Такий підхід підвищує якість програмного коду, спрощує його тестування та забезпечує можливість повторного використання функціональних компонентів.

Для реалізації серверної частини використовується фреймворк Django та Django REST Framework, які реалізують архітектурний підхід MTV (Model–Template–View). Основні сутності системи представлені моделями, що взаємодіють із базою даних через ORM-механізм. Для побудови прикладного програмного інтерфейсу використовуються класи ViewSet та APIView. Фактично архітектура системи поєднує React-клієнт, REST API, сервісний шар та рівень моделей даних, що забезпечує чіткий розподіл відповідальності між компонентами.

Для підвищення продуктивності роботи застосунку були використані засоби оптимізації доступу до даних. Під час формування списків подій

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

застосовуються механізми попереднього завантаження пов'язаних об'єктів та агрегування даних на рівні бази даних. Крім того, окремі статистичні показники кешуються, що дозволяє зменшити навантаження на сервер при повторних зверненнях користувачів. Для роботи з великими наборами даних використовується посторінкове відображення результатів, що забезпечує стабільну роботу системи навіть за умови значного збільшення кількості подій та користувачів.

Особлива увага приділена питанням безпеки. Для автентифікації використовується механізм JWT-токенів, що забезпечує безпечний доступ до захищених ресурсів системи. Додатково реалізовано перевірку прав доступу для різних категорій користувачів, валідацію вхідних даних та обмеження частоти виконання окремих операцій. Для фінансових операцій застосовується механізм підтвердження платежів через вебхуки, що дозволяє виключити можливість несанкціонованого підтвердження оплати з боку клієнтської частини.

Архітектура також передбачає можливість інтеграції із зовнішніми сервісами. Для реалізації електронних платежів використовується сервіс Stripe, який забезпечує проведення фінансових операцій та автоматичне підтвердження оплат. Формування текстового та графічного контенту для подій здійснюється із залученням сервісів штучного інтелекту. Для зберігання медіафайлів використовується окреме хмарне сховище, що дозволяє зменшити навантаження на сервер та забезпечити надійне зберігання користувацьких даних. Також у системі передбачено інтеграцію із сервісами електронної пошти для підтвердження реєстрації користувачів та відновлення доступу до облікового запису.

З точки зору організації програмного забезпечення система реалізована у вигляді модульного моноліту. Усі функціональні підсистеми працюють у межах єдиного серверного застосунку та використовують спільну базу даних, однак логічно розділені на незалежні програмні модулі. Такий підхід дозволяє уникнути складності, характерної для мікросервісної архітектури, водночас забезпечуючи достатній рівень масштабованості, простоту розгортання та

		Руденко О.І.			ІПЗ.КР.Б – 121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

підтримки програмного продукту. Обрана архітектура є оптимальною для реалізації веборієнтованої платформи управління подіями та відповідає вимогам до сучасних інформаційних систем.

Таким чином, обрана архітектура веборієнтованої системи забезпечує розподіл функцій між компонентами, ефективний обмін даними та можливість подальшого розвитку проєкту. Використаний підхід дозволяє реалізувати всі основні функції платформи, забезпечити її надійність, безпечність та продуктивність, а також створює основу для стабільної роботи системи в умовах зростання кількості користувачів і функціональних можливостей..

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для реалізації програмно-інформаційного комплексу було обрано сучасний стек інструментальних засобів, який забезпечує високу продуктивність, масштабованість, безпеку та зручність подальшого розвитку системи. Вибір технологій базувався на вимогах до побудови вебзастосунку з клієнт-серверною архітектурою, використанням REST API, обробкою великої кількості користувацьких запитів та інтеграцією сторонніх сервісів [6]. Обраний набір технологій також враховує необхідність підтримки фінансових операцій, роботи з мультимедійним контентом та забезпечення стабільної взаємодії між компонентами системи.

Серверна частина системи реалізована на мові програмування Python [7] із використанням вебфреймворку Django та розширення Django REST Framework [8]. Вибір даного стеку зумовлений його високою гнучкістю, зрілістю екосистеми та наявністю готових рішень для реалізації вебзастосунків. Django дозволяє будувати монолітну, але модульну архітектуру, в якій логічно виділені окремі компоненти системи. Така структура забезпечує чіткий поділ відповідальностей між модулями, спрощує супровід коду та дозволяє

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

масштабувати окремі функціональні частини системи без зміни загальної архітектури.

Використання Django REST Framework обумовлено необхідністю реалізації REST API, який забезпечує ефективну взаємодію між серверною та клієнтською частинами системи. Такий підхід є доцільним для SPA-архітектури, у якій клієнтська частина не містить бізнес-логіки, а виконує лише функції отримання, відображення та відправки даних через HTTP-запити.

Для забезпечення безпечної автентифікації користувачів використовується механізм JSON Web Token [9]. Вибір JWT обумовлений необхідністю реалізації stateless-автентифікації, що дозволяє уникнути зберігання сесійних даних на сервері та підвищує масштабованість системи. У межах системи використовуються access-токени з обмеженим терміном дії та refresh-токени, що забезпечує баланс між безпекою та зручністю користувацької взаємодії.

Клієнтська частина системи реалізується за допомогою бібліотеки React, яка є одним із найпоширеніших інструментів для створення сучасних вебінтерфейсів. Вибір React зумовлений необхідністю реалізації динамічного інтерфейсу користувача, який активно взаємодіє з серверною частиною через REST API та оперує великою кількістю змінних даних, зокрема списками подій, квитками, платежами та користувацькими діями. Компонентний підхід React дозволяє будувати інтерфейс як набір незалежних і перевикористовуваних модулів, що спрощує масштабування системи та її подальший супровід.

Додатковою перевагою використання React є ефективне оновлення DOM за рахунок віртуального DOM, що забезпечує високу продуктивність при частих змінах стану інтерфейсу, характерних для системи. Використання SPA-підходу дозволяє мінімізувати кількість перезавантажень сторінок і забезпечує більш плавну та швидку взаємодію користувача із системою, що є критично важливим для платформ, орієнтованих на перегляд і створення подій у реальному часі.

Для зберігання даних використовується система керування базами даних PostgreSQL. Вибір даної СКБД зумовлений тим, що система EventHub має велику кількість взаємопов'язаних сутностей, таких як користувачі, події, квитки,

		Руденко О.І.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

платежі та відгуки, що вимагає використання реляційної моделі даних із забезпеченням цілісності та узгодженості інформації. PostgreSQL є оптимальним рішенням для таких задач завдяки підтримці транзакційності, що є критично важливою для коректної обробки фінансових операцій, зокрема оплати квитків та рекламних послуг.

Додатковою перевагою PostgreSQL є висока продуктивність при виконанні складних SQL-запитів та можливість оптимізації через індексування, що дозволяє ефективно працювати з великими обсягами даних. Також важливим фактором вибору є тісна інтеграція з Django ORM, що спрощує взаємодію з базою даних, зменшує кількість низькорівневого SQL-коду та підвищує безпеку роботи з даними.

Зберігання медіафайлів, зокрема зображень подій, реалізовано через об'єктне хмарне сховище Cloudflare R2 [10]. Вибір даного рішення обумовлений необхідністю ефективного розвантаження основного сервера та забезпечення швидкого доступу до статичного контенту незалежно від навантаження на систему.

Для автоматизації генерації контенту використовується зовнішній сервіс штучного інтелекту Google Gemini, який застосовується для створення текстових описів подій. Окрім цього, інтегровано модель генерації зображень Hugging Face FLUX, яка використовується для автоматичного створення обкладинок подій. Використання таких інструментів дозволяє значно скоротити час підготовки контенту та підвищити зручність для організаторів.

Окремим важливим компонентом системи є адміністративна панель, реалізована на базі стандартних засобів Django із використанням розширення Django Unfold [11]. Вибір цього рішення зумовлений необхідністю покращення користувацького інтерфейсу адміністративної частини та підвищення зручності роботи з даними. Через адміністративну панель здійснюється керування користувачами, подіями, категоріями, платежами та рекламними матеріалами, а також виконується модерація контенту.

		Руденко О.І.			ІІЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

У процесі розробки використовувалися сучасні середовища розробки, зокрема PyCharm та Visual Studio Code, а також система контролю версій Git. Використання Git дозволило організувати ефективну роботу над проектом, забезпечити відстеження змін у коді та підтримувати узгоджену розробку функціональних модулів.

Для стабільної роботи системи визначено вимоги до апаратного забезпечення. Для середовища розробки достатньо персонального комп'ютера з дво- або чотириядерним процесором, не менше 8 ГБ оперативної пам'яті та стабільним доступом до мережі Інтернет. Для коректного виконання серверної частини системи достатньо конфігурації з 2–4 ГБ оперативної пам'яті, що забезпечує обробку користувацьких запитів та виконання бізнес-логіки.

Користувачі можуть взаємодіяти із системою через сучасні веббраузери без необхідності встановлення додаткового програмного забезпечення, що забезпечує високу доступність та зручність використання платформи.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було проведено аналіз предметної області та обґрунтовано актуальність розробки програмно-інформаційного комплексу для адміністрування та організації розважальних заходів. Дослідження показало, що сучасні системи івент-менеджменту відіграють важливу роль у забезпеченні взаємодії між організаторами та користувачами, а також у спрощенні процесів створення, пошуку та управління подіями. Було сформовано мету та основні завдання роботи, які полягають у розробці серверної частини вебзастосунку з використанням сучасних підходів до побудови API та системи автентифікації користувачів.

У ході роботи було проведено детальний аналіз існуючих програмних аналогів, зокрема платформ 2Event, Eventbrite та Meetup. Було встановлено, що частина платформ характеризується надмірною складністю інтерфейсу та перевантаженістю функціоналом, тоді як інші мають обмежені можливості та

		Руденко О.І.			ІІЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

недостатню гнучкість. Це підтвердило доцільність розробки власного рішення, яке поєднуватиме простоту використання та необхідний функціонал.

На основі проведеного аналізу було визначено основні вимоги до розроблюваної системи, зокрема необхідність реалізації базових функцій управління подіями, підтримки системи користувачів та забезпечення зручного пошуку і фільтрації. Особлива увага приділяється побудові ефективної серверної архітектури, яка забезпечує обробку запитів, управління даними та взаємодію з клієнтською частиною через REST API.

У процесі виконання першого розділу також було здійснено вибір інструментальних засобів розробки. Для реалізації серверної частини обрано мову програмування Python та фреймворк Django разом із Django REST Framework, що дозволяє створити масштабований та структурований веб-застосунок. Для забезпечення безпечного доступу до системи використовується механізм автентифікації на основі JSON Web Token. Зберігання даних реалізується за допомогою хмарної бази даних PostgreSQL, а для роботи з медіафайлами використовується об'єктне сховище Cloudflare R2. Окрему увагу приділено адміністративній панелі, реалізованій із використанням Django Unfold, яка спрощує управління даними системи.

Таким чином, у першому розділі було сформовано теоретичну та аналітичну основу для подальшої розробки системи, визначено основні вимоги до функціоналу та обрано технологічний стек. Отримані результати дозволяють перейти до наступного етапу кваліфікаційної роботи — проєктування архітектури системи та реалізації її основних компонентів.

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНО-ІНФОРМАЦІЙНОГО КОМПЛЕКСУ

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

Програмно-інформаційний комплекс для адміністрування та організації розважальних заходів надає інтерфейс, через який користувачі можуть переглядати події, реєструватися на івенти, купувати квитки та керувати профілем і бронюваннями. Для організаторів передбачено можливість створення та редагування заходів, налаштування їх параметрів і відстеження кількості учасників. Також реалізовано адміністративну панель для управління користувачами, подіями та квитками, перегляду статистики системи, а також контролю активності користувачів і фінансових операцій. Вона містить інструменти модерації, що дозволяють оперативно реагувати на порушення та забезпечувати стабільну роботу платформи. У зв'язку з цим користувачі розподіляються за рівнем доступу, що дозволяє розмежувати їхні можливості та забезпечити безпечну взаємодію із системою.

Короткий опис акторів представлено в табл. 2.1.

Таблиця 2.1

Визначення акторів системи

Актор	Короткий опис
Організатор	Зареєстрований користувач, який створює та публікує власні події на платформі. Має можливість додавати нові івенти, редагувати їх, керувати квитками та переглядати статистику відвідуваності своїх заходів
Адміністратор	Користувач із розширеними правами доступу. Має доступ до адміністративної панелі, де може переглядати статистику системи, а також повністю модерувати контент

Користувач	Людина, яка реєструється для перегляду та участі в подіях. Після реєстрації може керувати профілем, переглядати й реєструватися на івенти, ставити вподобання, залишати коментарі та відгуки, купувати квитки й переглядати історію бронювань
------------	---

Визначивши акторів системи, побудуємо діаграму варіантів використання, яка ілюструє їх взаємодію із застосунком. На рис. 2.1 показано діаграму використання акторів системи.

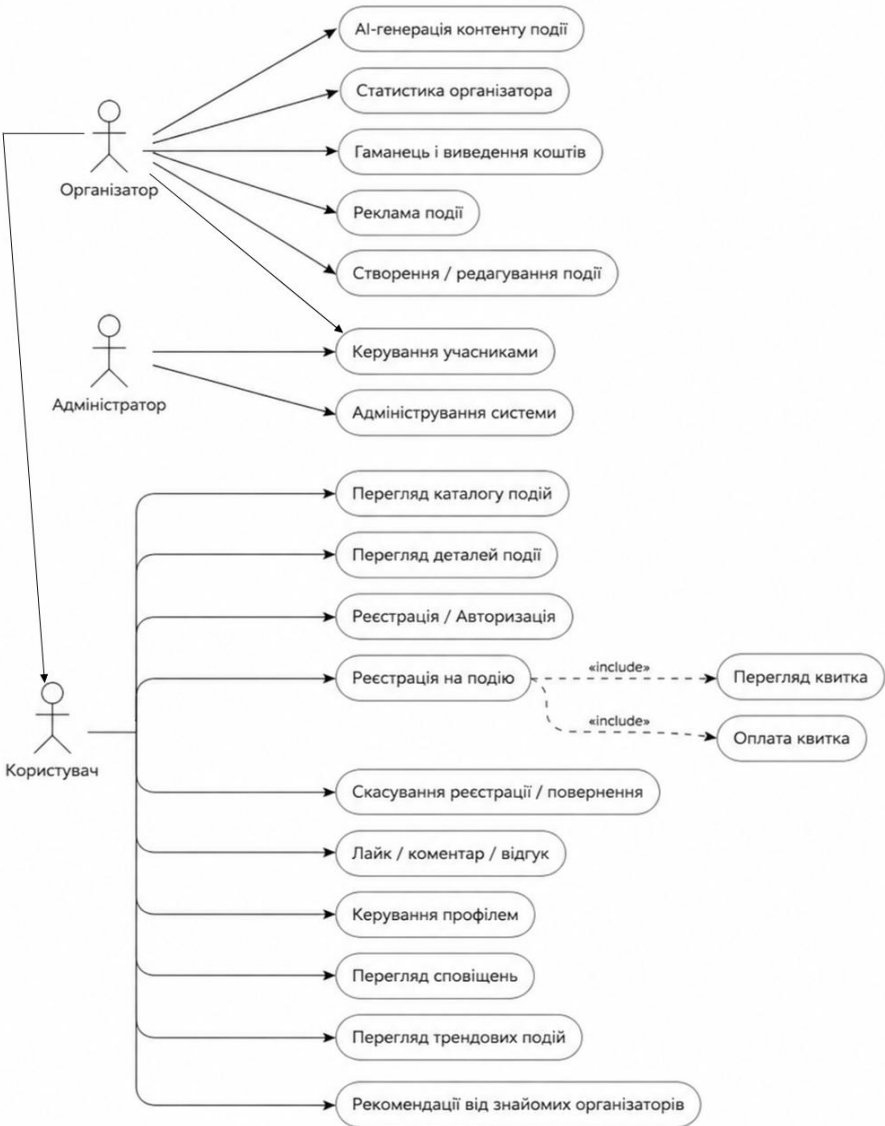


Рис. 2.1. Діаграма варіантів використання системи

Опис діаграми варіантів використання наведено в табл. 2.2 – 2.16.

Таблиця 2.2

Опис прецеденту «Авторизація користувача та профіль»

Назва варіанту використання	Реєстрація / Авторизація
Основні актори	Користувач
Короткий опис	Користувач створює обліковий запис через email або входить у систему за email і паролем для доступу до профілю, квитків і реєстрації на події

Таблиця 2.3

Опис прецеденту «Перегляд каталогу подій»

Назва варіанту використання	Перегляд каталогу подій
Основні актори	Користувач
Короткий опис	Користувач переглядає список подій на головній сторінці з можливістю фільтрації за містом, категоріями, пошуком і сортуванням

Таблиця 2.4

Опис прецеденту «Перегляд трендових та рекомендованих подій від організаторів»

Назва варіанту використання	Перегляд трендових та рекомендованих подій від організаторів
Основні актори	Користувач

продовж. табл. 2.4

Короткий опис	Користувач переглядає на головній сторінці популярні майбутні події, відсортовані за рейтингом активності , а також персональні рекомендації — майбутні події організаторів, на чиїх минулих заходах він уже був
---------------	--

Таблиця 2.5

Опис прецеденту «Перегляд деталей події»

Назва варіанту використання	Перегляд деталей події
Основні актори	Користувач
Короткий опис	Користувач відкриває сторінку події та переглядає детальну інформацію про захід, зокрема його опис, дату проведення, локацію, вартість квитка, коментарі та відгуки інших учасників, а також кількість зареєстрованих відвідувачів

Таблиця 2.6

Опис прецеденту «Реєстрація на подію та оплата квитка»

Назва варіанту використання	Реєстрація на подію та оплата квитка
Основні актори	Користувач
Короткий опис	Користувач записується на подію: для безкоштовної реєстрація виконується одразу після підтвердження участі, тоді як для платної події необхідно попередньо здійснити оплату квитка через платіжну систему

Таблиця 2.7

Опис прецеденту «Скасування реєстрації / повернення»

Назва варіанту використання	Скасування реєстрації / повернення
Короткий опис	Користувач скасовує реєстрацію на подію; для оплаченого квитка обирає повернення на внутрішній баланс або на банківську картку

Таблиця 2.8

Опис прецеденту «Лайк/коментар/відгук»

Назва варіанту використання	Лайк/коментар/відгук
Основні актори	Користувач
Короткий опис	Користувач може поставити лайк, залишити коментар або відгук до події, а також оцінити її за шкалою від 1 до 5, взаємодіючи з контентом та висловлюючи свою думку щодо заходу

Таблиця 2.9

Опис прецеденту «Керування профілем»

Назва варіанту використання	Керування профілем
Основні актори	Користувач
Короткий опис	Користувач редагує особисті дані (ім'я, нікнейм, біо, аватар, email, пароль), переглядає записані події, відстежує свою активність у системі та може за потреби видалити акаунт

Таблиця 2.10

Опис прецеденту «Перегляд сповіщень»

Назва варіанту використання	Перегляд сповіщень
Короткий опис	Користувач переглядає in-апп сповіщення про лайки, коментарі та інші події, пов'язані з його активністю на платформі

Таблиця 2.11

Опис прецеденту «Створення / редагування події»

Назва варіанту використання	Створення / редагування події
Основні актори	Організатор
Короткий опис	Організатор створює нову подію або редагує існуючу: назва, опис, дати, локація, категорії, зображення, ціна квитка та ліміт місць

Таблиця 2.12

Опис прецеденту «Реклама події»

Назва варіанту використання	Реклама події
Основні актори	Організатор
Короткий опис	Організатор створює та оплачує промо-банер своєї події, налаштовуючи його параметри та період показу; після успішної оплати банер автоматично відображається в блоці «Рекомендовано» на головній сторінці

Таблиця 2.13

Опис прецеденту «Керування учасниками»

Назва варіанту використання	Керування учасниками
Основні актори	Організатор, Адміністратор
Короткий опис	Організатор переглядає список зареєстрованих учасників своєї події та може скасувати їхню реєстрацію за потреби, забезпечуючи контроль над складом учасників заходу

Таблиця 2.14

Опис прецеденту «Гаманець і виведення коштів»

Назва варіанту використання	Гаманець і виведення коштів
Основні актори	Організатор
Короткий опис	Організатор переглядає баланс від продажу квитків, історію транзакцій і подає запит на виведення коштів

Таблиця 2.15

Опис прецеденту «AI-генерація контенту події»

Назва варіанту використання	AI-генерація контенту події
Основні актори	Організатор
Короткий опис	Організатор автоматично генерує опис події та обкладинку під час створення події, що дозволяє значно спростити процес підготовки контенту та прискорити публікацію заходу

Таблиця 2.16

Опис прецеденту «Адміністрування системи»

Назва варіанту використання	Адміністрування системи
Основні актори	Адміністратор
Короткий опис	Адміністратор керує користувачами, подіями, платежами, квитками, рекламою через адмінпанель

Комплексний аналіз предметної області, огляд існуючих рішень на ринку та чітка постановка задачі із виявленням акторів дозволили сформувати функціональні та нефункціональні вимоги, які необхідно врахувати під час реалізації програмно-інформаційного комплексу. Функціональні вимоги визначають взаємодію користувачів із системою, тоді як нефункціональні охоплюють технічні та якісні характеристики платформи.

