МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ З РЕКОМЕНДАЦІЙ ЗАКЛАДІВ ТА ЛОКАЦІЙ У МІСТІ»**

Виконав: студент 4 курсу, групи 6.1212-пі-1
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
І.В. Шелельо
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, к.т.н. Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

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

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

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шелельо Іллі Владиславовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку з рекомендацій закладів та локацій у місті

керівник роботи Мильцев Олександр Михайлович, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 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 та технічного завдання.
2. Проєктування вебзастосунку.
3. Реалізація та тестування.

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 09.12.2025 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.	19.12.2025	
2.	Збір вихідних даних.	19.01.2026	
3.	Обробка методичних та теоретичних	16.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	20.03.2026	
5.	Розробка третього розділу.	18.05.2026	
6.	Оформлення та нормоконтроль	25.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2026	

Студент _____ І.В. Шелельо
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку з рекомендацій закладів та локацій у місті»: 44 с., 17 рис., 1 табл., 9 джерел.

ВЕБЗАСТОСУНОК, ГЕОДАНІ, ІНТЕРАКТИВНА МАПА, ЛОКАЦІЇ, NODE.JS, POSTGRESQL, REACT.

Об’єкт дослідження – процеси проєктування, розроблення та розгортання вебзастосунків для роботи з просторовими геоданими.

Предмет дослідження – технології побудови клієнт-серверної архітектури, методи збереження та обробки просторових об’єктів у СУБД PostgreSQL з розширенням PostGIS.

Мета роботи – скорочення часу користувача на пошук і обмін рекомендаціями щодо міських локацій та зменшення витрат апаратних ресурсів при рендерингу маркерів шляхом розробки вебзастосунку з інтерактивною картою.

Методи дослідження – аналіз існуючих ГІС-рішень, методи об’єктно-орієнтованого проєктування та моделювання, методи побудови REST API та модульного тестування програмних систем.

У роботі спроєктовано та реалізовано вебзастосунок клієнт-серверної архітектури. Серверну частину реалізовано на базі платформи Node.js з використанням СУБД PostgreSQL та просторового розширення PostGIS. Клієнтську частину розроблено за допомогою бібліотеки React та інструментів Mapbox GL, що забезпечує високу швидкість рендерингу інтерактивної мапи та відображення маркерів. Вебзастосунок успішно розгорнуто у хмарному середовищі Render.

SUMMARY

Bachelor’s qualifying paper “Development of a Web Application with Recommendations for Establishments and Locations in the City”: 44 pages, 17 figures, 1 table, 9 references.

GEODATA, INTERACTIVE MAP, LOCATIONS, NODE.JS, POSTGRESQL, REACT, WEB APPLICATION.

The object of the study is the processes of designing, developing, and deploying web applications for working with spatial geodata.

The subject of the research is client-server architecture design technologies, methods of storing and processing spatial objects in PostgreSQL DBMS with the PostGIS extension.

The aim of the study is to reduce user time for searching and sharing recommendations regarding city locations and to decrease hardware resource consumption during marker rendering by developing a web application with an interactive map.

The methods of research include analysis of existing GIS solutions, object-oriented design and modeling methods, methods of building REST APIs, and unit testing of software systems.

In this paper, a web application based on a client-server architecture has been designed and implemented. The server side is implemented using the Node.js platform, PostgreSQL DBMS, and the PostGIS spatial extension. The client side is developed using the React library and Mapbox GL tools, which ensure high rendering speed of the interactive map and marker visualization. The web application has been successfully deployed in the Render cloud environment.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз предметної області та технічне завдання 9
1.1 Огляд предметної області та існуючих картографічних сервісів 9
1.2 Обґрунтування вибору технологічного стеку 12
1.3 Формування вимог до вебзастосунку 14
2 Проєктування вебзастосунку 17
2.1 Проєктування архітектури та реляційної бази даних..... 17
2.2 Моделювання поведінки системи за допомогою UML..... 19
2.3 Проєктування інтерфейсу користувача та користувацького досвіду 23
3 Реалізація та тестування 26
3.1 Розробка серверної частини та налаштування безпеки 26
3.2 Розробка клієнтської частини та інтеграція інтерактивної мапи..... 28
3.3 Розгортання застосунку та тестування системи 31
3.4 Керівництво користувача 32
3.4.1 Головний екран застосунку, авторизація та інструменти фільтрації 33
3.4.2 Процес створення та редагування просторових об’єктів 34
3.4.3 Перегляд та управління існуючими локаціями 36
3.4.4 Система оцінювання та соціальної взаємодії користувачів .. 39
Висновки 42
Перелік посилань..... 44

ВСТУП

Актуальність теми: сучасний ритм життя у великих містах вимагає від людей швидкого доступу до релевантної інформації про навколишню інфраструктуру. Зростання попиту на внутрішній туризм, локальні розваги та гастрономічні відкриття зумовлює необхідність у зручних інструментах для пошуку цікавих місць, ресторанів, зон відпочинку чи пам’яток. Хоча ринок геоінформаційних сервісів перенасичений різноманітними платформами, користувачі все частіше стикаються з проблемою недостовірності відгуків, агресивним просуванням проплачених закладів та надміру складним інтерфейсом, який перешкоджає швидкій навігації.

Більшість популярних картографічних рішень є універсальними гігантами, які споживають значні ресурси пристроїв і не фокусуються на формуванні локальних спільнот для обміну досвідом. Тому створення, легкого та інтуїтивно зрозумілого вебзастосунку, який дозволяв би користувачам обмінюватися власними знахідками та рекомендаціями безпосередньо на карті, є затребуваним інженерним завданням. Використання передового стеку технологій (React [1] для побудови динамічного інтерфейсу, Node.js [2] для обробки серверних запитів та PostgreSQL [4] з PostGIS [5] для ефективного управління просторовими даними) забезпечить високу продуктивність системи, її кросплатформеність та зручність масштабування.

Мета дослідження – скорочення часу користувача на пошук і обмін рекомендаціями щодо міських локацій та зменшення витрат апаратних ресурсів при рендерингу маркерів шляхом розробки вебзастосунку з інтерактивною картою.

Для досягнення поставленої мети було сформульовано такі **завдання** дослідження:

– провести аналіз предметної області, існуючих ГІС-рішень на ринку та теоретичних основ роботи з просторовими даними;

- спроєктувати клієнт-серверну архітектуру вебзастосунку та обґрунтувати вибір технологічного стеку;
- розробити реляційну структуру бази даних та налаштувати просторове розширення для збереження і обробки об’єктів із географічними координатами;
- програмно реалізувати логіку взаємодії з інтерактивною картою, механізми автентифікації користувачів та збереження мультимедійних файлів;
- створити адаптивний користувацький інтерфейс для коректного відображення мапи та маркерів;
- провести тестування та розгорнути програмний продукт у хмарному середовищі.

Об’єкт дослідження – процеси проєктування, розроблення та розгортання вебзастосунків для роботи з просторовими геоданими.

Предмет дослідження – технології побудови клієнт-серверної архітектури, методи збереження та обробки просторових об’єктів у СУБД PostgreSQL з розширенням PostGIS.

Методи дослідження – для розв’язання поставлених завдань застосовувалися: методи аналізу вимог до програмного забезпечення, методи об’єктно-орієнтованого проєктування та моделювання, методи побудови та організації REST API, методи інтеграції геоінформаційних систем, а також методи тестування програмних систем.

Практичне значення одержаних результатів полягає у створенні повністю функціонального вебзастосунку, готового до використання цільовою аудиторією. Розроблений продукт дозволяє зручно маркувати нові місця на карті, залишати відгуки з фотографіями та формувати персоналізовану базу улюблених локацій, що значно покращує соціальну взаємодію та спрощує орієнтацію користувачів у міському просторі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Огляд предметної області та існуючих картографічних сервісів

Сучасний етап розвитку інформаційного суспільства нерозривно пов’язаний із процесами глобальної цифровізації та впровадженням концепцій «розумного міста». В умовах стрімкої урбанізації та постійного розширення міської інфраструктури, жителі мегаполісів та туристи щоденно стикаються з необхідністю швидкого пошуку релевантної інформації про навколишні об’єкти: заклади громадського харчування, культурні пам’ятки, зони відпочинку та інші локації. Традиційні методи пошуку інформації поступилися місцем геоінформаційним системам (ГІС) та сервісам, заснованим на визначенні місцезнаходження (Location-Based Services, LBS).

Предметною областю даного кваліфікаційного дослідження є процеси збору, збереження, обробки та візуалізації просторових геоданих у вебсередовищі з метою надання користувачам рекомендацій щодо міських локацій. Геоінформаційні вебзастосунки стали невід’ємною частиною повсякденного життя, оскільки вони дозволяють візуалізувати абстрактні дані у вигляді зручних та зрозумілих маркерів на інтерактивній мапі.

Однак, незважаючи на стрімкий розвиток індустрії та наявність великої кількості картографічних сервісів, ринок рекомендаційних гео-систем має низку суттєвих недоліків. Для детального аналізу предметної області доцільно розглянути існуючі аналоги, які наразі займають лідируючі позиції на ринку: Google Maps, TripAdvisor та Foursquare.

Google Maps – це найбільш розповсюджений картографічний сервіс у світі, який надає детальні карти, можливість побудови маршрутів та пошуку закладів. До його безперечних переваг належать гігантська база даних об’єктів, висока точність геолокації та інтеграція з іншими сервісами компанії Google. Проте, з точки зору користувача, який шукає персоналізовані

рекомендації локацій, Google Maps має ряд критичних недоліків. Сервіс є надзвичайно перевантаженим функціоналом, який не завжди потрібен (навігація для автомобілів, розклад руху транспорту, 3D-тури). Крім того, платформа сильно комерціалізована: результати пошуку закладів часто формуються на основі проплаченої реклами, що знижує об'єктивність видачі. Інтерфейс застосунку є важковаговим, а рендеринг великої кількості об'єктів на карті вимагає значних апаратних ресурсів від пристрою клієнта.

