МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБСАЙТУ ДЛЯ ПОПУЛЯРИЗАЦІЇ ТУРИСТИЧНИХ ЛОКАЦІЙ»**

Виконав: студент 3 курсу, групи 6.1212-пі-с
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія (зі скороченим терміном навчання)
(назва освітньої програми)
Д.Й. Луців
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, PhD Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія (зі скороченим терміном навчання)

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Луціву Данилу Йосиповичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебсайту для популяризації туристичних локацій

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Проєктування інформаційної системи.
3. Реалізація та тестування вебсайту.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	13.01.2025	
2.	Збір вихідних даних.	31.01.2025	
3.	Обробка методичних та теоретичних	21.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	04.04.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	20.06.2025	

Студент _____ Д.Й. Луців
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебсайту для популяризації туристичних локацій»: 50 с., 17 рис., 2 табл., 8 джерел.

АДАПТИВНІСТЬ САЙТУ, БЕКЕНД, ВЕБДИЗАЙН, ВЕБРОЗРОБКА, ІНТЕРАКТИВНА КАРТА, МАРШРУТИ, ПОПУЛЯРИЗАЦІЯ ТУРИЗМУ, РОЗРОБКА ВЕБСАЙТУ, ТУРИСТИЧНІ ЛОКАЦІЇ, ФРОНТЕНД.

Об’єкт дослідження – процеси розробки інформаційних вебресурсів для туристичної галузі.

Мета роботи: створення сучасного вебсайту для популяризації туристичних локацій, який забезпечує зручний доступ користувачів до інформації про визначні місця.

Методи дослідження – методи збору та аналізу вимог до інформаційно-пошукової системи, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

У кваліфікаційній роботі розглянуто процес розробки вебсайту для популяризації туристичних локацій м. Запоріжжя.

Для реалізації проєкту використано сучасні вебтехнології: Next.js для фронтенду, Node.js з Express для бекенду, база даних MongoDB, інтеграція з Google Maps API для візуалізації локацій. Інтерфейс оптимізовано під різні типи пристроїв, враховано вимоги до зручності та доступності.

У результаті створено повнофункціональний вебресурс, який може бути використаний як туристичними організаціями, так і місцевою владою для розвитку туристичної інфраструктури. У перспективі сайт може бути доповнений особистими кабінетами користувачів, рейтингами локацій, можливістю бронювання екскурсій, а також розширений на інші міста або регіони України.

SUMMARY

Bachelor’s qualifying paper “Development of a Website for Promoting Tourist Locations”: 50 pages, 17 figures, 2 tables, 8 references.

BACKEND, FRONTEND, INTERACTIVE MAP, ROUTES, TOURISM PROMOTION, TOURIST LOCATIONS, WEB DESIGN, WEB DEVELOPMENT, WEBSITE ADAPTABILITY, WEBSITE DEVELOPMENT.

The object of the study is the development processes of informational web resources for the tourism industry.

The aim of the study is to create a modern website for promoting tourist locations that provides users with convenient access to information about landmarks.

The research methodology includes methods of gathering and analyzing requirements for an information retrieval system, as well as methods of modeling, software design, development, and testing.

This qualifying paper explores the development process of a website aimed at promoting tourist locations in Zaporizhzhia.

The project was implemented using modern web technologies: Next.js for the frontend, Node.js with Express for the backend, MongoDB as the database, and integration with the Google Maps API for location visualization. The interface has been optimized for various types of devices, with special attention to usability and accessibility.

As a result, a fully functional web resource was created, which can be used by tourism organizations and local authorities to support and develop tourism infrastructure. In the future, the website may be enhanced with user accounts, location ratings, excursion booking functionality, and expanded to cover other cities or regions of Ukraine.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз предметної області..... 9
1.1 Визначення потреб користувачів і цільової аудиторії 9
1.2 Аналіз стану розвитку туристичних систем..... 10
1.3 Аналіз функціональних та нефункціональних вимог 13
2 Проєктування вебсайту..... 15
2.1 Вибір технологій для розробки вебсайту 15
2.2 Розробка архітектури сайту 17
2.2.1 Формування загальної структури системи..... 17
2.2.2 Моделі взаємодії та сценарії використання 20
2.2.3 Структура бази даних 23
2.3 Побудова функціональної структури сайту на основі EPC-моделі... 25
2.4 Проєктування інтерфейсу користувача (UI/UX) 27
3 Реалізація та тестування вебсайту..... 32
3.1 Розробка фронтенд-частини сайту..... 32
3.2 Розробка бекенд-частини сайту..... 36
3.3 Інтеграція інтерактивної карти туристичних локацій..... 38
3.4 Тестування функціональності, продуктивності та безпеки сайту 42
3.5 Оптимізація та адаптивність сайту для різних пристроїв..... 44
Висновки 48
Перелік посилань..... 50

ВСТУП

У сучасному світі розвиток туристичної галузі є важливим чинником соціально-економічного зростання регіонів. Значна кількість мандрівників звертається до Інтернету для планування поїздок, вибору цікавих місць та прокладання маршрутів. Це обумовлює **актуальність** створення інноваційних вебресурсів, які дозволяють зручно й доступно ознайомитися з туристичним потенціалом території. Особливо це актуально для міст, що не є традиційними туристичними центрами, зокрема для Запоріжжя, яке має багату історико-культурну спадщину, але потребує сучасних інструментів для її популяризації.

Мета дослідження – створення сучасного вебсайту для популяризації туристичних локацій, який забезпечує зручний доступ користувачів до інформації про визначні місця.

Досягнення поставленої мети передбачає реалізацію наступних **завдань**:

- проаналізувати існуючі вебресурси у сфері туризму;
- дослідити потреби користувачів та сформулювати функціональні та нефункціональні вимоги;
- розробити архітектуру сайту;
- спроєктувати інтерфейс користувача з урахуванням принципів зручності та адаптивності;
- реалізувати функціонал сайту за допомогою сучасних вебтехнологій;
- протестувати вебресурс на відповідність вимогам.

Об’єкт дослідження – процеси розробки інформаційних вебресурсів для туристичної галузі.

Предмет дослідження – методи та засоби розробки адаптивного вебсайту з інтерактивною картою туристичних локацій.

Методи дослідження – методи збору та аналізу вимог до інформаційно-пошукової системи, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

Для досягнення результату було використано мову програмування TypeScript, яка забезпечила реалізацію як клієнтської, так і серверної частин. У якості фреймворку для фронтенду та бекенду було обрано Next.js, що забезпечує SSR/CSR-рендеринг і високу продуктивність. Для зберігання даних – MongoDB, що дозволило ефективно організувати структуру інформації у форматі JSON. Для інтеграції інтерактивної карти використано Google Maps API.

У результаті розробки створено повнофункціональний вебсайт, що дозволяє переглядати туристичні об’єкти Запоріжжя, отримувати інформацію про них та будувати маршрути. Вебресурс може бути використаний туристичними організаціями, місцевою владою, а також окремими користувачами. У перспективі проєкт може бути розширений за рахунок додаткових модулів: реєстрації користувачів, рейтингових систем, інтеграції з соціальними мережами або перенесення функціоналу на мобільні платформи.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, та переліку посилань. Загальний обсяг роботи становить 50 сторінок. У переліку посилань наведено 8 джерел.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення потреб користувачів і цільової аудиторії

Успішність будь-якого вебресурсу, зокрема туристичного, значною мірою залежить від того, наскільки чітко визначено цільову аудиторію та її потреби.

На початковому етапі проєктування вебсайту доцільно здійснити аналіз майбутніх користувачів, їх очікувань, технічних можливостей та поведінкових патернів.

Основною цільовою аудиторією розроблюваного вебсайту є:

– місцеві жителі, які прагнуть відкрити для себе нові або маловідомі туристичні об’єкти у власному місті;
– туристи з інших регіонів України, які планують поїздку до Запоріжжя та потребують зручного інструменту для ознайомлення з визначними місцями;
– іноземні відвідувачі, зацікавлені у культурному, історичному або екотуризмі, що шукають базову інформацію англійською мовою;
– працівники туристичної галузі, гіди, туроператори та представники місцевих органів влади, які можуть використовувати сайт для поширення інформації та організації маршрутів.

Для кожної категорії користувачів характерні певні функціональні та нефункціональні потреби, які необхідно враховувати у процесі розробки.

Основні критерії включають:

– швидкий доступ до інформації про туристичні об’єкти (опис, фото, розташування, графік роботи тощо);
– інтерактивну карту, яка дозволяє знаходити об’єкти за категоріями (історичні, природні, розважальні тощо) і прокладати маршрути;
– багатомовність – підтримку української та англійської мови;

– мобільну адаптацію інтерфейсу – більшість користувачів здійснює пошук інформації з мобільних пристроїв;
– зручну навігацію та інтуїтивно зрозумілий інтерфейс, особливо для користувачів старшого віку або тих, хто має обмежений досвід користування цифровими сервісами.

Окремо слід відзначити, що значна частина користувачів очікує високу швидкість завантаження ресурсу, незалежно від типу пристрою чи швидкості інтернету. Це особливо важливо для туристів, які перебувають у дорозі. Тому легкий дизайн, оптимізовані зображення та відкладене завантаження контенту – це вимога не лише технічна, але й пряма реакція на запити цільової аудиторії.

Під час дослідження було враховано наявні аналоги – українські та міжнародні туристичні платформи, які виявляють як успішні підходи до подачі інформації, так і типові недоліки: перевантаженість інтерфейсу, застарілий дизайн, відсутність інтерактивних функцій, складна навігація або неповна інформація. Це дозволило сформулювати вимоги до функціональності сайту, які максимально орієнтовані на реальні потреби цільової аудиторії, а не лише на формальні вимоги до вебресурсу.

Таким чином, глибоке розуміння користувачів – основа побудови ефективного і зручного сервісу, що сприятиме не лише задоволенню інформаційних потреб, а й популяризації туристичної привабливості міста Запоріжжя.

1.2 Аналіз стану розвитку туристичних систем

У сучасному цифровому середовищі туристичні ресурси є одним з основних джерел інформації для мандрівників. Вони виконують функції інформування, навігації, планування маршрутів, бронювання послуг та створення позитивного іміджу регіону. Оцінка поточного стану розвитку таких ресурсів дозволяє виявити ефективні рішення, поширені недоліки та визначити напрямки покращення.

