КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

_____

(факультет)

інформаційних технологій

_____

(кафедра)

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Розробка веб-системи автоматизації планування масових заходів»

Василишина Лада Володимирівна

(прізвище, ім’я та по батькові студента повністю)

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Розробка веб-системи автоматизації планування масових заходів»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувачка Василишина Лада Володимирівна
(прізвище, ім’я та по батькові повністю)
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21-1
Керівник Бородавка Є.В.
(прізвище та ініціали)
професор, доктор технічних наук
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Рябчун Ю.В.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Випускова кафедра: інформаційних технологій
Ступінь вищої освіти: «бакалавр»
Спеціальність: 122 «Комп`ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___” _____2025 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Василишина Лада Володимирівна

1. Тема роботи: Розробка веб-системи автоматизації планування масових заходів затверджена наказом ректора КНУБА № 235 від «14» лютого 2025 року
2. Керівник роботи: Бородавка Євгеній Володимирович, д.т.н, професор кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Проєктування архітектури системи
Р.3. Реалізація системи
Р.4. Впровадження системи
5. Інформаційні слайди:
С.1. Визначення цілей дослідження
С.2. Функціональні та нефункціональні вимоги
С.3. DFD-діаграма
С.4.Проєктування бази даних
С.5. Інтерфейс користувача

6. Консультанти розділів кваліфікаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Розділ 1	Гончаренко Т. А.	15.02.2025	
Розділ 2	Бородавка Є. В.	28.04.2025	
Розділ 3	Мацієвський О. О.	15.05.2025	
Розділ 4	Рябчун Ю. В.	31.05.2025	

7. Календарний план виконання кваліфікаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2025 р.
Р. 2. Проєктування архітектури системи	Лютий 2025 р.
Р. 3. Реалізація системи	Березень 2025 р.
Р. 4. Впровадження системи	Квітень 2025 р.
Остаточне оформлення роботи	Травень 2025 р.
Направлення роботи на рецензування	Червень 2025 р.
Попередній захист роботи на кафедрі	Червень 2025 р.

8. Дата видачі завдання: 20.01.2025 р.

Завідувачка _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Керівник _____ Бородавка Є.В.
(підпис) (прізвище та ініціали)

Здобувачка _____ Василишина Л.В.
(підпис) (прізвище та ініціали)

<table>
<tr><td>РЕЗЮМЕ (SUMMARY)
до кваліфікаційної випускної роботи Здобувача:</td><td colspan="3">Василишина Лада Володимирівна

Vasylyshyna Lada</td></tr>
<tr><td>ЗВО</td><td colspan="3">Київський національний університет будівництва і архітектури</td></tr>
<tr><td>Тема (українською та англійською)</td><td colspan="3">Розробка веб-системи автоматизації планування масових заходів.
Development of a web-based system for automating mass event planning.</td></tr>
<tr><td>Освітній ступінь</td><td colspan="3">Бакалавр</td></tr>
<tr><td>Факультет</td><td colspan="3">Автоматизації і інформаційних технологій</td></tr>
<tr><td>Випускова кафедра</td><td colspan="3">Інформаційних технологій</td></tr>
<tr><td>Спеціальність</td><td colspan="3">122 «Комп’ютерні науки»</td></tr>
<tr><td>Освітня програма</td><td colspan="3">Інформаційні управляючі системи та технології</td></tr>
<tr><td>Керівник</td><td colspan="3">Бородавка Євгеній Володимирович</td></tr>
<tr><td rowspan="2">Обсяг роботи:</td><td>пояснювальна записка, стор.</td><td>розділів</td><td>креслень формату А</td></tr>
<tr><td>99</td><td>4</td><td>0</td></tr>
<tr><td>Розділ 1.</td><td colspan="3">Аналіз предметної області та постановка задачі</td></tr>
<tr><td>Розділ 2.</td><td colspan="3">Проєктування архітектури системи</td></tr>
<tr><td>Розділ 3.</td><td colspan="3">Реалізація системи</td></tr>
<tr><td>Розділ 4.</td><td colspan="3">Впровадження системи</td></tr>
<tr><td>Ключові слова:

Keywords:</td><td colspan="3">веб-система, автоматизація, планування, організація подій, база даних, веб-технології.

web system, automation, planning, event organization, database, web technologies.</td></tr>
</table>

Здобувач: _____ /Лада ВАСИЛИШИНА /

Керівник: _____/ Євгеній БОРОДАВКА /

“___” _____ 2025р.

АНОТАЦІЯ

Василишина Л.В. Розробка веб-системи автоматизації планування масових заходів.

Кваліфікаційна випускна робота бакалавра за спеціальністю: 122 «Комп'ютерні науки», освітньо-професійна програма: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Робота присвячена розробці веб-системи автоматизації процесів планування та проведення масових заходів. Описано архітектуру системи, її функціональні та нефункціональні вимоги, алгоритми взаємодії користувачів різних ролей, а також реалізацію ключових функцій. Наведено результати тестування й оцінки, які підтверджують ефективність розробленої системи в умовах практичного використання.

Ключові слова: веб-система, автоматизація, планування, організація подій, база даних, веб-технології.

SUMMARY

Vasylyshyna L.V. Development of a web-based system for automating mass event planning.

Bachelor's thesis for a bachelor's degree in specialty: 122 “Computer Science”, specialization: "Information Management Systems and Technologies - Kyiv National University of Construction and Architecture - Kyiv, 2025.

The work is devoted to the development of a web-based system for automating the processes of planning and organizing mass events. The system's architecture, its functional and non-functional requirements, user interaction algorithms for different roles, as well as the implementation of key features are described. The testing and evaluation results presented confirm the effectiveness of the developed system in practical use.

Keywords: web system, automation, planning, event organization, database, web technologies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД — база даних.

API — Application Programming Interface.

СУБД — система управління базою даних.

HTML — Hypertext Markup Language.

HTTP — Hypertext Transfer Protocol.

CSS — Cascading Style Sheets.

JSON — JavaScript Object Notation.

URL — Uniform Resource Locator.

DFD — Data Flow Diagram.

REST — Representational State Transfer.

SQL — Structured Query Language.

ВСТУП

Перед сучасними організаторами масових заходів стоїть гостре питання підвищення ефективності планування, оптимізації ресурсів та зменшення витрат часу на координацію. Зі зростанням масштабів подій, кількості учасників та рівня складності організації, традиційні методи управління стають недостатньо ефективними. Використання цифрових технологій дозволяє автоматизувати процеси планування, обробки даних, розподілу завдань та комунікації між учасниками.

Бурхливий розвиток веб-технологій сприяє впровадженню інноваційних інструментів для автоматизації, що значно полегшує роботу організаторів. Сучасні веб-системи дозволяють інтегрувати різні сервіси, забезпечувати швидкий доступ до необхідної інформації та підвищувати рівень взаємодії між усіма задіяними сторонами.

На даний момент питання розробки та впровадження веб-системи для автоматизації планування масових заходів є актуальним, оскільки така система допоможе зменшити ймовірність помилок у плануванні, покращити контроль за виконанням завдань та забезпечити оперативний зворотний зв’язок.

Мета роботи: розробка веб-системи, яка автоматизує процеси планування та управління масовими заходами.

Об’єкт дослідження: система планування та організації масових заходів.

Предмет дослідження: методи та технології автоматизації планування заходів за допомогою веб-систем.

Методи дослідження: системний аналіз, методи веб-розробки, алгоритми управління подіями, база даних та API-інтеграція.

Робота присвячена створенню веб-системи, яка дозволить автоматизувати планування та проведення масових заходів, підвищуючи їхню ефективність та зручність для організаторів і учасників.

У першому розділі проведений аналіз предметної області та сформульовано основні задачі, які стоять перед системою. Розглянуто сучасний стан організації масових заходів, окреслено ключові проблеми, пов’язані з недостатньою автоматизацією та неефективністю існуючих підходів. Здійснений аналіз ринку подібних платформ в Україні та за кордоном, виявлені їхні переваги й недоліки. Також у розділі побудовані дерева цілей та задач дослідження, які наочно демонструють логіку побудови та реалізації веб-системи, та сформульовані функціональні, нефункціональні вимоги і ключові обмеження, що впливають на проєктування платформи.

У другому розділі здійснене проєктування архітектури веб-системи, яка реалізується за клієнт-серверною трирівневою моделлю. Визначені основні компоненти архітектури — рівень презентації, прикладний рівень та рівень даних, що дозволяє забезпечити гнучкість, масштабованість і модульність системи. У межах цього розділу розглянуто структуру взаємодії між компонентами, представлені діаграми потоків даних (DFD) для відображення логіки роботи окремих процесів, а також побудована концептуальна модель бази даних на основі документно-орієнтованої системи MongoDB.

У третьому розділі наведена програмна реалізація розробленої веб-системи. Проведений обґрунтований вибір технологічного стеку, до якого увійшли HTML, CSS, JavaScript на фронтенді та Node.js на серверній частині. Розглянуті основні алгоритми взаємодії користувачів із системою. Створена діаграма класів для відображення логіки взаємодії сутностней.

У четвертому розділі проведено техніко-економічне обґрунтування проєкту. Наведений прототип інтерфейсу, що відповідає принципам зручності та ергономічності, та представлено основні елементи користувацького інтерфейсу.

Розглянуто резюме проєкту, опис продукту. Проведений аналіз ринку збуту, сформована маркетингову стратегію просування веб-системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка і аналіз проблеми

Масові заходи – це важлива частина культурного, освітнього, наукового та бізнес-життя. Вони об'єднують людей для обміну знаннями, створення нових ідей, відпочинку та комунікації. Однак процес організації таких подій стає дедалі складнішим і ресурсозатратнішим. Від організаторів вимагається виконання багатьох завдань. Відсутність централізованої веб-системи ускладнює роботу з координації всіх аспектів, що знижує ефективність організації заходів і може негативно вплинути на досвід учасників.

Традиційні методи організації подій, які базуються на ручному плануванні, використанні таблиць для переліку персоналу та реквізиту або нескоординованій комунікації, часто призводять до неефективності, помилок і перевантаженості організаторів.

Також раніше інформація про заходи поширювалася переважно через друковані афіші, рекламні оголошення в газетах, а також за допомогою радіо- і телереклами. Деякі організатори розсилали запрошення поштою або розповсюджували друковані флаєри. Значну роль у популяризації заходів відігравало так зване «сарафанне радіо», коли люди дізнавалися про події від знайомих. Але таким способ інформація не завжди доходила до цільової аудиторії і на друкування та розповлюдження реклами могли витратити чимало коштів.

Продаж квитків також мав свої особливості — відвідувачі змушені були купувати їх у касах театрів, концертних залів або через уповноважених розповсюджувачів. Цей процес часто передбачав довгі черги, обмежені способи оплати (в основному готівку) та високий ризик підробки квитків.

За відсутності централізованої системи управління організація простору та розрахунок кількості учасників мав певні труднощі. Організатори визначали

необхідну кількість місць за приблизними оцінками або на основі попередніх заходів. У великих подіях реєстрація могла відбуватися вручну, що часто призводило до неточностей і ускладнювало процес підготовки. Відсутність чіткої аналітики створювала ситуації, коли захід був переповненим або, навпаки, організатори неефективно використовували ресурси через низьку відвідуваність. Щодо збору зворотного зв'язку, цей процес також мав певні труднощі: організатори пропонували заповнювати паперові анкети, які не завжди поверталися, а можливості швидкого аналізу відгуків майже не існувало.

Неможливість дізнатись скільки людей братиме участь у заході ускладнює ситуацію під час війни. Адже окрім забезпечення оптимального простору для самої події, потрібно впевнитись у тому, що всі перебуватимуть у безпеці під час повітряної тривоги. Для цього варто переконатись, що будівля, у якій проводитиметься захід, має укриття необхідного розміру або, якщо люди знаходитимуться на вулиці, що поблизу є метро чи інше загальнодоступне і достатньо велике укриття.

Однією з головних проблем традиційних методів організації подій є значні витрати часу та ресурсів на планування. Організаторам доводиться узгоджувати розклад заходу, розподіляти персонал, бронювати локації, забезпечувати інформаційну підтримку та враховувати безліч зовнішніх факторів.

Ще одним викликом є ризик втрати або дублювання інформації, що може призвести до накладання подій, нерівномірного розподілу ресурсів або інших організаційних збоїв. Особливо це критично при проведенні великих заходів, де залучено багато підрядників, волонтерів і учасників.

Однак завдяки розвитку цифрових технологій організація масових заходів може бути значно ефективнішою. Використання веб-платформ, мобільних додатків і соціальних мереж спрощує інформування потенційних відвідувачів, а алгоритми контекстної реклами допомагають досягати цільової аудиторії без значних витрат. Онлайн-продаж квитків дозволяє відвідувачам купувати їх у зручний для них час,

використовуючи різні способи оплати, зокрема банківські картки та електронні гаманці. Крім того, електронні квитки з QR-кодами не лише усувають необхідність друку, а й унеможливлюють їх підробку.

Щодо організації простору, автоматизовані системи реєстрації та аналітики дозволяють точніше прогнозувати кількість учасників і розподіляти місце відповідно до реальних потреб. Використання технологій розпізнавання облич, RFID-браслетів та мобільних додатків дає можливість контролювати потоки людей у режимі реального часу та запобігати перенавантаженню окремих зон. Дані, отримані з цифрових платформ, також дозволяють організаторам краще розуміти інтереси аудиторії та планувати майбутні заходи з урахуванням попереднього досвіду.

Зворотний зв’язок теж став значно простішим завдяки автоматизованим опитуванням через чат-боти, мобільні додатки та email-розсилки. Використання аналітичних інструментів та технологій Big Data дозволяє не лише швидко обробляти отримані відгуки, а й знаходити закономірності, які допомагають покращувати організаційні процеси.

Крім того, сучасні тенденції вимагають гнучкості та адаптивності в управлінні подіями. Зміни в умовах проведення заходу (наприклад, погодні умови, затримки виступів, технічні збої) потребують швидкого реагування, що складно реалізувати без відповідного програмного забезпечення.

Автоматизація цих процесів за допомогою веб-системи дозволяє значно підвищити ефективність організації заходів. Інтеграція цифрових інструментів дає можливість оперативно керувати розкладом, ресурсами, взаємодією з учасниками та аналітикою, що в результаті мінімізує людський фактор і помилки.

До того ж доцільно буде, для зручності, інтегрувати мапу укриттів, щоб люди знали куди йти під час небезпеки. Як приклад, український державний мобільний додаток і вебпортал «Дія», що надає громадянам доступ до цифрових документів, містить сервіс «Незламність» (рис. 1.1).

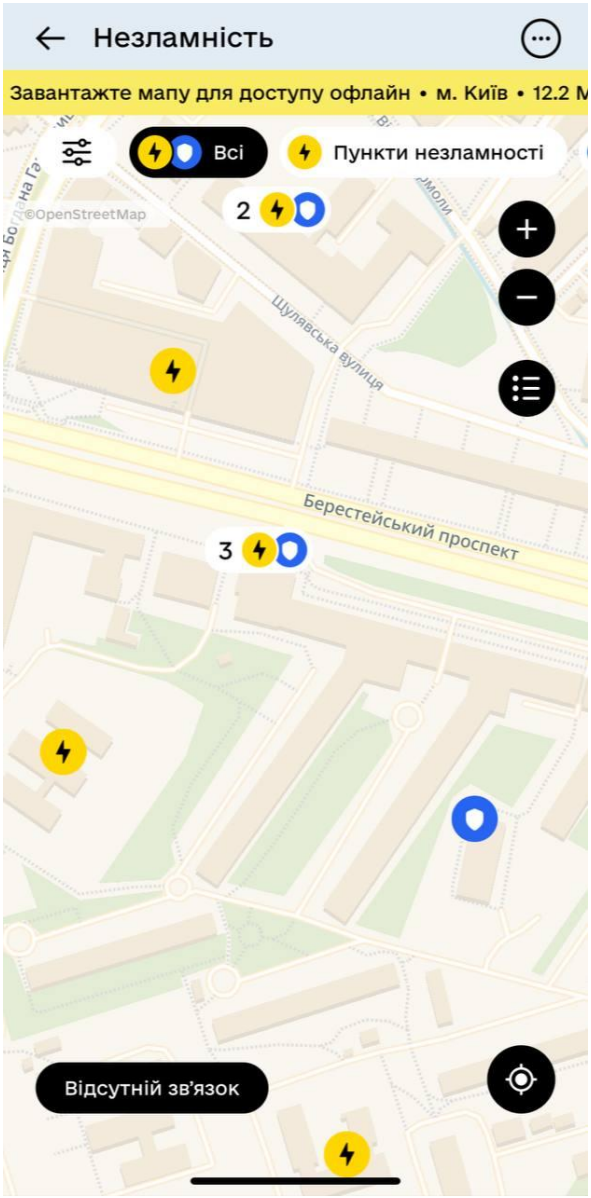

Рис. 1.1 Вигляд сервісу «Незламність» у мобільному додатку «Дія»

Який показує карту з відміченими на ній укриттями та пунктами незламності — місця, де незважаючи на відключення електроенергії передбачається наявність тепла, інтернету, води, мобільного зв'язку, живлення для мобільних пристроїв тощо.

Таким чином, розробка веб-системи для автоматизації планування масових заходів є актуальною задачею, яка спрямована на оптимізацію організаційних процесів, покращення комунікації між учасниками, підвищення загальної ефективності управління подіями, забезпечення швидкого аналізу подій і відгуків

на них та відповідає сучасним потребам суспільства, що все більше орієнтується на цифрові рішення.. Це дозволяє не лише мінімізувати помилки у плануванні, а й покращити загальний досвід відвідувачів, що в свою чергу підвищує успіх і популярність заходів.

1.2 Аналіз ринку та існуючих рішень

Ринок веб-систем для організації масових заходів активно розвивається у зв’язку зі зростанням популярності онлайн-платформ для купівлі квитків, автоматизації процесів реєстрації та управління подіями. В останні роки спостерігається значний перехід до цифрових технологій, що дозволяє організаторам оптимізувати процес планування, підвищувати ефективність комунікації з аудиторією та забезпечувати персоналізований підхід до кожного користувача.

За даними Pugpig, у їхньому "Звіті про стан ринку цифрових видавництв" за грудень 2022 року зазначено, що використання мобільних пристроїв зростає і надалі після різкого стрибка під час пандемії. «Згідно з даними eMarketer, користувачі мобільних пристроїв у США щодня витрачають більшу частину мобільного інтернету на застосунки, а не мобільні веб-браузери. І хоча з 2019 року час, проведений користувачами в мобільних браузерах, залишився незмінним, середній період активного користування застосунками зріс майже на годину» [3] (рис. 1.2).

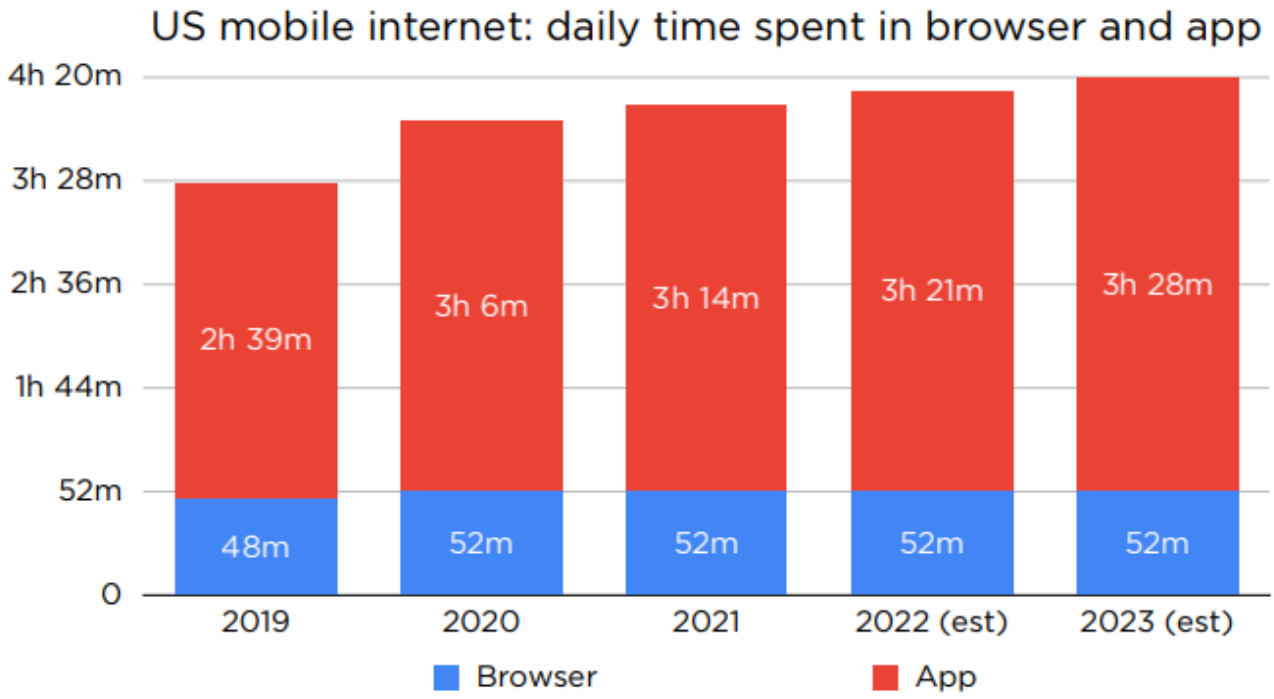

Рис. 1.2 Кількість годин витрачених на мобільні додатки та веб-браузери протягом дня у США (2019-2023 рр.)

