МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської кваліфікаційної роботи на тему:

«Система аналізу даних для опублікування афіш»

Студента групи САА Д-21, Демчук Ю.М.
(група, шифр, прізвище та ініціали)

Керівник роботи _____ (**Ольга ЛОЗИНСЬКА**)

Консультант _____ (**Зоряна КОВАЛЬ**)

Нормоконтроль _____ (**Андрій ВАСИЛЮК**)

Завідувач кафедри ІСМ _____ (**Василь ЛИТВИН**)

«_01_» __грудня_____ 2023 р.

ЛЬВІВ - 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Інститут комп’ютерних наук та інформаційних технологій
Кафедра “Інформаційні системи та мережі”
Спеціальність 124 “Системний аналіз”
ОПП Аналіз даних (Data Science)

ЗАТВЕРДЖУЮ:

Зав. кафедри ІСМ _____
“_01_” __грудня____2023р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу студента групи **САД-21**

Демчук Юрій Миколайович
(прізвище, ім’я, по батькові)

1.Тема роботи: Система аналізу даних для опублікування афіш

затверджена наказом за НУ «ЛП» від «01» _листопада____ 2023 р. № _4692-4-06_

2. Термін здачі студентом закінченої роботи: _____24.11.2023 р.

3. Вихідні відомості до роботи: застосунок, який за допомогою рекомендаційної системи надаватиме користувачеві відсортовані дані.

4. Зміст розрахунково-пояснювальної записки: (перелік питань, які належить розробити) _ аналітичний огляд літературних та інших джерел, системний аналіз об’єкта дослідження, програмні засоби розв’язання задачі, тестування, практична реалізація.

5. Перелік графічного матеріалу: схема поведінки системи аналізу комплектуючих, діаграма класів, схема процесу функціонування системи, діаграма розгортання та зображення розроблюваного продукту.

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту) _компоненти для створення веб-сторінки Angular, TypeScript, HTML, CSS, JavaScript, Bootstrap реляційна система керування базами

3

даних MySQL, програмна платформа Java, Spring Boot, Spring MVC, Spring Data, Spring Security, REST, Junit, Maven.

7. Консультування роботи, із зазначенням розділів роботи

Розділ	Консультанти	Підпис, дата	
		завдання видав	завдання отримав
Економічний	Коваль З.О.3		

8. Дата видачі завдання __04.09.2023____

Керівник _____

(підпис)

Завдання отримав_____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

	Етапи кваліфікаційної роботи	Термін виконання	Примітки
1	Вивчення літературних джерел	10.10.2023	Зроблено
2	Написання розділу 1	10.10.2023	Зроблено
3	Написання розділу 2	13.10.2023	Зроблено
4	Написання розділу 3	13.10.2023	Зроблено
5	Розробка програмного продукту	17.10.2023	Зроблено
6	Розгортання програмного продукту	17.10.2023	Зроблено
7	Оформлення пояснювальної записки до розділу 4	17.10.2023	Зроблено
8	Написання розділу 5	25.10.2023	Зроблено
9	Написання вступу та висновку	25.10.2023	Зроблено
10	Написання анотації	10.11.2023	Зроблено

Студент_____

(підпис)

Керівник роботи_____

(підпис)

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1 8
Аналіз сучасного стану та перспектив в галузі досліджень 8
1.1. Аналітичний огляд літературних та інших джерел 8
1.2. Порівняння аналогів розроблюваного продукту 8
Висновок до першого розділу 20
РОЗДІЛ 2 21
Системний аналіз та моделювання предметної області 21
2.1. Аналіз мети функціонування системи 21
2.2. Моделювання вимог системи та ризики 21
2.3. Моделювання об’єктів системи 26
2.4. Моделювання процесів системи 29
Висновок до другого розділу 31
РОЗДІЛ 3 32
Розроблення проекту розв’язання задачі 32
3.1. Формулювання та обґрунтування задачі 32
3.2. Побудова моделі для розв’язання задачі 32
3.3. Вибір та обґрунтування методів розв’язання задачі 34
3.4. Розроблення алгоритмів розв’язання задачі. 35
3.5. Вибір і обґрунтування засобів розроблення проекту 36
3.6. Опис розроблених засобів проекту 41
Висновок до третього розділу 43
РОЗДІЛ 4 44
Тестування та впровадження проекту 44
4.1. Тестування проекту 44

4.2. Валідація та верифікація 53
4.3. Розгортання 56
Висновок до четвертого розділу 57
РОЗДІЛ 5 59
Економічне обґрунтування доцільності роботи 59
5.1. Економічна характеристика проектного рішення 59
5.2. Розрахунок витрат на розробку та впровадження проектного рішення 59
5.3. Визначення комплексного показника якості 63
5.4. Визначення експлуатаційних витрат 65
5.5. Розрахунок ціни споживання проектного рішення 68
5.6. Визначення показників економічної ефективності 69
Висновки до п’ятого розділу 71
ВИСНОВКИ 72
СПИСОК ЛІТЕРАТУРИ 74
АНОТАЦІЯ 77
ABSTRACT 80
ДОДАТКИ 83

ВСТУП

Використання інтернет технологій значно збільшується в усіх сферах людського життя, включаючи розважальну галузь. Однією з основних причин зростаючого інтересу маркетологів є легкість та зручність побудови та створення веб-ресурсів, маючи у наявності велику кількість інструментів, а також велика кількість аудиторії та відносна доступність вартості. За допомогою веб-ресурсів можна значно підвищити ефективність процесів реклами і продажу, і вже зараз це є стандартом у сучасному світі.

Сфера продажу квитків на різноманітні події вже давно є насиченою, і є кілька великих квиткових операторів у кожній країні, які користуються популярністю серед користувачів протягом тривалого часу. Запровадження карантинних обмежень додало труднощів у цій сфері, оскільки культурно-масові заходи були заборонені у більшості країн світу. Однак з вакцинацією населення ситуація змінюється, і попит на розваги повертається після тривалого примусового «голодування».

Якщо у вас є бажання створити невеликий виступ для локального гурту або відвідати концерт улюбленого артиста, який оминув вашу країну, що робити. У першому випадку обмежені можливості фінансування не дають змоги ефективно привертати увагу аудиторії, тоді як у другому виникають труднощі з пошуком квитків на сайтах з незнайомою мовою.

Формування бачення. Покращення пошуку подій для користувачів.

Завдання роботи. Створення веб ресурсу для перегляду та опублікування афіш подій.

Актуальність роботи полягає у можливості створення власних оголошень, перегляду існуючих незалежно від організатора та країни, а також у можливості придбання квитків у верифікованих квиткових операторів.

Об’єкт дослідження. Процес автоматизації розміщення оголошення та побудови рекомендації.

Предмет дослідження. Методи і засоби аналізу даних про події, користувачів, а також розроблення системи аналізу для опублікування афіш.

Мета і *задача* дослідження полягає в створенні конкурентоздатної інтернаціональної платформи для розміщення афіш різноманітних подій - від невеликих акустичних вечорів до великих фестивалів. Основне завдання полягає в створенні зручного та ефективного способу перегляду і придбання квитків на події, які пропонуються різними квитковими операторами, усе це на єдиній онлайн-платформі. Такий підхід дозволить уникнути роздрібненості і запропонує користувачам зручний і однаковий досвід придбання квитків незалежно від масштабу події.

Завдання дослідження 1: Здійснити аналіз предметної області, а саме дослідити веб-ресурси квиткових операторів.

Завдання дослідження 2: Провести системний аналіз об’єкта дослідження.

Завдання дослідження 3: Моделювання бізнес процесів та вибір засобів реалізації.

Завдання дослідження 4: Створення системи аналізу, базуючись на визначеній концепції.

Інноваційність результатів роботи. Розроблена система надає можливість публікувати афіші про події усім користувачам, також надає зручний список рекомендацій.

РОЗДІЛ 1

Аналіз сучасного стану та перспектив в галузі досліджень

1.1. Аналітичний огляд літературних та інших джерел

Галузь розваг є складовою туристично-інфраструктурного комплексу, що включає готелі, парки атракціонів, нічні клуби, ресторани, бари, вар'єте та інші заклади[1]. Сучасна індустрія розваг тісно пов'язана з індустрією туризму, створюючи нові туристично привабливі "атракціони". Рівень задоволення якістю відпочинку та розваг став важливим показником соціального статусу людини, а для суспільства – індикатором економічного та соціокультурного розвитку країни загалом. Одним із ключових факторів у цій галузі є спосіб, яким об'єкт розваг представляється майбутнім клієнтам. Якість представлення вирішує, чи варто відвідати концерт, виступ чи фестиваль. Однією з таких "обгорток" є афіша подій у мережі Інтернет.

Системи розміщення оголошень грають важливу роль у взаємодії через Інтернет. Коли у людини з'являється бажання відвідати концерт чи фестиваль, він часто починає пошук квитків саме в Інтернеті через його зручність та простоту. Унікальність також є ключовим аспектом у цій галузі. Якщо сервіс може запропонувати щось, чого немає у конкурентів, це може привернути більше користувачів. Це можна досягти за допомогою введення унікальних функцій, зручного та привабливого дизайну, а також індивідуального підходу до клієнтів.

1.2. Порівняння аналогів розроблюваного продукту

Найпопулярніші продукти працюють у формі інтернет-сервісів та мають обмежене представлення у різних містах. Для отримання квитка можна звернутися до офісів обслуговування або скористатися послугами поштового сервісу. Крім того, доступний варіант - отримання електронного квитка на електронну пошту або у особистому кабінеті. Важливо зазначити, що українські сервіси вирізняються кращим дизайном та більш продуманим інтерфейсом порівняно з іноземними аналогами.

Давайте здійснимо більш ретельний аналіз трьох найвідоміших українських платформ: Concert.ua, TicketsBox, Karabas, а також однієї з найпопулярніших зарубіжних платформ Eventbrite. Розглянемо їх веб-сайти, оскільки саме це є ключовим інструментом для залучення клієнтів та взаємодії з ними.

Розглянемо веб-ресурс concert.ua.

Сайт Concert.ua вражає унікальною стилістикою, відмінним оформленням та зручними фільтрами (див. рис. 1.1-1.3), завдяки чому він вирізняється серед своїх конкурентів. Інтерфейс дуже привітний до нових користувачів і легко зрозумілий, що робить його особливо зручним. Concert.ua взаємодіє з клієнтами як через власний веб-сайт, так і через мережу квиткових офісів. Також існує можливість оформити підписку на електронну розсилку та отримувати актуальні новини про події у вашому місті. Крім того, компанія надає можливість придбати квитки на унікальні заходи.

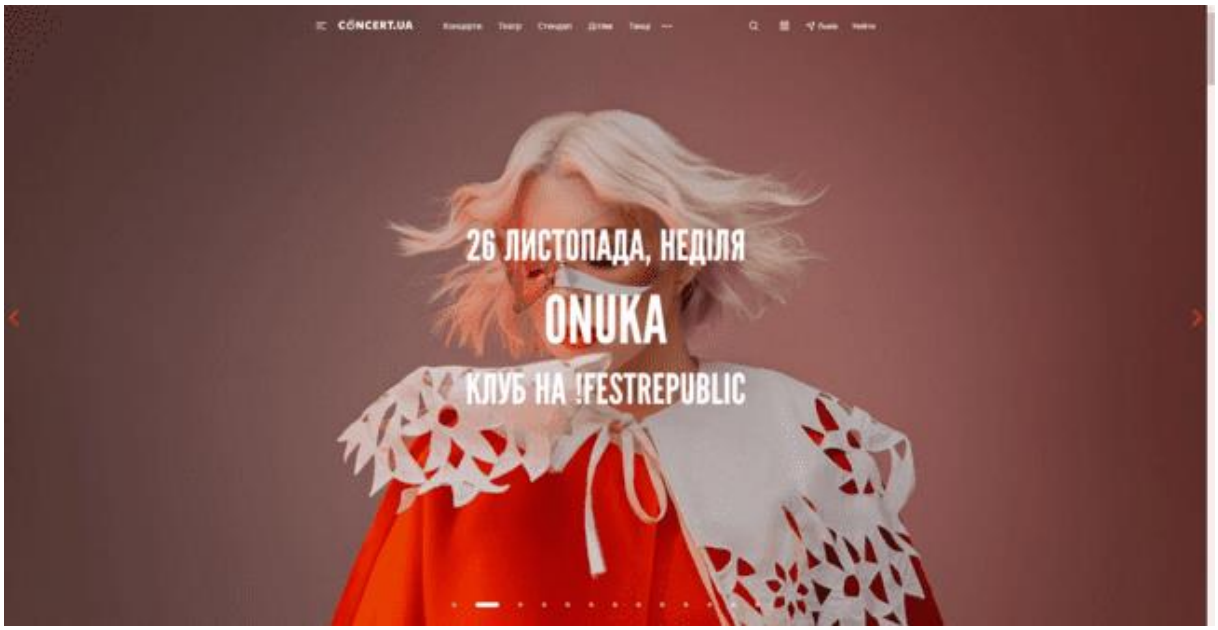

Рис. 1.1. Головне вікно Concert.ua, скріншот 1

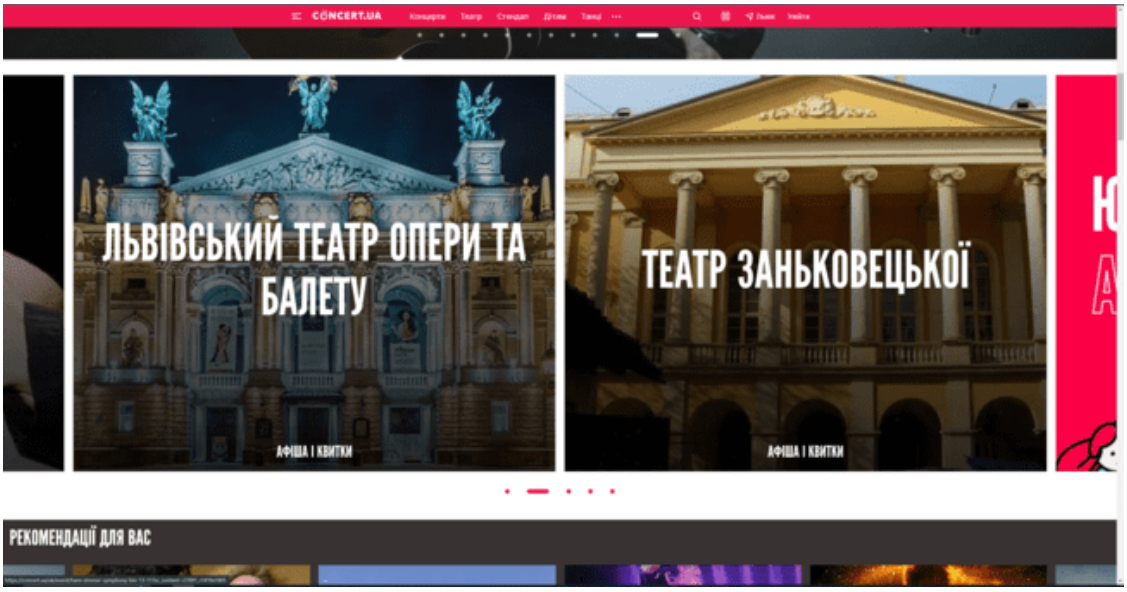

Рис. 1.2. Головне вікно Concert.ua, скріншот 2

Дизайн сайту добре продуманий і зосереджує на головному. Текст виглядає чітким завдяки контрастному фону. Concert.ua пропонує квитки на ексклюзивні події, які не доступні на інших платформах. Також помітно загальний стиль оформлення афіш, що становить його перевагу. Інформація на сайті добре відображається як на смартфонах, так і на великих екранах, забезпечуючи зручність для користувачів у будь-який момент.

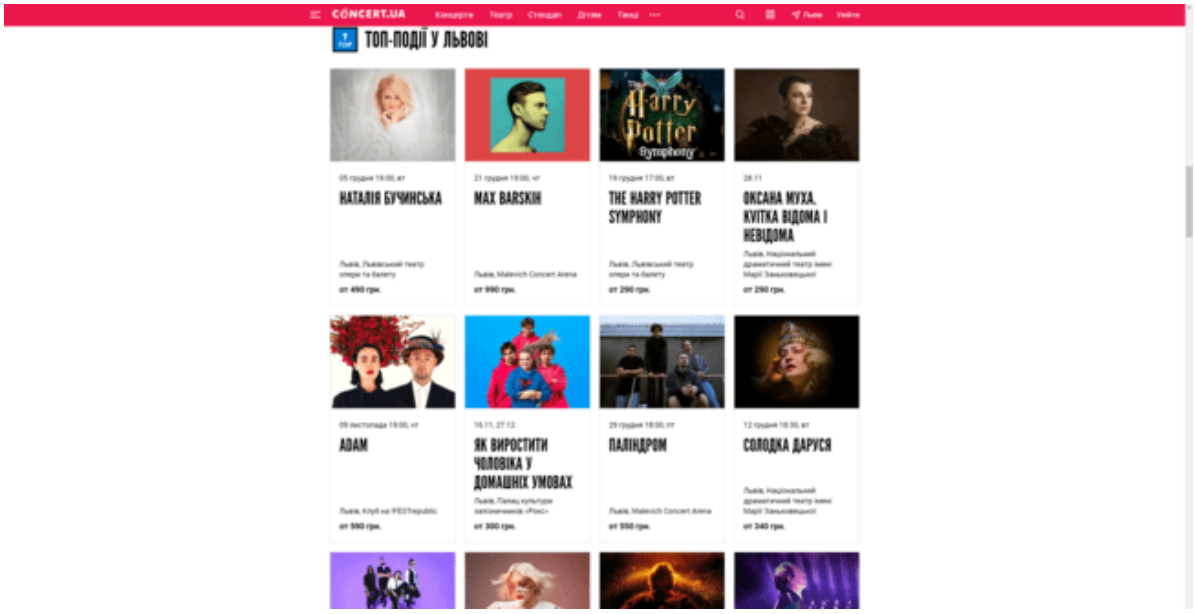

Рис. 1.3. Головне вікно Concert.ua, скріншот 3

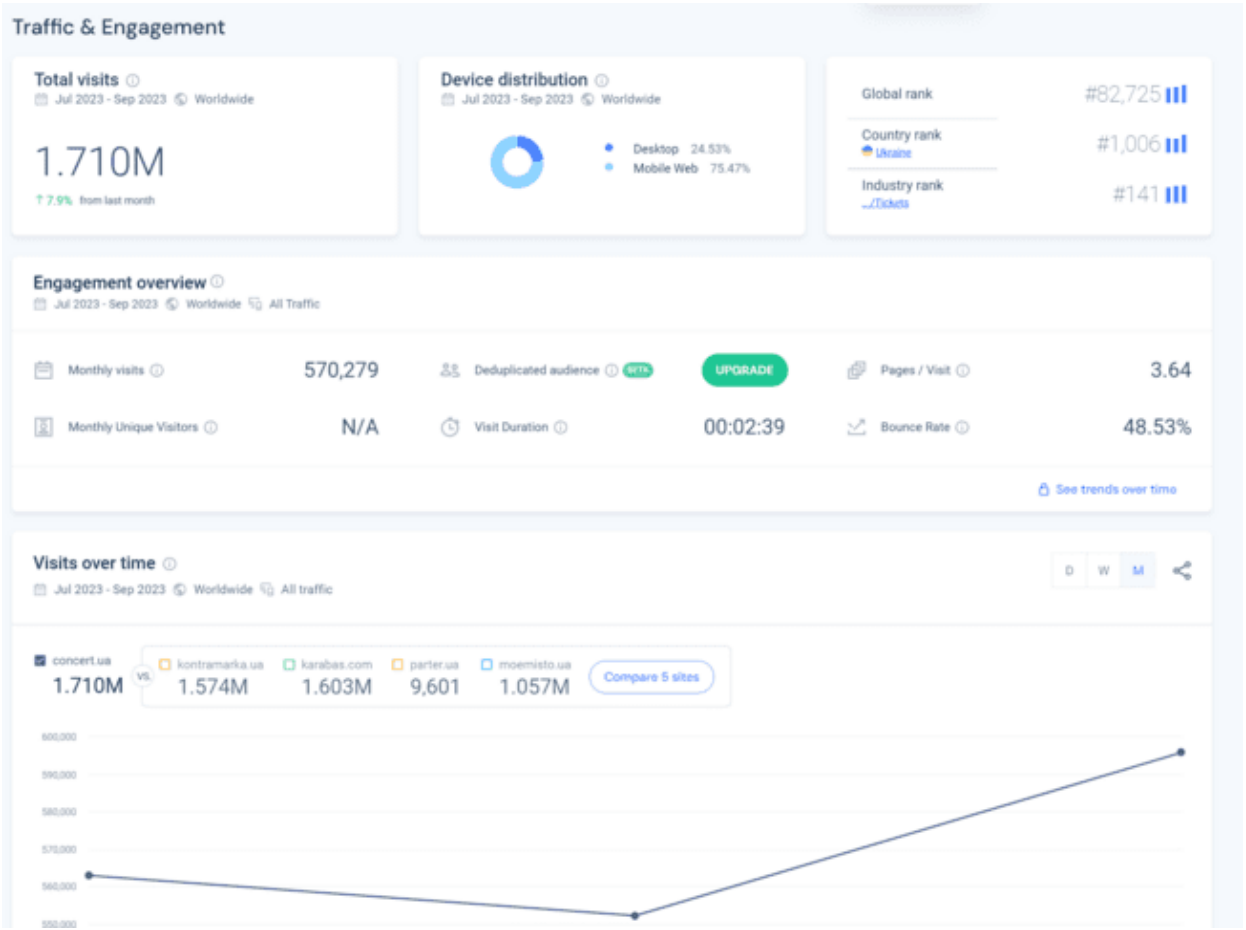

Рис. 1.4. Статистика зі сайту similarweb

Згідно із ресурсом similarweb за останній період у півроку на сайт перейшло майже 1710 тисяч користувачів відкрило 3,64 веб-сторінки з середньою тривалістю перебування 2 хвилини 39 секунд. Короткий час, проведений на сайті, свідчить про те, що відвідувачі заходять на нього з конкретною метою – знаходження певної події. Про це також свідчить велика кількість перегляду з мобільних пристроїв, де люди переходять за посиланням з сторінок соціальних мереж.

Розглянемо веб-ресурс TicketsBox .

Сайт TicketsBox має приємний дизайн (див. рис. 1.5-1.7), гарно оформлений і зручно розділений за категоріями. TicketsBox взаємодіє з клієнтами через власний веб-сайт. Отримати квиток можна лише у електронному варіанті. Також доступна можливість підписатися на розсилку за електронною поштою та отримувати актуальні новини.

