

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСЕРВІСУ ДЛЯ ПОШИРЕННЯ
КУЛЬТУРНИХ ПОДІЙ МІСТА

SOFTWARE IMPLEMENTATION OF A WEB SERVICE FOR PROMOTING
CULTURAL EVENTS IN THE CITY

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

БОГАЧУК А.М.

(ПРИЗВИЩЕ та ініціали)

Керівник КАПЛІЄНКО Т.І.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БОГАЧУКА Антона Максимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація вебсервісу для поширення культурних подій міста.

Software Implementation of a Web Service for Promoting Cultural Events in the City

керівник проєкту (роботи) к.т.н., доцент, КАПЛІЄНКО Тетяна Ігорівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та дослідження програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КАПЛІЄНКО Т.І., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Антон БОГАЧУК
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Тетяна КАПЛІЄНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 95 с., 4 табл., 19 рис., 2 дод., 21 джерел.

ВЕБСАЙТ, КУЛЬТУРНІ ПОДІЇ, NEXTJS, NESTJS, TYPESCRIPT.

Об'єкт дослідження – процес поширення інформації про культурні події міста та підтримки їх проведення.

Предмет дослідження – вебсайт для поширення культурних подій у місті.

Мета роботи – створення вебсайту для поширення та підтримки культурних подій міста.

Матеріали, методи та технічні засоби: дані про культурні події міста, об'єктно-орієнтований підхід до програмування, мова програмування TypeScript, фреймворки NextJS та NestJS, хостинг провайдер для розгортання сайту в мережі інтернет.

Результати. В результаті роботи був розроблений та запущений вебсайт, який надає користувачам зручну платформу для огляду та реєстрації на культурні події міста.

Висновки. Розроблений вебсайт є ефективним інструментом для поширення культурних подій міста та забезпечення зручного доступу до них для мешканців. Він дозволяє підвищити інтерес та участь у культурних заходах, сприяючи розвитку культурної сфери міста.

Галузь використання – вебплатформа для організації та поширення культурних подій у місті.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 95 pages, 4 tables, 19 figures, 2 appendixes, 21 sources.

WEBSITE, CULTURAL EVENTS, NEXTJS, NESTJS, TYPESCRIPT.

The object of research is the process of disseminating information about the city's cultural events and supporting their holding.

The subject of research is a website for the dissemination of cultural events in the city.

The purpose of the work is creation of a website to promote and support cultural events of the city.

Materials, methods, and technical means: data about cultural events of the city, object-oriented programming, TypeScript programming language, NextJS and NestJS frameworks, hosting provider for deploying the site on the Internet.

Results. As a result of the work, a website was developed and launched, which provides users with a convenient platform for viewing and registering for cultural events in the city.

Conclusions. The developed website is an effective tool for disseminating the city's cultural events and providing convenient access to them for residents. It allows to increase interest and participation in cultural events, contributing to the development of the city's cultural sphere.

Field of use - a web platform for the organization and distribution of cultural events in the city.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	9
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	12
1.1 Огляд існуючих рішень для поширення культурних подій	12
1.2 Аналіз вимог до вебсайту для поширення культурних подій міста	16
1.3 Висновки до розділу	17
2 МАТЕРІАЛИ І МЕТОДИ.....	18
2.1 Дані про культурні події міста.....	18
2.2 Технічні специфікації.....	18
2.2.1 Мова програмування	18
2.2.2 Об'єктно-орієнтоване програмування	21
2.2.3 Фреймворки	22
2.3 Висновки до розділу	26
3 ОПИС ПРОГРАМИ.....	28
3.1 Загальні вимоги до архітектури програми	28
3.2 Архітектура застосунку.....	29
3.3 Структура клієнтської частини.....	31
3.3.1 Діаграма прецедентів	32
3.4 Структура серверної частини.....	34
3.5 Структура бази даних.....	36
3.6 Функціонування програми.....	37
3.6.1 Головна сторінка.....	37
3.6.2 Сторінка детальної інформації про подію.....	39

3.6.3 Сторінка детальної інформації про локацію	40
3.6.4 Сторінка авторизації.....	41
3.6.5 Панель адміністратора	42
3.6.6 Обробка даних на сервері.....	43
3.7 Висновки до розділу	43
4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ВЕБСАЙТУ	45
4.1 Опис застосування вебсайту	45
4.1.1 Призначення програми.....	45
4.1.2 Обмеження для застосування.....	46
4.1.3 Мінімальна конфігурація технічних засобів	46
4.2 Методика та результати тестування	46
4.2.1 Методика тестування.....	46
4.2.2 Результати тестування.....	47
4.3 Експериментальні дослідження вебсайту	49
4.3.1 Загальна схема експериментів	50
4.3.2 Використане устаткування і вихідні дані.....	50
4.3.3 Параметри запуску методів і програмних засобів	50
4.3.4 Результати експериментальних досліджень	51
4.3.5 Аналіз отриманих результатів	52
4.4 Висновки до розділу	52
ВИСНОВКИ	54
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ.....	56
ДОДАТОК А Текст програми	58

ДОДАТОК Б Слайди презентації	85
------------------------------------	----

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	- Application Programming Interface;
CSRF	- Cross-Site Request Forgery;
FCP	- First Contentful Paint;
HTTP	- HyperText Transfer Protocol;
HTTPS	- HyperText Transfer Protocol Secure;
JSON	- JavaScript Object Notation;
LCP	- Largest Contentful Paint;
ORM	- Object-Relational Mapping;
SEO	- Search Engine Optimization;
SQL	- Structured Query Language;
SSL	- Secure Sockets Layer;
SSR	- Server Side Rendering;
TBT	- Total Blocking Time;
UI	- User Interface;
XSS	- Cross-Site Scripting;
ООП	- об'єктно-орієнтоване програмування.

ВСТУП

У сучасному світі інформаційні технології відіграють важливу роль у всіх сферах життя, включаючи культурну сферу. Одним з інноваційних підходів до сприяння культурному розвитку міст є використання вебтехнологій для поширення та підтримки культурних подій. Вебсайти не тільки покращують доступність інформації про культурні заходи, але й сприяють залученню мешканців до культурного життя міста.

Розробка ефективного вебсайту для підтримки культурних подій стає актуальною задачею, яка потребує врахування багатьох факторів, таких як зручність користування, інтеграція з соціальними мережами та інше. Сучасні вебтехнології дозволяють створювати динамічні та інтерактивні платформи, що можуть задовольнити потреби різних користувачів.

Метою розробки вебсайту для підтримки культурних подій міста є створення зручної та ефективної платформи, яка дозволить мешканцям легко отримувати інформацію про культурні заходи, брати участь у них та підтримувати культурне життя міста.

Об'єктом дослідження є процес поширення та підтримки культурних подій у міському середовищі. Предметом дослідження виступає вебсайт, що реалізує функції інформування про культурні події, інтеграції з соціальними мережами та інших корисних функцій.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- аналіз існуючих рішень у сфері підтримки культурних подій через вебсайти;
- розробка технічного завдання та визначення функціональних вимог до вебсайту;
- вибір інструментарію та платформ для реалізації вебсайту;
- проєктування архітектури та розробка вебсайту;
- тестування вебсайту та виявлення можливих проблем і недоліків у її роботі.

Завершенням даної роботи стане створення вебсайту, який допоможе зробити культурне життя міста більш доступним і цікавим для його мешканців.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих рішень для поширення культурних подій

У сучасному світі інформаційні технології значно впливають на культурну сферу, забезпечуючи нові можливості для організації, поширення та підтримки культурних заходів. Інтернет та вебтехнології дозволяють значно розширити аудиторію культурних подій, зробивши інформацію доступною для широкого кола користувачів. Це особливо актуально у міському середовищі, де щоденно проводяться численні культурні заходи різного масштабу та спрямованості.

Вебсайти, присвячені культурним подіям, виконують кілька важливих функцій: вони не лише інформують громадськість про майбутні заходи, але й дозволяють купувати квитки онлайн, ділитися інформацією в соціальних мережах, залишати відгуки та коментарі. Такі платформи сприяють підвищенню зацікавленості у культурному житті міста, залучаючи більше відвідувачів до подій.

З розвитком інформаційних технологій з'являються нові інструменти та рішення для поширення культурних подій. Від традиційних афіш на міських порталах до інтегрованих мобільних застосунків з функціями рекомендацій та персоналізації — всі ці інструменти спрямовані на покращення доступності та зручності користувачів. Однак, кожен з цих рішень має свої особливості та може бути більш або менш підходящим для різних типів подій та аудиторій.

Далі розглянемо кілька існуючих вебсайтів, що пропонують послуги з поширення інформації про культурні події в Україні. Серед них — concert.ua, karabas та musin, кожен з яких має свої унікальні функції та підходи до організації роботи.

Concert.ua - це одна з найбільших українських платформ для продажу квитків на різні культурні заходи, такі як вистави, концерти, фестивалі та інші події. Основні функції сайту включають можливість онлайн-купівлі квитків з вибором місць, зручний каталог подій з фільтрами за датами, жанрами та

місцями проведення, а також надання детальної інформації про кожну подію (рис. 1.1) [1].

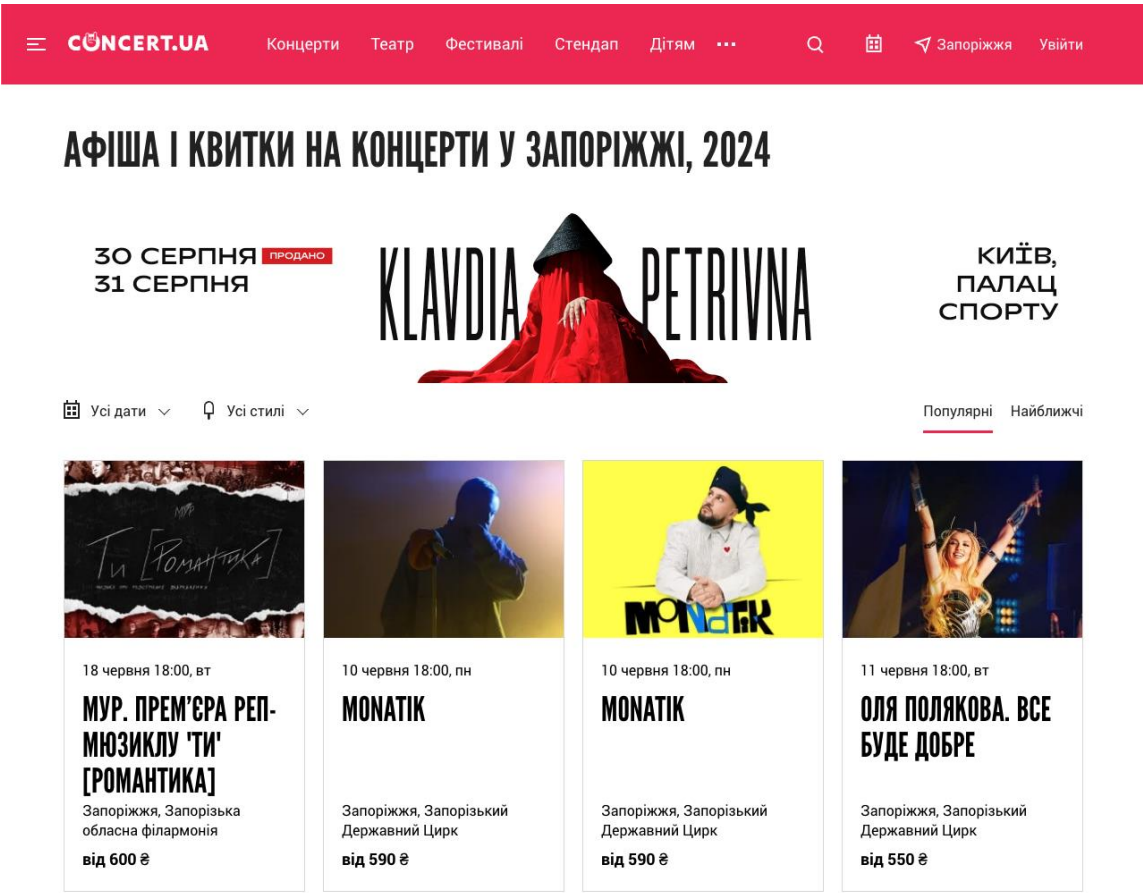


Рисунок 1.1 – Concert.ua

Karabas - це популярний в Україні онлайн-сервіс, який не лише надає інформацію про культурні події, але й допомагає з їх організацією та продажем квитків. Сервіс відомий своїм широким спектром подій, від концертів та театральних вистав до фестивалів та майстер-класів. Крім того, Karabas часто співпрацює з місцевими артистами та культурними закладами, що дозволяє пропонувати ексклюзивні заходи. Інтерфейс вебсайту простий та зручний у використанні, що робить процес пошуку та придбання квитків максимально комфортним для користувачів (рис. 1.2) [2].

KARABAS

ЖИТТЯ ТРИВАЄ!

Пошук

Запоріжжя

UA

увійти

Клуби Запоріжжі X

ОЧИСТИТИ ФІЛЬТР

Квитки на концерт у клубі 2024

Червень 2024

10 Пн

червня '2024

10.06

ARTERIALS

MONATIK

FCA

32 ПИЛІЖСЬКА

концерти | клуби

Запоріжжя, 10 червня 2024, 18:00

Запорізький Державний Цирк

590 - 1390 грн

КУПИТИ

Липень 2024

5 Пт

липня '2024

05.07

АРСЕН МІРЗОЯН

СОЛІЦЕ

АКУСТИКА

концерти | клуби

Запоріжжя, 5 липня 2024, 18:00

Запорізька обласна філармонія

450 - 750 грн

КУПИТИ

Рисунок 1.2 - Karabas

Musin.zp.ua - це провідна квиткова каса в Запоріжжі, що надає широкий вибір офіційних квитків на різноманітні культурні, розважальні та спортивні події. Забезпечуючи зручний онлайн-сервіс з інтуїтивно зрозумілим інтерфейсом та функціоналом сортування, фільтрації та пошуку, вони гарантують надійність та доступність квитків, а також надають клієнтам ряд додаткових переваг (рис. 1.3) [3].

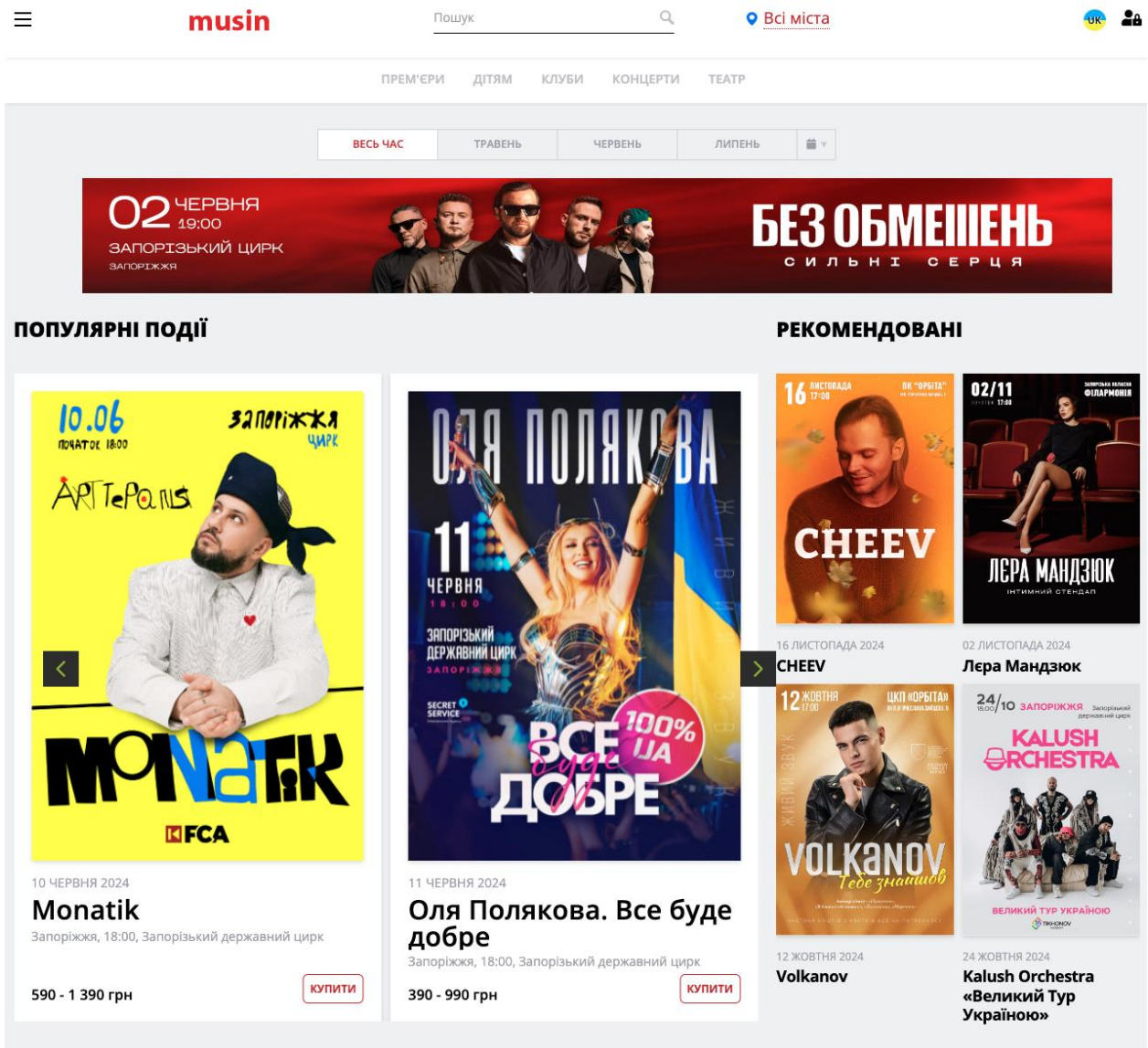


Рисунок 1.3 - Musin

Проведений огляд існуючих рішень для поширення культурних подій показав, що на українському ринку представлені різноманітні вебсайти, які спеціалізуються на культурних подіях. Однак, їх функціональність переважно зосереджена на продажу квитків та анонсах подій, а не на інтерактивній підтримці культурних ініціатив.