На глобальному рівні туристичні вебресурси, такі як izi.TRAVEL або Google Maps, пропонують комплексну інформацію про об’єкти, рейтинги, відгуки, інтерактивні карти та можливості бронювання. Вони мають високий рівень зручності, адаптивності до мобільних пристроїв та багатомовну підтримку. Однак більшість із них фокусуються на популярних туристичних напрямках та рідко надають повну інформацію про менш відомі регіони або міста, як-от Запоріжжя.

У таблиці 1.1 зведено порівняльну характеристику більш відомих інформаційних систем з розроблюваною.

Таблиця 1.1 – Порівняння туристичних ресурсів

	Основні функціональні можливості	**Переваги**	**Недоліки**
Туристичне Запоріжжя (*розроблюваний ресурс*) https://diploma-swart-xi.vercel.app/zaporizhzhia	– інтерактивна карта міста Запоріжжя; – опис пам’яток, музеїв, парків; – фільтрація за категоріями; – адмін-панель для оновлення контенту	– орієнтовано на конкретне місто; – сучасний веб-дизайн і зручний інтерфейс; – можливість динамічного оновлення контенту	потребує стабільного інтернет-з’єднання
Google Maps (*Google LLC*) https://www.google.com/maps	– глобальна навігація; – відгуки, фото, рейтинг локацій; – режим Street View	– потужний функціонал; – актуальні геодані; – інтеграція з іншими сервісами Google	– надлишок інформації; – відсутність вузької спеціалізації на культурних об’єктах Запоріжжя
Туристичний буклет	– короткий опис основних об’єктів; – фізична карта	не потребує інтернету	обмежена кількість інформації

Продовження таблиці 1.1

	Основні функціональні можливості	**Переваги**	**Недоліки**
Find & Follow (*український туристичний портал*) https://findandfollow.com.ua/	– туристичні маршрути; – опис локацій; – афіша подій	– актуальна інформація; – сучасний інтерфейс	– зосереджено переважно на великих містах (Київ, Львів); – інформація про Запоріжжя відсутня
izi.TRAVEL (*izi.TRAVEL B.V.*) https://izi.travel	– аудіогіди по містах і музеях; – геолокація для визначення об’єктів поруч	– багатомовність; – аудіоформат зручно в подорожі	– обмежена кількість українських маршрутів; – відсутність детальної інформації

Серед типових недоліків українських туристичних вебсайтів можна виокремити:

– відсутність або обмежена інтерактивна карта з можливістю фільтрації об’єктів;

– неадаптивний або застарілий дизайн, неорієнтований на мобільні пристрої;

– відсутність багатомовної підтримки;

– недостатньо повна або актуальна інформація;

– відсутність функціоналу побудови маршрутів.

Ці проблеми суттєво знижують ефективність використання таких ресурсів як інструменту для залучення туристів.

Таким чином, аналіз поточного стану розвитку туристичних вебсайтів засвідчує потребу у створенні нового функціонального, сучасного та адаптивного ресурсу для м. Запоріжжя. Такий вебсайт має включати

інтерактивну карту, структурований каталог локацій, багатомовність, можливість побудови маршрутів, мобільну адаптацію та оновлювану інформаційну базу. Це дозволить заповнити наявну нішу на ринку локального цифрового туризму та сприятиме розвитку внутрішнього й в'їзного туризму в регіоні.

1.3 Аналіз функціональних та нефункціональних вимог

Ефективна розробка вебресурсу починається з чіткого формулювання вимог до нього. Вимоги поділяються на функціональні – що саме система повинна робити, і нефункціональні – якою має бути якість її виконання. Визначення обох типів вимог дозволяє сформувати повне технічне завдання для реалізації вебсайту.

На основі аналізу потенційних користувачів визначено набір ключових функцій, що мають бути реалізовані у вебсайті.

Функціональні вимоги:

- користувач може переглядати інтерактивну карту Запоріжжя;
- користувач може переглянути список туристичних локацій з описом місця та фотографіями;
- користувач може переглянути детальну інформацію про локацію на мапі;
- адміністратор може додавати/редагувати/видаляти локації;
- користувач може виконувати пошук локацій по назві;
- користувач може фільтрувати певну категорію локацій;
- система повинна відображати на мапі локації;
- категоризація об'єктів;
- можливість фільтрувати локації за категоріями;
- користувач може обрати мову інтерфейсу на вибір;
- користувач може побачити свою локацію на мапі та прокласти маршрут до локації.

Нефункціональні вимоги:

– висока швидкодія при завантаженні мапи;
– зручність користування;
– адаптивність для мобільних пристроїв;
– інтуїтивно зрозумілий інтерфейс;
– безпека (обмеження доступу до сторінок адміністратора);
– надійність;
– сумісність з сучасними браузерами;
– масштабованість.

Загалом, визначені функціональні та нефункціональні вимоги формують основу для подальшого проєктування архітектури системи, вибору технологій і моделювання взаємодії між компонентами вебзастосунку.

2 ПРОЄКТУВАННЯ ВЕБСАЙТУ

2.1 Вибір технологій для розробки вебсайту

Вибір технологій для розробки вебсайту є критичним етапом, що визначає ефективність, масштабованість та зручність розробки. Було обрано сучасний стек технологій, який забезпечує високу продуктивність, гнучкість та зручність для користувачів.

Для розробки як клієнтської, так і серверної частини вебсайту обрано фреймворк Next.js [1], який працює в середовищі Node.js [2]. Next.js підтримує серверний рендеринг (SSR) та статичну генерацію сторінок (SSG), що покращує SEO-оптимізацію та швидкість завантаження сторінок. Це особливо важливо для туристичних сайтів, де користувачі очікують швидкого доступу до інформації.

Next.js дозволяє створювати API-роути безпосередньо в межах проєкту, що спрощує реалізацію серверної логіки, такої як обробка форм, автентифікація та взаємодія з базою даних. Це забезпечує зменшення залежностей від сторонніх серверів або окремих бекенд-фреймворків, що, в свою чергу, прискорює розгортання проєкту та знижує технічні витрати.

Для стилізації інтерфейсу використано Tailwind CSS [3], що дозволяє швидко створювати адаптивні та сучасні дизайни без необхідності написання великої кількості власних CSS-стилів. Tailwind забезпечує компонентний підхід до стилів, підтримку тем, мобільної адаптації, а також інтегрується з сучасними редакторами коду, що полегшує розробку.

Для зберігання даних про туристичні локації, маршрути та користувачів використано MongoDB [4] – документно-орієнтовану базу даних, яка забезпечує гнучкість у структурі даних та високу масштабованість.

MongoDB підтримує JSON-подібний формат даних, що дозволяє напряму зберігати об’єкти даних, мінімізуючи необхідність додаткових

трансформацій, які часто потрібні у реляційних БД при роботі з вкладеними структурами

Для візуалізації туристичних локацій на карті інтегровано Google Maps API [5], що надає можливість відображення інтерактивних карт, прокладання маршрутів та пошуку об’єктів поблизу. Це значно покращує користувацький досвід та дозволяє ефективно планувати подорожі. Використання маркерів, підказок, маршрутизатора та можливість відкриття Google Maps у новій вкладці з координатами – усе це покращує досвід користування сайтом.

Для забезпечення безпеки даних та автентифікації адміністраторів було використано JSON Web Tokens (JWT) [6]. Це дозволяє реалізувати захищену систему входу та управління сесіями користувачів без необхідності зберігання сесій на сервері. Токен зберігається у браузері користувача та автоматично додається до кожного запиту до захищених маршрутів. Це забезпечує масштабованість і безпечну реалізацію аутентифікації.

Також розглядались альтернативи:

– React – гнучкий, але потребує окремого сервера для SSR;
– Vue.js – зручний у деяких випадках, але менше поширений для SSR-рішень;
– PostgreSQL або MySQL як альтернатива MongoDB – потужні, але менш зручні при роботі з неструктурованими об’єктами;
– Firebase – сучасний бекенд як сервіс, однак менш контрольований і складніший у масштабуванні для складної авторизації.

Таким чином, обраний стек технологій забезпечує створення сучасного, швидкого та масштабованого вебсайту. Такий підхід дозволяє легко розширювати функціональність у майбутньому, додаючи нові можливості.

Крім того, обрані рішення є відкритими, добре документованими та мають активну спільноту з багатою екосистемою плагінів і регулярними оновленнями. Це забезпечує довготривалу підтримку проєкту, швидке розв’язання технічних труднощів і можливість гнучко впроваджувати нові функції без масштабних змін архітектури.

2.2 Розробка архітектури сайту

2.2.1 Формування загальної структури системи

Побудова архітектури вебзастосунку розпочинається з визначення основних сценаріїв взаємодії користувачів із системою. Для цього використовується діаграма варіантів використання, яка дозволяє визначити ролі (акторів) у системі та їхні функціональні можливості. Такий підхід є ефективним для етапу концептуального проєктування, оскільки забезпечує високий рівень абстракції при моделюванні логіки майбутнього вебсайту.

На рисунку 2.1 представлено діаграму варіантів використання, яка охоплює взаємодію двох основних акторів – звичайного користувача та адміністратора сайту.