TripAdvisor – це одна з найбільших світових платформ для мандрівників, яка спеціалізується на відгуках про готелі, ресторани та розваги. Перевагою сервісу є потужна соціальна складова та детальна система рейтингування. Недоліками цієї платформи є фокус переважно на туристичному сегменті та комерційних закладах, що робить її незручною для пошуку локальних неформальних місць («hidden gems» – затишних парків, вуличних майданчиків, невеликих кав'ярень). Крім того, TripAdvisor неодноразово піддавався критиці через проблему фальшивих (накручених) відгуків, що створюються бот-мережами або самими власниками бізнесу для підвищення рейтингу. Картографічний інтерфейс платформи відіграє другорядну роль і є незручним для швидкої взаємодії.

Foursquare (City Guide) – сервіс локального пошуку, який раніше був піонером у сфері чекінів та персональних рекомендацій. Система пропонує користувачам підбірки місць на основі їхніх попередніх вподобань. Головний недолік сучасного Foursquare полягає у поділі функціоналу на декілька різних додатків та загальному ускладненні користувацького інтерфейсу (UI/UX). База даних локацій у багатьох невеликих містах часто залишається неактуальною, оскільки платформа втратила значну частину своєї активної аудиторії на користь більш сучасних рішень.

Аналіз існуючих аналогів дозволяє виявити ключову проблему предметної області. Сучасні картографічні сервіси еволюціонували в універсальні платформи для бізнесу та реклами. Комерційна складова витіснила первинну мету – швидкий, об'єктивний та соціально-орієнтований

обмін враженнями між реальними людьми. Користувачам доводиться витрачати багато часу на те, щоб відфільтрувати рекламні пропозиції від реальних відгуків, а інтерфейси цих платформ часто є занадто складними для виконання простої дії – збереження цікавої точки на карті та додавання фотографії до неї.

Виходячи з цього, виникає обґрунтована необхідність у розробці спеціалізованого вебзастосунку з рекомендацій міських локацій “RecoMaps”. Цей застосунок повинен фокусуватися саме на картографічній взаємодії (Map-centric design), мінімізувати кількість зайвого функціоналу та надавати користувачам можливість швидко додавати маркери, обмінюватися мультимедійними відгуками та взаємодіяти з просторовими даними у режимі реального часу. На відміну від глобальних універсальних рішень, розроблюваний продукт має бути легким, кросплатформовим та оптимізованим для роботи з просторовими координатами, що потребує застосування спеціалізованого стеку вебтехнологій та систем керування базами даних із підтримкою геометрії. Для більш наочного відображення переваг та недоліків розглянутих рішень, результати порівняльного аналізу існуючих картографічних платформ та розроблюваного вебзастосунку зведено у таблицю 1.1.

Таблиця 1.1 – Порівняльний аналіз картографічних сервісів та платформ локальних рекомендацій

Критерій порівняння	**Google Maps**	**TripAdvisor**	**Foursquare**	**RecoMaps (Розроблюваний вебзастосунок)**
1. Основне призначення	Універсальна автомобільна та пішохідна навігація	Туристичний сервіс, бронювання готелів та ресторанів	Локальний пошук (соціальна мережа у минулому)	Швидкий обмін персональними локальними рекомендаціями
2. Наявність реклами (монетизація)	Дуже висока (пріоритет проплачених закладів у пошуку)	Висока (агресивна реклама послуг та готелів)	Середня	Відсутня (повністю безкоштовний сервіс)
3. Зручність інтерфейсу (UI/UX)	Перевантажений (багато зайвих функцій, 3D-режими)	Складний (акцент на текстових блоках та рейтингах)	Застарілий (розділений на декілька різних додатків)	Сучасний мінімалізм (Map-centric дизайн, фокус на карті)

Продовження таблиці 1.1

Критерій порівняння	**Google Maps**	**TripAdvisor**	**Foursquare**	**RecoMaps (Розроблюваний вебзастосунок)**
4. Витрати апаратних ресурсів пристрою	Високі (вимагає значної оперативної пам’яті)	Середні	Середні	Низькі (оптимізований рендеринг за допомогою Mapbox GL)
5. Соціальна складова (додавання місць)	Низька (переважно сухі відгуки, складний процес створення об’єкта)	Середня (фокус на розгорнутих рецензіях)	Низька (втрачена активність аудиторії)	Висока (інтуїтивне та швидке маркування локацій з фото)
6. Кросплатформеність	Повноцінний кросплатформовий сервіс	Потребує встановлення важковагового мобільного додатка	Потребує встановлення мобільного додатка	Адаптивний вебзастосунок (повноцінно працює з будь-якого браузера)

Як свідчать дані таблиці 1.1, розроблюваний продукт займає унікальну нішу, пропонуючи користувачам вузькоспеціалізований, швидкий та позбавлений реклами інструмент для взаємодії з міським простором. На відміну від громіздких аналогів, він не перевантажує пристрій і дозволяє сфокусуватися виключно на карті та рекомендаціях.

1.2 Обґрунтування вибору технологічного стеку

Для успішної реалізації клієнт-серверного вебзастосунку, орієнтованого на роботу з інтерактивними картами та геоданими, критично важливим етапом є вибір оптимального стеку технологій. Враховуючи вимоги щодо високої продуктивності, кросплатформеності, можливості масштабування та необхідності швидкої обробки просторових координат, архітектуру системи було розділено на три логічні рівні: рівень бази даних, серверний рівень (Back-end) та клієнтський рівень (Front-end).

Рівень бази даних та обробки геоданих. В якості основної системи керування базами даних (СУБД) було обрано об’єктно-реляційну систему PostgreSQL [4]. Вибір обґрунтовується її високою надійністю, відповідністю

стандартам SQL та потужними механізмами забезпечення цілісності даних. Однак, традиційні реляційні бази даних не пристосовані для ефективного виконання просторових запитів (наприклад, пошуку об’єктів у заданому радіусі або визначення перетину полігонів). Для розв’язання цієї проблеми до PostgreSQL було підключено просторове розширення PostGIS [5].

PostGIS додає підтримку географічних об’єктів у СУБД, дозволяючи зберігати координати локацій у спеціальному типі даних GEOMETRY. Завдяки використанню просторових індексів (зокрема R-Tree над GiST), система здатна миттєво обчислювати відстані між користувачем та закладами, фільтрувати локації за географічними межами (Bounding Box) екрана пристрою та оптимізувати видачу маркерів, що є неможливим при зберіганні координат у вигляді звичайних числових полів широти та довготи.

Серверний рівень (Back-end). Для розробки серверної частини та реалізації REST API було обрано платформу Node.js [2] у поєднанні з мінімалістичним вебфреймворком Express.js [3]. Вибір середовища Node.js зумовлений його асинхронною, подієво-орієнтованою моделлю вводу-виводу (Event Loop), що робить його ідеальним рішенням для застосунків із великою кількістю одночасних мережевих запитів. Під час навігації картою клієнт постійно надсилає запити на отримання нових локацій, і Node.js здатний обробляти цей потік даних з мінімальними витратами оперативної пам’яті.

Додатковою перевагою є використання єдиної мови програмування (JavaScript) як на клієнті, так і на сервері, що стандартизує процес розробки (ізоморфна архітектура). Для забезпечення безпеки системи було обрано механізм JWT (JSON Web Tokens) [7], який дозволяє реалізувати stateless-автентифікацію без необхідності зберігання сесій на сервері. Обробка мультимедійних файлів (фотографій локацій) реалізується за допомогою проміжного програмного забезпечення Multer [9].

Клієнтський рівень (Front-end). Для побудови інтерфейсу користувача було обрано бібліотеку React. Її ключова перевага полягає у компонентному підході та використанні віртуального DOM (Document Object Model). React

дозволяє створити односторінковий застосунок (Single Page Application, SPA), що забезпечує миттєву реакцію інтерфейсу на дії користувача без повного перезавантаження сторінки браузера. Компонентна архітектура значно спрощує підтримку коду та дозволяє повторно використовувати елементи інтерфейсу (модальні вікна відгуків, картки закладів).

Картографічний інструментарій. Одним із найважливіших технологічних рішень став вибір бібліотеки для рендерингу інтерактивної карти. Замість традиційних растрових рішень (таких як Leaflet або базовий Google Maps API), було обрано платформу Mapbox GL JS [6]. Її головна відмінність полягає у використанні технології WebGL для векторного рендерингу картографічних тайлів безпосередньо за допомогою графічного процесора (GPU) клієнтського пристрою. Це забезпечує плавне масштабування карти зі швидкістю 60 кадрів на секунду та дозволяє без затримок відображати тисячі маркерів і кастомних стилів одночасно.

Таким чином, застосування стеку технологій у складі PostgreSQL + PostGIS, Node.js та React з інтеграцією Mapbox GL повністю відповідає сучасним вимогам до розробки геоінформаційних вебзастосунків і гарантує створення надійного, масштабованого та високопродуктивного програмного продукту.

1.3 Формування вимог до вебзастосунку

Визначення чітких та вичерпних вимог є критично важливим етапом життєвого циклу розробки програмного забезпечення, оскільки саме вони формують базис для проєктування архітектури, написання коду та подальшого тестування готового продукту. Відповідно до специфіки предметної області та визначеної мети дослідження, вимоги до розроблюваного вебзастосунку доцільно розділити на дві основні категорії: функціональні та нефункціональні.

Функціональні вимоги визначають базову поведінку системи та перелік дій, які користувач може виконувати під час взаємодії з додатком. Розроблювана система повинна забезпечувати наступні **функціональні можливості**:

- система має надавати неавторизованим користувачам (гостям) можливість переглядати інтерактивну мапу та ознайомлюватися із загальним пулом маркерів локацій без можливості їх редагування або створення нових;
- система має підтримувати механізм реєстрації нових користувачів та їх подальшої авторизації з використанням логіну (електронної пошти) та пароля;
- авторизований користувач повинен мати можливість створювати нові локації на карті шляхом вибору відповідних географічних координат (широти та довготи);
- система повинна забезпечувати можливість додавання детального опису до локації, включаючи текстовий відгук, рейтингову оцінку та завантаження мультимедійних файлів (фотографій);
- система має надавати користувачам функціонал для редагування та видалення власних записів та маркерів;
- система повинна дозволяти адміністраторам (за наявності відповідних прав доступу) модерувати контент та керувати обліковими записами користувачів.

Нефункціональні вимоги описують атрибути якості системи, такі як безпека, продуктивність, масштабованість та зручність використання. Розглянемо наступні **нефункціональні вимоги** для розроблюваного вебзастосунку.