Вебсайти, такі як concert.ua, karabas та musin, пропонують зручні інструменти для пошуку та придбання квитків на культурні заходи. Вони надають детальну інформацію про кожну подію, фільтри для сортування подій за різними параметрами та можливість вибору місць. Проте, ці сайти не мають

інтерактивних функцій для підтримки та популяризації культурних ініціатив серед громадськості.

Отже, незважаючи на наявність корисних функцій продажу квитків та анонсів подій, розглянуті вебсайти не пропонують інтерактивних рішень для підтримки культурних ініціатив. Це свідчить про актуальність розробки спеціалізованого вебсайту, який би дозволяв користувачам не лише отримувати інформацію про культурні події, але й активно брати участь у їх підтримці та популяризації.

1.2 Аналіз вимог до вебсайту для поширення культурних подій міста

Розробка вебсайту для поширення культурних подій міста включає в себе аналіз потреб користувачів і ринку, створення технічного завдання з детальним описом вимог, а також вибір технологій для розробки та проєктування системи. Процес передбачає визначення основних функцій, які потрібно реалізувати, і розробку інтерфейсу, що дозволить користувачам легко отримувати інформацію про події (табл. 1.1).

Таблиця 1.1 – Вимоги до системи

Вимога	Опис
Інтерфейс користувача	Інтуїтивно зрозумілий інтерфейс, що включає афішу подій, сторінки з детальною інформацією об подіях та локаціях
Функціональність	Інформування про події, надання детальної інформації про локації
Платформи	Доступність на будь-якій платформі, що має вихід в мережу інтернет
Масштабованість	Можливість додавання нових функцій та підтримка великої кількості користувачів

Ці вимоги є основою для детального планування і реалізації проєкту, забезпечуючи відповідність реальним потребам користувачів та високу якість виконання.

1.3 Висновки до розділу

Проведений аналіз існуючих рішень для поширення культурних подій міста показав, що на ринку представлені численні інструменти, проте вони мають свої обмеження. Вебсайти, такі як concert.ua, karabas та musin, успішно виконують базові функції інформування користувачів про культурні заходи та продажу квитків. Однак, ці платформи часто не охоплюють повного спектру можливостей, які можуть бути корисними для кінцевих користувачів, зокрема, можливості користувачам самостійно пропонувати існуючі події в їх містах.

На основі вивчених вимог було сформульовано ключові функції, які повинні бути реалізовані у вебсайті для поширення культурних подій міста. Серед них: інтуїтивно зрозумілий інтерфейс, що дозволяє легко знаходити інформацію про події та локації; доступність налюбій платформі, що має вихід в мережу інтернет; можливість додавання нових функцій та підтримка великої кількості користувачів.

Ретельно розроблене технічне завдання з детальним описом цих вимог є критично важливим для успішного планування і реалізації проєкту. Виконання всіх зазначених вимог забезпечить відповідність вебсайту реальним потребам користувачів та високу якість його виконання, що в свою чергу сприятиме підвищенню інтересу до культурних подій міста та активній участі мешканців у культурному житті. Таким чином, розроблений вебсайт стане важливим інструментом для підтримки та розвитку культурних ініціатив у міській спільноті.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Дані про культурні події міста

Для розробки вебсайту було використано широкий спектр даних про культурні події міста. Ці дані включають в себе інформацію про різноманітні культурні події, такі як фестивалі, концерти, виставки, театральні постановки тощо. Дані були зібрані з різних джерел, таких як офіційні вебсайти організаторів подій, соціальні медіа, партнерські організації, а також бази даних місцевих культурних установ та мережеві інформаційні ресурси. Отримані дані включають в себе назву події, опис, дату та час проведення, місце проведення, зображення та інші деталі, необхідні для користувачів для прийняття рішення про участь у події [4]-[8].

2.2 Технічні специфікації

2.2.1 Мова програмування

Зважаючи на різноманітність мов програмування та їхній вплив на розробку програмного забезпечення, у цьому розділі проведемо аналіз різних мов програмування з орієнтацією на вибір оптимального варіанту для реалізації вебсайту для поширення культурних подій міста. Для цього розглянемо характеристики таких мов, як TypeScript, JavaScript, Python та Java, та порівняємо їхні переваги та недоліки (табл. 2.1). Це допоможе зробити обґрунтований вибір технологій для подальшої розробки та підтримки вебсайту.

Таблиця 2.1 – Порівняння мов програмування

Мова програмування	Опис	Переваги	Недоліки
1	2	3	4
Java	Java - це об'єктно-орієнтована мова програмування, яка використовується для розробки різноманітних застосунків, включаючи вебзастосунки, мобільні застосунки та великі підприємницькі системи [9]	Кросплатформенність: програми, написані на Java, можуть запускатися на будь-якому пристрої, який підтримує віртуальну машину Java; велика спільнота та багато інструментів розробки; висока продуктивність та надійність	Більш висока складність вивчення порівняно з іншими мовами; більший обсяг вимог до пам'яті та ресурсів

Продовження таблиці 2.1

1	2	3	4
Python	Python є інтерпретованою мовою програмування, яка має простий та зрозумілий синтаксис. Вона використовується для різних цілей, включаючи веброзробку, наукові дослідження, аналіз даних тощо [10]	Простий та читабельний синтаксис; велика кількість бібліотек та фреймворків для різних цілей; можливість використання у різних областях програмування	Помірна швидкодія у порівнянні з компільованими мовами; відсутність строгого контролю типів може призвести до помилок у великих проєктах.
JavaScript	JavaScript є широко використовуваною мовою програмування, що використовується для розробки вебсайтів та вебзастосунків. Вона є основною мовою веброзробки та має велику спільноту розробників [11]	Широке використання та підтримка; низький поріг входу для новачків; швидка реалізація та виконання коду	Відсутність статичної типізації, що може призвести до помилок у великих проєктах; час виконання може бути повільнішим у порівнянні з мовами, які компілюються в JavaScript

Продовження таблиці 2.1

1	2	3	4
TypeScript	TypeScript - це мова програмування, що є сучасним розширенням JavaScript, яке додає статичну типізацію та інші нові функції до мови. Вона компілюється в чистий JavaScript для виконання в браузерах або на сервері [12]	Статична типізація, яка полегшує виявлення помилок під час розробки; багато функцій, які покращують продуктивність розробки; сумісність з існуючим кодом JavaScript	Потреба у компіляції до JavaScript; навчання типів та інших особливостей може зайняти деякий час

Після аналізу різних мов програмування, включаючи TypeScript, JavaScript, Python та Java, можна прийти до висновку, що TypeScript є оптимальним варіантом для реалізації вебсайту. Статична типізація, багато функцій, які покращують продуктивність розробки, та сумісність з існуючим кодом на JavaScript роблять TypeScript привабливим вибором при створенні великого проєкту. Такий вибір допоможе забезпечити якісний та ефективний розвиток та підтримку вебсайту у майбутньому.

2.2.2 Об'єктно-орієнтоване програмування

В розробці вебсайту було використано парадигму об'єктно-орієнтованого програмування (ООП), що дозволяє структурувати код у вигляді об'єктів, що взаємодіють між собою. Це суттєво полегшує процес розробки та супроводу програмного забезпечення, оскільки код може бути

організований за логічними моделями, що відображають реальні об'єкти та їх взаємодію в програмному середовищі. Об'єктно-орієнтоване програмування також забезпечує високу перевикористовуваність коду та спадковість, що дозволяє ефективно використовувати код для різних компонентів вебсайту [13].

2.2.3 Фреймворки

Важливість правильного вибору фреймворків полягає у забезпеченні ефективності, продуктивності та масштабованості розробки. Для цього проведемо аналіз і порівняння різних варіантів для клієнтської (табл. 2.2) та серверної (табл. 2.3) частин, враховуючи їх особливості, переваги та недоліки. Результати цього аналізу допоможуть зробити обґрунтований вибір фреймворків для подальшої розробки та успішної реалізації проєкту.

Таблиця 2.2 – Порівняння фреймворків для клієнтської частини

Фреймворк	Опис	Переваги	Недоліки
1	2	3	4
React.js	Бібліотека для створення користувацького інтерфейсу, яка використовується для розробки інтерактивних вебсайтів [14]	Висока швидкодія; велика спільнота розробників; широкий вибір розширень; вбудована підтримка TypeScript	Проблема із SEO оптимізацією
Angular	Фреймворк для розробки вебзастосунків, що використовується для розробки високопродуктивних та масштабованих користувацьких інтерфейсів [15]	Вбудований механізм маршрутизації; вбудована підтримка TypeScript	Високий поріг входу для новачків; складніше вивчення порівняно з React та Vue
Vue.js	Сучасний JavaScript-фреймворк для створення користувацьких інтерфейсів [16]	Простий та зрозумілий синтаксис; вбудована підтримка TypeScript; швидко створює інтерактивні вебзастосунки	Менша екосистема порівняно з React та Angular

Продовження таблиці 2.2

1	2	3	4
Next.js	Реактивний фреймворк для рендерингу сторінок та розробки універсальних вебзастосунків, побудований на бібліотеці React [17]	Висока продуктивність при рендерингу сторінок; просте використання та інтеграція з React-компонентами; велика спільнота розробників від ком'юніті React; вбудований механізм маршрутизації; відсутність проблеми із SEO оптимізацією	Потреба в налаштуванні серверної частини застосунку для підтримки рендерингу на стороні сервера

Після ретельного аналізу та порівняння різних фреймворків для клієнтської частини, включаючи React.js, Angular та Vue.js, прийнято рішення вибрати Next.js для реалізації вебзастосунку. Next.js відзначається високою продуктивністю, швидкодією та ефективним рендерингом сторінок. Просте використання та інтеграція з React-компонентами дозволять нам ефективно створювати інтерактивні вебзастосунки. Також, враховуючи його підтримку рендерингу на стороні сервера, можливо забезпечити швидке завантаження та оптимальну індексацію вебсторінок пошуковими системами. Враховуючи ці переваги, Next.js стає оптимальним вибором для проєкту, що дозволить успішно реалізувати поставлені цілі та завдання.

Таблиця 2.3 – Порівняння фреймворків для серверної частини

Фреймворк	Опис	Переваги	Недоліки
1	2	3	4
Express.js	Швидкий та гнучкий фреймворк для написання серверної частини на Node.js [18]	Простий синтаксис; широкі можливості налаштування; велика спільнота розробників	Вимагає написання великої кількості коду для розробки деяких функцій
NestJS	Прогресивний фреймворк для Node.js, що використовує TypeScript [19]	Використання TypeScript; підтримка підходів та патернів проєктування, таких як Dependency Injection та модульність; велика кількість допоміжних розширень; легка вивчені документація; велика спільнота	Вимагає більше ресурсів та пам'яті порівняно з іншими фреймворками

Продовження таблиці 2.3

1	2	3	4
Koa.js	Коа — це вебплатформа, розроблена командою, що стоїть за Express [20]	Модульний та легкий у використанні; невеликий розмір та швидкодія	Більш низький рівень абстракції порівняно з NestJS

Після ретельного порівняльного аналізу різних фреймворків для серверної частини застосунку, таких як Express.js, Koa.js та NestJS, було прийнято рішення вибрати NestJS. NestJS відзначається використанням TypeScript, що сприяє підвищенню якості та надійності коду. Підтримка підходів та патернів проектування, таких як Dependency Injection та модульність, забезпечить масштабованість та зручність розробки. Також, NestJS володіє вбудованими інструментами для роботи з асинхронним програмуванням, що сприяє розробці швидких та ефективних вебзастосунків. З урахуванням цих факторів, можна бути переконаним, що вибір NestJS є оптимальним для досягнення цілей та завдань у реалізації серверної частини проєкту.

2.3 Висновки до розділу

Було ретельно розглянуто та обґрунтовано вибір ключових технічних аспектів проєкту. У контексті вибору мови програмування було розглянуто переваги TypeScript, який відомий своєю надійністю та ефективністю завдяки визначеним типам, що робить розробку застосунку ефективнішим та швидшим. Обговорення ООП відображало його значення для створення чистого та організованого коду. А вибір фреймворків, таких як Next.js для

клієнтської частини та NestJS для серверної частини, був обґрунтований їхніми перевагами та спрямований на досягнення визначених цілей. Завдяки цьому аналізу можна бути впевненим, що обрані технічні рішення будуть забезпечувати ефективну реалізацію проєкту з високою якістю та продуктивністю.

3 ОПИС ПРОГРАМИ

3.1 Загальні вимоги до архітектури програми

Проектування та розробка вебсайту для поширення культурних подій міста потребує ретельного планування архітектури системи, щоб забезпечити її функціональність, надійність та зручність для користувачів. Архітектура програми повинна відповідати сучасним вимогам та стандартам розробки вебзастосунків.

Першим етапом розробки є визначення загальних вимог до архітектури програми, що включає як функціональні, так і нефункціональні аспекти. Це дозволить сформулювати конкретні цілі, яким повинна відповідати система, а також забезпечити її масштабованість, продуктивність та безпеку. Функціональні вимоги охоплюють основні можливості програми, такі як управління подіями та локаціями. Нефункціональні вимоги зосереджуються на забезпеченні високої продуктивності, надійності та захисту даних.

а) функціональні вимоги:

1) керування подіями:

— програма повинна дозволяти додавання, редагування та видалення культурних подій;

— кожна подія повинна містити детальну інформацію: дата, час, місце проведення, опис, вартість;

2) керування локаціями:

— програма повинна дозволяти додавання, редагування та видалення локацій;

— кожна локація повинна містити детальну інформацію: назву, опис, адресу, посилання на вебсайт, зображення;

б) нефункціональні вимоги:

1) продуктивність:

- програма повинна швидко завантажуватися і працювати без значних затримок навіть при великій кількості одночасних користувачів;

- оптимізація запитів до бази даних для забезпечення швидкого доступу до інформації;

2) масштабованість:

- архітектура програми повинна бути розроблена з урахуванням можливості подальшого розширення функціоналу без значних змін у коді;

- можливість підтримки великої кількості подій та користувачів;

3) безпека:

- захист даних користувачів, включаючи особисту інформацію та дані про події;

- реалізація аутентифікації та авторизації користувачів для забезпечення доступу до адміністративних функцій лише для авторизованих осіб;

- впровадження SSL для забезпечення захищеного з'єднання між клієнтом та сервером;

4) кросплатформенність:

- програма повинна коректно працювати на різних операційних системах та браузерях.

Ці вимоги служать основою для детального планування та реалізації проєкту, забезпечуючи якість та відповідність реальним потребам користувачів.

3.2 Архітектура застосунку

Архітектура застосунку побудована з використанням модульної структури, що забезпечує високу гнучкість, масштабованість та підтримку.

Вона включає три основні компоненти: клієнтський застосунок, серверний застосунок і базу даних. Кожен з цих компонентів відіграє ключову роль у функціонуванні системи.

Клієнтський застосунок розроблений з використанням фреймворку Next.js, який забезпечує рендеринг на стороні сервера (SSR) і генерацію статичних сторінок. Це дозволяє значно підвищити продуктивність та покращити SEO оптимізацію. Клієнтський застосунок відповідає за взаємодію з користувачем, відображення інтерфейсу та передачу даних до серверного застосунку. Фреймворк Next.js дозволяє створювати динамічні вебсторінки, що завантажуються швидко і забезпечують користувачам високий рівень зручності.

Серверний застосунок реалізований на базі фреймворку NestJS, який побудований на Node.js і використовує TypeScript для забезпечення безпеки та масштабованості коду. NestJS надає модульну архітектуру, яка полегшує розробку та підтримку великих додатків. Серверний застосунок обробляє запити від клієнтського застосунку, взаємодіє з базою даних і повертає необхідні дані у форматі JSON. Завдяки використанню NestJS, серверний застосунок може легко розширюватись та інтегруватись з іншими сервісами.

База даних представлена MongoDB, що є NoSQL базою даних, яка забезпечує високу продуктивність та гнучкість у зберіганні даних. MongoDB зберігає дані у форматі документів, що дозволяє легко масштабувати та управляти великими обсягами даних. Використання MongoDB забезпечує ефективне зберігання та швидкий доступ до даних, необхідних для роботи застосунку.

На рисунку 3.1 представлено схематичне зображення архітектури застосунку, яке показує взаємодію між клієнтським застосунком (Next.js), серверним застосунком (NestJS) і базою даних (MongoDB). JSON використовується як формат передачі даних між клієнтським та серверним застосунками, тоді як запити та відповіді між сервером і базою даних здійснюються у вигляді запитів та даних.