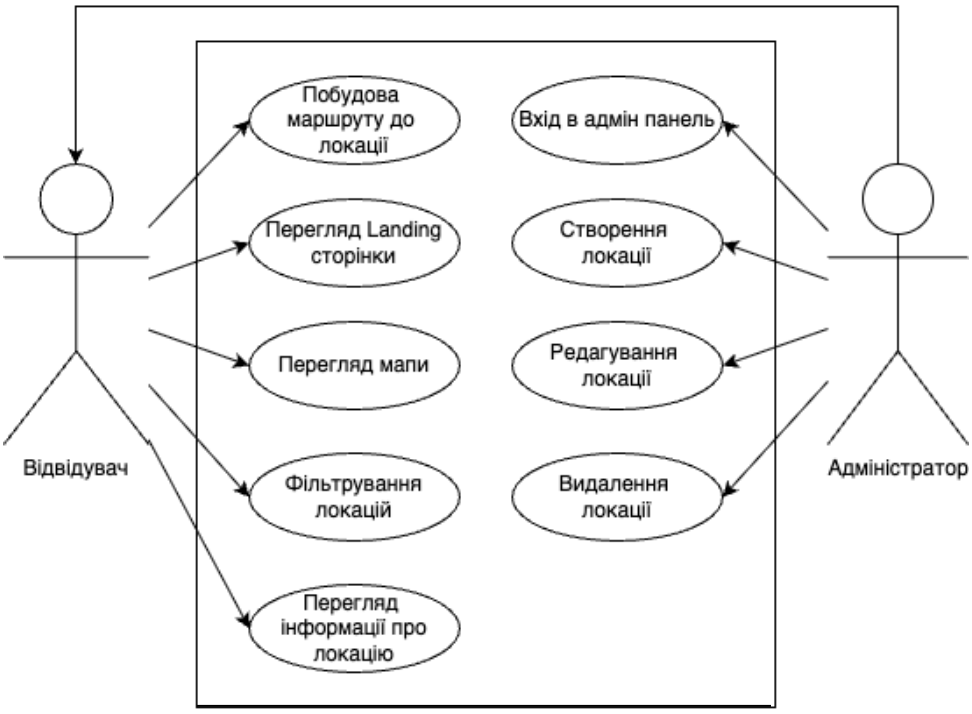


Рисунок 2.1 – Діаграма варіантів використання

Користувач має доступ до наступних функцій:

– перегляд списку туристичних локацій;
– фільтрація об’єктів на інтерактивній мапі;

– ознайомлення з детальною інформацією про кожну локацію;
– побудова маршруту до обраного об’єкта (через Google Maps API);
– перегляд Landing сторінки.

Адміністратор має розширений функціонал для управління контентом сайту:

– додавання нових локацій (з вказанням геоданих, опису, категорій, фото);
– редагування наявної інформації;
– видалення локацій.

Опишемо основні прецеденти відвідувача та адміністратора сайту.

Прецедент: «Побудова маршруту до локації».

Призначення: надати користувачеві можливість побудувати оптимальний маршрут до вибраної туристичної локації на інтерактивній карті, враховуючи його поточне місцезнаходження.

Основний потік подій:

– користувач відкриває сторінку з інтерактивною мапою;
– користувач обирає бажану локацію (точку на карті);
– система запитує дозвіл на доступ до поточного місцезнаходження;
– після отримання координат початкової точки, система звертається до картографічного API;
– система відображає його на мапі;
– користувач переглядає маршрут.

Передумова:

– користувач має доступ до інтернету;
– користувач дозволив використовувати своє місцезнаходження;
– локація, до якої будується маршрут, існує в системі.

Виняткові ситуації:

– користувач не надає дозволу на визначення місцезнаходження і не вводить адресу – система повідомляє про неможливість побудови маршруту;

– обрана локація недоступна або її координати не знайдено – система видає відповідне повідомлення;
– помилка при зверненні до картографічного API – система видає відповідне повідомлення.

Прецедент: «Створення локації».

Призначення: забезпечити можливість адміністратору додавати нові туристичні локації до системи для їх подальшого відображення на інтерактивній мапі сайту.

Основний потік подій:

– адміністратор входить у систему через адміністративну панель;
– обирає розділ «Додати локацію»;
– вводить назву локації, опис, категорію, додає зображення;
– вказує координати вручну або за допомогою інтерактивної мапи вибирає точку на карті;
– підтверджує створення локації;
– система зберігає дані про локацію в базі даних;
– нова локація автоматично з’являється на інтерактивній мапі користувацької частини сайту.

Передумова:

– адміністратор авторизований у системі;
– інтерфейс адміністративної панелі доступний та функціональний;
– мапа коректно відображається для введення координат або вибору точки.

Виняткові ситуації:

– введені не всі обов’язкові поля – система повідомляє про помилку та просить заповнити необхідні дані;
– введено некоректні координати – система відхиляє запит і вказує на помилку;
– помилка при збереженні даних у базу – система повідомляє про технічну помилку та пропонує повторити спробу пізніше.

Таким чином, побудова діаграми варіантів використання дозволяє системно окреслити функціональність сайту, сформувати початкове уявлення про його архітектуру та забезпечити основу для подальшого логічного та фізичного проєктування.

2.2.2 Моделі взаємодії та сценарії використання

Для розробки сучасних інформаційних систем надзвичайно важливо на початковому етапі створити чітку модель бізнес-процесів, що відображає логіку роботи майбутнього застосунку.

Візуалізація сценаріїв використання дозволяє узгодити розуміння системи між усіма учасниками проєкту.

Одним із найефективніших інструментів такого моделювання є BPMN (Business Process Model and Notation) [7] – графічна нотація, яка дозволяє формалізувати сценарії взаємодії між користувачами та системою, а також між різними підсистемами. BPMN підтримує інтуїтивно зрозумілий набір елементів: події, завдання, шлюзи, потоки та ролі (учасники процесу). Завдяки цьому модель легко читається, модифікується та масштабується.

У межах проєкту BPMN було обрано як основний підхід для моделювання ключових процесів вебплатформи, оскільки:

– дозволяє чітко описати поведінку користувача на сайті;
– враховує умови та варіанти розгалужень;
– може бути використаний як основа для побудови тест-кейсів та технічної реалізації.

BPMN є гнучким і водночас зрозумілим інструментом, який використовують як бізнес-аналітики, так і розробники для узгодження бачення функціональності. Саме тому для вебсайту популяризації туристичних локацій було створено BPMN-діаграму (див. рис. 2.2), яка охоплює ключові сценарії використання системи.

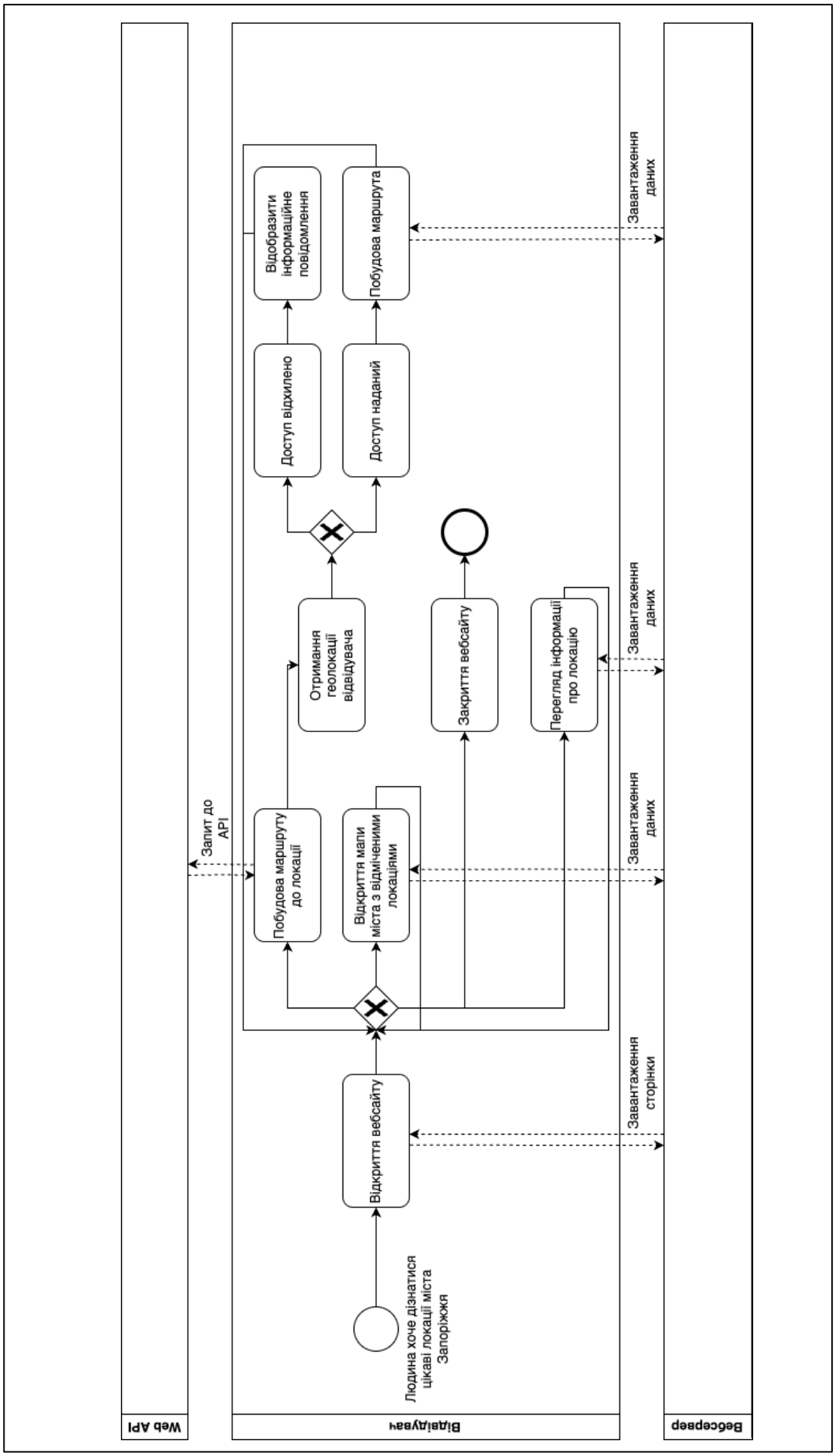


Рисунок 2.2 – BPMN-діаграма бізнес-процесів

На діаграмі представлено типову послідовність дій користувача:

– відкриття головної сторінки сайту;
– перегляд мапи з туристичними локаціями;
– перегляд детальної інформації про локацію;
– отримування геолокації користувача;
– побудова маршруту за допомогою інтеграції з Google Maps;
– відображення зворотного зв’язку.

Крім звичайних користувачів, у проєкті визначено ще одну роль – адміністратор, для якої також був сформований окремий BPMN-сценарій. На рисунку 2.3 зображено діаграму бізнес-процесів адміністратора.