Зведені вимоги до системи подано у табл. 2.17.

Таблиця 2.17

Вимоги до системи

Категорія	Підкатегорія	Вимога
Функціональні вимоги	Реєстрація та профіль	Реалізація авторизації та реєстрації користувачів із можливістю редагування профілю та перегляду історії подій.
	Каталог подій	Забезпечення перегляду списку подій із фільтрацією, пошуком і сортуванням.
	Перегляд подій	Можливість перегляду детальної інформації про подію, включаючи опис, дату, локацію, ціну та кількість учасників.

Функціональні вимоги	Реєстрація та оплата	Реєстрація на події з підтримкою безкоштовних та платних квитків із інтеграцією платіжної системи.
	Скасування участі	Можливість скасування реєстрації та повернення коштів на баланс або банківську картку.
	Взаємодія з контентом	Можливість ставити лайки, залишати коментарі та відгуки до подій.
	Сповіщення	Отримання користувачем системних та персональних сповіщень.
	Управління подіями	Створення, редагування та налаштування подій організаторами.
	AI-функціонал	Автоматична генерація опису та обкладинки події.
	Реклама подій	Створення та розміщення рекламних банерів для просування подій.
	Адміністрування	Керування користувачами, подіями, платежами, квитками та заявками на виведення коштів.
Нефункціональні вимоги	Надійність	Система повинна забезпечувати стабільну роботу з мінімальним часом простою (не більше 5%) та низьким рівнем критичних помилок.
	Продуктивність і масштабованість	Підтримка щонайменше 100 одночасних користувачів із можливістю масштабування до 1000 і більше.

Нефункціональні вимоги	Зручність використання	Інтерфейс має бути інтуїтивно зрозумілим, а основні функції доступні користувачу без додаткового навчання.
	Безпека	Захист персональних даних користувачів і безпечна обробка платіжних транзакцій.

Аналіз функціональних вимог допоміг сформуванати сутності об’єктно орієнтованої структури системи, що забезпечать реалізацію системи для адміністрування та організації розважальних заходів. Нижче наведено діаграму класів системи, яка демонструє всі зв’язки між сутностями (рис. 2.2).

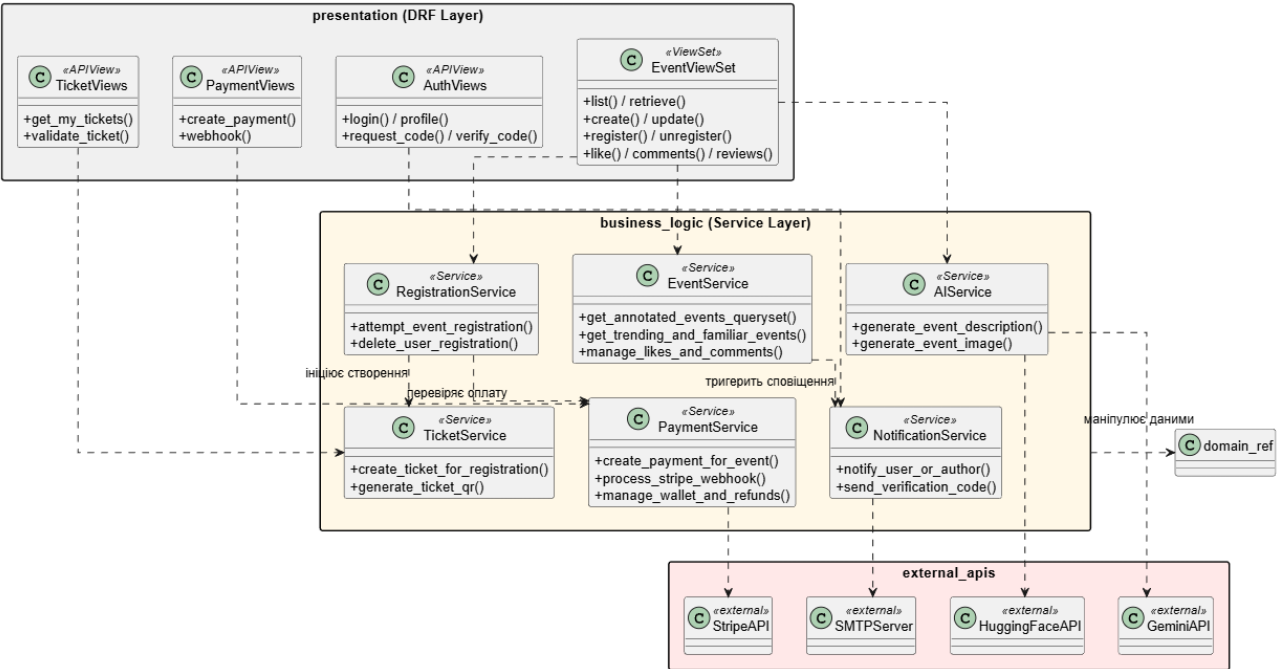


Рис. 2.2. Діаграма класів системи

Основним компонентом для роботи з подіями є EventViewSet, який забезпечує взаємодію клієнтської частини із системою через REST API. Він відповідає за перегляд, створення, редагування та видалення подій, а також за реєстрацію учасників, роботу з лайками, коментарями, відгуками та списками учасників.

Функціонал реєстрації та авторизації користувачів реалізовано за допомогою AuthViews. Цей модуль забезпечує створення облікових записів, вхід до системи, підтвердження електронної пошти, відновлення пароля та керування профілем користувача.

Бізнес-логіка реєстрації на заходи винесена до RegistrationService, який перевіряє можливість запису на подію, контролює наявність вільних місць і координує створення цифрових квитків та повернення коштів у разі скасування участі.

За проведення фінансових операцій відповідає PaymentService, який забезпечує оплату квитків, обробку транзакцій та повернення коштів. Для взаємодії із зовнішньою платіжною системою використовується StripeService, що інкапсулює всі деталі інтеграції зі Stripe.

Робота з цифровими квитками реалізована в TicketService. Сервіс створює квитки для зареєстрованих учасників, генерує QR-коди та надає інформацію, необхідну для проходження на захід.

Додаткова логіка, пов'язана з каталогом подій, зосереджена в EventService. Він відповідає за формування трендових і рекомендованих подій, а також обробку взаємодії користувачів із контентом платформи.

Для інформування користувачів про важливі події використовується NotificationService, який створює внутрішні повідомлення.

Функціонал просування заходів реалізовано через PromotionService та PromotionPaymentService. Вони забезпечують створення рекламних банерів, оплату рекламних кампаній і їх активацію після успішного проведення платежу.

Фінансовий облік організаторів здійснюється за допомогою WalletService, який веде облік доходів від продажу квитків, формує баланс та обробляє запити на виведення коштів.

Для автоматизації створення контенту в системі використовуються AIService та ImageGenerationService. Вони дозволяють генерувати текстові описи та графічні обкладинки подій за допомогою сервісів штучного інтелекту.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

2.2 Розробка бази даних

При створенні програмного продукту розробка бази даних є одним із ключових етапів, оскільки саме вона визначає структуру зберігання, обробки та взаємозв'язку інформації в системі. Для програмно-інформаційного комплексу для адміністрування та організації розважальних заходів було спроектовано реляційну базу даних, яка забезпечує зберігання даних про користувачів, події, реєстрації, квитки, платежі, рекламні кампанії та інші сутності платформи.

У якості системи керування базою даних використовується PostgreSQL [12] — потужна реляційна СУБД з відкритим вихідним кодом, яка забезпечує високу продуктивність, надійність та підтримку складних зв'язків між таблицями. PostgreSQL широко застосовується у сучасних вебзастосунках завдяки своїй стабільності, розширюваності та підтримці транзакційності, що є критично важливим для фінансових операцій у системі.

Для хостингу та розгортання бази даних використовується хмарна платформа Neon [13], яка надає serverless-реалізацію PostgreSQL. Це дозволяє автоматично масштабувати ресурси бази даних, забезпечує швидке створення середовищ (branching) та спрощує процес розробки і тестування. Використання Neon також зменшує навантаження на інфраструктуру та підвищує доступність системи.

З огляду на значний обсяг сутностей та їх взаємозв'язків, структура бази даних була логічно розділена на три основні підсистеми: підсистема керування користувачами та соціальною взаємодією; підсистема управління подіями та реєстраціями; підсистема фінансових операцій, квитків, промоцій та сповіщень.

На рис. 2.3 представлено структуру бази даних підсистеми керування користувачами та соціальною взаємодією. Вона включає сутності користувачів, профілів, а також елементи взаємодії користувачів із подіями, такі як коментарі, лайки та відгуки. Дана підсистема забезпечує зберігання та обробку даних, пов'язаних із активністю користувачів у системі.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

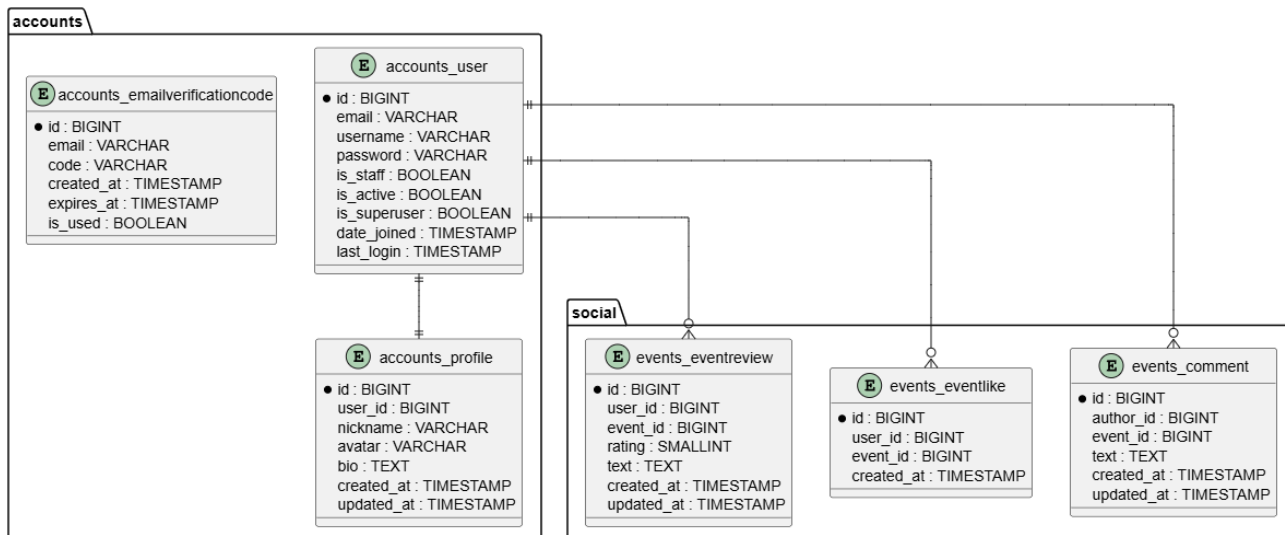


Рис. 2.3. Структура БД підсистеми керування користувачами та соціальною взаємодією

На рис. 2.4 зображено структуру бази даних підсистеми управління подіями та реєстраціями. Вона охоплює сутності подій, категорій, міст, локацій та реєстрацій користувачів на події. Дана підсистема забезпечує формування та управління інформацією про події, а також облік користувачів, які зареєструвалися на них.

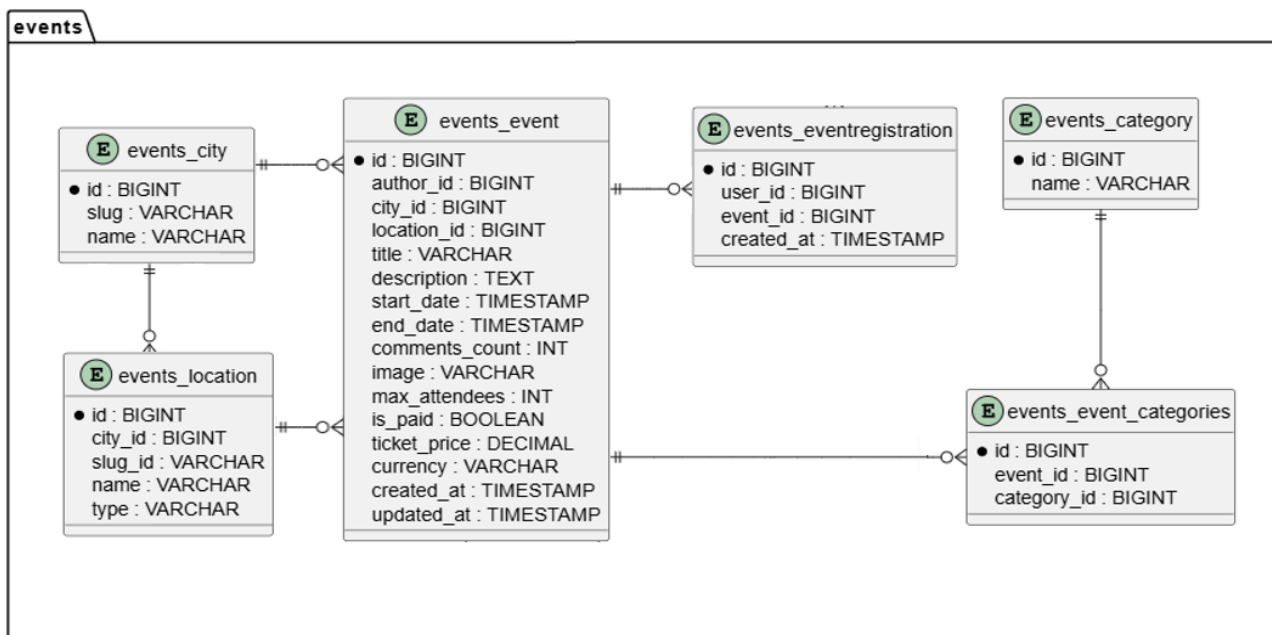


Рис. 2.4. Структура БД підсистеми управління подіями та реєстраціями

На рис. 2.5 представлено структуру бази даних підсистеми фінансових операцій, квитків, реклам та сповіщень. Вона включає сутності платежів,

гаманців, транзакцій, квитків, рекламних кампаній, а також системи сповіщень користувачів. Дана підсистема забезпечує обробку фінансових операцій, генерацію квитків та взаємодію з користувачами через сповіщення.

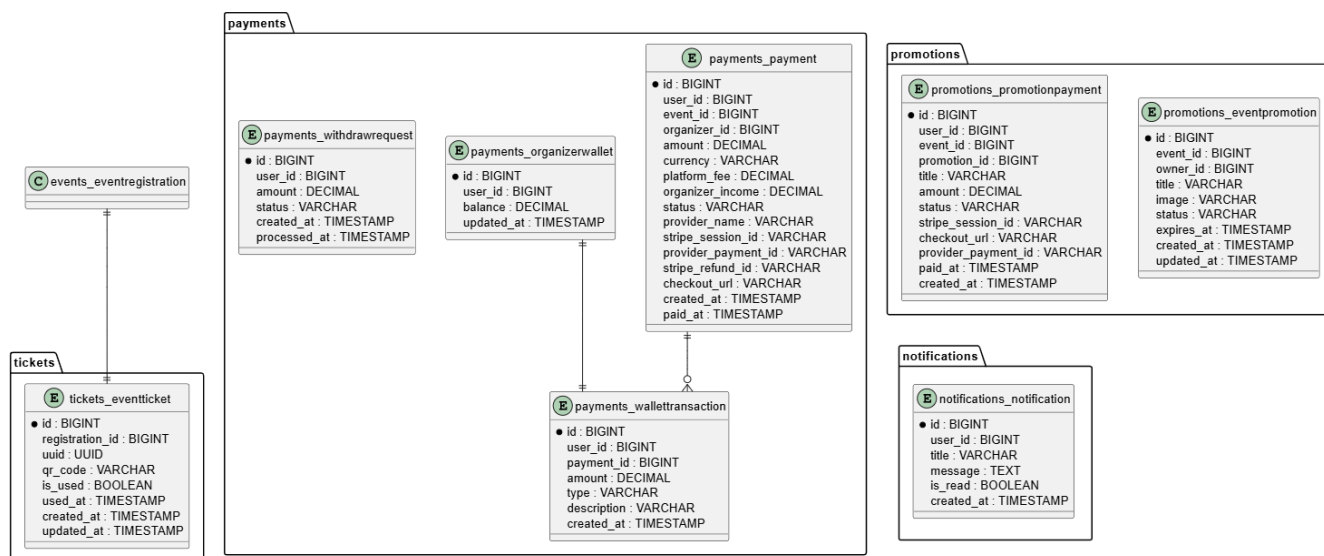


Рис. 2.5. Структура БД підсистеми фінансових операцій, квитків, промоцій та сповіщень

База даних програмно-інформаційного комплексу містить 20 таблиць.

Структура та опис полів кожної таблиці подані в табл. 2.18 – 2.37.