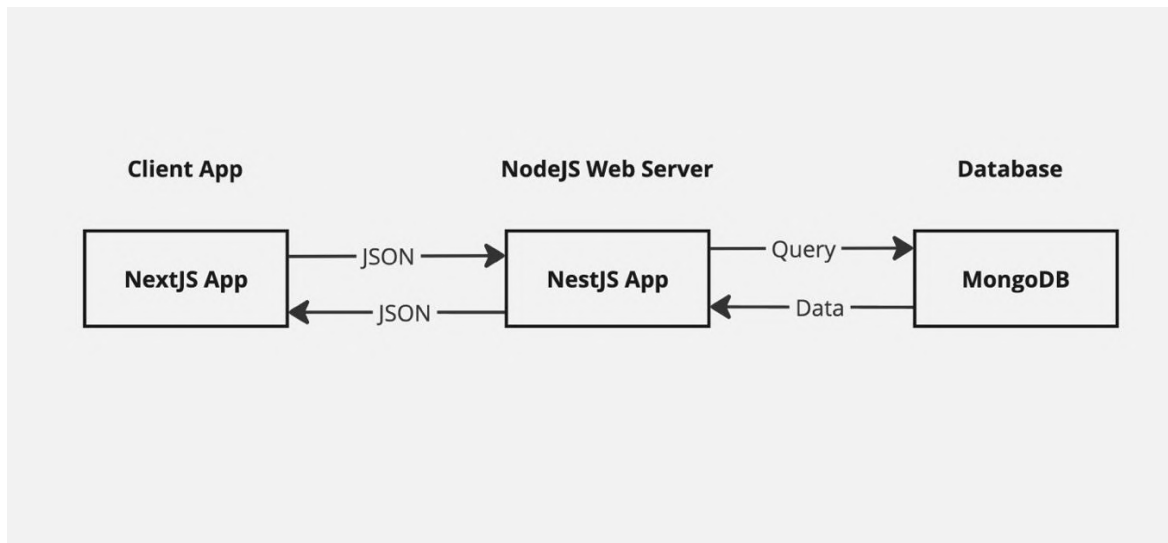


Рисунок 3.1 – Схема архітектури вебзастосунку

Ця архітектура забезпечує ефективну обробку запитів, високу продуктивність та масштабованість системи, що є критично важливими для успішного функціонування вебзастосунку.

3.3 Структура клієнтської частини

Клієнтська частина вебзастосунку реалізована за допомогою фреймворку Next.js, який забезпечує швидкий рендеринг, високу продуктивність та оптимізацію для пошукових систем (SEO). Основні компоненти клієнтської частини включають:

- головна сторінка, що відображає останні події та найпопулярніші локації;
- сторінка детальної інформації про подію, що відображає повну інформацію про конкретну подію, включаючи опис, дату, місце проведення та ціну;
- сторінка детальної інформації про локацію, що відображає інформацію про конкретну локацію, включаючи адресу, опис, зображення та

посилання на Google Maps, а також показує всі події, що відбуваються у цій локації;

- сторінка авторизації, що забезпечує захищений доступ до адміністративної панелі для адміністраторів та організаторів;
- адміністративна панель - інтерфейс для адміністраторів і організаторів, який дозволяє керувати подіями, локаціями, містами та користувачами.

На рисунку 3.2 показано структуру клієнтської частини, яка включає головну сторінку, сторінки з детальною інформацією про події та локації, а також сторінку авторизації та адміністративної панелі.

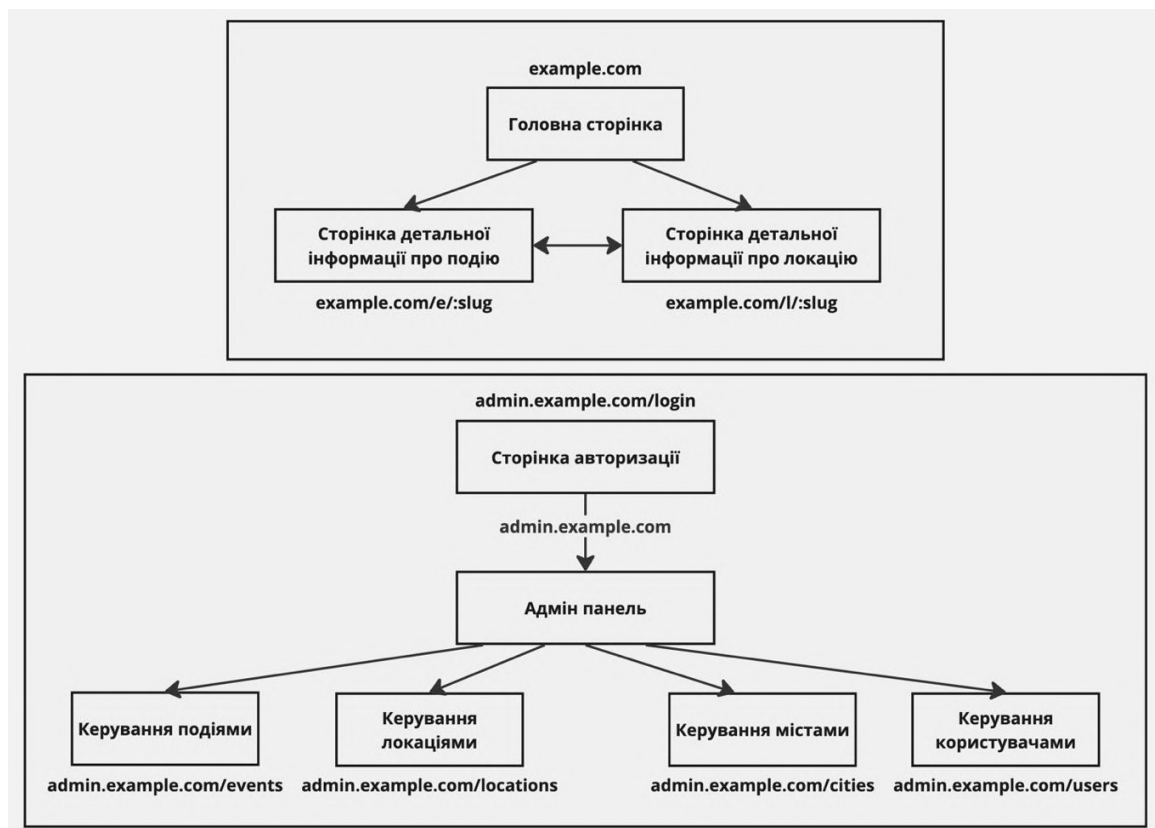


Рисунок 3.2 – Структура клієнтської частини

3.3.1 Діаграма прецедентів

Для візуалізації взаємодії користувачів із системою було розроблено діаграму прецедентів. На ній зображені основні дії, які можуть виконувати

користувачі, організатори та адміністратори. Діаграма прецедентів відображає сценарії використання системи, включаючи авторизацію, перегляд подій та локацій, і керування контентом (рис. 3.3).

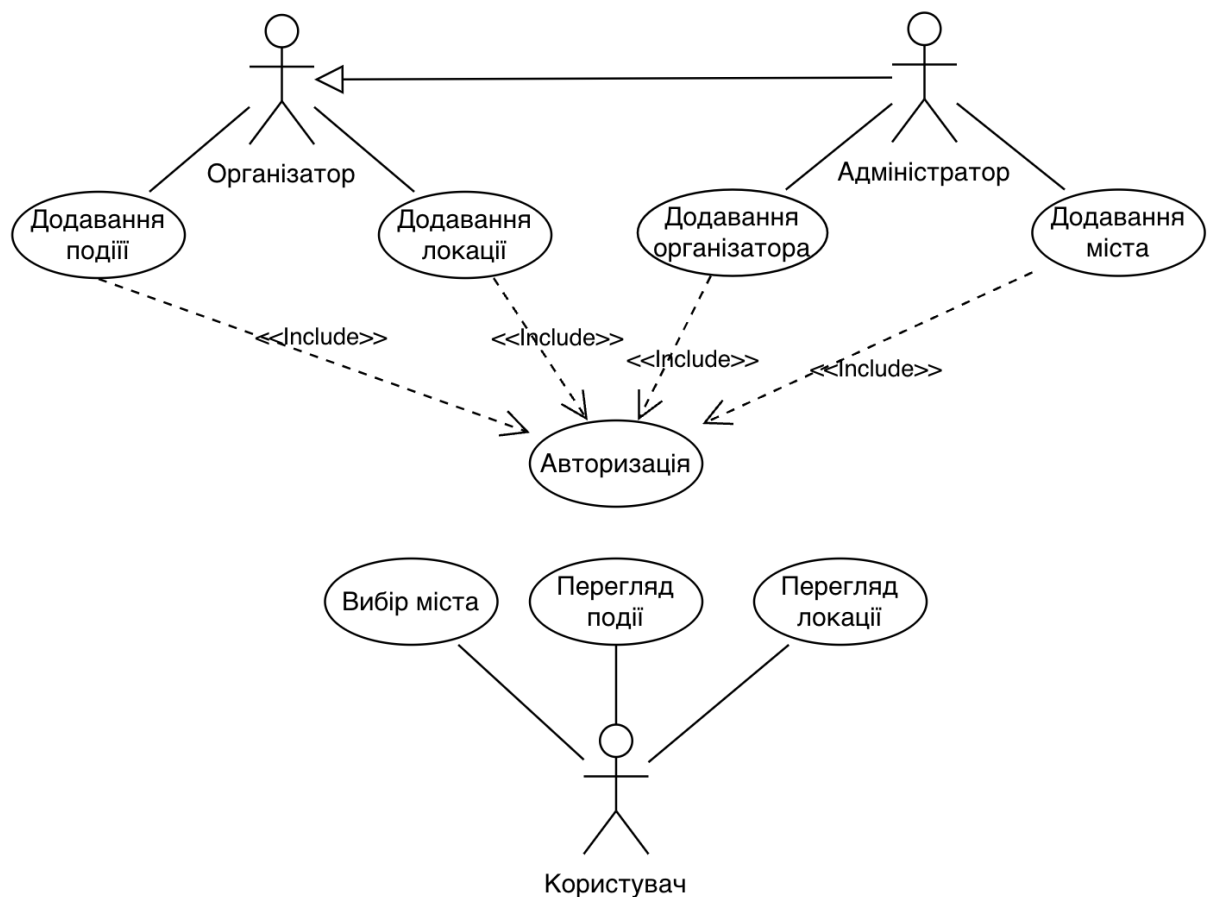


Рисунок 3.3 – Діаграма прецедентів

Авторизація включає вхід користувачів до системи з використанням електронної пошти та пароля. Перегляд подій та локацій дозволяє користувачам переглядати інформацію про події та локації. Керування контентом надає адміністраторам та організаторам можливість додавати, редагувати та видаляти події, локації та міста, а також додавати нових організаторів та керувати їх правами доступу.

Ця діаграма допомагає зрозуміти, які функції надає система різним типам користувачів та як ці функції пов'язані між собою. Вона є корисним

інструментом для аналізу та проєктування клієнтської частини застосунку, забезпечуючи її зручність та ефективність у використанні.

3.4 Структура серверної частини

Серверна частина вебзастосунку побудована на фреймворку NestJS, який забезпечує обробку запитів, бізнес-логіку та взаємодію з базою даних. Основні компоненти серверної частини включають:

а) контролери (controllers):

- 1) обробляють HTTP-запити від клієнтського застосунку;
- 2) визначають маршрути для різних функцій і викликають відповідні сервіси для обробки запитів;
- 3) повертають відповіді у форматі JSON;

б) сервіси (services):

- 1) реалізують бізнес-логіку застосунку;
- 2) виконують основні операції з даними, такі як обробка, валідація та маніпулювання даними;
- 3) взаємодіють з моделями для доступу до бази даних;

в) моделі (models):

- 1) визначають структуру даних, що зберігаються у базі даних;
- 2) використовуються для взаємодії з базою даних через ORM;
- 3) забезпечують маніпулювання даними, такими як створення, читання, оновлення та видалення записів.

На рисунку 3.4 показано структуру серверної частини, яка включає контролери, сервіси та моделі, а також їх взаємодію з клієнтською частиною та базою даних.

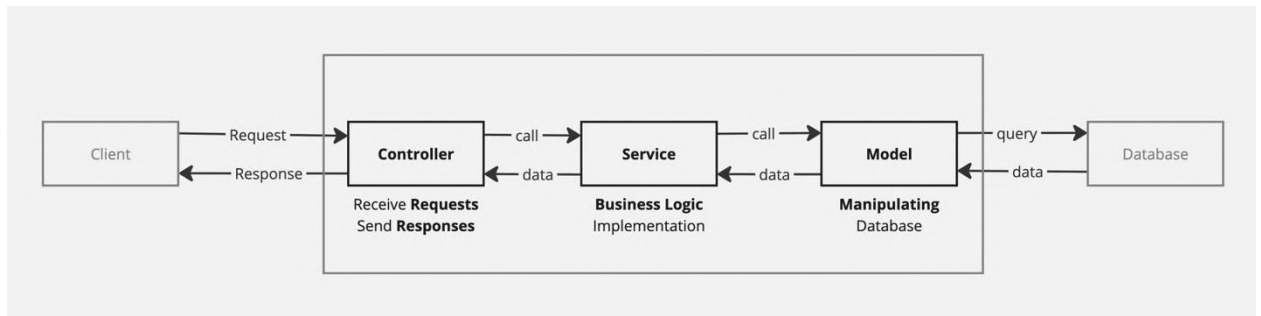


Рисунок 3.4 – Структура серверної частини

Для кращого розуміння структури серверної частини застосунку було розроблено діаграму класів. Діаграма класів відображає основні класи системи, їх атрибути, методи та взаємозв'язки між ними. Вона допомагає зрозуміти, як структуруються дані та яким чином вони взаємодіють у системі (рис. 3.5).

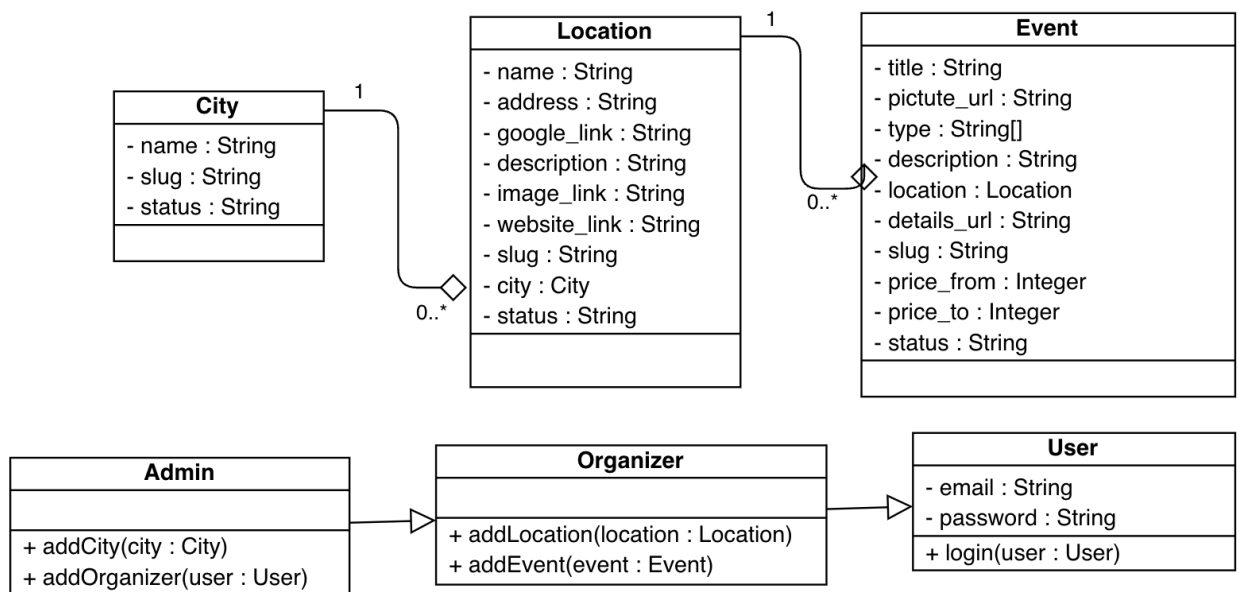


Рисунок 3.5 – Діаграма класів

Діаграма класів показує, як ці класи взаємодіють між собою. Наприклад, клас Event має посилання на Location, який, у свою чергу, має посилання на City. Клас User може бути пов'язаний з класами Admin та Organizer через ролі користувачів.

3.5 Структура бази даних

База даних вебзастосунку реалізована за допомогою MongoDB, що забезпечує високу продуктивність та гнучкість у зберіганні даних. Основні колекції бази даних включають:

- міста (city) - зберігає інформацію про міста, включаючи назву, slug (коротке ім'я для URL) та статус;
- локації (location) - зберігає інформацію про локації, включаючи назву, адресу, посилання на Google Maps, опис, зображення та місто, до якого належить локація;
- події (event) - Зберігає інформацію про події, включаючи назву, опис, зображення, тип, локацію, ціну та статус;
- користувачі (user) - зберігає інформацію про користувачів, включаючи електронну пошту та пароль.

На рисунку 3.6 показано структуру бази даних, яка включає колекції для зберігання інформації про міста, локації, події та користувачів, а також їх взаємозв'язки.