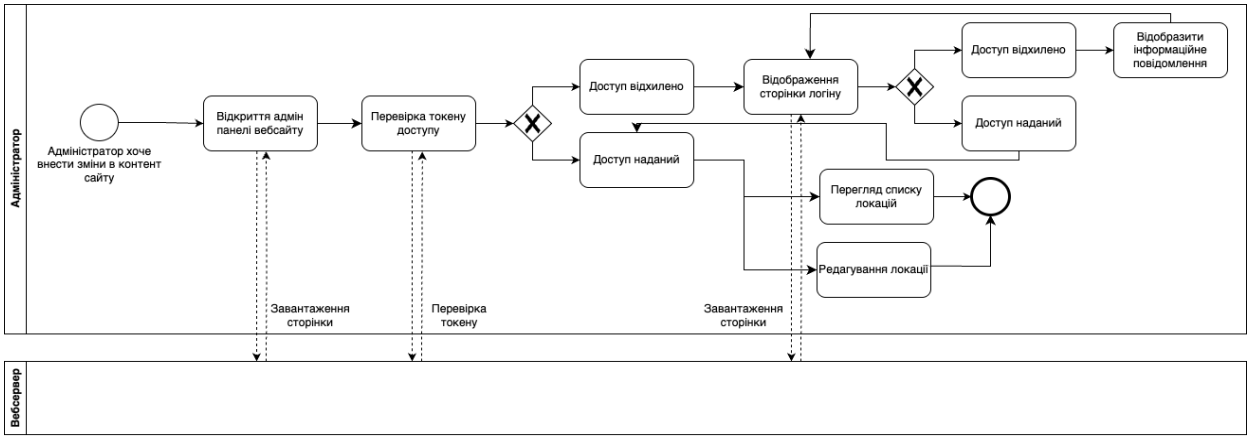


Рисунок 2.3 – BPMN-діаграма бізнес-процесів адміністратора

Сценарії, описані у діаграмі, безпосередньо вплинули на реалізацію відповідних компонентів інтерфейсу: кнопки побудови маршруту, вікна з детальною інформацією, обробку дозволу на доступ до геолокації, логіку переходів між сторінками.

Використання BPMN-діаграми на етапі проєктування вебсайту дозволяє системно й формально описати всі ключові сценарії взаємодії з системою. Це сприяє точнішій реалізації функціоналу, скороченню кількості помилок у роботі вебсайту, а також є основою для подальшого технічного та логічного проєктування. Побудовані BPMN-моделі охоплюють основні бізнес-процеси платформи і дозволяє ефективно спланувати та реалізувати логіку взаємодії користувачів із сервісом.

2.2.3 Структура бази даних

Для зберігання даних у розробленому вебзастосунку використовується документо-орієнтована база даних MongoDB. Цей тип СКБД дозволяє гнучко працювати зі структурами JSON-документів, легко масштабувати систему та зручно організовувати зв’язки між сутностями. На відміну від реляційних баз даних, MongoDB не вимагає жорсткої схеми, що пришвидшує розробку MVP-продукту.

У структурі бази даних передбачено такі основні колекції (див. рис. 2.4):

– location (зберігає дані про туристичні об’єкти: назву, опис, координати, зображення, категорію, статус);
– category (зберігає дані про категорію локації, що дозволяє фільтрувати локації на сайті);
– admin (зберігає дані про облікові записи адміністраторів для реалізації доступу до панелі управління).

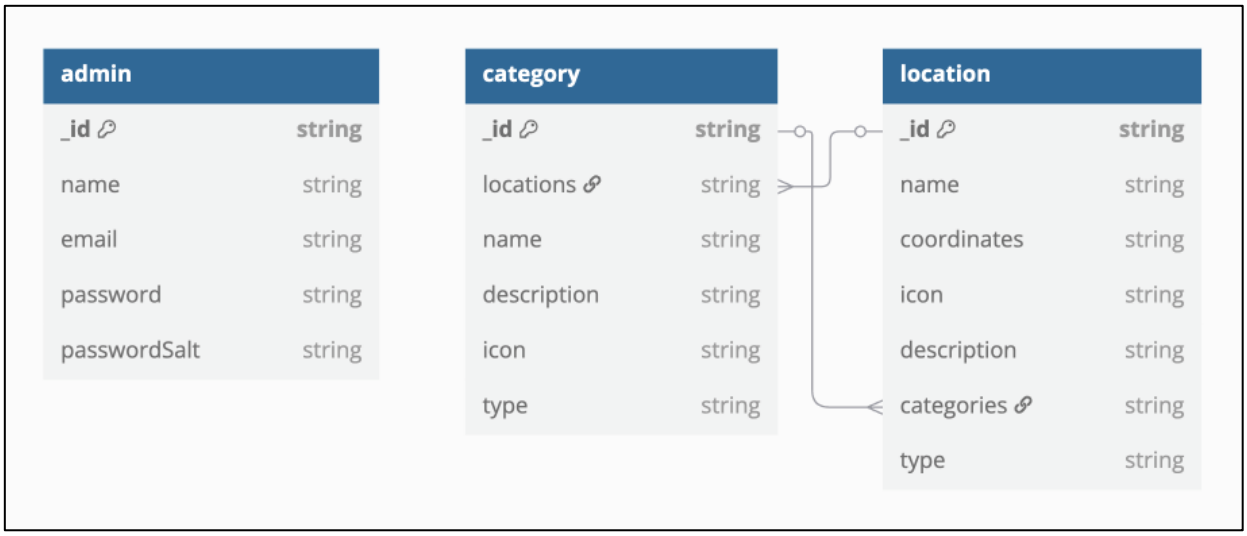


Рисунок 2.4 – Структура бази даних

На рисунку можна побачити структуру та схему зв’язків між основними колекціями. Колекція admin містить наступні поля:

– _id – унікальний ідентифікатор адміністратора (генерується автоматично);

– name – ім’я адміністратора;
– email – електронна пошта (використовується як логін);
– password – хешований пароль;
– passwordSalt – додатковий захист для зберігання паролів.

Колекція category:

– _id – унікальний ідентифікатор категорії (генерується автоматично);
– name – назва категорії, це поле містить в собі закодований в строку об’єкт JSON;
– description – короткий опис категорії, це поле містить в собі закодований в строку об’єкт JSON;
– icon – шлях до зображення категорії;
– type – визначає, тип категорії (для фільтрації або приховування неактуальних категорій);
– locations – масив _id локацій, до яких належить категорія.

Колекція location:

– _id – унікальний ідентифікатор локації;
– name – назва локації, це поле містить в собі закодований в строку об’єкт JSON;
– description – опис локації, це поле містить в собі закодований в строку об’єкт JSON;
– coordinates – JSON рядок з координатами у вигляді {"lat": number, "lng": number };
– icon – зображення або піктограма об’єкта;
– categories – масив _id категорій, до яких належить локація;
– type – додатковий тип об’єкта (наприклад, «екологічна», «історична» тощо).

У структурі бази даних особливу увагу приділено багатомовності контенту, що реалізується через зберігання текстових значень у форматі об’єктів JSON. Зокрема, поля name та description у колекціях location та category мають формат зображений на рисунку 2.5.

```
{
  "ua": "Текст українською мовою",
  "en": "Текст англійською мовою "
}
```

Рисунок 2.5 – Формат JSON для багатомовного контенту

Це дозволяє зручно реалізувати підтримку кількох мов у межах однієї структури без дублювання документів. Мовні значення обираються динамічно залежно від вибраної локалі користувача на сайті.

У колекціях використовуються вбудовані піддокументи та посилання на інші колекції, що дозволяє:

– оптимізувати запити до бази (наприклад, отримати список усіх локацій певної категорії);
– уникнути дублювання даних;
– легко додавати або змінювати категорії та локації без порушення цілісності системи.

MongoDB також забезпечує високу швидкодію при великій кількості читання, що є критично важливим для туристичного вебсайту, де більшість користувачів переглядають дані, а не створюють нові.

Таким чином, структура бази даних розроблена з урахуванням особливостей MongoDB та специфіки проєкту. Така модель забезпечує зручність у розробці, високу гнучкість при оновленні вмісту сайту та можливість масштабування функціоналу в майбутньому.

2.3 Побудова функціональної структури сайту на основі EPC-моделі

Для детального опису процесів вебсайту, доцільно використати EPC [8] діаграму. Цей тип моделювання широко застосовується в бізнес-аналізі для візуалізації логіки функціонування інформаційних систем і дозволяє наочно зобразити, які події ініціюють певні дії в системі та які рішення приймаються у процесі.

EPC-діаграми є зручним інструментом для документування функціональних ланцюгів дій, де кожна функція або подія пов’язана з наступною дією, логічною умовою або реакцією системи. Її використання дозволяє будувати чіткі моделі користувацької логіки, що слугують основою для технічного проєктування, тестування, а також майбутнього масштабування.

У межах поточного проєкту діаграма дозволяє відобразити:

– які кроки виконує користувач (від входу на сайт до перегляду інформації);
– які події викликають ті чи інші дії (наприклад, натискання на маркер локації);
– які логічні переходи (умови/альтернативи) відбуваються між етапами;
– як взаємодіють користувач та система на рівні процесів.

На рисунку 2.6 представлено EPC-діаграму, на якій відображено процес взаємодії користувача з сайтом.

Типовий процес взаємодії користувача з вебресурсом:

– початкова подія – користувач відкриває сайт (це є стартова точка всієї логіки, яка активується при першому завантаженні сайту);
– функція – завантаження головної сторінки (система відображає основний інтерфейс із мапою та списком локацій);
– подія – користувач натискає на маркер локації (тригер взаємодії користувача з інтерактивною картою);
– функція – виведення короткої інформації про локацію (спливаюча підказка відображає базову інформацію про об’єкт);
– фінальні події – відображення карти маршруту або сторінки локації з розгорнутою інформацією.

Структура EPC-моделі дає змогу чітко відокремити системні реакції від дій користувача та показати логіку взаємодії сайту з урахуванням можливих варіантів.