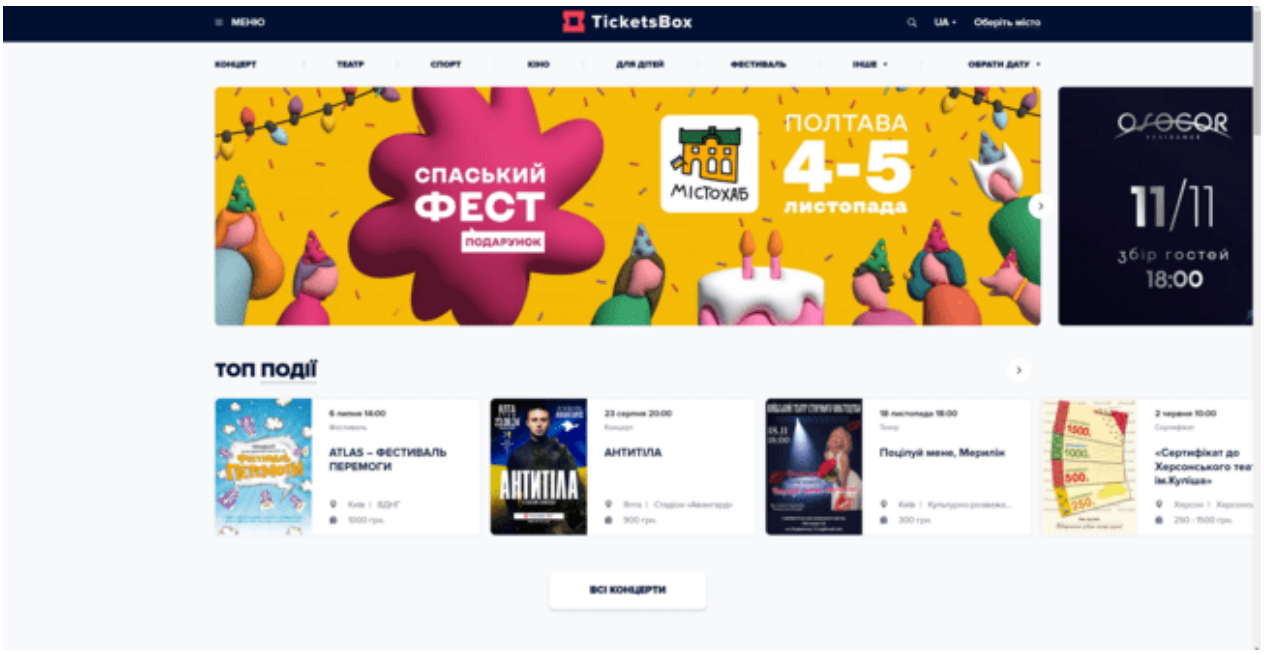

Рис. 1.5. Головна сторінка TicketsBox, скріншот 1

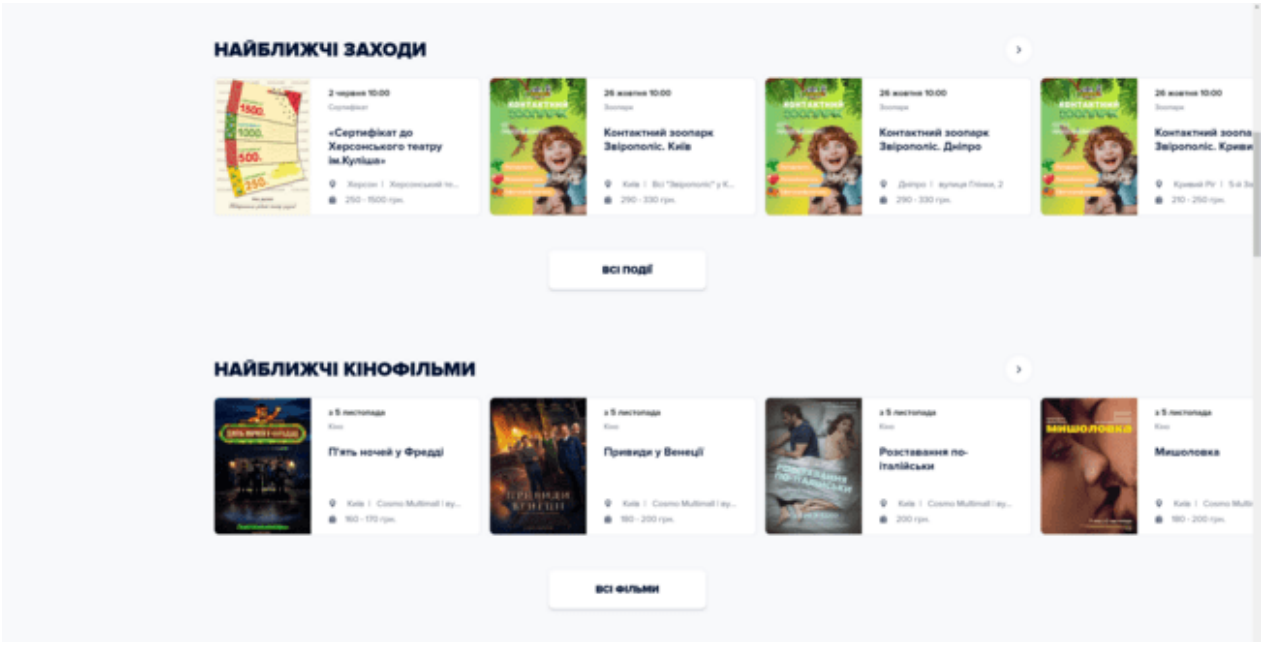

Рис. 1.6. Головна сторінка TicketsBox, скріншот 2

Сервіс вийшов на ринок Латвії, що свідчить про можливість розвитку та розширення завдяки хорошій конкурентній спроможності.

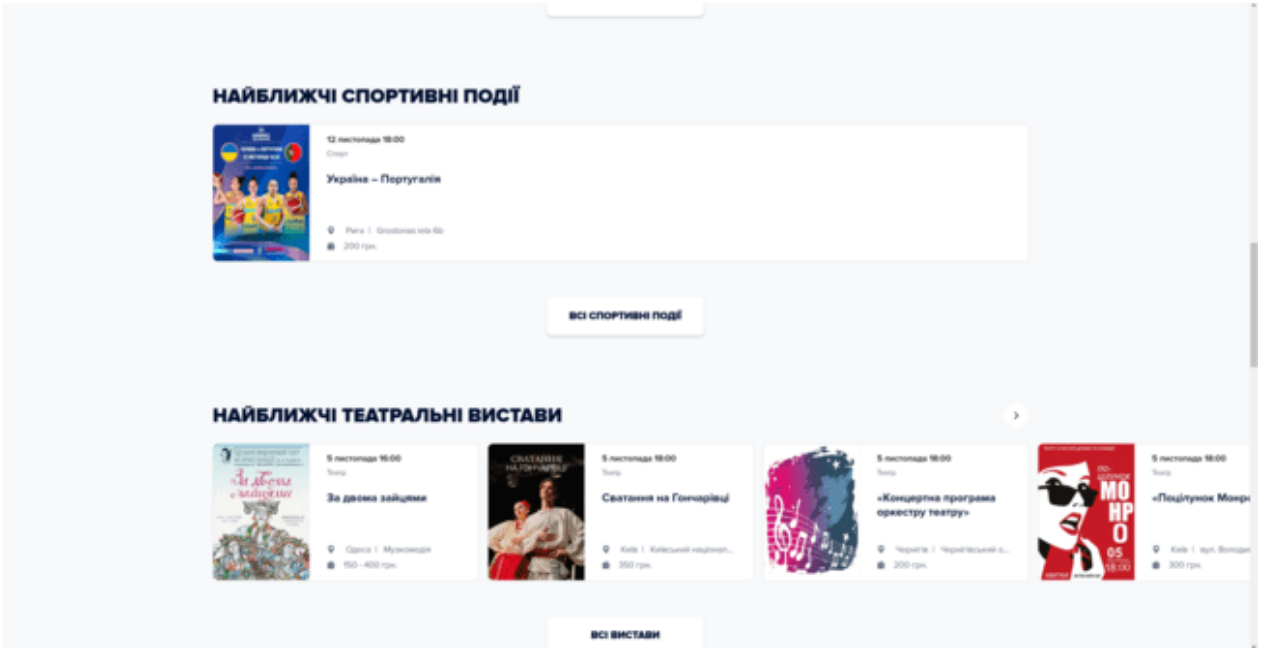

Рис. 1.7. Головна сторінка TicketsBox, скріншот 3

З рисунків видно, що багато простору на сайті залишається не використаним, і може справити враження, що на ресурсі нічого немає. Що ще гірше, зображення, зокрема афіші подій, не адаптовані під мобільні пристрої, що може ускладнити користування сайтом на смартфонах і планшетах. Це може вплинути на загальний вигляд та зручність використання ресурсу для користувачів, особливо тих, хто відвідує сайт за допомогою мобільних пристроїв.

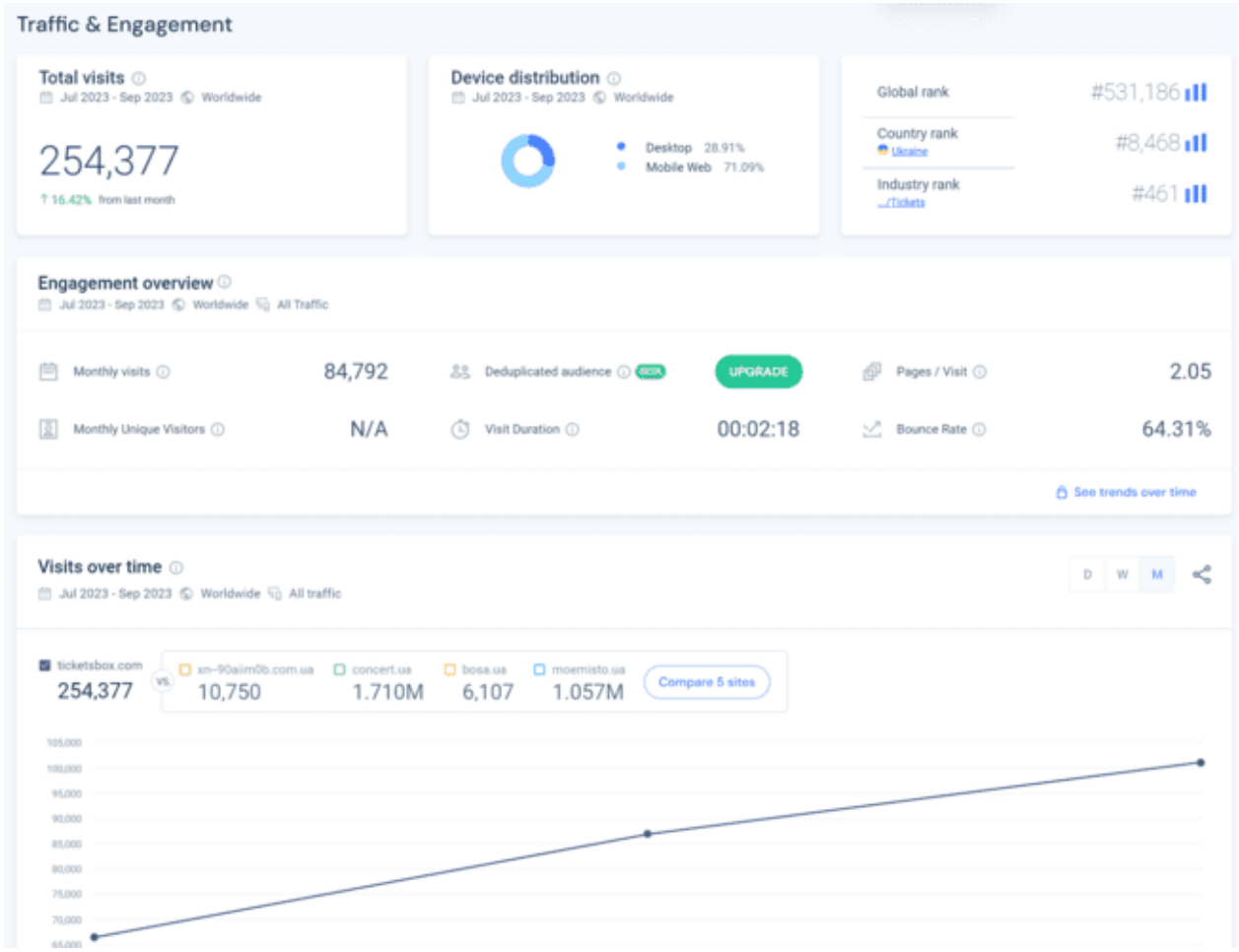

Рис. 1.8. Статистика зі сайту similarweb

Згідно з similarweb за останні півроку сайт відвідало майже 255 тисячі користувачів відкрило 2,05 веб-сторінки з середньою тривалістю перебування 2 хвилини 18 секунд. Короткий час перебування на сайті та високий відсоток людей, які покидають сайт після перегляду першої сторінки(Bounce Rate), що люди можуть не знайти потрібного контенту чи потрапляють на нього випадково. Також показники трафіку є доводі таки низькими.

Розглянемо веб-ресурс Karabas.

Сайт Karabas має привабливий інтерфейс (див. рис. 1.9 - 1.11) та приємне кольорове оформлення. Дизайн ресурсу не перевантажений надмірною інформацією, але він може здатися досить однотипним. Функціонал сайту схожий на інших конкурентів у цій галузі. Важливо відзначити, що Karabas не надає можливості

купувати квитки у фізичних офісах чи касах, що може бути недоліком для тих, хто віддає перевагу офлайн-покупкам.

Також сервіс активно присутній у соціальних мережах та популярних месенджерах, де має свої власні сторінки, що відкриває можливості для зв'язку з користувачами через ці платформи та розширює його присутність у Інтернеті.

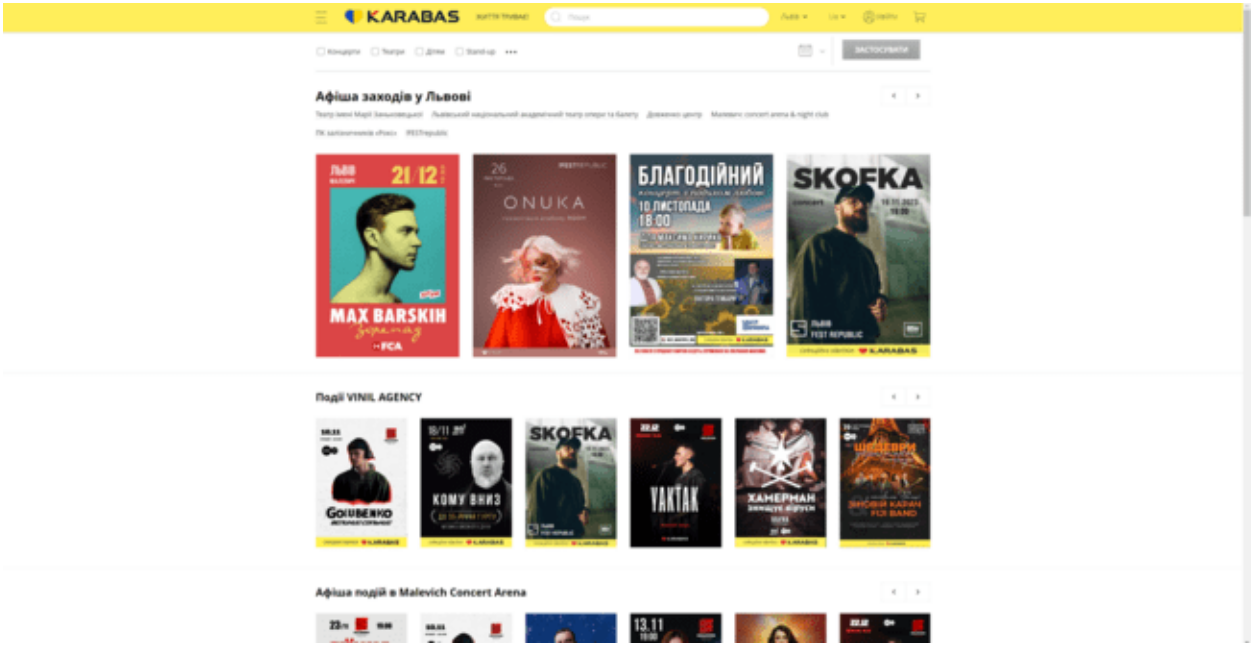

Рис. 1.9. Головне вікно Karabas, скріншот 1

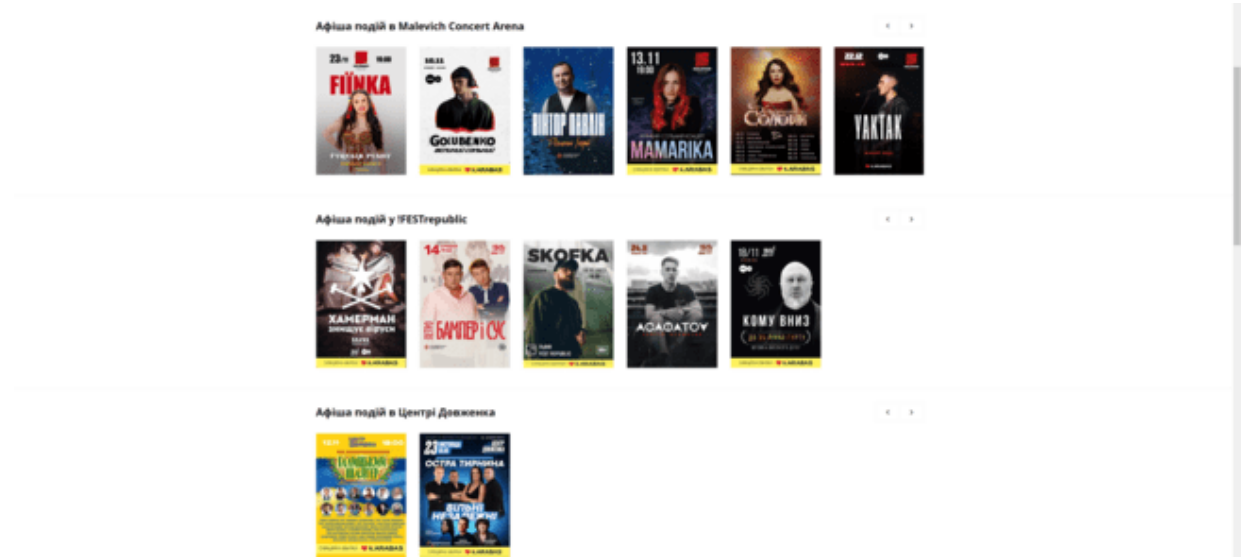

Рис. 1.10. Головне вікно Karabas, скріншот 2

Карабас пропонує обмежену кількість висококласних ексклюзивних шоу, більшість із представлених унікальних виконавців та артистів є регіональними або користуються популярністю лише у вузькому колі людей.

Рис. 1.11. Головне вікно Karabas, скріншот 3

Сайт чудово адаптований під мобільні пристрої і має посилання на додатки для IOS та Android. Однак на сторінці App Store неможливо знайти додаток, а за відгуками на Google Play Market здається, що він працює з помилками та має проблеми з локалізацією. Що означає, що додатки на мобільні пристрої не робочі.

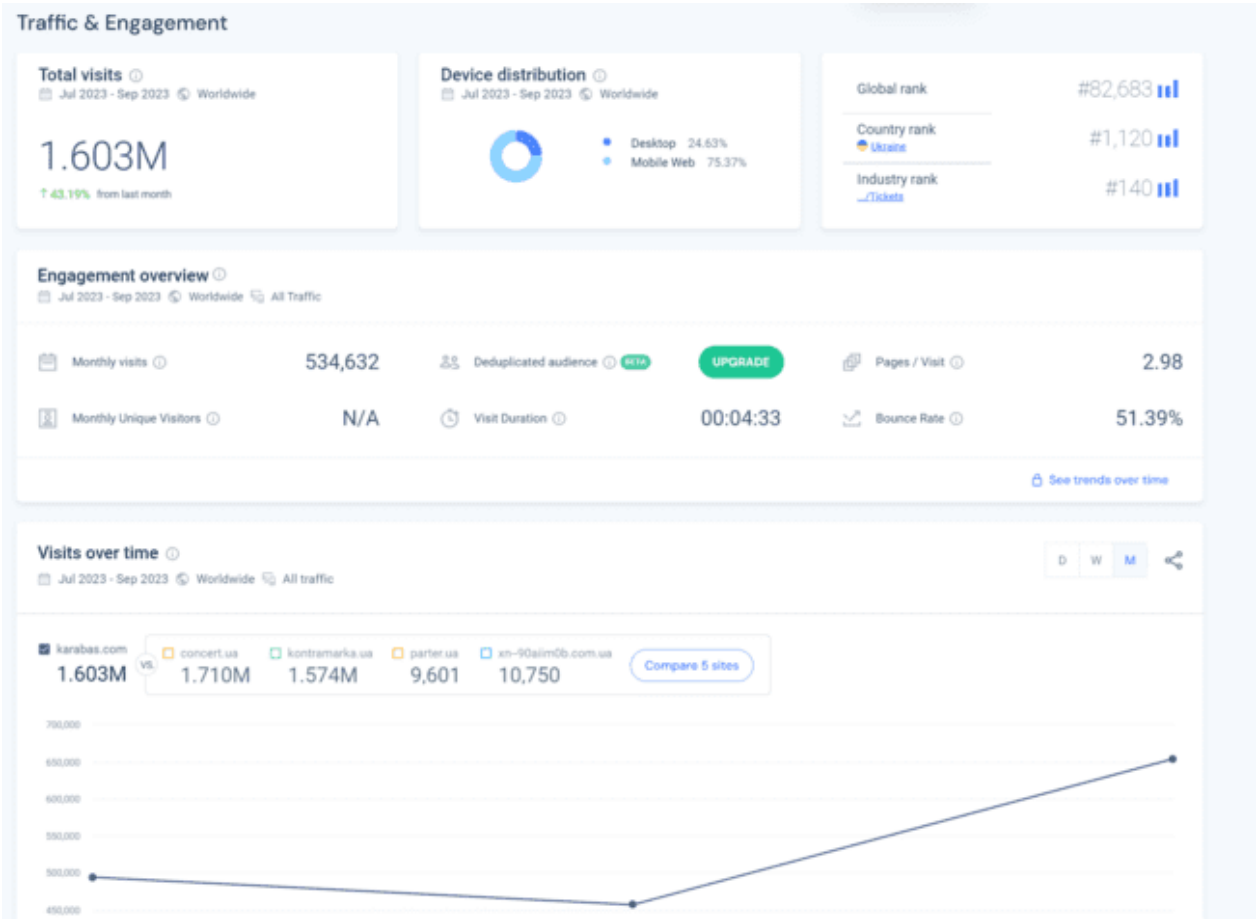

Рис. 1.12. Статистика зі сайту similarweb

За даними сайту similarweb за останні півроку, на сайт завітали майже 1 603 000 користувачів, які переглянули середньо 2,98 сторінки з тривалістю візиту 4 хвилини 33 секунди. Тривалість перебування на сайті виглядає підозріло, оскільки середня кількість переглянутих сторінок надто мала для такого тривалого візиту. Тому можна припустити, що дані були кориговані за допомогою певних методів, і на них можна не повністю спиратися.