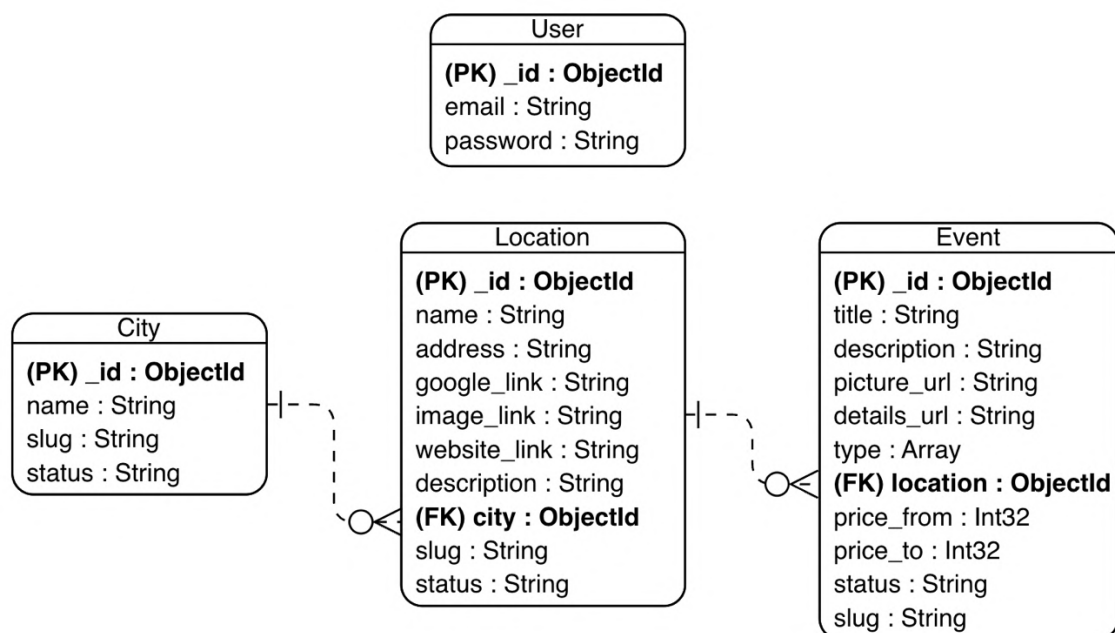


Рисунок 3.6 – Структура бази даних

3.6 Функціонування програми

Функціонування вебсайту для поширення культурних подій міста орієнтоване на надання користувачам легкої та зручної можливості ознайомлення з актуальними подіями та популярними локаціями. Вебсайт складається з кількох основних сторінок, які забезпечують основні функції системи.

3.6.1 Головна сторінка

Головна сторінка є основним вхідним пунктом для користувачів вебсайту. Вона забезпечує швидкий доступ до найновіших подій та найпопулярніших локацій міста. На головній сторінці користувачі можуть переглядати коротку інформацію про події та локації, а також переходити до детальних описів. Основні функції головної сторінки включають відображення останніх подій та популярних локацій.

На головній сторінці вебсайту автоматично відображаються найближчі культурні події за часом. Ці події сортуються за датою та часом проведення, забезпечуючи користувачам швидкий доступ до актуальної інформації. Завантаження даних про нові події здійснюється з бази даних, після чого відображається список останніх подій у зручному форматі, надаючи користувачам можливість переглянути короткий опис та перейти до детальної інформації (рис. 3.7).



Події у Запоріжжі

СМЕТАНА BAND 2 червня - 2 липня Сметана band Незламний Хаб від 300 грн.	Kids Neon Party 2 червня - 2 липня Kids Neon Party Незламний Хаб від 100 грн.	ВЕЛИКОДНІЙ ВІНОК 2 червня - 2 липня Великодній вінок Палац культури "Орбіта" від 250 грн.	ЗА РУКУ З ЩАСТЯМ 2 червня - 2 липня За руку з щастям Палац культури "Орбіта" від 250 грн.
ЗАПОРОЖЕЦЬ ЗА ДУНАЄМ 2 червня - 2 липня ЗАПОРОЖЕЦЬ ЗА	Великодня арт-мастерня 2 червня - 2 липня Великодня арт-	ДЕ МИ БУЛИ ЦІ 10 РОКІВ? 2 червня - 2 липня Де ми були ці 10 років?	ЛІСОВА КАЗКА 2 червня - 2 липня Лісова казка

Рисунок 3.7 – Найближчі події на головній сторінці

Окрім останніх подій, на головній сторінці також відображаються найпопулярніші локації міста. Процес включає завантаження даних про популярні локації з бази даних, відображення списку локацій з коротким описом та зображеннями, а також надання користувачам можливості переглянути деталі про локації (рис. 3.8).

Локації

Запорізька обласна філармонія

📍 проспект Соборний, 183
🌐 Веб-сайт

Запорізька обласна філармонія – це серце музичного життя міста, де класика зустрічається з сучасністю. Наш заклад пропонує насичену програму концертів, включаючи симфонічні та камерні оркестри, сольні виступи, хорові концерти та музичні фестивалі. Вишуканий концертний зал з прекрасною акустикою створює неповторну атмосферу, яка зачаровує кожного слухача. Ми запрошуємо вас зануритися у світ музики, де звучать найкращі твори відомих композиторів у виконанні талановитих музикантів. Приходьте до Запорізької обласної філармонії, щоб насолодитися якісною музикою і отримати незабутні емоції!

Перейти до локації →



Рисунок 3.8 – Найпопулярніші локації міста

3.6.2 Сторінка детальної інформації про подію

Сторінка детальної інформації про подію надає користувачам можливість отримати повний опис події, включаючи всі необхідні деталі для відвідування. Користувач може перейти зі списку подій на головній сторінці до сторінки з детальною інформацією про конкретну подію. Ця сторінка містить повний опис події, дату та час проведення, місце, а також будь-які додаткові деталі. Процес включає завантаження повної інформації про подію з бази даних, відображення опису події, включаючи зображення, дату, час та місце, а також можливість перейти до мапи з розташуванням події (рис. 3.9).

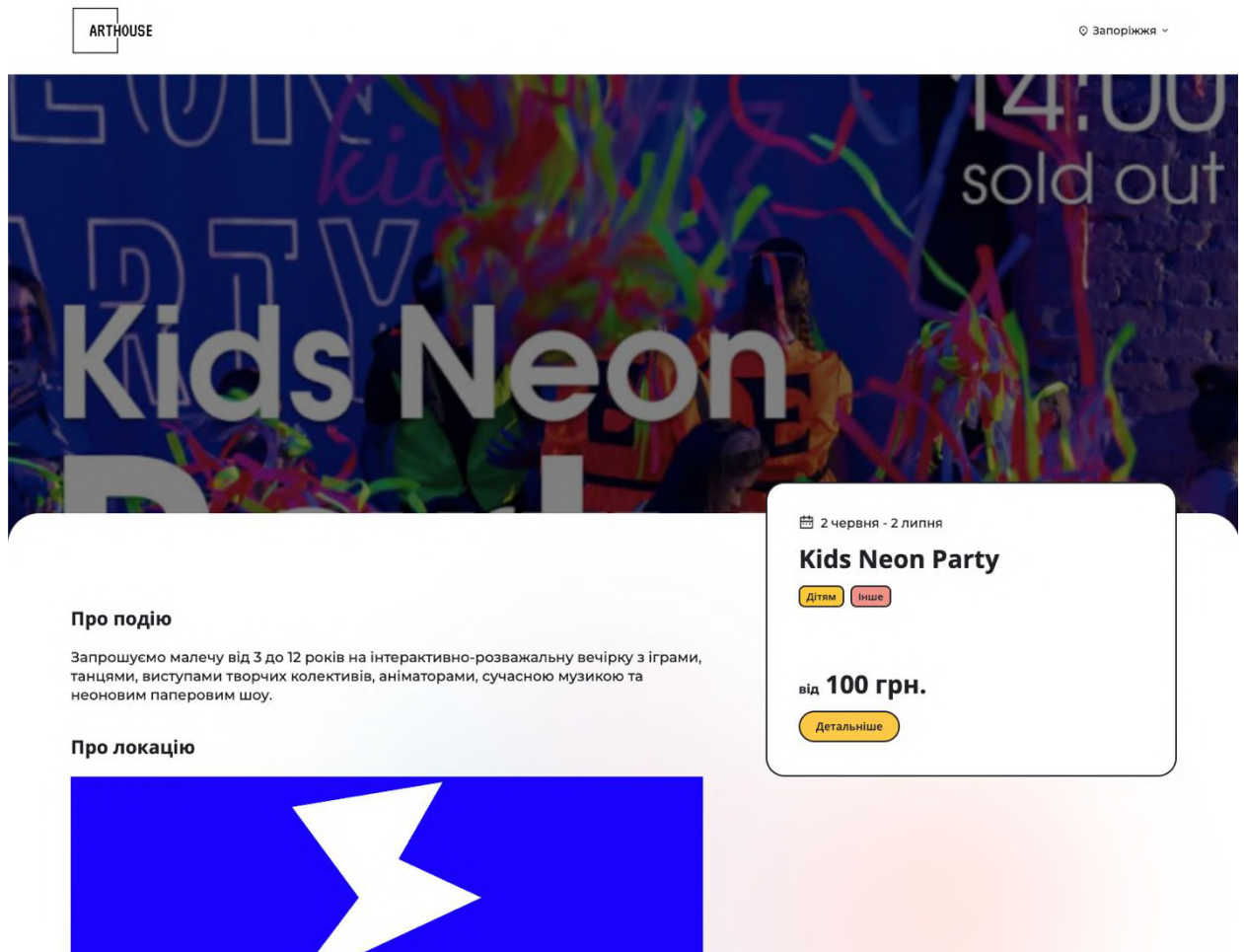


Рисунок 3.9 – Сторінка детальної інформації про подію

3.6.3 Сторінка детальної інформації про локацію

Сторінка детальної інформації про локацію забезпечує користувачів вичерпною інформацією про цікаві місця в місті. Користувач може перейти зі списку локацій на головній сторінці до сторінки з детальною інформацією про конкретну локацію. Ця сторінка містить детальний опис локації, фотографії та посилання на мапу з розташуванням. Процес включає завантаження повної інформації про локацію з бази даних, відображення опису локації, включаючи всі деталі, а також можливість перегляду афіши найближчих подій (рис. 3.10).

Запорізька обласна філармонія

📍 проспект Соборний, 183

🌐 Веб-сайт

Запорізька обласна філармонія – це серце музичного життя міста, де класика зустрічається з сучасністю. Наш заклад пропонує насичену програму концертів, включаючи симфонічні та камерні оркестри, сольні виступи, хорові концерти та музичні фестивалі. Вишуканий концертний зал з прекрасною акустикою створює неповторну атмосферу, яка зачаровує кожного слухача. Ми запрошуємо вас зануритися у світ музики, де звучать найкращі твори відомих композиторів у виконанні талановитих музикантів. Приходьте до Запорізької обласної філармонії, щоб насолодитися якісною музикою і отримати незабутні емоції!



Афіша



📅 2 червня - 2 липня
ПЕС ПАТРОН І К
📍 Запорізька обласна філармонія
від 150 грн.



📅 2 червня - 2 липня
ЗАПОРОЖЕЦЬ ЗА ДУНАЄМ
ОПЕРА
АКАДЕМІЧНИЙ
📍 Запорізька обласна філармонія
від 150 грн.

Рисунок 3.10 – Сторінка детальної інформації про локацію

3.6.4 Сторінка авторизації

Сторінка авторизації дозволяє адміністраторам та організаторам здійснювати вхід до адміністративної панелі. Вона включає два поля вводу для електронної пошти та пароля, а також кнопку для підтвердження введених даних. Після введення даних та натискання кнопки "Login", інформація передається на сервер для перевірки (рис. 3.11).

Login

Email

Password

Log in

Рисунок 3.11 – Сторінка авторизації

3.6.5 Панель адміністратора

Панель адміністратора надає адміністраторам та організаторам зручний інтерфейс для керування подіями, локаціями, користувачами та містами. Вона дозволяє додавати, редагувати та видаляти записи, а також переглядати статус та деталі кожного елемента (рис. 3.12).

Entities

Events

Locations

Users

Cities

Events

Add Event +

NAME	SLUG	STATUS	ACTIONS
Кицин дїм	kitsin-dim	published	
Муха-Цокотуха	muha-tsokotuha	published	
Лїсова казка	lisova-kazka	published	
В пошуках Дракона	v-poiskakh-drakona	published	
Великодня арт-майстерня	velikodnya-art-masternya	published	

Load More

Рисунок 3.12 – Панель адміністратора

Панель адміністратора забезпечує централізоване керування всіма аспектами системи, дозволяючи адміністраторам та організаторам ефективно управляти контентом і користувачами.

3.6.6 Обробка даних на сервері

Серверна частина вебсайту відіграє ключову роль у забезпеченні функціонування всіх компонентів системи. Вона відповідає за обробку запитів користувачів, оновлення бази даних та оптимізацію запитів. Сервер регулярно оновлює базу даних, додаючи нові події та актуалізуючи інформацію про локації, що забезпечує постійний доступ до свіжої та точної інформації. Процес включає прийом та обробку нових даних від організаторів або інших джерел, оновлення бази даних з новими подіями та інформацією про локації, а також відправку оновлених даних на клієнтську частину для відображення користувачам.

Для забезпечення швидкого завантаження сторінок та високої продуктивності системи сервер оптимізує запити до бази даних, використовуючи індекси, кешування та інші методи оптимізації. Процес включає використання індексів для прискорення пошуку даних, кешування популярних запитів для зменшення навантаження на сервер, а також оптимізацію структури бази даних для підвищення продуктивності.

Таким чином, функціонування вебсайту забезпечує інтегровану та зручну взаємодію користувачів з системою, надаючи їм актуальну інформацію про культурні події та локації міста. Описані процеси враховують потреби користувачів та забезпечують високу якість та ефективність роботи вебсайту.

3.7 Висновки до розділу

Цей розділ розкриває деталі розробки архітектури вебсайту для поширення культурних подій міста, зокрема, визначення загальних вимог,

логічної структури, вимог до окремих елементів та опису функціонування програми. Важливість цього розділу полягає у забезпеченні структурованого підходу до створення вебсайту, що дозволяє ефективно впроваджувати всі необхідні функції та забезпечувати зручність користування.

Загальні вимоги до архітектури програми, окреслені у підрозділі 3.1, охоплюють основні принципи, які повинні бути враховані при розробці: зручність користування, ефективність роботи, масштабованість та безпека. Ці вимоги є основоположними для побудови надійної та ефективної системи.

У підрозділах 3.2-3.5 було детально описано логічну структуру програми, яка включає основні компоненти вебсайту, такі як клієнтська та серверна частини та база даних, їхні функції та взаємодію між собою. Це забезпечує чітке розуміння того, як різні частини системи працюють разом для досягнення загальних цілей.

Опис функціонування програми у підрозділі 3.6 надає детальний огляд того, як користувачі взаємодіють з вебсайтом і як різні компоненти системи забезпечують цю взаємодію. Включення процесів відображення останніх подій, популярних локацій, завантаження детальної інформації про події та локації, а також оптимізація серверних запитів демонструють комплексний підхід до забезпечення функціональності вебсайту.

Таким чином, цей розділ надає всебічне розуміння архітектури та функціонування вебсайту для поширення культурних подій міста, що є важливим етапом у процесі його розробки. Він слугує основою для реалізації проєкту, забезпечуючи високу якість та відповідність реальним потребам користувачів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ВЕБСАЙТУ

4.1 Опис застосування вебсайту

Розроблений вебсайт для поширення культурних подій міста має на меті забезпечити користувачів зручним та ефективним інструментом для отримання інформації про останні культурні заходи та популярні локації. Програма орієнтована на широке коло користувачів, включаючи мешканців міста, туристів, організаторів подій та культурні установи. Основними завданнями вебсайту є інформування про культурні заходи, надання детальної інформації про події та локації, а також підвищення обізнаності про культурне життя міста.

Вебсайт охоплює декілька ключових областей застосування:

- інформування про культурні заходи: програма дозволяє користувачам отримувати актуальну інформацію про майбутні та поточні події, включаючи концерти, виставки, театральні вистави та інші заходи. Це допомагає мешканцям міста та гостям бути в курсі найцікавіших подій;
- детальна інформація про події та локації: кожна подія має свою сторінку з детальним описом, включаючи дату, час, місце проведення, опис та іншу корисну інформацію. Локації також мають свої сторінки з описом, фотографіями та посиланнями на сайт та мапу розташування локації, що допомагає користувачам дізнатися більше про місця проведення заходів;
- підвищення обізнаності про культурне життя: вебсайт сприяє популяризації культури та мистецтва, надаючи платформу для організаторів подій та культурних установ для анонсування своїх заходів.

4.1.1 Призначення програми

Програма призначена для всіх, хто цікавиться культурними подіями міста. Вона допомагає користувачам швидко знайти потрібну інформацію про

заходи, запланувати відвідування подій та дізнатися більше про культурні локації. Організатори подій можуть використовувати вебсайт для анонсування своїх заходів та залучення більшої аудиторії.

4.1.2 Обмеження для застосування

Основними обмеженнями для застосування програми є необхідність постійного доступу до Інтернету для отримання та оновлення даних про події. Також вебсайт залежить від актуальності та точності інформації, що надається організаторами заходів та культурними установами.

4.1.3 Мінімальна конфігурація технічних засобів

Для використання вебсайту потрібен пристрій з браузером та стабільним з'єднанням з Інтернетом. Вебсайт оптимізований для роботи на різних пристроях, включаючи комп'ютери, планшети та смартфони, що забезпечує доступність інформації для широкого кола користувачів.