Таблиця 2.18

Структура таблиці «User»

Поле	Тип	Призначення
id	BIGINT	Унікальний ідентифікатор користувача
Поле	Тип	Призначення
email	String	Email користувача
username	String	Ім'я користувача
password	String	Захешований пароль
is_staff	Boolean	Права адміністратора
is_active	Boolean	Активний акаунт

продовж. табл. 2.18

is_superuser	Boolean	Супер-адмін доступ
date_joined	DateTime	Дата реєстрації
last_login	DateTime	Останній вхід

Таблиця 2.19

Структура таблиці «Profile»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID профілю
user_id	FK	Посилання на користувача
nickname	String	Відображуване ім'я користувача
avatar	String	Посилання на зображення аватара
bio	Text	Короткий опис користувача
created_at	DateTime	Дата створення профілю
updated_at	DateTime	Дата оновлення профілю

Таблиця 2.20

Структура таблиці «EventReview»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID відгуку
user_id	FK	Автор відгуку
event_id	FK	Подія
rating	SmallInt	Оцінка події (1–5)
Поле	Тип	Призначення
text	Text	Текст відгуку
created_at	DateTime	Дата створення
updated_at	DateTime	Дата редагування

Таблиця 2.21

Структура таблиці «EventComment»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID коментаря
author_id	FK	Автор коментаря
event_id	FK	Подія, до якої залишено коментар
text	Text	Текст коментаря
created_at	DateTime	Дата створення

Таблиця 2.22

Структура таблиці «EventVerificationCode»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID запису
email	String	Email для верифікації
code	String	Код підтвердження
Поле	Тип	Призначення
created_at	DateTime	Дата створення коду
expires_at	DateTime	Час закінчення дії коду
is_used	Boolean	Статус використання коду

Таблиця 2.23

Структура таблиці «EventLike»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID лайка
user_id	FK	Користувач, який поставив лайк
event_id	FK	Подія
created_at	DateTime	Дата створення лайка

Таблиця 2.24

Структура таблиці «Category»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID категорії
name	String	Назва категорії подій

Таблиця 2.25

Структура таблиці «Location»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID локації
city_id	FK	Місто розташування
slug_id	String	Унікальний slug локації
name	String	Назва місця проведення
type	String	Тип локації (hall, outdoor тощо)

Таблиця 2.26

Структура таблиці «City»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID міста
slug	String	URL-ідентифікатор міста
name	String	Назва міста

Таблиця 2.27

Структура таблиці «EventCategory»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID зв'язку
event_id	FK	Подія
category_id	FK	Категорія

Таблиця 2.28

Структура таблиці «EventRegistration»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID реєстрації
user_id	FK	Користувач
event_id	FK	Подія
created_at	DateTime	Дата реєстрації

Таблиця 2.29

Структура таблиці «Event»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID події
author_id	FK	Організатор події
city_id	FK	Місто проведення
location_id	FK	Конкретна локація
title	String	Назва події
description	Text	Опис події
start_date	DateTime	Початок події
end_date	DateTime	Завершення події
comments_count	Integer	Кількість коментарів
image	String	Зображення події
max_attendees	Integer	Максимальна кількість учасників
is_paid	Boolean	Чи є подія платною
ticket_price	Decimal	Ціна квитка
currency	String	Валюта оплати
created_at	DateTime	Дата створення
updated_at	DateTime	Дата оновлення

Таблиця 2.30

Структура таблиці «EventTicket»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID квитка
registration_id	FK	Реєстрація користувача
uuid	UUID	Публічний ідентифікатор квитка
qr_code	String	Шлях до QR-коду
is_used	Boolean	Чи використаний квиток
used_at	DateTime	Час сканування
created_at	DateTime	Дата створення
updated_at	DateTime	Дата оновлення

Таблиця 2.31

Структура таблиці «Payment»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID платежу
user_id	FK	Покупець
event_id	FK	Подія
organizer_id	FK	Організатор
amount	Decimal	Сума платежу
currency	String	Валюта
platform_fee	Decimal	Комісія платформи
Поле	Тип	Призначення
organizer_income	Decimal	Дохід організатора
status	String	Статус платежу
provider_name	String	Платіжний провайдер
stripe_session_id	String	Stripe session ID
provider_payment_id	String	ID платежу провайдера
stripe_refund_id	String	ID повернення

продовж. табл. 2.31

checkout_url	String	URL оплати
created_at	DateTime	Дата створення
paid_at	DateTime	Дата оплати

Таблиця 2.32

Структура таблиці «WithdrawRequest»

Поле	Тип	Призначення
id	BIGINT	ID заявки
user_id	FK	Організатор
amount	Decimal	Сума виводу
status	String	Статус заявки
created_at	DateTime	Дата створення
processed_at	DateTime	Дата обробки

Таблиця 2.33

Структура таблиці «EventPromotion»

Поле	Тип	Призначення
id	BIGINT	Унікальний ідентифікатор рекламної кампанії
event_id	FK	Подія, яка просувається
owner_id	FK	Користувач, який створив промоцію
title	String	Назва рекламної кампанії
image	String	Зображення рекламного банера
status	String	Поточний статус промоції
expires_at	DateTime	Дата завершення рекламної кампанії
created_at	DateTime	Дата створення
updated_at	DateTime	Дата останнього оновлення

Таблиця 2.34

Структура таблиці «WalletTransaction»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID транзакції
user_id	FK	Користувач
payment_id	FK	Пов'язаний платіж
amount	Decimal	Сума транзакції
type	String	Тип (CREDIT, REFUND, PURCHASE)
Поле	Тип	Призначення
description	String	Опис транзакції
created_at	DateTime	Дата створення

Таблиця 2.35

Структура таблиці «PromotionPayment»

Поле	Тип	Призначення
id	BIGINT	ID платежу
user_id	FK	Користувач
event_id	FK	Подія
promotion_id	FK	Промоція
title	String	Назва
amount	Decimal	Сума
currency	String	Валюта
status	String	Статус платежу
stripe_session_id	String	Stripe session
checkout_url	String	URL оплати
provider_payment_id	String	ID платежу
paid_at	DateTime	Дата оплати
created_at	DateTime	Дата створення

Таблиця 2.36

Структура таблиці «OrganizerWallet»

Поле	Тип	Призначення
id	BIGINT	Унікальний ID гаманця
user_id	FK	Організатор
balance	Decimal	Поточний баланс
updated_at	DateTime	Оновлення балансу

Таблиця 2.37

Структура таблиці «Notification»

Поле	Тип	Призначення
id	BIGINT	ID повідомлення
user_id	FK	Отримувач
title	String	Заголовок
message	Text	Текст повідомлення
is_read	Boolean	Статус прочитання
created_at	DateTime	Дата створення

2.3 Проєктування та реалізація алгоритмів роботи системи

Створений програмно-інформаційний комплекс має інтуїтивно зрозумілий інтерфейс і забезпечує зручну навігацію, що дозволяє користувачам ефективно взаємодіяти з основними функціональними модулями системи. Процес взаємодії користувача з платформою починається з головної сторінки, яка виконує роль центральної точки входу до системи та містить каталог подій, механізми пошуку, фільтрації та сортування, а також блоки з популярними та рекомендованими подіями.

Побудова каталогу подій реалізована таким чином, щоб забезпечити швидкий доступ до важливого контенту навіть при великій кількості записів у системі. Фільтрація за містом, категорією та датою виконується на рівні

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

серверної частини, що дозволяє зменшити навантаження на клієнтський застосунок та підвищити продуктивність системи.

Після вибору події користувач переходить на сторінку її детального перегляду, де отримує повну інформацію про подію, включаючи опис, дату проведення, локацію, кількість доступних місць та вартість участі. Даний рівень деталізації необхідний для прийняття користувачем рішення щодо участі в події та зменшує кількість додаткових запитів до системи. Додатково сторінка містить інформацію про організатора та статистичні показники, що підвищує прозорість платформи.

Взаємодія користувача з подією реалізована через соціальні механізми, зокрема додавання вподобань, коментарів та відгуків (Додаток А). Такі дії доступні лише авторизованим користувачам, що дозволяє контролювати якість контенту та зменшити кількість некоректних або анонімних взаємодій.

Процес авторизації передбачає реєстрацію або вхід користувача з підтвердженням електронної пошти. Після успішної автентифікації користувач отримує доступ до особистого кабінету, де відображається його активність у системі, включаючи створені та зареєстровані події, придбані квитки, персональні налаштування профілю та історію взаємодії з платформою.

Реєстрація на подію виконується з обов'язковою перевіркою бізнес-правил системи, зокрема доступності місць, актуальності події та статусу користувача. У випадку платної події система ініціює процес оплати через платіжну систему Stripe або внутрішній гаманець користувача (Додаток Б). Після успішної оплати система автоматично створює реєстрацію та цифровий квиток із QR-кодом, який використовується для подальшого доступу до події. Усі придбані квитки зберігаються в особистому кабінеті та можуть бути переглянуті в будь-який момент.

Для організаторів передбачено функціонал створення та управління подіями, редагування інформації про заходи, перегляду фінансової інформації, управління рекламними кампаніями та виведення коштів. Додатково реалізовано використання технологій штучного інтелекту для автоматичної генерації описів

		Руденко О.І.			ІПЗ.КР.Б – 121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

подій та створення графічних обкладинок, що дозволяє значно спростити процес публікації нового заходу (Додаток В).

Окремо реалізовано механізм check-in, який дозволяє організатору перевіряти квитки учасників за допомогою QR-коду та контролювати фактичну присутність відвідувачів. Система також підтримує внутрішні сповіщення, які інформують користувачів про зміни статусу подій, успішні платежі, реєстрації та інші важливі дії (Додаток Г).

Загалом взаємодія з платформою побудована таким чином, щоб забезпечити простий доступ до подій, швидку реєстрацію, безпечне проведення оплат та зручну роботу як для звичайних користувачів, так і для організаторів.

Для узагальнення процесу взаємодії користувача з системою та відображення основних сценаріїв використання платформи було побудовано діаграму активності (рис. 2.6). Вона відображає послідовність дій користувача, починаючи від перегляду каталогу подій та завершуючи процесом реєстрації або оплати участі, а також взаємодію з додатковими функціональними можливостями системи.

На діаграмі відображено логіку переходів між основними станами користувача в системі, включаючи альтернативні сценарії виконання операцій. Зокрема, враховано відмінності між авторизованими та неавторизованими користувачами, перевірку можливості реєстрації на подію, а також розгалуження процесу залежно від типу події (безкоштовна або платна).

Окремо виділено взаємодію з зовнішньою платіжною системою Stripe, яка інтегрується в процес оформлення платних подій через механізм створення платіжної сесії та підтвердження транзакції. Після успішного завершення оплати система автоматично формує реєстрацію користувача та генерує цифровий квиток із QR-кодом, який використовується для подальшої верифікації участі.

Таким чином, діаграма відображає не лише користувацький сценарій, але й ключові бізнес-процеси системи, які реалізуються на рівні серверної логіки та взаємодії через REST API. Це дозволяє забезпечити чітке розділення

		Руденко О.І.			ІІЗ.КР.Б – 121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

відповідальності між клієнтською та серверною частинами системи та підвищує зрозумілість архітектури проєкту.

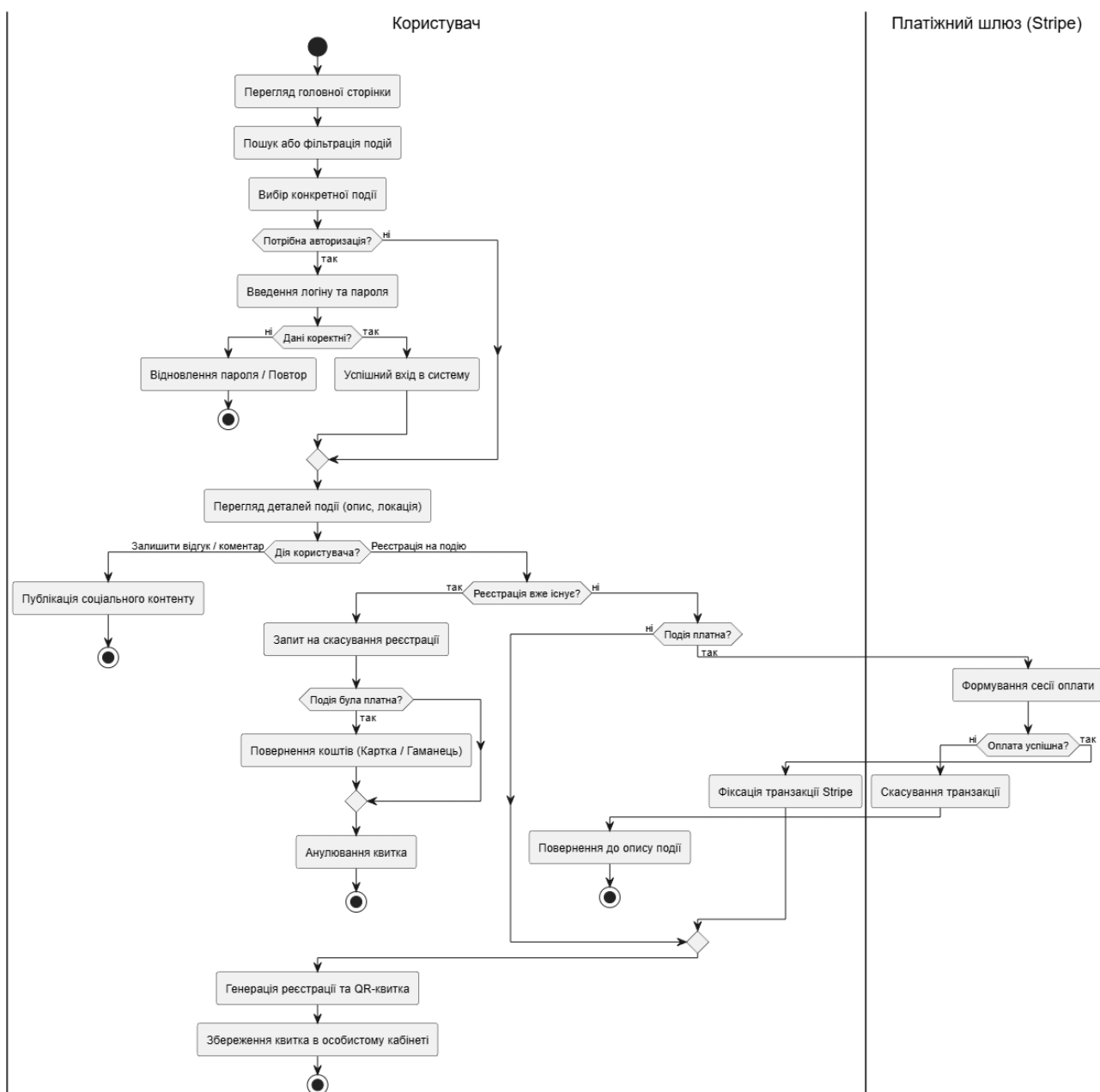


Рис.2.6. Діаграма активності

2.4 Реалізація функціоналу програмно-інформаційного комплексу

Реалізація програмно-інформаційного комплексу побудована на клієнт-серверній архітектурі з розділенням відповідальності між рівнем представлення, бізнес-логікою та рівнем доступу до даних. Система реалізована у вигляді

		Руденко О.І.			ІІЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

модульного моноліту: серверна частина представлена єдиним Django-застосунком, структурованим на предметні модулі, тоді як клієнтська частина є самостійним односторінковим застосунком. Такий підхід не передбачає мікросервісної архітектури, однак забезпечує логічну ізоляцію доменів і спрощує підтримку транзакційної цілісності операцій, зокрема в ланцюжку «оплата — реєстрація — квиток — сповіщення».

Серверна частина реалізована з використанням Python, Django [14] та Django REST Framework, що дозволяє будувати масштабоване REST API та організувати систему у вигляді окремих модулів, таких як accounts, events, payments, tickets, promotions та notifications. Кожен модуль містить власні моделі, серіалізатори, представлення та сервісний шар. Доступ до даних здійснюється через Django ORM; окремий шар репозиторіїв не виділявся, оскільки ORM інкапсулює базові операції з даними, а бізнес-логіка реалізована у сервісах.

Архітектура бекенду базується на багаторівневій моделі, де рівень представлення реалізовано через Django REST Framework ViewSets та APIViews, бізнес-логіка винесена у сервісний шар, а рівень зберігання даних реалізовано через Django ORM із використанням PostgreSQL. Це дозволяє відокремити HTTP-логіку від предметної логіки та спрощує подальше масштабування системи.

Основна бізнес-логіка реалізована у вигляді сервісів за окремими доменами. EventService відповідає за формування каталогу подій, фільтрацію, сортування та агрегацію показників активності (лайки, коментарі, реєстрації). RegistrationService забезпечує процес реєстрації користувачів із перевіркою доступності місць, актуальності події та статусу оплати. PaymentService реалізує обробку платежів через Stripe [15] та внутрішній гаманець, включаючи створення платіжних сесій і обробку webhook-подій. WalletService відповідає за облік балансів та транзакцій, а TicketService — за генерацію цифрових квитків і QR-кодів. Додатково PromotionService забезпечує функціонал рекламних кампаній і їх оплату.

		Руденко О.І.			ІІЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Клієнтська частина реалізована як SPA на базі React. Взаємодія з сервером здійснюється через REST API з використанням JWT-аутентифікації. Стан авторизації керується через AuthContext, а доступ до захищених маршрутів реалізовано через ProtectedRoute. Для роботи з API використовується окремий клієнтський модуль, який уніфікує запити та додає токени доступу. Обмін даними виконується у форматі JSON, а для завантаження файлів використовується multipart/form-data. Між клієнтом і сервером налаштовано CORS-політику.

Каталог подій реалізований через API, який підтримує пошук, фільтрацію за містом, категорією та датою, а також сортування за популярністю. На стороні сервера виконується агрегація даних та повернення результату у форматі JSON з підтримкою пагінації, після чого фронтенд динамічно відображає список подій у вигляді карток. Окремо завантажуються стрічка рекламних подій, що формуються на основі активних промоцій.

Сторінка детального перегляду події надає користувачу повну інформацію про подію, включаючи опис, дату, локацію, ціну квитка, а також соціальні взаємодії у вигляді коментарів, лайків та відгуків. Додатково відображається інформація про організатора та кількість учасників події.

Процес створення події реалізований через багатокрокову форму з можливістю вибору категорії, локації, завантаження зображення та встановлення параметрів участі. Інтегровано AI-генерацію опису та зображень, а також передбачено нормалізацію HTML-вмісту, що надходить із зовнішніх сервісів.

Процес реєстрації на подію реалізований як послідовність перевірок і операцій, що включають перевірку наявності вільних місць, актуальності події та статусу авторизації користувача. У випадку платної події створюється платіж із початковим статусом pending, після чого користувач перенаправляється на Stripe Checkout або використовує внутрішній баланс. Після підтвердження оплати через webhook або внутрішню транзакцію система автоматично створює запис реєстрації та ініціює формування цифрового квитка, який надалі використовується для доступу до події.

		Руденко О.І.			ІПЗ.КР.Б – 121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Система авторизації базується на email-реєстрації з одноразовим кодом підтвердження та подальшою видачею JWT-токенів (access і refresh). Для захисту від зловживань застосовано rate limiting. Розмежування прав доступу реалізовано через механізми Django та DRF-permissions.

Особистий кабінет користувача об'єднує функції учасника та організатора, включаючи перегляд квитків, історії подій, керування профілем, створення та редагування подій, а також перегляд статистики. Для організаторів додатково доступні інструменти аналітики та управління фінансами.

Інтеграція Stripe [16] забезпечує обробку онлайн-платежів через checkout-сесії та webhook-механізм із перевіркою підпису. Додатково реалізовано внутрішній гаманець для обліку транзакцій, комісій та виплат організаторам.

У системі також реалізовано механізми генерації контенту із використанням зовнішніх AI-сервісів. AIService використовує Google Gemini API [17] для автоматичного формування описів подій, а ImageGenerationService генерує зображення обкладинок із використанням Hugging Face API [18] з резервним механізмом у разі недоступності зовнішнього сервісу.