Розглянемо сайт Everbrite.

Everbrite – це американський веб-сайт, спеціалізований на організації подій та продажу квитків. Цей сервіс був першопроходьцем серед великих квиткових операторів у США, тому завоював велику аудиторію. Крім того, Everbrite має відділи у різних країнах. Цей ресурс дозволяє користувачам переглядати, створювати та рекламувати місцеві події. Порівняно з українськими сайтами, його дизайн (див.

рисунки 1.13-1.15) трохи схожий на інші, і його функціональність на тому ж рівні, що й у конкурентів.

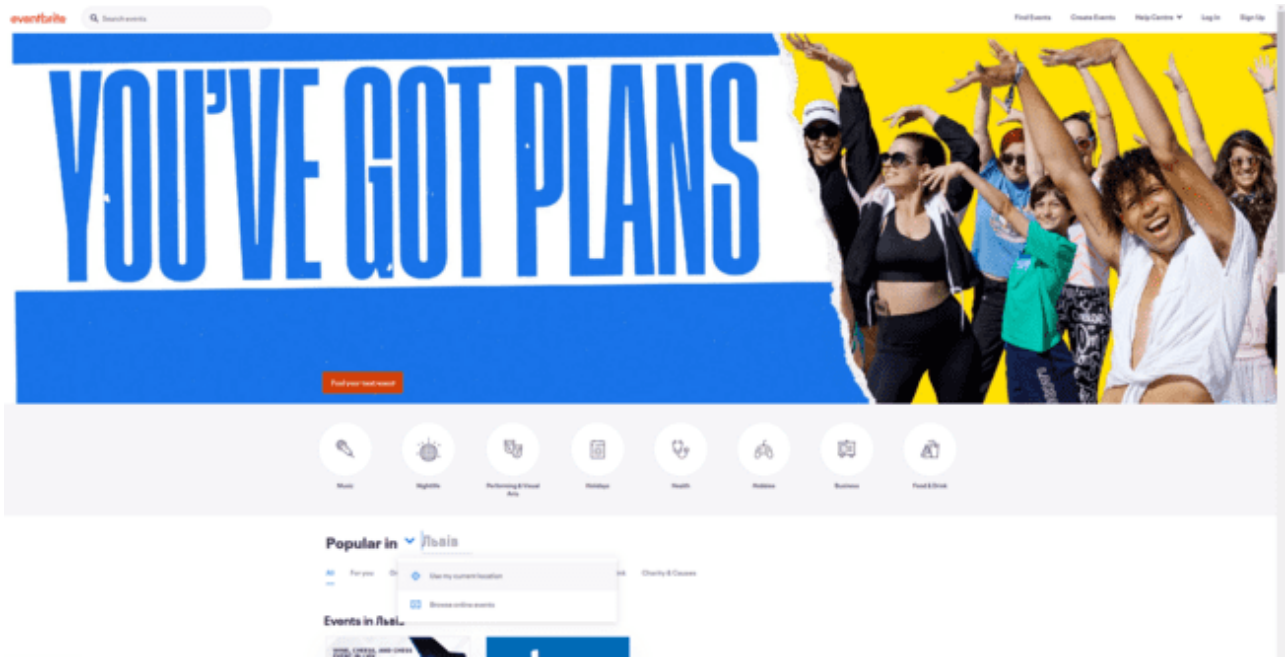

Рис. 1.13. Головна сторінка Everbrite, скріншот 1

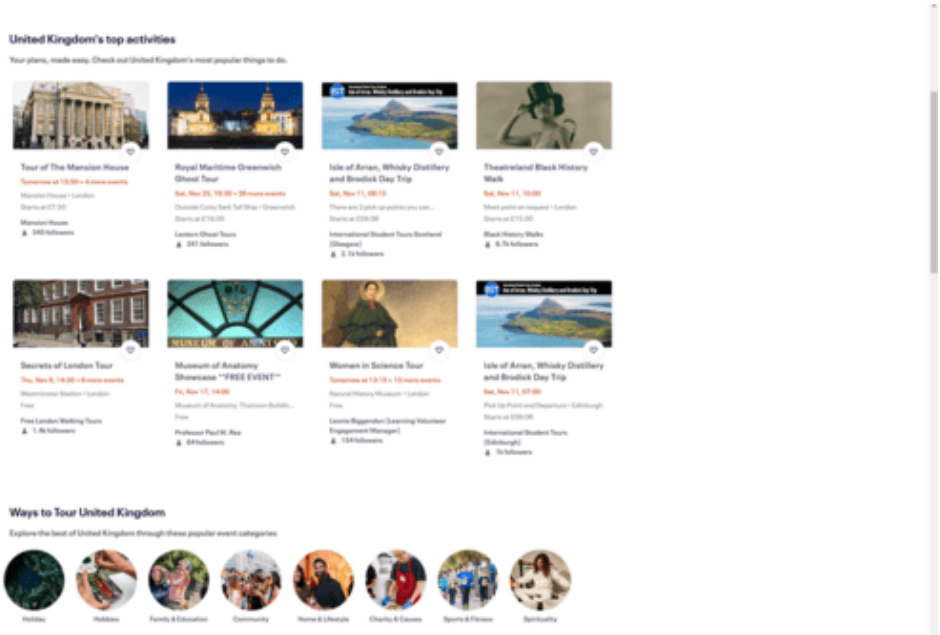

Рис. 1.14. Головна сторінка, скріншот 2

Помітна різниця у наповненні сайту, одразу кидаються у вічі фільтри. Це дуже зручно, адже пришвидшує пошук того, що цікавить. На Everbrite кольорова гамма підібрана дуже вдало.

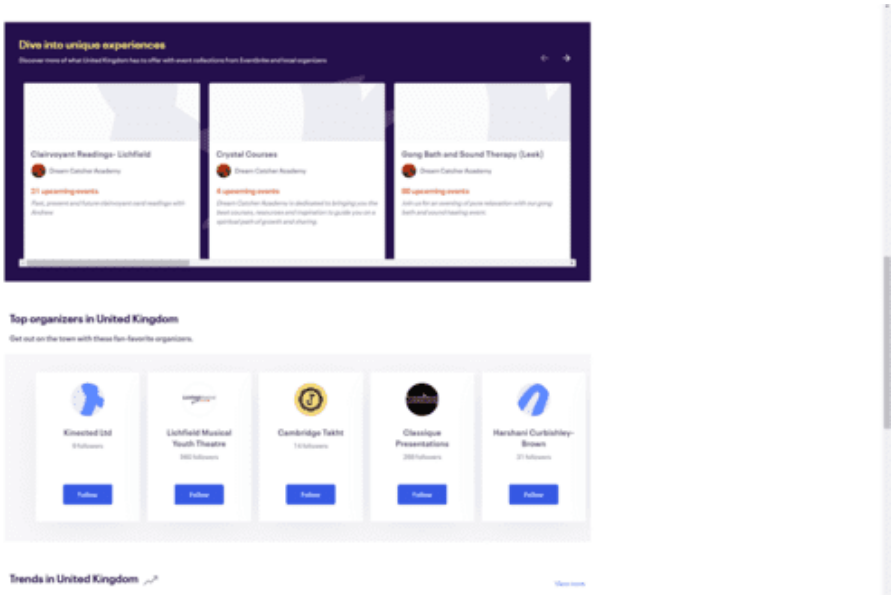

Рис. 1.15. Головна сторінка, скріншот 3

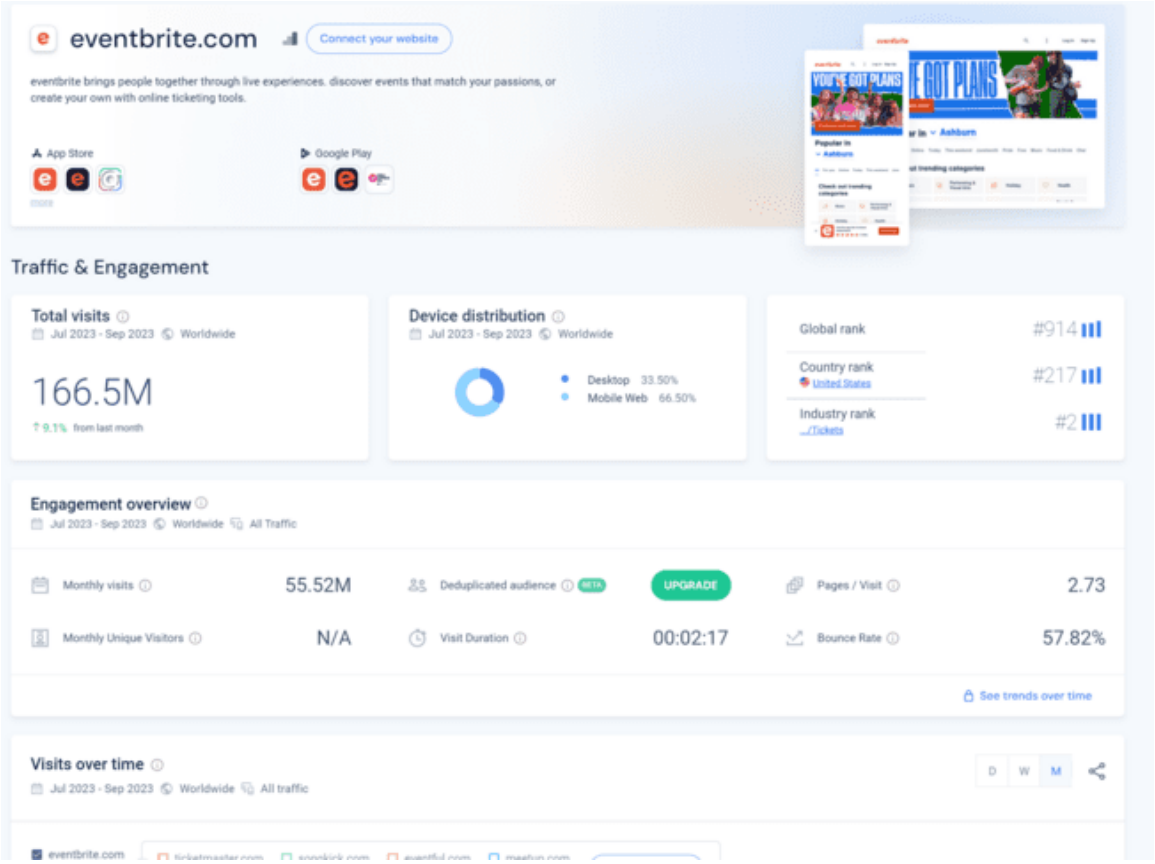

Рис. 1.16. Статистика зі сайту similarweb

Згідно з similarweb за останні півроку сайт відвідало майже 165,5 мільйонів користувачів відкрило 2,73 веб-сторінки з середньою тривалістю перебування 2 хвилини 17 секунд. Кількість відкритих сторінок та тривалість не надто велика, проте

значення відвідувачів вражає. Однією з переваг над конкурентами є можливість створювати власні події.

У результаті дослідження існуючих рішень, можливо скласти порівняльну таблицю 1.1.

Таблиця 1.1

Порівняння власної системи з аналогами

Характеристика	**Програми**			
Платформа	Concert.ua	TicketsBox	Karabas	Everbrite
ОС	Вебсайт	Вебсайт	Вебсайт	Вебсайт, Android, IOS
Уніфікація	+	+	+	+
Функціональність	середня	середня	середня	висока
Придатність до використання	середня	середня	середня	середня
Надійність	середня	середня	середня	середня
Продуктивність	середня	середня	середня	середня
Експлуатаційна придатність	висока	висока	висока	висока
Ціна	безкоштовно	безкоштовно	безкоштовно	безкоштовно

Висновок до першого розділу

У даному розділі проведено дослідження актуальності створення системи. Проведено аналіз основних конкурентів системи, виокремлюючи їх переваги та недоліки. Зокрема, було виявлено, що жодна з конкуруючих систем не надає можливості переглядати квитки у інших країнах, залишати відгук про відвіданий концерт чи створювати власні події, за винятком Everbrite. Також була визначена функціональність запланованої системи аналізу даних для опублікування афіш разом з її атрибутами, та були виявлені критичні функції, які потрібно реалізувати в першій версії.

РОЗДІЛ 2

Системний аналіз та моделювання предметної області

2.1. Аналіз мети функціонування системи

Головна ціль проекту – це розробка програмного забезпечення для опублікування афіш з вбудованою рекомендаційною системою. Користувачів системи можна умовно поділити на дві групи: споживачі та творці. Цей поділ є умовний, бо перехід від одного типу до іншого не потребує додаткових операцій. Реєстрація для обох є обов'язковою. Користувач «творець», який зацікавлений у публікації власної події, повинен мати доступ до сторінки створення події, сторінки аналітики та сторінки адміністрування. У свою чергу «споживач» повинен мати доступ до сторінки подій

Розділимо завдання на ряд цілей для кращого розуміння пріоритетів та термінів виконання.

Ціль 1. Створити систему авторизації користувача для збору історії відвіданих подій для побудови рекомендацій.

Ціль 2. Створити сторінку для створення події.

Ціль 3. Створити стрічку у якій відображаються існуючі події.

Ціль 4. Забезпечити роботу системи рекомендацій. Це є основний принцип застосунку, оскільки робота системи базується на ньому. Реалізація цього принципу повинна бути вищим пріоритетом..

Ціль 5. Організувати сховище даних для збереження усієї необхідної інформації.

Ціль 6. Забезпечити закрите тестування системи.

Ціль 7. Розширити доступ до системи. Розробити додатки для IOS та Android.

2.2. Моделювання вимог системи та ризики

Вимоги до програмного забезпечення (ПЗ) представляють собою набір визначень та очікувань, які відносяться до властивостей, функцій та якості програми, яку планують розробити або вже знаходяться в стадії розробки. Ці вимоги визначаються та зафіксовані під час аналізу у специфікаціях, діаграмах прецедентів

та інших артефактах, пов'язаних із процесом аналізу та розробки програмного продукту.

Перелік основних вимог до розроблюваної системи включає бізнес-вимоги, функціональні та нефункціональні вимоги, а також вимоги користувачів.

Бізнес-вимоги представляють собою ключові обов'язки та функції, які бізнес-структури очікують від системи аналізу для публікації афіш. Ці вимоги є фундаментальними для успішного вирішення проблеми та досягнення мети. Деякі з основних бізнес-вимог для цієї системи включають в себе:

- Забезпечити можливість перегляду афіш;
- Забезпечити швидкодію системи;
- Створити рекомендаційну систему;
- Забезпечити можливість.

Функціональні вимоги - це опис того, що програмне забезпечення має робити[2]. Вони представляють собою конкретні функції, операції та послуги, які програма повинна надавати користувачам або іншим системам. Ці вимоги визначають, як система повинна реагувати на певні вхідні дані або події, які функції повинні бути доступні для користувачів та як система повинна вести себе у певних умовах. До функціональних вимог системи аналізу для публікації афіш можна віднести:

- Авторизація;
- Створення персонального облікового засобу;
- Можливість перегляду афіш;
- Можливість створення подій;
- Коментування на сторінці події;
- Можливість перегляду афіш;
- Модерування події;
- Купівля/продаж білетів;
- Надсилання списку рекомендацій.

Нефункціональні вимоги - це вимоги, які не стосуються конкретних функцій чи можливостей програмного забезпечення, але визначають характеристики, якості, надійність, продуктивність, безпеку та інші аспекти системи. Ці вимоги описують якості програмного продукту та обмеження, які повинні бути дотримані.

Основні категорії нефункціональних вимог включають:

1. Продуктивність: Вимоги до продуктивності визначають швидкість, завантаженість та ефективність роботи системи.
2. Надійність: Ці вимоги стосуються стабільності та надійності системи. Вони можуть включати в себе вимоги до доступності (uptime), відмовостійкості та можливості відновлення системи після збоїв.
3. Безпека: Вимоги до безпеки визначають рівень захищеності системи від несанкціонованого доступу, втрати даних та інших загроз безпеці.
4. Інтерфейс та взаємодія з користувачем: Ці вимоги описують зручність та ефективність інтерфейсів користувача, а також вимоги до документації та підтримки користувачів.
5. Сумісність та масштабованість: Вимоги до сумісності визначають, як програмне забезпечення взаємодіє з іншими системами чи платформами. Масштабованість визначає, як система може розширюватися або зменшуватися в розмірі чи обсязі відповідно до потреб користувачів.
6. Легкість супроводу (Maintainability): Ця категорія визначає, наскільки легко систему можна підтримувати, виправляти помилки, розширювати та покращувати.

Нефункціональні вимоги грають важливу роль у процесі розробки програмного забезпечення, оскільки вони визначають стандарти якості та надійності, які повинні бути враховані під час розробки та тестування системи.

Користувацькі вимоги - це специфікації та очікування, які мають користувачі щодо функцій і характеристик програмного забезпечення. Ці вимоги визначають те, як користувачі очікують, що програмне забезпечення буде працювати та взаємодіяти з ними. Користувацькі вимоги повністю повторюють бізнес вимоги.

Створення вимог веде до створення UML-діаграми варіантів використання (Use Case Diagram), яка служить для відображення структурної схеми функціональних та користувацьких вимог. Цю діаграму можна побачити на рисунку 2.1. Це графічне представлення взаємодій між різними акторами та системою, яке допомагає зрозуміти, як користувачі взаємодіють з системою та які функції вони можуть виконувати[3].

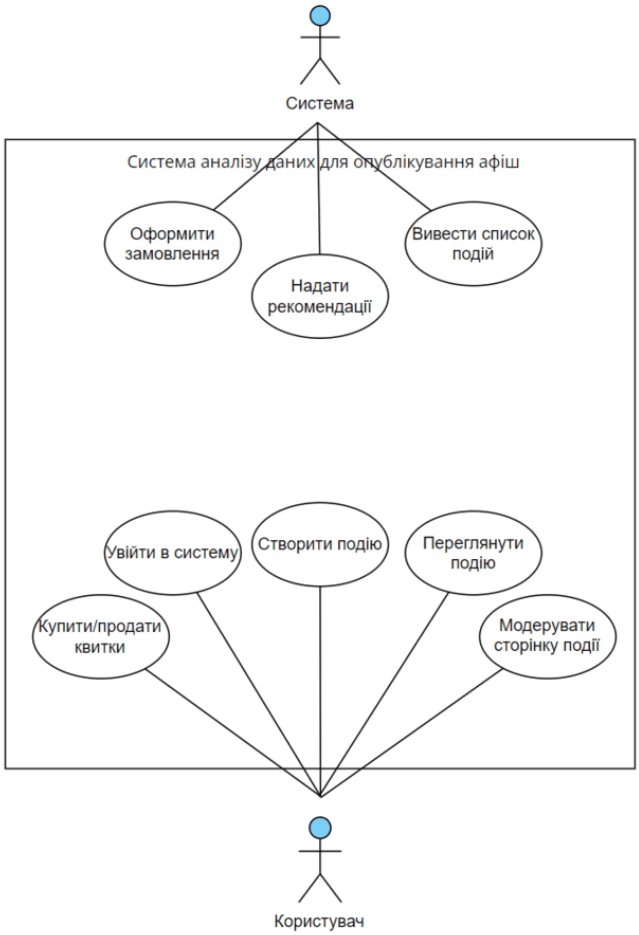

Рис. 2.1. Структурна схема поведінки системи

Схема побудована на основі функціональних вимог зі сторони актора «Користувач». Ця сутність має вісім варіантів взаємодії із системою. Між варіантами використання і Користувачем асоціативні зв’язки.

Під час роботи над будь-яким проектом існують ризики, які можуть призвести до невдачі, відставання від графіку або інших проблем. У розробці програмного забезпечення велика увага приділяється управлінню ризиками.

Ризики при розробці програмного забезпечення можна класифікувати за наслідками на дві основні групи: чисті та спекулятивні. Чисті ризики зазвичай відзначаються лише можливими втратами для підприємства. У разі спекулятивних ризиків можливі як додатковий прибуток, так і додаткові втрати. Управління ризиками включає в себе розпізнавання, аналіз та планування заходів щодо зменшення ймовірності негативних наслідків та максимізації можливостей отримання позитивних результатів.

Спочатку розглянемо спекулятивні ризики:

Ризик фінансового обмеження – більшість програмного забезпечення є з відкритим вихідним кодом, що означає – всі витрати йду саме на розробку. Таким чином потрібно добре спланувати бажаний темп розробки та поріг витрат.

Ризик зміни кон'юнктури ринку виникає тоді, коли умови, які були передбачені на етапі планування проекту, відрізняються від фактичної ситуації на ринку. Це може включати фактори, які не були передбачені під час початкового аналізу. Однак у випадку, коли економічна ситуація на ринку не має впливу на розроблюваний проект, цей аспект не вважається ризиком, що відображає нашу ситуацію.

Розглянемо чисті ризики.

Ризик поганої взаємодії між замовником і виконавцем – для вирішення конфліктних ситуацій можна створити контактний центр.

Ризик управління проектом – відсутність навичок проектного менеджменту у людини, яка відповідає за проект. Рішення – побудова процесів, які задовільнили усіх учасників.

Ризик планування – цей ризик є актуальним і повністю його уникнути не можливо. Хорошим підходом вважається розбиття великих робіт на менші, таким чином спланувати дрібні процеси легше, відповідно помилок менше.

Ризик відсутності системи контролю – цей ризик можна уникнути формуючи чіткі вимоги, які зрозумілі усім. Так, усім буде зрозуміло чи робота виконана чи ні.

2.3. Моделювання об’єктів системи

Визначимо головні класи для системи, з їх методами та атрибутами.

Система аналізу даних для публікації афіш буде основний клас, який розпочинатиму роботу програми і матиме наступні атрибути: Інтерфейс системи та Сервер.

Інтерфейсом системи буде програма з атрибутами: «Список подій». Також буде два методи: «Обробити подання подій»(відправити запит на сервер для отримання рекомендацій) та «Обробити створення нової події»(створити нову подію).

Сервер – основний об’єкт класу, який обробляє дані, керує підпроцесами. Атрибути наступні: «Список подій», «Рекомендаційна система». Методи класу: «Оновлення даних»(створення чи оновлення записів у базі даних), «Сортування вибірки»(побудова рекомендацій), «Оновити дані».

Подія – подія створена користувачем

Користувач – може створювати та переглядати події.