Вимоги до інтерфейсу користувача (Usability): застосунок повинен мати адаптивний вебдизайн (Responsive Web Design), що забезпечує коректне відображення карти та елементів керування на пристроях із різною роздільною здатністю екрана (десктопні монітори, планшети, смартфони).

Підтримка браузерів: клієнтська частина повинна стабільно функціонувати у всіх сучасних веббраузерах останніх версій (Google Chrome, Mozilla Firefox, Safari, Edge).

Вимоги до продуктивності (Performance): час завантаження базового інтерфейсу та ініціалізації карти не повинен перевищувати 2 секунди. Рендеринг маркерів на карті має здійснюватися з використанням апаратного прискорення (WebGL) для запобігання затримкам при масштабуванні або переміщенні камери. Час обробки запитів сервером до бази даних має бути мінімізований за рахунок використання просторових індексів.

Вимоги до безпеки (Security): паролі користувачів у базі даних повинні зберігатися виключно у хешованому вигляді (з використанням алгоритму bcrypt). API серверу має бути захищене від несанкціонованого доступу за допомогою механізму JSON Web Tokens (JWT). Важливою вимогою є передача всіх мережевих запитів між клієнтом та сервером виключно через зашифрований протокол HTTPS (SSL/TLS) для запобігання перехопленню даних та блокуванню запитів політиками безпеки сучасних браузерів.

Сформовані функціональні та нефункціональні вимоги є достатніми для початку етапу проєктування інформаційної архітектури вебзастосунку, створення логічної структури бази даних та розробки інтерфейсів взаємодії, що буде детально розглянуто у наступному розділі.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Проєктування архітектури та реляційної бази даних

На етапі проєктування системи здійснюється перехід від сформованих функціональних та нефункціональних вимог до безпосереднього створення технічної структури програмного продукту. Враховуючи специфіку обробки просторових даних та необхідність забезпечення високої швидкодії, основою розроблюваного вебзастосунку стала класична багаторівнева клієнт-серверна архітектура.

Запропонована архітектурна модель розділяє систему на три незалежні рівні.

Рівень представлення (Presentation Layer): клієнтський вебзастосунок, реалізований за допомогою бібліотеки React. Цей рівень відповідає за рендеринг інтерфейсу користувача, відображення інтерактивної мапи (Mapbox GL) та відправку асинхронних HTTP-запитів до сервера.

Рівень бізнес-логіки (Application Layer): серверний додаток, побудований на базі середовища Node.js та фреймворку Express. Цей рівень реалізує архітектурний стиль REST API. Сервер є «безстановим» (stateless) і обробляє вхідні запити, виконує перевірку прав доступу (JWT-валідацію), обробляє бізнес-правила та формує відповіді у форматі JSON.

Рівень доступу до даних (Data Access Layer): система керування базами даних PostgreSQL із розширенням PostGIS, а також файлове сховище для збереження завантажених користувачами медіафайлів.

Такий розподіл обов’язків забезпечує слабку зв’язність (loose coupling) компонентів системи, що дозволяє незалежно масштабувати серверну та клієнтську частини в майбутньому.

Важливим етапом проєктування архітектури є розробка логічної та фізичної структури реляційної бази даних. Для забезпечення цілісності даних,

уникнення дублювання та оптимізації запитів схему бази даних було нормалізовано до третьої нормальної форми (3НФ).

Розглянемо **ключові сутності** (таблиці) та зв'язки між ними, що лежать основі інформаційної моделі.

Користувачі (users): таблиця для збереження облікових записів. Містить унікальний ідентифікатор, ім'я користувача (username), електронну пошту, хеш пароля та роль (звичайний користувач або адміністратор).

Локації (locations): центральна сутність системи. Окрім стандартних текстових полів (назва, опис) та масиву посилань на медіафайли, таблиця містить поле geom типу GEOMETRY(Point, 4326). Цей просторовий тип даних дозволяє базі даних розпізнавати координати локації на сферичній поверхні Землі (в системі координат WGS 84) для оптимізованого картографічного пошуку. Таблиця пов'язана з таблицею users зв'язком «один-до-багатьох» (кожен запис має creator_id).

Відгуки (reviews) та *Відповіді* (review_replies): сутності для реалізації соціальної взаємодії. Таблиця reviews містить оцінку (рейтинг) та текстовий коментар, пов'язуючи конкретну локацію (location_id) з користувачем (user_id). Для забезпечення можливості дискусій спроєктовано підпорядковану таблицю відповідей на відгуки (review_replies).

Теги (tags) та *Зв'язки локацій з тегами* (location_tags): для гнучкої фільтрації закладів реалізовано відношення «багато-до-багатьох». Таблиця tags містить унікальні назви категорій, а проміжна таблиця location_tags забезпечує зв'язок між конкретною локацією та її категоріями.

Історія пошуку (search_history): таблиця для збереження текстових пошукових запитів (search_query) та кешування останніх переглянутих користувачем локацій (location_id), що дозволяє персоналізувати видачу та покращує користувацький досвід (UX).

Спроєктована структура бази даних є оптимальною для вирішення поставлених завдань. Інтеграція просторових полів PostGIS та налаштування зв'язків за допомогою зовнішніх ключів гарантують консистентність даних та

забезпечують швидкий доступ до геоінформації на вимогу клієнтського застосунку.

Візуальну модель спроєктованої інформаційної архітектури, що відображає набір сутностей бази даних, їхні атрибути, ключі (первинні та зовнішні) та кардинальність відношень між ними, наведено у вигляді ER-діаграми на рисунку 2.1.

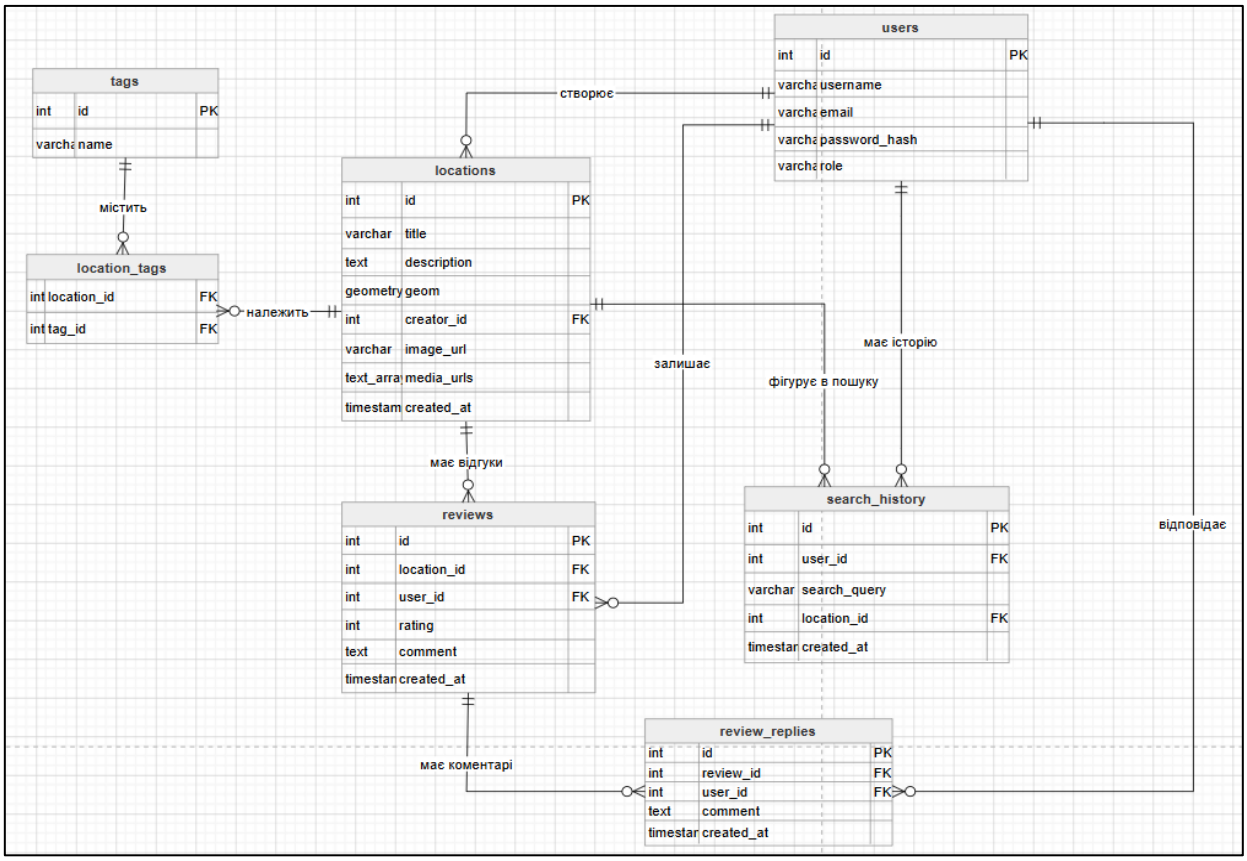


Рисунок 2.1 – ER-діаграма структури бази даних

2.2 Моделювання поведінки системи за допомогою UML

Для всебічного опису архітектури, статичної структури та динамічної поведінки розроблюваного вебзастосунку було застосовано уніфіковану мову моделювання (UML). Використання стандартизованих графічних нотацій дозволяє наочно продемонструвати взаємодію компонентів системи та деталізувати функціональні вимоги на етапі проєктування.

Першим етапом моделювання є визначення меж системи та можливих сценаріїв взаємодії користувачів із програмним продуктом RecoMaps. Для цього було побудовано діаграму варіантів використання (див. рис. 2.2).