Система сповіщень реалізована через NotificationService (Додаток Г), який формує повідомлення під час ключових подій у системі: лайків або коментарів до подій, нових реєстрацій та виключення учасників організатором.

У результаті реалізації було створено повнофункціональний програмно-інформаційний комплекс, який забезпечує користувачам можливість перегляду та фільтрації подій, реєстрації та оплати участі, а організаторам надає інструменти для створення подій, їх просування, управління фінансами та проведення контролю доступу учасників через систему цифрових квитків.

Висновки до розділу 2

Під час проєктування програмно-інформаційного комплексу для організації та управління розважальними заходами проведено аналіз основних сценаріїв використання системи з урахуванням ролей користувачів, зокрема

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

відвідувачів, організаторів подій та адміністраторів, а також визначено функціональні вимоги до програмного забезпечення. На основі цього було спроектовано архітектуру системи з чітким розділенням серверної та клієнтської частин, а також визначено принципи їх взаємодії через REST API.

Було розроблено логіку роботи основних процесів системи, включаючи перегляд та фільтрацію подій, детальний перегляд інформації про подію, процес реєстрації користувача та оплати квитків. Це дозволило формалізувати ключові бізнес-процеси платформи та забезпечити їх коректну реалізацію.

Реалізовано структуру бази даних на основі PostgreSQL, яка забезпечує зберігання інформації про користувачів, події, реєстрації, платежі, квитки, промоційні кампанії та сповіщення. Особливу увагу приділено зв'язкам між сутностями для коректного відображення взаємодії користувачів із подіями та фінансовими операціями.

Була побудована адміністративна панель на основі Django Admin із розширенням Django Unfold, що забезпечує сучасний інтерфейс управління даними. Вона використовується для модерації користувачів, подій, платежів, промоційних кампаній та сповіщень, а також для контролю стану системи.

У процесі реалізації продемонстровано основний функціонал системи, що охоплює реєстрацію та авторизацію користувачів, створення та перегляд подій, а також обробку платежів через зовнішні сервіси та внутрішній гаманець.

Таким чином, у другому розділі реалізовано програмний прототип серверної та клієнтської частини програмно-інформаційного комплексу для організації та управління розважальними заходами, що забезпечує повний життєвий цикл подій — від створення до відвідування.

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1 Порядок встановлення та налаштування системи

Для забезпечення роботи програмно-інформаційного комплексу використовується хмарна клієнт–серверна архітектура, яка включає клієнтську частину, серверну частину, базу даних та зовнішні сервіси. Такий підхід забезпечує централізоване зберігання даних, віддалений доступ до системи з будь-якого пристрою та спрощує масштабування програмного продукту залежно від навантаження. Розподіл функціональності між компонентами підвищує надійність платформи, зменшує зв'язність модулів та полегшує її супровід і подальший розвиток.

Клієнтська частина реалізована як односторінковий вебзастосунок на базі React [19] і розгортається на платформі Vercel [20]. Збірка виконується командою `npm run build`, після чого статичні файли розміщуються на CDN і обслуговуються як SPA з перенаправленням маршрутів на `index.html`. Для підключення до API використовується змінна оточення `REACT_APP_API_URL`, що визначає адресу бекенду. Користувач взаємодіє із системою через браузер без додаткового програмного забезпечення. Обмін даними здійснюється через REST API із використанням HTTPS, що забезпечує захищену передачу інформації.

Серверна частина реалізована на основі Python, Django та Django REST Framework і розгортається на платформі Fly.io [21] у вигляді Docker-контейнера [22]. Контейнер збирається за допомогою Dockerfile, у якому встановлюються залежності проєкту, виконується збір статичних файлів (`collectstatic`), після чого застосунок запускається через Gunicorn [23] як WSGI-сервер. У конфігураційному файлі Fly.io визначено регіон розгортання, порт, HTTPS-режим та автоматичне виконання міграцій через `release_command`, що забезпечує актуальність структури бази даних після кожного оновлення. Конфіденційні параметри (ключі API, `DATABASE_URL`, `DJANGO_SECRET_KEY`, Stripe, R2, Gemini) зберігаються у захищених секретах платформи, що виключає їх потрапляння до коду репозиторію.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Сервер забезпечує обробку HTTP-запитів, реалізацію бізнес-логіки та взаємодію з базою даних і зовнішніми сервісами. На цьому рівні реалізовано механізми автентифікації та авторизації користувачів, управління подіями, реєстраціями, квитками, платежами, рекламними кампаніями та сповіщеннями. Додатково сервер відповідає за валідацію даних, обробку помилок, логування подій та забезпечення цілісності транзакцій. Між клієнтом і сервером налаштовано CORS-політику, що дозволяє безпечно крос-доменну взаємодію лише з дозволених джерел.

Для зберігання даних використовується PostgreSQL, розгорнута на платформі Neon, що забезпечує хмарне масштабоване реляційне сховище. Підключення здійснюється через змінну DATABASE_URL із використанням бібліотеки dj-database-url. Медіафайли зберігаються у Cloudflare R2 через django-storages, що дозволяє винести файлове сховище за межі серверного контейнера та зменшити навантаження на бекенд. Статичні файли адміністративної панелі обслуговуються через WhiteNoise безпосередньо з контейнера.

Система інтегрована із зовнішніми сервісами Stripe, Google Gemini [24] та Hugging Face, які забезпечують відповідно обробку платежів, генерацію текстового контенту та створення зображень. Для надсилання електронних листів використовується SMTP-сервер Gmail. Після розгортання бекенду в Stripe налаштовується webhook-ендпоінт для підтвердження платежів у режимі, близькому до реального часу, що забезпечує синхронізацію фінансових операцій із системою.

Для адміністрування платформи використовується Django Admin із розширенням Django Unfold, що надає сучасний інтерфейс для керування користувачами, подіями, платежами та іншими сутностями системи. Адміністратор має можливість контролювати стан системи, виконувати модерацію контенту, переглядати статистику та керувати об'єктами без необхідності прямого доступу до бази даних.

Загальну архітектуру розгортання програмно-інформаційного комплексу наведено на рис. 3.1. Діаграма відображає основні компоненти системи:

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

клієнтський пристрій із React SPA, CDN для фронтенду, сервер Fly.io з Django REST Framework у Docker-контейнері [25], PostgreSQL як основну базу даних, Cloudflare R2 [26] для зберігання медіа, а також зовнішні API із взаємодією через HTTPS.

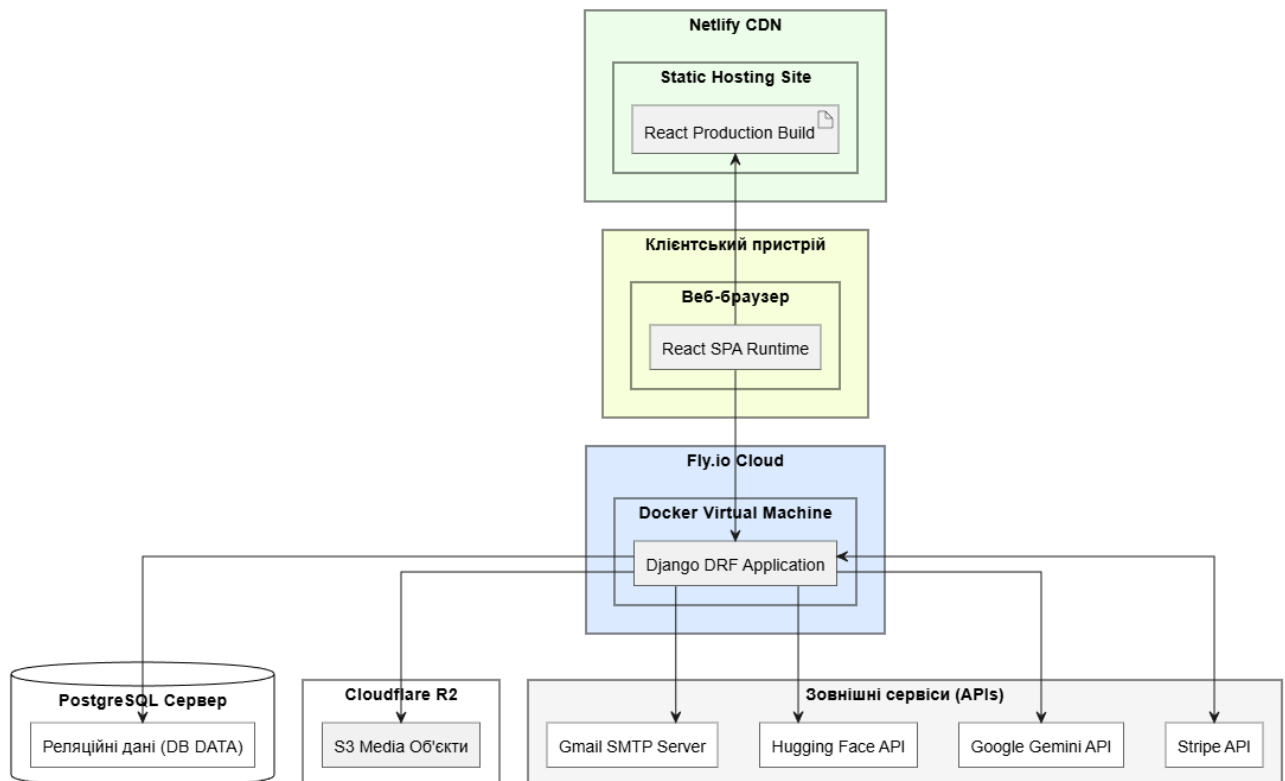


Рис. 3.1. Діаграма розгортання

Представлена архітектура забезпечує стабільне функціонування системи, захищену передачу даних між компонентами, високу відмовостійкість, а також можливість незалежного масштабування окремих частин програмного комплексу відповідно до навантаження. Додатково така структура сприяє спрощенню супроводу системи, дозволяє оновлювати окремі компоненти без впливу на інші частини платформи та забезпечує гнучкість при подальшому розвитку функціоналу. Це особливо важливо для систем із змінним навантаженням, де кількість користувачів і обсяг оброблюваних даних можуть динамічно зростати.

3.2 Структура інтерфейсу та порядок взаємодії із системою

Головна сторінка є точкою входу в систему. У верхній частині розміщено логотип платформи та елементи навігації для авторизації або переходу до профілю користувача. Центральна частина сторінки містить короткий опис можливостей сервісу та рекламну стрічку активних промо-подій.

Основний функціональний блок представлений каталогом подій, який включає вкладки для перегляду всіх подій і трендових заходів, панель фільтрації (місто, категорія, дата, пошук) та список карток подій. Дані підвантажуються динамічно під час прокручування сторінки (рис. 3.2).

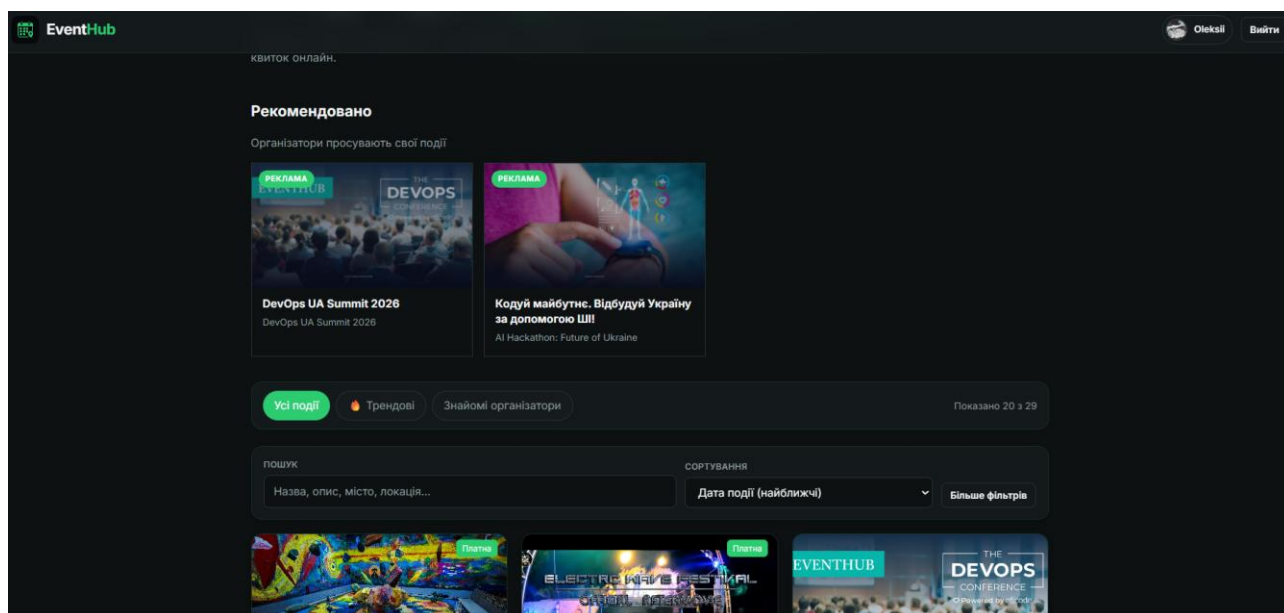


Рис. 3.2. Головна сторінка

Сторінка події відображає повну інформацію про захід: обкладинку, назву, категорії, дату та час проведення, місце проведення і детальний опис. Для зручності користувачів у верхній частині сторінки також відображаються основні показники активності події, зокрема кількість вподобань, коментарів та зареєстрованих учасників. Окремий блок містить інформацію про організатора заходу, включаючи його ім'я та аватар, а також надає можливість швидкого переходу до публічного профілю для ознайомлення з іншими організованими подіями.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Для подій із платною участю додатково відображається вартість квитка, кількість доступних місць та доступні способи оплати. Якщо реєстрацію на подію ще не закрито та наявні вільні місця, користувач може зареєструватися або придбати квиток безпосередньо зі сторінки заходу без необхідності переходу до інших розділів системи. Такий підхід забезпечує швидкий і зручний процес взаємодії з подією.

Сторінка детального перегляду події представлена на рис. 3.3.

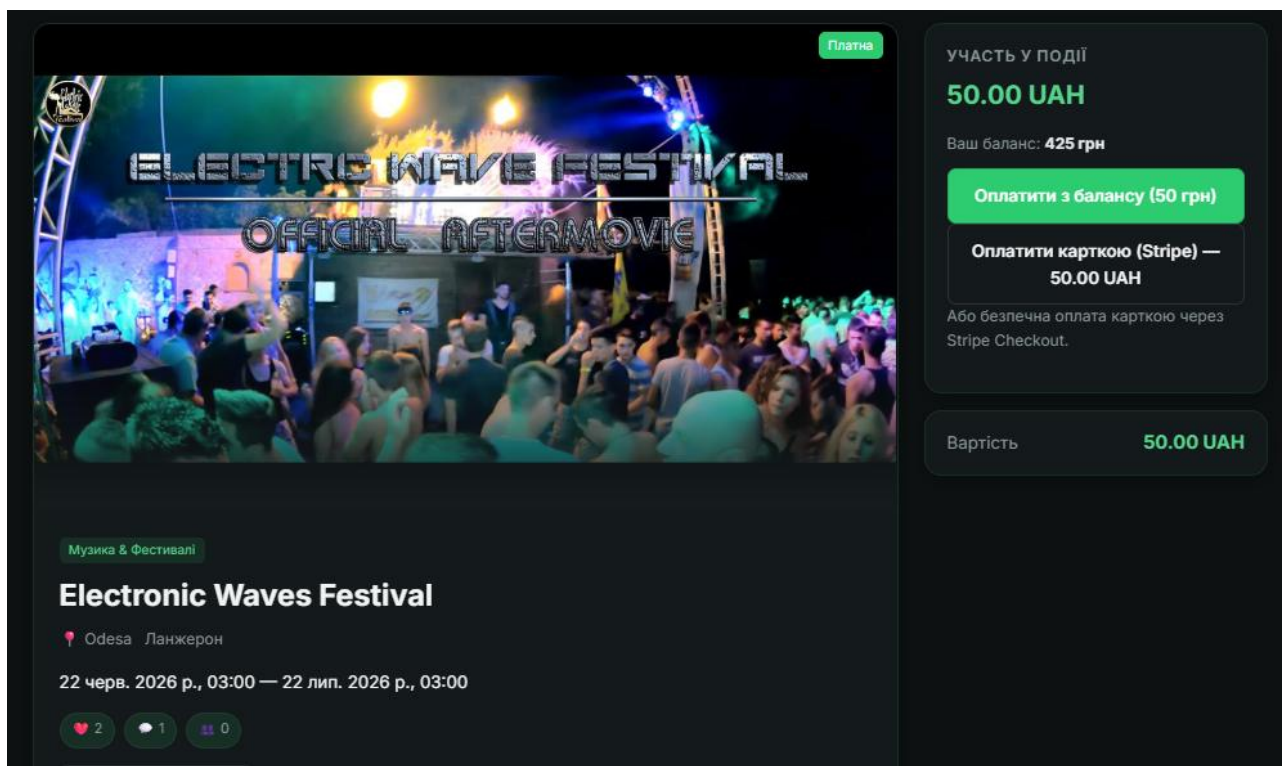


Рис. 3.3. Сторінка детального перегляду події

Авторизований користувач може активно взаємодіяти із заходом. Зокрема, він має можливість додавати подію до вподобаних, залишати коментарі для обговорення деталей заходу, а після його завершення — публікувати відгуки та оцінювати власний досвід участі. Коментарі та відгуки сприяють формуванню спільноти навколо подій і допомагають іншим користувачам отримати додаткову інформацію перед реєстрацією (рис. 3.4).