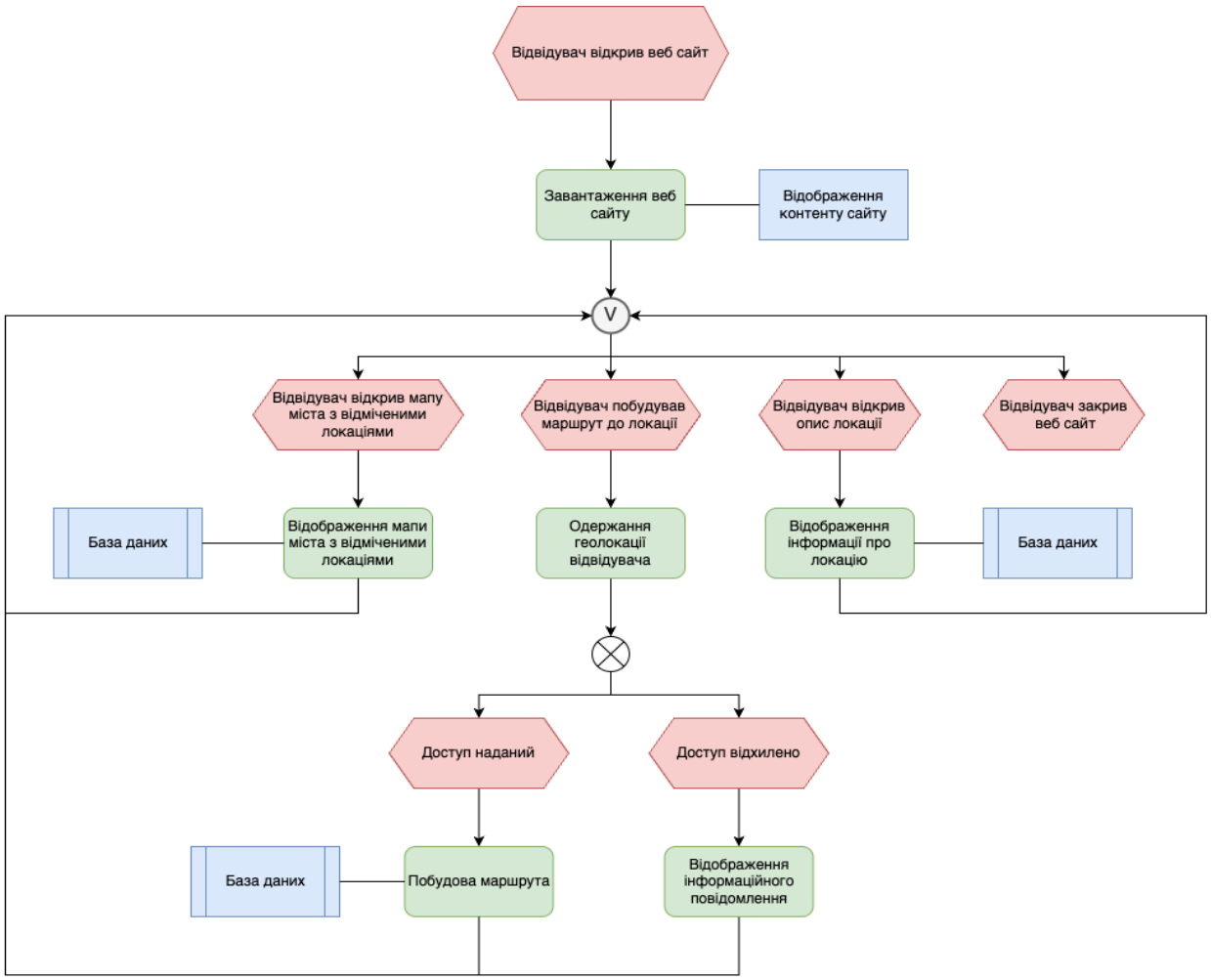


Рисунок 2.6 – EPC-діаграма логіки роботи сайту

Діаграма дозволяє описати функціональну структуру вебресурсу у вигляді послідовного процесу з чіткими подіями, функціями та умовами переходу, починаючи з входу і завершуючи отриманням та відображенням даних. Такий підхід є корисним не лише на етапі проєктування, але й для технічної документації, тестування та майбутнього масштабування функціональності. Побудована модель допомагає краще зрозуміти ланцюг дій користувача та взаємозв’язок між окремими блоками системи.

2.4 Проєктування інтерфейсу користувача (UI/UX)

Інтерфейс користувача є частиною будь-якого вебзастосунку, адже саме через нього здійснюється безпосередня взаємодія відвідувача із системою.

Якісний інтерфейс має бути інтуїтивно зрозумілим, доступним з різних пристроїв і привабливим з естетичної точки зору.

У рамках проєкту вебсайту для популяризації туристичних локацій особливу увагу приділено UI/UX-дизайну, з урахуванням принципів простоти, зручності та адаптивності.

Розробка інтерфейсу здійснювалася з урахуванням сучасних вимог до доступності, адаптивності та орієнтації на кінцевого користувача. Враховано типову поведінку відвідувачів туристичних ресурсів, які шукають зручний і швидкий спосіб отримання інформації про цікаві місця. Тому інтерфейс створено максимально легким для навігації, з чіткою структурою сторінки, акцентом на основний контент і відсутністю надлишкових візуальних елементів.

Дизайн реалізовано у відповідності до таких принципів:

– мінімалізм (на екрані відображається лише найнеобхідніша інформація);
– ієрархія візуальних елементів (важлива інформація має візуальний акцент);
– адаптивність (інтерфейс коректно відображається на ком’ютерах, планшетах та мобільних пристроях);
– єдність стилю (кольори, шрифти, іконки та блоки відповідають загальній стилістиці бренду);
– інтуїтивна навігація (головне меню, фільтри, кнопки мають зрозуміле розташування та логіку дій).

Дизайн розроблено з урахуванням UX-патернів, поширених у туристичних сервісах: вертикальне розташування контенту на мобільних пристроях, доступ до мапи одним натисканням, спрощене меню та мінімум кроків до виконання основної дії.

На рисунку 2.7 представлено макет сторінки сайту. Головна сторінка складається з наступних функціональних блоків:

– шапка сайту (містить логотип, навігаційне меню, перемикач мови);
– карусель з фотографіями міста;

– список рекомендованих локацій (виводиться в центрі сторінки, дозволяє переглядати локації та переходити до їхніх карток);
– футер сайту: контактна інформація, посилання на корисні ресурси.

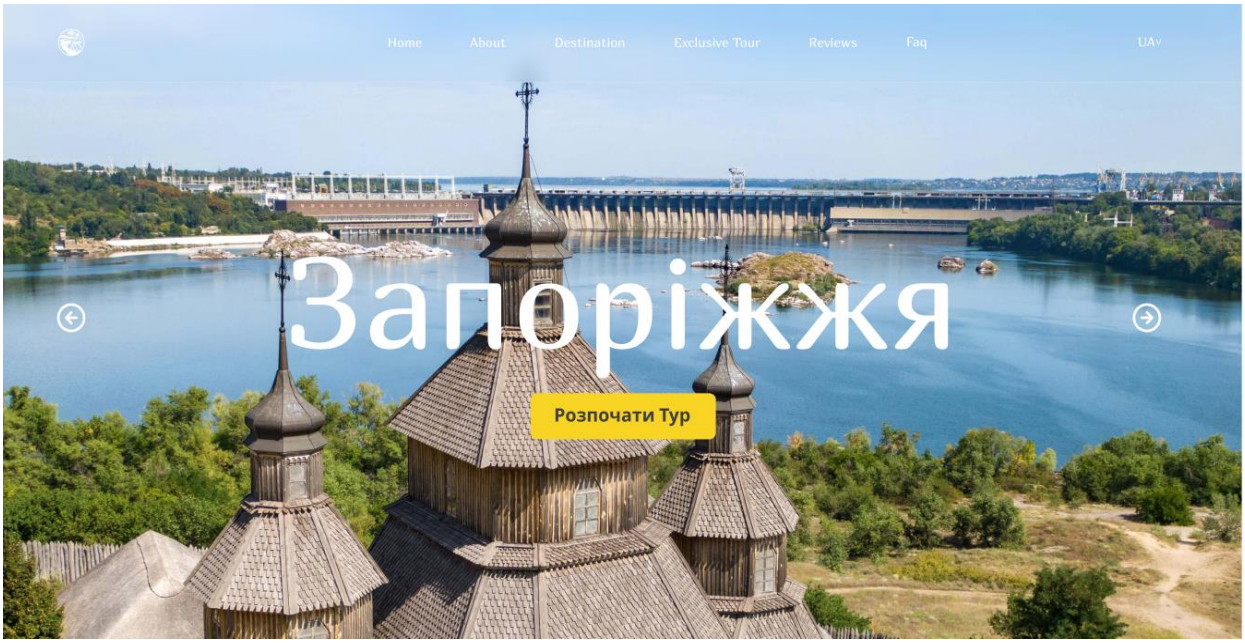


Рисунок 2.7 – Макет сторінки вебсайту для туристичних локацій

Макет розроблено з урахуванням адаптації до мобільних пристроїв (див. рис. 2.8). Для мобільної версії передбачено:

– вертикальне компонування елементів;
– навігаційне меню у вигляді бургер-іконки з випадаючим списком;
– великі кнопки дій, зручно розташовані в межах досяжності великого пальця;
– автоматичне масштабування шрифтів та блоків в залежності від ширини екрана.

Макет служить основою для реалізації клієнтської частини проєкту за допомогою технології Next.js, де кожен компонент інтерфейсу відображається як незалежний функціональний модуль.

Компонентний підхід дозволяє масштабувати інтерфейс, змінювати частини вмісту без порушення загальної структури та спрощує повторне використання блоків.



Рисунок 2.8 – Макет мобільної версії

Візуальна ідентичність сайту підсилюється оригінальним логотипом, який розроблено спеціально для цього проєкту. Логотип включає стилізований образ козака із сучасним натяком на історичну спадщину міста. Хвилі річки або лінії рельєфу – у нижній частині, вказують на річку Дніпро та природну географію міста. Він має мінімалістичну форму, чіткий силует та адаптується до темного і світлого фону. Логотип (див. рис. 2.9) використано у шапці сайту, та в мета-тегах як favicon.



Рисунок 2.9 – Логотип сайту

Таким чином, створений UI/UX-дизайн поєднує сучасні дизайнерські підходи з технічними рішеннями, що забезпечують зручну навігацію, естетичний вигляд і повну адаптивність на всіх типах пристроїв. Це створює привабливе середовище для користувачів сайту та сприяє ефективній популяризації туристичних локацій міста Запоріжжя.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБСАЙТУ

3.1 Розробка фронтенд-частини сайту

Фронтенд-частина сайту відповідає за візуальну складову та взаємодію користувача із системою. Її основною метою є відображення контенту, обробка дій користувача, динамічне оновлення сторінок та забезпечення інтуїтивно зрозумілого інтерфейсу.

У межах даного проєкту для реалізації клієнтської частини використано фреймворк Next.js, який поєднує переваги React з функціональністю серверного рендерингу (SSR) та статичної генерації (SSG). Такий вибір дозволив значно підвищити продуктивність сайту, покращити SEO та скоротити час завантаження сторінок. Для стилізації інтерфейсу використано Tailwind CSS, який забезпечує швидке створення адаптивного дизайну без необхідності створення великої кількості кастомних стилів.