4.2 Методика та результати тестування

Тестування програми є критичним етапом розробки, що дозволяє забезпечити її стабільну роботу, відповідність вимогам користувачів та високу якість. У цьому розділі детально описується методика тестування розробленого вебсайту для поширення культурних подій міста, а також результати проведених тестувань.

4.2.1 Методика тестування

Методика тестування включала кілька основних етапів. Спершу проводилося функціональне тестування, метою якого було перевірити

коректність виконання основних функцій вебсайту. Цей процес охоплював перевірку відображення останніх подій, навігації по сайту, завантаження детальної інформації про події та локації. Виконання сценаріїв користувача дозволило перевірити всі функціональні можливості програми, включаючи можливість переходу між сторінками та відображення всіх необхідних даних.

Наступним етапом було тестування, яке мало на меті оцінити зручність використання інтерфейсу вебсайту. Це тестування проводилося з участю реальних користувачів, які виконували типові завдання на сайті. Спостереження за діями користувачів та збір відгуків дозволили визначити, наскільки інтуїтивно зрозумілим є інтерфейс, та виявити можливі проблеми у навігації чи доступності інформації.

Під час навантажувального тестування перевірялася стабільність роботи вебсайту при високих навантаженнях. Моделювання одночасного доступу великої кількості користувачів до вебсайту дозволило оцінити, як система справляється з великим обсягом запитів.

4.2.2 Результати тестування

Результати тестування показали, що всі основні функції вебсайту працюють коректно. Сторінка головних подій та популярних локацій завантажуються швидко, забезпечуючи користувачам доступ до актуальної інформації. Сторінки з детальною інформацією про події та локації також працюють без помилок, забезпечуючи повну інформацію для відвідувачів.

Показники продуктивності вебсайту, такі як FCP, LCP, TBT, та інші ключові метрики, свідчать про високу швидкість завантаження та оптимізовану роботу сайту. На рисунку 4.1 представлено результати тестування продуктивності вебсайту, які показують, що сайт відповідає високим стандартам швидкості та ефективності.

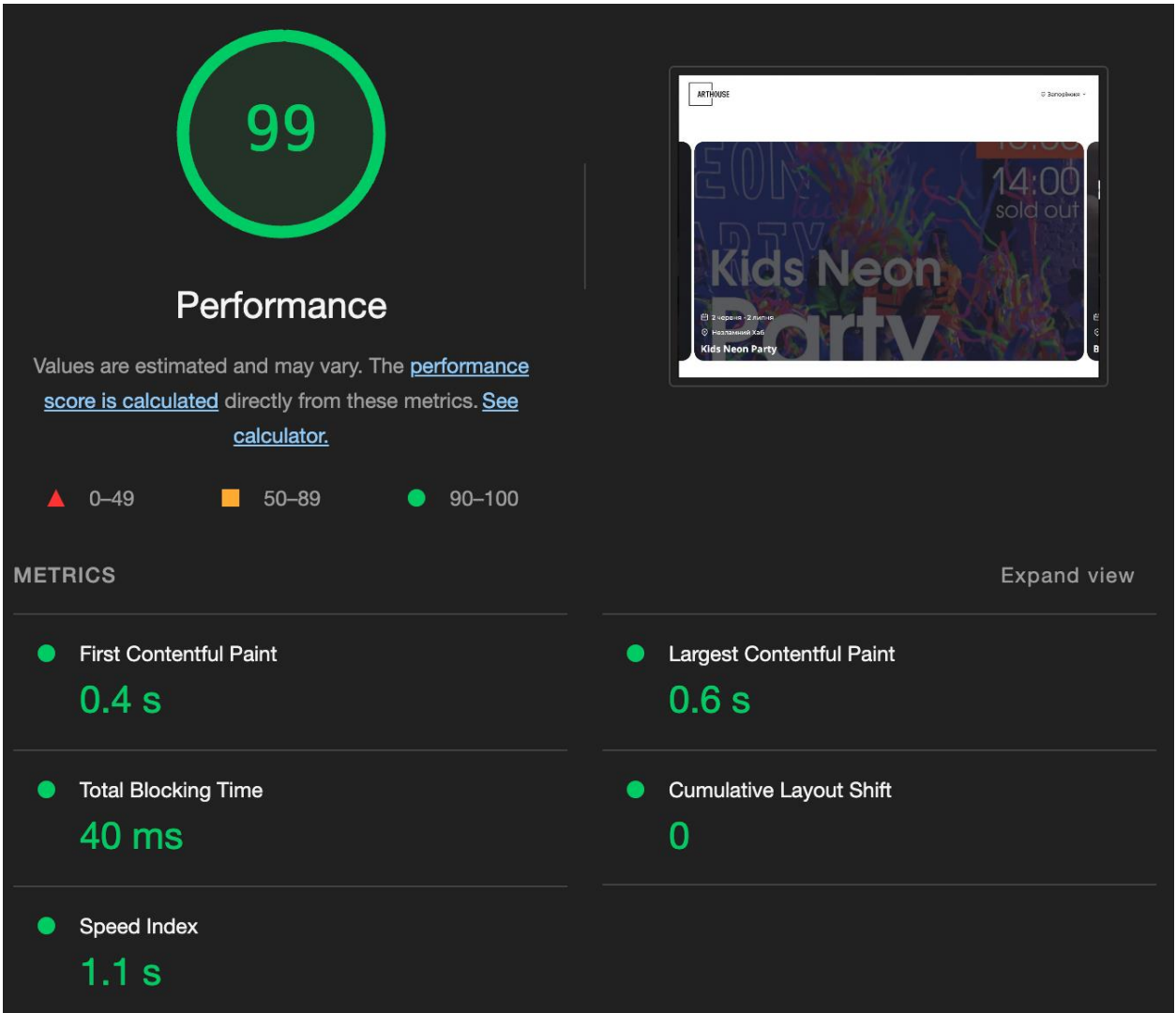


Рисунок 4.1 – Результати тестування продуктивності

Тестування зручності використання продемонструвало, що інтерфейс вебсайту є інтуїтивно зрозумілим та зручним для користувачів різних вікових груп. Користувачі відзначили легкість навігації, зручне розташування елементів управління та швидкий доступ до потрібної інформації.

Навантажувальне тестування підтвердило, що вебсайт може стабільно працювати при високих навантаженнях. Система здатна обробляти велику кількість одночасних запитів без значних затримок, забезпечуючи швидке завантаження сторінок та коректну роботу всіх функцій.

На рисунку 4.2 представлено результати навантажувального тестування, яке було проведено за допомогою Postman [21]. Графік демонструє, що навіть при великій кількості запитів система підтримує

низький час відповіді, що свідчить про високу ефективність та надійність вебсайту.

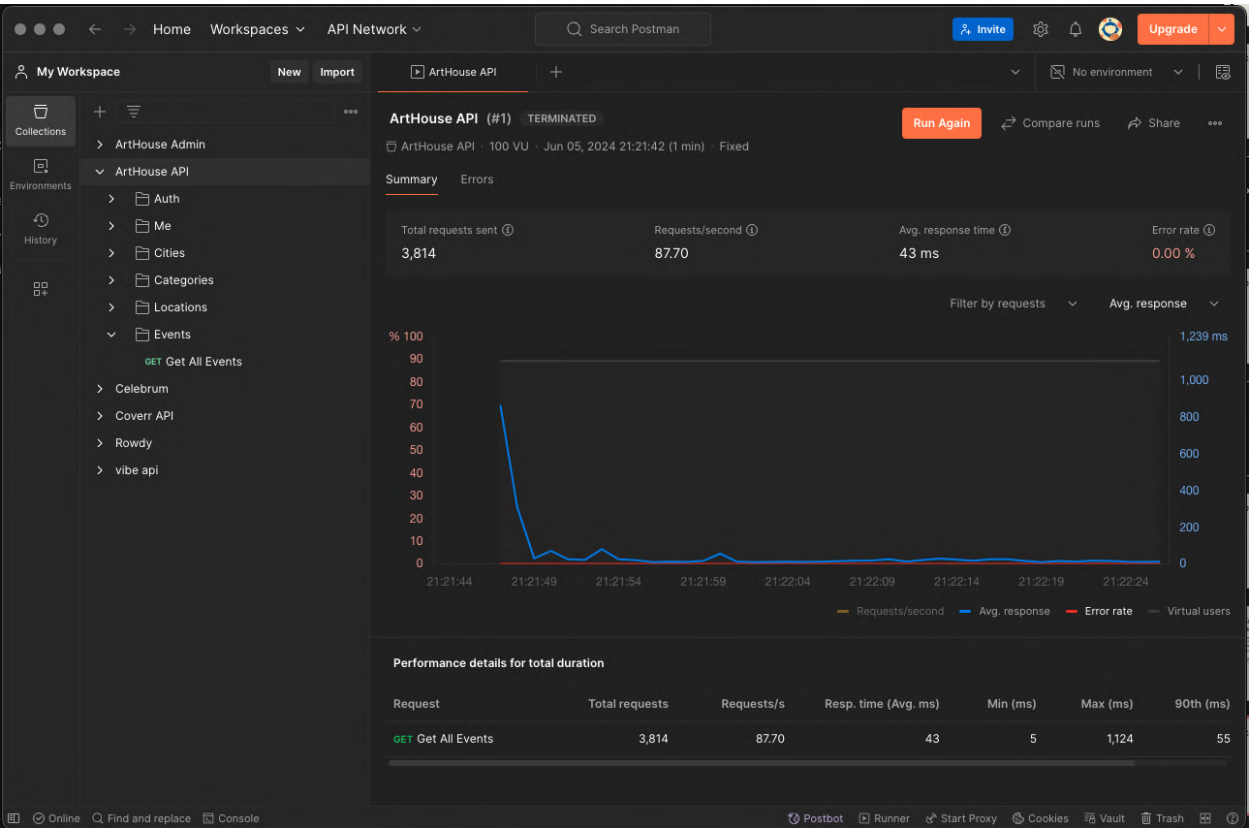


Рисунок 4.2 – Результати навантажувального тестування

Результати проведеного тестування підтвердили, що розроблений вебсайт для поширення культурних подій міста відповідає всім заявленим вимогам та стандартам якості. Програма забезпечує стабільну роботу, високу продуктивність та зручність використання, що робить її ефективним інструментом для інформування мешканців міста та гостей про культурні події.

4.3 Експериментальні дослідження вебсайту

Експериментальні дослідження програми є важливою частиною розробки, оскільки вони дозволяють оцінити ефективність та надійність вебсайту в реальних умовах. Метою цих досліджень є перевірка

функціональності, продуктивності та зручності використання програми при вирішенні практичних завдань інформування про культурні події міста.

4.3.1 Загальна схема експериментів

Експериментальні дослідження проводилися у кілька етапів. На першому етапі були визначені основні сценарії використання програми, включаючи перегляд головної сторінки, відображення останніх подій, перехід до сторінок з детальною інформацією про події та локації. На другому етапі були підготовлені тестові дані, що включали інформацію про різноманітні культурні події та популярні локації міста. Третій етап включав проведення експериментів з використанням реальних користувачів, що дозволило отримати зворотній зв'язок та оцінити зручність і ефективність використання вебсайту.

4.3.2 Використане устаткування і вихідні дані

Для проведення експериментів було використано наступне устаткування: сервер з розгорнутим вебсайтом, клієнтські пристрої з різними браузерами (Google Chrome, Mozilla Firefox, Safari) та стабільне з'єднання з Інтернетом. Вихідні дані включали інформацію про різні культурні заходи, такі як концерти, театральні вистави, виставки та інші події, а також детальну інформацію про популярні локації міста. Ці дані були підготовлені та завантажені у базу даних перед початком експериментів.

4.3.3 Параметри запуску методів і програмних засобів

Під час проведення експериментів використовувалися наступні параметри запуску методів і програмних засобів:

- перевірка доступності сайту: запуск вебсайту на різних пристроях та браузерях, перевірка коректного відображення сторінок та доступності функцій;
- швидкість завантаження сторінок: вимірювання часу завантаження головної сторінки, сторінок з подіями та локаціями, а також швидкості відображення детальної інформації;
- правильність відображення контенту: перевірка коректного відображення текстової та графічної інформації, включаючи зображення та карти;
- зручність навігації: оцінка інтуїтивності навігаційних елементів та зручності використання інтерфейсу користувачами.

4.3.4 Результати експериментальних досліджень

Експериментальні дослідження показали, що вебсайт ефективно виконує свої функції та відповідає заявленим вимогам. Основні результати можна підсумувати наступним чином:

- доступність сайту: вебсайт коректно завантажується на всіх тестованих пристроях та браузерях, забезпечуючи стабільну роботу та доступність всіх функцій;
- швидкість завантаження сторінок: час завантаження головної сторінки та сторінок з детальною інформацією про події та локації знаходиться в межах прийнятних значень, що забезпечує швидкий доступ користувачів до необхідної інформації;
- правильність відображення контенту: усі текстові та графічні елементи відображаються коректно, зображення завантажуються швидко, а посилання на карти карти точно відображають місця проведення подій та розташування локацій;

– зручність навігації: користувачі відзначили високу зручність навігаційних елементів, легкість у пошуку інформації та інтуїтивність інтерфейсу.

4.3.5 Аналіз отриманих результатів

Отримані результати експериментальних досліджень свідчать про те, що вебсайт для поширення культурних подій міста ефективно виконує свої функції та відповідає потребам користувачів. Доступність, швидкість завантаження сторінок, правильність відображення контенту та зручність навігації були підтверджені під час проведення тестувань.

4.4 Висновки до розділу

У цьому розділі було проведено всебічний аналіз експлуатації, тестування та експериментального дослідження розробленого вебсайту для поширення культурних подій міста. Основна мета полягала в тому, щоб оцінити ефективність та надійність вебсайту, забезпечити його стабільну роботу та відповідність вимогам користувачів.

Першим етапом було визначення методів та областей застосування програми. Було встановлено, що вебсайт успішно виконує свою основну функцію інформування мешканців міста та туристів про культурні заходи, забезпечуючи зручний доступ до детальної інформації про події та локації. Програма також служить платформою для організаторів подій, які можуть анонсувати свої заходи та залучати широку аудиторію.

Методика та результати тестування детально описали процес перевірки основних функцій вебсайту, зокрема функціональне тестування, тестування зручності використання та навантажувальне тестування. Отримані результати підтвердили, що вебсайт забезпечує швидке завантаження сторінок, коректне відображення контенту та високу зручність навігації.

Експериментальні дослідження програми підтвердили її ефективність у реальних умовах експлуатації. Проведені тести з використанням реальних користувачів показали, що вебсайт відповідає заявленим вимогам та стандартам якості. Вебсайт стабільно працює при високих навантаженнях, забезпечує захист даних користувачів та є зручним у використанні.

Узагальнюючи результати, можна стверджувати, що розроблений вебсайт для поширення культурних подій міста є ефективним інструментом для інформування жителів і відвідувачів міста про культурні події. Програма забезпечує надійну та безпечну роботу, високу продуктивність та зручність використання, що робить її цінним ресурсом для розвитку культурного життя міста.

На основі проведених тестувань та експериментальних досліджень можна рекомендувати подальший розвиток та вдосконалення вебсайту. Розробка нових функцій, інтеграція сучасних технологій та регулярне оновлення інформації дозволять зберегти високу якість та актуальність програми, забезпечуючи її відповідність потребам користувачів та розвиток культурного середовища міста.

ВИСНОВКИ

У даній кваліфікаційній роботі бакалавра була розроблена програма для поширення інформації про культурні події міста, яка включає створення вебсайту з інтуїтивно зрозумілим інтерфейсом, ефективними функціями та забезпеченням надійної роботи. Проведена робота включала аналіз потреб користувачів, визначення вимог до системи, розробку архітектури програми, реалізацію та тестування вебсайту, а також експериментальне дослідження програми у реальних умовах експлуатації.

Під час аналізу предметної області було проведено всебічний огляд існуючих рішень для поширення інформації про культурні події. Було визначено основні вимоги до вебсайту, такі як зручність використання, швидкість завантаження та надійність. На основі цього аналізу була розроблена архітектура програми, що включає логічну структуру вебсайту з головною сторінкою, сторінками з детальною інформацією про події та локації, а також серверну частину для обробки даних. Детальні вимоги до окремих елементів структури програми забезпечують коректну роботу всіх компонентів системи.

В процесі реалізації програми було створено вебсайт, що забезпечує швидкий та зручний доступ до інформації про культурні події міста. Реалізовано інтерфейс, який дозволяє користувачам з легкістю знаходити потрібну інформацію та отримувати детальні відомості про події та локації. Після реалізації програма пройшла всебічне тестування. Проведено функціональне, зручності використання та навантажувальне тестування вебсайту. Виявлені недоліки були усунені, що забезпечило високу якість та надійність програми.

Експериментальні дослідження підтвердили ефективність вебсайту у реальних умовах експлуатації. Вебсайт забезпечує стабільну роботу при високих навантаженнях, швидке завантаження сторінок та зручність використання.

Розроблений вебсайт для поширення культурних подій міста є ефективним інструментом для інформування жителів та відвідувачів міста про культурні події. Програма забезпечує надійну та безпечну роботу, високу продуктивність та зручність використання, що робить її цінним ресурсом для розвитку культурного життя міста. На основі проведених тестувань та експериментальних досліджень можна рекомендувати подальший розвиток та вдосконалення вебсайту. Розробка нових функцій, інтеграція сучасних технологій та регулярне оновлення інформації дозволять зберегти високу якість та актуальність програми, забезпечуючи її відповідність потребам користувачів та розвиток культурного середовища міста.