Сформований опис класів подано на наступній сторінці в таблиці 2.1.

Таблиця 2.1

Опис об’єктів класу

Класи об’єктів		**Атрибути класу**		**Методи класу**	
Назва класу	*Призначення класу*	*Назва атрибута*	*Зміст атрибута*	*Назва методу*	*Зміст дії*
Система	Клас, який реалізовує всі функції	Інтерфейс	Інтерфейс користувача	Отримати дані	Отримати дані
		Користувач	Користувач системи	Ввести дані	Ввести дані
		Сервер	Сервер	Обробка даних	Передати дані серверу
Інтерфейс системи	Безпосередня робота з користувачем	Список подій	Список подій	Вивести список подій	Вивести список подій

Продовження таблиці 2.1

Класи об'єктів		**Атрибути класу**		**Методи класу**	
Назва класу	*Призначення класу*	*Назва атрибута*	*Зміст атрибута*	*Назва методу*	*Зміст дії*
Інтерфейс системи	Безпосередня робота з користувачем			Обробити створення нової події	Створення запису
Сервер	Обробляє дані	Список подій	Список подій	Оновити дані	Оновлення системи
				Опрацювати дані	Опрацювати дані
		Рекомендаційна система	Подання даних	Сортування вибірки	Повернення інформації
Подія	Опис події	Місце, час, учасники, автор, тип подій, фотографії	Атрибути для опису події		
Кори-стувач	Опис користувача	ПІБ, email, пароль, логін	Атрибути для опису користувач	Створити подію	Створити нову подію
				Переглянути список подій	Переглянути список подій

Система аналізу даних для опублікування афіш та Інтерфейс, Інтерфейс та Сервер, Сервер та Система – основні відношення між класами. Їх опис поданий на наступній сторінці у таблиці 2.2.

Таблиця 2.2.

Опис відношення між класами

Назва відношення	Класи, між якими визначено відношення		Вид відношення	Розмірність відношення
Ініціалізація	Система	Інтерфейс	Агрегація	1-1
Оброблення подій інтерфейсу	Інтерфейс	Сервер	Агрегація	1-1
Оновлення рекомендацій	Сервер	Система	Агрегація	1-1

Діаграма класів була створена на основі даних з таблиць 2.1 та 2.2 для ідентифікації об'єктів, що втілюють усі основні процеси.

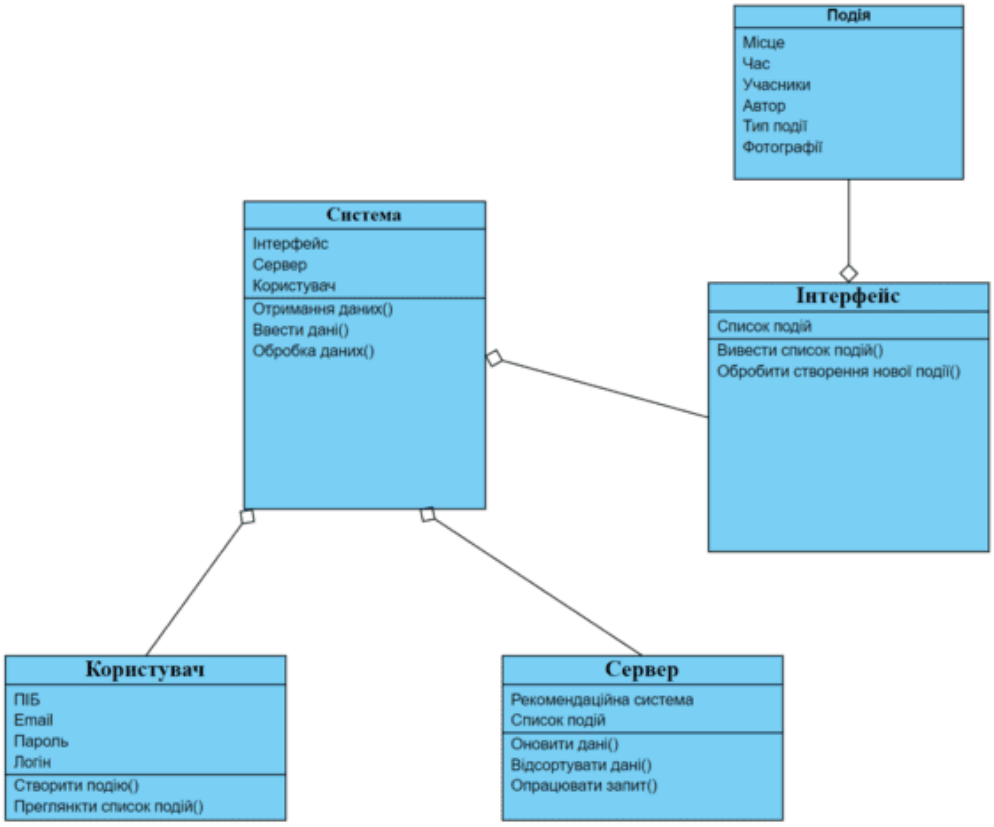

Рис. 2.2. Діаграма класів

2.4. Моделювання процесів системи

Тепер, після визначення основних класів, слід визначити процеси системи. Моделювання процесів системи - це метод відображення реальних або імовірних процесів для аналізу, оптимізації та вдосконалення цих процесів. Моделювання системи може бути використане в багатьох сферах, таких як бізнес, інженерія, наука та інші галузі, де потрібно зрозуміти та оптимізувати дії, взаємодії та ресурси системи.

Незалежно від обставин, послідовність дій завжди однакова: система аналізу даних для опублікування афіш запускає інтерфейс користувача, інтерфейс у свою чергу очікує запиту від користувача, отримавши його, запит передається серверу. Також, сервер отримуючи нові дані про користувача, оновлює рекомендаційну систему.

Усі етапи процесу представлені у вигляді структурної схеми функціонування системи, яка зображена на рисунку 2.4.

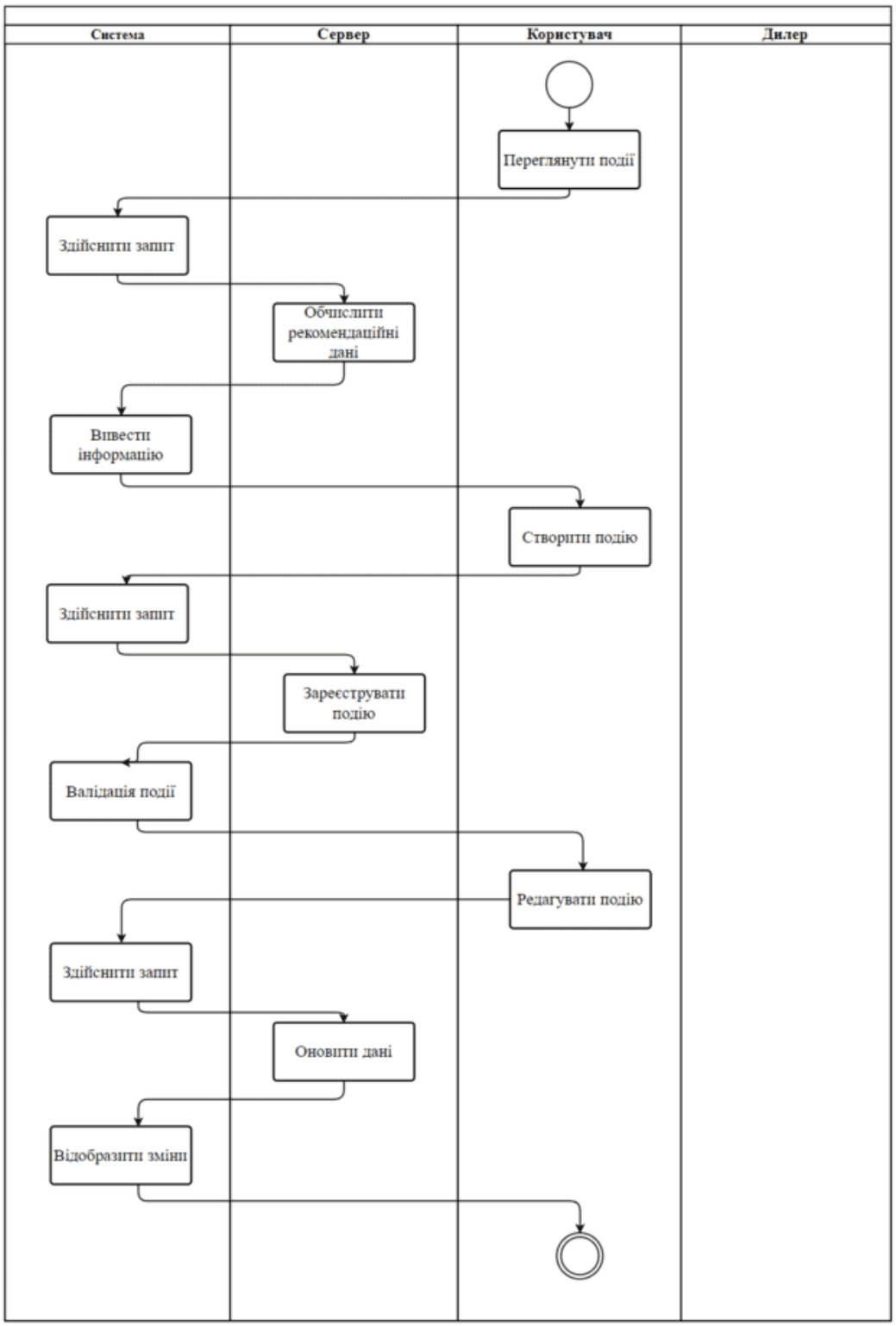

Рис. 2.4. Структурна схема процесу функціонування системи аналізу зображень

Висновок до другого розділу

У процесі розробки моделі системних вимог визначено ключові категорії: бізнес-вимоги, функціональні вимоги, нефункціональні вимоги та вимоги користувачів. Глибокий аналіз можливих ризиків, які можуть виникнути під час впровадження проекту, допомагає зрозуміти можливі загрози, збитки та визначити стратегії їх уникнення чи зменшення. Це важливий етап, який дозволив врахувати можливі труднощі та забезпечити більш ефективний процес розробки та впровадження системи.

Системні об'єкти моделюються у форматі UML-діаграм, що визначає ключові класи системи, їх атрибути та методи, і подає їх на діаграмі класів. Це дозволяє чітко відобразити структуру системи та її складові елементи.

Моделювання основних процесів системи сприяє кращому розумінню функціональності класів, їх взаємозв'язків та послідовності виконання операцій. Ці процеси зазвичай відображаються на схемах, які показують послідовність операцій та їх взаємозв'язки.

РОЗДІЛ 3

Розроблення проекту розв’язання задачі

3.1. Формулювання та обґрунтування задачі

Метою створення проекту є надання зручного сервісу для опублікування афіш. Для невеликих творців це шанс знайти власну аудиторію, а для споживачів – знайти щось цікаве та нове за допомогою зручних рекомендацій. Метою самої системи є аналіз активності користувача і побудова списку рекомендацій.

Цільова аудиторія – це звичайні користувачі будь-якого віку, у яких є бажання відати концерт, стендап чи театральну виставу; та власне організатори цих подій. Це означає, що сервіси маж бути доволі таки популярним. Розглянемо типові способи використання сервісу: організація невеликих виступів, квартирників; збори, події у клубах за інтересами; організація професійних вебінарів, лекцій для кар’єрного розвитку.

У світі де все має свою причину і наслідок, неможливо створити продукт, який би не ніс зміни у життя людини. Далі розглянемо потенційні ефекти від розробки системи аналізу для опублікування афіш.

Часовий ефект – зменшення витрат часу користувачів на пошук подій.

Фінансовий ефект – зменшиться витрати на маркетинг афіш.

Економічний ефект – збільшення відвідувачів подій, що приводить до фінансування креативної сфери та споруд які обслуговують її.

Культурно-соціальний ефект – привернення уваги людей до сфери культури.

Екологічний ефект – не впливає.

Соціальний ефект – сервіс спонукає відвідувати нові події, де є імовірність зав’язати нові знайомства, розширити своє коло спілкування.

3.2. Побудова моделі для розв’язання задачі

Вирішення задачі починається із форму проблеми, що ми і зробили в попередньому розділі. Задачею є побудова системи, що буде рекомендувати події на основі активності користувача.

Далі слід побудувати модель для заданої системи. Інформаційно-математична модель є наступною:

1. Отримання критеріїв для наступного опрацювання;
2. Опрацювання даних для подальшого аналізу;
3. Формування системи рекомендацій;
4. Надсилання рекомендацій користувачеві.

Розглянемо детальніше рекомендаційну систему задля розв'язання задачі та поставленої мети.

Нехай U = { U_1, U_2, …, U_n} – множина векторів профілів користувачів, G = {G_1, G_2, …, G_m} – масив груп параметрів, G_i = {U_1G_i, U_2G_i, …, U_kG_i} – множина профілів користувачів для групи Gi. Необхідно здійснити прогноз рекомендацій для груп користувачів C_{gi} = Predict(G_i).

Методи кластеризації використовуються для пошуку груп подібних користувачів. У матриці "користувач-подія" значна кількість елементів рівна нулю. Кількість ненульових компонентів становить менше 10% від загальної кількості компонентів у даній матриці. Отже, для ефективної кластеризації користувачів доцільно використовувати характеристики, задані самими користувачами. Основні атрибути включають розмір події, тип події, місце проведення та час. Місце та час представляють числові атрибути, тоді як розмір та тип події є категоріальними атрибутами [4].

Нехай вектор рейтингів для користувача і-того визначається за допомогою формули 3.1.

$$U_i = (u_{1i}, u_{2i}, \dots, u_{mi}), \tag{3.1}$$

де u_{ji} – рейтингова оцінка j-го події і-тим користувачем.

Розширимо цей вектор, додавши технологічні атрибути користувача до формули 3.2.

$$U_i^{ext} = (u_{1i}, u_{2i}, \dots, u_{mi}, d_{1i}, d_{2i}, d_{3i}, d_{4i}), \tag{3.2}$$

де $d_{1i}, d_{2i}, d_{3i}, d_{4i}$ – категоріальні атрибути користувачів.

Для спрощення опису методу позначатимемо вектор U_i^{ext} за допомогою вектора $X_i = \{x_{1i}, x_{2i}, \dots, x_{ni}\}$.

Отже, ми отримуємо комбінований вектор характеристик профілю, який включає в себе як числові, так і категоріальні значення.

Кластеризація мішаних векторів параметрів користувачів здійснюється за допомогою методу мішаної кластеризації, який ґрунтується на розрахунку щільності розміщення мішаних векторів профілів параметрів. Цей метод визначає кількість та положення центрів кластерів. Щільність вимірюється як кількість векторів параметрів користувачів, які перебувають в околі з радіусом d_c навколо кожного користувача, згідно формул 3.3 та 3.4.

$$\rho_i = \sum_{j=1}^{N} f(d_{ij} - d_c), \tag{3.3}$$

де d_{ij} – відстань між і-м та j-м векторами параметрів користувачів; d_c – порогове значення; N – кількість користувачів.

$$f(x) = \begin{cases} 1, & x = d_{ij} - d_c \le 0 \\ 0, & x = d_{ij} - d_c > 0 \end{cases} \tag{3.4}$$

3.3. Вибір та обґрунтування методів розв’язання задачі

Система рекомендацій – це підклас інформаційної системи, яка визначає рейтинг об'єктів, які можуть бути цікаві для користувача, використовуючи інформацію з його профілю.

Для створення рекомендаційних систем використовуються дві основні стратегії: фільтрація вмісту і колаборативна фільтрація. Під час колаборативної фільтрації формуються профілі користувачів і об'єктів.

При фільтрації вмісту враховується інформація про оцінки, дії, зроблені користувачами у минулому. У цьому випадку не має значення, з якими типами об'єктів працює система, оскільки можна враховувати навіть неявні характеристики, що ускладнюють створення профілю. Однак основна проблема цього типу рекомендаційних систем полягає у "холодному старті": відсутність даних про користувачів чи об'єкти, які тільки що з'явилися у системі.

Один із існуючих методів це мішаної категоріально-числової кластеризації для пошуку груп користувачів. Далі невеликий опис основних кроків цього підходу. [40]

На першому етапі відбувається кластеризація векторів профілів користувачів за їхніми категоріальними характеристиками, утворюючи групи користувачів зі схожими параметрами. На наступному етапі проводиться числова кластеризація векторів профілів користувачів, які містять рейтингові оцінки предметів. Прогнозування рекомендацій для отриманих груп може бути реалізоване як класичними методами колаборативного прогнозування користувач-користувач або предмет-предмет, а також методами прогнозування в групі, такими як метод адитивної корисності або метод мультиплікативної корисності.

Використовуючи цей метод, користувача можна віднести до певної категорії, а згодом, коли даних ставатиме все більше, значення векторів можна міняти. Слід також врахувати, що користувач протягом певного часу переглядає події з однієї категорії, що може створити враження для споживача про брак різноманіття. Таким чином слід ввести випадкову складову, яка буде підкидати формат подій, які потенційно могли б приваби користувача своєю новизною.

3.4. Розроблення алгоритмів розв’язання задачі.

Далі планується розробити алгоритм, використовуючи попередньо створену модель. Розглянемо загальний процес створення системи аналізу для публікації афіш.

У даній системі аналізу існують два типи користувачів з власними ролями:

- Споживач.
- Творець/організатор.

Під час користування системою споживач має можливість переглядати панель рекомендацій, де надається інформація про подію та кроки, які необхідно виконати для купівлі квитка. Клієнт може використовувати мапу, на якій відображені події, а також переглядати список всіх доступних подій у будь-якому місті.

Якщо споживач бажає відвідати подію, йому потрібно пройти етапи, передбачені організатором, наприклад, заповнити форму, підтвердити вік, здійснити оплату тощо.

Після цього він отримує власний квиток, який підтверджує, що всі необхідні дії було виконано.

Після проведеної події користувач може залишити відгук чи поділитися фотографіями, якщо його білет було використано.

Творець або організатор може створювати нові події, додавати афішу, фотографії та опис. Він також має доступ до звітів про активність на сторінці події та може бачити кількість проданих білетів.

Розроблена система для публікації афіш представлена у вигляді веб-ресурсу, який доступний для всіх користувачів з доступом до Інтернету. Основною метою цієї системи є надання користувачам можливості переглядати існуючі події та створювати власні, створюючи платформу для культурних подій. Ця система розроблена за допомогою передових технологій веб-розробки, таких як Java, Spring та Angular.

3.5. Вибір і обґрунтування засобів розроблення проекту

Враховуючи простоту взаємодії клієнта з веб-сайтами, легкість доступу та можливість переглядати їх з будь-яких пристроїв, вирішено розробляти саме веб-сторінку. Крім того, важливо звернути увагу на стабільність системи, оскільки під час старту продажу квитків може надійти велика кількість запитів в один момент. Тому вирішено використовувати комбінацію технологій Java та Angular для реалізації проекту для розробки веб-сайту, що забезпечить надійну та швидку роботу системи.

Для front-end частини вибрано наступні технології: TypeScript, Angular, Bootstrap, HTML, CSS, JavaScript[5][6][7].

Для back-end частини обрано: Java, Junit, REST, Spring Data, Spring MVC, Spring Security, Spring Boot, Maven, MySQL.

Також розглянемо GitLab, як технологію для спрощення процесу розробки.

Розглянемо кожну технологію окремо.

TypeScript – це мова програмування зі статичною типізацією, розроблена компанією Microsoft та випущена у 2012 році. Ця мова була створена для розширення можливостей JavaScript. TypeScript спроектована для розробки великих програмних

застосунків. Код, написаний на TypeScript, може бути перетворений (транскомпільований) в JavaScript, що дозволяє виконувати його в будь-якому місці, де працює JavaScript[8][9].

Angular – це фреймворк для розробки веб-додатків з відкритим кодом, написаний на TypeScript. Він розробляється під керівництвом Angular Team у компанії Google, а також залучає спільноту приватних розробників та корпорацій. Код, написаний на Angular, має модульну структуру, що є великою перевагою під час розробки. Бібліотеки Angular охоплюють повний цикл написання коду. Фреймворк генерує структуру та код за допомогою команд, надає готові шаблони та велику кількість різноманітних анімацій. Усе це робить Angular одним із найкращих frontend фреймворків[10][11][12].

Bootstrap – це безкоштовний набір відкритих шаблонів, інструментів та прикладів, призначений для створення веб-додатків. Він містить готові CSS та HTML шаблони для оформлення різних елементів інтерфейсу, таких як кнопки, форми, навігація тощо. Bootstrap полегшує розробку динамічних веб-додатків та веб-сайтів. У порівнянні з аналогічними системами, такими як Foundation, UIkit, Semantic UI, InK тощо, Bootstrap є найпопулярнішим фреймворком. Він дозволяє верстати сайти значно швидше, ніж за допомогою "чистого" CSS та JavaScript, і при цьому надає адаптивний дизайн за замовчуванням та є легким у вивченні[13].

Розглянемо технології для розробки back-end частини.

Java – це об'єктно-орієнтована, компільована мова програмування, яку випустила компанія "Sun Microsystems" у 1995 році. Після поглиблення "Sun Microsystems" у 2009 році мовою займається компанія "Oracle". Програми, написані на мові Java, спочатку компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною (JVM) для конкретної платформи[14][15].

«Oracle» надає компілятор (javac) та віртуальну машину (JVM) під ліцензією GNU General Public License.

Мова Java значно взяла синтаксис із мов програмування C і C++. Базові типи, які їх ще називають примітивами, були запозичені з мови C++, але були модифіковані та

змінені для уникнення можливих конфліктів, які можуть виникнути через помилки програміста. Java полегшила процес розробки об'єктно-орієнтованих програм, усунувши деякі складні дії, такі як очищення пам'яті, які тепер виконує віртуальна машина Java (JVM).

Основна ідея розвитку Java полягала в тому, щоб зробити її мовою, яка була б незалежною від платформи, що призводило до того, що вона має менше низькорівневих можливостей для взаємодії з апаратним забезпеченням. Це обмеження може призвести до зниження швидкості роботи програм. Однак у випадках, коли потрібна більша продуктивність, Java може викликати підпрограми, написані на інших мовах програмування.

Spring Framework – це фреймворк з відкритим вихідним кодом та контейнери, які підтримують інверсію управління для платформи Java. Spring Framework може бути використаний будь-яким додатком Java завдяки своїм основним особливостям. Також існують розширення для розробки веб-додатків на платформі Java EE. Spring розвивається як окремий продукт і функціонує незалежно від Java EE та її спільноти. Незважаючи на це, Spring Framework не нав'язує жодної конкретної моделі програмування і став популярним у спільноті Java як альтернатива чи навіть доповнення моделі Enterprise JavaBean (EJB)[16][17].