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

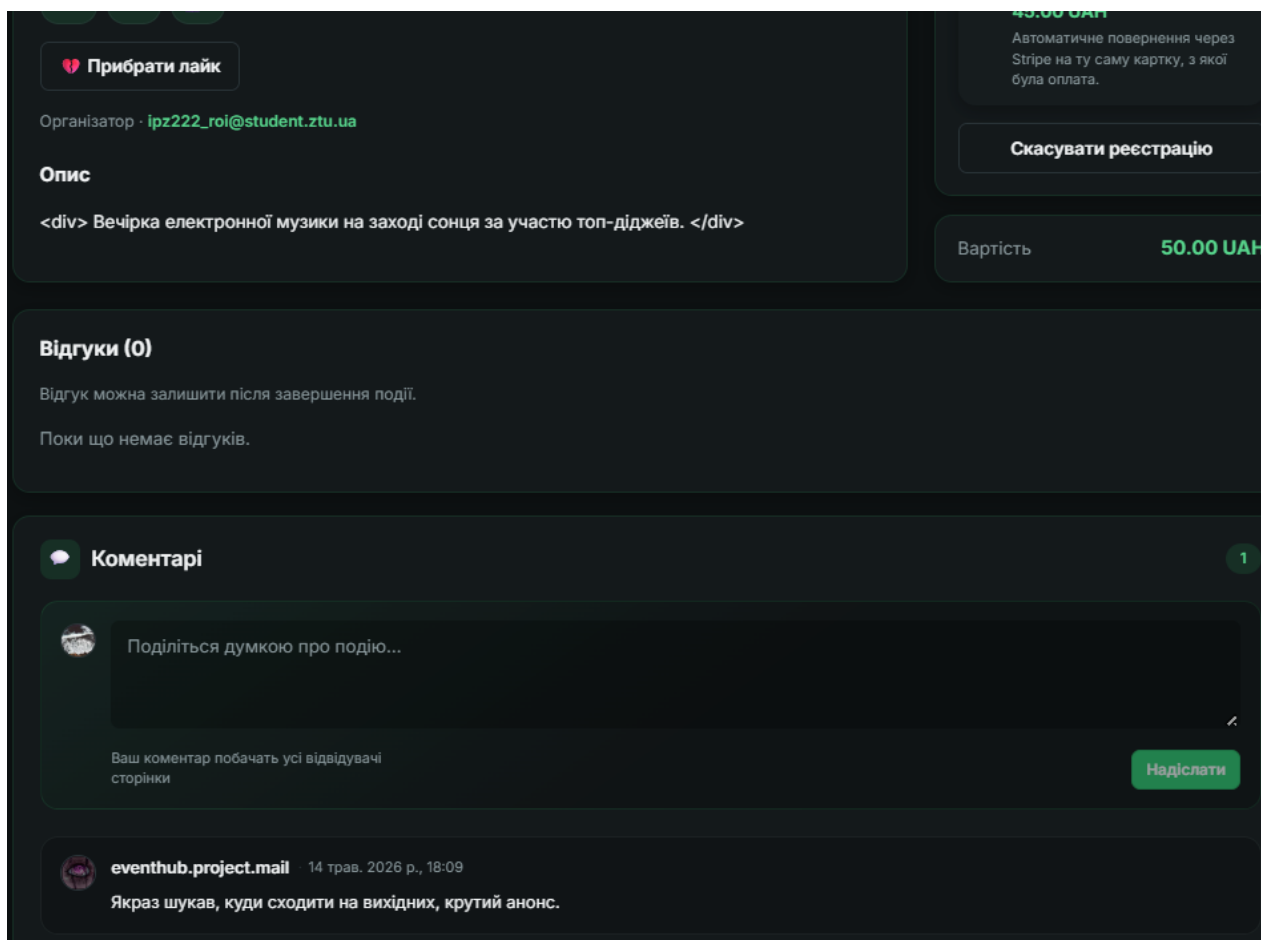


Рис. 3.4. Блок з коментарями, лайками і відгуками

Сторінка публічного профілю організатора містить основну інформацію про користувача, зокрема аватар, ім'я або нікнейм, короткий опис (біографію) та дату реєстрації на платформі. Даний розділ призначений для формування публічного представлення організатора та забезпечує користувачам можливість швидко ознайомитися з його діяльністю, досвідом організації заходів та рівнем активності на платформі.

Нижче розташовано перелік створених організатором подій у вигляді окремих карток. Кожна картка містить основну інформацію про захід, зокрема назву, дату проведення, короткий опис, категорію та зображення-обкладинку. Завдяки цьому користувачі можуть швидко переглядати інші події автора та переходити до їх детального опису. Сторінка профілю організатора представлена на рис. 3.5.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

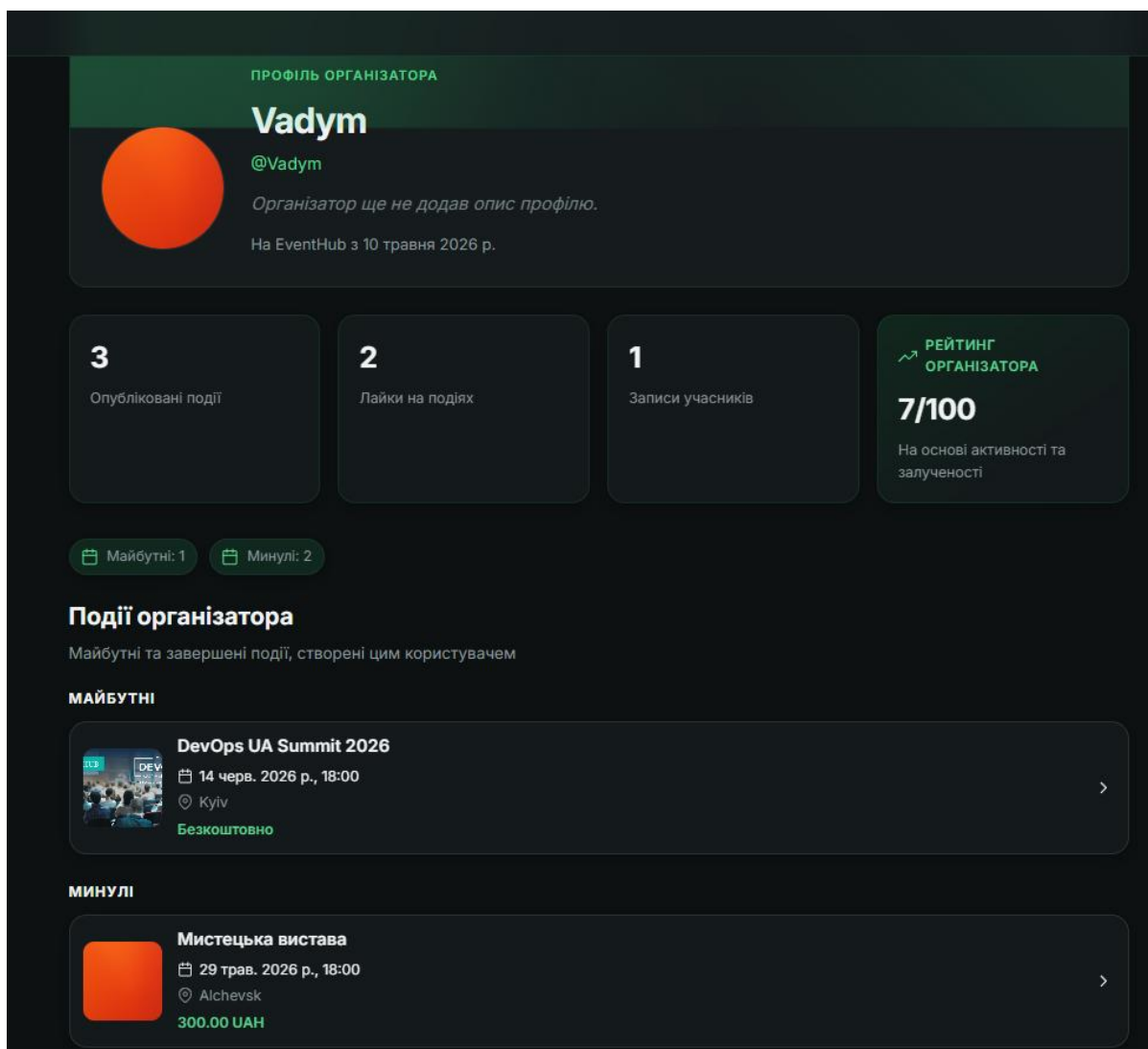


Рис. 3.5. Публічний профіль організатора

Для платних подій передбачено обов'язкову попередню оплату участі. Користувач може обрати один із доступних способів оплати: здійснити платіж через зовнішню платіжну систему Stripe або скористатися внутрішнім гаманцем платформи за умови наявності достатньої кількості коштів на балансі. Такий підхід забезпечує гнучкість під час придбання квитків та дозволяє користувачам обирати найбільш зручний спосіб проведення оплати.

Під час здійснення платежу через Stripe користувач перенаправляється на захищену сторінку платіжної системи, де вводить необхідні реквізити банківської картки та підтверджує транзакцію (рис. 3.6). Після успішного завершення оплати інформація автоматично передається до системи за допомогою механізму вебхуків, що забезпечує своєчасне оновлення статусу

		Руденко О.І.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

замовлення та реєстрації. У випадку використання внутрішнього балансу списання коштів виконується миттєво без необхідності переходу на сторонні сервіси.

← Eventhub sandbox Sandbox

Electronic Waves Festival
UAH 50.00

Pay with link

OR

Contact information

Email hoowhoower@gmail.com

Payment method

Card

Card information

4242 4242 4242 4242

12 / 34 123

Cardholder name

Oleksii Rudenko

Billing address

Ukraine

Житомир вул. Чуднівська

Address line 2

Житомир

Oblast

10000

☐ Save my information for faster checkout
Pay securely at Eventhub sandbox and everywhere [Link](#) is accepted.

Pay

Powered by stripe | Terms Privacy

Рис. 3.6. Оплата квитка за допомогою Stripe

До моменту підтвердження успішної транзакції система блокує можливість реєстрації на подію, що запобігає несанкціонованому доступу до платних заходів та гарантує коректну обробку фінансових операцій. Після підтвердження платежу користувач автоматично реєструється на подію, а інформація про придбаний квиток зберігається у його особистому кабінеті.

Сторінка успішної оплати підтверджує завершення транзакції та надає доступ до квитка або списку квитків користувача (рис. 3.7).

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

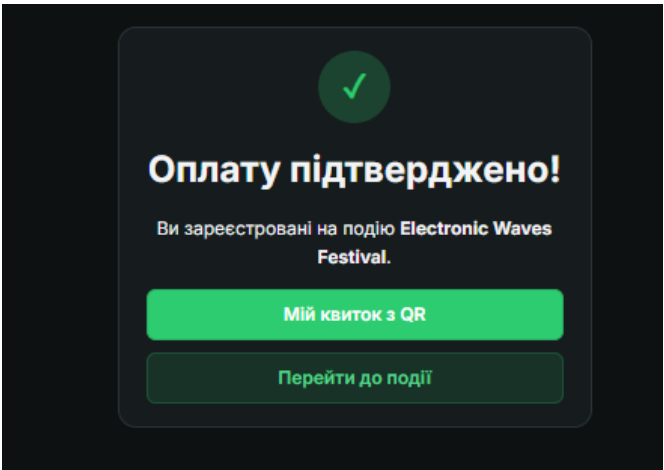


Рис. 3.7. Сторінка зі сповіщенням про успішну оплату

Сторінка «Мої квитки» містить перелік усіх придбаних квитків користувача з можливістю переходу до їх детального перегляду. Для кожного квитка відображається основна інформація про подію, зокрема її назва, дата проведення та поточний статус. Це дозволяє користувачу швидко знаходити необхідні квитки та контролювати свої реєстрації на заходи. (рис. 3.8).

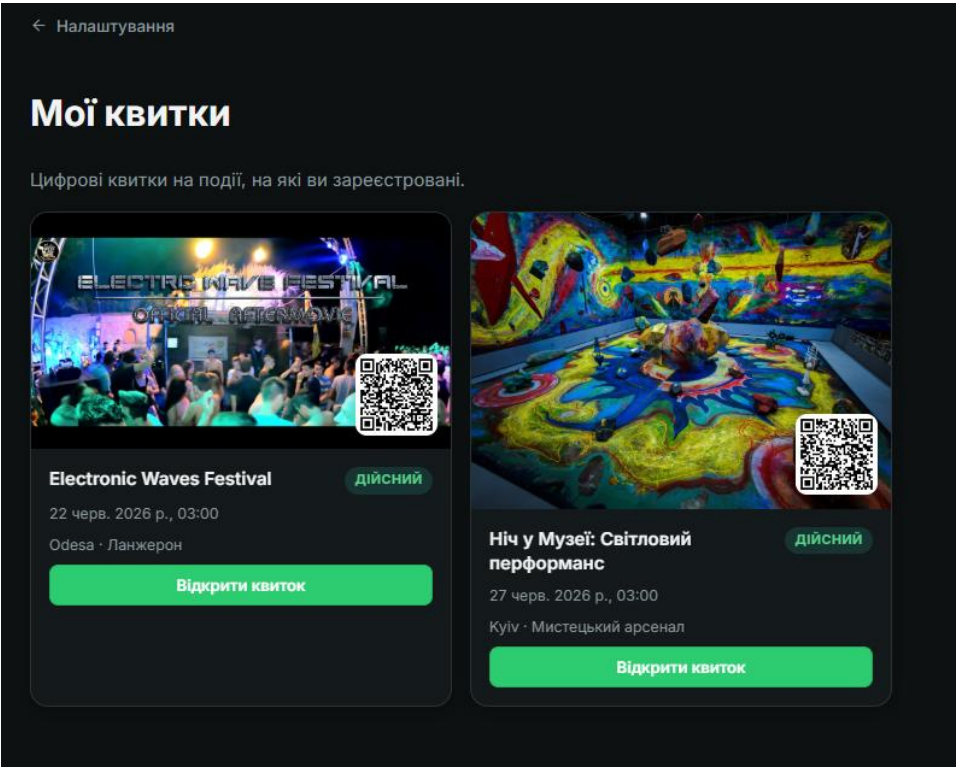


Рис. 3.8. Список квитків користувача

Кожен цифровий квиток містить інформацію про подію, учасника та QR-код для проходження перевірки на вході (рис. 3.9).

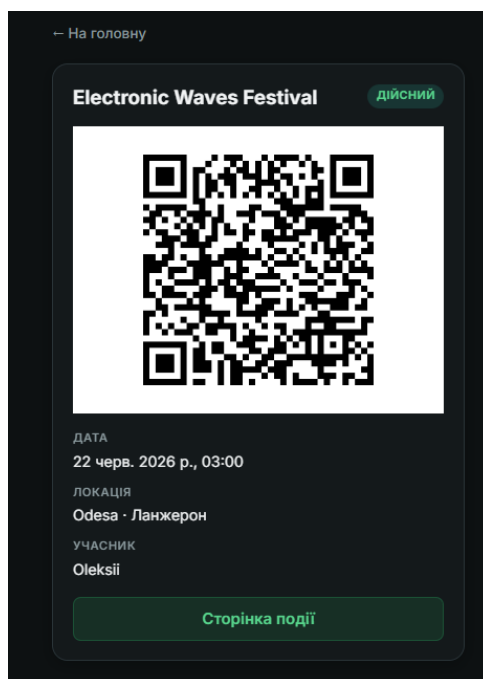


Рис. 3.9. Цифровий квиток

Особистий кабінет містить основну інформацію про користувача, статистику активності та швидкий доступ до основних функцій системи. На сторінці відображаються персональні дані користувача, показники його активності на платформі, а також навігаційні елементи для переходу до інших розділів особистого кабінету (рис. 3.10).

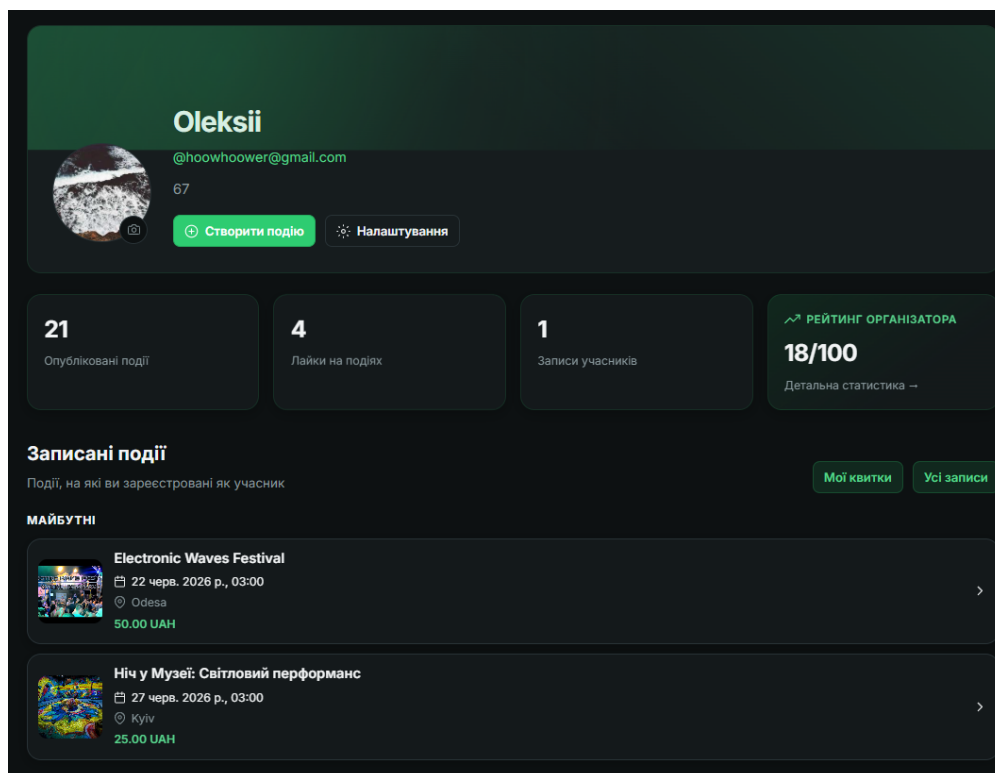


Рис. 3.10. Особистий кабінет користувача

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

Натиснувши кнопку «Налаштування», користувач переходить до сторінки керування особистим профілем. На сторінці налаштувань користувач може змінювати своє ім'я та нікнейм, редагувати інформацію про себе, оновлювати адресу електронної пошти, яка використовується для авторизації та отримання системних повідомлень, а також змінювати пароль для входу до системи. Додатково передбачено можливість завантаження або оновлення аватара профілю, що дозволяє персоналізувати обліковий запис та покращити його візуальне представлення для інших користувачів платформи (рис. 3.11).

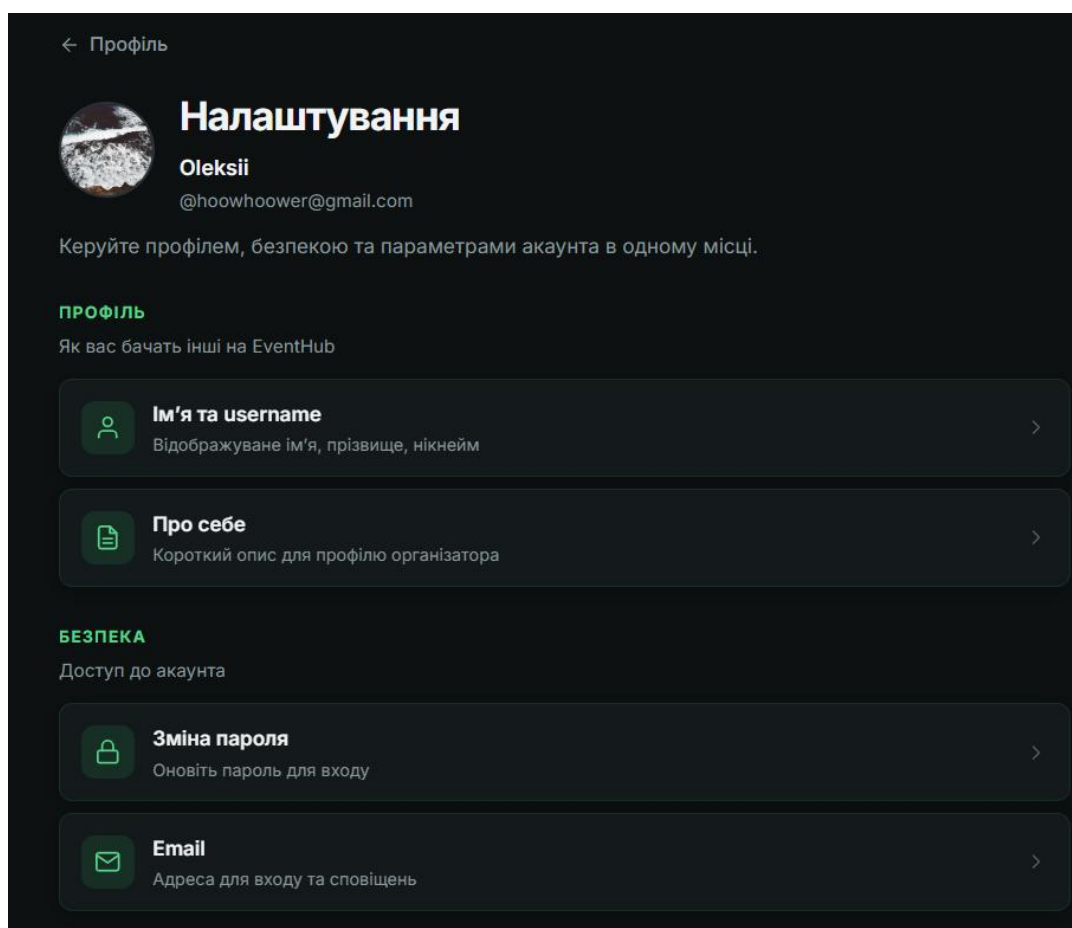


Рис. 3.11. Сторінка з налаштування кабінету користувача

У нижній частині сторінки налаштувань розміщено блоки «Активність» та «Акаунт», які забезпечують швидкий доступ до основних функціональних можливостей платформи. Блок «Активність» об'єднує інструменти для взаємодії користувача з подіями та системою загалом. Зокрема, тут доступні функції створення нової події, налаштування та перегляду рекламних кампаній, перегляд сповіщень, а також доступ до списку квитків і зареєстрованих подій. Окрім

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

цього, користувач може переглядати історію оплат, контролювати стан внутрішнього гаманця та використовувати інструмент перевірки QR-кодів для валідації квитків на подіях.