Таким чином, виконана кваліфікаційна робота бакалавра підтверджує, що розроблений вебсайт відповідає заявленим вимогам та досягає поставлених цілей, забезпечуючи ефективне інформування про культурні події та сприяючи розвитку культурного життя міста.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Concert.ua [Електрон. ресурс]. – Режим доступу: <https://concert.ua>
2. Karabas [Електрон. ресурс]. – Режим доступу: <https://zaporozhje.karabas.com>
3. Musin [Електрон. ресурс]. – Режим доступу: <https://musin.zp.ua/>
4. Комунальний заклад «Запорізький академічний обласний театр юного глядача» Запорізької обласної ради [Електрон. ресурс]. – Режим доступу: <https://teatrmolodi.zp.ua/>
5. Палац культури "Орбіта" [Електрон. ресурс]. – Режим доступу: https://www.facebook.com/pkorbita/?locale=uk_UA
6. Запорізька обласна філармонія [Електрон. ресурс]. – Режим доступу: <https://www.filarmonic.zp.ua/>
7. Незламний Хаб Запоріжжя [Електрон. ресурс]. – Режим доступу: https://www.instagram.com/nezlamnyi_hub/
8. THE GARAGE PARTY [Електрон. ресурс]. – Режим доступу: <https://www.instagram.com/the.garage.party/>
9. Java™ Programming Language [Electronic resource]. – Access mode: <https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html>
10. Python [Electronic resource]. – Access mode: <https://www.python.org/>
11. JavaScript [Electronic resource]. – Access mode: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
12. TypeScript [Electronic resource]. – Access mode: <https://www.typescriptlang.org/>
13. Object-oriented programming [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Object-oriented_programming
14. React [Electronic resource]. – Access mode: <https://react.dev/>
15. Angular [Electronic resource]. – Access mode: <https://angular.io/>
16. Vue.js [Electronic resource]. – Access mode: <https://vuejs.org/>
17. gNext.js [Electronic resource]. – Access mode: <https://nextjs.org/>

18. Express [Electronic resource]. – Access mode: <https://expressjs.com/>
19. NestJS [Electronic resource]. – Access mode: <https://nestjs.com/>
20. Koa [Electronic resource]. – Access mode: <https://koajs.com/>
21. Postman [Electronic resource]. – Access mode: <https://www.postman.com/>

ДОДАТОК А
Текст програми

Серверна частина

```
// main.ts
```

```
async function bootstrap() {
  const app = await NestFactory.create(AppModule, {
    cors: true,
  });

  const port = process.env.PORT || 3000;
  await app.listen(port);

  Logger.log(`□ Application is running on: http://localhost:${port}`);
}
```

```
// app.module.ts
```

```
@Module({
  imports: [
    MongooseModule.forRoot(dbUrl),
    AuthModule,
    CitiesModule,
    EventsModule,
    LocationsModule,
    MeModule,
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {
  configure(consumer: MiddlewareConsumer) {
    consumer
      .apply(CamelCaseMiddleware)
      .forRoutes({ path: '*', method: RequestMethod.ALL });
  }
}
```

```
// app.service.ts
```

```
@Injectable()
export class AppService {
  getData(): { message: string } {
    return { message: 'Hello API' };
  }
}
```

```
// app.controller.ts
```

```
@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getData() {
    return this.appService.getData();
  }
}
```

```
// camelcase.middleware.ts
```

```
@Injectable()
export class CamelCaseMiddleware implements NestMiddleware {
  use(req: Request, res: Response, next: NextFunction): void {
    req.query = this.transformQueryToCamelCase(req.query);
    next();
  }

  private transformQueryToCamelCase(query: unknown): any {
    const camelCasedQuery = {};
    Object.keys(query).forEach((key) => {
      camelCasedQuery[camelCase(key)] = query[key];
    });

    console.log(camelCasedQuery)
    return camelCasedQuery;
  }
}
```

```
// auth.module.ts
```

```
@Module({
  imports: [
    MongooseModule.forFeature([
      {
        name: User.name,
        schema: UserSchema,
      }
    ]),
    JwtModule.register({
      global: true,
      secret: jwtConstants.secret,
```

```

    signOptions: { expiresIn: '24h' },
  }),
],
controllers: [AuthController],
providers: [
  UsersService,
  AuthService,
  TokenService,
]
})
export class AuthModule {}

```

```
// auth.controller.ts
```

```

@Controller('auth')
export class AuthController {
  constructor(
    private authService: AuthService,
  ) {}

  @Post('signin')
  async signin(@Body() body: UserSigninDto) {
    try {
      const tokenData = await this.authService.signIn(body.userDto);

      return {
        ...tokenData,
      };
    } catch(error) {
      throw new UnauthorizedException(error);
    }
  }
}

```

```
// auth.service.ts
```

```

@Injectable()
export class AuthService {
  constructor(
    private usersService: UsersService,
    private jwtService: JwtService,
  ) {}

  async signIn (data: UserDTO) {

```

```

    try {
      const payload = {
        id: data.id,
        email: data.email,
        role: data.role,
      };

      return {
        access_token: await this.jwtService.signAsync(payload, {
          expiresIn: '30d',
        }),
      };
    } catch (error) {
      console.log(error);
      throw new UnauthorizedException(error);
    }
  }
}

// token.service.ts

export class TokenService {
  public static signToken(payload: any, expiresIn = '24h'): string {
    return sign(payload, SECRET, { expiresIn })
  }

  public static verifyToken(token: string) {
    return verify(token, SECRET)
  }
}

// users.service.ts

@Injectable()
export class UsersService extends ApiBaseService<
  UserDocument,
  CreateUserDto,
  UpdateUserDto
> {
  constructor(@InjectModel(User.name) model: Model<UserDocument>) {
    super(model);
  }
}

```

```
// cities.module.ts
```

```
@Module({
  imports: [
    MongooseModule.forFeature([
      { name: City.name, schema: CitySchema },
    ]),
  ],
  providers: [
    CitiesService,
  ],
  controllers: [CitiesController],
})
export class CitiesModule {}
```

```
// cities.controller.ts
```

```
@Controller('cities')
export class CitiesController {
  constructor(private service: CitiesService) {}
```

```
  @UseGuards(ApiKeyGuard)
  @Get()
  findAll() {
    return this.service.findAll({
      status: CityStatus.Published,
    });
  }
```

```
  @UseGuards(ApiKeyGuard)
  @Get(':slugOrId')
  getOne(@Param('slugOrId') slugOrId: string) {
    return this.service.getOneBySlugOrId(slugOrId);
  }
}
```

```
// cities.service.ts
```

```
@Injectable()
export class CitiesService extends ApiBaseService<
  CityDocument,
  CreateCityDto,
  UpdateCityDto
```

```
> {
  constructor(@InjectModel(City.name) model: Model<CityDocument>) {
    super(model);
  }
}
```

```
// events.module.ts
```

```
@Module({
  imports: [
    MongooseModule.forFeature([
      { name: Event.name, schema: EventSchema },
      { name: Location.name, schema: LocationSchema },
    ]),
  ],
  providers: [
    EventsService,
    LocationsService,
  ],
  controllers: [EventsController],
})
export class EventsModule {}
```

```
// events.controller.ts
```

```
@Controller('events')
export class EventsController {
  constructor(
    private service: EventsService,
    private locationsService: LocationsService
  ) {}

  @Get()
  @UseGuards(ApiKeyGuard)
  async findAll(@Query() query): Promise<{ data: Event[] }> {
    const today = new Date();
    today.setHours(0, 0, 0, 0);
    const nextMonth = new Date();
    nextMonth.setHours(0, 0, 0, 0);
    nextMonth.setMonth(nextMonth.getMonth() + 1);

    const filters: FilterQuery<Event> = {
      'event_occurrences.date_to': { $gte: today },
      status: { $ne: 'draft' },
    };
  }
}
```

```
};
```

```
const locationId: string = query?.location;
if (locationId) {
  filters.location = new mongoose.Types.ObjectId(locationId);
}
```

```
const eventTypes: string = query.event_types;
if (eventTypes) {
  const eventTypesParsed = eventTypes.split('|');
  filters.type = { $in: eventTypesParsed };
}
```

```
const cityId = query?.city;
if (cityId) {
  const locationsRes = await this.locationsService.findAll({
    city: query.city,
  });
  const locationsIds = locationsRes.data.map((l) => l._id);
  filters.location = {
    $in: locationsIds,
  };
}
```

```
return this.service.findAll(filters, null, {}, query);
}
```

```
@Get(':slugOrId')
@UseGuards(ApiKeyGuard)
getOne(@Param('slugOrId') slugOrId: string): Promise<Event | null> {
  return this.service.getOneBySlugOrId(slugOrId);
}
}
```

```
// events.service.ts
```

```
@Injectable()
export class EventsService extends ApiBaseService<
  EventDocument,
  CreateEventDto,
  UpdateEventDto
> {
  constructor(@InjectModel(Event.name) model: Model<EventDocument>)
{
```

```
const sortMap = new Map();
sortMap.set('new', { created_at: -1 });
sortMap.set('fresh', { updated_at: -1 });
sortMap.set('closest', { 'event_occurrences.date_to': 1 });
```

```
super(model, {
  populateFields: [
    'location',
    'location.city',
    'tags',
    'involved_artists',
    'created_by',
  ],
  sort: 'closest',
  sortMap: sortMap,
});
}
```

```
/**
 * Get actual events that are not in draft
 *
 * @param filters
 * @returns
 */
findActual(filters: EventFilterQuery) {
  const mergedFilters = this.mergeFilters(filters);
  return this.findAll(mergedFilters);
}
```

```
private mergeFilters(filters: EventFilterQuery): EventFilterQuery {
  const today = new Date();
  today.setHours(0, 0, 0, 0);
```

```
  return {
    'event_occurrences.date_to': { $gte: today },
    status: { $ne: 'draft' },
    ...filters,
  };
}
}
```

```
// locations.module.ts
```

```
@Module({
```

```

imports: [
  MongooseModule.forFeature([
    { name: Location.name, schema: LocationSchema },
    { name: Event.name, schema: EventSchema },
    { name: City.name, schema: CitySchema },
  ]),
],
providers: [
  LocationsService,
  EventsService,
],
controllers: [LocationsController],
})
export class LocationsModule {}

```

```
// locations.controller.ts
```

```

@Controller('locations')
export class LocationsController {
  constructor(
    private service: LocationsService,
    private eventsService: EventsService,
  ) {}

```

```

  @Get()
  @UseGuards(ApiKeyGuard)
  async findAll(@Query() query: any) {
    const locationsQuery: any = {
      status: { $ne: 'draft' },
    };

```

```

    if (query?.city) {
      locationsQuery.city = query.city;
    }

```

```

    return this.service.findAll(locationsQuery, {}, {}, query);
  }

```

```

  @Get(':slugOrId')
  @UseGuards(ApiKeyGuard)
  async getOne(@Param('slugOrId') slugOrId: string) {
    return this.service.getOneBySlugOrId(slugOrId);
  }

```

```

    @Get(':slugOrId/events')
    @UseGuards(ApiKeyGuard)
    async getEvents(@Param('slugOrId') slugOrId: string): Promise<{ data:
Event[] }> {
        const location = await this.service.getOneBySlugOrId(slugOrId);
        const today = new Date();
        today.setHours(0, 0, 0, 0);

        const filters: FilterQuery<Event> = {
            location: location._id,
            'event_occurrences.date_to': { $gte: today },
            status: { $ne: 'draft' },
        };

        return this.eventsService.findAll(filters, null,
{'event_occurrences.date_from' : 1});
    }
}

```

```

// locations.service.ts

@Injectable()
export class LocationsService extends ApiBaseService<
    LocationDocument,
    CreateLocationDto,
    UpdateLocationDto
> {
    constructor(@InjectModel(Location.name) model:
Model<LocationDocument>) {
        const sortMap = new Map();
        sortMap.set('name', { name_ua: 1 });
        sortMap.set('new', { created_at: -1 });

        super(model, {
            populateFields: ['city'],
            sort: 'name',
            sortMap,
        });
    }
}

```

```

// me.module.ts

```

```

@Module({

```

```

    imports: [
      MongooseModule.forFeature([ { name: User.name, schema: UserSchema
    })
  ],
  providers: [UsersService],
  controllers: [MeController],
})
export class MeModule {}

```

```
// me.controller.ts
```

```

@Controller('me')
export class MeController {
  constructor(private service: UsersService) {}

  @Get()
  @UseGuards(UserAuthGuard)
  getMe(@Req() request: any) {
    const id = request.user?.id ?? request.user?.user_id;
    if (!id) {
      throw new BadRequestException('User id not provided.')
    }

    if (Types.ObjectId.isValid(id)) {
      return this.service.getOne(id);
    }

    return this.service.findOne({ legacy_id: id });
  }

  @Patch()
  @UseGuards(UserAuthGuard)
  update(@Req() request: any, @Body() body: UpdateUserDto) {
    const id = request.user?.id ?? request.user?.user_id;
    const data: Partial<UpdateUserDto> = {
      fullname: body.fullname,
      email: body.email,
      phone_number: body.phone_number,
      preferences: body.preferences,
    };

    return this.service.update(id, data);
  }
}

```

```

@Delete()
@UseGuards(UserAuthGuard)
async delete(@Req() request: any) {
  const id = request.user?.id ?? request.user?.user_id;
  await this.service.delete(id);
  return { message: 'Successfully deleted.' };
}
}

```

```
// api-base.service.ts
```

```

export interface PagedResult<T> {
  data: T[];
  total: number;
  page: number;
  pageSize: number;
  totalPages: number;
  page_size: number;
  total_pages: number;
}

```

```

export interface ServiceConfig {
  populateFields?: string | string[];
  sort?: string;
  sortMap?: Map<string, string>;
}

```

```

export interface QueryParams {
  page: string;
  pageSize: string;
  sort: string;
}

```

```

@Injectable()
export class ApiBaseService<T, C, U> {
  constructor(protected model: Model<T>, protected config?:
ServiceConfig) {}

```

```

  async getOne(id: string): Promise<T> {
    const model = (await this.model
      .findById(new Types.ObjectId(id))
      .exec()
      .then((res) => this.populateResponse(res))) as T;

```

```
if (!model) throw new NotFoundException();
```

```
return model;
}
```

```
async findOne(query: FilterQuery<T>): Promise<T> {
  const model = (await this.model
    .findOne(query)
    .exec()
    .then((res) => this.populateResponse(res))) as T;
```

```
if (!model) throw new NotFoundException();
```

```
return model;
}
```

```
async getOneBySlugOrId(slugOrId: string): Promise<T> {
  const filters = this.createFiltersBySlugOrId(slugOrId);
  const model = (await this.model
    .findOne({ $or: filters })
    .exec()
    .then((res) => this.populateResponse(res))) as T;
```

```
if (!model) throw new NotFoundException();
```

```
return model;
}
```

```
async findAll(
  query: FilterQuery<T>,
  projection?: ProjectionType<T> | null | undefined,
  options?: QueryOptions<T> | null | undefined,
  queryParams?: Partial<QueryParams>
): Promise<PagedResult<T>> {
  const page = parseInt(queryParams?.page || '1', 10) || 1;
  const pageSize = parseInt(queryParams?.pageSize || '9999', 10) || 9999;
  const skip = (page - 1) * pageSize;
  const sort = this.getSort(queryParams?.sort);
```

```
const [data, total] = await Promise.all([
  this.model
    .find(query, projection, {
      ...options,
      limit: pageSize,
      skip: skip,
```

```

    })
    .sort(sort)
    .exec()
    .then((res) => this.populateResponse(res)),
    this.model.countDocuments(query).exec(),
  ]);

```

```

// Calculate the total number of pages
const totalPages = Math.ceil(total / pageSize);

```

```

return {
  data,
  total,
  page,
  pageSize,
  totalPages,
  page_size: pageSize,
  total_pages: totalPages,
};
}

```

```

async create(data: C): Promise<T> {
  const created = new this.model(data);
  return created.save();
}

```

```

async update(id: string, data: Partial<U>) {
  // TODO: Figure out why this is necessary
  return this.model
    .findByIdAndUpdate(id, data as any, {
      new: true,
    })
    .exec();
}

```

```

async delete(id: string) {
  return this.model.findOneAndDelete({ _id: id }).exec();
}

```

```

async deleteMany(filter: FilterQuery<T>) {
  return this.model.where(filter).deleteMany().exec();
}

```

```

getSort(sort?: string) {
  const sortKey = sort || this.config?.sort;

```

```

        return (
            (this.config?.sortMap      &&      sortKey      &&
this.config.sortMap.get(sortKey)) ||
            {}
        );
    }

    getPopulate() {
        return this.config?.populateFields || [];
    }

    async populateResponse(res: any) {
        const populateFields = this.getPopulate();
        for (const path of populateFields) {
            await this.model.populate(res, { path });
        }
    }

    return res;
}

createFiltersBySlugOrId(slugOrId: string) {
    const filters: FilterQuery<T>[] = [{ slug: slugOrId }];

    if (Types.ObjectId.isValid(slugOrId)) {
        filters.push({ _id: new Types.ObjectId(slugOrId) });
    }

    return filters;
}
}