Spring Model-View-Controller (MVC) – це фреймворк, спроектований навколо DispatcherServlet, який відповідає за отримання та перенаправлення запитів до контролера, який може їх обробити. Ця технологія реалізована за допомогою слабкого зв'язування готових компонентів. Суть фреймворку, як і патерну MVC, полягає в розділенні обов'язків між різними класами компонентів, забезпечуючи при цьому вільне зв'язування між ними[18].

Основні типи класів у Spring MVC:

- модель (model) об’єднує в собі дані, які передаються між сервером і клієнтом;
- вигляд (view) відповідає за відображення моделі, переважно готовий шаблон HTML у якому записуються дані з моделей;

- контролер (controller).

Spring Data - це проект, розроблений командою Spring Framework, який призначений для забезпечення спрощеного доступу до даних з можливістю використання різних джерел даних, не втрачаючи при цьому характеристики вихідного джерела даних. Цей проект дозволяє використовувати технології доступу до даних для реляційних та нереляційних баз даних, а також моделі розподілених обчислень та хмарні дата-сервіси. Він містить багато підпроектів, які призначені для роботи з кожною специфічною базою даних. Spring Data був розроблений спільно з численними компаніями та програмістами, які стоять за цими технологіями[19].

Spring Security – це потужний і гнучкий фреймворк для автентифікації і контролю доступу, який є стандартом для додатків на основі Spring. Spring Security - це фреймворк, що забезпечує автентифікацію та авторизацію для Java-програм. Його справжня сила полягає в легкості налаштування, що дозволяє задовольнити різноманітні вимоги проектів. Цей фреймворк зосереджується на забезпеченні безпеки програм шляхом надання можливостей автентифікації користувачів і контролю над їхніми доступами до ресурсів[20].

JUnit – це фреймворк, розроблений для тестування програм, написаних з використанням Java. Він є ключовим елементом Test-Driven Development (TDD) і входить в сімейство фреймворків для тестування xUnit[21][22].

Тестування – це процес перевірки функціоналу програми з метою підтвердження того, що вона працює відповідно до певних вимог. Unit-тестування - це вид тестування, який здійснюється безпосередньо розробником на рівні окремих сутностей, методів або класів. Це вкрай важливий етап розробки програмного забезпечення, який допомагає створювати якісний продукт.

MySQL – це система керування реляційними базами даних (СКБД) з відкритим кодом, розроблена для ефективної обробки великих баз даних. Початково вона була схожа на mSQL, але з часом значно розширилася і стала однією з найпоширеніших СКБД. MySQL широко використовується для створення динамічних веб-сторінок, завдяки відмінній підтримці різноманітних мов програмування. Вона постійно

оновлюється та розвивається, що робить її однією з популярних виборів для роботи з базами даних[23].

Maven – це інструмент, призначений для спрощення та автоматизації процесу збирання проектів. В основному ним користуються розробники Java, хоча є плагіни, які дозволяють інтегрувати його з іншими мовами програмування, такими як C/C++, Ruby, Scala, PHP та інші[24]. Maven допомагає в управлінні залежностями проекту, збиранні виконуваних файлів, виконанні тестів та інших повсякденних завдань розробки програмного забезпечення.

Maven пропонує використовувати POM-файл, який в форматі XML описує процес побудови додатку та підключення всіх необхідних ресурсів. Цей підхід є дуже простим, але дуже ефективним, особливо коли використовуються сторонні бібліотеки та ресурси. Наявність різноманітних архетипів, які дозволяють будувати структуру проекту та включати всі необхідні файли, разом із гнучкістю налаштувань, зробило Maven надзвичайно популярним серед розробників[25].

Не менш важливо, фреймворк є добре налаштований з коробки, що означає зменшення часу на розробку.

GitLab - це веб-сервіс та система керування репозиторіями програмного коду для Git, яка включає в себе додаткові можливості, такі як власна вікі та система відстеження помилок[26]. Ця компанія пропонує послуги, аналогічні до GitHub, але з додатковими перевагами, такими як можливість користуватися приватними репозиторіями для безкоштовних підписників. Крім того, є можливість розгорнути систему на сторонніх серверах, і програмне забезпечення доступне через систему керування пакунками Omnibus.

GitLab був розроблений українським програмістом Дмитром Запорожцем. Керівний офіс та генеральний директор, Ситце Сайбрандей (Sytse Sijbrandij), розташовані в Утрехті. Код написаний мовою Ruby. Станом на травень 2016 року, компанія мала понад 80 співробітників та більше 1000 внесли свій внесок до відкритого коду.

Система GitLab має безліч безкоштовних розширень та плагінів, які дозволяють інтегрувати її з іншими сервісами.

3.6. Опис розроблених засобів проекту

Для опису розроблених засобів системи аналізу для опублікування афіш використаємо ER діаграму. ER-діаграма (схема сутності-зв'язку) - це графічний інструмент, який використовується для моделювання взаємозв'язків між різними сутностями у системі. У контексті баз даних ER-діаграми допомагають відображати структуру даних та взаємозв'язки між різними об'єктами в цій структурі[27].

У ER-діаграмах існують три основні компоненти:

- Сутності (Entities): Сутності - це об'єкти чи об'єктні класи, які можуть мати дані або атрибути.
- Зв'язки (Relationships): Зв'язки показують взаємозв'язки між сутностями. Зв'язки можуть бути один до одного (1:1), один до багатьох (1:N) або багато до багатьох (N:N).
- Атрибути (Attributes): Атрибути - це конкретна інформація, яка відноситься до сутностей.

ER-діаграми допомагають розробникам і аналітикам зрозуміти структуру даних у системі та відобразити її у зручний та зрозумілий спосіб. Це важливий етап при проектуванні бази даних, оскільки правильно спроектована ER-діаграма допомагає уникнути помилок та розуміти логіку взаємодії між даними в системі[28].

Для побудови ER діаграми потрібно визначити головні сутності. У даному контексті можна виділити три основні сутності: "Користувач", "Подія" та "Сторінка події". Зображення цих сутностей подане на діаграмі, яку можна знайти на рисунку 3.1.

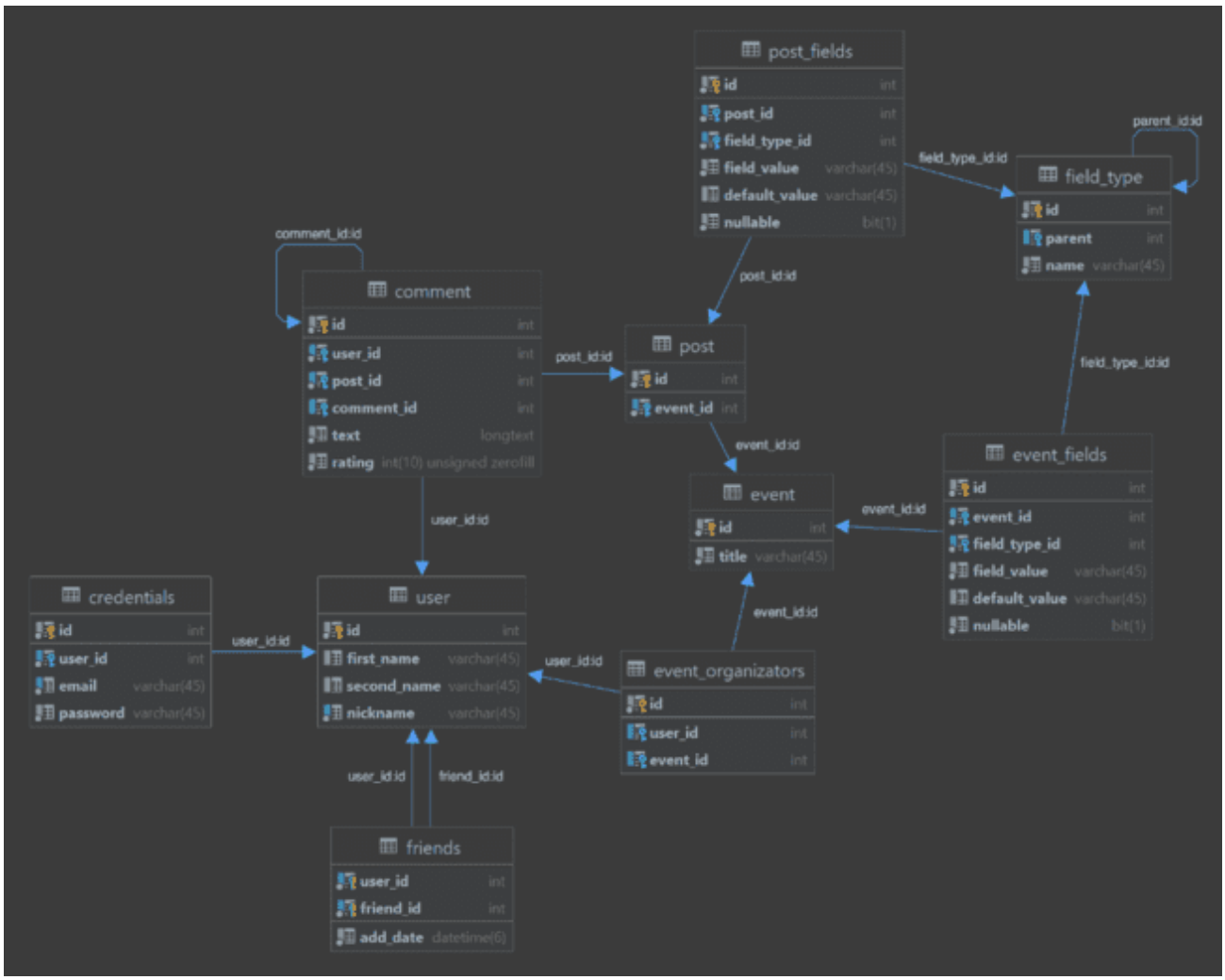

Рис. 3.1. ER діаграма.

У системі також присутні додаткові сутності, серед яких важливі наступні: «Поля для події», «Тип поля», «Поля для сторінки подій», «Організатор події», «Коментарі», «Друзі» та «Ідентифікатори».

- «Поля для події» та «Поля для сторінки подій» призначені для заповнення інформацією про події та їхні сторінки відповідно. «Тип поля» створює ієрархію полів для обох попередніх сутностей.

- «Коментарі» зберігає в собі відгуки, які користувачі залишають на сторінках подій або реагують на висловлювання інших користувачів.

- «Ідентифікатори» та «Друзі» є допоміжними сутностями для сутності «Користувач». У першій зберігається інформація, що використовується для входу в систему, а у другій - інформація про список друзів користувача.

- «Організатор подій» містить дані про користувачів, які створюють події. Ця сутність виокремлена для швидкого доступу до цієї інформації.

Висновок до третього розділу

У третьому розділі було здійснено формулювання та наукове обґрунтування постановки завдання, деталізовано мету розробки та визначено призначення та можливі області використання системи. Також вивчались потенційні наслідки впровадження системи аналізу даних для опублікування афіш, враховуючи різні сценарії її застосування.

У даному розділі була розроблена інформаційно-математична модель для вирішення проблеми завдання, а також створений відповідний алгоритм для її реалізації. Великий акцент був зроблений на обґрунтуванні вибору методів розв'язання, включаючи використані методології розробки. Результати цього розділу служать основою для подальшої практичної реалізації системи та її впровадженні.

Також проаналізовано технології для створення системи. Основними виборами стали Java і Angular для написання серверної та клієнтської частин відповідно. У контексті Java використовуються додаткові технології, такі як Spring Framework, Spring Boot, Spring MVC, Spring Data, Spring Security, JUnit та Maven. Реляційна база даних MySQL використовується як основне сховище для даних. Для передачі даних між сервером та клієнтом використовується архітектурний стиль REST. Ці технологічні вибори були здійснені на основі їхньої ефективності, надійності та можливості забезпечення потрібного функціоналу системи.

Побудовано ER діаграму для деталізації сховища даних, визначено основні сутності та атрибути.

РОЗДІЛ 4

Тестування та впровадження проекту

4.1. Тестування проекту

Тестування програмного забезпечення - це процес перевірки та валідації програмного продукту з метою забезпечення його відповідності вимогам і очікуванням користувачів. Основна мета тестування - виявлення помилок, недоліків або невідповідностей у програмному продукті до його випуску на ринок або в експлуатацію.

Тестування програмного забезпечення може включати в себе різні види тестів, такі як функціональне тестування, тестування продуктивності, тестування безпеки, тестування сумісності, тестування відмовостійкості тощо. Ці види тестів виконуються для перевірки різних аспектів програмного продукту та його взаємодії з користувачем і середовищем.

Процес тестування може бути автоматизованим або виконуватися вручну. Автоматизоване тестування використовує спеціальні програми і скрипти для виконання тестових сценаріїв, тоді як ручне тестування виконується тестувальниками, які вручну перевіряють різні функції програмного продукту.

Тестування програмного забезпечення є важливою частиною розробки програмних продуктів, оскільки воно допомагає виявляти і виправляти помилки перед тим, як програмний продукт буде випущений користувачам, що сприяє покращенню його якості та надійності.

Система реалізована у вигляді клієнт-серверної архітектури. Основними технологіями в проекті є Java та Angular. Для зберігання даних було обрано реляційну базу даних MySQL. У якості перевірки розглянемо основні сторінки системи та перевіримо їх фунціонал.

Користувачі можуть легко переходити між різними сторінками веб-сайту за допомогою головного меню, яке містить такі пункти як "Головна", "Профіль", "Допомога", "Про нас", "Увійти" і "Вийти". При цьому, після входу в систему, пункт "Увійти" автоматично зникає, а для тих, хто не авторизований, недоступні пункти

"Профіль" і "Вийти". Крім того, на головній сторінці розташований банер "Привітання", який вітає відвідувачів і заохочує їх увійти в систему або зареєструватись. Після входу в систему цей банер автоматично замінюється на "Створити подію", що дозволяє користувачам швидко створювати нові події на сайті.

Давайте детально розглянемо функціонал сайту та сторінки, які відкриваються користувачеві після входу. Коли користувач виконує вхід на сайт, він автоматично перенаправляється на головну сторінку, де відображається весь спектр доступних можливостей(рис. 4.1 - 4.4) звідки він має можливість перейти на інші. Також тут присутні рекомендовані події.

Рис. 4.1. Головна сторінка

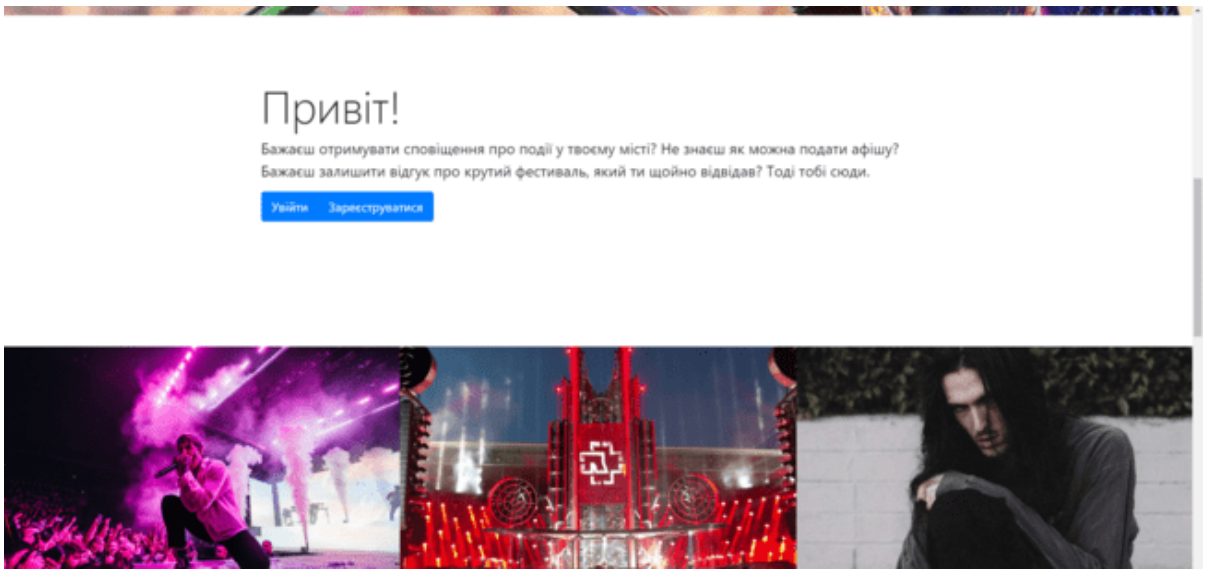

Рис. 4.2. Головна сторінка

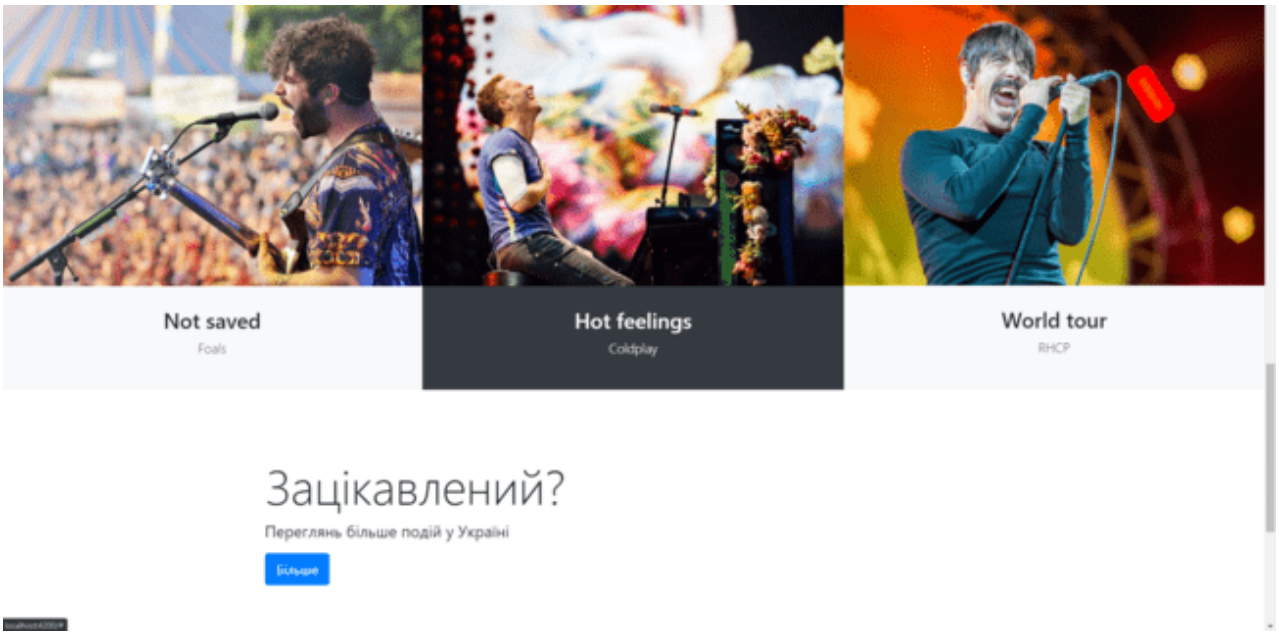

Рис. 4.3. Головна сторінка

Рис. 4.4. Головна сторінка

На головній сторінці розташована карусель з найактуальнішими та найпопулярнішими подіями. При відвідуванні сторінки новим користувачам вітають та пропонують вибір: увійти в систему чи зареєструватися. Третій розділ містить афіші популярних подій у країні користувача. Відвідувач може обрати одну з запропонованих подій або переглянути більший список. Сторінку завершує блок з посиланнями на соціальні мережі, навігацією по сайту та контактною інформацією.

Для доступу до афіші та профілю потрібно авторизуватися. Користувач переходить на сторінку входу(рис. 4.5), де йому потрібно ввести електронну пошту та пароль.

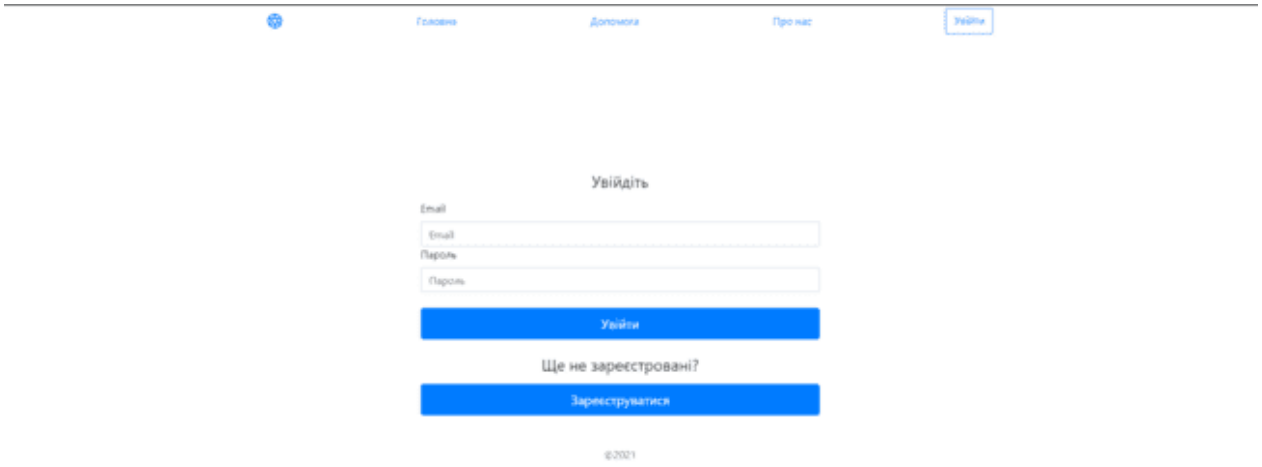

Рис. 4.5. Сторінка входу

Якщо користувач ще не зареєстрований в системі, він має можливість створити новий обліковий запис, ввівши інформацію сласну у всі необхідні поля, зокрема:

- ім’я;
- прізвище;
- псевдонім;
- номер мобільного телефону;
- електрону пошту;
- пароль.