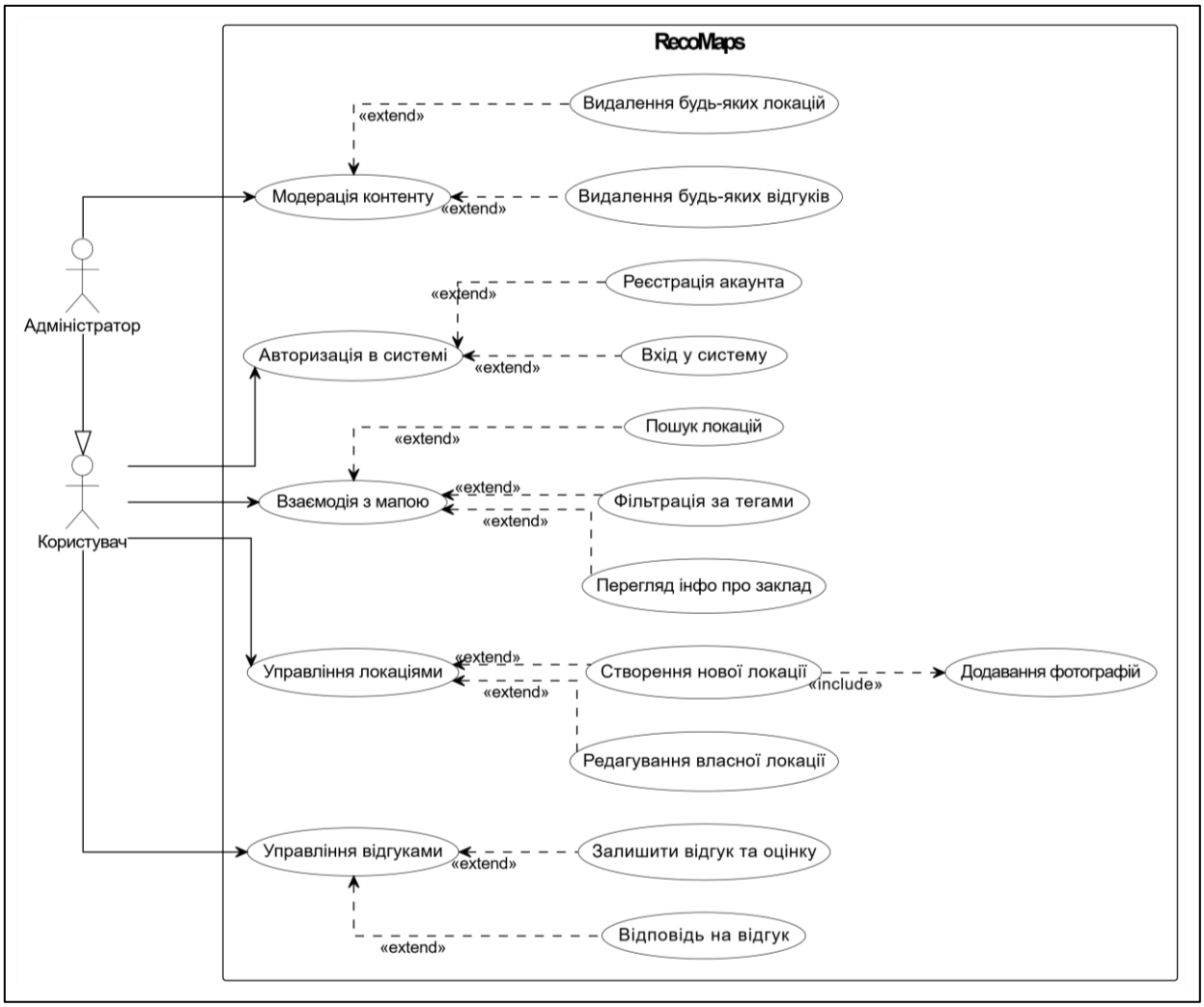


Рисунок 2.2 – Діаграма варіантів використання вебзастосунку

У системі виділено двох основних акторів: «Користувач» (авторизована особа) та «Адміністратор» (особа з розширеними правами доступу). Користувач ініціює базові прецеденти: авторизація в системі (що розширюється реєстрацією та входом), взаємодія з мапою (перегляд, фільтрація, пошук), управління локаціями (створення, редагування, додавання фото) та управління відгуками. Адміністратор, своєю чергою, успадковує всі можливості базового користувача, але додатково має доступ до прецедентів модерації контенту (видалення будь-яких локацій та відгуків). Використання

зв’язків типу «extend» та «include» дозволило деталізувати та структурувати складні процеси системи.

Для опису структури системи та взаємозв’язків між її програмними компонентами розроблено архітектурну діаграму класів (див. рис. 2.3).

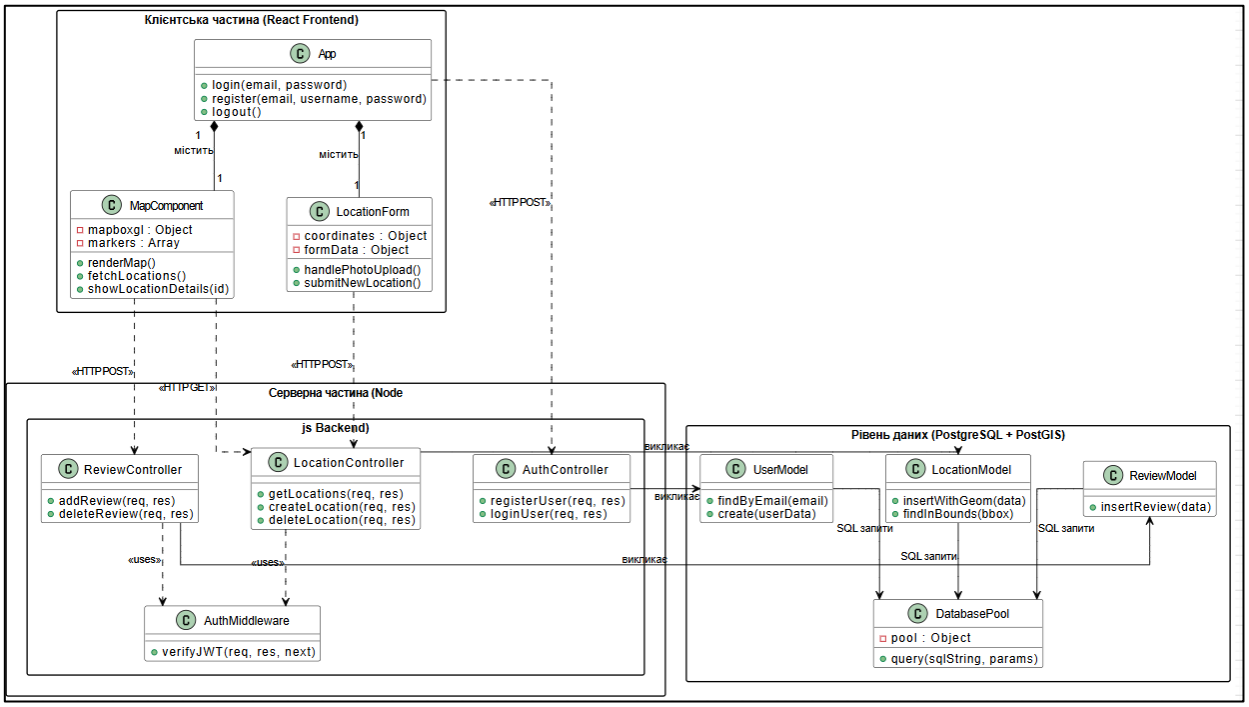


Рисунок 2.3 – Діаграма класів архітектури застосунку

На відміну від класичної моделі сутностей предметної області, дана діаграма відображає реальну багаторівневу архітектуру розроблюваного вебзастосунку та розділяє систему на три логічні пакети: клієнтську частину (React Frontend), серверну частину (Node.js Backend) та рівень доступу до даних (PostgreSQL).

Діаграма відображає ключові класи кожного рівня: UI-компоненти (MapComponent, LocationForm), контролери обробки бізнес-логіки (LocationController, ReviewController) та моделі взаємодії з базою даних (LocationModel, UserModel). Кожен клас інкапсулює власні атрибути та визначає публічні методи доступу. Крім типів відношень між самими класами, діаграма додатково демонструє динаміку міжсистемної взаємодії – маршрутизацію HTTP-запитів від клієнта до сервера та виконання SQL-

запитів до бази даних, що дозволяє повною мірою оцінити архітектурну складність застосунку.

Для розуміння динаміки роботи системи та послідовності обміну повідомленнями між клієнтською частиною, сервером та базою даних у часі, побудовано діаграму послідовності (див. рис. 2.4).

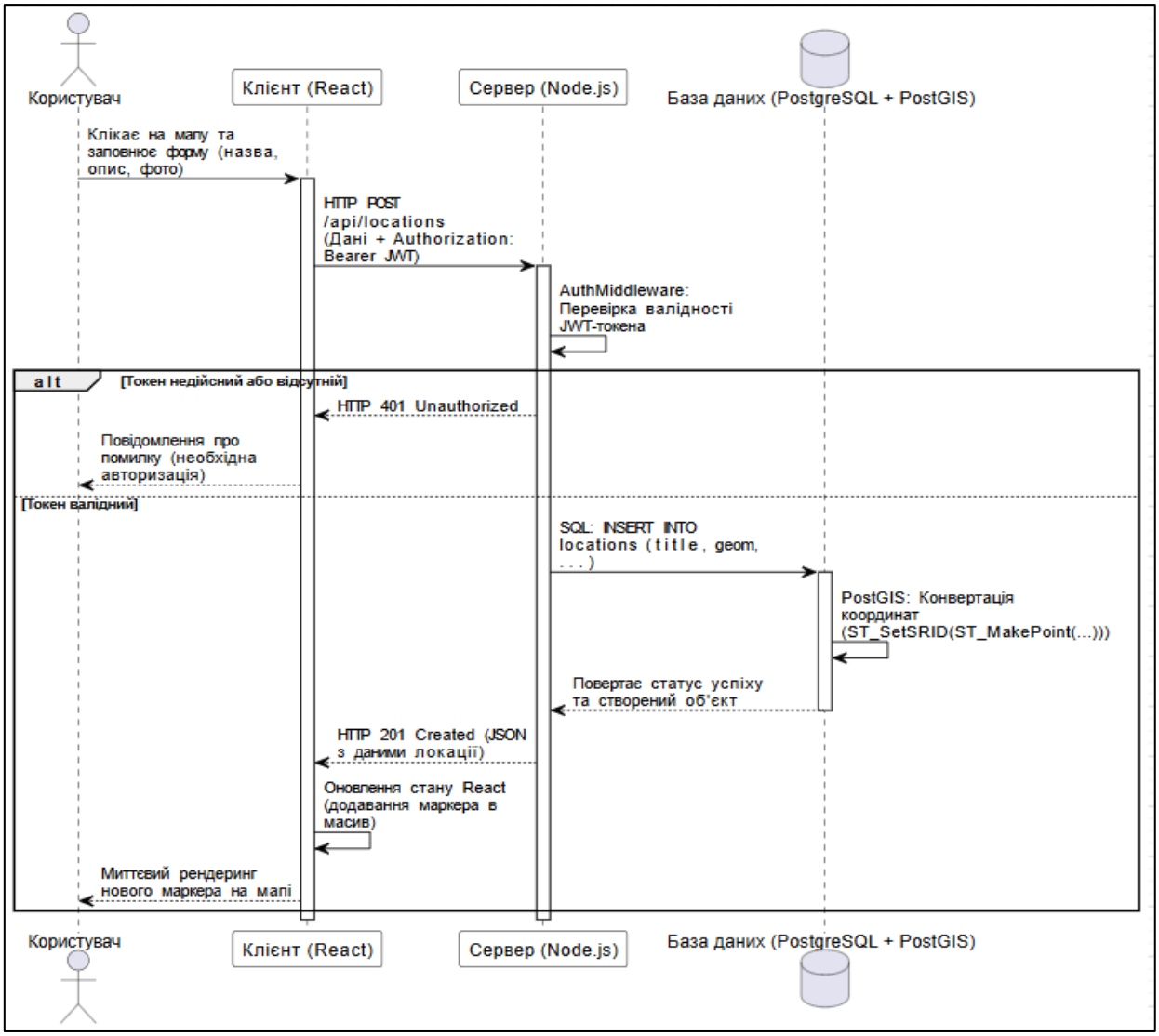


Рисунок 2.4 – Діаграма послідовності процесу створення нової локації

В якості основного сценарію для моделювання обрано найбільш складний та критично важливий прецедент – «Створення нової локації на карті».