У звіті про цифрові новини за 2022 рік Інститут вивчення журналістики Reuters в Оксфорді виявив, що в кількох європейських країнах вранці люди йшли за новинами саме в смартфони. У США телебачення та смартфони поділили перше місце, у Франції ж ТБ трохи випередило мобільні девайси як перше джерело ранкових новин (рис 1.6).

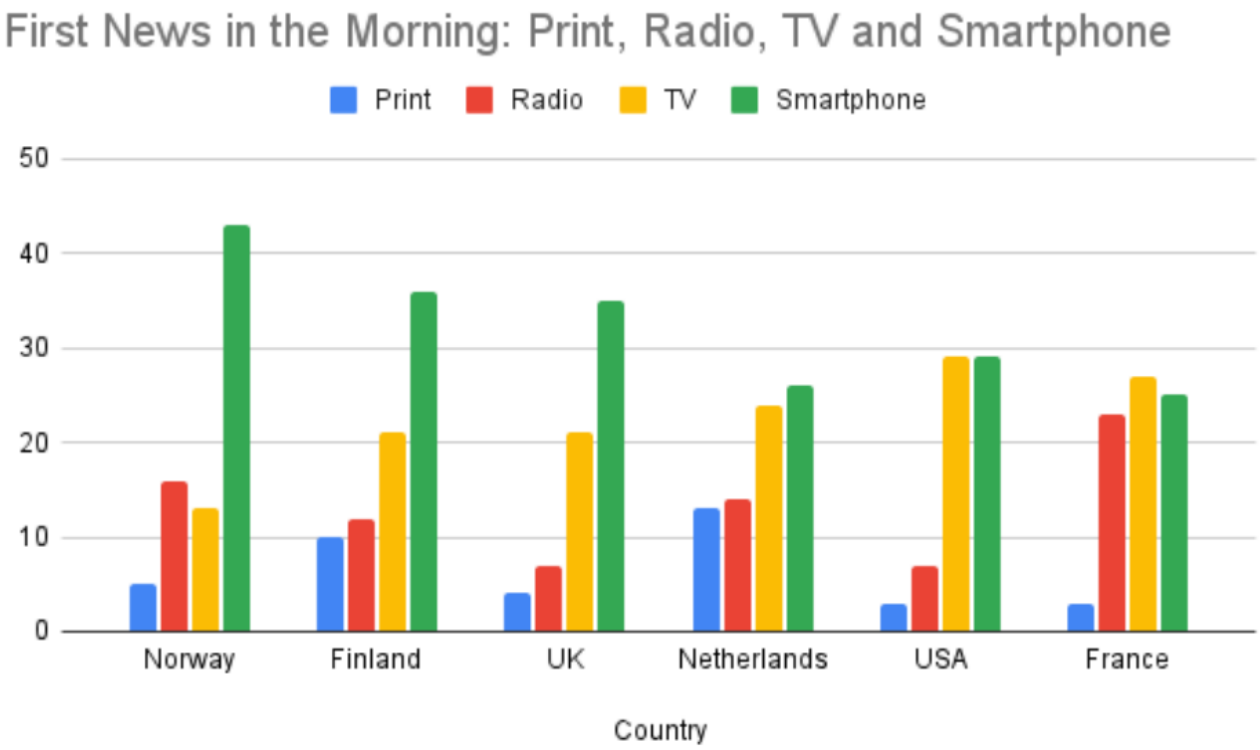

Рис. 1.3 Де дивляться новини вранці люди з різних країн

І це підтверджує той факт, що сучасній людині зручніше та лешге сприймати інформацію з ґаджетів, ніж з аналогічних джерел. Тому доцільно буде розробити веб-сайт планування подій, який забезпечить зручний доступ до інформації з будь-якого пристрою, зокрема смартфонів, що є основним джерелом отримання контенту для сучасної аудиторії. Крім того, веб-сайт дозволить централізовано керувати контентом, забезпечувати високу швидкість оновлення даних та покращити SEO-видимість у пошукових системах, що сприятиме залученню більшої кількості відвідувачів.

Керівники засобів масової інформації, яких опитали, сказали, що веб-сайти залучають найбільшу аудиторію, тоді як додатки — найбільш зацікавлену аудиторію. Утім, розсилки є ключовими в стратегіях розвитку, бо має великий текстовий обсяг і високу залученість (рис. 1.4 – 1.5).

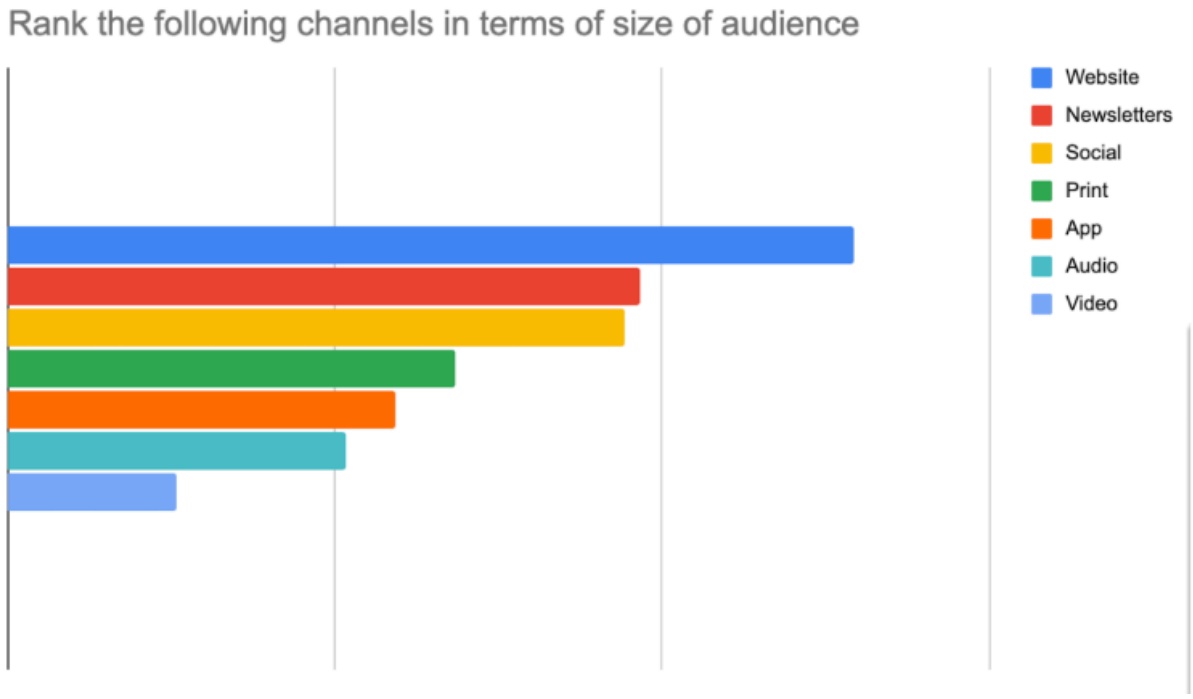

Рис. 1.4 Розмір залученої аудиторії за допомогою різних комунікаційних каналів

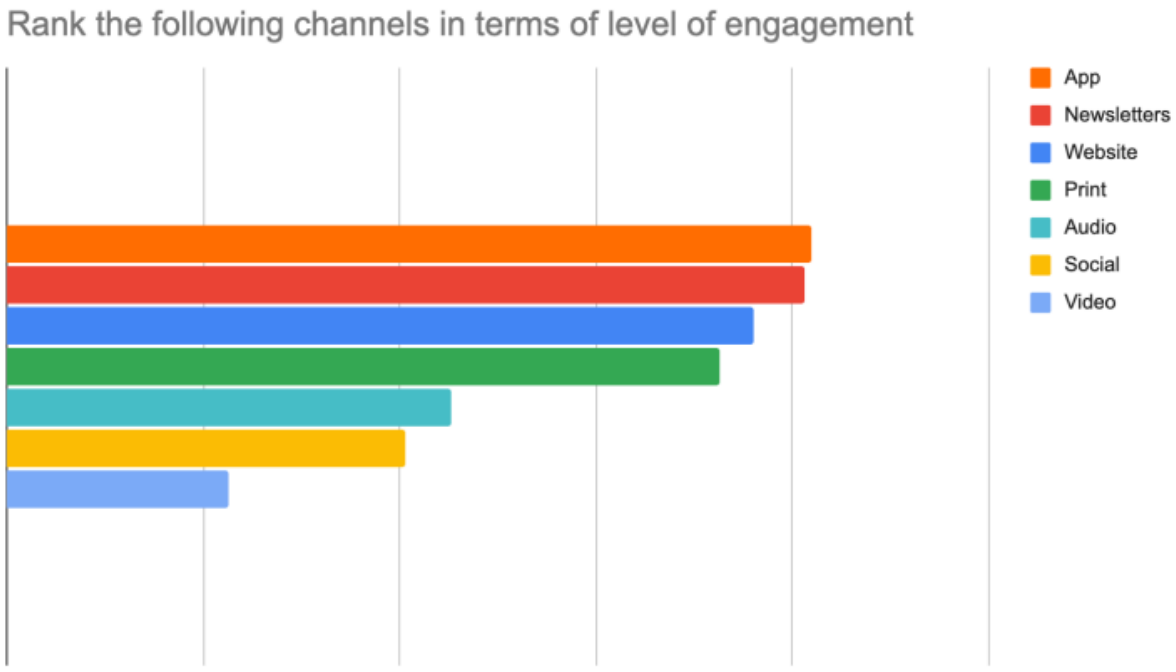

Рис. 1.5 Розмір зацікавленості за допомогою різних комунікаційних каналів

«Є оптимізм щодо можливостей, які пропонують цифрові платформи та формати, тому видавці зосереджуються на веб-сайтах, інформаційних бюлетенях і своїх програмах. Пандемія прискорила прийняття цифрових звичок серед споживачів, і вони бачать інфляцію та вищу вартість друкованої продукції як продовження цих тенденцій. З цієї причини вони намагаються постачати нові цифрові продукти, які використовують персоналізацію та популярні формати, як-от аудіо, для стимулювання підписок» [3] (рис. 1.7).

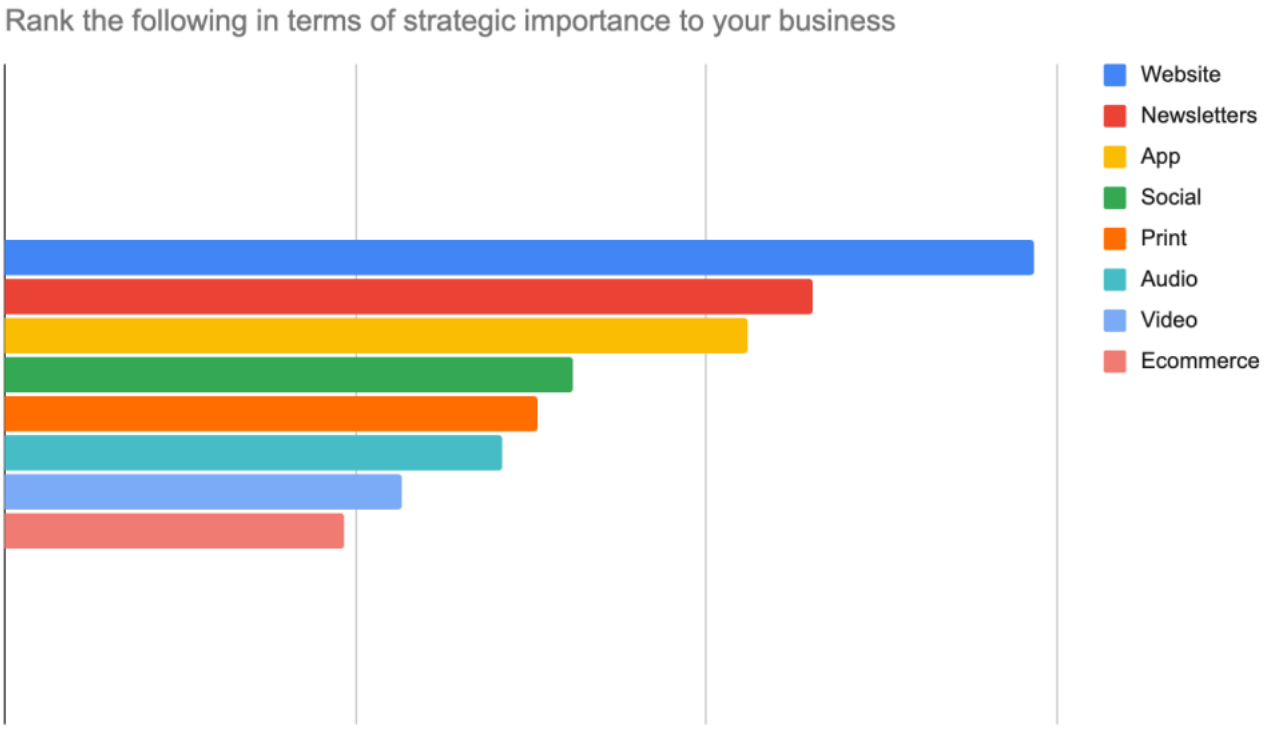

Рис. 1.6 На яких форматах найбільше зосереджуються медіа-лідери

Серед основних гравців на ринку веб-систем для організації масових заходів можна виділити такі популярні платформи, як Eventbrite, Ticketmaster, Meetup та Bizzabo. Кожна з цих платформ має свої переваги і недоліки.

Наприклад, Eventbrite пропонує зручну систему управління квитками та інтеграцію з соціальними мережами, однак може мати високі комісії як для

організаторів так і для покупців квитків. Незважаючи на те, що на безкоштовні заходи немає комісій, на платні заходи сервісні збори досить високі. 3.7% + 1.79$ за квиток та комісія за обробку замовлення 2.9% від загальної суми замовлення. Комісії сплачує покупець, якщо організатор не вирішить оплатити їх самостійно [4]. Тобто фактична вартість квитка для клієнта зростає. Це може впливати на рівень продажів та сприйняття сервісу покупцями, адже квитки можуть здаватися дорожчими, ніж очікувалося спочатку. Особливо для організаторів та клієнтів з України, адже Eventbrite дозволяє розміщувати події з будь-якої країни.

Ticketmaster є однією з найпопулярніших платформ у світі, але часто зазнає критики через непрозору політику цін та наявність додаткових зборів. Зокрема, практика динамічного ціноутворення, яка призводить до значного підвищення цін на квитки під час високого попиту.

Meetup, навпаки, орієнтується на спільноти та локальні події, що робить її менш ефективною для масштабних заходів. Ця платформа призначена для організації зустрічей людей із спільними інтересами, але не забезпечує необхідних інструментів для управління великими подіями.

Bizzabo — це сучасна платформа для організації подій, орієнтована переважно на корпоративний сегмент, зокрема бізнес-заходи, конференції, саміти та гібридні події. Вона забезпечує повний цикл управління заходом: від продажу квитків до глибокої аналітики та персоналізації досвіду учасників. Серед ключових переваг платформи — розширені аналітичні інструменти, підтримка гібридного формату, інтеграція з CRM-системами та маркетинговими сервісами. Проте, основними недоліками є висока вартість користування, що робить платформу недоступною для невеликих організаторів, складність у використанні через велику кількість функцій, а також орієнтація саме на ділові події, що робить її менш придатною для масових розважальних заходів.

Окрім великих міжнародних платформ, існує також низка локальних сервісів, які спеціалізуються на організації заходів у певних країнах або регіонах.

Наприклад, в Україні популярністю користуються сервіси Karabas, Concert.ua та Gastroli.ua. Вони пропонують інтеграцію з місцевими платіжними системами та адаптацію під потреби вітчизняного ринку, проте часто мають обмежений функціонал порівняно з глобальними конкурентами.

Concert.ua активно використовує сучасні маркетингові інструменти для залучення клієнтів. Вони пропонують зручний інтерфейс користувача, що дозволяє швидко знаходити та купувати квитки на різноманітні заходи, від концертів до театрів. Завдяки інтуїтивній навігації, можливості фільтрації подій за жанром, містом чи датою, платформа забезпечує комфортне користування навіть для недосвідчених відвідувачів. Сайт також інтегрований із соціальними мережами — Facebook, Instagram та Telegram, що сприяє органічному поширенню інформації про події, розіграші квитків та спеціальні пропозиції. Крім того, Concert.ua пропонує мобільний застосунок, що підвищує зручність користування. Однак, за даними дослідження, проведеного у 2020 році, компанія стикається з проблемою недостатньої лояльності клієнтів. Це може бути наслідком відсутності чітко вираженої унікальної ціннісної пропозиції, яка б вирізняла платформу серед конкурентів, а також обмеженого впровадження програм лояльності чи персоналізованих рекомендацій.

Karabas.com є одним із лідерів українського ринку електронного продажу квитків, який активно конкурує з Concert.ua. Платформа пропонує широкий спектр послуг: онлайн-продаж, система розміщення подій для організаторів, інтеграція з CRM, а також ексклюзивні партнерства з найвідомішими концертними майданчиками країни. Karabas.com також має широку офлайн-мережу кас, що дозволяє охоплювати користувачів, які віддають перевагу традиційним каналам покупки квитків. Крім цього, компанія активно інвестує в digital-рекламу та SEO-просування. Проте, як зазначає засновник компанії, ринок концертних квитків в Україні оцінюється приблизно в 150 мільйонів гривень щорічно, і це створює серйозний конкурентний тиск, особливо з боку нових цифрових гравців та міжнародних платформ. Незважаючи на свою масштабність, Karabas.com може

дещо програвати в гнучкості інтерфейсу та інноваційності окремим нішевим платформам.

Gastroli.ua — це ще один помітний гравець на українському ринку онлайн-продажу квитків, який позиціонує себе як культурна платформа, що підтримує українських митців і події. Сайт зосереджується не лише на комерційних концертах, а й на театральних постановках, фестивалях, лекціях, виставках та локальних культурних ініціативах. Gastroli.ua вирізняється візуально привабливим дизайном та акцентом на події, що мають соціальне чи культурне значення. Завдяки співпраці з багатьма незалежними організаторами подій, платформа часто пропонує ексклюзивні квитки на нестандартні заходи. Їхньою перевагою є гнучкий підхід до організаторів — просте розміщення подій, можливість персоналізованого оформлення сторінки, прозора система аналітики продажів. Проте, через вужчу аудиторію та обмежену кількість масштабних подій, Gastroli.ua не завжди має широкий асортимент або таку ж маркетингову потужність, як конкуренти. Саме тому, при створенні власного сайту для організації заходів варто взяти від Gastroli.ua приклад орієнтації на локальну культуру та оригінальний контент, поєднавши це з технічною зручністю та сучасним дизайном, що може створити унікальну нішу на ринку.

Основними проблемами, з якими стикаються існуючі системи, є складність налаштування подій, недостатня гнучкість у тарифах, відсутність зручної аналітики для організаторів, а також недостатньо ефективні механізми залучення аудиторії. Багато платформ також не пропонують інструментів для інтеграції з соціальними мережами або не мають можливості автоматичного прогнозування кількості відвідувачів на основі попередніх заходів.

Використання сучасних технологій, таких як машинне навчання та великі дані (Big Data), може значно покращити ці сервіси. Крім того, інтеграція з системами блокчейн може зробити квитки захищеними від підробок, а використання рекомендаційних алгоритмів дозволить краще таргетувати аудиторію.

Таким чином, ринок платформ для організації масових заходів активно розвивається, пропонуючи організаторам широкий вибір рішень. Водночас існують певні недоліки, які можна усунути завдяки впровадженню інноваційних технологій. Це створює значні можливості для розробки нових веб-систем, які будуть не лише автоматизувати процеси організації, а й робити їх більш зручними, гнучкими та ефективними для всіх учасників заходів.

1.3 Формування цілей та задач дослідження

Головною метою даної роботи є створення ефективної веб-системи, що автоматизує процес планування масових заходів, зменшує адміністративні витрати та підвищує зручність взаємодії між організаторами та учасниками подій. Актуальність такої системи обумовлена потребою в цифровізації організаційної діяльності, швидкому реагуванні на зміни та підвищенні інформованості потенційної аудиторії заходів.

Для досягнення основної мети дослідження — розробки веб-системи для автоматизації планування масових заходів — необхідно чітко визначити стратегічні напрямки, які дозволять втілити ідею на практиці. З цією метою сформоване дерево цілей (рис. 1.8), яке наочно демонструє ключові аспекти, що мають бути враховані під час реалізації проєкту. Такий підхід дозволяє структурувати цілі за рівнями узагальнення, виокремити пріоритетні напрями розробки та визначити, як кожна підціль сприяє досягненню головної мети. Візуалізація в такому вигляді також допомагає ефективніше комунікувати бачення проєкту з потенційними користувачами або інвесторами.

Рис. 1.7 Дерево цілей розробки веб-сайту для планування масових заходів

На основі сформованих цілей логічним наступним кроком є деталізація конкретних дій, необхідних для їх досягнення. Для цього використовується дерево задач (рис. 1.9), яке представляє собою ієрархічну структуру дій, поділених на логічні підзадачі. Такий підхід дозволяє послідовно розпланувати етапи розробки веб-системи, розподілити обов’язки між виконавцями, оцінити обсяги робіт і врахувати всі ключові аспекти реалізації — від аналітики та проєктування до програмування і тестування. Візуальне подання задач забезпечує цілісне уявлення про шлях досягнення мети та допомагає знизити ризики втрати важливих кроків.