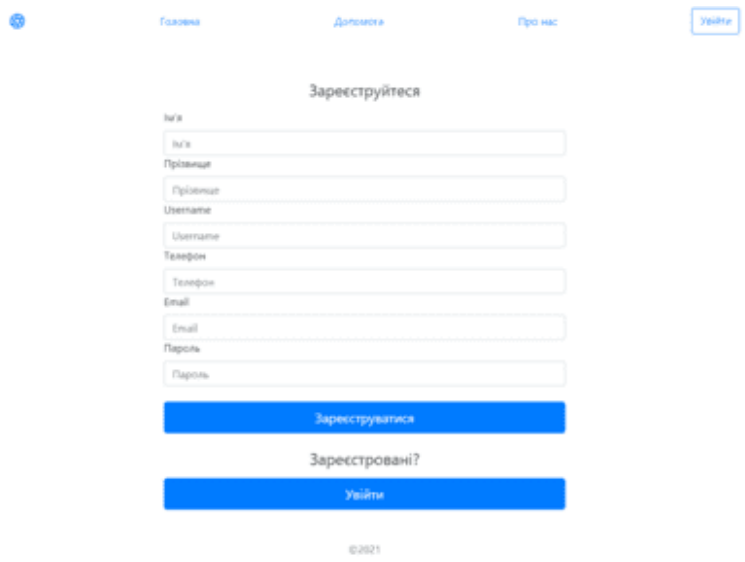

Рис. 4.6. Сторінка реєстрації

Лише після введення необхідних даних можна виконати вхід на сайт. Після успішної авторизації користувач переходить на головну сторінку, де відображається кнопка для доступу до особистого кабінету та функція виходу із системи. Важливо зауважити, що другий блок інтерфейсу також змінюється: тепер у ньому користувач може створити нову подію, використовуючи зручний інтерфейс.

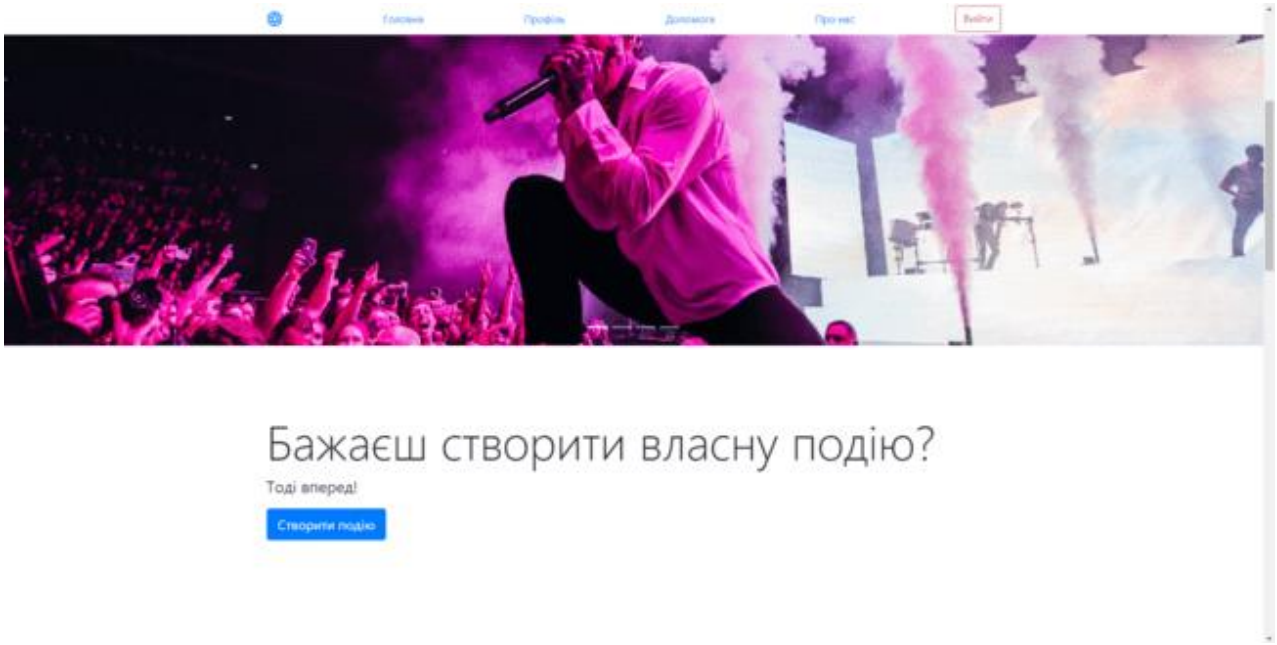

Рис. 4.7. Головна сторінка після входу

У розділ "Профіль" (зображено на рис. 4.8), система вітає ідентифікованого користувача. На цій сторінці можна відредагувати вже існуючі події користувача або створити нові.

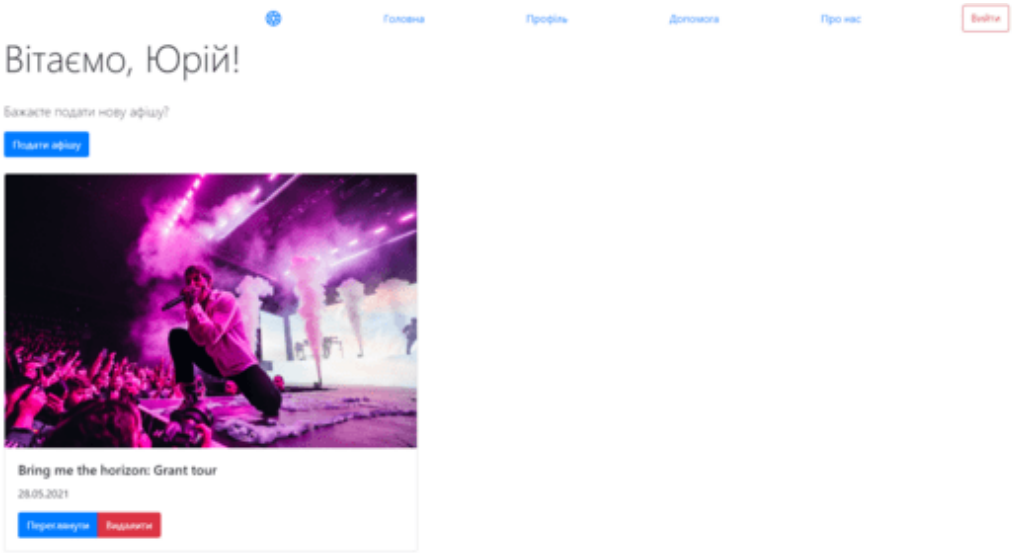

Рис. 4.8. Сторінка користувача

Для створення нової події необхідно заповнити форму (рис. 4.9) з наступними полями:

- заголовок події;
- опис події;
- зображення афіш;
- жанр події;
- дата проведення.

Після заповнення, користувач може надіслати запит на створення сторінки події. Після цього він автоматично повертається на свій особистий профіль.

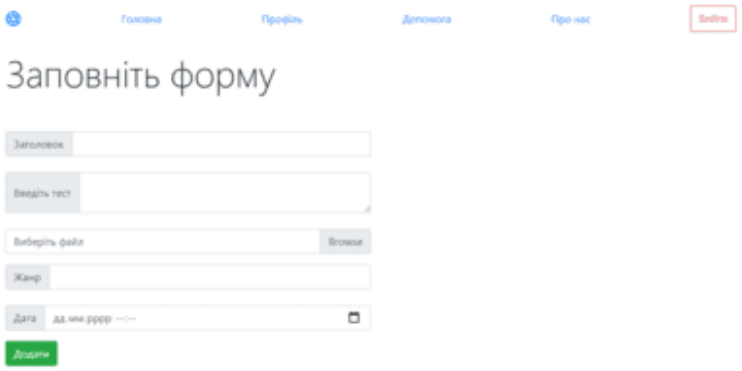

Рис. 4.9. Форма для створення події

Далі давайте переглянемо сторінку події, яку можна побачити на рисунках 4.10-4.12. У першому блоці знаходяться фотографії учасників, стила інформація про подію (місце та час проведення) та кнопка для купівлі білетів.

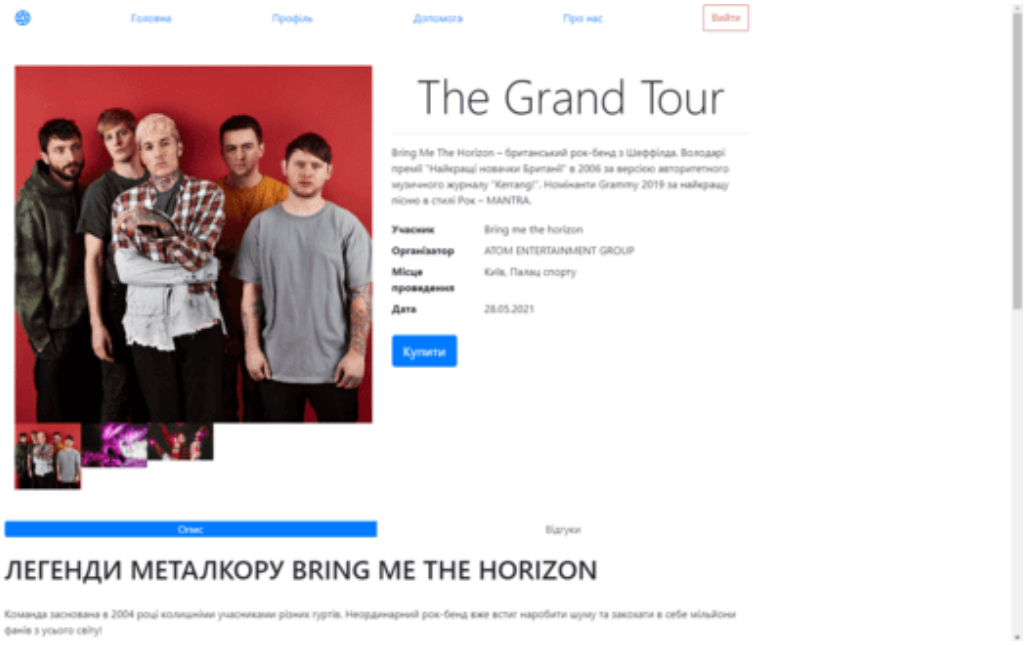

Рис. 4.10. Сторінка події

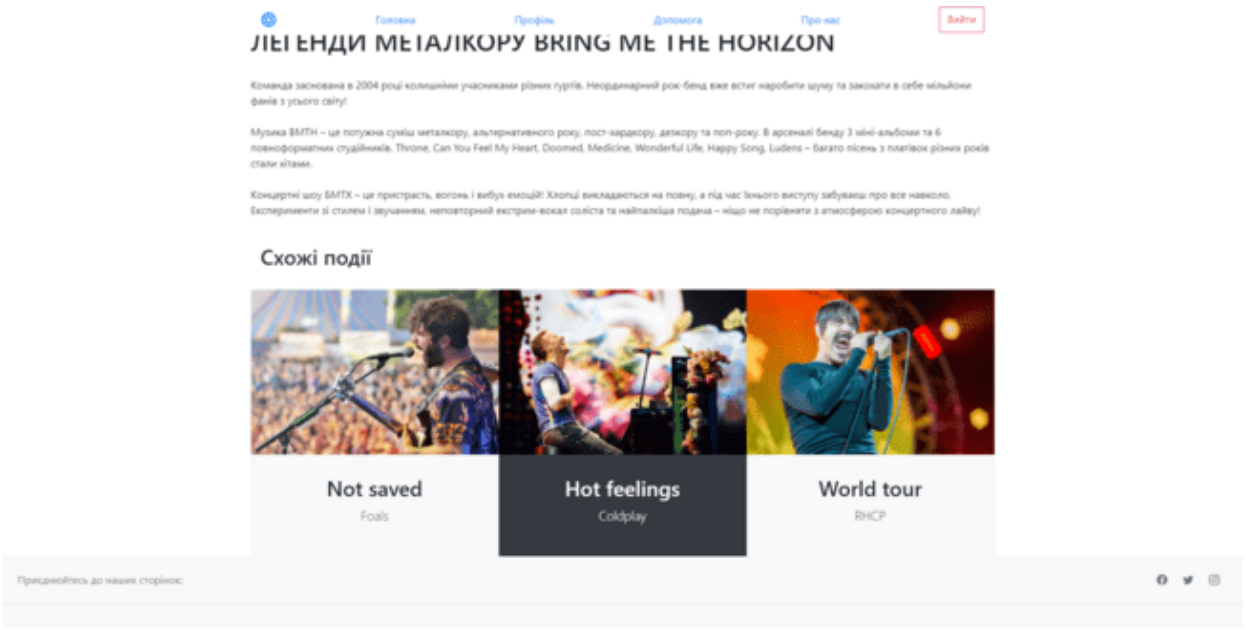

Рис. 4.11. Сторінка події

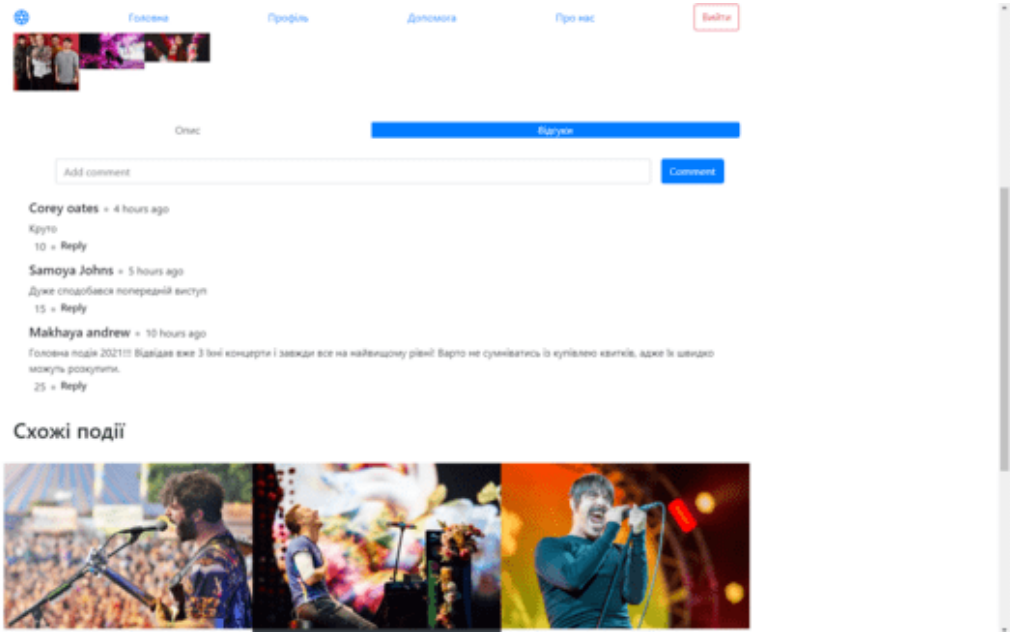

Рис. 4.12. Сторінка події

У другому блоку розташовані вкладки "Опис" та "Відгуки". У вкладці "Опис" (рис. 4.11) подана більш докладна інформація про подію, виконавця та інше. У вкладці "Відгуки" (рис. 4.12) користувачі можуть залишати свої коментарі до події та ділитися враженнями.

У третьому блоку можна знайти схожі події, які можуть зацікавити користувача. Перейдемо до допоміжних сторінок, зокрема до сторінки "Про нас" (рис. 4.13).

Рис. 4.13. Сторінка «Про нас»

На цій сторінці наведена стисла інформація про сервіс, його цілі та завдання, підходи до співпраці з клієнтами та сфера діяльності.

Далі розглянемо сторінку «Допомога» (рис. 4.14 – 4.15).

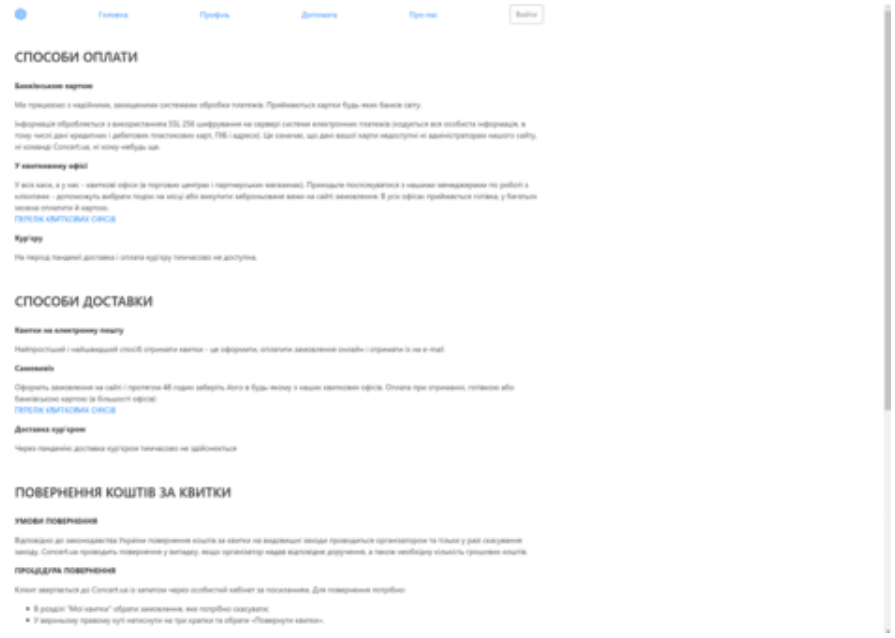

Рис. 4.14. Сторінка «Допомога»

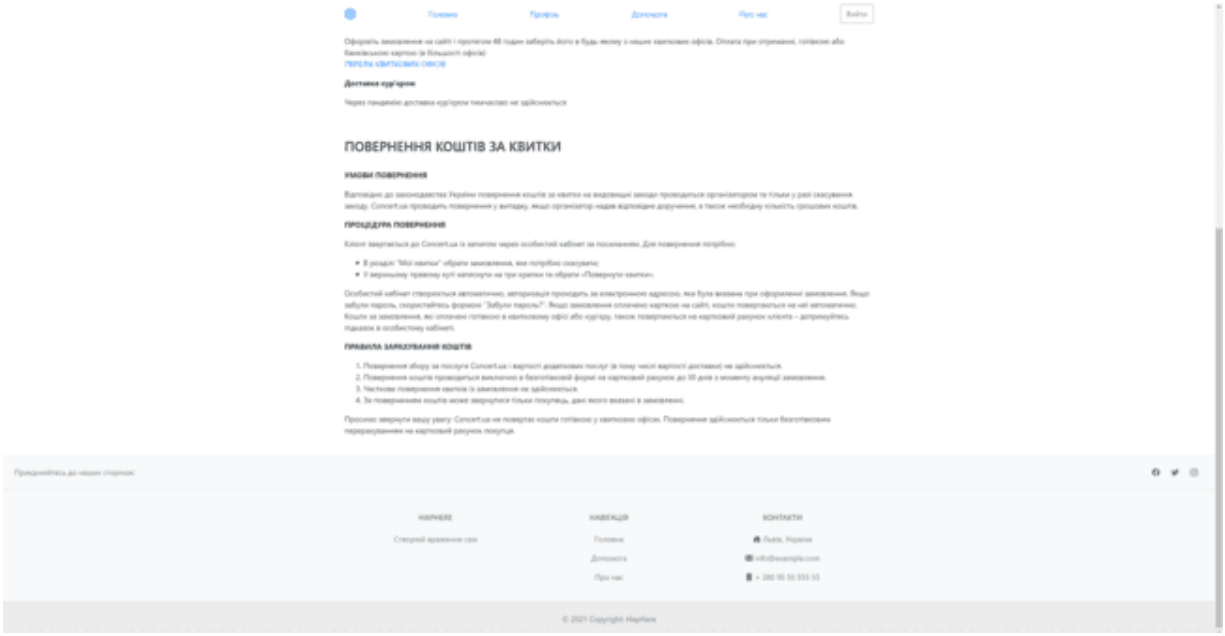

Рис. 4.15. Сторінка «Допомога»

На даній сторінці розташована вичерпна інформація про процес доставки квитків, різні способи оплати та умови повернення коштів.

4.2. Валідація та верифікація

Верифікація програмного забезпечення - це процес перевірки, чи відповідає програмний продукт визначеним специфікаціям і стандартам. У контексті розробки програмного забезпечення верифікація зазвичай означає перевірку, чи програма виконує те, що було заплановано на етапі проектування та специфікації.

Основна мета верифікації - це визначення того, чи програмний продукт правильно реалізований, чи він відповідає усім вимогам та чи дотримується стандартів розробки. Під час верифікації перевіряються аспекти програми, такі як коректність логіки, правильність виконання алгоритмів, відповідність специфікаціям та стандартам програмування.

У свою чергу, валідація (апробація) - це процес підтвердження здатності засобів, розроблених у рамках проекту, виконати розв'язання конкретної задачі за визначених умов. Цей процес включає в себе перевірку, чи задовольняють отримані результати функціональним вимогам, а також виконання контрольного розв'язання задачі

проекту (експерименту) для підтвердження належності, актуальності та коректності отриманих результатів.

Далі буде наведено реалізовані вимоги та можливості, які з’являються від створеного функціоналу.

Таблиця 4.1

Перелік функції системи

№	Функція	Можливість
1	Функція додавання нових користувачів.	Надає можливість додавати нових користувачів.
2	Функція авторизації користувача.	Надає можливість входити в систему.
3	Функція додавання афіш.	Надає можливість додавати афіші.
4	Функція перегляду.	Надає можливість переглядати афіші.
5	Функція редагування.	Надає можливість редагувати афіші.
6	Функція видалення.	Надає можливість видаляти афіші.
7	Функція фільтрування.	Надає можливість переглядати афіші за певними критеріями.
8	Функція коментування.	Надає можливість коментувати записи зареєстрованим користувачам.
9	Функція модерування.	Надає можливість модерувати афіші та блокувати користувачів.
10	Функція купівлі квитків.	Надає можливість змінювати кольорове оформлення сайту зареєстрованим користувачам.
11	Функція надсилання рекомендацій.	Надає можливість отримувати пропозиції електронною поштою.

Усі перелічені функції реалізовані за допомогою бібліотек Java, дані збережено в реляційній базі даних MySql. Від реалізації базових вимог можна перейти до більш розширених у побудові наступних версій системи. Так функцію авторизації можна реалізувати за допомогою Технологія єдиного входу (Single Sign-On, SSO), що значно спростить досвід користування користувачам.

Відповідно до користувацьких вимог:

1. Аналіз заданих критеріїв - досягнуто завдяки рекомендаційній системі.
2. Аналіз типу подачі критеріїв - досягнуто завдяки навчанню рекомендаційної системи.

3. Систему подання статистики – буде реалізовано у наступній версії.

Перевірка відповідності нефункціональних вимог:

1. Адаптивний інтерфейс - розроблений інтерфейс.
2. Масштабованість - система здатна розширюватись та працювати на декількох віртуальних машинах, як на одному пристрої.
3. Надійність - система працює без збоїв, сенситивні дані користувачів зашифровано.
4. Простота у використанні - інтерфейс системи є не перевантаженим, простим та зрозумілим.

Уніфікація, згідно стандарту ISO/IEC 9126, визначає зовнішні та внутрішні характеристики якості програмного продукту. Для встановлення кількісних критеріїв якості, які дозволяють перевіряти та підтверджувати відповідність програмного забезпечення заданим вимогам, використовують різні зовнішні вимірювані властивості програмного забезпечення. Ці метрики включають час виконання компонентів, моделі та їх оцінки, а також діапазони зміни значень. Зазвичай ці метрики використовуються на етапі тестування функціоналу і відомі як зовнішні метрики. Вони служать як моделі для оцінки властивостей програмного забезпечення.