```

Клієнтська частина

```
// /app/layout.tsx
```

```
const openSans = Open_Sans({
  subsets: ['latin', 'cyrillic'],
  variable: '--display-font',
});
```

```
const montserrat = Montserrat({
  subsets: ['latin', 'cyrillic'],
  variable: '--body-font',
});
```

```
export const dynamic = 'force-dynamic';
```

```
export const metadata: Metadata = {
  title: AppMeta.TITLE,
  description: AppMeta.DESCRIPRION,
  openGraph: {
    title: AppMeta.TITLE,
    description: AppMeta.DESCRIPRION,
    type: 'website',
  },
  icons: {
    icon: '/favicon.ico',
  },
  other: {
    'verify-admitad': '2fae7df94f'
  }
};
```

```
const RootLayout: FC<PropsWithChildren> = ({ children }) => {
  return (
    <html>
      <body className={` ${openSans.variable} ${montserrat.variable}`} >
        <ReduxProvider>
          <ToastProvider>{children}</ToastProvider>
        </ReduxProvider>
      </body>
    </html>
  );
};
```

```
export default RootLayout;
```

```
// /app/(web)/layout.tsx
```

```
const RootLayout: FC<PropsWithChildren> = ({ children }) => {
  return (
    <div className={styles.layout}>
      <Header />
      <main>{children}</main>
      <Footer />

      <Dialog />
    </div>
  );
};

export default RootLayout;
```

```
// header.tsx
```

```
const Header = () => {
  const [bgColor, setBgColor] = useState(false);

  const handleChangeColor = () => {
    if (window.scrollY > 30) {
      setBgColor(true);
    } else {
      setBgColor(false);
    }
  };

  useEffect(() => {
    if (window.scrollY > 30) {
      setBgColor(true);
    }
  });

  window.addEventListener("scroll", handleChangeColor);
}, []);

const citySlug = getCookie(StorageKey.SELECTED_CITY);

return (
  <header className={` ${styles.header} ${bgColor ? styles.fill : ""} `>
    <Container>
      <div className={styles["header-inner"]}>
```

```

    <div className={styles["left-items"]} >
      <Link href={`${AppPath.HOME}${citySlug || ""}`} >
        <ArthouseLogo />
      </Link>

```

```

    <CitySelect className={styles["city-select"]} />
  </div>
</div>
</Container>
</header>
);
};

```

```
export { Header };
```

```
// footer.tsx
```

```

const Footer = () => {
  return (
    <footer className={styles.footer}>
      <Container>
        <div className={styles["footer-inner"]} >
          <div className={styles.menu}>
            <div className={styles["menu-top"]} >
              <span className={styles.copyright}>© ArtHouse 2024</span>
            </div>
          </div>
        </div>
      </Container>
    </footer>
  );
};

```

```
export { Footer };
```

```
// /app/(web)/page.tsx
```

```

interface HomePageProps {
  params: { citySlug: string };
}

```

```

const HomePage = async ({ params }: HomePageProps) => {
  let selectedCity: ICity | null = null;

```

```

    if (params.citySlug) {
      selectedCity = await citiesService.findById(params.citySlug);
    } else {
      const citySlug = cookies().get(StorageKey.SELECTED_CITY)?.value;

      if (citySlug) {
        redirect(`/${citySlug}`);
      }

      selectedCity = await citiesService.findById(CitySlug.DEFAULT);
    }

    const events = await eventsService.findAll({
      pageSize: "12",
      city: selectedCity._id,
    });

    const { data: locations } = await locationsService.findAll({
      pageSize: "5",
      city: selectedCity._id,
    });

    return (
      <div className={styles["home-page"]} >
        <div className={styles.content} >
          <HeroSection events={events.data.slice(0, 6)} />
          <EventsGrid events={events} city={selectedCity} />
          <Locations locations={locations} />
        </div>
      </div>
    );
  };
};

export default HomePage;

// /app/(web)/[city]/e/[slug]/page.tsx

type EventPageProps = {
  params: { slug: string };
};

export async function generateMetadata({
  params,
}: EventPageProps): Promise<Metadata> {

```

```

const slug = params.slug;
const event = await eventsService.findById(slug);

const date = formatEventDate(
  event?.event_occurrences[0]?.date_from,
  event?.event_occurrences[0]?.date_to
);

const title = `${event?.title}`;
const description = `Купити квитки ${event?.title} ${date}`;

return {
  title: title,
  description: description,
  openGraph: {
    title: title,
    description: description,
    images: [event?.picture_url],
  },
};
}

const EventPage = async ({ params }) => {
  const selectedCitySlug =
    cookies().get(StorageKey.SELECTED_CITY)?.value ||
CitySlug.DEFAULT;
  const selectedCity = await citiesService.findById(selectedCitySlug);

  const eventSlugOrId = params.slug as string;
  const event = await eventsService.findById(eventSlugOrId);

  const promEvents = await eventPromotionsService.findAll({
    city: selectedCity._id,
  });

  const promLocations = await locationPromotionsService.findAll({
    city: selectedCity._id,
  });

  const locationLink =
`/${event.location.city.slug}/l/${event.location.slug}`;

  return (
    <div className={styles.page}>
      <div className={styles["page-hero"]}>

```

```

<Image
  className={styles["hero-image"]}
  src={event.picture_url}
  alt={event.title}
  width="400"
  height="200"
  unoptimized
/>

```

```

<div className={styles["hero-backdrop"]} ></div>
</div>

```

```

<div className={styles["content-wrapper"]} >
  <Container>
    <div className={styles.content}>
      <div className={styles["more-info"]} >
        <div className={styles["about-event"]} >
          <h2 className={styles.title}>Про подію</h2>
          <p className={styles.description}>{event.description}</p>
        </div>
      </div>
    </div>
  </div>

```

```

<div className={styles["about-location"]} >
  <h2 className={styles.title}>Про локацію</h2>

```

```

<div className={styles["location-info"]} >
  <Link href={locationLink}>
    <Image
      src={event.location.image_link}
      alt={event.title}
      width="400"
      height="200"
      className={styles.img}
      unoptimized
    />
  </Link>

```

```

<div className={styles.info}>
  <Link href={locationLink}>
    <h3 className={styles.title}>
      {event.location_name || event.location.name_ua}
    </h3>
  </Link>

```

```

<div className={styles["additional-info"]} >
  {event.location.google_link && (

```

```

    <Link href={event.location.google_link} target="_blank">
      <div className={styles.location}>
        <RiMapPin2Line className={styles.icon} />
        <span>{event.location.address_ua}</span>
      </div>
    </Link>
  )}

```

```

    {event.location.website_link && (
      <Link href={event.location.website_link}>
        <div className={styles.site}>
          <RiGlobalLine className={styles.icon} />
          <span>Веб-сайт</span>
        </div>
      </Link>
    )}
  </div>
</div>
  {event.location.description && (
    <QuillText value={event.location.description} />
  )}
</div>
</div>
</div>

```

```

    <div className={styles["card-wrapper"]} >
      <EventCard event={event} />
    </div>
  </div>
</Container>

```

```

    <PageGradient />
  </div>

```

```

  {promEvents.data.length > 0 && (
    <div className={styles["feature-events-slider-wrapper"]} >
      <FeatureEventCardsSlider
        title="Топ події"
        events={promEvents.data.map((it) => it.event)}
      />
    </div>
  )}

```

```

  {promLocations.data.length > 0 && (
    <div className={styles["locations-slider-wrapper"]} >

```

```

        <LocationsSlider
          title="Топ локації"
          locations={promLocations.data.map((it) => it.location)}
        />
      </div>
    )}
  </div>
);
};

export default EventPage;

// /app/(web)/[city]/l/[slug]/page.tsx
interface ILocationPageProps {
  params: { slug: string };
}

export async function generateMetadata({
  params,
}: ILocationPageProps): Promise<Metadata> {
  const location = await locationsService.findById(params.slug);
  const year = new Date().getFullYear();

  const title = `${location?.name_ua}: ${year}`;
  const description = `${location?.name_ua}`;

  return {
    title: title,
    description: description,
    openGraph: {
      title: title,
      description: description,
      images: [location?.image_link],
    },
  };
}

const LocationPage = async ({ params }: ILocationPageProps) => {
  const id = params.slug as string;
  const location = await locationsService.findById(id);

  const selectedCitySlug =

```

```

        cookies().get(StorageKey.SELECTED_CITY)?.value ||
CitySlug.DEFAULT;
        const selectedCity = await citiesService.findById(selectedCitySlug);

        const { data: events } = await eventsService.findAll({
            location: location._id,
        });

        const { data: promEvents } = await eventPromotionsService.findAll({
            city: selectedCity._id,
        });

        const { data: promLocations } = await
locationPromotionsService.findAll({
            city: selectedCity._id,
        });

        return (
            <div>
                <div className={styles["container-wrapper"]}>
                    <Container>
                        <LocationHero location={location} />
                        <LocationEvents events={events} />
                    </Container>

                    {events.length > 0 && promEvents.length > 0 && (
                        <div className={styles["events-slider-wrapper"]}>
                            <FeatureEventCardsSlider
                                title="Топ події"
                                events={promEvents.map((it) => it.event)}
                            />

                            {/* <EventCardsSlider
                                title="Топ події"
                                events={promEvents.map((it) => it.event)}
                            /> */}
                        </div>
                    )}

                    {events.length > 0 && promLocations.length > 0 && (
                        <div className={styles["locations-slider-wrapper"]}>
                            <LocationsSlider
                                title="Топ локації"
                                locations={promLocations.map((it) => it.location)}
                            />

```

```

    </div>
  })

```

```

    <PageGradient />
  </div>
</div>
);
};

```

```
export default LocationPage;
```

```
// api-base.service.ts
```

```

export interface PageResult<T> {
  data: T[];
  total: number;
  page: number;
  pageSize: number;
  totalPages: number;
}

```

```

export class ApiBaseService<T, C> {
  protected API_URL = process.env.API_URL;

```

```
  protected API_KEY = process.env.API_KEY;
```

```
  protected IS_SERVER = typeof window === 'undefined';
```

```
  constructor(protected model: string) {}
```

```

  async findAll(queryParams?: Record<string, string>):
  Promise<PageResult<T>> {

```

```
    const queryString = new URLSearchParams();
```

```

    if (queryParams) {
      Object.keys(queryParams).forEach((key) => {
        queryString.append(key, queryParams[key]);
      });
    }
  }

```

```

  const url = this.IS_SERVER
    ? `${this.API_URL}/${this.model}?api_key=${
      this.API_KEY
    }&${queryString.toString()}`

```

```
: `/api/${this.model}?${queryString.toString()}`;
```

```
const response = await fetch(url);
const data = await response.json();
```

```
return data;
}
```

```
async findById(id: string): Promise<T> {
  const url = this.IS_SERVER
    ? `${this.API_URL}/${this.model}/${id}?api_key=${this.API_KEY}`
    : `/api/${this.model}/${id}`;
```

```
const response = await fetch(url);
const data = await response.json();
```

```
return data;
}
```

```
async create(data: C): Promise<T> {
  const url = this.IS_SERVER
    ? `${this.API_URL}/${this.model}`
    : `/api/${this.model}`;
```

```
const response = await fetch(url, {
  method: 'POST',
  body: JSON.stringify(data),
  headers: {
    'Content-type': 'application/json',
  },
});
const created = await response.json();
```

```
return created;
}
}
```

ДОДАТОК Б
Слайди презентації

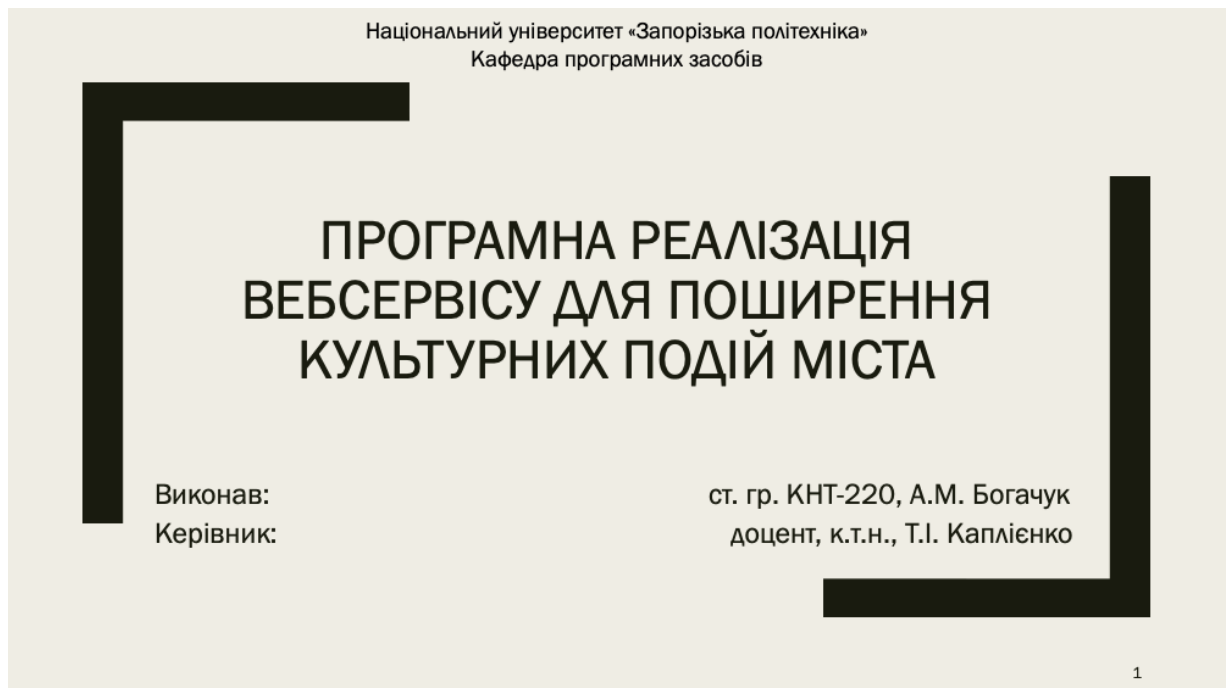


Рисунок Б.1 – Слайд №1

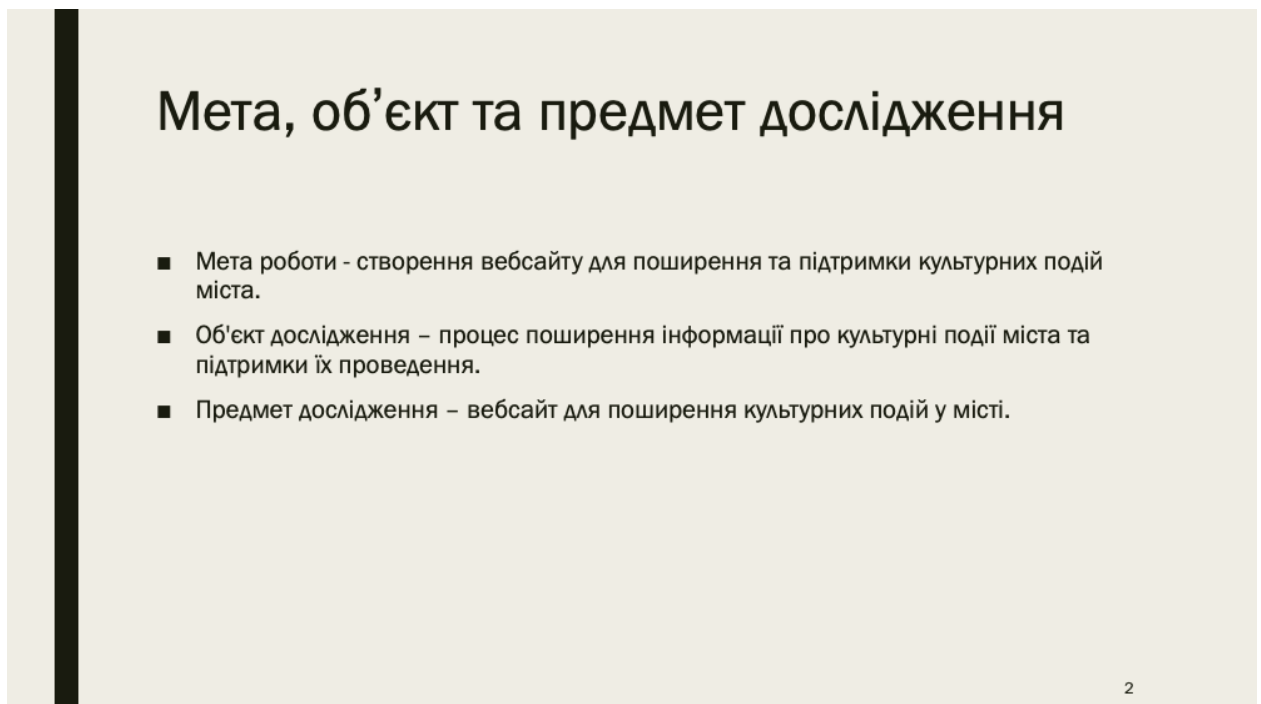


Рисунок Б.2 – Слайд №2

Завдання до роботи

- Аналіз предметної області
- Вибір мови програмування та фреймворків
- Розробка архітектури вебзастосунку
- Розробка вебзастосунку
- Тестування та експериментальне дослідження програмного забезпечення