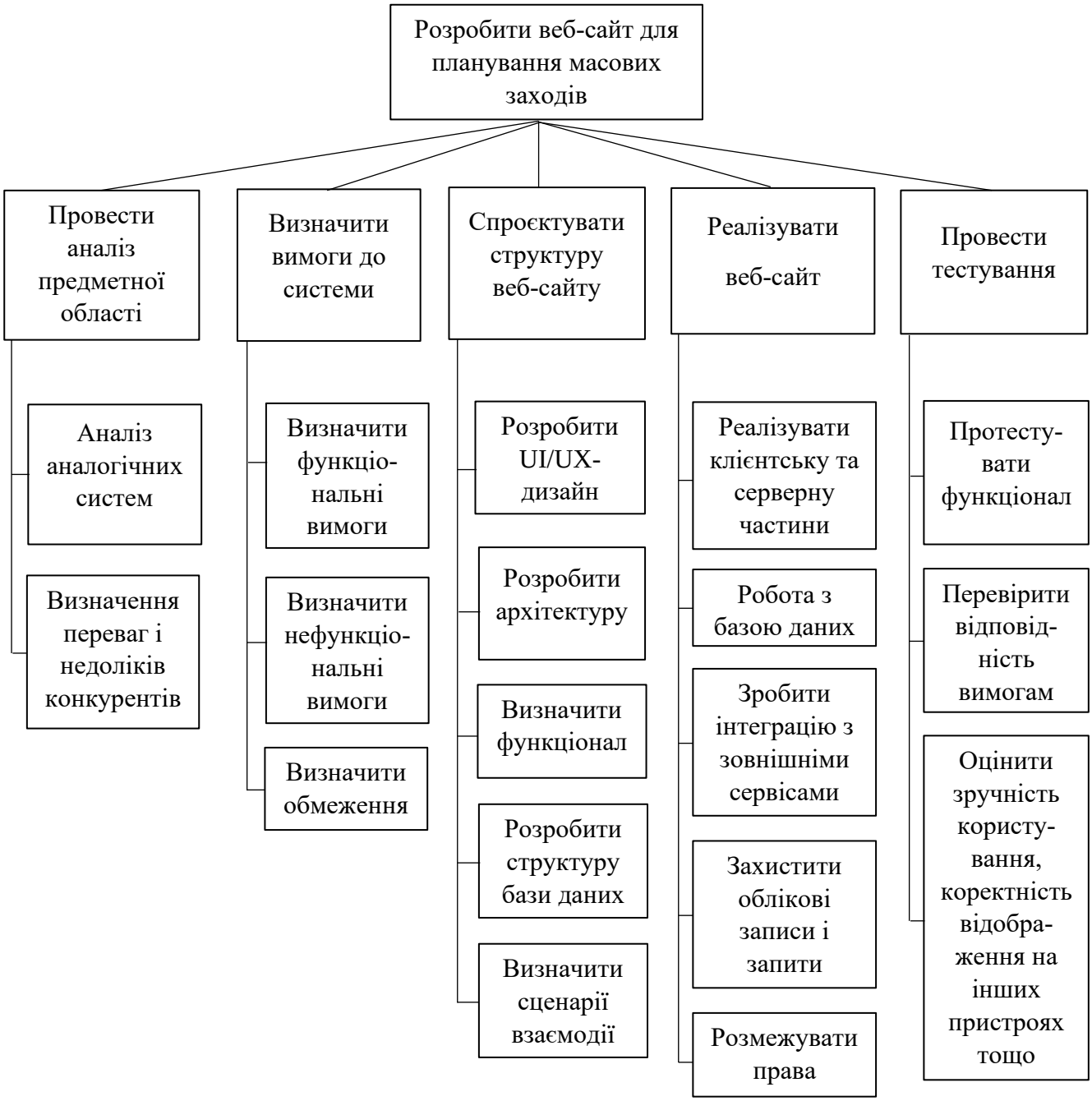

Рис. 1.8 Дерево задач розробки веб-сайту для планування масових заходів

1.4 Визначення вимог до системи

1.4.1 *Функціональні вимоги*

Успішна реалізація інформаційної системи неможлива без чіткого формулювання функціональних вимог. Саме вони описують, які дії має бути здатна виконувати система, які функції вона повинна забезпечувати для кінцевих користувачів, і як саме ці функції мають проявлятись у поведінці системи в реальних сценаріях.

Функціональні вимоги відображають очікування до логіки роботи системи, її взаємодії з користувачем, базою даних і зовнішніми ресурсами. Ці вимоги поділяються відповідно до ролей користувачів, які взаємодіють із платформою: незареєстровані користувачі (гості), зареєстровані користувачі (відвідувачі заходів), організатори заходів та адміністратори системи. Також врахована функціональність, яка забезпечує загальну працездатність системи, незалежно від ролі користувача.

Структурований перелік функціональних вимог до системи:

1. Реєстрація та авторизація користувачів:
    - Система повинна забезпечувати можливість реєстрації нових користувачів (як відвідувачів, так і організаторів заходів) через електронну пошту або через авторизацію за допомогою сторонніх сервісів (наприклад, Facebook або Google).
    - Має бути реалізований захищений механізм входу в обліковий запис, включаючи перевірку правильності введених даних і повідомлення про помилки.
    - Користувач після входу в обліковий запис повинен мати змогу вийти з системи або відновити пароль у разі його втрати.
2. Робота з подіями (афішами):

- Незареєстровані користувачі повинні мати можливість переглядати список усіх доступних заходів у вигляді афіші.
- Кожна подія повинна містити детальну інформацію: назва, опис, дата, час, місце проведення (із візуалізацією на мапі), вартість квитка, наявність місць.
- Повинен бути реалізований фільтр та пошук подій за назвою, датою, місцем проведення, категорією або ціною.
- Організатори повинні мати можливість створювати, редагувати та видаляти свої події, додаючи до них зображення, опис, кількість місць тощо.

3. Купівля квитків:
   - Зареєстровані користувачі мають змогу купувати квитки на події шляхом заповнення форми та підтвердження замовлення.
   - Після успішної покупки квиток повинен генеруватися у вигляді унікального QR-коду, який буде відображено на екрані та надіслано на пошту.
   - Система повинна забезпечувати збереження інформації про куплені квитки в обліковому записі користувача з можливістю їх перегляду.
   - Після досягнення встановленого ліміту (наприклад, максимум квитків або кінець продажу) система автоматично припиняє продаж квитків на відповідну подію.
4. Панель організатора:
   - Організатори повинні мати персональну панель керування подіями, де вони можуть:
     - створювати нові події;
     - редагувати інформацію про вже створені події;
     - переглядати статистику продажів по кожному заходу;
     - бачити список користувачів, які придбали квитки;
     - експортувати статистику у форматі CSV або PDF.

- Організатор повинен отримувати сповіщення про нові покупки квитків або зміни в статусі заходу.

5. Панель адміністратора:
   - Адміністратор системи повинен мати змогу керувати всіма подіями та користувачами, включаючи перегляд, блокування та видалення акаунтів.
   - Повинен бути доступний інтерфейс моніторингу загальної активності на платформі: кількість активних подій, кількість користувачів, кількість покупок.
   - Система повинна фіксувати ключові події в журналі активності для внутрішнього контролю.
6. Інтеграція з зовнішніми сервісами:
   - Передбачена можливість автоматичного додавання події до Google Calendar користувача після покупки квитка.
   - Реалізується інтеграція з Facebook для авторизації.
   - Система повинна бути здатною відправляти email-повідомлення про покупку, скасування або зміну події.
7. Аналітика та статистика:
   - Платформа має забезпечити аналітичну панель для організатора, де він бачить:
     - загальну кількість відвідувачів, які переглядали подію;
     - кількість куплених квитків;
     - джерела трафіку (наприклад, перехід із соцмереж, з пошуку);
     - динаміку продажів у часі (наприклад, графік по днях).
   - Дані мають бути подані у зручному візуальному форматі: таблиці, графіки, діаграми.
8. Адаптивність і доступність:
   - Система повинна мати адаптивний дизайн, що дозволяє комфортно використовувати її на мобільних пристроях, планшетах і десктопах.

- Інтерфейс має відповідати основним принципам доступності: контрастність, масштабованість, підтримка клавіатурної навігації, семантика HTML.

9. Захист даних і надійність:
   - Всі паролі користувачів повинні зберігатися у захешованому вигляді.
   - Взаємодія з API має бути захищена через токени (JWT).
   - Обмеження прав доступу повинно забезпечувати розмежування функцій для різних ролей.
   - Доступ до адміністративної частини повинен бути обмежений лише для авторизованих адміністраторів.

Функціональні вимоги формують ядро майбутньої системи і забезпечують її відповідність очікуванням користувачів і організаторів подій. Їх реалізація дозволить створити не лише зручну у користуванні, а й надійну, масштабовану та конкурентоспроможну платформу для організації масових заходів.

1.4.2 *Нефункціональні вимоги*

Окрім функціональних вимог, які визначають, що саме повинна робити система, важливе місце в процесі розробки займають нефункціональні вимоги. Нефункціональні вимоги — це вимоги до програмного забезпечення, які задають критерії для оцінки якості його роботи.Вони визначають якою система повинна бути.

Нефункціональні вимоги є ключовими для забезпечення надійної експлуатації веб-системи у реальних умовах, особливо з огляду на її публічний характер і потенційно велику кількість користувачів.

Опис основних нефункціональних вимог, що висуваються до розроблюваного програмного забезпечення:

1. Продуктивність і масштабованість:
    - Система повинна забезпечувати швидкий відгук інтерфейсу користувача (час відповіді сервера не більше 1 секунди для більшості запитів).
    - Платформа має бути здатною обробляти одночасні запити від сотень користувачів, не знижуючи продуктивності.
    - Структура системи повинна бути розроблена з урахуванням можливості масштабування, тобто підтримки зростання кількості користувачів, заходів та транзакцій без необхідності повної перебудови архітектури.

2. Безпека:

    - Усі дані, що передаються між клієнтом і сервером, повинні бути захищені протоколом HTTPS.
    - Паролі користувачів повинні зберігатися у хешованому вигляді з використанням стійкого алгоритму.
    - Для авторизації має використовуватись JWT з обмеженням терміну дії токена.
    - Система повинна реалізовувати розмежування прав доступу до функціональності відповідно до ролей (гість, користувач, організатор, адміністратор).
    - Критичні дії (створення події, купівля квитка, редагування профілю тощо) мають бути захищені від CSRF та XSS-атак.

3. Надійність та доступність:

    - Система повинна бути стійкою до збоїв: при відмові одного з компонентів (наприклад, бази даних) має бути можливість швидкого відновлення.
    - Передбачено резервне копіювання даних — періодично створюються копії бази подій та користувачів.

- Сайт повинен бути доступний 24/7, без тривалих простоїв, і з передбаченою обробкою помилок (відображення відповідного повідомлення користувачу).

4. Адаптивність і кросбраузерність:

- Веб-система повинна коректно відображатися на різних типах пристроїв (ПК, планшет, смартфон), забезпечуючи адаптивний дизайн.
- Платформа має підтримувати основні сучасні браузери.
- Інтерфейс має відповідати базовим принципам UI/UX-дизайну, бути інтуїтивно зрозумілим та зручним для навігації.

5. Модульність і підтримуваність:

- Кодова база повинна бути організована модульно, з логічним розділенням на компоненти (API, бази даних, маршрути, шаблони інтерфейсу тощо).
- Уся логіка має бути задокументована, що полегшить майбутні модифікації, усунення помилок та розвиток функціоналу.
- Передбачена можливість легкого додавання нових модулів у майбутньому — наприклад, модуля коментарів, чат-підтримки або інтеграції з новими сервісами.

6. Локалізація і доступність:

- Інтерфейс має підтримувати українську мову як основну, з можливістю подальшого додавання інших мов (наприклад, англійської).

7. Час навчання користувача:

- Інтерфейс платформи повинен бути настільки зрозумілим, щоб звичайний користувач міг почати використовувати її без потреби у додатковому навчанні.

Нефункціональні вимоги не менш важливі, ніж функціональні, оскільки вони визначають якість використання системи. Їх дотримання забезпечить зручну, захищену та надійну платформу, яка буде адаптована до потреб як кінцевих користувачів, так і адміністраторів. Залучення принципів масштабованості, безпеки, адаптивності та доступності дозволить системі залишатися актуальною та стійкою до навантажень навіть при її подальшому розширенні.

1.4.3 *Обмеження*

У процесі розробки та впровадження будь-якого програмного забезпечення важливо враховувати не лише функціональні та нефункціональні вимоги, а й наявні обмеження, які можуть впливати на структуру, реалізацію, вибір технологій, строки виконання та кінцеві результати. Для розробки веб-системи автоматизації планування масових заходів ці обмеження умовно можна розділити на технічні, часові та організаційні, хоча на практиці вони часто взаємопов’язані.

До технічних обмежень передусім належить вибір інструментів, наявністю підтримки відповідних технологій у середовищі розгортання, а також ресурсними обмеженнями (наприклад, відсутністю платного хостингу чи обмеженою підтримкою серверної інфраструктури). Система повинна бути побудована на безкоштовних або відкритих технологіях, таких як Node.js, Express, MongoDB чи аналогічних. Також через брак повноцінного середовища тестування система реалізується переважно на локальному сервері або безкоштовному хостингу, що не завжди дозволяє симулювати високі навантаження або перевірити безпеку в справжніх умовах. Веб-сайт не повинен вимагати надмірного обсягу ресурсів при завантаженні (висока вага зображень, складна анімація тощо), оскільки це може негативно впливати на швидкість роботи в умовах повільного інтернету чи на мобільних пристроях.

Часові обмеження є одними з найбільш критичних, оскільки розробка проєкту ведеться в рамках навчального курсу, а отже — має суворо визначені дедлайни. У дипломній роботі, як правило, на реалізацію всієї системи (від аналітики до презентації) відводиться кілька місяців, під час якого проходить навчальний семестр, з яких безпосередньо на програмування може залишитися 3–5 тижнів. У такому обмеженому часовому проміжку неможливо реалізувати повний спектр функцій, який міг би бути бажаним у реальному комерційному продукті. Через це частина ідей або інтеграцій може бути спрощена або взагалі винесена до перспектив подальшого розвитку. Наприклад, система може не включати складну платіжну інфраструктуру або багаторівневу перевірку безпеки. Також часові рамки обмежують кількість тестових сценаріїв і тривалість їх перевірки, що у свою чергу впливає на повноту тестування.

Не менш важливими є організаційні обмеження, які визначаються умовами навчального процесу, очікуваннями наукового керівника, методичними вказівками до написання дипломної роботи, а також обсягом доступної допомоги. Наприклад, студент може працювати над проєктом самостійно, без участі фахових дизайнерів чи тестувальників, що автоматично ускладнює досягнення професійного рівня оформлення інтерфейсу або всебічного тестування. Організаційні рамки можуть також включати вимоги щодо використання конкретної термінології, структури документації, способу подання звітності (наприклад, надання пояснювальної записки у друкованому форматі, додавання архіву з кодом у додатках тощо). Обмеження можуть стосуватися і етичної сторони: використання лише відкритих API, дотримання авторських прав при завантаженні зображень, відповідність стандартам цифрової доступності тощо.

Усі зазначені обмеження, попри свою обмежувальну природу, виконують важливу роль — вони задають реалістичні рамки проєкту, в яких необхідно приймати зважені рішення, встановлювати пріоритети та вчитися адаптуватися до обставин. У дипломній роботі ключовим показником є не кількість реалізованих функцій, а логічна завершеність, функціональна цілісність і відповідність

початковим цілям, сформованим на етапі аналізу. Саме тому критично важливо враховувати обмеження ще на етапі проєктування — це допомагає уникнути перевантаження системи надлишковою логікою, не виходити за рамки дедлайнів і зберегти послідовність в архітектурі рішення.

Постановка задачі

У результаті проведеного аналізу предметної області виявили проблему, яка полягає в недостатній автоматизації процесів організації масових заходів, зокрема щодо формування аналітичної звітності, відсутність або обмеженість інтеграції з зовнішніми сервісвми, яка б дозволила автоматичну реєстрацію. Більшість існуючих рішень або надто вузькоспеціалізовані, або не охоплюють усіх потреб як організаторів, так і відвідувачів подій, наприклад, відсутність можливості автоматично додати подію у Google Calendar, тому учасники вимушенні вручну заповнювати потрібну дату в календарі. Це ускладнює планування та знижує ефективність проведення заходів.

Зі сторони організатора події виникає необхідність у створенні інструменту, який дозволяє швидко формувати афішу, керувати даними про подію, бачити аналітику продажів, джерела трафіку та кількість учасників. Зі сторони користувача — необхідна зручна платформа для ознайомлення з афішами, перегляду детальної інформації про події, здійснення онлайн-купівлі квитків та збереження їх у цифровому вигляді.

Таким чином, основна задача — створення веб-системи, яка забезпечить двосторонню взаємодію між організаторами заходів та аудиторією, з підтримкою автоматизованих функцій керування подіями та статистичної аналітики.

Вхідні дані:

- Інформація про подію (назва, опис, час, місце, афіша, ціни).
- Дані користувача (ПІБ, email, обрана подія, спосіб оплати).

- Реєстраційні та авторизаційні дані (логін, пароль, роль користувача).
- Джерело трафіку, з якого користувач прийшов на сайт.
- Дії користувача (перегляди, купівлі, реакції).

Вихідні дані:

- Згенеровані квитки з унікальним QR-кодом.
- Інформація в особистому кабінеті організатора (кількість куплених квитків, джерела трафіку, географія користувачів).
- Афіші з фільтрами та пошуком для користувача.
- Повідомлення/сповіщення про нові покупки або зміни статусу заходу.
- Аналітичні графіки (популярність подій, динаміка продажів, час активності користувачів).

2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

2.1 Загальний опис архітектури системи

Розроблювана веб-система автоматизації планування масових заходів реалізується за трирівневою клієнт-серверною архітектурою (рис. 2.1). Цей підхід дозволяє логічно розділити систему на три основні рівні. Кожен рівень відповідає за певну функцію програми. Перший рівень — це рівень презентації або інтерфейсу користувача. Другий рівень — прикладний, на якому обробляються дані. І третій рівень — це рівень даних, де зберігаються дані, пов’язані з програмою.

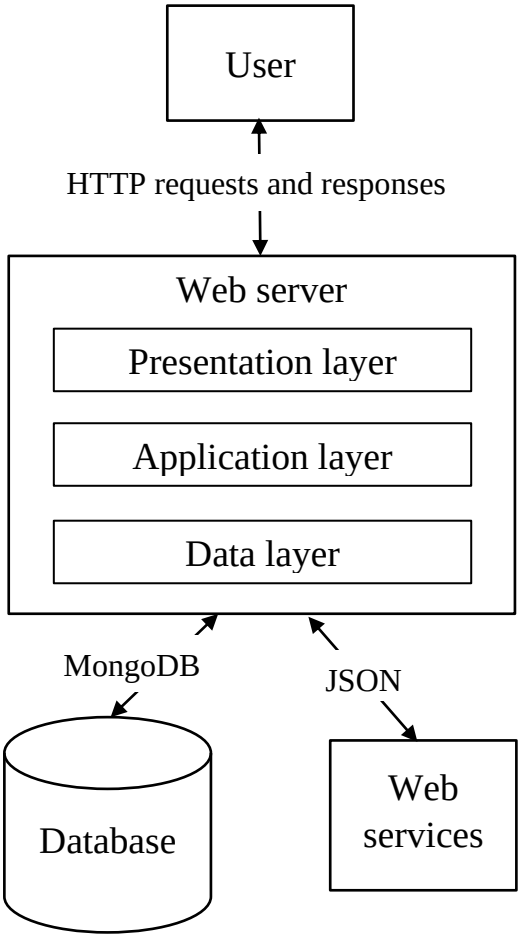

Рис. 2.1 Складові трирівневої клієнт-серверної архітектури

На рівні презентації користувач взаємодіє з програмою через графічний інтерфейс. Прикладний рівень відповідає за обробку інформації, отриманої від презентаційного рівня, і виконання бізнес-правил програми. Нарешті, рівень даних відповідає за зберігання даних програми.

Трирівнева архітектура — це архітектура клієнт-сервер, у якій функціональна логіка процесу, доступ до даних, зберігання комп'ютерних даних та інтерфейс користувача розробляються та підтримуються як незалежні модулі на окремих платформах. Ці модулі можна оновлювати незалежно один від одного, що робить обслуговування програми простішим і ефективнішим.

Така архітектура є масштабованою, зручною для підтримки та подальшого розвитку, і є однією з найпоширеніших моделей побудови веб-додатків. Розділення рівнів дозволяє розробляти та підтримувати кожен рівень незалежно, що робить розробку та обслуговування додатків простішими та ефективнішими.