Процес ініціюється користувачем через клієнтський застосунок (React), який надсилає HTTP POST запит із даними локації та JWT-токеном

авторизації. Сервер (Node.js) перехоплює запит і виконує middleware-перевірку токена. Як показано на діаграмі за допомогою альтернативного блоку (alt), у разі недійсного токена процес переривається помилкою 401 Unauthorized. У разі успішної валідації сервер звертається до СУБД PostgreSQL. База даних за допомогою розширення PostGIS конвертує координати у просторовий об’єкт, зберігає запис та повертає статус виконання. На фінальному етапі сервер відправляє клієнту відповідь 201 Created, після чого клієнт оновлює стан і миттєво рендерить новий маркер на мапі.

2.3 Проєктування інтерфейсу користувача та користувацького досвіду

Етап проєктування інтерфейсу користувача (UI) та користувацького досвіду (UX) є одним із найважливіших для геоінформаційних застосунків, оскільки саме він визначає зручність взаємодії з просторовими даними та ефективність пошуку цільових об’єктів.

Основою візуальної архітектури розроблюваного вебзастосунку з рекомендацій закладів та локацій у місті стала концепція «Map-centric» (орієнтація на карту). Згідно з цим підходом, інтерактивна мапа не є ізольованим віджетом на сторінці, а виступає базовим фоном та головним робочим простором (Viewport). Усі інші елементи керування (навігаційне меню, фільтри, форми) спроєктовано у вигляді плаваючих панелей (Floating Panels) та модальних вікон, що накладаються поверх картографічної основи. Це дозволяє максимізувати корисну площу екрана для огляду міської інфраструктури.

На етапі логічного проєктування інтерфейсу було розроблено карту навігації (User Flow). Маршрути переміщення користувача між усіма екранами системи зображено на діаграмі навігації (див. рис. 2.5).

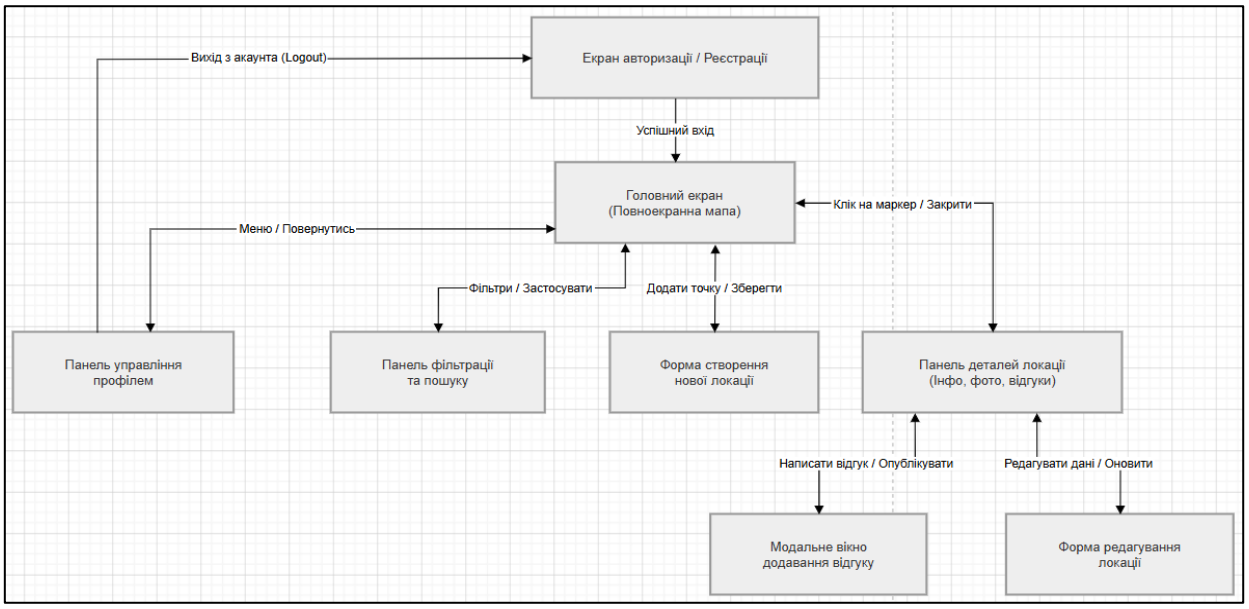


Рисунок 2.5 – Діаграма навігації

Спроєктована архітектура покриває весь основний функціонал застосунку: від базових переходів між екраном авторизації, картою та панеллю профілю, до інструментів фільтрації просторових даних. Детально опрацьовано сценарії взаємодії з конкретним закладом: з інформаційної панелі локації користувач має змогу перейти до модального вікна додавання відгуку, або викликати форму редагування даних (якщо він є автором запису).

Для забезпечення цілісності сприйняття інформації було сформовано базові **принципи дизайн-системи застосунку**.

Колірна палітра: спроєктовано контрастну схему, де інтерфейсні елементи (панелі, картки) мають нейтральні відтінки (білий, світло-сірий) із легкими тінями (Drop Shadow) для візуального відокремлення від строкатої карти. Самі маркери локацій та ключові кнопки цільових дій виділяються яскравими акцентними кольорами для швидкого розпізнавання.

Типографіка: використано сучасні шрифти без зарубок (Sans-serif) із високим рівнем читабельності. Розміри шрифтів спроєктовано за принципом ієрархії – великі заголовки для назв закладів, середні для описів, дрібні для метаданих (дати, категорії).

Зворотний зв'язок: спроєктовано систему мікроінтеракцій. Кожна дія користувача (наведення на маркер, успішне створення локації, помилка

валідації) супроводжується візуальним відгуком системи, що підвищує загальний рівень ергономіки.

Окрему увагу під час проєктування інтерфейсу було приділено процесу наповнення бази даних користувацьким контентом. Інтерфейс форми створення нової локації враховує необхідність точного збору геоданих. Замість ручного введення широти та довготи, UX-сценарій передбачає інтуїтивну взаємодію: користувач обирає точку безпосередньо на карті за допомогою курсора, після чого координати автоматично фіксуються в системі. Форма заповнення текстових даних (назви, опису, тегів) та завантаження фотографій з’являється як логічне продовження цієї дії, не перекриваючи обраний маркер.

Усі елементи інтерфейсу розроблено з урахуванням принципів адаптивного вебдизайну (Responsive Web Design). Завдяки використанню гнучких сіток, інтерфейс застосунку коректно масштабується на мобільних пристроях: бічні панелі автоматично ховаються, викликаючись за допомогою меню, а інформаційні вікна (Popups) перетворюються на нижні панелі (Bottom Sheets), оптимізовані для керування великим пальцем однієї руки.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Розробка серверної частини та налаштування безпеки

Формування серверної інфраструктури розроблюваного вебзастосунку базується на використанні програмної платформи Node.js. В якості основного інструменту для маршрутизації та обробки HTTP-запитів застосовано каркас (фреймворк) Express, що дозволило побудувати гнучкий та масштабований REST API.

Специфіка геоінформаційного сервісу передбачає активну роботу з користувацьким контентом, зокрема фотографіями локацій. Тому, окрім базового налаштування кросдоменних політик (CORS) та парсингу JSON, до конвеєра обробки запитів інтегровано модуль multer. Він відповідає за розбір multipart/form-data запитів та автоматичне збереження завантажених файлів у локальну директорію uploads із присвоєнням їм унікальних часових міток.

Організацію взаємодії з реляційною базою даних PostgreSQL реалізовано через драйвер node-postgres. Для мінімізації накладних витрат на встановлення нових з’єднань при кожному зверненні клієнта, використано механізм пулу (Connection Pool). Оскільки архітектура проєкту передбачає розгортання на хмарному хостингу, конфігурація підключення в обов’язковому порядку включає параметри SSL-шифрування. Усі критично важливі дані для підключення ізольовано у файлі змінних оточення .env. Фрагмент коду ініціалізації бази даних наведено на рисунку 3.1.

Під час реєстрації сервер генерує унікальну сіль (salt) та застосовує алгоритм хешування Blowfish. Процедура входу в систему зводиться до пошуку користувача за електронною поштою та криптографічного порівняння отриманого пароля із хешем, що лежить у базі.