Інтероперабельність – здатність до правильної взаємодії. Логіка програми перевірена, проблем не виникає.

Мобільність – до системи можна під’єднатись за допомогою веб браузера, що є найдоступнішим способом на з’єднання.

Масштабованість – система може бути легко масштабована за допомогою сучасних технологій хмарного сервісу.

Взаємодія з користувачем – система не викликає труднощів при роботі з нею.

Перевірка відповідності вимогам якості FURPS+ передбачає оцінку розробленого програмного забезпечення за п'ятьма основними критеріями:

- *Функціональність*: Розроблене програмне забезпечення повинно вирішувати поставлені користувачами завдання навіть у різних умовах

експлуатації. Це включає в себе правильне виконання функцій і завдань, які користувачі очікують від системи.

- *Придатність до використання*: Система повинна бути зручною у використанні для користувачів. Це включає в себе інтуїтивний і легкий для зрозуміння інтерфейс користувача, що сприяє зручності та ефективності взаємодії з системою.
- *Надійність*: Розроблене програмне забезпечення повинно бути надійним та стійким, здатним підтримувати автоматичну працездатність в заданих умовах. Це означає, що система повинна бути стійкою до помилок та збоїв, які можуть виникнути у процесі роботи.
- *Продуктивність*: Розроблене програмне забезпечення повинно бути продуктивним, тобто воно не повинно вимагати значних додаткових ресурсів при виконанні робіт з плином часу. Це може включати в себе оптимізацію роботи системи, щоб вона працювала ефективно та швидко навіть при великому обсязі даних чи користувачів.
- *Експлуатаційна придатність*: Розроблене програмне забезпечення повинно надавати швидку відповідь та бути готовим до використання у реальних умовах експлуатації. Це може включати в себе швидкий відгук системи на запити користувачів та забезпечення безперебійної роботи сервісу впродовж тривалого часу.

Розроблене програмне забезпечення повністю відповідає цим вимогам.

4.3. Розгортання

Для роботи системи потрібно таке програмне забезпечення: Java (JDK 11), Maven, Node.js, Angular, MySQL Workbench. Для розгортання системи потрібно виконати наступні кроки:

- Встановити Java (JDK 11). https://jdk.java.net/
- Додати змінну JAVA_HOME у змінні середовища комп’ютера.
- Встановити Maven. https://maven.apache.org/

- Додати змінну MAVEN_HOME у змінні середовища комп’ютера.
- Встановити Node.js https://nodejs.org/uk/
- Встановити Angular за вказівками документації.
- Встановити MySQL Workbench. https://dev.mysql.com/downloads/installer/
- Перезавантажити комп’ютер для внесення змін у змінні середовища.
- Скачати архів з проектом.
- Відкрити MySQL Workbench.
- Створити базу даних HapHere.
- Запустити init.sql з архіву для розгортання бази.
- Завантажити залежності Maven з pom.xml за допомогою команди mvn install.
- Запустити проект.
- Відкрити консоль.
- Перейти в директорію проекту, відкрити папку client.
- Встановити модуль для Node.js ввівши команду в консоль `npm install --global http-server`
- Запустити Angular через команду `ng serve`.
- Відкрити папку static.
- Запустити команду `http-server ./`
- Відкрити браузер.
- Перейти за посиланням http://localhost:4200/
- Готово.

У результаті виконання цих дій, вдалось розгорнути систему, яка працює надійно та продуктивно.

Висновок до четвертого розділу

У цьому розділі було надано детальний опис програмного продукту у формі веб-сайту, що відповідає встановленому стандарту. Проведено загальний огляд

реалізованого програмного продукту, розкрито його функціональне призначення, розглянуто логічну структуру та всі використані технічні засоби.

Було здійснено аналіз контрольного прикладу, проведено тестування та підтверджено, що система успішно відповідає встановленим цілям і працює відповідно до вимог, враховуючи всі специфікації та стандарти, що були зазначені у вихідних вимогах. Це свідчить про досягнення системою своєї основної мети та готовності до подальшого використання у практичних сценаріях.

РОЗДІЛ 5

Економічне обґрунтування доцільності роботи

5.1. Економічна характеристика проектного рішення

Метою роботи є створити системи аналізу даних для опублікування афіш, яка буде допомагати створювати події, надавати зручні інструменти та можливості для їх поширення.

Кінцевим результатом є веб-застосунок, у якому необхідно зареєструватися, щоб його можна було використовувати для купівля чи продажу білетів. Основним функціоналом веб-сайту є завантаження афіш; створення подій; купівлі білетів.

Сферою застосування є соціально-культурна сфера, сфери, які дотичні до неї, сфера обслуговування.

Потенційними споживачами запропонованої системи є виконавці, артисти, митці, особливо початкового рівня; люди, які зацікавлені у відвіданні культурних подій; заклади із сфери обслуговування, які готові здавати приміщення для проведення подій.

Розроблення запропонованої системи несе впровадження нових підходів у сферу продажі квитків, оскілки поєднює в собі всі процеси для організації подій.

5.2. Розрахунок витрат на розробку та впровадження проектного рішення

Витрати на розробку і впровадження програмного засобу (К) визначаються як:

$$K_{\text{заг}} = K_1 + K_2, \tag{5.1}$$

де K_1 – витрати на розробку програмного засобу, грн.; K_2 – витрати на відлагодження і дослідну експлуатацію програмного засобу на ЕОМ, грн.

Для здійснення розробки програмного продукту потрібно такі працівники:

- Front-end (F) розробник та back-end (B) програміст;
- Дизайнер (D).

Відповідно до цього витрати на оплату праці становлять:

- місячний оклад front-end програміста 60 000,0 грн/міс;
- місячний оклад back-end програміста 80 000,0 грн/міс;

- місячний оклад дизайнера 35 000,0 грн/міс.

Витрати на розробку і впровадження програмного засобу

Середньоденна заробітна плата *i*-го розробника ($ЗП_{Д\,i}$) обчислюється за формулою:

$$ЗП_{Д\,i} = \frac{ЗП_i}{Ф_м}, \quad (5.2)$$

де $ЗП_i$ - основна місячна заробітна плата розробника *i*-ої спеціальності, грн.; $Ф_м$ - місячний фонд робочого часу, днів (24 дні).

Для обраних спеціалістів середньоденна заробітна плата буде становити:

$$ЗП_D = \frac{35\,000}{24} = 1\,458{,}3 \text{ грн}$$

$$ЗП_F = \frac{60\,000}{24} = 2\,500{,}0 \text{ грн}$$

$$ЗП_B = \frac{80\,000}{24} = 3\,333{,}3 \text{ грн}$$

Розрахунок витрат на оплату праці усіх розробників проекту обчислюємо за формулою:

$$В_{ОП} = \sum_{i=1}^{N} n_i * t_i * ЗП_{Д\,i}, \quad (5.3)$$

де n_i – кількість розробників проекту *i*-ої спеціальності, чол.; t_i – час, витрачений на розробку проекту працівником *i*-ої спеціальності, дні; $ЗП_{Д\,i}$ – денна заробітна плата розробника *i*-ої спеціальності, грн..

Загальна витрата на оплату праці становить:

$В_{оп} = 1 * 120 * 3\,333{,}3 + 1 * 60 * 2\,500{,}0 + 1 * 40 * 1\,458{,}3 = 608\,329$ грн

У табл. 5.1 зручно представлено бюджет витрат на оплату праці спеціалістів для розробки запропонованої системи.

Таблиця 5.1

Розрахунок витрати на оплату праці

Спеціальність розробника	Кількість розробників, чол.	Час роботи, дні	Денна заробітна плата розробника, грн.	Витрати на оплату праці, грн.
Дизайнер	1	40	1 458,3	58 332
Front-end розробник	1	60	2 500,0	150 000
Back-end програміст	1	120	3 333,3	399 996
Разом:	3	220	6 291,6	608 329

Витрати на оплату праці працівникам

Розмір єдиного внеску в залежності від класу рівня ризику становить 36,7%. У таблиці 5.2 наведено розрахунки витрат на сплату єдиного соціального внеску відповідно для кожного спеціаліста. Загальна сума єдиного соціального внеску становить:

$$B_{\text{єсв}} = 21\,407{,}8 + 55\,050{,}0 + 146\,798{,}5 = 223\,256{,}3\text{грн}$$

Таблиця 5.2

Розрахунок витрати на сплату єдиного соціального внеску

Спеціаліст	Сума основної заробітної плати, грн	Сума додаткового заробітної плати, грн	Разом витрати на оплату праці, грн	Сума ЄСВ, грн
Дизайнер	58 332	-	58 332	21 407,8
Front-end розробник	150 000	-	150 000	55 050,0
Back-end програміст	399 996	-	399 996	146 798,5
Разом:	608 329	-	608 329	223 256,3

Витрати на додаткові вироби

Крім цього, під час розробки потрібно використовувати професійне програмне забезпечення, тому враховуються витрати на ліцензії програми менеджменту Jira,

вебсервісу для розробки ПЗ Github та серверу зберігання AWS. Витрати на ці додаткові пристрої представлені у табл. 5.3. Транспортно-заготівельні витрати становлять 10 % суми витрат на додаткові вироби, а саме:

$$В_д = 328\ 103 + 8370 + 17\ 520 + 4\ 752 = 358\ 745 \text{ грн}$$

Таблиця 5.3

Розрахунок витрат на куповані вироби

№ п/п	Найменування купованих виробів	Марка, тип	Кількість на розробку, шт.	Ціна за одиницю, грн.	Сума витрат, грн.	Сума витрат з урахуванням транспортно-заготівельних витрат, грн.
1	Jira	Atlassian	1	9 000	9 000	0
2	AWS	Amazon	1	20 000	20 000	0
3	Github	Microsoft	1	5 000	5 000	0
Всього						36 000

Накладні витрати проектних організацій включають витрати на управління, загальногосподарські, невиробничі витрати. Вони становлять 30% витрат на оплату праці, а саме:

$$B_н = 608\ 329 * 0{,}3 = 202\ 776{,}3 \text{ грн}$$

Інші витрати – це витрати, які не враховані в попередніх статтях витрат. Вони розраховуються за 10% до витрат на оплату праці, що становить:

$$B_{ін} = 608\ 329 * 0{,}1 = 60\ 832{,}9 \text{ грн}$$

Витрати на розробку проектного рішення обчислюємо за формулою:

$$K_1 = B_{оп} + B_{єсв} + B_д + B_н + B_{ін} \quad (5.4)$$

$$K_1 = 608\ 329 + 223\ 256{,}3 + 36\ 000 + 202\ 776{,}3 + 60\ 832{,}9 = 1\ 131\ 194{,}5 \text{ грн}$$

Витрати на відлагодження і дослідну експлуатацію системи визначаємо згідно формули:

$$K_2 = S_{м.г.} * t_{від}, \quad (5.5)$$

де $S_{м.г.}$ – вартість однієї години роботи ПК, грн./год; $t_{від}$ – кількість годин роботи ПК на відлагодження програми, год.

Загальна кількість днів роботи на ЕОМ рівна 225 днів. Середній щоденний час роботи на ЕОМ – 8 год., тому:

$$t_{\text{від}} = 225 * 8 = 1\,800 \text{ год.}$$

За даними обчислювального центру НУ «Львівська Політехніка» для ЕОМ типу IBM PC/AT $S_{\text{м.г.}}$ = 5 грн. Отже, витрати на відлагодження і дослідну експлуатацію системи становить:

$$K_2 = 5 * 1\,800 = 9\,000 \text{ грн}$$

$$\text{К} = 608\,329 + 223\,256{,}3 + 36\,000 + 202\,776{,}3 + 60\,832{,}9 + 9\,000$$
$$= 1\,140\,194{,}5 \text{ грн}$$

У таблиці 5.4 представлено результати всіх попередніх розрахунків.

Таблиця 5.4

Кошторис витрат на розробку проектного рішення

Найменування елементів витрат	**Сума витрат, грн.**
Витрати на розробку проектного рішення, у т.ч.:	
витрати на оплату праці	608 329
сплата єдиного соціального внеску	223 256,3
витрати на додаткові вироби, що закуповуються	36 000
накладні витрати	202 776,3
інші витрати	60 832,9
Витрати на відлагодження і дослідну експлуатацію системи	9 000
Всього	1 140 194,5

5.3. Визначення комплексного показника якості

Комплексний показник якості визначається шляхом порівняння показників якості проектованої системи і вибраного аналогу. Для визначення доцільно використовувати систему показників технічного рівня і якості, яка містить в собі наступні показники:

1. функціональні показники;
2. надійність;
3. зручність застосування;
4. продуктивність;
5. підтримуваність.

Комплексний показник якості проектованої системи визначаємо методом арифметичного середньозваженого з формули:

$$П_Я = \sum_{i=1}^{m} C_i * q_i, \quad (5.6)$$

де m – кількість одиничних показників (параметрів), прийнятих для оцінки якості проектованої системи; q_i – коефіцієнт вагомості кожного з параметрів щодо їхнього впливу на технічний рівень та якість проектованої системи, у сумі має дорівнювати 1; C_i – часткові показники якості, визначені порівнянням числових значень одиничних показників проектованої системи і аналога за формулами:

$$C_i = \frac{П_{прі}}{П_{аі}} \text{ або } C_i = \frac{П_{аі}}{П_{прі}}, \quad (5.7)$$

де $П_{прі}$, $П_{аі}$ – кількісні значення i-го одиничного показника якості відповідно проектованої системи і аналога.

Розрахунок коефіцієнту вагомості кожного з параметрів здійснюється за методом попарних порівнянь та формулою для визначення власного вектору пріоритетів (5.8).

$$x_i = \frac{\sqrt[n]{\prod_{j=1}^{n} a_{ij}}}{\sum_{i=1}^{n} \sqrt[n]{\prod_{j=1}^{n} a_{ij}}} \quad (5.8)$$

де x_i – вектор пріоритету; n - кількість порівнювальних елементів(параметрів); a_{ij} – оціночна ступінь важливості.

Розрахунок визначення комплексного показника якості проектованої системи представлений у таблиці 5.5.

Таблиця 5.5

Визначення комплексного показника якості проектованої системи (аналога)

Показники	Числове значення показників, бали		Відносний показник якості, C_i	Коефіцієнт вагомості q_i	$C_i * q_i$
	Аналог	Проект. прогр. продукт			
Функціональність	8	8	1	0,20868	0,20866
Надійність	6	7	1,17	0,19182	0, 22459
Зручність	7	8	1,14	0,19701	0,26268
Продуктивність	7	9	1,29	0,20802	0,26745
Підтримуваність	7	9	1,29	0,19448	0, 25088
Всього	32	41	5,89	1	1,21267

Комплексний показник рівня якості розроблюваного програмного забезпечення визначається порівнянням його показників з відповідними значеннями показників аналога і визначення вектору пріоритетів. Отже, комплексний показник якості дорівнює $П_Я$ = 1,21.

5.4. Визначення експлуатаційних витрат

При порівнянні програмних засобів в експлуатаційні витрати включають вартість підготовки даних (E_1) і вартість годин роботи ПК (E_2). Одноразові експлуатаційні витрати визначаються за формулою:

$$E_{П(А)} = E_{1П(А)} + E_{2П(А)} \quad (5.9)$$

де $E_{П(А)}$ – одноразові експлуатаційні витрати на проектне рішення (аналог), грн.; $E_{1П(А)}$ – вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.; $E_{2П(А)}$ – вартість машино-годин роботи ПК для проектного рішення (аналогу), грн.

Річні експлуатаційні витрати визначаються за формулою:

$$B_{(e)П(А)} = E_{П(А)} * N_{П(А)} \quad (5.10)$$

де $B_{(e)\text{П(А)}}$ – експлуатаційні річні витрати проектного рішення, грн.; $N_{\text{П(А)}}$ – періодичність експлуатації проектного рішення (аналогу), разів/рік.

Вартість підготовки даних для експлуатації проектного рішення (аналогу) (E_1) визначаються за формулою:

$$E_1 = \sum_{i=1}^{N} n_i * t_i * \text{ЗПг}_i, \tag{5.11}$$

де i – номери категорій персоналу, які беруть участь у підготовці даних; n_i – кількість співробітників і-ї категорії, чол.; t_i – трудомісткість роботи співробітників і-ї категорії, чол.; ЗПг_i – середньо-годинна ставка робітника і-ї категорії з врахуванням сплати єдиного соціального внеску, грн./год.

Середньо-годинна ставка оператора визначається за формулою:

$$\text{ЗПг}_i = \frac{\text{ЗПг}_{0i}(1+b)}{\text{Фг}}, \tag{5.12}$$

де ЗПг_{0i} – основна місячна зарплата працівника і-ї категорії, грн.; b – коефіцієнт, який враховує сплату єдиного соціального внеску (b = 0,3677); Фг – місячний фонд робочого часу, год.

Для проектного рішення середньо-годинна ставка для обраних спеціалістів становить:

$$\text{ЗПг}_1 = \frac{90\,000 * (1+0{,}3677)}{24*8} = 641{,}11 \text{ грн}$$

$$\text{ЗПг}_2 = \frac{60\,000 * (1+0{,}3677)}{24*8} = 427{,}41 \text{ грн}$$

$$\text{ЗПг}_3 = \frac{35\,000 * (1+0{,}3677)}{24*8} = 249{,}32 \text{ грн}$$

На підготовку даних кожен працівник витрачатиме 10 днів, тому вартість підготовки даних для роботи на ПК становить:

$$E_{1\text{П}} = 1*10*641{,}11 + 1*10*427{,}41 + 1*10*249{,}32 = 13\,178{,}4 \text{ грн}$$

Одноразові експлуатаційні витрати на проектне рішення становлять:

$$E_{\text{П}} = 13\,178{,}4 + 10*8*8 = 13\,818{,}4 \text{ грн}$$

Річні експлуатаційні витрати з урахуванням того, що періодичність експлуатації проектного рішення дорівнює 3 разів/рік, становлять:

$$B_{(е)П} = 13\,818{,}4 \ * 3 = 41\,455{,}2 \text{ грн}$$

Над проектом-аналогом працюють схожого рівня спеціалісти. Їх середньо-годинна ставка розраховується наступним чином:

$$ЗПг_1 = \frac{90\,000 * (1 + 0{,}3677)}{24 * 8} = 641{,}11 \text{ грн}$$

$$ЗПг_2 = \frac{60\,000 * (1 + 0{,}3677)}{24 * 8} = 427{,}41 \text{ грн}$$

$$ЗПг_3 = \frac{35\,000 * (1 + 0{,}3677)}{24 * 8} = 249{,}32 \text{ грн}$$

На підготовку даних програми-аналога кожен працівник витратив 15 днів, то вартість підготовки даних для роботи на ПК становить:

$$E_{1П} = 1 * 15 * 641{,}11 + 1 * 15 * 427{,}41 + 1 * 15 * 249{,}34 = 19\,767{,}6 \text{ грн}$$

Одноразові експлуатаційні витрати на проектне рішення становлять:

$$E_П = 19\,767{,}6 + 15 * 5 * 8 = 20\,367{,}6 \text{ грн}$$

Річні експлуатаційні витрати з урахуванням того, що періодичність експлуатації проектного рішення дорівнює 3 разів/рік, становлять:

$$B_{(е)П} = 20\,367{,}6 * 3 = 61\,102{,}8 \text{ грн}$$

Усі здійсненні розрахунки для проектного рішення та програми-аналога представленні у таблиці 5.6.

Таблиці 5.6

Розрахунок витрат на підготовку даних для роботи на ЕОМ

Категорія персоналу	Чисельність співробітників і-ої категорії, чол.	Час роботи співробітників і-ої категорії, год.	Середньо-годинна ЗП співробітника і-ої категорії, грн.	Витрати на підготовку даних, грн.
Проектне рішення				
Дизайнер	1	10	249,32	2 493,2
Front-end розробник	1	10	427,41	4 274,1
Back-end розробник	1	10	641,11	6 411,1
Всього	3	30	1 317,84	13 178,4
Аналог				
Дизайнер	1	15	249,32	2 493,2
Front-end розробник	1	15	427,41	4 274,1
Back-end розробник	1	15	641,11	6 411,1
Всього	3	45	1 317,84	19 767,6

5.5. Розрахунок ціни споживання проектного рішення

Ціна споживання ($Ц_С$) – це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

$$Ц_{С(П)} = Ц_П + В_{(е)NPV}, \quad (5.13)$$

де $Ц_П$ – ціна придбання проектного рішення, грн.; $В_{(е)NPV}$ – теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

$$Ц_П = К * \left(1 + \frac{Р}{100}\right) + К_0 + К_к, \quad (5.14)$$

де Р – норматив рентабельності (23%); $К_0$ – витрати на прив'язку та освоєння проектного рішення на конкретному об’єкті, грн. ($К_0$ = 0 грн); $К_к$ – витрати на доукомплектування технічних засобів на об'єкті, грн ($К_к$ = 0 грн).

Розрахунок ціни споживання для проектного рішення дорівнює:

$$Ц_П = 1\,131\,194{,}5 * (1 + 0{,}23) + 0 + 0 = 1\,391\,369{,}24 \text{ грн}$$

Ціна споживання для проекту-аналога дорівнює: 1 391 369,24 грн.

Теперішня вартість витрат на експлуатацію проектного рішення розраховується за формулою:

$$\text{В}_{(\text{е})NPV} = \sum_{t=1}^{T} \frac{B_{(T)Nt}}{(1+R)^{t}}, \tag{5.15}$$

де $B_{(T)Nt}$ – річні експлуатаційні витрати в t-ому році, грн.; T – строк служби проектного рішення, років; R – річна ставка проценту банків (18%).