Основні компоненти архітектури:

- Фронтенд — реалізований за допомогою HTML, CSS та JavaScript. Це інтерфейс, з яким взаємодіє кінцевий користувач (відвідувач, організатор або адміністратор). Забезпечує динамічне відображення афіш, форм купівлі квитків, статистики, панелей керування подіями тощо.
- Бекенд — реалізується з використанням Node.js та фреймворку Express. Відповідає за обробку запитів, автентифікацію, бізнес-логіку (облік квитків, сповіщення, аналітику), управління подіями та користувачами.
- БД — MongoDB. Зберігає всі дані про користувачів, події, куплені квитки, ролі, журнали подій тощо. Забезпечує швидкий доступ і масштабованість.
- API — REST API використовується для комунікації між фронтендом і бекендом. API відповідає за обмін даними у форматі JSON між

інтерфейсом та сервером, а також забезпечує інтеграцію з зовнішніми сервісами.

Інтеграції:

- Google Calendar API – для автоматичного додавання заходу в календар.
- Email-сервіси – для відправки підтверджень купівлі та сповіщень.
- Facebook OAuth / Google Sign-in – для зручної авторизації.

2.2 DFD-діаграма

Діаграми потоків даних (Data Flow Diagrams) показують, звідки надходять дані, як дані обробляються всередині системи, куди передаються результати обробки і чи використовуються вихідні результати іншою діяльністю або зовнішнім суб'єктом. DFD-діаграми є одним із ключових інструментів структурного аналізу системи, що широко використовуються в проєктуванні інформаційних систем.

У рамках даної роботи побудова DFD-діаграм дає змогу чітко визначити основні процеси, що реалізуються у веб-системі автоматизації планування масових заходів, взаємодію між користувачами (відвідувачами, організаторами, адміністраторами), зовнішніми сервісами (такими як платіжні системи, Google Calendar, email-сервіси) та базою даних. Така модель допомагає зрозуміти функціональну структуру системи, ідентифікувати точки взаємодії з ключовими сутностями, а також полегшує подальше проєктування логіки обробки даних і розробку програмних модулів.

Діаграми потоків даних можуть складатися з декількох рівнів абстракції. Діаграма найвищого(нульового) рівня - це контекстна діаграма, яка представляє всю систему.Цей рівень моделі дозволяє сформувати загальне уявлення про

основні зовнішні джерела та приймачі інформації в межах системи. На зображені нижче представлено нульовий рівень DFD веб-системи:

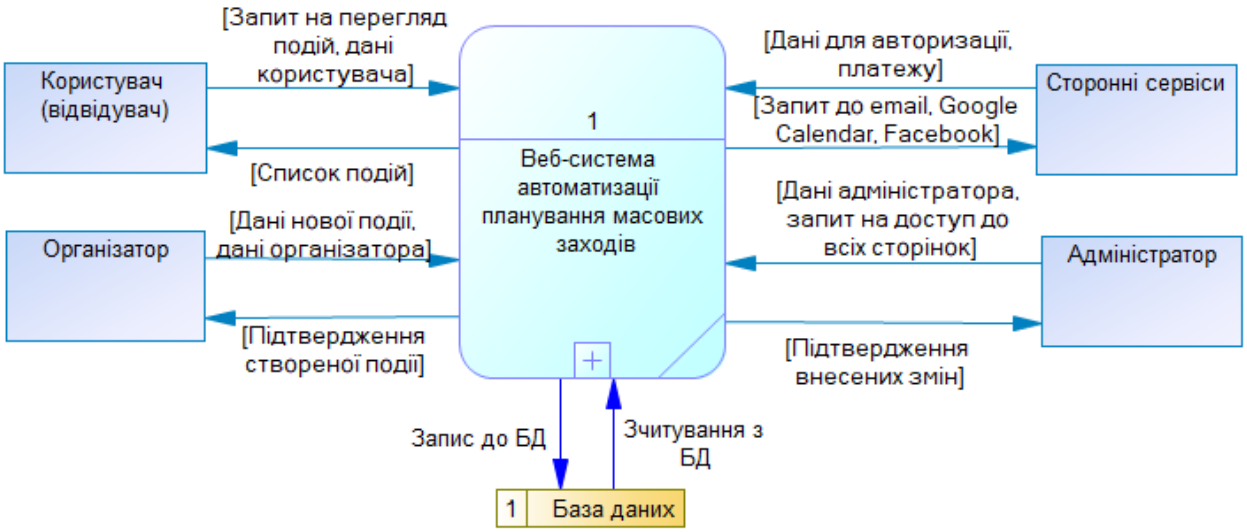

Рис. 2.2 Нульовий рівень DFD-діаграми

DFD-діаграма першого рівня деталізує основні функціональні компоненти системи та їх взаємозв’язки. Вона дозволяє глибше проаналізувати, з яких підпроцесів складається загальний процес, які саме потоки даних проходять між ними, які дані зберігаються в базі даних і як відбувається обмін інформацією з зовнішніми сутностями.

Побудова діаграми першого рівня забезпечує кращу структурованість системи та створює основу для розробки другого рівня деталізації або безпосередньої реалізації архітектури.

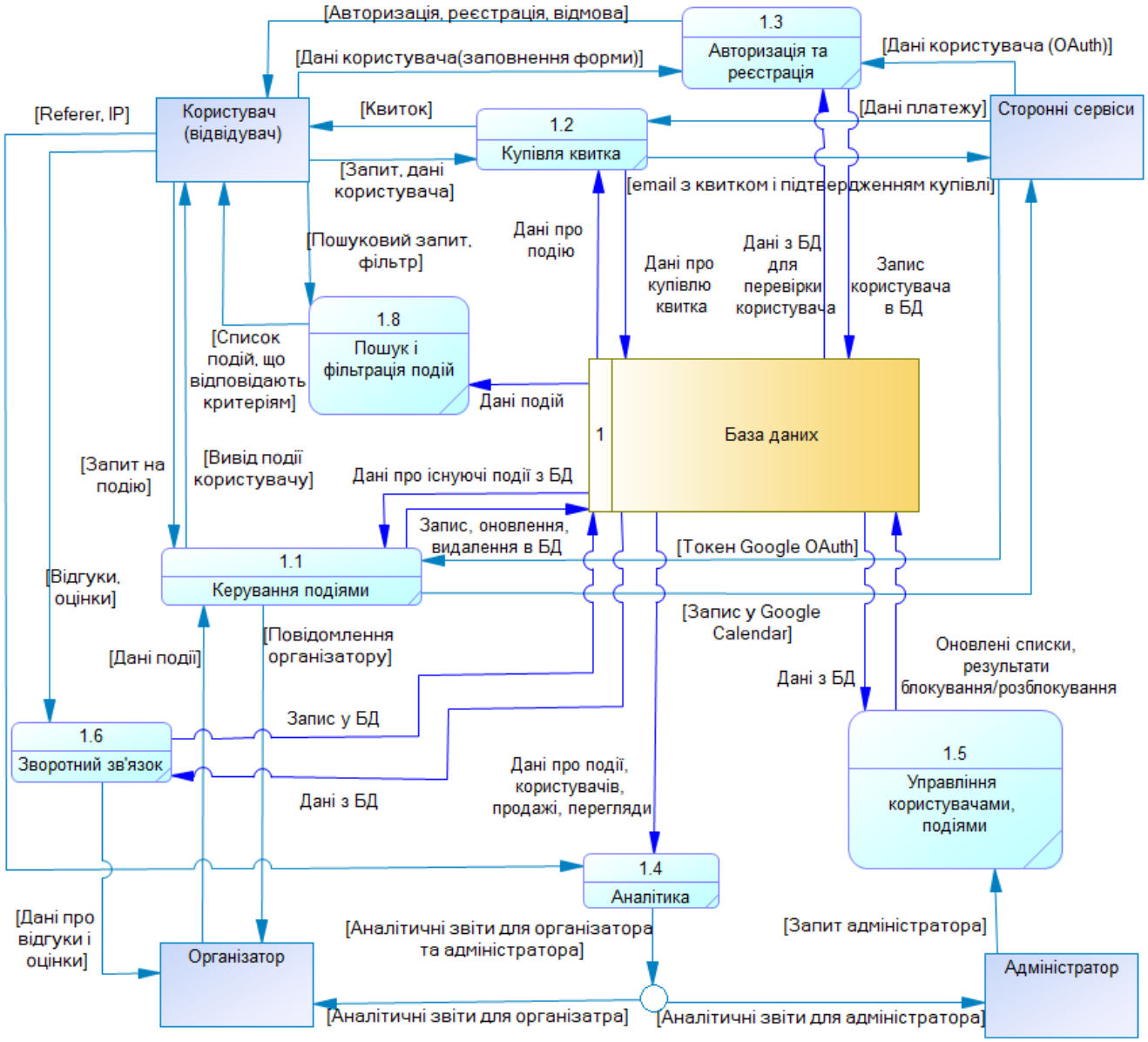

Рис. 2.3 Перший рівень DFD-діаграми

DFD-діаграми другого рівня дозволяють розглянути внутрішню структуру кожного процесу, з яких підпроцесів він складається, які дані надходять до них та які результати вони генерують. Такий рівень деталізації є необхідним для повного розуміння функціональної моделі системи, а також для подальшого проєктування програмної реалізації.

Кожна з наступних таблиць представляє логічну структуру одного з основних процесів, описаних на першому рівні DFD-діаграми. У таблицях наведено

підпроцеси, які входять до відповідного процесу, вхідні та вихідні дані для кожного з них, а також джерело або напрямок цих даних (від якої сутності чи до якої системи вони передаються). Таке подання дає змогу описати функціонування кожного процесу.

У процесі керування подіями організатор взаємодіє із системою через спеціальну панель, де може створювати, редагувати, видаляти події або публікувати афіші. На етапі створення події вводяться такі дані, як назва, дата, час, опис, місце проведення, зображення тощо. Ці дані надходять на сервер, де створюється новий запис у базі даних. У випадку редагування чи видалення передається ідентифікатор події (ID), щоб система могла знайти відповідний запис. Для публікації афіші змінюється статус події на “опубліковано”. Для інтеграції з Google Calendar потрібна авторизація організатора через OAuth2, після чого викликається відповідне API і подія записується у календар користувача. Перегляд події реалізовано як запит до БД за ID події.

Таблиця 2.1

Другий рівень DFD-діаграми для процесу керування подіями

Процес	Підпроцеси	Вхідні дані	Вихідні дані
1. Керування подіями	1.1. Створення події	Дані події (від організатора)	Новий запис у БД
	1.2. Редагування події	ID події, оновлені дані події (від організатора)	Оновлений запис у БД
	1.3. Видалення події	ID події (від організатора)	Видалення запису з БД
	1.4. Публікація афіші	ID події, статус (від організатора)	Статус публікації в БД
	1.5. Збереження в Google Calendar	Дані події (від організатора)	Запис у Google Calendar
	1.6. Перегляд події	ID події (від організатора/відвідувача)	Інформація про подію (з БД)

Користувач взаємодіє з процесом купівлі квитка у кілька етапів. Спершу він переглядає список доступних подій, які приходять із бази даних. Після вибору події — переходить на окрему сторінку з деталями. Потім обирає кількість квитків, які хоче купити. Дані про подію, замовлення та користувача зберігаються і надсилаються в платіжну систему. Після підтвердження оплати система генерує квиток із QR-кодом і надсилає його на email користувача, а також зберігає відповідні дані в БД. Таким чином, на вхід надходять дані користувача, оплати, ID події; на вихід — квиток у цифровому вигляді, запис у БД і підтвердження на email.

Таблиця 2.2

Другий рівень DFD-діаграми

Процес	Підпроцеси	Вхідні дані	Вихідні дані
2. Купівля квитків	2.1. Вибір події	Список подій (від БД)	Обрана подія (до фронтенду)
	2.2. Перехід на сторінку події	ID події (від користувача)	Сторінка з деталями (від сервера)
	2.3. Вибір кількості квитків	Дані про місця (з БД)	Замовлення (до сервера)
	2.4. Оплата	Дані платежу, запит на оплату (до сторонніх сервісів)	Підтвердження транзакції(від сторонніх сервісів)
	2.5. Генерація QR	ID квитка (в системі)	QR-код (до користувача)
	2.6. Надсилання квитка	QR-код, email (від системи)	Лист із квитком (до стороннього сервісу, щоб надіслати на пошту користувачу)

Авторизація охоплює реєстрацію нового користувача, вхід у систему, вихід із неї та перевірку доступу до певних сторінок. Під час реєстрації користувач вводить свої дані (ПІБ, email, пароль), які зберігаються у базі даних. Також можливий варіант входу через сторонні сервіси (наприклад, Google, Facebook), у

цьому разі використовується протокол OAuth, який дозволяє безпечно отримати доступ до облікового запису без передачі пароля. А у разі звичайної авторизації використовується токен JWT (JSON Web Token), що підтверджує особу користувача та передається у кожному запиті для перевірки прав доступу. Під час авторизації на сервер надходять логін і пароль, а система повертає токен або повідомлення про помилку. При виході токен або сесія знищується. У разі доступу до захищених сторінок система перевіряє права користувача за токеном.

Таблиця 2.3

Другий рівень DFD-діаграми

Процес	Підпроцеси	Вхідні дані	Вихідні дані
3. Авторизація	3.1. Реєстрація	Дані користувача (від форми або стороннього сервісу)	Новий запис у БД (користувачі)
	3.2. Вхід	Логін, пароль (від користувача)	JWT/сесія (до клієнта)
	3.3. Вихід	JWT (від клієнта)	Завершення сесії (на сервері)
	3.4. Перевірка доступу	Токен доступу (від користувача)	Дозвіл/відмова (від сервера)

Процес аналітики призначений для збору, обробки та подання статистичних даних щодо подій, відвідувань, продажів та поведінки користувачів. Інформація для аналітики надходить із кількох джерел: бази даних (наприклад, кількість проданих квитків, переглядів події), а також із запитів користувачів (зокрема, HTTP-заголовок Referer, що дозволяє визначити джерело трафіку, та IP-адреса — для встановлення географічного розташування). Усі ці дані використовуються для побудови персоналізованої аналітичної панелі для організатора, де він може бачити популярність своїх подій, ефективність джерел трафіку та географію користувачів. Адміністратор системи має доступ до зведеної статистики по всій платформі,

включно з кількістю зареєстрованих користувачів, організаторів, активних заходів тощо. Вихідними даними цього процесу є графіки, таблиці та числові звіти, які формуються в реальному часі на основі актуальних даних.

Таблиця 2.4

Другий рівень DFD-діаграми

Процес	Підпроцеси	Вхідні дані	Вихідні дані
4. Аналітика	4.1. Збір переглядів подій	Ідентифікатор події (від користувача)	Кількість переглядів (до БД/аналітики)
	4.2. Джерела трафіку, географія відвідувачів	Referer-заголовок HTTP, IP-адреса користувача (від браузера користувача)	Визначене джерело, регіон/місто (до організатора/ адміністратора)
	4.4. Статистика організатора	Дані про події, квитки (з БД)	Аналітичний звіт (до організатора)
	4.5. Статистика адміністратора	Агреговані дані з БД	Зведений звіт (до адміністратора)

Адміністратор має доступ до списку всіх зареєстрованих користувачів, який він може переглядати, фільтрувати за ролями (відвідувач, організатор, адміністратор) та здійснювати певні адміністративні дії. Наприклад, він може заблокувати користувача або подію, якщо є порушення правил, та розблокувати назад. Для цього адміністратор надсилає команду на сервер, яка оновлює статус у відповідному записі в базі даних. Також адміністратор має змогу переглядати лог-файли системи, які містять інформацію про дії користувачів (наприклад, вхід, редагування події, покупку квитка), помилки або інші важливі події. Це дозволяє швидко виявляти збої або зловживання та оперативно реагувати на них.

Таблиця 2.5

Другий рівень DFD-діаграми

Процес	Підпроцеси	Вхідні дані	Вихідні дані
5. Управління користувачами, подіями	5.1. Перегляд списку користувачів	Запит адміністратора	Список користувачів (з БД)
	5.2. Блокування і розблокування користувачів/подій	ID користувача або події(з БД)	Зміна статусу в БД

Після завершення події користувачі мають змогу залишати відгуки та оцінювати захід. Цей процес починається з надсилання користувачу автоматичного запиту на зворотний зв’язок (через email або інтерфейс). Користувач може залишити текстовий коментар, вибрати оцінку за шкалою (від 1 до 5 зірок) або відповісти на коротке опитування. Усі зібрані відповіді потрапляють у базу даних і доступні організатору, який може переглядати їх у власному кабінеті. Окрім індивідуального перегляду, система також може формувати зведені звіти — середній бал, відсоток позитивних/негативних оцінок, найбільш поширені ключові слова тощо. Це дозволяє організаторам аналізувати якість заходів і враховувати думку відвідувачів для подальших подій.

Таблиця 2.6

Другий рівень DFD-діаграми

Процес	Підпроцеси	Вхідні дані	Вихідні дані
6. Зворотний зв’язок	6.1. Надсилання відгуків	Текст відгуку (від користувача)	Запис у БД
	6.2. Оцінювання заходу	Оцінка (від користувача)	Збережена оцінка в БД
	6.3. Перегляд відгуків	Оцінки і відгуки	Список відгуків(до організатора)

2.3 Алгоритм роботи API

API — це програмний інтерфейс, який дозволяє різним програмним компонентам взаємодіяти між собою. На практиці API — це набір компонентів, за допомогою яких одна комп'ютерна програма або сайт може використовувати іншу за допомогою наборів визначень та протоколів. У контексті веб-розробки API є посередником між клієнтською частиною (frontend) та серверною логікою (backend), а також між веб-додатком і сторонніми сервісами, такими як платіжні системи, системи автентифікації тощо.

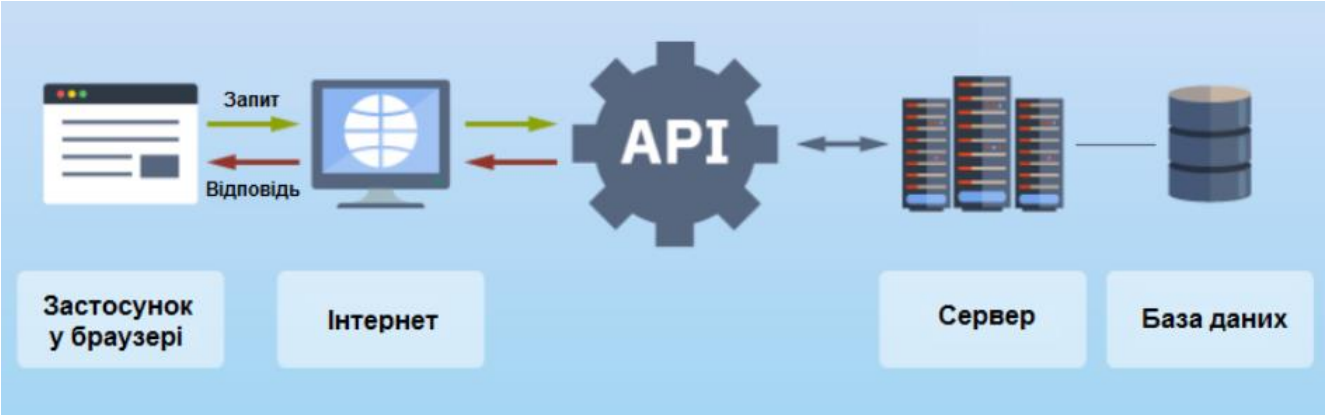

Рис. 2.4 Схема роботи API

За типом доступу API поділяються на:

- Публічні API — створені для певної програми, доступні для всіх, наприклад, Google з API Google Maps.
- Приватні чи внутрішні API — приховані від всього світу.
- Партнерські API — розробники компаній пишуть для партнерів та діляться лише з ними.
- Складні API — складаються з кількох API, щоб виконувати багато послідовних дій.

На сьогодні найбільш поширеними типами архітектури API є REST, SOAP та RPC. Кожен з яких має унікальні характеристики та використовується для різних цілей. Є й інші типи API, але ці три одні з найпопулярніших.

REST (Representational State Transfer) — це архітектурний стиль, заснований на використанні стандартних HTTP-методів для доступу до ресурсів, які представлені у вигляді URL-адрес. Таких методів всього 4: отримання (GET), створення (POST), оновлення (PUT) та видалення (DELETE). Він широко застосовується у веб-розробці завдяки простоті реалізації, хорошій підтримці з боку браузерів та високій гнучкості. REST API зазвичай використовує формат обміну даними JSON, що дозволяє легко передавати структуровану інформацію між клієнтом і сервером. Його використовують для швидкого обміну простими параметрами, у тому числі тими, з яких складаються бази даних. Такий тип API ідеально підходить для проєктів, де важливо забезпечити масштабованість, розділення логіки й чіткий доступ до ресурсів. У межах розробки веб-системи для організації масових заходів REST API застосовується для реалізації основної бізнес-логіки: керування подіями, купівлі квитків, авторизації користувачів та отримання аналітики.

SOAP (Simple Object Access Protocol) — це більш формалізований протокол, що базується на XML та призначений для обміну повідомленнями між сервісами. SOAP часто використовується для внутрішніх або партнерських API. Він забезпечує високий рівень безпеки, контроль над структурою повідомлень та підтримку транзакцій, що робить його доцільним для фінансових систем або державних реєстрів, де важлива сувора стандартизація та сумісність. Проте, у порівнянні з REST, SOAP має вищу складність реалізації, важчу вагу повідомлень і не настільки гнучкий у веб-середовищі, особливо при інтеграції з фронтендом. У проєктах на зразок розробки веб-сайту для подій SOAP не підходить, оскільки він занадто складний для задач, які можна ефективно вирішити за допомогою REST.