Застосунок має компонентну структуру (див. рис. 3.1), типовою для React/Next.js застосунків.

Опис основних елементів структури:

– pages – відповідає за маршрути сайту (у Next.js кожен файл у цій теці автоматично стає сторінкою, що доступна за відповідною URL-адресою; назви директив і файлів можуть бути записанні в лапках, що робить роут динамічним);
– components – містить багаторазові функціональні компоненти інтерфейсу, які використовуються на різних сторінках;
– styles – тека зі стилями: глобальні налаштування та теми, створені за допомогою Tailwind CSS;
– public – публічно доступні ресурси (зображення, фавікон, файли robots.txt, тощо);
– types – тека зі TypeScript типами;

– lib – бібліотеки та утиліти, зокрема підключення до бази даних, функції для роботи з API, хелпери, тощо.

```
/pages
   /[city]
      /index.tsx  (головна сторінка, що відображає загальну інформацію, карту та список локацій)
      /map.tsx (сторінка з розгорнутою інтерактивною мапою для перегляду туристичних об'єктів)
      /[location].tsx (динамічна сторінка, що відображає повну інформацію про обрану локацію)
   /admin (панель адміністратора)
   /api (API-маршрути)
   …
/components
   /header.tsx (компонент шапки сайту з навігацією)
   /footer.tsx (нижній колонтитул з контактною інформацією та посиланнями)
   /map.tsx (компонент інтерактивної мапи з маркерами та логікою фільтрації)
   …
/styles
   /global.css (стилі Tailwind)
   /theme.css (кастомні змінні стилів)
/public
   /images (папка зі статичними ресурсами)
   /locales (папка зі перекладами)
   …
/types (папка зі глобальними типами)
/lib (папка зі утилітами для роботи з WebAPI та базою даних)
```

Рисунок 3.1 – Структура проєкту

За допомогою механізму динамічних маршрутів реалізовано сторінки для кожної окремої локації. Користувач може переглянути детальний опис, зображення, координати, контактну інформацію та прокласти маршрут за допомогою інтеграції з Google Maps.

Усі ключові елементи інтерфейсу винесені у функціональні компоненти, що дозволяє багаторазово їх використовувати, покращити масштабованість проєкту та спростити тестування. Tailwind CSS дає змогу створити адаптивну верстку, використовуючи утилітарні класи, що прискорює розробку і забезпечує сумісність між різними пристроями.

Для зручного керування контентом створено окремий інтерфейс адміністратора, доступний за маршрутом /admin. Цей функціональний модуль призначений для адміністраторів, які мають повноваження на додавання, редагування та видалення інформації про туристичні локації. Доступ до панелі захищено через JWT. Користувач проходить автентифікацію через форму логіну, після чого токен зберігається cookie. При кожному запиті до захищених API-роутів токен надсилається в заголовку запиту. Якщо токен недійсний або відсутній, користувача автоматично перенаправляє на сторінку входу.

Сторінка входу включає:

– поле введення логіна;
– поле введення пароля;
– кнопку входу;
– повідомлення про помилку при неправильних даних.

Після авторизації адміністратор потрапляє на головну сторінку панелі, яка складається з таких ключових блоків: перегляд, видалення локацій, створення локацій, редагування локацій.

Сторінка створення та редагування локацій використовують одну і ту саму форму, яка адаптується в залежності від режиму (створення або редагування).

Форма включає поля:

– назва локації (поле вводу);
– координати (поле вводу);
– завантаження зображення (передбачено drag-and-drop або вибір із файлової системи);
– короткий опис (поле вводу);
– категорія (випадаючий список, заповнений з API);
– статус (радіокнопка);
– тип (чекбокс);
– кнопка збереження/оновлення форми;
– повідомлення про помилку при неправильних даних.

При натисканні кнопки «Зберегти» відправляється POST запит до API. Перед збереженням відбувається валідація полів: порожні обов’язкові поля підсвічуються, а форма не надсилається. Після успішної операції виводиться повідомлення «Локацію збережено» та адміністратора перенаправляє назад до списку всіх локацій.

Головна сторінка є першим елементом взаємодії користувача з вебсайтом. Вона реалізована у файлі [city]/index.tsx, де [city] – параметр маршруту, що дозволяє у майбутньому масштабувати сайт для різних міст. Це забезпечує гнучкість і багаторазове використання шаблонів.

Головна сторінка виконує наступні функції:

– інформує про мету сайту: на сторінці розміщується вступний текст з описом туристичного потенціалу міста Запоріжжя;
– презентує місто як туристичну одиницю: зображення ілюструють ключові пам’ятки, архітектурні об’єкти, природні ресурси тощо;
– заклик до дії: на сторінці розміщені елементи які закликають користувачів перейти до мапи або до конкретної локації;
– список рекомендованих локацій: на сторінці показуються картки популярних туристичних об’єктів.

Візуально головна сторінка структурована блоками:

– навігаційна панель;
– карусель фотографій з панорамними фотографіями міста;
– секція фотографіями та описом міста;
– секція з популярними туристичними локацями;
– секція яка описує історичне минуле міста;
– нижній колонтитул.

Для SEO оптимізації використано компонент <Head> з відповідними мета-тегами (назва міста, ключові слова, опис).

Сторінка з повноцінною картою реалізована у файлі [city]/map.tsx. Вона призначена для користувачів, які хочуть бачити всі туристичні локації візуально на мапі з можливістю взаємодії в реальному часі.

Основна функціональність:

– завантаження інтерактивної карти Google Maps через Google Maps API;
– відображення локацій відповідно до отриманих;
– фільтрація маркерів за категоріями локацій;
– клік по маркеру відкриває коротку інформацію;
– побудова маршруту;
– адаптивність карти (розміри змінюються залежно від пристрою).

Сторінка складається з блоків:

– навігаційна панель;
– карта з елементами управління;
– нижній колонтитул.

Головна сторінка та мапа є ядром взаємодії користувача з системою. Головна – інформує, мотивує та направляє, тоді як сторінка мапи – інструмент активного користування. Обидві сторінки реалізовано з урахуванням адаптивності та продуктивності, що дозволяє забезпечити якісний користувацький досвід на будь-якому пристрої.

3.2 Розробка бекенд-частини сайту

Бекенд-частина сайту відповідає за обробку запитів від клієнтської частини, взаємодію з базою даних та реалізацію бізнес-логіки. У межах Next.js серверна логіка реалізується у вигляді API-роутів, розміщених у теці /pages/api/. Це дозволяє поєднати фронтенд та бекенд в одному проєкті без додаткового серверного середовища.

Для підключення до MongoDB використовується файл /lib/mongodb.ts, у якому визначено логіку підключення з використанням змінної середовища MONGODB_URI. Підключення реалізовано з кешуванням, що запобігає створенню нових з’єднань при кожному запиті до сервера. На рисунку 3.2 зображено приклад коду підключення до бази даних.

```
// Імпортування бібліотеки для підключення до MongoDB
import { MongoClient } from "mongodb";

// Перевіряємо наявність змінної середовища з URI для підключення до бази даних
if (!process.env.MONGODB_URI) {
  throw new Error('Invalid/Missing environment variable: "MONGODB_URI"');
}

// Отримуємо URI з файлу .env
const uri = process.env.MONGODB_URI;
const options = {};
// Оголошуємо змінну для зберігання екземпляра MongoClient
let client: MongoClient;

// Розширюємо глобальний об'єкт для зберігання MongoClient
let globalWithMongo = global as typeof globalThis & {
  _mongoClient?: MongoClient;
};
// Якщо клієнт ще не створено - створюємо і зберігаємо його у глобальній змінній
if (!globalWithMongo._mongoClient) {
  globalWithMongo._mongoClient = new MongoClient(uri, options);
}
// Присвоюємо глобального клієнта до змінної client
client = globalWithMongo._mongoClient;

// Експортуємо екземпляр MongoClient для повторного використання в інших модулях
// Це дозволяє використовувати одне з'єднання для всієї програми
export default client;
```

Рисунок 3.2 – Приклад коду підключення до бази даних

У бібліотеці Next.js API-роути можуть обробляти HTTP-запити різних типів: GET, POST, PUT, DELETE. Це дозволяє реалізувати повноцінний REST API без необхідності використання окремого бекенд-фреймворку (наприклад, Express). Кожен API-роут є окремою функцією обробки запитів (handler), що приймає об’єкти req (запит) і res (відповідь), аналогічно до Node.js/Express.

На рисунку 3.3 зображено приклад структури API.

```
/pages/api/
├── locations/
│   ├── index.ts   – обробка GET (усі локації), POST (створення локації)
│   └── [id].ts    – обробка GET, PUT, DELETE для конкретної локації
├── auth/
    └── login.ts   – авторизація адміністратора (POST запит)
```

Рисунок 3.3 – Структура API

Логіка авторизації використовує JWT для обмеження доступу до захищених API. У проміжному обробнику verifyToken зчитується токен з заголовка Authorization, далі виконується перевірка на справжність за допомогою jwt.verify(). У разі успішної перевірки користувач допускається до запиту. Цей механізм застосовується для:

– створення нових локацій;
– оновлення локацій;
– видалення локацій;
– перегляду панелі адміністратора.

Кожен API-роут містить базову обробку помилок:

– некоректний метод (405 Method Not Allowed);
– неавторизований запит (401 Unauthorized);
– помилки при записі/читанні з бази (500 Internal Server Error).

API-роути у Next.js реалізують повноцінну серверну логіку без необхідності розгортання окремого сервера. Така реалізація дає змогу централізовано керувати даними, забезпечити захист даних через JWT, масштабувати функціональність за потреби та інтегрувати інші сервіси. Структура є гнучкою, ефективною та зручною для розробки та управлінням проєктом.

3.3 Інтеграція інтерактивної карти туристичних локацій

Одним із ключових функціональних елементів сайту є інтерактивна карта, яка дозволяє користувачеві зручно переглядати розташування

туристичних локацій, фільтрувати їх за категоріями та прокладати маршрути до обраних точок. Для реалізації цієї функціональності використано Google Maps JavaScript API у поєднанні з бібліотекою react-google-maps, яка забезпечує зручну інтеграцію Google Maps із Next.js-застосунком.