Управління сеансами та розмежування прав доступу побудовано за принципом відсутності стану (stateless) за допомогою технології JSON Web

Token (JWT). При успішній автентифікації клієнт отримує криптографічно підписаний токен, у корисне навантаження (payload) якого зашито його ідентифікатор (userId) та поточну роль у системі (role). Реалізацію логіки авторизації та видачі цифрового підпису продемонстровано у лістингу 3.2.

```
const pool = new Pool({
   user: process.env.DB_USER,
   password: process.env.DB_PASSWORD,
   host: process.env.DB_HOST,
   port: process.env.DB_PORT,
   database: process.env.DB_NAME,
   ssl: { rejectUnauthorized: false }
});
```

Рисунок 3.1 – Налаштування підключення до СУБД PostgreSQL через пул з’єднань

```
app.post('/api/auth/login', async (req, res) => {
// Генерація JWT-токена після успішної валідації
const token = jwt.sign(
{ userId: user.rows[0].id, role: user.rows[0].role },
JWT_SECRET,
{ expiresIn: '24h' }
);
 res.json({ token, user: { id: user.rows[0].id, role: user.rows[0].role } }); }
);
```

Рисунок 3.2 – Обробник авторизації та генерації JWT-токена

Більшість функцій застосунку, таких як створення закладів або модерація відгуків, є закритими для анонімних відвідувачів. Для фільтрації вхідного трафіку на такі маршрути було створено власну функцію-перехоплювач (middleware) під назвою authMiddleware. Алгоритм її роботи передбачає екстракцію токена із заголовка HTTP-запиту (за схемою Bearer) та його синхронну верифікацію на основі секретного ключа сервера.

Якщо криптографічний підпис збігається і термін дії токена не вичерпано, розшифровані атрибути користувача делегуються до об’єкта запиту для подальшого використання у бізнес-логіці. Будь-які маніпуляції з токеном на стороні клієнта призводять до помилки верифікації та негайного блокування запиту (статус 401).

Код розробленого модуля автентифікації наведено на рисунку 3.3.

```
const authMiddleware = (req, res, next) => {
   const token = req.header('Authorization')?.split(' ')[1];
   if (!token) return res.status(401).json({ message: 'Немає токена' });

   try {
      const decoded = jwt.verify(token, JWT_SECRET);
      req.userId = decoded.userId;
      req.role = decoded.role;
      next(); // Передача управління наступному контролеру
   } catch (err) {
      res.status(401).json({ message: 'Токен недійсний' });
   }
};
```

Рисунок 3.3 – Модуль проміжного контролю для перевірки JWT-токена

3.2 Розробка клієнтської частини та інтеграція інтерактивної мапи

Практичну реалізацію клієнтської частини застосунку виконано на базі бібліотеки React, що дозволило створити швидкий односторінковий вебзастосунок (SPA). Основою архітектури інтерфейсу є компонентний підхід, який забезпечує інкапсуляцію логіки та візуального представлення кожного елемента (навігаційної панелі, модальних вікон, форми додавання локацій). Для управління станами компонентів використовуються вбудовані хуки React, зокрема useState для збереження динамічних даних та useEffect для виконання побічних ефектів, таких як асинхронні HTTP-запити до серверного API.

Ключовим елементом системи є інтерактивна карта, інтеграція якої здійснена за допомогою картографічної платформи Mapbox GL JS та бібліотеки-обгортки react-map-gl. Цей інструментарій використовує технологію WebGL для апаратного прискорення рендерингу векторних тайлів, що гарантує високу продуктивність навіть при відображенні великої кількості просторових об’єктів. Стан карти (координати центру та масштаб) зберігається у локальному стані компонента, що дозволяє динамічно оновлювати її положення при взаємодії користувача. Фрагмент логіки ініціалізації стану та завантаження просторових даних із сервера наведено на рисунку 3.4.

```
const [locations, setLocations] = useState([]);
const [viewState, setViewState] = useState({ longitude: 30.5234, latitude: 50.4501, zoom: 12
}); const fetchLocations = async (search = searchQuery) => {
try {
const response = await axios.get(url);
// Сервер повертає стандартизований об'єкт GeoJSON
setLocations(response.data.features || []);
 } catch (error) {
console.error("Помилка завантаження локацій", error);
}
};
useEffect(() => {
fetchLocations();
 }, [searchQuery, mainFilter, subFilter]);
```

Рисунок 3.4 – Ініціалізація стану та функція завантаження геоданих

Отримані від сервера дані формуються у стандартному форматі GeoJSON, де кожен об’єкт (Feature) містить просторові координати та набір метаданих (назву, опис, рейтинг, масив фотографій). Для візуалізації цих даних на карті застосовується метод ітерації масиву, який динамічно генерує компоненти Marker.

Кожному маркеру присвоюється унікальний ідентифікатор та координати довготи і широти. Крім того, зовнішній вигляд маркера кастомізується за допомогою CSS-класів залежно від головної категорії закладу (використовується кольорове кодування для швидкої візуальної ідентифікації). Структуру компонента карти та відображення набору маркерів продемонстровано на рисунку 3.5.

```
<Map {...viewState} onMove={evt => setViewState(evt.viewState)}
mapStyle="mapbox://styles/mapbox/dark-v11"
mapboxAccessToken={MAPBOX_TOKEN}<NavigationControl position="top-right" />

{locations.map((loc) => (
  <Marker
    key={loc.properties.id}
    longitude={loc.geometry.coordinates[0]}
    latitude={loc.geometry.coordinates[1]}
    anchor="bottom"
    onClick={(e) => handleMarkerClick(e, loc)}
  >
    <div className="custom-marker">
    </div>
  </Marker>
))}
</Map>
```

Рисунок 3.5 – Рендеринг інтерактивної мапи та масиву маркерів

Окрему увагу приділено проєктуванню користувацького досвіду (UX) при взаємодії з просторовими об’єктами. Замість переходу на окрему сторінку, детальна інформація про заклад відображається безпосередньо над картою у вигляді спливаючого вікна (компонент Popup). При кліку на маркер локальний стан selectedLocation оновлюється, що ініціює миттєвий рендеринг вікна.

У межах цього вікна реалізовано відображення базової інформації, обчисленого середнього рейтингу та каруселі мультимедійних файлів. Якщо

локація містить кілька фотографій або відео, користувач має змогу перемикати їх за допомогою спеціальних навігаційних кнопок, а також відкривати медіа у повноекранному режимі (Lightbox).

3.3 Розгортання застосунку та тестування системи

Завершальним етапом розробки геоінформаційного вебзастосунку став процес його впровадження (розгортання) у робоче середовище та комплексне тестування розробленого функціоналу. Для забезпечення доступності системи в мережі Інтернет було обрано хмарну PaaS-платформу (Platform as a Service) Render [8]. Вибір даного сервісу обґрунтований його нативною інтеграцією з системами контролю версій (GitHub) та вбудованою підтримкою середовищ Node.js і баз даних PostgreSQL.

Процес розгортання серверної частини здійснювався з використанням методології безперервної інтеграції та розгортання (CI/CD). Було налаштовано автоматичний пайплайн: при кожній фіксації змін (коміті) у головну гілку репозиторію платформа Render автоматично ініціює процес збірки проєкту (встановлення залежностей через npm install) та його перезапуск. Усі конфіденційні параметри, такі як секретний ключ JWT та реквізити доступу до бази даних, були перенесені з локального файлу .env у захищене середовище змінних оточення (Environment Variables) на панелі управління хостингу.

Окрему увагу було приділено розгортанню реляційної бази даних. Після створення екземпляра PostgreSQL у хмарі, до нього було підключено просторове розширення PostGIS шляхом виконання відповідного SQL-скрипта. Для забезпечення безпеки передачі даних між сервером додатків та базою даних було активовано протокол шифрування SSL/TLS, що відображено у конфігурації пулу з’єднань.

Після успішного розгортання інфраструктури було проведено функціональне тестування REST API за допомогою інструментів розробника

(Chrome DevTools) та платформи тестування API Postman. Тестування охоплювало перевірку всіх ключових кінцевих точок (End-points), коректність обробки HTTP-статусів та валідацію форматів даних.

Зокрема, тестування маршруту отримання локацій (GET /api/locations) підтвердило, що сервер успішно підключається до бази даних, виконує агрегацію відгуків та тегів, і повертає клієнту відповідь зі статусом 200 OK. Дані повертаються у стандартизованому картографічному форматі GeoJSON, що дозволяє клієнтському застосунку миттєво відмальовувати маркери на мапі.

Під час тестування модуля безпеки було перевірено стійкість системи до несанкціонованого доступу. При спробі відправки POST-запиту на створення нової локації без дійсного JWT-токена у заголовку Authorization, спрацьовує розроблений раніше фільтр authMiddleware, який миттєво перериває виконання операції та повертає помилку 401 Unauthorized.

Тестування клієнтської частини включало перевірку адаптивності інтерфейсу (Responsive Design) на різних роздільних здатностях екранів та кросбраузерну сумісність. Взаємодія з картою Mapbox GL (масштабування, панорамування) відбувалася плавно, без просідання частоти кадрів (FPS), що підтверджує ефективність використання технології WebGL для рендерингу просторових даних клієнтським пристроєм.

3.4 Керівництво користувача

Розроблений вебзастосунок має інтуїтивно зрозумілий інтерфейс, спроєктований за принципом мінімізації когнітивного навантаження на користувача. Взаємодія з системою відбувається через єдину сторінку (Single Page Application), де всі ключові елементи керування розташовані у швидкому доступі. Нижче наведено покрокову інструкцію щодо роботи з основним функціоналом платформи.

3.4.1 Головний екран застосунку, авторизація та інструменти фільтрації

При першому відкритті вебзастосунку користувач потрапляє на головний екран, який візуально розділений на дві основні зони: повноформатну інтерактивну мапу та бічну навігаційну панель (Sidebar). Інтерфейс виконано у темній колірній гамі (Dark Mode), що забезпечує високий контраст об’єктів та знижує втому очей при тривалому використанні. Зовнішній вигляд головного екрана наведено на рисунку 3.6.