RPC (Remote Procedure Call) —фокусується на виклику функцій або процедур, розташованих на віддаленому сервері, ніби вони виконуються локально.

На відміну від REST, який орієнтований на ресурси, RPC орієнтований на дії. RPC може використовувати дві мови для кодування — JSON та XML. RPC добре підходить для мікросервісної архітектури, внутрішньосистемних викликів і високонавантажених систем, де важлива продуктивність і компактність трафіку. Проте для відкритих веб-систем, які працюють через браузер, RPC менш зручний у використанні, ніж REST, оскільки потребує складнішого налаштування та не підтримується стандартними інструментами JavaScript.

Отже, у межах створення веб-системи автоматизації планування заходів найдоцільнішим вибором є REST API. Він забезпечує простий, структурований і ефективний механізм взаємодії між клієнтом і сервером, легко масштабується, сумісний із сучасними веб-технологіями, і забезпечує достатній рівень безпеки при правильній реалізації. REST API використовується як для реалізації внутрішніх функцій платформи, так і для інтеграції з зовнішніми сервісами, такими як Google Calendar, платіжні системи або системи email-сповіщення.

У розроблюваній системі можна виділити кілька основних груп API:

API для управління подіями — дозволяє організатору створювати, редагувати та видаляти події. Наприклад, запит POST /events створює нову подію, GET /events/:id — отримує деталі, PUT /events/:id — оновлює, а DELETE /events/:id — видаляє.

API для реєстрації та авторизації — взаємодіє з базою даних для збереження облікових записів користувачів та реалізує перевірку прав доступу через JWT або OAuth2.

API для покупки квитків — забезпечує зв’язок з платіжними сервісами(LiqPay), включаючи надсилання запиту на оплату, отримання підтвердження та генерування квитка.

API для інтеграції з Google Calendar — дозволяє організаторам та відвідувачам зберігати події в календарі. В цьому випадку виконується запит до зовнішнього Google Calendar API (через авторизацію OAuth2), система викликає

метод events.insert, передаючи сформовані дані події (назва, дата, місце), після чого створює нову подію у календарі користувача.

API для аналітики — надає адміністраторам і організаторам статистичні дані, що формуються на основі інформації з бази даних та додаткових параметрів (Referer, IP).

Завдяки API досягається модульність, гнучкість та масштабованість системи. Кожна функціональна частина сайту взаємодіє з іншими через чітко визначені запити, що дозволяє у разі необхідності оновлювати або замінювати окремі компоненти без порушення цілісності всієї системи.

Для забезпечення надійності роботи системи API реалізовано з дотриманням основних принципів захисту, перевірки доступу та обробки помилок. Кожен запит до критичних маршрутів (наприклад, створення події або купівля квитка) супроводжується перевіркою автентифікації та авторизації. У випадку стандартної авторизації це реалізується через перевірку JWT-токена. Якщо користувач увійшов у систему через email/пароль, йому видається JWT-токен — спеціальний цифровий ключ. При кожному запиті до API цей токен передається у заголовку Authorization. Сервер перевіряє токен, і якщо він дійсний — дозволяє діяти. Якщо ні — повертає помилку (401 Unauthorized). Цей спосіб дозволяє уникнути повторної авторизації. У випадку інтеграції зі сторонніми сервісами (OAuth2) користувач надає одноразовий доступ, і система використовує відповідний токен для взаємодії з зовнішнім API.

Також важливим елементом є валідація вхідних даних. Кожен запит перед обробкою перевіряється на коректність: формат електронної пошти, обов’язкові поля для заповнення, правильність форматів для оплати тощо. Це дозволяє зменшити кількість помилок на сервері, а також запобігти некоректній поведінці користувача або потенційним атакам, наприклад XSS-атакам, під час яких зловмисник вставляє шкідливий JavaScript-код у поля для заповнення і цей код виконується у браузері інших користувачів..

Ще один важливий аспект — це обробка помилок і відповідей. API має повертати структуровані та передбачувані відповіді. Наприклад, у випадку помилки в аутентифікації — 401 Unauthorized. Усі повідомлення оформлюються у форматі JSON, що забезпечує зручність обробки на стороні фронтенду.

2.4 Діаграма прецедентів

Діаграма прецедентів (Use case diagram) візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання), описує функціональні аспекти системи. Але не показує порядок виконання кроків. Зображує функціональні вимоги (те, що система може зробити) з точки зору користувача.

Діаграми прецедентів складаються з 4 об’єктів (рис. 2.5): актор, прецедент (варіант використання), система, зв’язок.

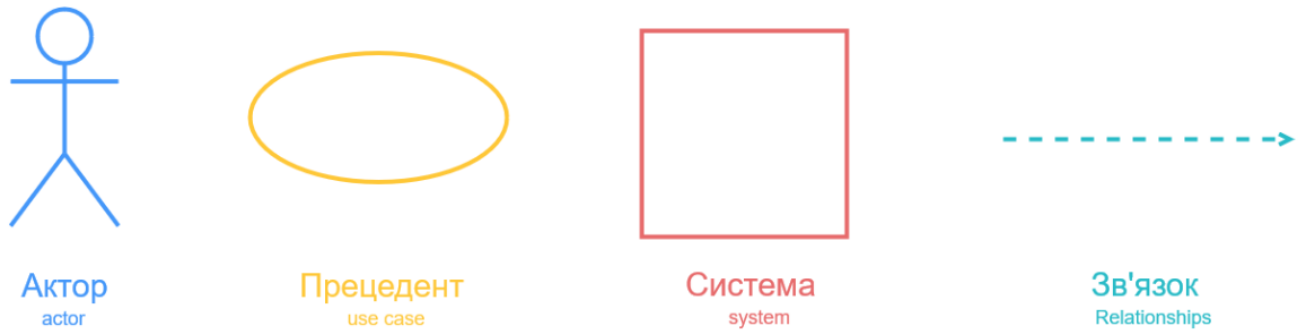

Рис. 2.5 Основні позначення діаграми прецедентів

Актор — хтось або щось, що взаємодіє з системою, але не належить до неї (знаходиться за межами системи). Зазвичай позначається у вигляді стилізованого чоловічка.

Усіх акторів умовно можна розділити на первинних (ті, що ініціюють взаємодію з системою) та вторинних (ті що реагують на взаємодію). Первинні зображуються зліва, а вторинні — справа від системи.

Оскільки будь-яка система може взаємодіяти не лише з людьми, актор не завжди позначає користувача-людину. Він може позначати іншу систему або пристрій з яким взаємодіє.

Актор — це поняття, що представляє клас користувачів (узагальнення групи користувачів), а не конкретного користувача, та може поєднувати в собі декілька ролей. Наприклад актор — працівник компанії може мати ролі інженер, менеджер, директор. В той час як користувач — це тип актора або його конкретна реалізація. Декілька користувачів можуть грати одну роль, тобто бути одним актором.

Прецеденти визначають очікувану поведінку та відповідають на питання, що робить система. Представляють набір можливих функцій або дій. Зображаються у вигляді еліпса з назвою дії всередині. Прецедент вказує, що має трапитись, проте не відповідає на запитання, як це має статись.

Система — те, що моделюється, наприклад сайт, мобільний додаток тощо. Позначається прямокутником, в середині якого знаїодяться прецеденти та зв’язки між ними.

Зв’язки між акторами та прецедентами ілюструють, хто має доступ до яких дій. Для уточнення залежностей між окремими прецедентами застосовуються різні типи зв’язків:

- Асоціація — зв’язок актора та прецеденту. Позначається суцільною непідписаною лінією.
- Extend — показує додаткову функціональність або можливий не обов’язковий варіант поведінки системи. Позначається пунктирною лінією зі стрілкою, що вказує на базовий прецедент, та стереотипом (підписом) <<extend>>. Розширюючий(extend) прецедент активується лише за виконання умови.

- Include — поведінка одного прецеденту включається як обов’язковий складовий елемент поведінки іншого прецеденту. Відношення включення використовується для уникнення дублювання однакових прецедентів та додає функціональність, не вказану в базовому. Відношення позначається пунктирною лінією зі стрілкою та стереотипом <<include>>.
- Generalization — батьківсько-дочірні відношення. Генералізація позначається суцільною лінією з трикутним не зафарбованим вказівником, що вказує на прецедент-предок. Предок або предки передають всі властивості до дочірніх прецедентів.

Для веб-системи автоматизації планування масових заходів була розроблена діаграма прецедентів (рис. 2.6) з 4-ма акторами(гість, відвідувач, організатор, адміністратор) та використанням 3-х типів зв’язків(association, extend, include). Вона демонструє, які функціональні можливості доступні кожному типу користувача. Це дозволяє отримати цілісне уявлення про логіку взаємодії користувачів із веб-системою.

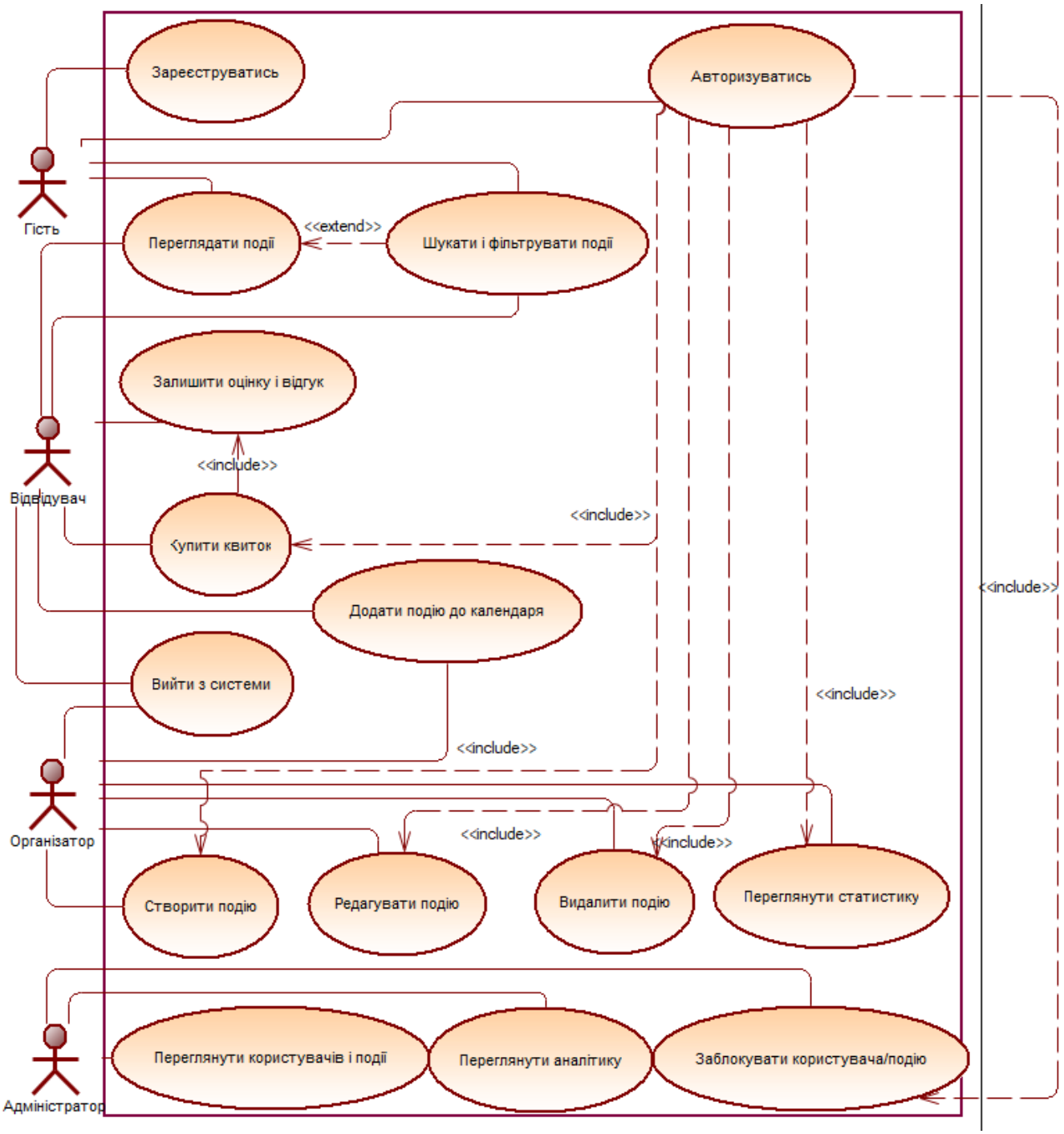

Рис. 2.6 Діаграма прецедентів

2.5 Вибір та проєктування бази даних

База даних — це структурована колекція даних, яка зберігається в електронному форматі та доступна для операцій пошуку, обробки та аналізу. БД є цифровим сховищем, що містить інформацію про різні об’єкти, які можуть бути пов’язані між собою. Вони є основою сучасних інформаційних систем, оскільки забезпечують централізований доступ до даних і дозволяють користувачам або системам швидко отримувати необхідну інформацію без дублювання чи втрати даних.

Залежно від структури організації даних та способів їх обробки, бази даних поділяють на прості, реляційні та нереляційні типи БД.

Прості типи БД мають просту структуру, зазвичай використовуються в невеликих програмах, для зберігання обмежених обсягів інформації або у випадках, коли складна структура не потрібна. Вони поділяються на:

- Текстові файли — інформація збирається в простих за структурою файлах різних форматів (txt, csv тощо). Для поділу полів застосовують табуляцію, крапку з комою, пробіли, коми та двокрапку. Такі БД легко використовувати. Але у таких базах важко встановити зв’язок між компонентами даних. Крім того, вони підходять не для всіх типів інформації.
- Ієрархічні БД — об’єкти поділяються на «пращурів» та «нащадків». При цьому кожен «нащадок» може мати не більше ніж одного «пращура». Графічно це деревоподібна структура.
- Мережеві БД — розширюють функціональність ієрархічного підходу за допомогою моделювання складних відносин між об’єктами. Тут «нащадки» можуть мати більше, ніж одного «пращура». Мережеві БД подаються загальним графом, замість дерева.

Реляційні БД організовані у вигляді таблиць, які мають чітко визначену схему, а кожен запис у таблиці є рядком з фіксованими полями. Реляційна модель заснована на строгих зв’язках між таблицями через первинні й зовнішні ключі. Такий підхід забезпечує структурованість і цілісність даних, а також дозволяє ефективно здійснювати складні запити за допомогою мови SQL (Structured Query Language). Серед популярних реляційних СУБД (систем управління базами даних) можна виділити MySQL, PostgreSQL, Oracle Database та Microsoft SQL Server.

Нереляційні бази даних (NoSQL) з’явилися у відповідь на потребу в масштабованості, гнучкості та швидкій роботі з неструктурованими або слабо структурованими даними. На відміну від реляційних БД, NoSQL-системи не потребують фіксованої схеми, що дозволяє легко змінювати структуру збережених об’єктів. Такі БД поділяються на такі підтипи:

- Документно-орієнтовані — зберігають дані у форматі документів, зазвичай XML, JSON або BSON. Вміст документа може мати різний набір властивостей.
- Колонкові — організовують дані по стовпцях замість рядків, що дає перевагу в аналітичних задачах.
- «Ключ-значення» — зберігають дані як пари, кожному ключу відповідає певне значення. У таких БД можливе зберігання та обробка різних за типом і змістом даних
- Графові бази — призначені для зберігання та обробки даних із складними взаємозв’язками у вигляді графів.

Кожен тип бази даних має свої переваги та обмеження. Прості бази даних відіграють важливу роль у задачах з невеликою складністю або коли потрібна мінімальна залежність від сторонніх серверів. Але у складніших веб-системах прості бази не забезпечують достатньої гнучкості. Реляційні БД відзначаються високою надійністю, підтримкою транзакцій і суворими правилами цілісності, однак можуть втрачати ефективність при роботі з великою кількістю змінюваних або слабо структурованих даних. Натомість NoSQL БД забезпечують

масштабованість і гнучкість, краще підходять для веб-додатків із високим навантаженням, але потребують більшої відповідальності з боку розробника у забезпеченні цілісності та узгодженості даних.

У проєкті автоматизації планування масових заходів, доцільно використовувати NoSQL-базу, оскільки вона найкраще поєднується з архітектурою REST API, JavaScript-стеком та дозволяє ефективно працювати з динамічними структурами даних. А саме документно-орієнтовану БД MongoDB.

MongoDB дозволяє зберігати документи з різною структурою в межах однієї колекції. Ця особливість робить MongoDB зручною для застосунків, які швидко розвиваються або у випадках, коли дані мають варіативну природу, як в розроблюваній системі, де об’єкти подій, користувачів чи квитків можуть мати додаткові поля в залежності від типу події або ролі користувача.

Також, концептуальна модель бази даних для MongoDB (рис. 2.7) має більш вільну структуру ніж реляційні БД, відображає лише основні сутності (колекції) та атрибути, де на рівні концептуальної моделі не обов’язково зазначати типи даних, не вимагає повної нормалізації (допустимо зберігати частину пов’язаних даних прямо всередині документа) та орієнтована на гнучке збереження та масштабування даних. Така модель краще підходить для веб-додатків, де швидкість розробки, адаптивність до змін і продуктивність мають вирішальне значення.

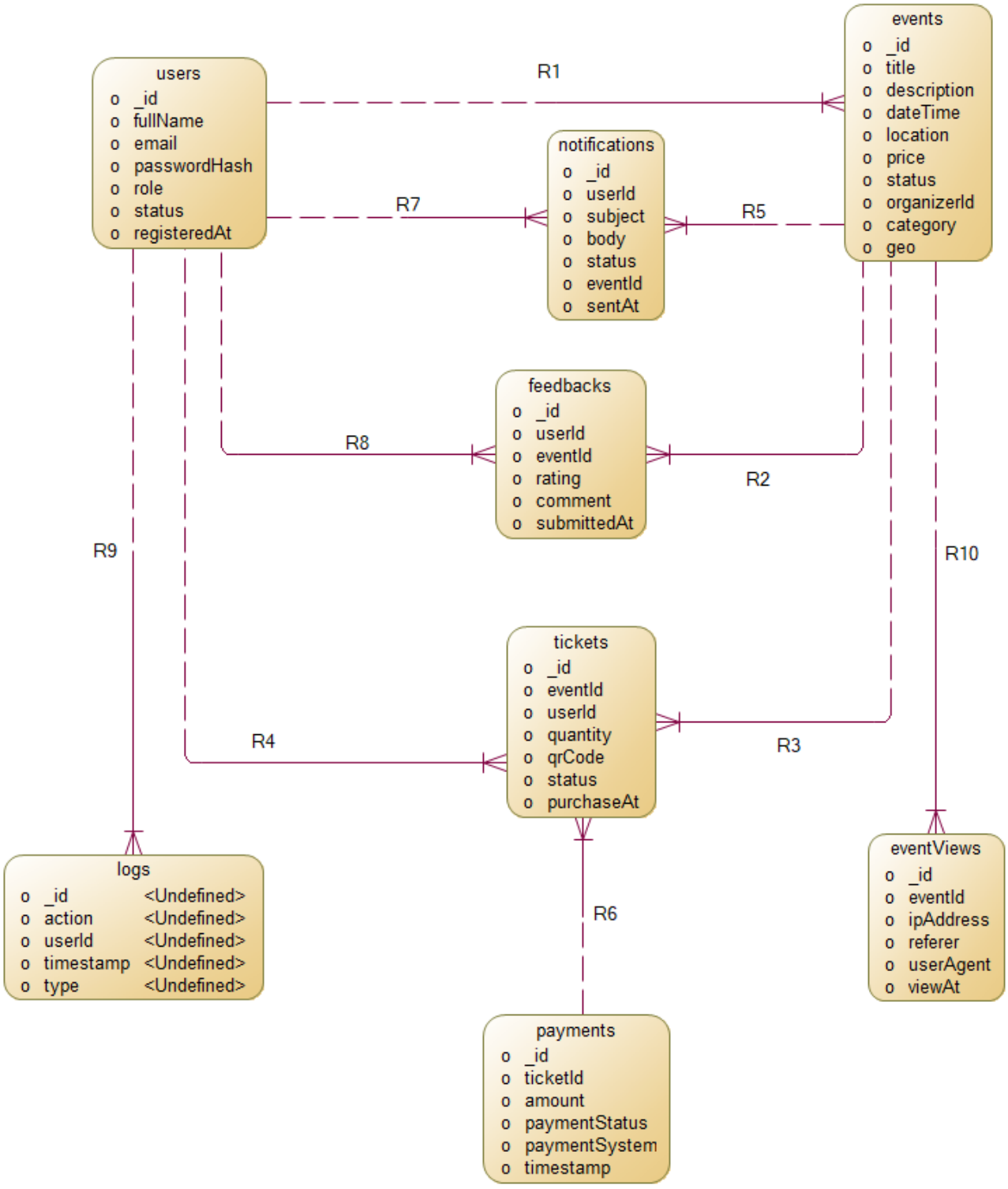

Рис. 2.7 Концептуальна модель бази даних

Опис зв’язків концептуальної моделі БД:

- R1 — кожен користувач (організатор) може створити багато подій, але кожна подія має лише одного організатора.