Блок «Акаунт» призначений для керування параметрами облікового запису. У ньому реалізовано можливість виходу з системи, а також повного видалення акаунта. Дані функції забезпечують користувачу контроль над власним профілем та дозволяють гнучко керувати доступом до платформи (рис. 3.12).

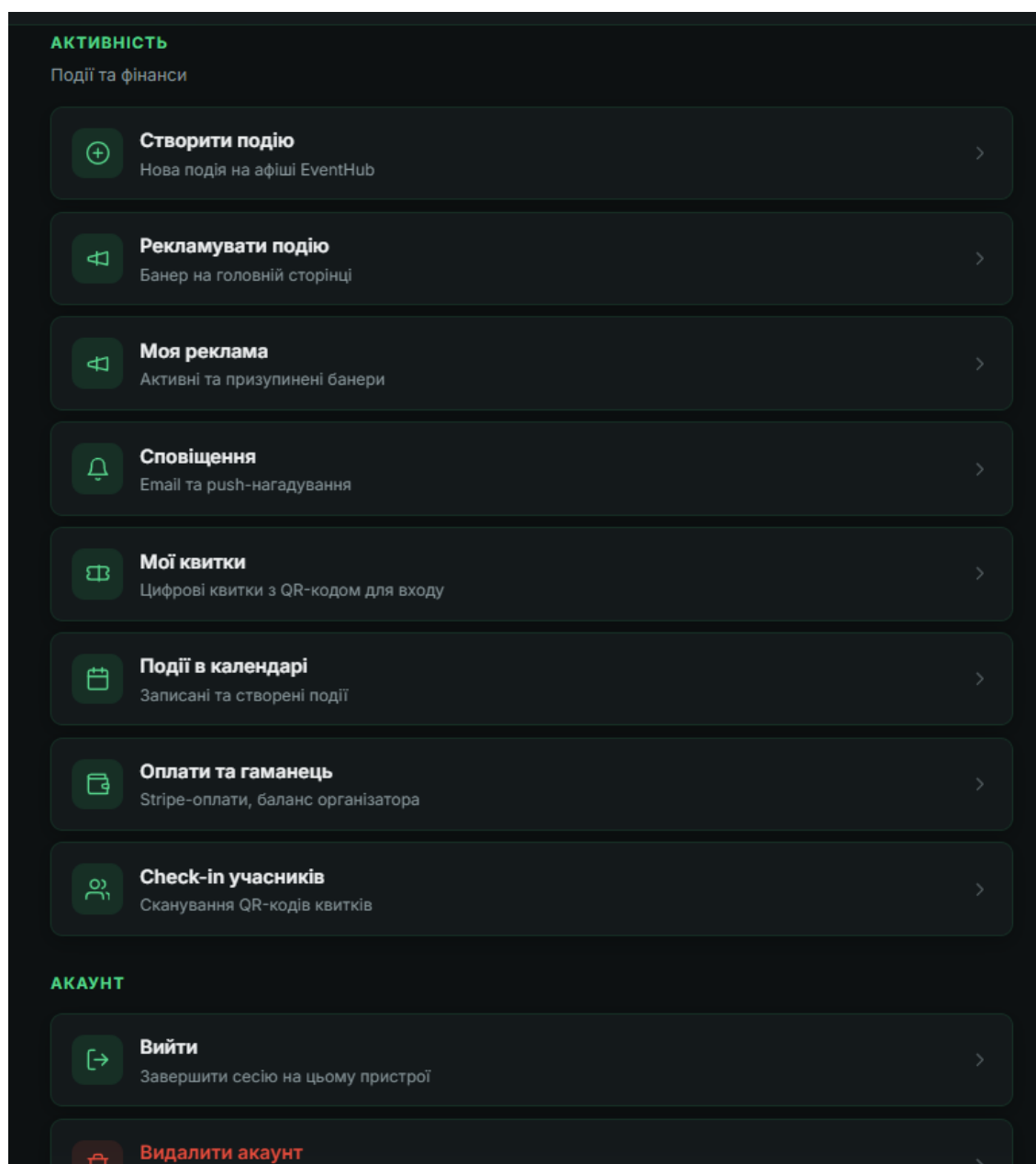


Рис. 3.12. Блоки «Активність» та «Акаунт» особистого кабінету

		Руденко О.І.			ІІЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

Окремий блок особистого кабінету призначений для перегляду інформації про створені, вподобані та зареєстровані події користувача. У цьому розділі зібрано ключові дані щодо всіх заходів, пов'язаних з користувачем, що дозволяє зручно керувати власною активністю на платформі. Окрім подій, у даному блоці також відображаються дані щодо рекламних кампаній, що надає організаторам можливість контролювати процес просування своїх заходів, аналізувати їх ефективність та відстежувати загальну активність на платформі (рис. 3.13).

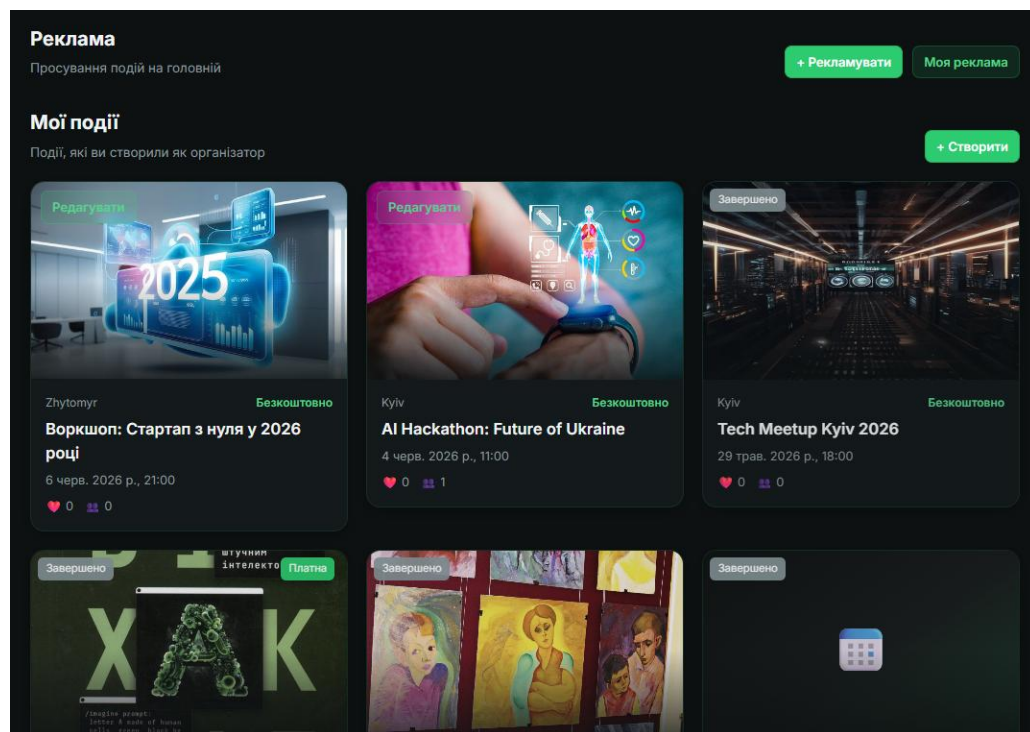


Рис. 3.13. Блок з інформацією про події користувача і рекламу

Натиснувши кнопку «Створити» в блоці з інформацією про події, користувач переходить до форми створення нового івенту. Дана форма дозволяє організатору задавати основні параметри заходу, зокрема назву, детальний опис, дату та час проведення, локацію, категорії та вартість квитка. Такий підхід забезпечує структуроване введення даних і дозволяє коректно відобразити подію на платформі.

Додатково в процес створення інтегровано можливості штучного інтелекту, які дозволяють автоматично генерувати текстовий опис події та тематичну обкладинку на основі введеної користувачем інформації. Це значно спрощує підготовку контенту, скорочує час створення події та підвищує якість її

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

оформлення, навіть за відсутності заздалегідь підготовлених матеріалів (рис. 3.14).

Рис. 3.14. Створення події

Після введення назви та вибору категорії події користувач може скористатися функцією «Згенерувати за допомогою AI» для автоматичного формування опису заходу. У разі необхідності отриманий текст може бути відредагований або доповнений перед публікацією. Крім того, за наявності текстового опису бажаної обкладинки система надає можливість генерації відповідного зображення, яке використовується як візуальне оформлення події. Це дозволяє швидко отримати повністю оформлену сторінку заходу, готову до публікації та подальшого використання на платформі (рис. 3.15).

		Руденко О.І.			ІПЗ.КР.Б–121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

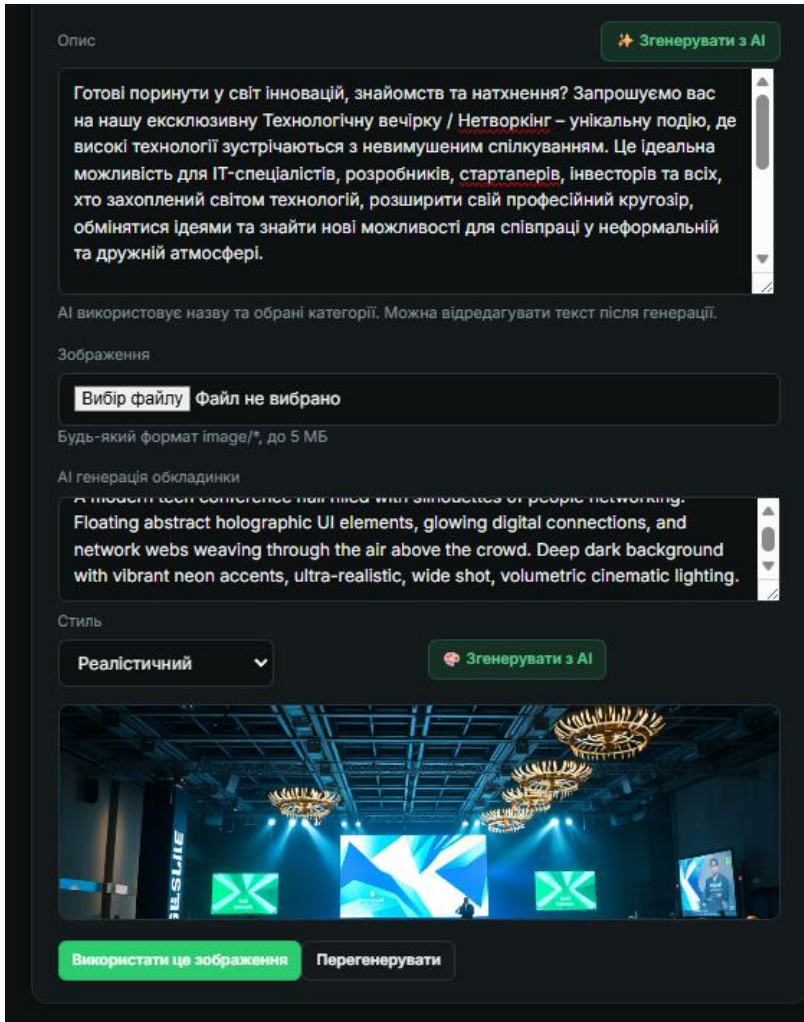


Рис. 3.15. Згенерований опис і банер події

Створена подія після публікації автоматично відображається в загальному каталозі на головній сторінці платформи, де стає доступною для перегляду та взаємодії з боку користувачів (рис.3.16).

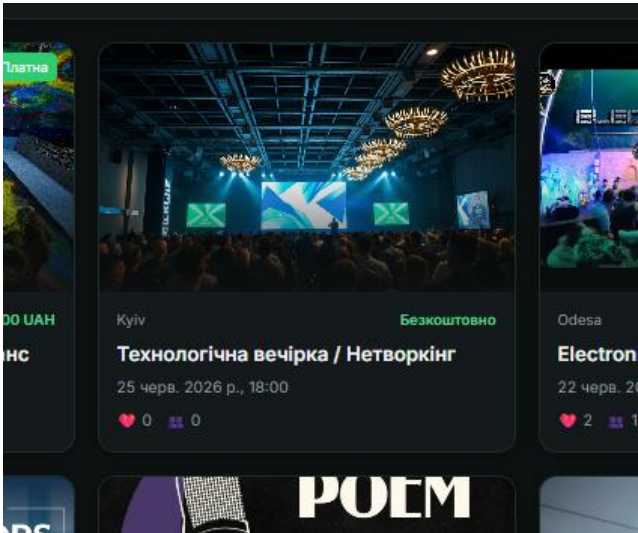


Рис. 3.16. Створена подія на головній сторінці

Редагування події дозволяє оновлювати інформацію про вже створений захід (рис. 3.17).

— До події

Видалити

Редагувати подію

Оновіть деталі — зміни одразу на афіші.

ОСНОВНЕ

Назва події

Технологічна вечірка / Нетворкінг

Опис

Згенерувати з AI

Готові поринути у світ інновацій, знайомств та натхнення? Запрошуємо вас на нашу ексклюзивну Технологічну вечірку / Нетворкінг – унікальну подію, де високі технології зустрічаються з невимушеним спілкуванням. Це ідеальна можливість для IT-спеціалістів, розробників, стартаперів, інвесторів та всіх, хто захоплений світом технологій, розширити свій професійний кругозір, обмінятися ідеями та знайти нові можливості для свого бізнесу та кар'єри.

AI використовує назву та обрані категорії. Можна відредагувати текст після генерації.

Зображення

Вибір файлу

Файл не вибрано

Будь-який формат image/*, до 5 МБ

Рис. 3.17. Сторінка редагування створеного заходу

Система реклами дозволяє створювати та керувати промоційними банерами для подій (рис. 3.18).

Рекламувати подію

Банер з постера та заголовка з'явиться в блоці «Рекомендовано» на головній. Вартість: 99.00 UAH на 14 дн.

Подія

Технологічна вечірка / Нетворкінг

Заголовок банера

Технологічна вечірка / Нетворкінг

☒ Згенерувати банер з постера події

Або своє зображення

Вибір файлу

Файл не вибрано

Прев'ю банера

Оплатити 99.00 UAH

Моя реклама

Рис. 3.18. Сторінка створення реклами

Після успішного здійснення оплати автор отримує доступ до списку всіх створених ним рекламних кампаній, де відображається основна інформація про кожну з них, зокрема прив’язана подія, період дії та поточний статус. Це дозволяє організатору контролювати ефективність просування своїх заходів та керувати рекламною активністю на платформі (рис. 3.19).

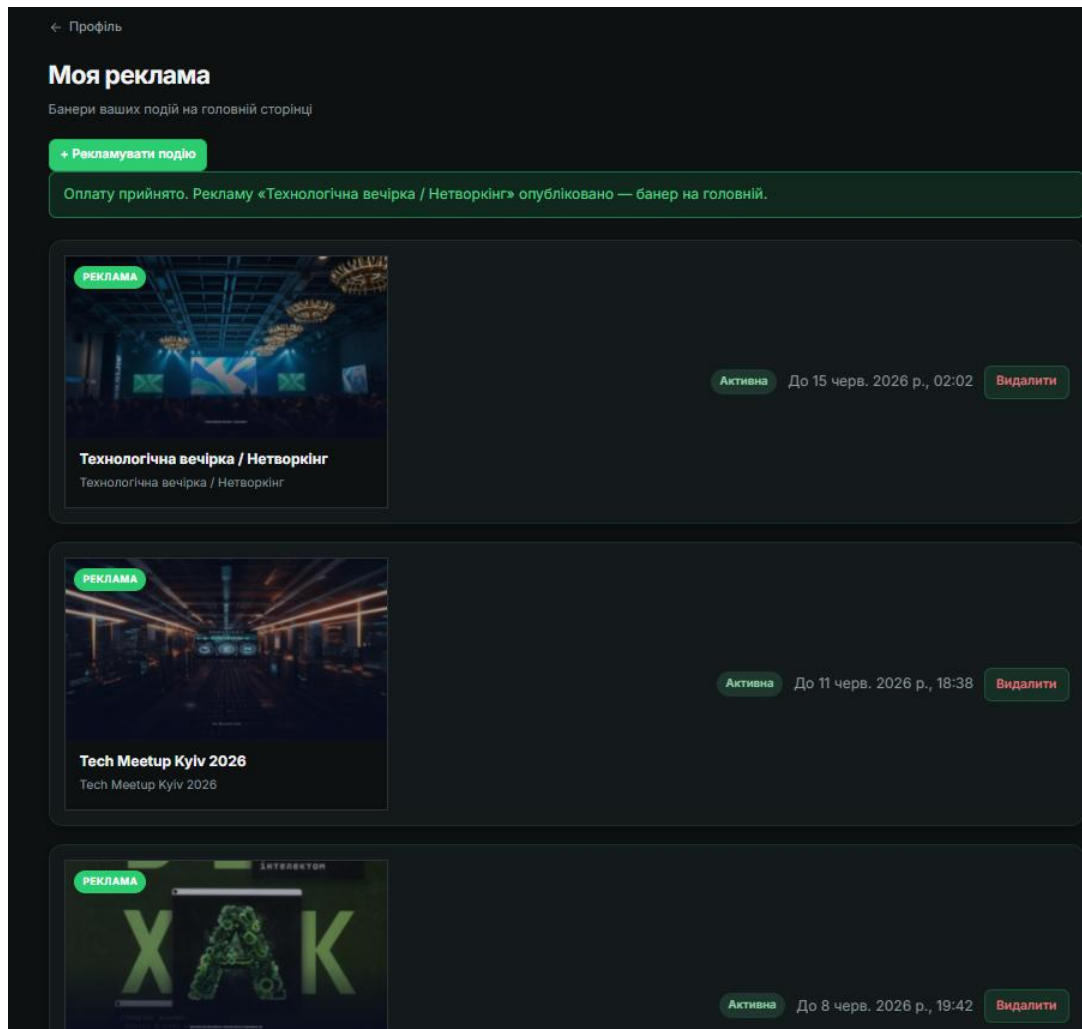


Рис. 3.19. Сторінка з усіма рекламними кампаніями автора

Гаманець організатора містить інформацію про поточний баланс користувача, який формується на основі надходжень від продажу квитків та інших платних послуг платформи. Окремо відображається історія фінансових операцій, що включає всі зарахування, списання, комісії платформи, а також повернення коштів у разі скасування подій або платежів. Крім того, у межах гаманця реалізовано механізм запитів на виведення коштів, який дозволяє

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

організатору ініціювати вивід зароблених коштів за допомогою автоматизованої системи виплат (рис. 3.20).