Дані про локації отримуються через API-запит /api/locations. Кожна локація має координати широти та довготи, які використовуються для створення маркерів на мапі. На рисунку 3.4 зображено приклад інтеграції мапи в застосунок.

```
<APIProvider apiKey={API_KEY} libraries={["marker"]}>
  <Map {...settings}>
    <Markers pois={locations} />
  </Map>
</APIProvider>
```

Рисунок 3.4 – Приклад підключення Google Maps

При натисканні на маркер відкривається модальне вікно, у якому відображається коротка інформація про об’єкт (назва, категорія, кнопка заклику до дії).

Відвідувач сайту може відфільтрувати об’єкти за категоріями (музеї, природа, архітектура тощо). Панель фільтрації реалізована через окремий компонент, що оновлює стан відображення маркерів на мапі. Усі обчислення здійснюються на фронтенді після отримання повного списку локацій. Карта повністю адаптована під мобільні пристрої, а саме: автоматично масштабується відповідно до розміру вікна, підтримує сенсорні жести, інтерфейс маркерів оптимізовано під дотик, використовуються стандартні контролери Google Maps.

Для забезпечення візуальної узгодженості карти із загальним стилем сайту було реалізовано кастомну тему (див. рис. 3.5) для Google Maps, яка змінює колірну палітру елементів (дороги, водойми, ландшафт, будівлі) відповідно до дизайну інтерфейсу. Це дозволяє уникнути візуального дисонансу між картою та іншими компонентами сайту.

Мапа містить набір правил, які змінюють відображення карти:

– відтінки сірого або пастельні кольори замість стандартних насичених відтінків;
– приховані неактуальні об’єкти (наприклад, АЗС або школи);
– збільшена контрастність ліній вулиць для кращої видимості.

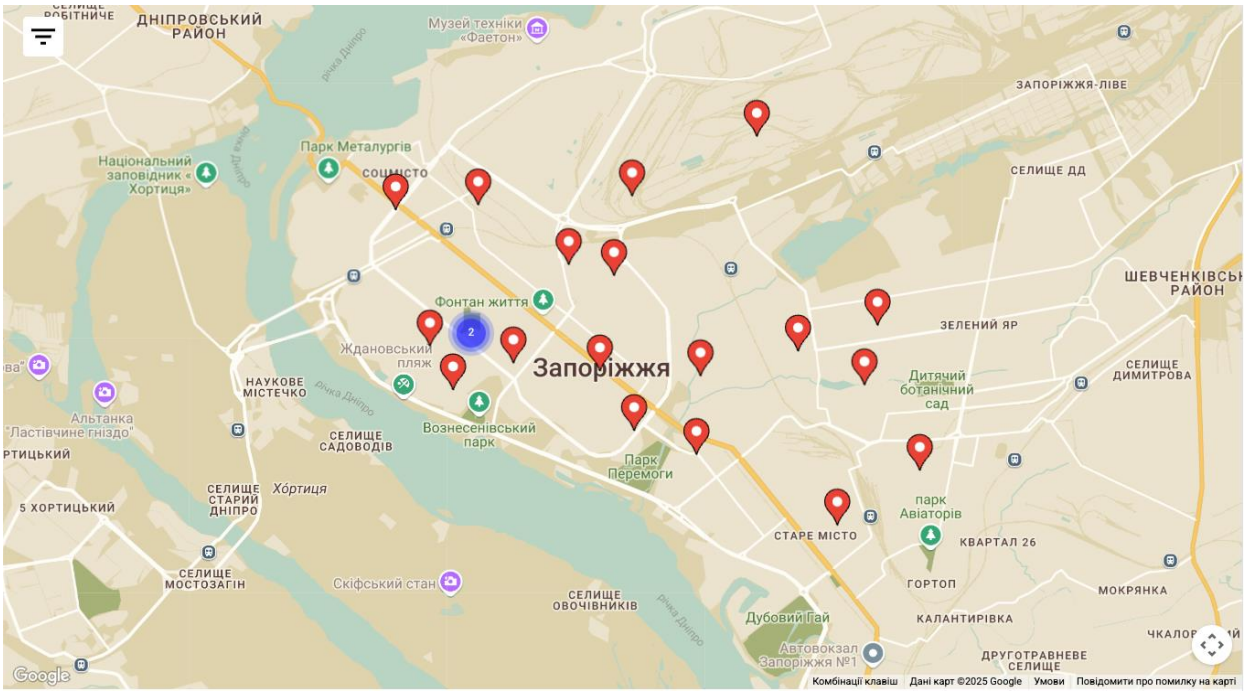


Рисунок 3.5 – Приклад стилізованої мапи

У результаті мапа не лише функціонально інтегрована, а й візуально гармонійна, що покращує користувацький досвід та відповідає сучасним трендам у дизайні туристичних сервісів.

У межах реалізації карти також було впроваджено кластеризацію маркерів, яка дозволяє підвищити зручність перегляду мапи при великій кількості локацій. Замість того, щоб одразу відображати всі маркери на карті (що може призвести до їх накладання та перевантаження візуального простору), маркери автоматично групуються в кластери відповідно до масштабу карти та щільності розташування об’єктів.

Застосування кластеризації має низку переваг:

– зменшує навантаження на карту, особливо на мобільних пристроях;
– покращує читабельність та візуальне сприйняття інформації;

– сприяє масштабованості, дозволяючи в майбутньому додавати сотні або тисячі локацій без втрати зручності.

Таким чином, кластеризація є важливим технічним та UX-покращенням, яке суттєво підвищує ефективність використання інтерактивної мапи, особливо на територіях з великою концентрацією об’єктів – наприклад, у центрі міста або туристичних зонах.

Для додаткового покращення візуального сприйняття туристичних локацій на інтерактивній мапі сайту, доцільно використовувати індивідуальні іконки для кожної категорії об’єктів. Це дозволяє користувачам швидко орієнтуватися на мапі та знаходити цікаві для них місця. На рисунку 3.6 зображено приклад стилізованих іконок для основних категорії, а саме:

– історичні пам’ятки;
– природні локації;
– музеї та галереї;
– архітектура;
– культура та розваги;
– спорт і активності;
– гастрономія;
– сувеніри та ремесла;
– для сімей з дітьми.



Рисунок 3.6 – Приклад стилізованих іконок для локацій

Інтеграція мапи значно покращує функціональність та зручність користування сайтом. Користувач може швидко зорієнтуватися в просторі, знайти найближчі туристичні об’єкти та прокласти маршрут до них. Реалізація карти в поєднанні з динамічним виводом локацій створює потужний інструмент для візуального представлення туристичної інфраструктури міста та сприяє її популяризації.

3.4 Тестування функціональності, продуктивності та безпеки сайту

Тестування програмного забезпечення є критично важливим етапом життєвого циклу розробки, оскільки дозволяє виявити помилки, забезпечити надійність роботи системи, перевірити її відповідність функціональним вимогам і гарантувати безпеку для кінцевого користувача. Особливо це актуально для вебзастосунків, що працюють у відкритому доступі, де кожен елемент інтерфейсу чи API має бути захищеним та стабільним. У межах проєкту було реалізовано три основні типи тестування: функціональне, продуктивне та безпекове. Кожен із них мав свою мету та відповідні підходи до реалізації.

Функціональне тестування передбачає перевірку основної логіки роботи сайту: завантаження сторінок, взаємодія з мапою, фільтрація об’єктів, відкриття карток локацій, побудова маршрутів, а також авторизація адміністратора і керування контентом через адмін-панель. Усі ці сценарії перевірялися вручну, шляхом проходження основних юзер-флоу. Для кожного модуля створювалися тест-кейси у структурованому вигляді з описом передумов, кроків і очікуваного результату.

Розглянемо приклади тест-кейсів.

Тест-кейс: «Перегляд інформації про локації».

Передумови: користувач знаходиться на сторінці з картою.

Кроки: клікнути на маркер локації.

Очікуваний результат: відкривається модальне вікно з описом локації.

Тест-кейс: «Користувач обирає фільтр на мапі».

Передумови: користувач знаходиться на сторінці з картою.

Кроки:

– відкрити фільтр категорій;

– користувач обирає категорію.

Очікуваний результат: на мапі відображаються лише локації з обраної категорії.

У таблиці 3.1 наведено матрицю тестування основних сценаріїв.

Таблиця 3.1 – Матриця тестування

	Пристрій	**ОС / Браузер**	**Мова**	**Результат**
Побудова маршруту до локації	MacBook Pro	MacOS / Chrome	Українська	Пройдено
Побудова маршруту до локації	Ноутбук	Windows / Chrome	Англійська	Пройдено
Відображення маркерів локації на мапі	MacBook Pro	MacOS / Safari	Українська	Пройдено
Відображення маркерів локації на мапі	iPhone 12	IOS / Safari	Українська	Пройдено
Вхід в адмін-панель	MacBook Pro	MacOS / Safari	Англійська	Пройдено
Створення локації	Google pixel 6	Androind / Chrome	Українська	Пройдено

Під час перевірки окремо фіксувалися не лише успішні сценарії, а й некоректні спроби – наприклад, спроба доступу до адмін-панелі без прав доступу, відправлення неповної форми тощо.

Тестування продуктивності проводилось із використанням інструмента Google Lighthouse, а також вручну в браузерах Chrome, Firefox, Safari. Було оцінено швидкість завантаження головної сторінки, інтерактивної мапи, час відповіді API-роутів, а також поведінку сайту на пристроях із різною продуктивністю. Окрема увага приділялась адаптивності – перевірці коректного відображення вмісту на мобільних, планшетах і десктопах. Високі показники Lighthouse за продуктивністю, доступністю та SEO свідчать про якісну реалізацію фронтенд-частини.

Безпекове тестування стосувалося перевірки автентифікації адміністратора, захисту API-роутів та обробки введених даних. На етапі реалізації було впроваджено авторизацію через JWT-токени, які передаються в заголовках HTTP-запитів і перевіряються перед виконанням CRUD-операцій. Паролі користувачів зберігаються у зашифрованому вигляді, а захищені маршрути недоступні без дійсного токена. Було перевірено, що запити без авторизації повертають відповідь з кодом 401 Unauthorized, а запити з невалідними даними – з кодом 400 Bad Request.

Результати тестування підтвердили, що реалізований вебсайт працює стабільно, відповідає заявленим функціональним вимогам, забезпечує захист критичних операцій і демонструє хорошу швидкодію на більшості сучасних пристроїв. Проведене тестування дозволило виявити і виправити дрібні логічні неточності, оптимізувати продуктивність інтерфейсу та впевнено готувати проєкт до використання у відкритому доступі.