- R2 — одна подія може мати багато відгуків, але кожен відгук стосується лише однієї події.
- R3 — кожна подія має багато квитків, але кожен квиток пов’язаний лише з однією подією.
- R4 — один користувач може один і більше квитків, але кожен квиток належить лише одному користувачу.
- R5 — про кожну подію може надходити багато сповіщень, але кожне сповіщення повідомляє тільки про одну подію.
- R6 — за один платіж можна купити більше одного квитка, але кожен квиток належить до однієї оплати.
- R7 — користувач може отримати багато сповіщень, але кожне сповіщення адресоване лише одному користувачу.
- R8 — користувач може залишити багато відгуків, але кожен відгук належить тільки одному користувачу.
- R9 — один користувач може мати багато логів (журналів дій), але кожен лог пов’язаний лише з одним користувачем.
- R10 — подія можна переглянути багато разів, але кожен запис перегляду належить до конкретної події.

3. РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір середовища розробки та стеку технологій

Розробка веб-системи для автоматизації планування масових заходів вимагає чітко сформованого технологічного стеку, який охоплює інструменти для створення інтерфейсу користувача, реалізації серверної логіки, обміну даними через API, а також підтримки інтеграцій зі сторонніми сервісами. У цьому підрозділі розглянуто основні мови програмування, фреймворки та бібліотеки, які використовуються у проєкті, з описом їхнього функціонального призначення в системі.

Фронтенд — це частина системи, яка відповідає за те, що бачить і з чим взаємодіє користувач. Спершу треба створити макет інтерфейсу за допомогою онлайн-сервісу Figma. Для розробки інтерфейсу обрано класичний стек веб-технологій: HTML, CSS та JavaScript.

HTML — мова гіпертекстової розмітки документів, яка працює через систему тегів і допомагає структурувати та компонувати елементи вебсторінок. Вона визначає, де має бути заголовок, параграф, зображення, кнопока, форма тощо. HTML не виконує жодних дій, але вона критично важлива для організації візуальної ієрархії та доступності контенту.

CSS використовується для візуального оформлення веб-сторінок. Завдяки CSS можна керувати кольорами, шрифтами, відступами, адаптивністю, створювати анімації і компоненти. У системі планування заходів CSS застосовується для стилізації інтерфейсу, розробки адаптивного дизайну для мобільних пристроїв.

JavaScript — це мова програмування, яка забезпечує динамічну поведінку веб-сайту. У межах розроблюваної системи JavaScript використовується для:

- валідації форм (реєстрація, створення події).

- обробки подій користувача (натискання кнопок, взаємодія з модальними вікнами).
- реалізації фільтрації та пошуку подій без перезавантаження сторінки.
- асинхронного отримання та надсилання даних на сервер.

Бекенд — це частина системи, що працює на сервері та відповідає за обробку запитів, взаємодію з базою даних, автентифікацію користувачів, обробку платежів та реалізацію логіки системи. У цьому проєкті для створення серверної частини використовується середовище виконання Node.js та фреймворк Express.js.

Node.js — це середовище виконання JavaScript на сервері. Воно дозволяє використовувати JavaScript не лише у браузері, а й для побудови серверної логіки. Його основною перевагою є асинхронна модель виконання, що дозволяє ефективно обробляти велику кількість одночасних запитів, що особливо важливо для систем, де відбувається постійна взаємодія користувачів із сервером.

У контексті даної системи Node.js використовується для реалізації:

- API-запитів від фронтенду.
- логіки створення, оновлення та видалення подій і квитків.
- перевірки прав доступу (через JWT).
- підключення до бази даних.
- обробки помилок та логування подій.

Express.js — це фреймворк для Node.js, що спрощує створення серверних додатків і API. Express забезпечує простий спосіб організувати маршрути, які обробляють HTTP-запити типу GET, POST, PUT, DELETE. Завдяки Express можливо легко налаштувати обробку запитів.

Взаємодія між фронтендом і сервером реалізується через REST API.

REST (Representational State Transfer) — це архітектурний стиль, який передбачає використання стандартних HTTP-методів (GET, POST, PUT, DELETE) для доступу до ресурсів.

Для забезпечення додаткової функціональності в системі використовуються різноманітні бібліотеки та сторонні сервіси:

JWT (JSON Web Token) використовується для автентифікації користувачів. Після успішного входу сервер генерує токен(механізм безпеки, який використовується для підтвердження ідентичності користувача під час доступу до ресурсів або послуг), який містить зашифровану інформацію про користувача. Токен зберігається у браузері користувача та передається з кожним запитом. Сервер перевіряє його, щоб впевнитися, що запит здійснено авторизованим користувачем.

OAuth 2.0 застосовується для інтеграції з сторонніми сервісами, такими як Google чи Facebook. Замість зберігання логіну/паролю система отримує тимчасовий токен від стороннього сервісу, який дозволяє здійснювати авторизовані запити від імені користувача.

Для інтеграції з Google Calendar використовується офіційне API — Google Calendar API, що дозволяє додавати події до календаря користувача за допомогою авторизованого запиту через OAuth 2.0. Це забезпечує зручну функцію планування для користувача.

Для визначення географічного розташування користувача, яке використовується в аналітиці, система може використовувати API сторонніх сервісів, які за IP-адресою визначають країну, місто або регіон.

Усі ці технології та бібліотеки формують єдину інфраструктуру, яка дозволяє реалізувати гнучку, масштабовану та функціонально насичену систему для планування масових заходів. Використання перевірених сучасних рішень забезпечує швидку розробку, зручність підтримки та можливість майбутнього розширення системи.

3.2 Функціональні алгоритми системи

3.2.1 *Алгоритм авторизації*

Авторизація є критичним етапом у роботі веб-системи, що забезпечує доступ користувача до індивідуального функціоналу відповідно до його ролі — відвідувача події, організатора або адміністратора. Основна мета авторизації — ідентифікувати користувача, підтвердити його особу та надати доступ лише до дозволених дій. У системі реалізовано два основні способи авторизації: традиційний вхід за допомогою електронної пошти та пароля та автентифікація через сторонні сервіси — Google і Facebook. Такий підхід покращує зручність користування та підвищує рівень безпеки, оскільки дозволяє уникнути зберігання паролів на стороні сервери.

Після успішного входу система визначає роль користувача та перенаправляє його до відповідного інтерфейсу. Для адміністратора відкривається панель керування всією платформою, для організатора — інструменти управління подіями, а для звичайного користувача — сторінка з афішами, особистий кабінет, можливість купити квитки на подію та залишити відгук. Таким чином, авторизація є не лише перевіркою особи, а й точкою розгалуження доступу до функціоналу системи.

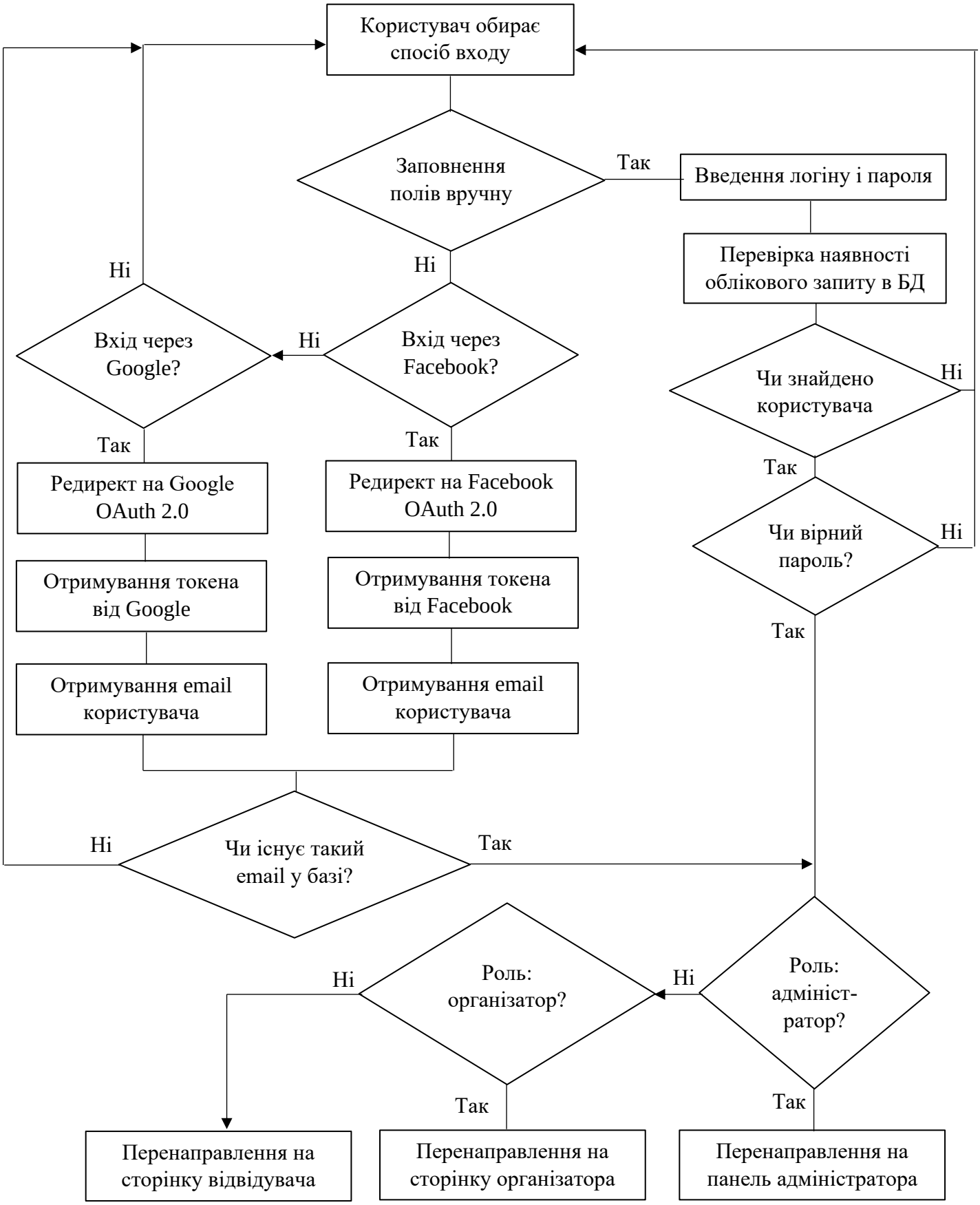

Рис. 3.1Алгоритм авторизації

3.2.2 *Алгоритм реєстрації*

Оскільки система підтримує декілька ролей (відвідувач події, організатор, адміністратор), важливо забезпечити правильне призначення ролі під час реєстрації. Кожна роль відкриває доступ до різного функціоналу: учасник може переглядати афіші та купувати квитки, організатор — створювати та керувати заходами, а адміністратор — контролювати всю систему, включаючи користувачів та події.

З міркувань безпеки та контролю, можливість самостійно зареєструватися як адміністратор не передбачена — ця роль призначається виключно вручну через базу даних. Це дозволяє уникнути несанкціонованого доступу до критично важливої частини платформи.

Щодо ролі організатора, то, хоча користувач може самостійно обрати її під час реєстрації, обліковий запис не стає дійсним зразу, а отримує статус "очікує підтвердження". Такий механізм дає змогу адміністраторам перевірити достовірність даних та запобігти створенню фейкових або недобросовісних облікових записів, які можуть зловживати можливістю публікації подій, продаючи квитки на неіснуючі події. Лише після ручного схвалення адміністрацією користувач отримує повноцінний доступ до функціоналу організатора.

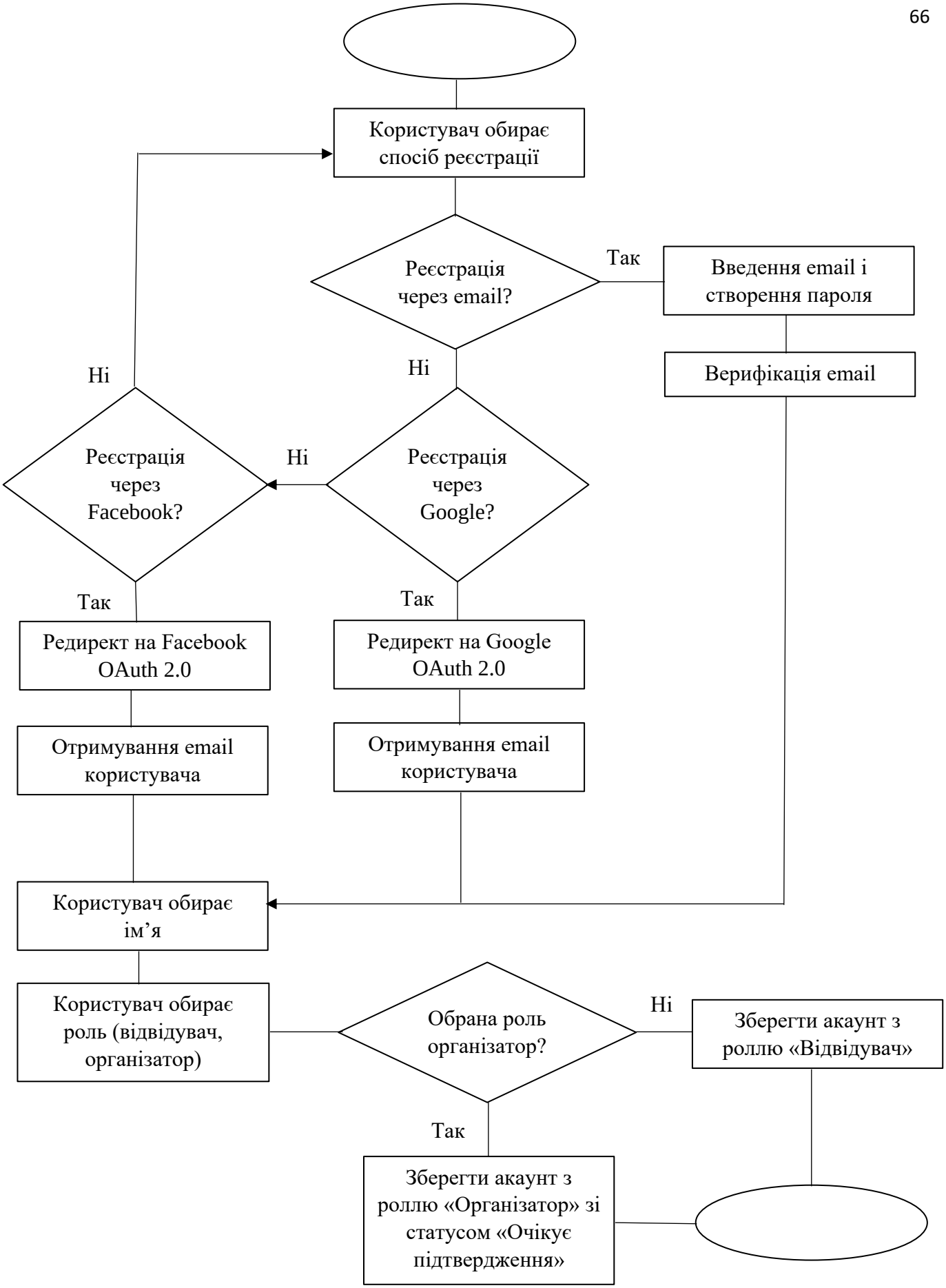

Рис. 3.2 Алгоритм реєстрації

3.3.3 *Алгоритм купівлі квитка*

Процес купівлі квитка забезпечує зручну взаємодію користувача із системою під час придбання квитка до обраної події, включає інтеграцію з платіжною системою LiqPay, а також подальшу генерацію унікального електронного квитка. Для максимальної зручності користувачів передбачено можливість оплати банківськими картками Visa та MasterCard.

Після вибору події та кількості квитків, користувач здійснює оплату через зовнішній платіжний шлюз. У випадку підтвердження транзакції, система автоматично створює електронний квиток, генерує унікальний QR-код і надсилає його на вказану електронну пошту. Також відповідна інформація зберігається у базі даних для подальшої верифікації при вході на подію. У разі скасування або помилки під час платежу користувачу надається можливість повторити спробу або змінити спосіб оплати. Така реалізація забезпечує надійний облік, автоматизований контроль доступу та зменшує можливість зловживань або дублювань квитків.

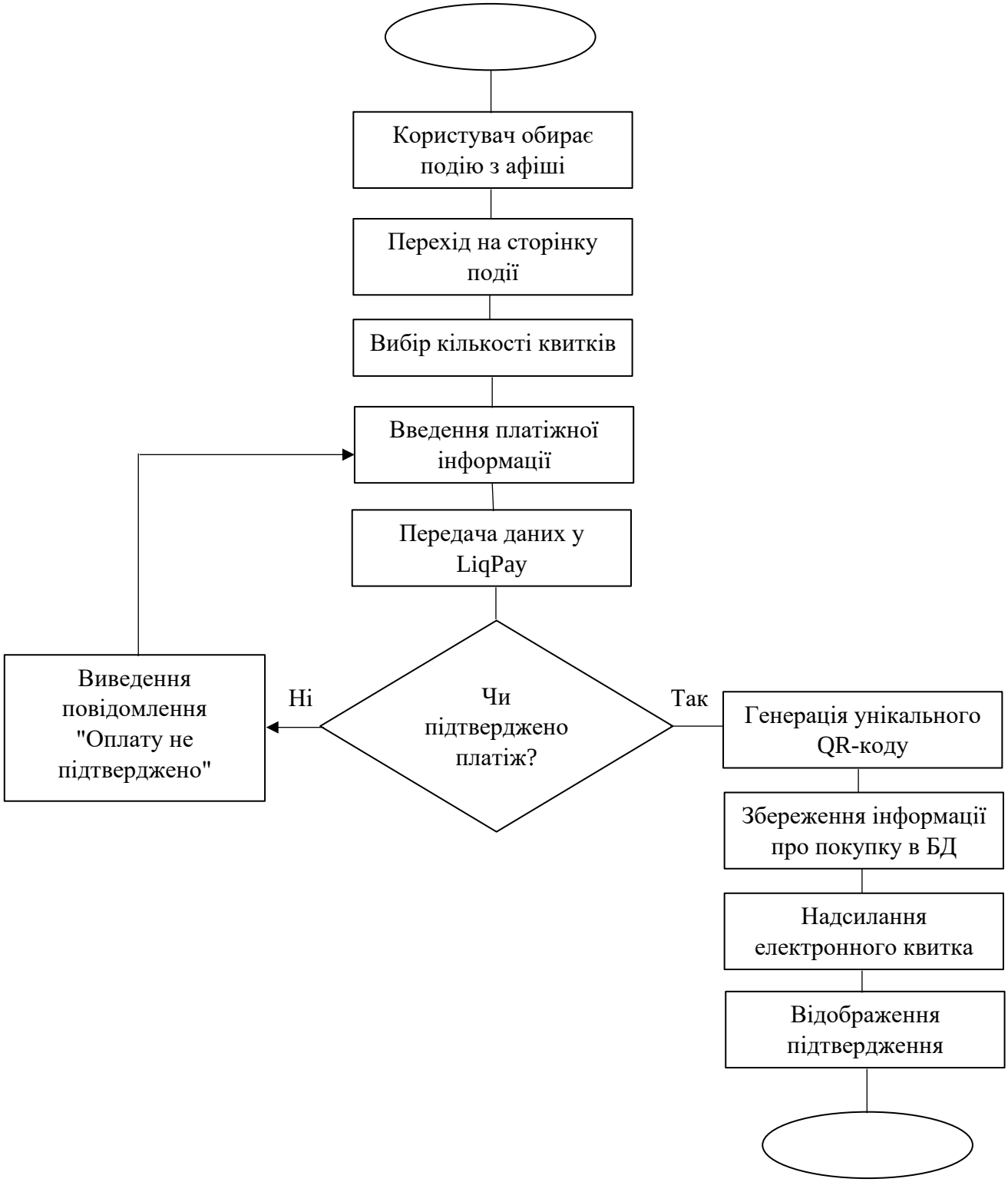

Рис. 3.3 Алгоритм купівлі квитка

3.3 Діаграма класів

Діаграма класів — статичне представлення структури моделі в UML. Вона демонструє, які класи існують у системі, які атрибути та операції вони мають, а також які зв'язки існують між цими класами. Кожен клас у діаграмі представляє собою узагальнення об'єкта реального світу або логічної одиниці програмного забезпечення, що дозволяє чітко описати логіку предметної області.