3

Рисунок Б.3 – Слайд №3

Порівняння існуючих аналогів

Критерій порівняння	Concert.ua	Karabas	Musin
Можливість онлайн-купівлі квитків	Так	Так	Так
Фільтри пошуку подій	Дата, тип події	Дата, тип події	Дата, тип події
Інтерфейс користувача	Зручний, з фільтрами та категоріями	Зручний, з фільтрами та категоріями	Зручний, з фільтрами та категоріями
Наявність детальної інформації	Повна інформація про події, але мало інформації про локації	Мало інформації про події та локації	Достатньо інформації про події, але відсутня про локації
Доступність на різних платформах	Вебсайт	Вебсайт	Вебсайт
Мова інтерфейсу	Українська, англійська	Українська, російська, англійська	Українська, російська

4

Рисунок Б.4 – Слайд №4

Вибір мови програмування

Критерій порівняння	Java	Python	JavaScript	TypeScript
Сфера застосування	Веброзробка, мобільні застосунки	Веброзробка, наука, аналіз даних, машинне навчання	Веброзробка	Веброзробка
Типізація	Статична	Динамічна	Динамічна	Статична
Продуктивність	Висока	Висока	Висока для веброзробки	Висока для веброзробки
Легкість вивчення	Середня	Висока	Висока	Середня
Підтримка масштабування	Висока	Висока	Середня	Висока
Розгортання та інтеграція	Складна	Проста	Проста	Проста

5

Рисунок Б.5 – Слайд №5

Вибір фреймворку для клієнтської частини

Критерій порівняння	React.js	Angular	Vue.js	Next.js
Тип фреймворку	Бібліотека для побудови інтерфейсу	Повноцінний фреймворк	Прогресивний фреймворк	Фреймворк на базі React.js
Синтаксис	JSX	TypeScript/JavaScript	JavaScript	JSX, TypeScript
Легкість вивчення	Середня	Низька	Середня	Середня
Можливості SEO	Потребує налаштування	Потребує налаштування	Потребує налаштування	Висока, з вбудованою підтримкою SSR
Спільнота	Велика, активна спільнота	Велика, активна спільнота	Велика, активна спільнота	Велика, активна спільнота, з підтримкою від Vercel

6

Рисунок Б.6 – Слайд №6

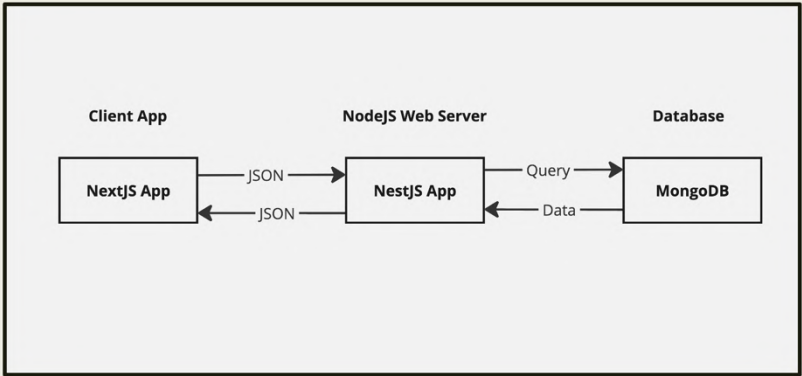
Вибір фреймворку для серверної частини

Критерій порівняння	Express.js	NestJS	Koa.js
Тип фреймворку	Мінімалістичний фреймворк	Повноцінний фреймворк з модульною архітектурою	Мінімалістичний фреймворк
Синтаксис	JavaScript, TypeScript	TypeScript	JavaScript, TypeScript
Легкість вивчення	Висока	Середня	Висока
Продуктивність	Залежить від налаштувань	Висока	Висока
Можливості масштабування	Залежить від реалізації	Висока	Залежить від реалізації
Спільнота	Велика та активна	Велика та активна	Середня
Підтримка TypeScript	Обмежена, потребує додаткових налаштувань	Повна, TypeScript є основною мовою	Обмежена, потребує додаткових налаштувань

7

Рисунок Б.7 – Слайд №7

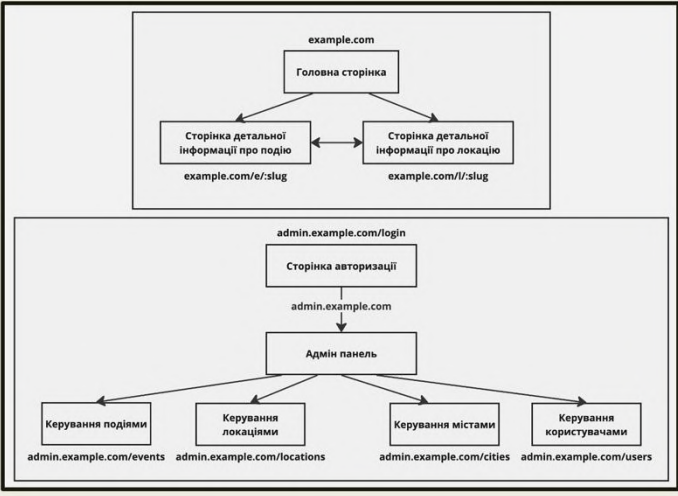
Архітектура вебзастосунку



8

Рисунок Б.8 – Слайд №8

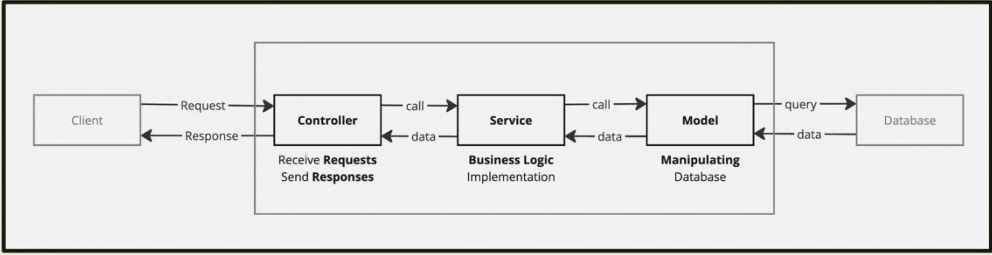
Структура клієнтської частини



9

Рисунок Б.9 – Слайд №9

Структура серверної частини



10

Рисунок Б.10 – Слайд №10

Структура бази даних

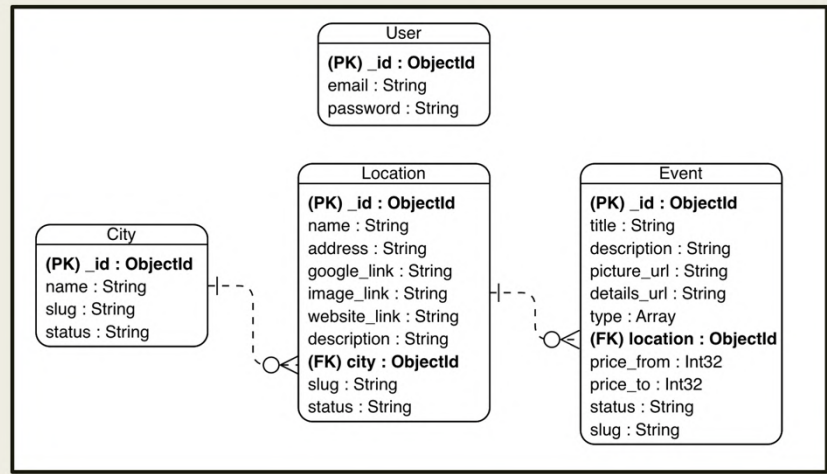


Рисунок Б.11 – Слайд №11

Діаграма класів

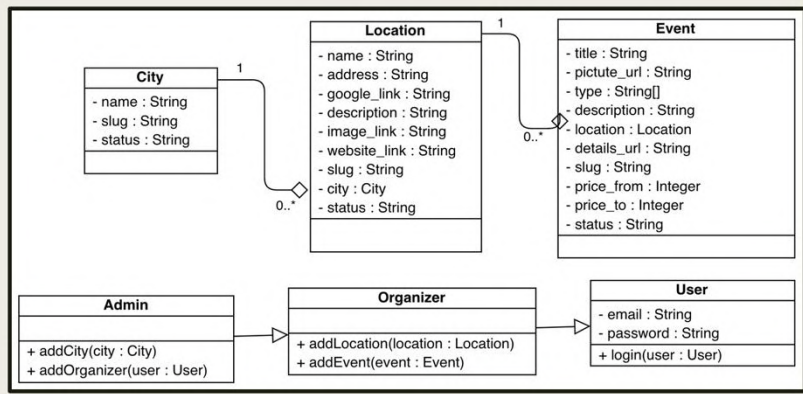
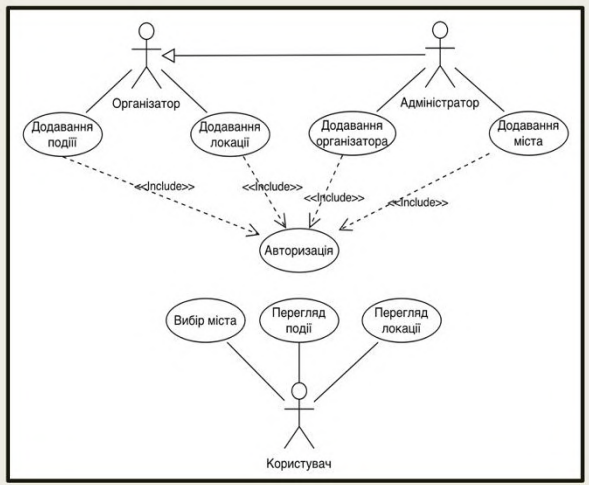


Рисунок Б.12 – Слайд №12

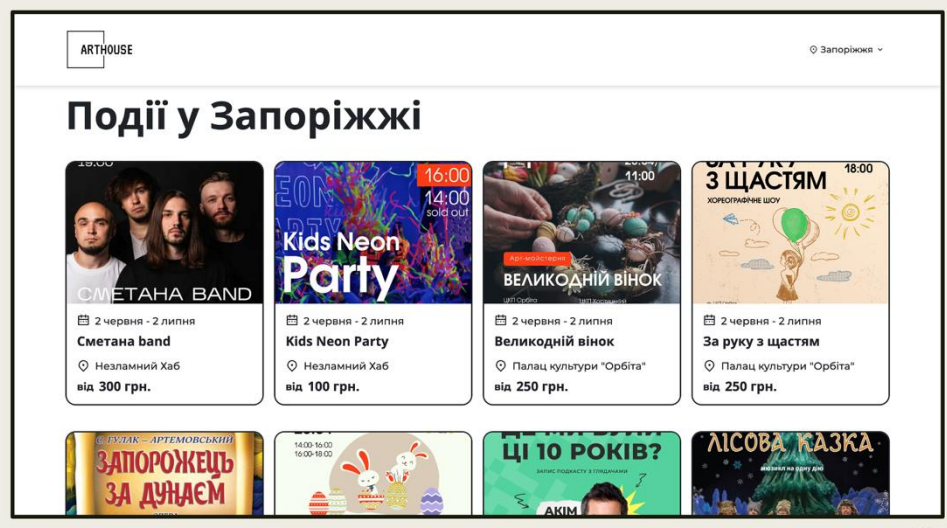
Діаграма прецедентів



13

Рисунок Б.13 – Слайд №13

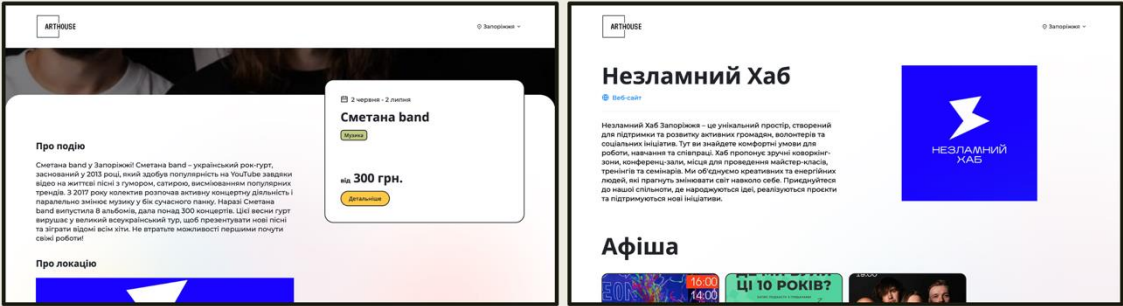
Інтерфейс головної сторінки



14

Рисунок Б.14 – Слайд №14

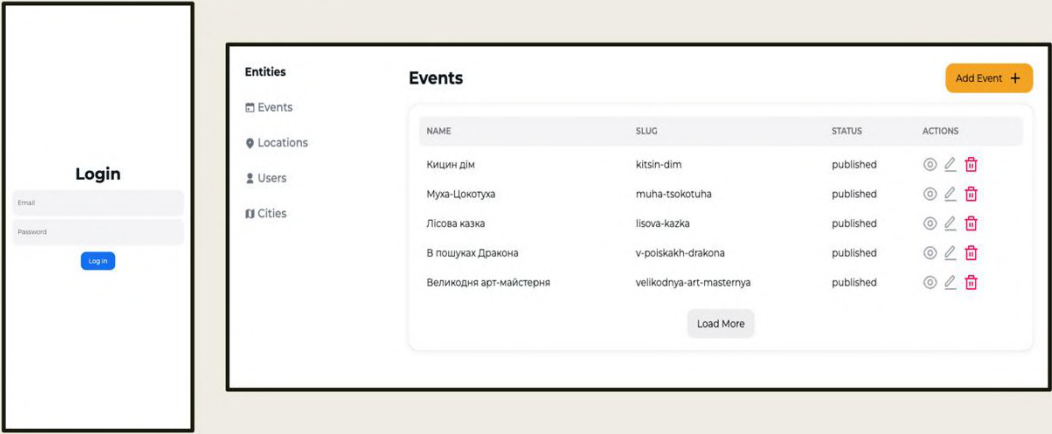
Інтерфейс сторінок події та локації



15

Рисунок Б.15 – Слайд №15

Інтерфейс сторінок авторизації та адміністратора

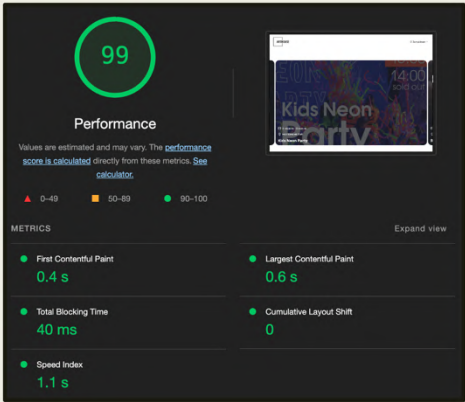


16

Рисунок Б.16 – Слайд №16

Результати функціонального тестування

Тестований компонент	Результат	Примітки
Головна сторінка	Успішно	Завантаження <1 секунди
Перегляд подій	Успішно	Відображення без помилок
Перегляд локацій	Успішно	Відображення без помилок

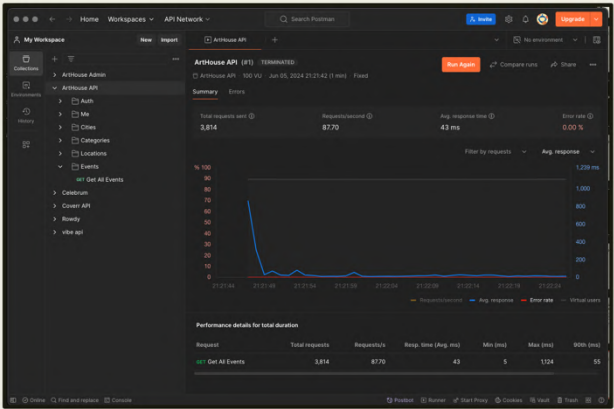


17

Рисунок Б.17 – Слайд №17

Результати навантажувального тестування

Тестований аспект	Результат	Примітки
Кількість користувачів	100 одночасно	Без втрати продуктивності
Середній час відповіді	43 мс	Стабільний при високих навантаженнях
Відсоток помилок	0%	Тестування проходить без неочікуваних помилок



18

Рисунок Б.18 – Слайд №18

Висновки

- Проведено всебічний аналіз існуючих рішень для поширення інформації про культурні події міста.
- Для реалізації проєкту були обрані мови програмування та фреймворки, які найкраще підходять для даного типу проєкту: TypeScript, Next.js для клієнтської частини та NestJS для серверної частини.
- Створено вебсайт, що забезпечує швидкий та зручний доступ до інформації про культурні події міста, інтерфейс, якого дозволяє користувачам з легкістю знаходити потрібну інформацію та отримувати детальні відомості про події та локації.
- Проведено функціональне та навантажувальне тестування вебсайту, що підтвердило ефективність вебсайту у реальних умовах експлуатації.
- Результати роботи випробувані на науково-практичній конференції «Тиждень науки-2024», яка дистанційно проводилася у Національному університеті «Запорізька політехніка» у 2024 році.

19

Рисунок Б.19 – Слайд №19

Подальші вдосконалення

- Збільшення кількості подій та локацій
- Додавання можливості користувачам залишати відгуки та оцінки подіям та локаціям
- Функція, що дозволить користувачам зберігати події та локації до списку обраного для швидкого доступу в майбутньому
- Впровадження різноманітних фільтрів для пошуку подій за жанром, датою, місцем проведення тощо
- Додавання опцій сортування подій за популярністю, датою, рейтингом та іншими критеріями
- Впровадження пошукової системи для швидкого і точного знаходження подій та локацій

20

Рисунок Б.20 – Слайд №20