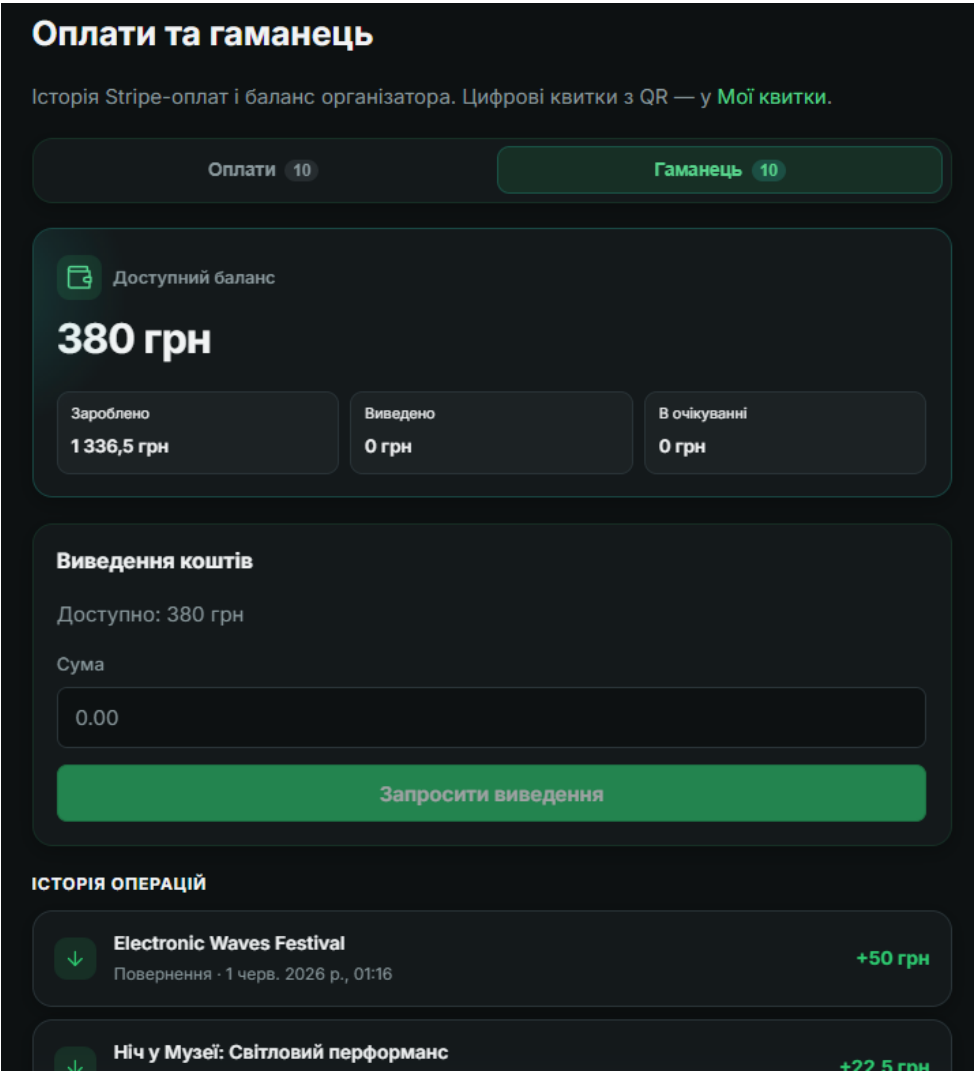


Рис. 3.20. Гаманець організатора

Адміністратор системи має доступ до спеціалізованої адміністративної панелі, яка призначена для керування основними сутностями платформи. Після авторизації адміністратор може переглядати, створювати, редагувати та видаляти записи, пов’язані з користувачами, подіями, платежами, квитками, рекламними кампаніями та сповіщеннями. Також передбачені інструменти пошуку, сортування та фільтрації даних, що спрощують моніторинг роботи системи та її адміністрування. Окремо в адміністративній панелі відображається загальна статистика системи, яка включає ключові показники діяльності платформи, зокрема кількість зареєстрованих користувачів, створених подій,

проведених реєстрацій, успішних платежів та активних рекламних кампаній, що дозволяє оперативно оцінювати стан і навантаження системи (рис. 3.21).

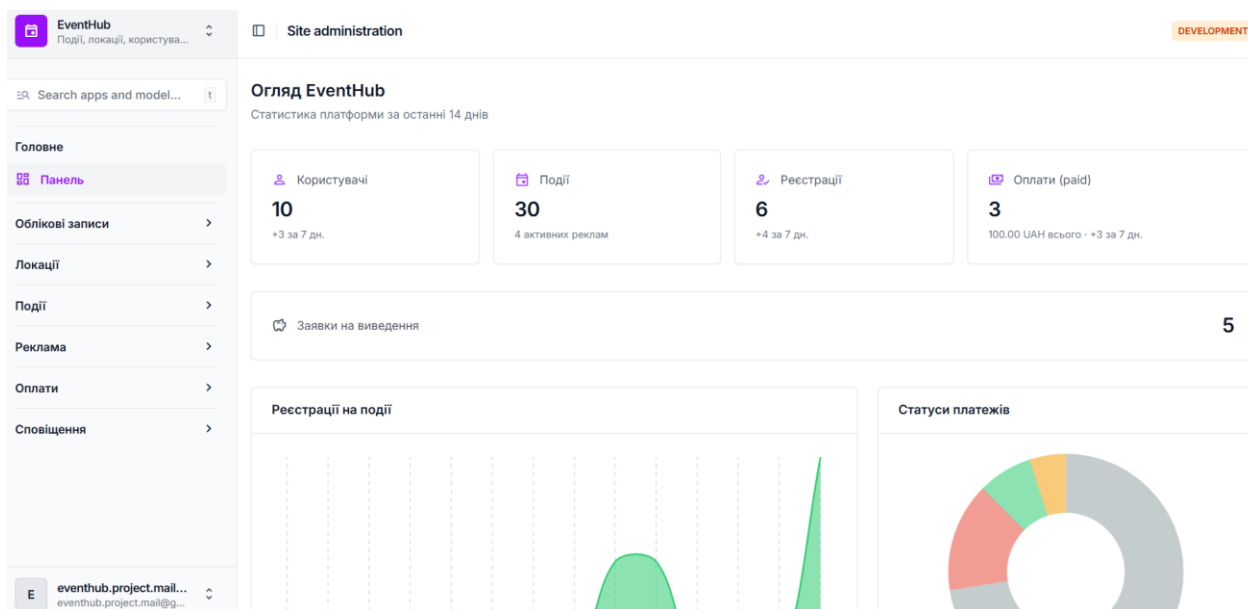


Рис. 3.21. Блок статистики в адміністративній панелі

Для кожного модуля доступні окремі сторінки зі списками записів. Адміністратор може переглядати детальну інформацію про об'єкти системи, змінювати їх параметри та контролювати коректність даних. Використання адміністративної панелі дозволяє здійснювати централізоване керування платформою без необхідності взаємодії з API або базою даних напряму (рис. 3.22).

Events > Events

Список подій Категорії Коментарі Лайки Реєстрації

← All dates April 2026 May 2026 June 2026

Title	City	Location	Author	Start date	End date	Comments count	Max attendees	Created at
Ніч у Музеї: Світловий перформанс	Kyiv	Мистецький арсенал (Kyiv)	eventhub.project.mail@gmail.com	June 27, 2026, midnight	June 29, 2026, midnight	10	50	May 2026 12:42
Технологічна вечірка / Нетворкінг	Kyiv	Майдан Незалежності (Kyiv)	hoowhoower@gmail.com	June 25, 2026, 3 p.m.	June 25, 2026, 10:49 p.m.	0	-	May 2026 10:54
Electronic Waves Festival	Odesa	Ланжерон (Odesa)	ipz222_roi@student.ztu.ua	June 22, 2026, midnight	July 22, 2026, midnight	1	-	June 2026 midn
DevOps UA Summit 2026	Kyiv	НСК «Олімпійський» (Kyiv)	ipz222_mvyyu@student.ztu.edu.ua	June 14, 2026, 3 p.m.	June 14, 2026, 5 p.m.	0	-	May 2026 6:40
Поетичний батл: Вільні Стріхи	Lviv	Львівський палац мистецтв (Lviv)	eventhub.project.mail@gmail.com	June 9, 2026, midnight	June 25, 2026, 3 a.m.	0	100	Marc 2026 3:47

Рис. 3.22. Списки даних в адміністративній панелі

3.3 Тестування роботи програмного продукту

Після завершення розробки програмно-інформаційного комплексу було проведено функціональне тестування основних можливостей системи. Метою тестування було перевірити коректність роботи всіх модулів, відповідність реалізованого функціоналу вимогам, стабільність взаємодії між клієнтською та серверною частинами, а також інтеграцію із зовнішніми сервісами. Тестування виконувалося вручну у двох середовищах: локальному (React dev server + Django runserver) та продакшен-середовищі (Vercel + Fly.io + Neon + Cloudflare R2), що дозволило оцінити працездатність системи як на етапі розробки, так і після розгортання.

Перевірка здійснювалася за сценаріями користувача (use case testing): для кожного тесту фіксувалися вхідні дані, очікуваний результат, фактичний результат і статус виконання. Окремо перевірялися позитивні сценарії (успішне виконання операцій) та негативні (відхилення при некоректних даних або порушенні бізнес-правил). У разі виявлення помилок виконувалося їх усунення з повторним проходженням відповідних сценаріїв.

Особливу увагу приділено перевірці авторизації та реєстрації користувачів (email-код, JWT, refresh-токен), доступу до захищених сторінок через ProtectedRoute, створенню та редагуванню подій, а також процесу реєстрації на події з урахуванням обмежень (закриття реєстрації після початку події, ліміт місць, заборона повторної реєстрації). Окремо протестовано сценарії оплати квитків через Stripe Checkout та внутрішній гаманець, включаючи обробку webhook-подій, розподіл коштів між організатором і платформою, генерацію цифрових квитків із QR-кодом та створення сповіщень.

Додатково перевірено роботу каталогу подій із пошуком і фільтрацією, стрічки трендових і рекламних подій, перегляду деталей події, соціальних взаємодій (лайки, коментарі, відгуки), календаря користувача (створені, зареєстровані та вподобані події), публічних профілів організаторів, історії сповіщень, адміністративної панелі Django Unfold, а також AI-функцій генерації

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

описів (Gemini) і зображень (Hugging Face). Окремо перевірено коректність відображення описів після нормалізації HTML-вмісту.

Результати проведеного тестування наведено у табл. 3.1.

Таблиця 3.1

Результати тестування програмного забезпечення

Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
Авторизація користувача	Email та код підтвердження	Успішний вхід та доступ до профілю	Користувач авторизується, відкривається профіль	Пройдено
Реєстрація користувача	Email, код, пароль	Створення акаунта та видача JWT	Акаунт створено, токен видано	Пройдено
Доступ до захищеної сторінки	Перехід без авторизації	Перенаправлення на сторінку входу	Користувач перенаправлений на /login	Пройдено
Перегляд каталогу подій	Відкриття головної сторінки	Відображення списку подій	Події коректно завантажуються	Пройдено
Фільтрація подій	Місто, категорія, дата	Відфільтрований список подій	Фільтрація працює коректно	Пройдено
Перегляд події	Натискання на подію	Відображення деталей події	Сторінка події відкривається	Пройдено
Лайк події	Натискання «Лайкнути»	Лайк збережено, лічильник оновлено	Лайк додається, організатор отримує сповіщення	Пройдено
Створення події	Форма з AI-генерацією	Подія створена та збережена	Подія успішно додається	Пройдено
Реєстрація на безкоштовну подію	Клік Зареєструватися	Створення реєстрації	Реєстрація створена	Пройдено

Реєстрація на завершеному подію	Спроба реєстрації	Відмова з повідомленням про закриття реєстрацію	Реєстрація заблокована	Пройдено
Оплата квитка (Stripe)	Перехід на оплату	Успішна оплата та створення квитка	Оплата підтверджена, квиток створено	Пройдено
Оплата квитка (гаманець)	Достатній баланс на рахунку	Миттєва оплата та реєстрація	Кошти списано, квиток створено	Пройдено
Перегляд квитка	UUID квитка	Відображення QR-коду	Квиток відкривається коректно	Пройдено
Check-in квитка	Сканування QR-коду	Підтвердження відвідування	Вхід підтверджено системою	Пройдено
Створення події	Форма з AI-генерацією	Подія створена та збережена	Подія успішно додається	Пройдено
Реклама події	Оплата промо через Stripe	Активна рекламна кампанія	Подія відображається у промо-блоці	Пройдено
Адміністратив на панель	Вхід адміністратора	Доступ до керування даними	Панель працює стабільно	Пройдено
Перегляд профілю організатора	ID користувача	Відображення подій організатора	Дані коректно відображаються	Пройдено

За результатами тестування всі сценарії виконано успішно. Система коректно реалізує ключові бізнес-процеси: автентифікацію, роботу з подіями, реєстрацію та оплату, генерацію квитків, соціальну взаємодію та адміністрування. Виявлені дефекти усунуто та повторно перевірено. Результати підтверджують працездатність комплексу та його готовність до експлуатації в хмарному середовищі.

Висновки до розділу 3

У третьому розділі було детально розглянуто структуру користувацького інтерфейсу та порядок взаємодії користувача з програмно-інформаційним комплексом для організації та адміністрування розважальних заходів, Проаналізовано архітектуру розгортання системи та представлено діаграму розгортання, яка відображає фізичну взаємодію між клієнтською частиною, сервером додатку на базі Django REST Framework, базою даних PostgreSQL, об'єктним сховищем Cloudflare та зовнішніми сервісами, зокрема Stripe, Google Gemini та Hugging Face API.

У межах розділу описано основні сторінки користувацького інтерфейсу, включаючи головну сторінку з каталогом подій і системою фільтрації, сторінку детального перегляду події, публічні профілі організаторів, сторінки авторизації та реєстрації, а також розділи, пов'язані з управлінням квитками та оплатами. Окрему увагу приділено функціоналу платних подій, де передбачено попередню оплату квитків через платіжну систему Stripe або внутрішній гаманець користувача з подальшим автоматичним створенням реєстрації та цифрового квитка.

Представлено порядок встановлення та налаштування системи, включаючи процес розгортання клієнтської та серверної частин, а також інтеграцію зовнішніх сервісів. Проведено тестування основних функцій вебзастосунку, яке підтвердило коректну роботу всіх ключових сценаріїв використання та стабільність системи в цілому.

Отже, реалізована система повністю відповідає поставленим функціональним вимогам, забезпечує зручний інтерфейс для користувачів та організаторів, підтримує оплату квитків, створення та управління подіями, а також інтеграцію з зовнішніми сервісами, що робить платформу готовою до практичного використання.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

ВИСНОВКИ

У межах виконання даної кваліфікаційної роботи було розроблено повноцінний програмно-інформаційний комплекс для організації, пошуку та відвідування розважальних заходів. Проект охоплює повний цикл створення сучасного вебсервісу: від аналізу вимог і проектування архітектури до реалізації функціоналу, інтеграції зовнішніх сервісів та тестування системи.

Основною метою роботи було створення зручної платформи, яка дозволяє користувачам переглядати події, здійснювати пошук і фільтрацію за різними параметрами, а також реєструватися та оплачувати участь у заходах онлайн. Для організаторів передбачено можливість створення та редагування подій, керування реєстраціями, фінансовими операціями та рекламними кампаніями.

У системі реалізовано клієнт-серверну архітектуру з використанням React на стороні клієнта та Django REST Framework на сервері. Для зберігання даних використовується PostgreSQL, а медіафайли зберігаються в об'єктному сховищі Cloudflare R2. Автентифікація користувачів побудована на основі JWT-токенів, що забезпечує безпечну взаємодію з API.

Окрему увагу приділено інтеграції платіжної системи Stripe, яка забезпечує обробку оплат квитків та рекламних послуг. Після успішної транзакції система автоматично створює реєстрацію користувача на подію та формує цифровий квиток. Додатково реалізовано використання AI-сервісів для автоматичної генерації описів подій та обкладинок, що спрощує процес створення контенту для організаторів.

Інтерфейс системи є адаптивним та зручним у використанні, а проведене тестування підтвердило коректність роботи основних функціональних сценаріїв та стабільність системи в цілому. Реалізований вебзастосунок забезпечує ефективну взаємодію між користувачами та організаторами заходів.

Отриманий результат є сучасною платформою для управління подіями, яка має потенціал до подальшого розвитку, зокрема розширення аналітичних можливостей, системи рекомендацій та інтеграції нових сервісів.

		Руденко О.І.			ІПЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Руденко О. І., Лімінович І. Д., Локтікова Т. М. Розробка серверної частини платформи для управління розважальними заходами // Всеукраїнська науково-практична онлайн-конференція здобувачів вищої освіти і молодих учених, присвячена Дню науки. 15 травня 2026 р.
2. REST API Tutorial. URL: <https://restfulapi.net/> (дата звернення: 03.06.2026).
3. 2Event. URL: <https://2event.com.ua/> (дата звернення: 03.06.2026).
4. Eventbrite. URL: <https://www.eventbrite.com/> (дата звернення: 03.06.2026).
5. Meetup. URL: <https://www.meetup.com/home/> (дата звернення: 03.06.2026).
6. OWASP Top Ten Web Application Security Risks / OWASP Foundation. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 03.06.2026).
7. Python 3.13.3 Documentation / Python Software Foundation. URL: <https://docs.python.org/3/> (дата звернення: 03.06.2026).
8. Django REST Framework Documentation / Encode Core Team. URL: <https://www.django-rest-framework.org/> (дата звернення: 03.06.2026).
9. JSON Web Tokens (JWT). Introduction and Documentation. URL: <https://auth0.com/docs/secure/tokens/json-web-tokens> (дата звернення: 03.06.2026).
10. Cloudflare R2 / Cloudflare. URL: <https://www.cloudflare.com/products/r2/> (дата звернення: 03.06.2026).
11. Django Unfold Admin Theme. GitHub Repository. URL: <https://github.com/unfoldadmin/django-unfold> (дата звернення: 03.06.2026).
12. PostgreSQL Documentation / PostgreSQL Global Development Group. URL: <https://www.postgresql.org/docs/> (дата звернення: 03.06.2026).
13. Neon Database / Neon Inc. URL: <https://neon.tech/> (дата звернення: 03.06.2026).
14. Django Documentation / Django Software Foundation. URL: <https://docs.djangoproject.com/> (дата звернення: 03.06.2026).
15. Stripe API Documentation / Stripe. URL: <https://docs.stripe.com/> (дата звернення: 03.06.2026).