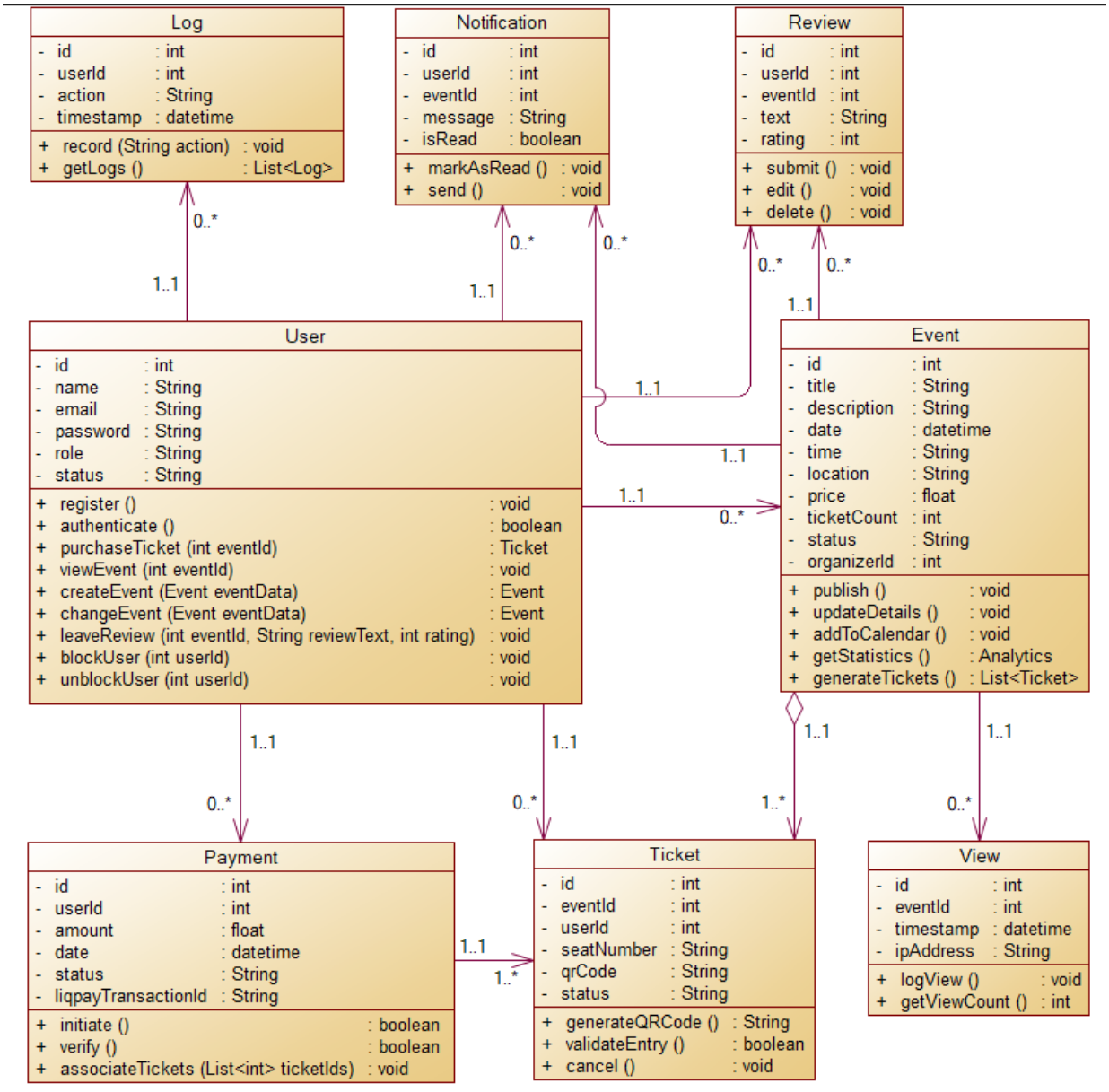

Рис. 3.4 Діаграма класів

Головною метою діаграми класів є забезпечення візуального представлення архітектури системи, яке є зрозумілим як для розробників, так і для аналітиків. Вона дозволяє заздалегідь продумати структуру даних, спроектувати взаємозв’язки між елементами системи, уникнути дублювання функціональності та забезпечити узгодженість у розробці. Завдяки наочності діаграми класів легше визначити ролі класів, розподіл обов’язків між ними, а також масштабованість і гнучкість архітектури.

У процесі створення інформаційної системи діаграма класів слугує основою для реалізації бази даних, формування бізнес-логіки та побудови взаємодії між фронтендом і бекендом. Зокрема, при розробці веб-системи автоматизації планування масових заходів, діаграма класів дає змогу систематизувати основні компоненти системи, такі як користувачі, події, квитки, платежі тощо, визначити між ними логічні зв’язки та забезпечити коректну побудову функціоналу. Це значно спрощує подальше програмування, тестування та супровід програмного продукту.

4. ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Інтерфейс веб-сайту

Інтерфейс користувача є важливою складовою будь-якого веб-сайту, особливо якщо він орієнтований на взаємодію з широким колом користувачів. Інтерфейс виконує не лише функцію візуального представлення даних, але й забезпечує зручність, ефективність та безпеку взаємодії між користувачем і системою.

При створенні інтерфейсу враховуються такі параметри, як інтуїтивність, швидкість доступу до ключових функцій, адаптивність до різних типів пристроїв (ПК, планшети, смартфони), а також візуальна привабливість. Добре продуманий інтерфейс зменшує час пристосування користувача, мінімізує кількість помилок і підвищує загальну задоволеність від користування системою.

Ергономіка інтерфейсу спрямована на те, щоб мінімізувати когнітивне навантаження користувача, скоротити кількість дій для досягнення цілей, забезпечити логічне групування елементів та передбачувану поведінку системи.

Інтерфейс повинен бути простим, логічним та адаптованим до потреб різних груп користувачів. Для цього реалізовано поділ на ролі: гість, зареєстрований користувач (відвідувач), організатор, адміністратор. Кожна роль має доступ лише до відповідного функціоналу, що забезпечує зрозумілий і мінімалістичний інтерфейс без перевантаження зайвими елементами.

Необхідними компонентами інтерфейсу системи це:

- головна сторінка з афішами подій;
- сторінка події з деталями та можливістю купити квиток для зареєстрованих відвідувачів;
- сторінка реєстрації/авторизації;

- профіль користувача;
- панель організатора для керування подіями;
- панель адміністратора для модерації;
- форма купівлі квитків;
- сторінка з аналітикою;
- адаптивне меню та фільтри.

Це загальний огляд елементів, що необхідні для забезпечення реалізації визначеного функціоналу. Після створення прототипу можна переходити до дизайну.

Макет інтерфейсу розроблений в сучасному онлайн-інструменті для дизайну інтерфейсів — Figma. А реалізований інтерфейс за допомогою сучасних технологій HTML, CSS, JavaScript.

При вході на сайт перед гостем відкривається головна сторінка сайту (рис. 4.1).

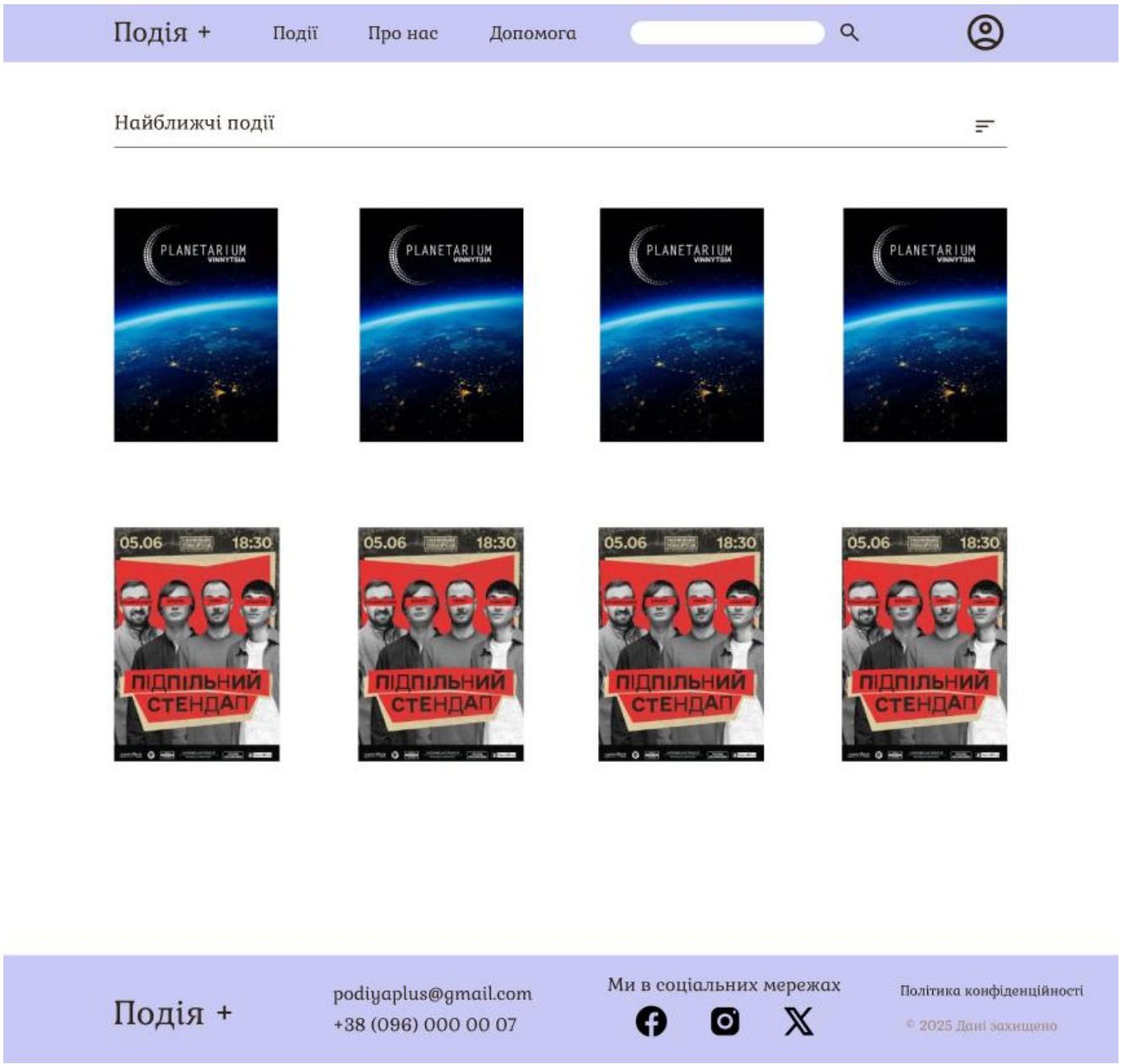

Рис. 4.1 Макет головної сторінки

У верхній частині розташовано навігаційний хедер, який включає логотип платформи, що водночас виконує функцію кнопки повернення на головну сторінку. Поруч знаходяться основні пункти меню: «Події», «Про нас» і «Допомога», які дозволяють користувачам швидко зорієнтуватися та перейти до відповідних розділів. Праворуч розміщена панель пошуку для зручного знаходження подій за ключовими словами, іконка доступу до профілю користувача, після натискання на яку відкриється модальне вікно аторизації та функції фільтрації подій під хедером.

Така навігаційна панель є простою та інтуїтивно зрозумілою, що відповідає принципам зручності інтерфейсу.

Основна частина сторінки присвячена блоку з найближчими подіями. Візуальне представлення реалізоване у вигляді сітки афіш — кожна картка містить зображення події, що дозволяє користувачеві швидко зорієнтуватися у візуальному контенті. Цей блок слугує основною функціональною частиною головної сторінки, адже одразу демонструє користувачеві актуальні події, тим самим сприяючи залученню до участі.

Нижня частина сторінки (футер) містить логотип, контактну інформацію, посилання на соціальні мережі та політику конфіденційності. Це забезпечує користувачам легкий доступ до підтримки та правової інформації, підвищує довіру до платформи та демонструє її відкритість.

Головна сторінка має такий вигляд на сайті:

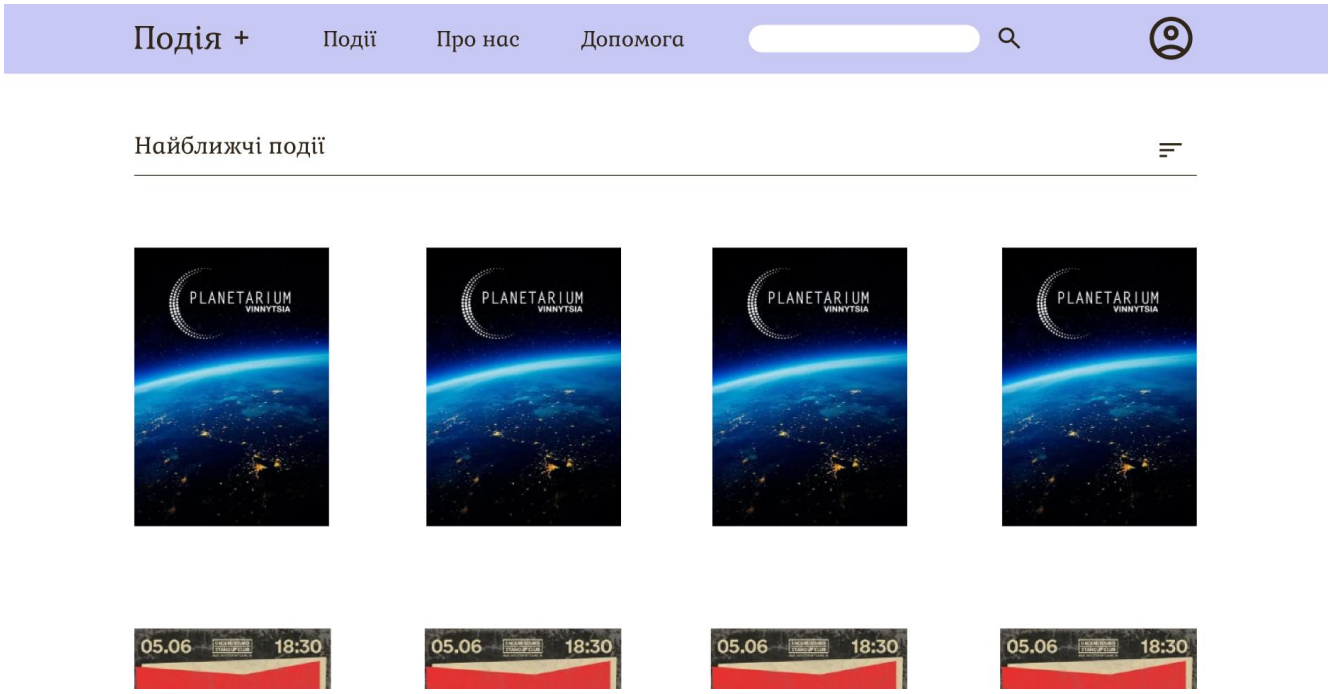

Рис. 4.2 Початок головної сторінки

Рис. 4.3 Кінець головної сторінки

При наведенні на афішу з’являється коротка інформація про неї, щоб користувач міг дізнатися, в який день та о котрій годині проходитиме подія, її вартість, місце проведення та назву. Зазвичай цього, разом із зображенням афіші, достатньо, щоб зацікавити потенційного клієнта.

Тож, якщо подія йому не сподобалася чи видалася занадто дорогою, він може продовжити пошук на головній сторінці. Такий підхід дозволяє зекономити час на завантаженні сторінки кожної події

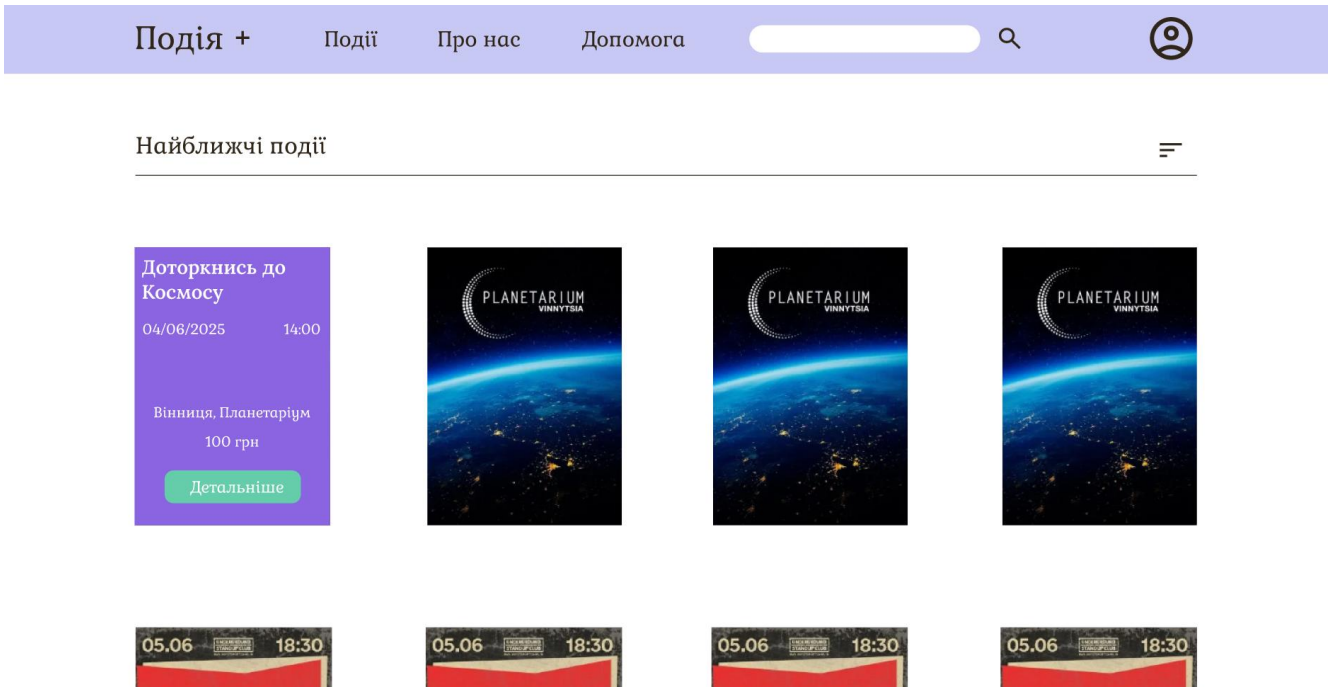

Рис. 4.4 Головна сторінка після наведення мишки на картку події

Якщо користувач зацікавився подією і хоче детальніше ознайомитись з подією, він може натиснути на кнопку «Детальніше» і перейти на сторінку події.

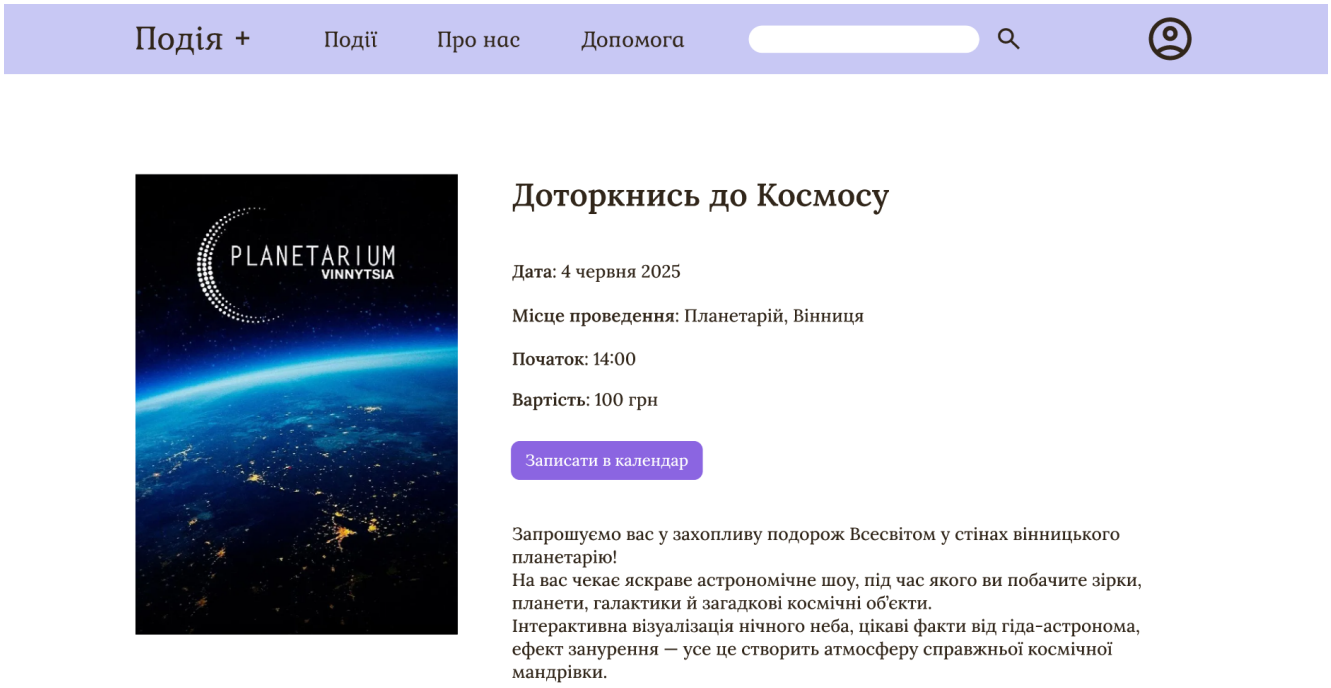

Рис. 4.5 Сторінка події. Частина 1

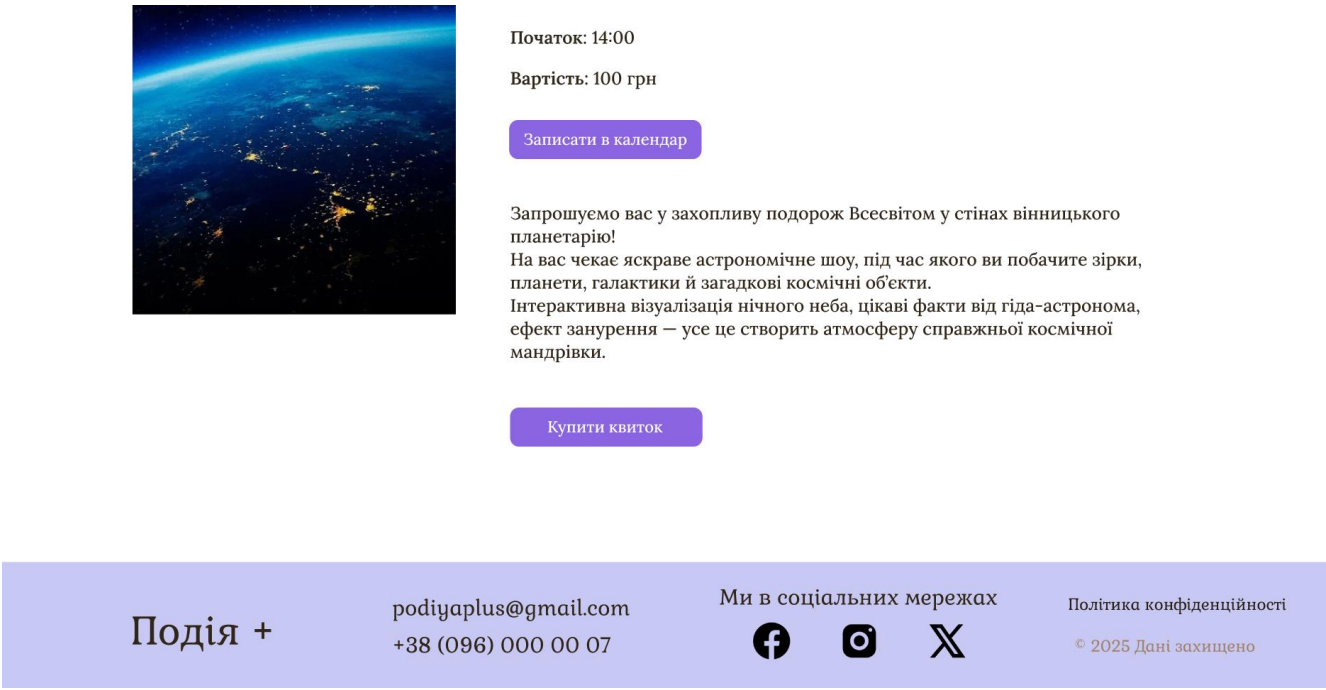

Рис. 4.6 Сторінка події. Частина 2

На цій сторінці розміщена та сама інформація, що й при наведенні на подію, доповнена детальнішим описом заходу, кнопкою купівлі квитка та кнопкою додавання події до Google Calendar.