Оскільки термін експлуатації проектного рішення становить 3 роки, теперішня вартість витрат на експлуатацію проектного рішення становить:

$$\text{В}_{(\text{е})NPV} = \frac{41\,455{,}2}{(1+0{,}18)^{1}} + \frac{41\,455{,}2}{(1+0{,}18)^{2}} + \frac{41\,455{,}2}{(1+0{,}18)^{3}}$$

$$= 35\,131{,}53 + 29\,772{,}48 + 25\,230{,}91 = 90\,134{,}92 \text{ грн}$$

Таким чином ціна споживання проектного рішення становить:

$$\text{Ц}_{\text{С(П)}} = 1\,391\,369{,}24 + 90\,134{,}92 = 1\,481\,504{,}16 \text{ грн}$$

Аналогічно визначається ціна споживання для аналогу. Визначимо теперішню вартість витрат на експлуатацію аналогу. Термін експлуатації аналогу становить 4 років, тоді:

$$\text{В}_{(\text{е})NPV} = \frac{61\,102{,}8}{(1+0{,}18)^{1}} + \frac{61\,102{,}8}{(1+0{,}18)^{2}} + \frac{61\,102{,}8}{(1+0{,}18)^{3}} + \frac{61\,102{,}8}{(1+0{,}18)^{4}}$$

$$= 51\,782{,}03 + 43\,883{,}08 + 37\,189{,}05 + 31\,516{,}14 = 164\,370{,}3 \text{ грн}$$

Отже, ціна споживання проекту-аналогу становить:

$$\text{Ц}_{\text{С(А)}} = 1\,391\,369{,}24 + 164\,370{,}3 = 1\,555\,739{,}54 \text{ грн}$$

5.6. Визначення показників економічної ефективності

Якщо базою для порівняння обрані відповідні програмні засоби, в даному розділі розраховуються такі показники:

1) Показник конкурентоспроможності:

$$\text{К}_{\text{кс}} = \frac{\text{Ц}_{\text{с(п)}} * \text{П}_{\text{Я}}}{\text{Ц}_{\text{С(а)}}} \tag{5.16}$$

$$\text{К}_{\text{кс}} = \frac{1\,481\,504{,}16 * 1{,}21}{1\,555\,739{,}54} = 1{,}152$$

2) Економічний ефект в сфері експлуатації (грн.):

$$E_{\text{екс}} = \text{В}_{(\text{е})a} - \text{В}_{(\text{е})n} \quad (5.17)$$

$$E_{\text{екс}} = 164\,370{,}3 - 90\,134{,}92 = 74\,235{,}38 \text{ грн}$$

3) Економічний ефект в сфері проектування (грн.):

$$E_{\text{пр}} = \text{Ц}_a - \text{Ц}_n \quad (5.18)$$

$$E_{\text{пр}} = 1\,555\,739{,}54 - 1\,481\,504{,}16 = 74\,235{,}38 \text{ грн}$$

Так як $E_{\text{пр}} > 0$ та $E_{\text{екс}} > 0$, то розраховуються додаткові економічні ефекти в сфері експлуатації та проектування:

4) Додатковий економічний ефект в сфері експлуатації (грн.):

$$E_{\text{екс Д}} = \sum_{i=1}^{T} E_{\text{екс}} * (1+R)^{T-i} \quad (5.19)$$

$$E_{\text{екс Д}} = 74\,235{,}38 * (1{,}18^0 + 1{,}18^1 + 1{,}18^2) = 265\,198{,}47 \text{ грн}$$

5) Додатковий економічний ефект в сфері проектування (грн.):

$$E_{\text{пр Д}} = E_{\text{пр}} * (1+R)^T \quad (5.20)$$

$$E_{\text{пр Д}} = 74\,235{,}38 * (1+0{,}18)^3 = 121\,971{,}10 \text{ грн}$$

6) Термін окупності витрат на проектування рішення (років):

$$T_{\text{ок}} = \frac{\text{К}}{E_{\text{екд}}} \quad (5.21)$$

$$T_{\text{ок}} = \frac{1\,131\,194{,}5}{265\,198{,}47} = 4{,}2655 \text{ років}$$

Результуючі обчислених показників економічної ефективності представленні у таблиці 5.7.

Таблиця 5.7

Показники економічної ефективності проектного рішення

Найменування показників	**Одиниці вим.**	**Значення показників**	
		Аналог	**Проектне рішення**
1. Капітальні вкладення	грн.	-	1 131 194,5
2. Ціна придбання	грн.	1 391 369,24	1 391 369,24
3. Річні експлуатаційні витрати	грн.	61 102,8	4 1455,2
4. Ціна споживання	грн.	1 555 739,54	1 481 504,16

Продовження таблиця 5.7

5. Економічний ефект в сфері експлуатації	грн.	-	74 235,38
6. Додатковий економічний ефект в сфері експлуатації	грн.	-	265 198,47
7. Економічний ефект в сфері проектування	грн.	-	74 235,38
8. Додатковий економічний ефект в сфері проектування	грн.	-	121 971,10
9. Термін окупності витрат на проектування рішення	роки	-	4,2655
10. Коефіцієнт конкурентоспроможності		-	1,152

Висновки до п’ятого розділу

У цьому розділі здійснено розрахунок витрат на розробку та впровадження системи аналізу даних для опублікування афіш. Враховуються обчислення комплексного показника якості, експлуатаційних витрат, ціни споживання проектного рішення та показники економічної ефективності. Витрати на розробку проектного рішення вийшли 1 131 194,5 грн., а показник якості у порівнянні проектного рішення із аналогом становить 1,21. Показник конкурентоспроможності вийшов 1,152, це свідчить, що запропонована система переважає над аналогами. Крім цього, отримано економічний ефект у сфері експлуатації, що становить 74 235,38 грн. Саме тому розробка та впровадження запропонованої системи для догляду за деревами є доцільним.

ВИСНОВКИ

Результатом виконання магістерської кваліфікаційної роботи є створена веб-система аналізу даних для опублікування афіш. В процесі виконання роботи були проведені аналіз предметної області, системний аналіз та аналіз технологій, необхідних для створення системи. Проведено тестування програмного продукту на коректність роботи та оцінку всього функціоналу. Були також проведені економічні розрахунки для підтвердження рентабельності створення системи.

На початковому етапі проведено аналіз актуальності створення системи та детальне дослідження основних конкурентів. У результаті були виокремлені переваги та недоліки цих конкурентів. Особлива увага була приділена порівняльному аналізу, де визначено, що деякі конкуренти не надають можливості переглядати квитки за межами власної країни та залишати відгуки про події, а також створювати власні. На цьому етапі також була детально розглянута функціональність запланованої системи для аналізу даних та опублікування афіш. Атрибути цієї функціональності були визначені, а критичні функції, які потрібно врахувати в першій версії системи, були ідентифіковані. Цей етап визначає стратегічний напрямок для подальшого розвитку та реалізації системи.

Під час розробки моделі системних вимог були визначені та класифіковані ключові категорії вимог, такі як бізнес-вимоги, функціональні вимоги, нефункціональні вимоги та вимоги користувачів. Глибокий аналіз потенційних ризиків, пов'язаних із впровадженням проекту, дозволив розпізнати можливі загрози та передбачити можливі збитки, що виникають, та розробити стратегії для їх уникнення чи зменшення. Цей етап став ключовим для врахування можливих труднощів та забезпечення більш ефективного процесу розробки та впровадження системи.

У третьому розділі було здійснено формулювання та наукове обґрунтування постановки завдання, деталізовано мету розробки та визначено призначення та можливі області використання системи. Також вивчались потенційні наслідки

впровадження системи аналізу даних для опублікування афіш, враховуючи різні сценарії її застосування.

У даному розділі була розроблена інформаційно-математична модель для вирішення проблеми завдання, а також створений відповідний алгоритм для її реалізації. Великий акцент був зроблений на обґрунтуванні вибору методів розв'язання, включаючи використані методології розробки. Результати цього розділу служать основою для подальшої практичної реалізації системи та її впровадженні.

У четвертому розділі було надано детальний опис програмного продукту у формі веб-сайту, що відповідає встановленому стандарту. Проведено загальний огляд реалізованого програмного продукту, розкрито його функціональне призначення, розглянуто логічну структуру та всі використані технічні засоби.

У п’ятому розділі здійснено розрахунок витрат на розробку та впровадження системи аналізу даних для опублікування афіш. Враховуються обчислення комплексного показника якості, експлуатаційних витрат, ціни споживання проектного рішення та показники економічної ефективності.

СПИСОК ЛІТЕРАТУРИ

1. Індустрія розваг [Електронний ресурс] – Режим доступу до ресурсу: https://leksika.com.ua/10440722/turizm/industriya_rozvag (дата звернення 07.10.2023)

2. Люшенко Л.А., Хіцко Я.В. Розробка та аналіз вимог до програмного забезпечення. Компоненти програмної інженерії. Курсове проєктування. Київ: КПІ 2020 (дата звернення 07.10.2023)

3. Кузьменко Б.В., Чайковська О.А. МОДЕЛЮВАННЯ СИСТЕМ (навчальний посібник) Київ – 2008 (дата звернення 07.10.2023)

4. Лобур М. В., Шварц М. Є., Стех Ю. В. Моделі і методи прогнозування рекомендацій для колаборативних рекомендаційних систем // Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі. – 2018. – № 901. – С. 68–75. (дата звернення 07.10.2023)

5. What is HTML? [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/standards/webdesign/htmlcss (дата звернення 07.10.2023)

6. Що таке HTML? Як створюють веб-сайти? [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/html/index.html#gsc.tab=0 (дата звернення 07.10.2023)

7. HTML5+CSS3 [Електронний ресурс] – Режим доступу до ресурсу: https://astwellsoft.com/uk/blog/tehnology/html5-css3.html (дата звернення 07.10.2023)

8. Сучасний підручник з JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://uk.javascript.info/ (дата звернення 07.10.2023)

9. What is TypeScript? [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/ (дата звернення 07.10.2023)

10. The modern web developer's platform [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/ (дата звернення 07.10.2023)

11. AngularJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/angular/ (дата звернення 07.10.2023)

12. Npm Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/bootstrap (дата звернення 07.10.2023)

13. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/ (дата звернення 07.10.2023)

14. Реліз Java 21: Long live a new LTS! [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/45335/ (дата звернення 07.10.2023)

15. The Java Language Specification, Third Edition. / James Gosling; Bill Joy, Guy Steele, Gilad Bracha, 2005. – 652 p. (дата звернення 07.10.2023)

16. Preparing for Spring Boot 3.0 [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/blog/2022/05/24/preparing-for-spring-boot-3-0 (дата звернення 07.10.2023)

17. Why Spring? [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/why-spring (дата звернення 07.10.2023)

18. Web MVC framework [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html (дата звернення 07.10.2023)

19. Spring Data [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-data (дата звернення 07.10.2023)

20. Spring Security [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-security (дата звернення 07.10.2023)

21. JUnit [Електронний ресурс] – Режим доступу до ресурсу: https://junit.org/junit5/ (дата звернення 07.10.2023)

22. What is a RESTful API? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/restful-api/?nc1=h_ls (дата звернення 07.10.2023)

23. The Main Features of MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/refman/8.0/en/features.html (дата звернення 07.10.2023)

24. Maven [Електронний ресурс] – Режим доступу до ресурсу: https://maven.apache.org/ (дата звернення 07.10.2023)

25. What is Maven: Here's What You Need to Know [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven (дата звернення 07.10.2023)

26. GitLab [Електронний ресурс] – Режим доступу до ресурсу: https://about.gitlab.com/ (дата звернення 07.10.2023)

27. ER-діаграма в нотації Чена [Електронний ресурс] – Режим доступу до ресурсу: https://stud.com.ua/93800/informatika/diagrama_notatsiyi_chena (дата звернення 07.10.2023)

28. Проектування БД [Електронний ресурс] – Режим доступу до ресурсу: https://rdb.dp.ua/uk/chapter_02 (дата звернення 07.10.2023)

29. Берко А.Ю., Литвин В. В., Василюк А.С, Верес О.М. МЕТОДИЧНІ ВКАЗІВКИ до виконання магістерських кваліфікаційних робіт для студентів освітньо-професійної програми "Аналіз даних (Data Science)" зі спеціальності 124 «Системний аналіз» другого (магістерського) рівня вищої освіти: Методичні вказівки. / Львів: 2021 (дата звернення 07.10.2023)

АНОТАЦІЯ

Демчук Ю.М., Лозинська О.В., (керівник). Система аналізу даних для опублікування афіш. Магістерська кваліфікаційна робота. – Національний університет «Львівська політехніка», Львів 2023.

Розширена анотація.

Використання інтернет технологій значно збільшується в усіх сферах людського життя, включаючи розважальну галузь. Однією з основних причин зростаючого інтересу маркетологів є легкість та зручність побудови та створення веб-ресурсів, маючи у наявності велику кількість інструментів, а також велика кількість аудиторії та відносна доступність вартості. За допомогою веб-ресурсів можна значно підвищити ефективність процесів реклами і продажу, і вже зараз це є стандартом у сучасному світі[1][2].

Сфера продажу квитків на різноманітні події вже давно є насиченою, і є кілька великих квиткових операторів у кожній країні, які користуються популярністю серед користувачів протягом тривалого часу. Запровадження карантинних обмежень додало труднощів у цій сфері, оскільки культурно-масові заходи були заборонені у більшості країн світу. Однак з вакцинацією і послабленням заходів безпеки ситуація полегшилась, і попит на розваги повертається після тривалого примусового «голодування».

Якщо у вас є бажання створити невеликий виступ для локального гурту або відвідати концерт улюбленого артиста, який оминув вашу країну, що робити. У першому випадку обмежені можливості фінансування не дають змоги ефективно привертати увагу аудиторії, тоді як у другому виникають труднощі з пошуком квитків на сайтах з незнайомою мовою.

Формування бачення. Покращення пошуку подій для користувачів.

Завдання роботи. Створення веб ресурсу для перегляду та опублікування афіш подій.

Актуальність роботи полягає у можливості створення власних оголошень, перегляду існуючих незалежно від організатора та країни, а також у можливості придбання квитків у верифікованих квиткових операторів.

Об'єкт дослідження. Процес автоматизації розміщення оголошення та побудови рекомендації.

Предмет дослідження. Методи і засоби аналізу даних про події, користувачів, а також розроблення системи аналізу для опублікування афіш.

Мета і *задача* дослідження полягає в створенні конкурентоздатної інтернаціональної платформи для розміщення афіш різноманітних подій - від невеликих акустичних вечорів до великих фестивалів. Основне завдання полягає в створенні зручного та ефективного способу перегляду і придбання квитків на події, які пропонуються різними квитковими операторами, усе це на єдиній онлайн-платформі. Такий підхід дозволить уникнути роздрібненості і запропонує користувачам зручний і однаковий досвід придбання квитків незалежно від масштабу події.

Завдання дослідження 1: Здійснити аналіз предметної області, а саме дослідити веб-ресурси квиткових операторів.

Завдання дослідження 2: Провести системний аналіз об'єкта дослідження.

Завдання дослідження 3: Моделювання бізнес процесів та вибір засобів реалізації.

Завдання дослідження 4: Створення системи аналізу, базуючись на визначеній концепції.

Інноваційність результатів роботи. Розроблена система надає можливість публікувати афіші про події усім користувачам, також надає зручний список рекомендацій.

У магістерській кваліфікаційній роботі була розроблена система аналізу даних для опублікування афіш. Робота включала аналіз предметної області, системний та технологічний аналіз. Проведено тестування та економічні розрахунки. На першому етапі визначено актуальність системи, проведено порівняльний аналіз конкурентів.

Розглянуто функціональність майбутньої системи та ідентифіковані критичні функції. Побудовано таблицю для порівняння аналогів з проектованою системою.

У другому розділі розроблено модель системних вимог та проведено аналіз потенційних ризиків. Цей етап був ключовим для врахування можливих труднощів та забезпечення ефективного процесу розробки та впровадження.

Третій розділ включає постановку завдання, наукове обґрунтування, вивчення наслідків впровадження системи та розробку інформаційно-математичної моделі.

У четвертому розділі детально описано програмний продукт - веб-сайт, його функціональність та структуру. Описано спосіб розгортання системи.

П'ятий розділ містить розрахунки витрат на розробку та впровадження системи, включаючи комплексний показник якості, експлуатаційні витрати та економічну ефективність.

Ключові слова – система, аналіз, веб, публікація, поширення, афіша, подія.

Перелік використаних літературних джерел.

1. Індустрія розваг [Електронний ресурс] – Режим доступу до ресурсу: https://leksika.com.ua/10440722/turizm/industriya_rozvag (дата звернення 07.10.2023)

2. Міхо О.І. Перспективи використання об’єктів індустрії розваг для розвитку туризму. Київ: Матеріали X аспірантських читань Київського університету туризму, економіки і права «Праксеологічні проблеми туризму в Україні»

ABSTRACT

Demchuk Y.M., Lozynska O.V., (supervisor). Data analysis system for publishing posters. – Lviv Polytechnic National University, Lviv, 2023.

Extended abstract.

The use of Internet technologies is significantly increasing in all aspects of human life, including the entertainment industry. One of the main reasons for the growing interest of marketers is the ease and convenience of building and creating web resources, given the availability of many tools and a considerable audience, along with the relative affordability of costs. Web resources can greatly enhance the efficiency of advertising and sales processes, making it a standard practice in the modern world [1][2].

The ticket sales sector for various events has long been saturated, with several major ticket operators in each country being popular among users for an extended period. The introduction of quarantine restrictions posed challenges to this sector, as cultural and mass events were prohibited in most countries worldwide. However, with the vaccination of the population and the reduction of security measures, the situation has changed, and the demand for entertainment is returning fast.

If you wish to organize a small performance for a local band or attend a concert of your favorite artist that bypassed your country, what should you do? In the first case, limited funding options hinder effective audience engagement, while in the second case, there are difficulties in finding tickets on websites with unfamiliar languages.

Vision Formation. Improving event search for users.

Research Task. Creating a web platform for viewing and publishing event posters.

The relevance of the study lies in the ability to create personal announcements, view existing ones regardless of the organizer and country, as well as the ability to purchase tickets from verified ticket operators. The research object is the process of automating announcement placement and recommendation building. The research subject is methods and means of analyzing data about events, users, and the development of a system for publishing event posters.

The goal and objective of the study are to create a competitive international platform for posting posters of various events, from small acoustic evenings to large festivals. The main task is to create a convenient and efficient way to view and purchase tickets for events offered by various ticket operators, all on a unified online platform. This approach will avoid fragmentation and offer users a convenient and consistent ticket-purchasing experience regardless of the event's scale.

Research Task 1: Analyze the subject area, specifically study ticket operators' websites.

Research Task 2: Conduct a systematic analysis of the research object.

Research Task 3: Model business processes and choose implementation tools.

Research Task 4: Create an analysis system based on the defined concept.

Innovativeness of the research results: The developed system allows users to publish event posters and provides a convenient list of recommendations.

In this master's qualification work, a data analysis system for publishing event posters was developed. The work included an analysis of the subject area, systematic and technological analysis, testing, and economic calculations. The relevance of the system was determined, and a comparative analysis of competitors was conducted in the first stage. The functionality of the future system was considered, and critical functions were identified. A table for comparing analogs with the designed system was built.

In the second chapter, a model of system requirements was developed, and an analysis of potential risks was conducted. This stage was crucial for anticipating difficulties and ensuring an effective development and implementation process.

The third chapter includes the task formulation, scientific justification, the study of the consequences of system implementation, and the development of an information-mathematical model.

The fourth chapter describes the software product - the website, its functionality, and its structure. The deployment method of the system is also described.

The fifth chapter contains calculations of costs for the development and implementation of the system, including the overall quality index, operating costs, and economic efficiency.

Keywords: system, analysis, web, publication, distribution, poster, event.

List of References:

1. Entertainment Industry [Electronic resource] – Access mode: https://leksika.com.ua/10440722/turizm/industriya_rozvag (accessed on 07.10.2023)

2. Mikho O.I. Prospects for using entertainment industry objects for tourism development. Kyiv: Materials of the 10th graduate readings of Kyiv University of Tourism, Economics, and Law "Praxeological problems of tourism in Ukraine."

ДОДАТКИ

User.java

```
package com.nulp.hapHere.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.util.Objects;

@Getter
@Setter
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_user")
    private Integer idUser;
    @Basic
    @Column(name = "username")
    private String username;
    @Basic
    @Column(name = "first_name")
    private String firstName;
    @Basic
    @Column(name = "last_name")
    private String lastName;
    @Basic
```

```
@Column(name = "email")
private String email;
@Basic
@Column(name = "phone")
private String phone;
@Basic
@Column(name = "password")
private String password;
@Column(name = "role")
@Enumerated(value = EnumType.STRING)
private Role role = Role.USER;
@Basic
@Column(name = "active", columnDefinition = "boolean default true")
private Boolean active = true;
@Basic
@Column(name = "banned", columnDefinition = "boolean default false")
private Boolean banned = false;

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    User user = (User) o;
    return Objects.equals(idUser, user.idUser) &&
            Objects.equals(firstName, user.firstName) &&
            Objects.equals(lastName, user.lastName) &&
            Objects.equals(email, user.email) &&
            Objects.equals(phone, user.phone) &&
            Objects.equals(password, user.password) &&
```

```
            Objects.equals(banned, user.banned);
    }

    @Override
    public int hashCode() {
        return Objects.hash(idUser, firstName, lastName, email, phone, password, banned);
    }

    @Override
    public String toString() {
        return "User{" +
                "idUser=" + idUser +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
                ", password='" + password + '\'' +
                ", role=" + role +
                ", active=" + active +
                ", banned=" + banned +
                '}';
    }
}
```

Role.java

```
package com.nulp.hapHere.entity;

public enum Role {
    USER,
    MODERATOR,
```

```
    ADMIN;
}
```

Post.java

```
package com.nulp.hapHere.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.sql.Timestamp;
import java.util.Objects;

@Getter
@Setter
@Entity
public class Post {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_post")
    private Integer idPost;
    @Basic
    @Column(name = "title")
    private String title;
    @Basic
    @Column(name = "text")
    private String text;
    @Basic
    @Column(name = "img")
    private String img;
```

```
@Basic
@Column(name = "category")
private String category;
@Basic
@Column(name = "date")
private Timestamp date;
@ManyToOne
@JoinColumn(name = "author", referencedColumnName = "id_user", nullable = false)
private User author;

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Post post = (Post) o;
    return Objects.equals(idPost, post.idPost) &&
            Objects.equals(title, post.title) &&
            Objects.equals(img, post.img) &&
            Objects.equals(category, post.category) &&
            Objects.equals(date, post.date);
}

@Override
public int hashCode() {
    return Objects.hash(idPost, title, img, category, date);
}

@Override
public String toString() {
```

```
        return "Post{" +
                "idPost=" + idPost +
                ", title='" + title + '\'' +
                ", category='" + category + '\'' +
                '}';
    }
}
```

CreatedPost.java

```
package com.nulp.hapHere.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.util.Objects;

@Getter
@Setter
@Entity
@Table(name = "created_post", schema = "mydb")
public class CreatedPost {
    @Id
    @Column(name = "post")
    private Integer postId;
    @Basic
    @Column(name = "moderated")
    private Boolean moderated;
    @Basic
    @Column(name = "edited")
```

```
    private Boolean edited;
    @Basic
    @Column(name = "moderated_by")
    private Integer moderated_by;
    @OneToOne
    @JoinColumn(name = "post", referencedColumnName = "id_post", nullable = false)
    private Post post;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        CreatedPost that = (CreatedPost) o;
        return Objects.equals(postId, that.postId) &&
                Objects.equals(moderated, that.moderated) &&
                Objects.equals(edited, that.edited);
    }

    @Override
    public int hashCode() {
        return Objects.hash(postId, moderated, edited);
    }


}
```