		Руденко О.І.			ІІЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

- 16.Stripe / Stripe Inc. URL: <https://stripe.com/> (дата звернення: 03.06.2026).
- 17.Google Gemini API / Google. URL: <https://ai.google.dev/> (дата звернення: 03.06.2026).
- 18.Hugging Face. URL: <https://huggingface.co/> (дата звернення: 03.06.2026).
- 19.React Documentation / Meta Open Source. URL: <https://react.dev/> (дата звернення: 03.06.2026).
- 20.Vercel Documentation / Vercel Inc. URL: <https://vercel.com/docs> (дата звернення: 03.06.2026).
- 21.Fly.io Documentation. URL: <https://fly.io/docs/> (дата звернення: 03.06.2026).
- 22.Docker Documentation / Docker Inc. URL: <https://docs.docker.com/> (дата звернення: 03.06.2026).
- 23.Gunicorn Documentation. URL: <https://docs.gunicorn.org/> (дата звернення: 03.06.2026).
- 24.Gemini API Documentation and Specifications / DeepWiki. URL: <https://deepwiki.com/google-gemini/gemini-skills/3.4-api-documentation-and-specifications> (дата звернення: 03.06.2026).
- 25.Dirgantara D. P., Kusumo D. S., Utomo R. G. Docker-Based Monolithic and Microservices Architecture Performance Comparison. URL: <https://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/1338> (дата звернення: 03.06.2026).
- 26.Cloudflare Developers Documentation / Cloudflare. URL: <https://developers.cloudflare.com/> (дата звернення: 03.06.2026).
- 27.Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley Professional, 2002.
- 28.Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O'Reilly Media, 2020.
- 29.Richardson L., Amundsen M. RESTful Web APIs: Services for a Changing World. Sebastopol : O'Reilly Media, 2013. 406 с.
- 30.Lutz M. Learning Python. Sebastopol : O'Reilly Media, 2020. 1552 с.

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

ДОДАТКИ

		Руденко О.І.			ІПЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

Лайки/Коментарі/Відгуки

Лістинг А.1 Поставити/видалити лайк:

```

def try_create_event_like(user, event):
    """
    Returns (success, payload, status_code).
    On success: (True, like_instance, 201)
    On failure: (False, error_dict, status_code)
    """
    if EventLike.objects.filter(user=user, event=event).exists():
        return False, {"non_field_errors": ["Ви вже лайкнули цю подію."]}, 400
    like = EventLike.objects.create(user=user, event=event)
    NotificationService.notify_event_author(
        event=event,
        actor=user,
        title="New Like",
        message=f"{user.email} liked your event",
        data={"type": "event_action"}
    )
    return True, like, 201

def try_delete_event_like(user, event):
    """Returns (success, error_body_or_none, status_code)."""
    deleted_count, _ = EventLike.objects.filter(user=user, event=event).delete()
    if not deleted_count:
        return False, {"detail": "Лайк не знайдено."}, 404
    return True, None, 204

```

Лістинг А.2 Створення/Видалення відгуків:

```
def event_is_in_past(event: Event) -> bool:
```

```
    """Подія вважається завершеною, коли настав час end_date."""
```

```
    return event.end_date <= timezone.now()
```

```
def user_was_registered_for_event(user, event: Event) -> bool:
```

```
    """
```

```
    Користувач був зареєстрований: активна реєстрація або оплачений квиток  
(paid),
```

```
    якщо організатор видалив учасника після події.
```

```
    """
```

```
    if EventRegistration.objects.filter(user=user, event=event).exists():
```

```
        return True
```

```
    from payments.models import Payment
```

```
    return Payment.objects.filter(
```

```
        user=user,
```

```
        event=event,
```

```
        status=Payment.Status.PAID,
```

```
    ).exists()
```

```
def validate_review_creation(user, event: Event):
```

```
    """
```

```
    Returns None if review can be created, else (error_body, status_code).
```

```
    """
```

```
    if not event_is_in_past(event):
```

```
        return (
```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

        {"detail": "Відгук можна залишити лише після завершення події."},
        400,
    )

    if not user_was_registered_for_event(user, event):
        return (
            {"detail": "Відгук можуть залишати лише учасники, зареєстровані на цю
подію."},
            403,
        )

    if EventReview.objects.filter(user=user, event=event).exists():
        return (
            {"detail": "Ви вже залишили відгук на цю подію."},
            400,
        )

    return None

    queryset = EventReview.objects.select_related("user", "user__profile",
"event").all()
    serializer_class = EventReviewSerializer
    permission_classes = [permissions.IsAuthenticated, IsReviewAuthorOrAdmin]

    @swagger_auto_schema(
        operation_summary="Видалити відгук",
        operation_description="Доступно автору відгуку або адміністратору.",
        responses={
            204: openapi.Response(description="Відгук успішно видалено"),
            403: openapi.Response(description="Недостатньо прав"),

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ПЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        404: openapi.Response(description="Відгук не знайдено"),
    },
)
def destroy(self, request, *args, **kwargs):
    return super().destroy(request, *args, **kwargs)

```

Лістинг А.3 Створення/видалення коментарів:

```

@api_action(detail=True, methods=["get", "post"], url_path="comments")
def comments(self, request, pk=None):
    event = self.get_object()

    if request.method == "GET":
        queryset = event.comments.select_related("author").all()
        page = self.paginate_queryset(queryset)
        if page is not None:
            serializer = CommentSerializer(page, many=True, context={"request":
request})
            return self.get_paginated_response(serializer.data)
        serializer = CommentSerializer(queryset, many=True, context={"request":
request})
        return Response(serializer.data, status=status.HTTP_200_OK)

    serializer = CommentSerializer(data=request.data, context={"request": request})
    serializer.is_valid(raise_exception=True)
    comment = serializer.save(author=request.user, event=event)
    after_comment_created(event, request.user)
    read_serializer = CommentSerializer(comment, context={"request": request})
    return Response(read_serializer.data, status=status.HTTP_201_CREATED)

```

```

class CommentViewSet(mixins.DestroyModelMixin, viewsets.GenericViewSet):
    queryset = Comment.objects.select_related("author", "event").all()
    serializer_class = CommentSerializer
    permission_classes = [permissions.IsAuthenticated, IsCommentAuthorOrAdmin]

    @swagger_auto_schema(
        operation_summary="Видалити коментар",
        operation_description="Видаляє коментар. Доступно автору коментаря або адмініу.",
        responses={
            204: openapi.Response(description="Коментар успішно видалено"),
            403: openapi.Response(description="Недостатньо прав"),
            404: openapi.Response(description="Коментар не знайдено"),
        },
    )

    def destroy(self, request, *args, **kwargs):
        instance = self.get_object()
        event_id = instance.event_id
        self.check_object_permissions(request, instance)
        instance.delete()
        decrement_event_comments_count(event_id)
        return Response(status=status.HTTP_204_NO_CONTENT)

```

Управління платежами

Лістинг Б.1 Створення оплат:

```

@transaction.atomic
def create_payment_for_event(user, event_id: int, *, client: str = "web"):
    """
    Creates Payment (pending) + Stripe Checkout Session.
    Returns (response_dict, None) or (None, (error_body, status_code)).
    """
    if not is_stripe_configured():
        return None, (
            {"detail": "Stripe не налаштовано. Зверніться до адміністратора."},
            503,
        )

    event = _get_event_or_none(event_id)
    if event is None:
        return None, ({"detail": "Подію не знайдено."}, 404)

    err = _validate_payment_creation(event, user)
    if err is not None:
        return None, err

    pending = Payment.objects.filter(
        user=user,
        event=event,
        status=Payment.Status.PENDING,
    ).first()
    if pending and pending.checkout_url:
        # Reuse open Stripe session if still valid.

```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

if pending.stripe_session_id:
    try:
        session = retrieve_checkout_session(pending.stripe_session_id)
        if session.status == "open" and session.url:
            return _pending_checkout_response(pending), None
    except stripe.error.StripeError:
        logger.warning(
            "Could not retrieve Stripe session %s for payment %s",
            pending.stripe_session_id,
            pending.id,
        )
    return _pending_checkout_response(pending), None

```

```

try:
    payment = Payment.objects.create(
        user=user,
        event=event,
        organizer=event.author,
        amount=event.ticket_price,
        currency=event.currency,
        status=Payment.Status.PENDING,
        provider_name="stripe",
    )
    session = create_checkout_session(
        payment_id=payment.id,
        amount=payment.amount,
        currency=payment.currency,
        event_title=event.title,
        customer_email=user.email,
        client=client,

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```

)
payment.checkout_url = session.url or ""
payment.stripe_session_id = session.id
payment.save(update_fields=["checkout_url", "stripe_session_id"])
except IntegrityError:
    pending = Payment.objects.filter(
        user=user,
        event=event,
        status=Payment.Status.PENDING,
    ).first()
    if pending and pending.checkout_url:
        return _pending_checkout_response(pending), None
    return None, ({ "detail": "Оплату для цієї події вже ініційовано." }, 400)
except stripe.error.InvalidRequestError as exc:
    logger.warning("Stripe invalid request: %s", exc)
    if "payment" in locals():
        payment.status = Payment.Status.FAILED
        payment.save(update_fields=["status"])
        user_message = str(exc.user_message) if getattr(exc, "user_message", None)
    else str(exc)
    if "at least" in user_message.lower() or "50 cent" in user_message.lower():
        user_message = validate_checkout_amount(event.ticket_price,
        event.currency) or user_message
    return None, ({ "detail": user_message }, 400)
except stripe.error.StripeError:
    logger.exception("Stripe Checkout Session creation failed")
    if "payment" in locals():
        payment.status = Payment.Status.FAILED
        payment.save(update_fields=["status"])
    return None, (

```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІІЗ	Арк.
		Локтікова Т.М.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {"detail": "Не вдалося створити сесію оплати Stripe."},
        502,
    )
except ValueError as exc:
    logger.warning("Checkout validation failed: %s", exc)
    if "payment" in locals():
        payment.delete()
    return None, ({"detail": str(exc)}, 400)
except RuntimeError as exc:
    logger.exception("Stripe configuration error: %s", exc)
    return None, (
        {"detail": "Платіжний сервіс тимчасово недоступний."},
        503,
    )

return _pending_checkout_response(payment), None

```

Лістинг Б.2 Підтвердження оплати + реєстрація:

@transaction.atomic

```

def _fulfill_paid_payment(payment: Payment, provider_payment_id: str = "") ->
None:

```

```

    """Mark paid, split fees, credit organizer wallet, register attendee (idempotent)."""

```

```

    was_already_paid = payment.status == Payment.Status.PAID

```

```

    if not was_already_paid:

```

```

        platform_fee, organizer_income = calculate_fee_split(payment.amount)

```

```

        payment.platform_fee = platform_fee

```

```

        payment.organizer_income = organizer_income

```

```

        payment.status = Payment.Status.PAID

```

```

        payment.paid_at = timezone.now()

```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІЗ	Арк.
		Локтікова Т.М.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if provider_payment_id:
    payment.provider_payment_id = provider_payment_id
payment.save(
    update_fields=[
        "status",
        "paid_at",
        "provider_payment_id",
        "platform_fee",
        "organizer_income",
    ]
)

distribute_funds(payment)
_ensure_registration_after_payment(payment)

```

ЛІСТИНГ Б.3 Webhook Stripe:

```

@transaction.atomic
def _handle_checkout_session_completed(session: dict) -> bool:
    payment = _payment_from_session_metadata(session)
    if payment is None:
        return False

    # Only webhook confirms payment — never trust client redirect alone.
    if session.get("payment_status") != "paid":
        logger.info(
            "checkout.session.completed but payment_status=%s payment=%s",
            session.get("payment_status"),
            payment.id,
        )

    return True

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

provider_id = session.get("payment_intent") or session.get("id") or ""
_fulfill_paid_payment(payment, provider_payment_id=str(provider_id))
return True

```

Лістинг А.4 Stripe API (сесія + перевірка підпису):

```

def validate_checkout_amount(amount: Decimal, currency: str) -> str | None:
    """
    Returns Ukrainian error message if amount is below Stripe minimum, else None.
    """
    minimum = get_minimum_ticket_amount(currency)
    if amount < minimum:
        return (
            f"Мінімальна ціна квитка для Stripe — {minimum} {currency.upper()} "
            f"(еквівалент ~0.50 USD). Поточна сума: {amount}."
        )
    return None

```

		Руденко О.І.			ІПЗ.КР.Б-121-26-ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

AI-генерація

Лістинг В.1 Генерація опису:

```
def generate_event_description(title: str, category: str) -> str:
```

```
    prompt = f"""
```

```
Create a professional event description.
```

```
Title: {title}
```

```
Category: {category}
```

```
Write 1-2 paragraphs, engaging tone.
```

```
"""
```

```
    models = [
```

```
        "models/gemini-2.5-flash",
```

```
        "models/gemini-2.5-pro",
```

```
        "models/gemini-2.0-flash",
```

```
    ]
```

```
    for model in models:
```

```
        try:
```

```
            response = client.models.generate_content(
```

```
                model=model,
```

```
                contents=prompt
```

```
            )
```

```
            return response.text
```

```
        except Exception:
```

```
            continue
```

```
    return "AI generation temporarily unavailable."
```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		95

Лістинг В.2 Генерація картинок:

```
def _svg_fallback(prompt: str) -> ImageGenerationResult:
    safe = html.escape((prompt or "Event").strip()[:120])
    svg = f'<svg xmlns="http://www.w3.org/2000/svg" width="1024"
height="576">
<defs>
<linearGradient id="g" x1="0" y1="0" x2="1" y2="1">
<stop offset="0%" stop-color="#171f2d"/>
<stop offset="100%" stop-color="#2f495e"/>
</linearGradient>
</defs>
<rect width="100%" height="100%" fill="url(#g)"/>
<text x="64" y="290" fill="#f2f4f8" font-size="42" font-family="Arial,sans-
serif">{safe}</text>
</svg>'

    encoded = base64.b64encode(svg.encode("utf-8")).decode("ascii")
    return {
        "image_base64": f'data:image/svg+xml;base64,{encoded}',
        "fallback": True,
        "provider": "fallback",
    }
```

```
def _generate_huggingface(prompt: str, style: str) -> ImageGenerationResult:
    api_url = getattr(settings, "IMAGE_GEN_API_URL", "").strip()
    api_token = getattr(settings, "IMAGE_GEN_API_TOKEN", "").strip()
    if not api_url or not api_token:
        raise ValueError("IMAGE_GEN_API_TOKEN is not configured.")
```

```
    read_timeout = int(getattr(settings, "IMAGE_GEN_TIMEOUT", 90))
```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

resp = requests.post(
    api_url,
    json={"inputs": _enhance_prompt(prompt, style)},
    headers={
        "Accept": "image/png",
        "Authorization": f"Bearer {api_token}",
    },
    timeout=(_connect_timeout(), read_timeout),
)
resp.raise_for_status()
content_type = (resp.headers.get("Content-Type") or
"image/png").split(";")[0].strip()
if not content_type.startswith("image/"):
    raise ValueError("HuggingFace returned non-image response.")
return _bytes_to_result(resp.content, content_type, "huggingface")

def generate_event_image(prompt: str, style: str = "realistic") ->
ImageGenerationResult:
    if not prompt or not prompt.strip():
        raise ValueError("Prompt is required.")

    if not getattr(settings, "IMAGE_GEN_API_TOKEN", "").strip():
        logger.error("IMAGE_GEN_API_TOKEN is not configured.")
        return _svg_fallback(prompt)

    try:
        return _generate_huggingface(prompt, style)
    except Exception as exc:
        logger.error("HuggingFace image generation failed: %s", exc)
        return _svg_fallback(prompt)

```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІІЗ	Арк.
		Локтікова Т.М.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

Сповіщення

Лістинг Г.1 Сповіщення юзерів:

```

class NotificationService:
    """
    Notification system without Celery (Render-friendly MVP version)
    """

    # =====
    # PUBLIC API
    # =====

    @staticmethod
    def notify_user(
        user: User,
        title: str,
        message: str,
        data: Dict[str, Any] | None = None
    ):
        """
        1. Save notification to DB
        2. Send push asynchronously via thread
        """

        notification = Notification.objects.create(
            user=user,
            title=title,
            message=message,
        )

```

		Руденко О.І.			ІПЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		98

```

# async push (no Celery)
threading.Thread(
    target=NotificationService.send_push_to_user_tokens,
    args=(user.id, title, message, data),
    daemon=True
).start()

return notification

@staticmethod
def notify_event_author(
    event: Event,
    actor: User,
    title: str,
    message: str,
    data: Dict[str, Any] | None = None
):
    """
    Notify event author (skip self notifications)
    """

    if event.author_id == actor.id:
        return None

    data = data or {}
    data.update({"event_id": str(event.id)})

    return NotificationService.notify_user(
        user=event.author,
        title=title,

```

		Руденко О.І.			ІІЗ.КР.Б-121-26-ІІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

        message=message,
        data=data
    )

```

Лістинг Г.2 Отримання списку сповіщень:

```

class NotificationListView(ListAPIView):
    permission_classes = [permissions.IsAuthenticated]
    serializer_class = NotificationSerializer

    def get_queryset(self):
        return Notification.objects.filter(user=self.request.user).order_by("-created_at")

    @swagger_auto_schema(
        operation_summary="Отримати сповіщення",
        operation_description=(
            "Повертає сповіщення поточного користувача від нових до старих. "
            "Після відповіді всі непрочитані сповіщення користувача позначаються "
            "як прочитані."
        ),
        responses={200: NotificationSerializer(many=True)},
        manual_parameters=[
            openapi.Parameter(
                "page",
                openapi.IN_QUERY,
                description="Номер сторінки (починаючи з 1)",
                type=openapi.TYPE_INTEGER,
            ),
            openapi.Parameter(
                "page_size",
                openapi.IN_QUERY,

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        description="Кількість елементів на сторінку (максимум 100)",
        type=openapi.TYPE_INTEGER,
    ),
],
)
def get(self, request, *args, **kwargs):
    return super().get(request, *args, **kwargs)

def list(self, request, *args, **kwargs):
    queryset = self.filter_queryset(self.get_queryset())
    page = self.paginate_queryset(queryset)
    if page is not None:
        serializer = self.get_serializer(page, many=True)
        serialized_data = serializer.data
        Notification.objects.filter(user=request.user,
is_read=False).update(is_read=True)
        return self.get_paginated_response(serialized_data)
    serializer = self.get_serializer(queryset, many=True)
    serialized_data = serializer.data
    Notification.objects.filter(user=request.user,
is_read=False).update(is_read=True)
    return Response(serialized_data)

```

Лістинг Г.3 Отримання кількості непрочитаних сповіщень:

```

class NotificationUnreadCountView(views.APIView):
    permission_classes = [permissions.IsAuthenticated]

    @swagger_auto_schema(
        operation_summary="Кількість непрочитаних сповіщень",

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		101

```

operation_description="Повертає кількість непрочитаних сповіщень
поточного користувача.",
responses={
    200: openapi.Response(
        "Об'єкт з полем unread_count",
        schema=openapi.Schema(
            type=openapi.TYPE_OBJECT,
            properties={
                "unread_count": openapi.Schema(type=openapi.TYPE_INTEGER),
            },
        ),
    ),
},
)

def get(self, request, *args, **kwargs):
    unread_count = Notification.objects.filter(user=request.user,
is_read=False).count()
    return Response({"unread_count": unread_count})

```

		Руденко О.І.			ІІЗ.КР.Б –121–26–ІЗ	Арк.
		Локтікова Т.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		102