Після натискання на будь-яку з цих кнопок з'явиться модальне вікно авторизації, якщо користувач не був авторизований раніше. У протилежному випадку користувач одразу зможе додати подію до календаря для планування відвідування або придбати квиток.

Таким чином, інтерфейс веб-сайту забезпечує швидкий доступ до ключових функцій, інтуїтивну навігацію, адаптивність до мобільних пристроїв і інтеграції з зовнішніми сервісами. Завдяки цьому платформа є зручною у використанні, сучасною та функціональною.

4.2 Техніко-економічне обгрунтування розробки

4.2.1 *Резюме*

Даний проєкт передбачає створення веб-системи «Подія+» для автоматизації планування масових заходів. Веб-платформа реалізована як зручний інструмент для відвідувачів, які зможуть швидко знаходити актуальні події, купувати квитки, зберігати їх у своєму обліковому записі та додавати заходи до Google Calendar. Також передбачена реалізація для організаторів подій, що дозволить створювати та публікувати афіші заходів, продавати квитки онлайн, відслідковувати статистику відвідуваності та взаємодіяти з відвідувачами. Водночас метою проєкту є створення ефективної, інтуїтивно зрозумілої та безпечної платформи, яка мінімізує витрати організаторів на технічну реалізацію, розширює можливості управління подіями, а також покращує досвід користувачів у сфері відвідування заходів. Система орієнтована на український ринок і буде доступна через веб-браузер із будь-якого пристрою.

Платформа «Подія+» — це веб-сайт, що виконує роль цифрової екосистеми для масових заходів. Основними функціональними можливостями є:

- створення та редагування афіш заходів з можливістю завантаження зображень, додавання опису, дати, часу, місця проведення;
- купівля квитків онлайн із отриманням QR-коду на email;
- особистий кабінет користувача з можливістю перегляду власних подій, історії квитків, відгуків тощо;
- інтеграція з Google Calendar для планування;
- адміністративна панель для модерації контенту, блокування акаунтів, перегляду логів та аналітики;
- аналітична панель для організаторів із візуалізацією статистики переглядів, покупок, відгуків;

4.2.2 *Оцінка ринку збуту*

Оцінка ринку збуту для веб-системи автоматизації планування масових заходів в Україні базується на аналізі поточних тенденцій у сфері електронної комерції та онлайн-продажу квитків.

Станом на 2023 рік, майже 10 мільйонів українців здійснюють покупки онлайн, з яких 1,5 мільйона зробили свою першу покупку в інтернеті цього року . Це свідчить про зростаючу довіру до цифрових сервісів та потенціал для розвитку нових онлайн-платформ.

У 2024 році Укрзалізниця перевезла 23,3 мільйона пасажирів, з яких 86% придбали квитки онлайн . Це демонструє, що більшість людей віддає перевагу онлайн квитким, а не купівлі фізичної версії у касі.

Хоча конкретні дані щодо обсягу ринку онлайн-продажу квитків на культурні, освітні та розважальні заходи в Україні обмежені, загальні тенденції електронної комерції та успіхи у сфері онлайн-продажу квитків на транспортні послуги свідчать про значний потенціал для розвитку подібних сервісів у інших галузях.

З огляду на зростаючу кількість інтернет-користувачів в Україні та успішний досвід впровадження онлайн-продажу квитків, існує значний потенціал для розвитку веб-систем автоматизації планування масових заходів. Це створює сприятливі умови для впровадження нових цифрових рішень у сфері організації подій.

4.4.3 *Стратегія маркетингу*

Успішне просування веб-системи «Подія+» базується на комплексному підході, що поєднує цифровий маркетинг, партнерства з організаторами та соціальну активність. Основна мета маркетингової стратегії — охопити якнайширше коло користувачів (відвідувачів і організаторів), привернути їхню увагу до функціональних переваг платформи та забезпечити стійке зростання активності.

Основні маркетингові інструменти включають:

- Просування у соціальних мережах (Facebook, Instagram, TikTok) — з використанням таргетованої реклами для різних сегментів аудиторії;
- Контент-маркетинг — створення інформаційних статей, відеороликів і гідів з організації подій, що підвищує впізнаваність бренду;
- Партнерські програми з організаторами подій — платформа пропонує вигідні умови співпраці для організаторів, що забезпечує постійне оновлення контенту;
- Email-розсилки — підтримка взаємодії з користувачами та повернення їх на платформу;
- Промокоди для знижок на квитки — інструмент залучення нових відвідувачів.

Ціль полягає не лише у залученні нових користувачів, а й у побудові довготривалих відносин із кожною аудиторією — через якісний сервіс, підтримку та функціонал, орієнтований на їхні потреби.

4.4.4 *Оцінка ризиків*

У процесі реалізації та експлуатації веб-системи можуть виникати як технічні, так і організаційні ризики. Їхнє вчасне виявлення дозволяють заздалегідь підготуватися до можливих проблем і зменшити ймовірність невдачі проєкту.

До ключових ризиків належать:

Технічні ризики:

- Можливі помилки у програмному коді, які можуть призвести до порушення роботи сервісу;
- Нестабільна робота API сторонніх сервісів (наприклад, поштових або платіжних);
- Збої в роботі хостингу або втрати даних у базі;
- Низький рівень безпеки, що може призвести до витоку особистої інформації.

Фінансові ризики:

- Перевищення початкового бюджету;
- Недостатній дохід на старті або повільна окупність;
- Залежність від зовнішніх сервісів із платною підпискою.

Маркетингові ризики:

- Недостатня впізнаваність бренду та низький трафік;
- Складність у залученні організаторів і відвідувачів на нову платформу;
- Високий рівень конкуренції з боку великих платформ, таких як Concert.ua, Karabas, TicketsBox тощо.

Для зменшення впливу ризиків передбачається:

- Регулярне тестування системи;

- Резервне копіювання даних;
- Використання перевірених API та хостингів з високим SLA;
- Забезпечення шифрування особистих даних користувачів;

Таким чином, усвідомлення потенційних загроз на ранньому етапі дозволяє не лише знизити їхній вплив, а й вибудувати систему з підвищеною стійкістю до зовнішніх і внутрішніх викликів. Комплекс заходів, спрямований на моніторинг, тестування та захист, є необхідною складовою як під час розробки, так і під час подальшої підтримки веб-системи. Збалансований підхід до управління ризиками сприятиме стабільному функціонуванню платформи, формуванню довіри серед користувачів і забезпеченню її конкурентоспроможності на ринку.

ВИСНОВКИ

У межах даної кваліфікаційної роботи була розроблена веб-системи для автоматизації планування масових заходів. Проведено аналіз предметної області, визначено актуальні проблеми, обґрунтовано доцільність цифровізації процесів організації заходів, сформульовано вимоги до системи та спроєктовано її архітектуру.

Розроблена система дозволяє організаторам подій ефективно керувати інформацією про заходи, спрощує процес публікації афіш, реалізує механізми онлайн-продажу квитків із інтеграцією платіжних сервісів, а також забезпечує зручний інтерфейс для користувачів — відвідувачів, які мають змогу переглядати події, застосовувати фільтри пошуку та купувати квитки у декілька кліків.

Ключовими перевагами запропонованої веб-системи є:

- автоматизація більшості дій, що раніше виконувались вручну;
- підтримка ролей (користувач, організатор, адміністратор) із відповідним функціоналом;
- інтеграція з зовнішніми сервісами, зокрема Google Calendar та email-розсилкою;
- можливість збору та аналізу аналітичних даних щодо переглядів, продажів та трафіку;
- захищена авторизація через JWT та OAuth2.

У процесі проєктування було створено функціональну архітектуру системи, реалізовано інтерфейс з різним рівнем доступу. Всі основні процеси системи описано за допомогою DFD-діаграм кількох рівнів, а також побудовано концептуальну модель бази даних з урахуванням специфіки нереляційного підходу.

Інтерфейс системи орієнтований на кінцевого користувача, відповідає принципам адаптивності, зручності та ергономічності. Для забезпечення якісної

взаємодії з системою було враховано вимоги до UI/UX, а також протестовано основні сценарії використання.

Таким чином, запропонована система може бути використана для ефективної організації масових заходів різного типу — концертів, лекцій, фестивалів, конференцій тощо. Вона дозволяє організаторам зекономити час на планування, отримати більш точну аналітику та покращити взаємодію з аудиторією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Правила проведення масових заходів під час війни [Електронний ресурс] // Укрінформ. [сайт]. – URL: https://www.ukrinform.ua/rubric-society/3918440-u-genstabi-nagadali-pravila-provedenna-masovih-zahodiv-pid-cas-vijni.html (date of access: 05.06.2025).
2. Нова послуга в Дії: знайдіть найближче укриття [Електронний ресурс] // Дія. [сайт]. – URL: https://diia.gov.ua/news/nova-posluga-v-diyi-znahodte-najblizhche-ukrittya-v-kilka-klikiv (date of access: 05.06.2025).
3. Pugpig State of the Digital Publishing Market Report 2022. URL: https://www.pugpig.com/wp-content/uploads/sites/3/2022/12/Pugpig-State-of-the-Digital-Publishing-Market-Report-2022-FINAL-1.pdf?utm_medium=email&_hsmi=237179599&_hsenc=p2ANqtz--MgNxQ7bEx7STWk8sLbBzp8T7GAvrMQRbDzZ7ijVxYikPmlYJB6lvP2XvTZTi0NfKN4T_gVx8pUeunIAcy3aCFRrJ4Qg&utm_content=237179599&utm_source=hs_automation (date of access: 05.06.2025).
4. Eventbrite – тарифні плани для організаторів [Електронний ресурс] // Eventbrite. [сайт]. – URL: https://www.eventbrite.com/organizer/pricing/ (date of access: 05.06.2025).
5. Ticketmaster – офіційний сайт [Електронний ресурс] // Ticketmaster. [сайт]. – URL: https://www.ticketmaster.com (date of access: 05.06.2025).
6. Neil Young rejects dynamic ticket pricing [Електронний ресурс] // The Guardian. [сайт]. – URL: https://www.theguardian.com/music/2025/mar/17/neil-young-rejects-dynamic-ticket-pricing-robert-smith (date of access: 05.06.2025).
7. Офіційний сайт Meetup [Електронний ресурс] // Meetup. [сайт]. – URL: https://www.meetup.com (date of access: 05.06.2025).

8. Bizzabo – платформа для організаторів подій [Електронний ресурс] // Bizzabo. [сайт]. – URL: https://www.bizzabo.com (date of access: 05.06.2025).
9. Concert.ua – квитковий сервіс [Електронний ресурс] // Concert.ua. [сайт]. – URL: https://concert.ua/uk (date of access: 05.06.2025).
10. Karabas – квитки онлайн [Електронний ресурс] // Karabas. [сайт]. – URL: https://karabas.com/ (date of access: 05.06.2025).
11. Gastroli.ua – квитковий сервіс [Електронний ресурс] // Gastroli.ua. [сайт]. – URL: https://gastroli.ua/ (date of access: 05.06.2025).
12. Принципи побудови сервлетів [Електронний ресурс] // JavaRush. [сайт]. – URL: https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture01 (date of access: 05.06.2025).
13. Архітектура з трьома рівнями [Електронний ресурс] // Portal Cripto. [сайт]. – URL: https://portalcripto.com.br/uk/dicionario/o-que-significa-arquitetura-de-tres-camadas-definicao-tipos-e-aplicacao/ (date of access: 05.06.2025).
14. Що таке DFD-діаграма? [Електронний ресурс] // MaxZosim.com. [сайт]. – URL: https://www.maxzosim.com/data-flow-diagrams/ (date of access: 05.06.2025).
15. Що таке API? [Електронний ресурс] // Hostiq.ua. [сайт]. – URL: https://hostiq.ua/blog/ukr/what-is-api/ (date of access: 05.06.2025).
16. Як спілкуються програми між собою [Електронний ресурс] // Speka. [сайт]. – URL: https://speka.media/use-pro-api-yak-spilkuyutsya-programi-miz-soboyu-vzd116\ (date of access: 05.06.2025).
17. Як будувати UML-діаграми [Електронний ресурс] // DOU.ua. [сайт]. – URL: https://dou.ua/forums/topic/40575/ (date of access: 05.06.2025).
18. Що таке база даних? [Електронний ресурс] // Sigma Software University. [сайт]. – URL: https://university.sigma.software/what-is-database/ (date of access: 05.06.2025).

19. Типи баз даних [Електронний ресурс] // DOU.ua. [сайт]. – URL: https://dou.ua/lenta/articles/types-of-databases/ (date of access: 05.06.2025).
20. Що таке MongoDB? [Електронний ресурс] // Guru99. [сайт]. – URL: https://www.guru99.com/uk/what-is-mongodb.html (date of access: 05.06.2025).
21. Що таке CSS? [Електронний ресурс] // CSS.in.ua. [сайт]. – URL: https://css.in.ua/article/shcho-take-css_3 (date of access: 05.06.2025).
22. Що таке HTML? [Електронний ресурс] // CSS.in.ua. [сайт]. – URL: https://css.in.ua/article/shcho-take-html_10 (date of access: 05.06.2025).
23. Що таке Node.js? [Електронний ресурс] // Dan-it.com.ua. [сайт]. – URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/
24. Express.js – огляд [Електронний ресурс] // Foxminded.ua. [сайт]. – URL: https://foxminded.ua/express-js/ (date of access: 05.06.2025).
25. Як український e-commerce пережив 2023 [Електронний ресурс] // Promodo.ua. [сайт]. – URL: https://www.promodo.ua/yak-ukrayinskiy-ecommerce-pereziv-2023?utm_source=chatgpt.com (date of access: 05.06.2025).
26. Відновлення онлайн-продажу квитків Укрзалізницею [Електронний ресурс] // Forbes.ua. [сайт]. – URL: https://forbes.ua/news/ukrzaliznitsya-vidnovila-onlayn-prodazh-kvitkiv-27032025-28330?utm_source=chatgpt.com (date of access: 05.06.2025).

ДОДАТКИ

Додаток А — index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/CSS/index.css">
                                        <link                rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:ops
z,wght,FILL,GRAD@40,400,0,0" />
    <link href="https://fonts.googleapis.com/css2?family=Gabriela&display=swap"
rel="stylesheet">

    <title>Document</title>
</head>
<body>
    <header>
        <div class="main-container">
            <div class="header-container">
                <a class="logo" href="index.html">Подія +</a>
                <nav class="header-nav">
```

```
            <ul>
                <li><a href="#events">Події</a></li>
                <li><a href="#">Про нас</a></li>
                <li><a href="#">Допомога</a></li>
            </ul>
        </nav>
        <input type="text" id="search">
            <span class="material-symbols-outlined" style="font-size:33px; position:relative; right: 65px;">search</span>
                <span class="material-symbols-outlined" style="font-size:45px;">account_circle</span>
    </div>
  </div>
</header>

<main>
  <section class="filter-section">
    <div class="main-container">
      <div class="filter-container">
        <p>
          Найближчі події
        </p>
```

```
                        <span class="material-symbols-outlined" style="font-size:30px;">sort</span>
            </div>
        </div>
    </section>

    <section class="events-section">
        <div class="main-container">
            <div class="events-grid-container">
                <div class="event-card">
                    <img src="/Images/event1.png" alt="event" class="event1-img">
                </div>

                <div class="event-card">
                    <img src="/Images/event1.png" alt="event" class="event1-img">
                </div>

                <div class="event-card">
                    <img src="/Images/event1.png" alt="event" class="event1-img">
                </div>

                <div class="event-card">
                    <img src="/Images/event1.png" alt="event" class="event1-img">
```

```
            </div>

            <div class="event-card">
                <img src="/Images/event2.png" alt="event" class="event2-img">
            </div>

            <div class="event-card">
                <img src="/Images/event2.png" alt="event" class="event2-img">
            </div>

            <div class="event-card">
                <img src="/Images/event2.png" alt="event" class="event2-img">
            </div>

            <div class="event-card">
                <img src="/Images/event2.png" alt="event" class="event2-img">
            </div>

            <div class="event-card">
                <img src="/Images/event2.png" alt="event" class="event2-img">
            </div>
        </div>
    </div>
  </section>
</main>
```

```
    <footer>
        <div class="main-container">
            <div class="footer-container">
                <a class="logo" href="index.html">Подія +</a>
                <ui>
                    <li>podiyaplus@gmail.com</li>
                    <li>+38 (096) 000 00 07</li>
                </ui>
                <p>Ми в соціальних мережах</p>
                <ui>
                    <li>Політика конфіденційності</li>
                    <li>© 2025 Дані захищено</li>
                </ui>
            </div>
        </div>
    </footer>
</body>
</html>
```

Додаток Б — index.css

```
body{
    width: 100%;
    height: 100%;
    font-family: gabriela;
    background-color: #ffffff;
    margin: 0;
    padding: 0;
}

.no-scroll {
    overflow: hidden;
    height: 100vh;
}

html{
    box-sizing: border-box;
}

*,
*::before,
*::after{
```

```
    box-sizing: inherit;
}

:root{
    --main-color: #C8C8F4;
    --text-color: #31261A;
    --accent-color: #8B65E1;
    --green-color: #66CDAA;
    --red-color: #8b6570;
}

header{
    height: 80px;
    background-color: var(--main-color);
    display: flex;
}

.main-container{
    width: 1200px;
    height: 100%;
    margin: 0 auto;
    display: flex;
```

```
    justify-content: space-between;
    align-items: center;
}

.header-container{
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

a{
    text-decoration: none;
    color: var(--text-color);
}

.logo{
    font-size: 36px;
}

.logo:hover{
```

```
    color: var(--accent-color);
    transition: color 0.3s;
}

.header-nav{
    display: inline-block;
}

.header-nav ul{
    list-style-type: none;
    display: flex;
    gap: 80px;
}

.header-nav a{
    font-size: 22px;
}

.header-nav a:hover{
    color: var(--accent-color);
    transition: color 0.3s;
}
```

```
.header-container input{
    width: 250px;
    height: 31px;
    border: none;
    border-radius: 20px;
    outline-style: none;
    padding-left: 15px;
    font-family: gabriela;
    font-size: 14px;
}

.material-symbols-outlined:hover{
    cursor: pointer;
    color: var(--accent-color);
    transition: color 0.3s;
}

.filter-section{
    height: 100px;
}
```

```
.filter-container{
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding-top: 40px;
    position: relative;
}

.filter-container p{
    font-size: 24px;
}

.filter-container::after{
    content: "";
    position: absolute;
    bottom: 0;
    width: 100%;
    height: 1.5px;
    background-color: var(--text-color);
}
```

```
.events-section{
    height: 900px;
}

.events-grid-container{
    width: 100%;
    height: 100%;
    padding-top: 100px;
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: auto auto;
    column-gap: 105px;
    row-gap: 20px;
}

.event-card{
    width: 220px;
    height: 314px;
}

.event-card img{
```

```
    width: auto;
    height: auto;
}

footer{
    height: 90px;
    background-color: var(--main-color);
}

.footer-container{
    width: 100%;
    height: 100%;
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 30px;
    align-items: center;
}
```