3.5 Оптимізація та адаптивність сайту для різних пристроїв

У сучасних вебзастосунках особливе значення має оптимізація продуктивності та забезпечення коректного відображення вмісту на пристроях із різними розмірами екранів. З огляду на те, що значна частина користувачів отримує доступ до сайту саме з мобільних пристроїв, реалізація адаптивної верстки та продуктивного інтерфейсу була одним з пріоритетів при розробці проєкту.

У процесі реалізації було застосовано принципи mobile-first дизайну, що передбачає побудову інтерфейсу, оптимізованого насамперед для малих екранів із подальшою адаптацією до більших.

Для реалізації адаптивного інтерфейсу було використано CSS-фреймворк Tailwind CSS, який надає утилітарні класи для створення адаптивної розмітки. Це дозволило уникнути дублювання стилів та прискорити процес візуального налаштування під різні розміри екранів.

Під час верстки сторінок застосовувалися медіа-запити Tailwind (sm, md, lg, xl), що забезпечили:

– автоматичне розташування блоків у колонку або ряд, залежно від ширини екрана;
– зміну розмірів шрифтів, відступів і зображень для оптимального сприйняття інформації;
– адаптивну поведінку елементів інтерфейсу, таких як кнопки, форма пошуку, мапа.

Також було реалізовано:

– відкладене завантаження (lazy loading) зображень для покращення продуктивності;
– використання оптимізованого компонента <Image> з бібліотеки Next.js;
– динамічне центрування та масштабування карти в залежності від кількості локацій;
– обмеження використання надлишкових анімацій та скриптів на слабких пристроях.

Сайт було протестовано на наступних пристроях та розширеннях:

– смартфони (Android, iOS) – 360×640, 414×896;
– планшети – 768×1024;
– ноутбуки та десктопи – 1280×800, 1920×1080;
– емулятори Chrome DevTools та Safari Responsive Design Mode.

На рисунках 3.7 та 3.8 можна побачити, що всі елементи інтерфейсу відображаються коректно, навігація зручна, а інтерфейс лишається зрозумілим навіть на найменших екранах.

Крім того, були використані інструменти Google Lighthouse та Chrome Performance Audit для виявлення можливих проблем із адаптивністю, розміром шрифтів, контрастністю, масштабуванням тощо. Оцінки за показниками “Accessibility” та “Best Practices” становили понад 90%, що свідчить про відповідність загальноприйнятим вимогам UX/UI.

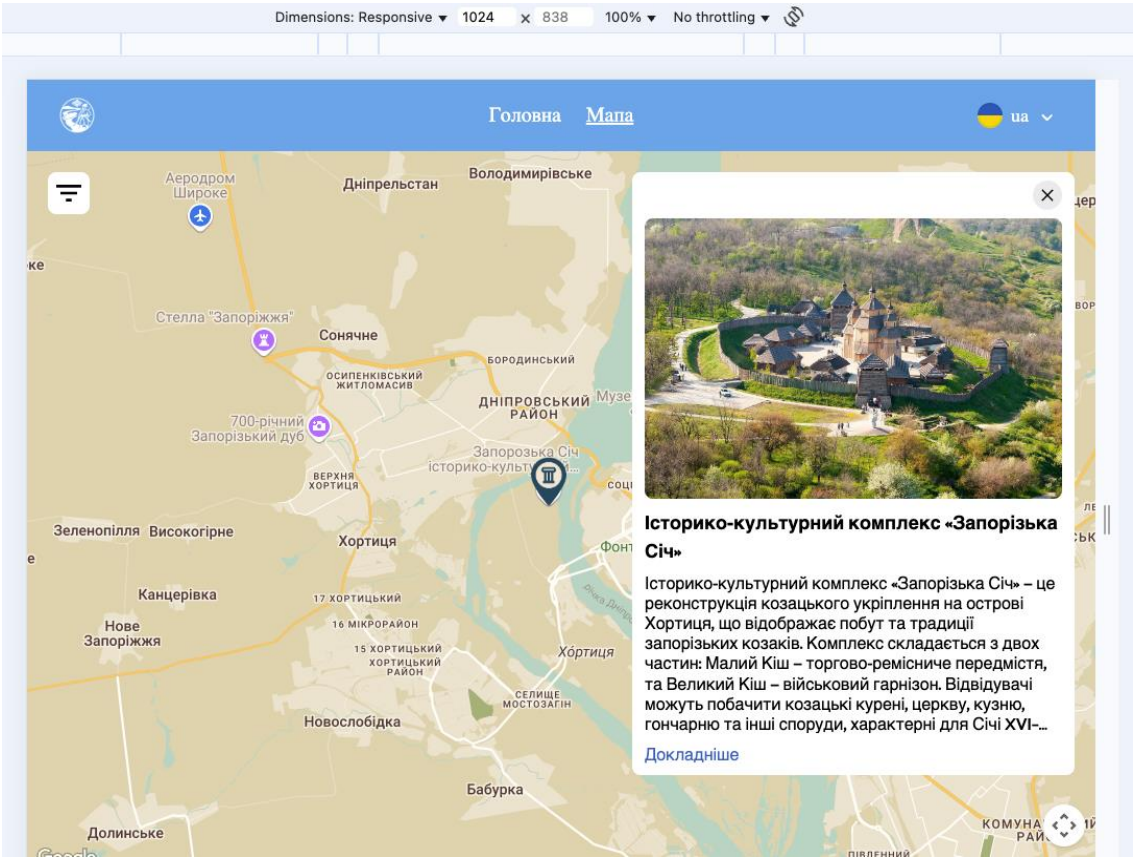


Рисунок 3.7 – Приклад адаптивного інтерфейсу сторінки мапи для ноутбука

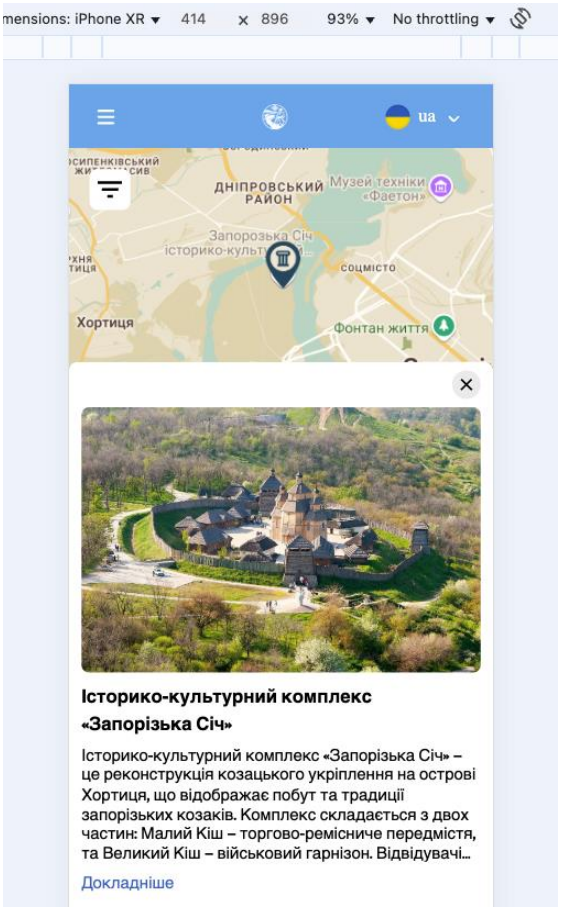


Рисунок 3.8 – Приклад адаптивного інтерфейсу сторінки мапи для телефону

У результаті оптимізації сайт демонструє стабільну роботу на широкому спектрі пристроїв, швидке завантаження основного контенту та високий рівень доступності. Це підвищує загальний користувацький досвід та робить ресурс привабливим і зручним для широкої аудиторії – як туристів, так і адміністраторів.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було реалізовано повнофункціональний вебсайт для популяризації туристичних локацій, що відповідає сучасним вимогам до якості, зручності користування, адаптивності та безпеки.

У теоретичній частині проведено аналіз сучасного стану розвитку туристичних вебресурсів, визначено основні потреби цільової аудиторії, сформульовано функціональні та нефункціональні вимоги до системи.

У процесі проєктування було обґрунтовано вибір технологій (Next.js, MongoDB, Tailwind CSS, Google Maps API), розроблено архітектуру сайту, включаючи діаграми варіантів використання, BPMN діаграму бізнес-процесів та EPC-схему логіки роботи сайту, структуру бази даних та інтерфейсу. На основі моделі було створено макети інтерфейсу, а також передбачено подальше масштабування проєкту для використання в інших містах України.

У практичній частині:

– реалізовано клієнтську частину сайту на базі Next.js з адаптивною верстkою;
– створено бекенд із REST API, авторизацією через JWT та підключенням до MongoDB;
– інтегровано інтерактивну карту з маркерами, фільтрацією та прокладанням маршрутів;
– розроблено адміністративну панель з повним циклом керування локаціями;
– проведено функціональне, продуктивне та безпекове тестування, що підтвердило стабільну та безпечну роботу сайту.

У результаті було досягнуто поставленої мети – створено зручний, інформативний та сучасний вебресурс, що може бути використаний як місцевими жителями, так і туристами для ознайомлення з цікавими локаціями міста Запоріжжя.

Проєкт має потенціал для подальшого розвитку, зокрема:

– впровадження особистих кабінетів користувачів;
– додавання рейтингових систем, коментарів, бронювання екскурсій;
– розширення географії (інтеграція інших міст України);
– інтеграція з мобільними додатками на базі React Native.

Отримані результати свідчать про практичну значущість та придатність розробленої системи до реального використання у туристичній галузі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 22.02.2025).

2. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 24.02.2025).

3. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 24.02.2025).

4. MongoDB Documentation. URL: https://www.mongodb.com/docs/ (дата звернення: 24.02.2025).

5. Google Maps Platform Documentation. URL: https://developers.google.com/maps/documentation (дата звернення: 26.02.2025).

6. JSON Web Tokens Introduction. URL: https://jwt.io/introduction/ (дата звернення: 02.03.2025).

7. Бізнес-моделювання BPMN. URL: http://www.economy.nayka.com.ua/?op=1&z=3514 (дата звернення: 04.03.2025).

8. What is EPC Diagram – Visual Paradigm. URL: https://online.visual-paradigm.com/knowledge/business-design-tools/what-is-epc-diagram/ (дата звернення: 05.03.2025).