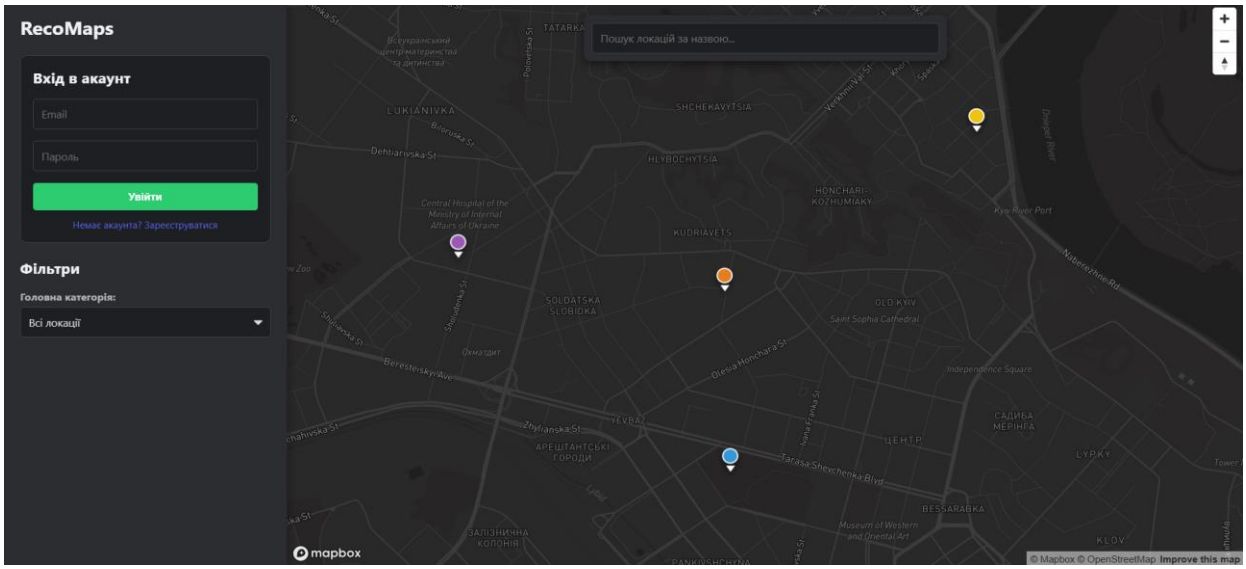


Рисунок 3.6 – Головний екран вебзастосунку RecoMaps

У лівій частині екрана розташована панель управління, яка містить логотип системи «RecoMaps» та блок авторизації. Для отримання доступу до розширеного функціоналу (додавання нових локацій, публікація відгуків) неавторизований користувач повинен увійти в систему. Форма містить поля для вводу електронної пошти та пароля, а також акцентну кнопку «Увійти». У разі відсутності облікового запису передбачено посилання «Немає акаунта? Зареєструватися», яке динамічно перемикає форму на режим реєстрації нового профілю, додаючи поле для вводу імені користувача.

Під блоком авторизації розміщено інструменти фільтрації просторових даних. Користувач може обрати головну категорію закладів (наприклад, «Їжа

та ресторани», «Культура», «Спорт») з відповідного випадаючого списку. При виборі основної категорії система автоматично генерує додатковий фільтр для уточнення запиту за вузькоспеціалізованим тегом (підкатегорією). Це дозволяє миттєво відсортувати маркери на мапі та залишити для перегляду лише релевантні об’єкти.

Центральну та праву частину екрана займає робоча область інтерактивної мапи. У її верхній частині по центру інтегровано пошуковий рядок («Пошук локацій за назвою...»), який дозволяє здійснювати швидкий текстовий пошук конкретних закладів.

Самі заклади відображаються на картографічній підкладці у вигляді краплеподібних маркерів із кольоровим кодуванням. Кожен колір відповідає певній головній категорії об’єкта (наприклад, помаранчевий маркер позначає заклади харчування, синій – місця для дозвілля та відпочинку, фіолетовий – культурні об’єкти). Такий підхід дозволяє користувачеві візуально орієнтуватися в інфраструктурі міста без необхідності додатково взаємодіяти з кожною точкою. У правому верхньому куті закріплено стандартні елементи керування (масштабування та скидання кута огляду) для зручної навігації.

3.4.2 Процес створення та редагування просторових об’єктів

Для наповнення платформи новими даними авторизований користувач має змогу створювати власні просторові об’єкти. Процес ініціюється кліком лівої кнопки миші по будь-якому вільному місцю на інтерактивній мапі. Після цього за обраними координатами миттєво з’являється тимчасовий маркер нової точки (візуально виділений білим кольором із синім контуром), а у лівій бічній панелі відкривається форма «Додати нову локацію». Інтерфейс процесу створення об’єкта наведено на рисунку 3.7.

Форма створення закладу складається з кількох логічних блоків. У верхній частині користувач обов’язково вводить назву локації у відповідне

текстове поле. Нижче розташовано модуль завантаження мультимедійних файлів (фото та відео). За допомогою акцентної кнопки «+ Додати медіа» користувач може прикріпити візуальний контент з пам’яті пристрою. Обрані файли одразу відображаються у вигляді списку з назвами (наприклад, SuperDuperMart-Fallout4.webp). Поруч із кожним файлом є кнопка із червоним хрестиком, що дозволяє легко видалити помилково доданий документ ще до моменту публікації.

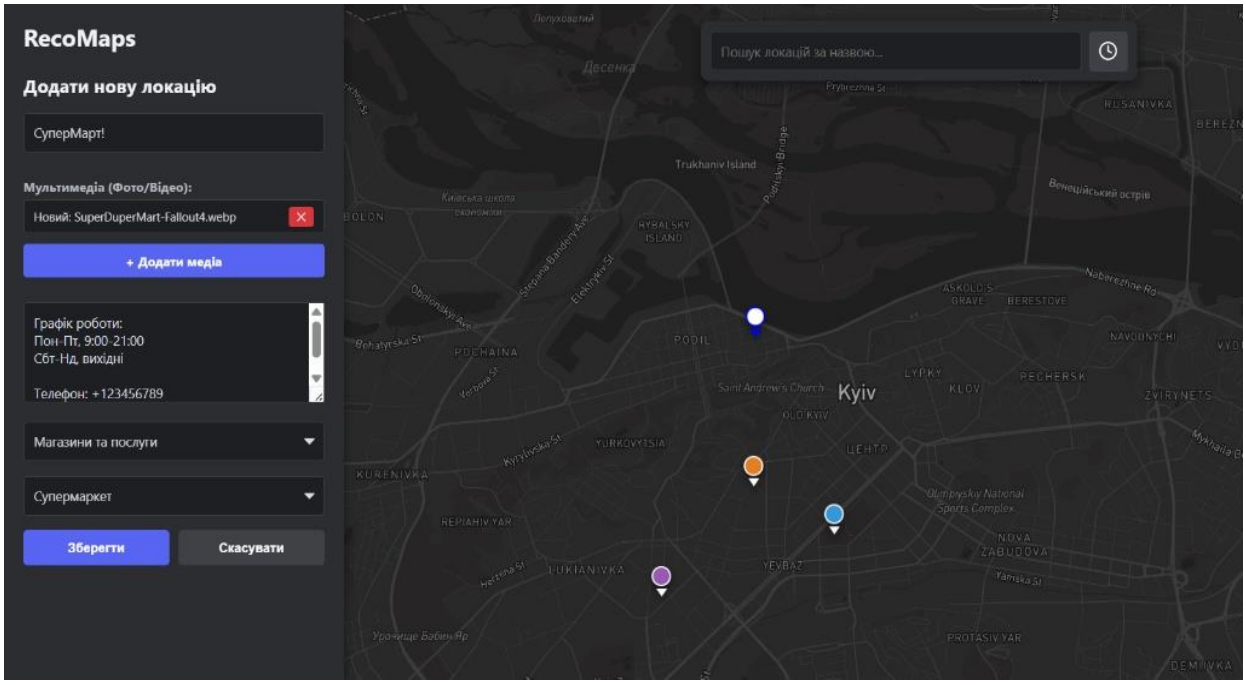


Рисунок 3.7 – Інтерфейс створення нової локації на мапі

Для розширеного опису закладу передбачено багаторядкове текстове поле. Сюди користувач може внести будь-яку корисну інформацію, яка допоможе іншим відвідувачам: детальний графік роботи, контактні номери телефонів, посилання на соціальні мережі чи загальний опис послуг.

Наступним кроком є класифікація просторового об’єкта для його коректної роботи в системі фільтрації. Користувач обирає релевантну опцію з двох випадаючих списків: спочатку вказується головна категорія (наприклад, «Магазини та послуги»), а потім уточнюючий тег (наприклад, «Супермаркет»).

Завершується процедура натисканням синьої кнопки «Зберегти». Після цього всі заповнені текстові дані, мультимедійні файли та точні географічні координати відправляються на сервер, форма закривається, а маркер отримує відповідний категорії колір і стає доступним для всіх відвідувачів платформи. Якщо ж користувач передумав додавати об’єкт, він може скористатися кнопкою «Скасувати», яка очищує введені дані та прибирає тимчасовий маркер з мапи. Також слід зазначити, що у верхньому правому куті пошукового рядка авторизованим користувачам доступна кнопка з іконкою годинника, яка відкриває випадаючий список з історією останніх переглянутих локацій.

3.4.3 Перегляд та управління існуючими локаціями

При взаємодії з маркером на мапі користувач не перенаправляється на іншу сторінку, що зберігає безшовність користувацького досвіду. Замість цього безпосередньо над маркером з’являється спливаюче інформаційне вікно (Popup). У цьому вікні відображається базова інформація про локацію: головна фотографія, назва закладу, присвоєні теги-категорії (наприклад, «Магазини та послуги», «Супермаркет»), а також короткий опис (графік роботи, контактні дані).

Для авторів публікації або адміністраторів системи у нижній частині спливаючого вікна автоматично генеруються кнопки «Редагувати» та «Видалити». Для переходу до повного опису закладу користувачу необхідно натиснути кнопку «Детальніше...». Інтерфейс спливаючого вікна локації наведено на рисунку 3.8.

Після натискання кнопки детального перегляду вся розширена інформація завантажується у ліву бічну панель (Sidebar), звільняючи простір мапи. Інтерфейс детального перегляду містить повний текстовий опис та збільшене зображення об’єкта. У правому нижньому куті фотографії

передбачено спеціальну іконку для переходу в повноекранний режим (Lightbox), що дозволяє роздивитися медіаконтент у високій роздільній здатності.

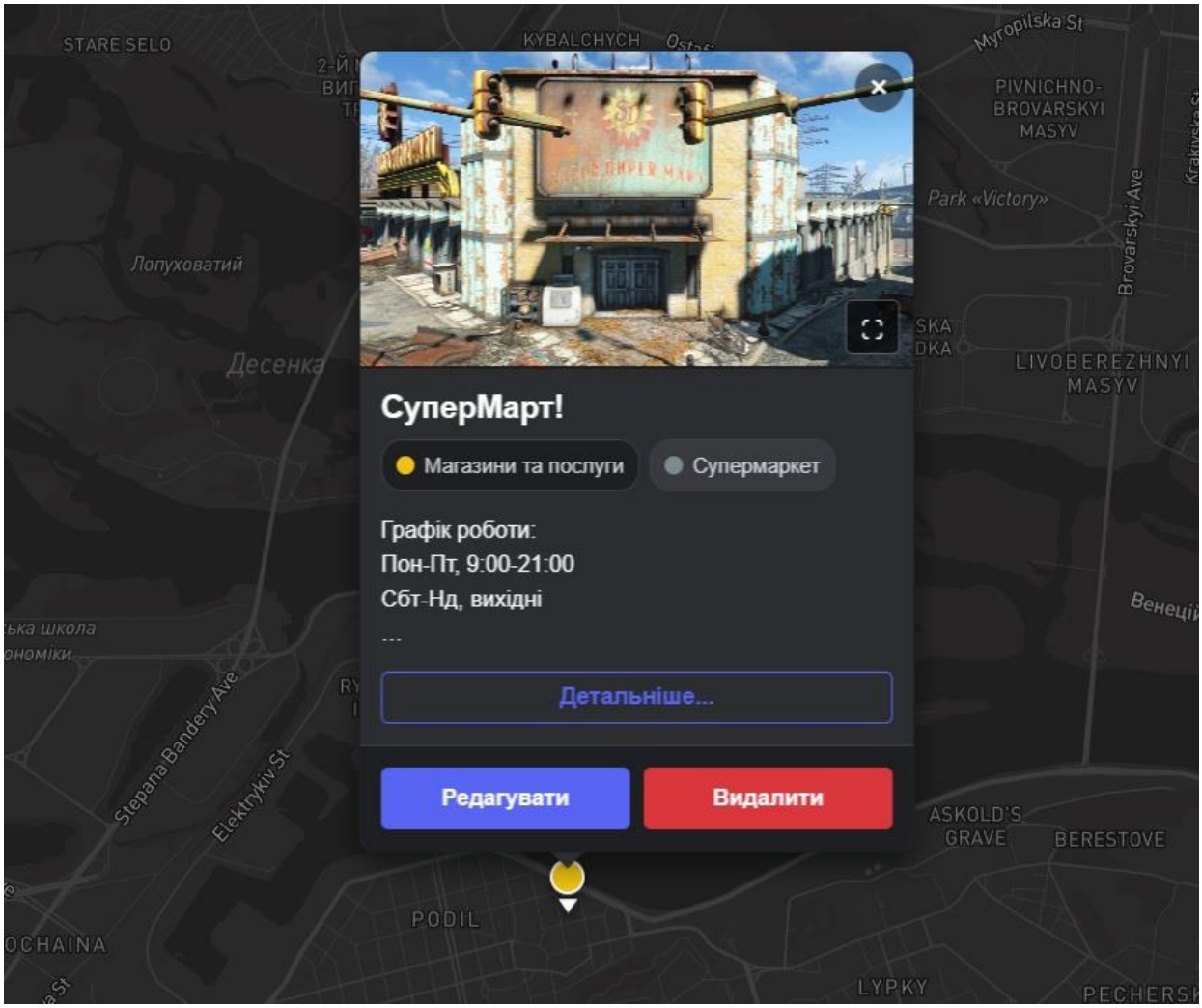


Рисунок 3.8 – Спливаюче інформаційне вікно локації на мапі

У верхній частині панелі розміщена кнопка-посилання «Повернутися до списку», яка дозволяє швидко закрити детальну інформацію та повернутися до результатів пошуку або загального списку локацій. Зовнішній вигляд панелі детального перегляду представлено на рисунку 3.9.

Важливою функцією системи є можливість підтримки актуальності даних. Якщо користувач є автором локації (або має права адміністратора), він може перейти в режим редагування. При активації цієї функції відкривається форма «Редагувати локацію», у якій всі поля вже попередньо заповнені

поточними даними із бази даних. Особливістю даного модуля є гнучке управління мультимедійним контентом.

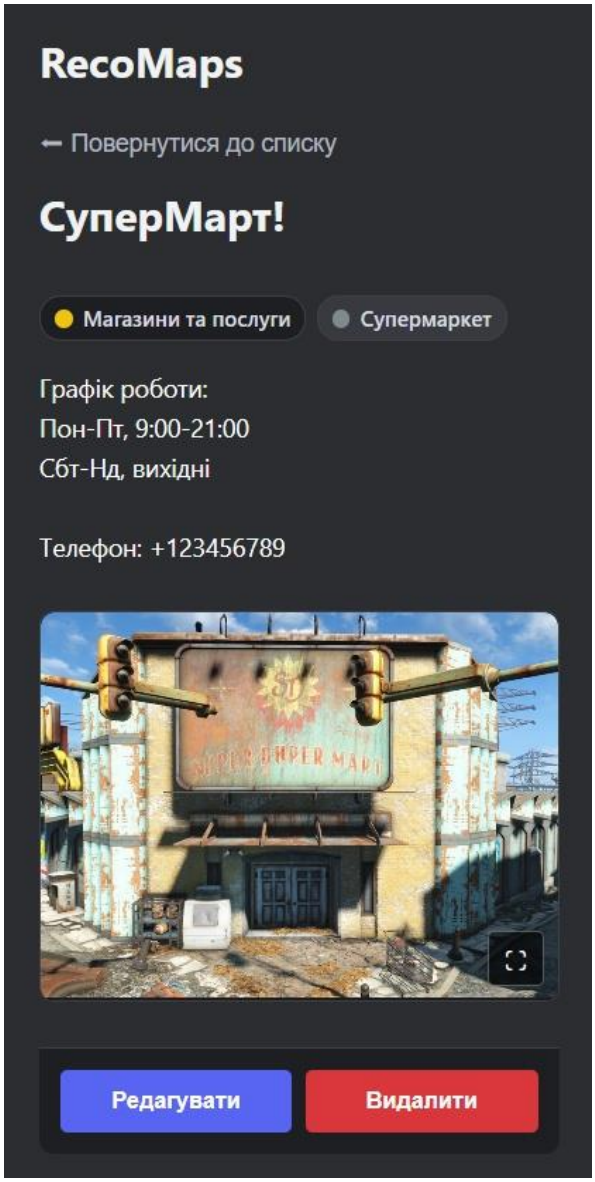


Рисунок 3.9 – Детальний перегляд інформації про заклад у бічній панелі

Користувач бачить список уже збережених файлів на сервері і може вибірково видаляти застарілі фотографії за допомогою червоної кнопки «Видалити» поруч із назвою файлу. Одночасно з цим через кнопку «+ Додати медіа» можна завантажити нові зображення. Після внесення необхідних змін у текстовий опис або зміни категорії, користувач натискає кнопку «Оновити», і дані миттєво синхронізуються із сервером. Інтерфейс модуля редагування наведено на рисунку 3.10.

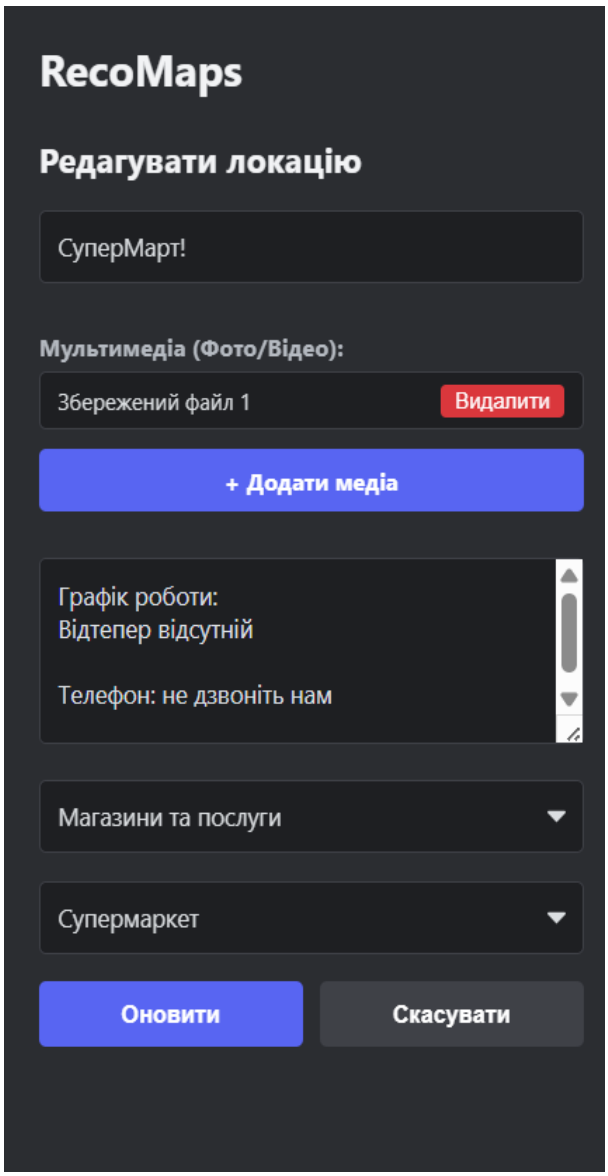


Рисунок 3.10 – Інтерфейс редагування існуючої локації та управління медіаконтентом

3.4.4 Система оцінювання та соціальної взаємодії користувачів

Для формування об’єктивного рейтингу закладів у вебзастосунку реалізовано розширену систему соціальної взаємодії. Блок відгуків розташований у нижній частині панелі детального перегляду локації. Система автоматично агрегує всі залишені оцінки та виводить середній бал великим шрифтом (наприклад, 3.0), загальну кількість відгуків та візуальну діаграму розподілу оцінок. Діаграма реалізована у вигляді прогрес-барів для кожної

кількості зірок (від 1 до 5), що дозволяє користувачам миттєво оцінити співвідношення позитивних та негативних коментарів.

Під блоком статистики розташована форма «Залишити відгук», яка доступна виключно для авторизованих користувачів. Процес оцінювання є максимально інтуїтивним: користувач обирає кількість зірок шляхом наведення та кліку на інтерактивний елемент рейтингу, після чого вводить текстовий коментар і натискає кнопку «Відправити». Опубліковані відгуки виводяться у вигляді стрічки нижче, де зазначається ім’я автора, дата публікації, виставлена оцінка та сам текст коментаря. Інтерфейс блоку статистики та списку відгуків продемонстровано на рисунку 3.11.

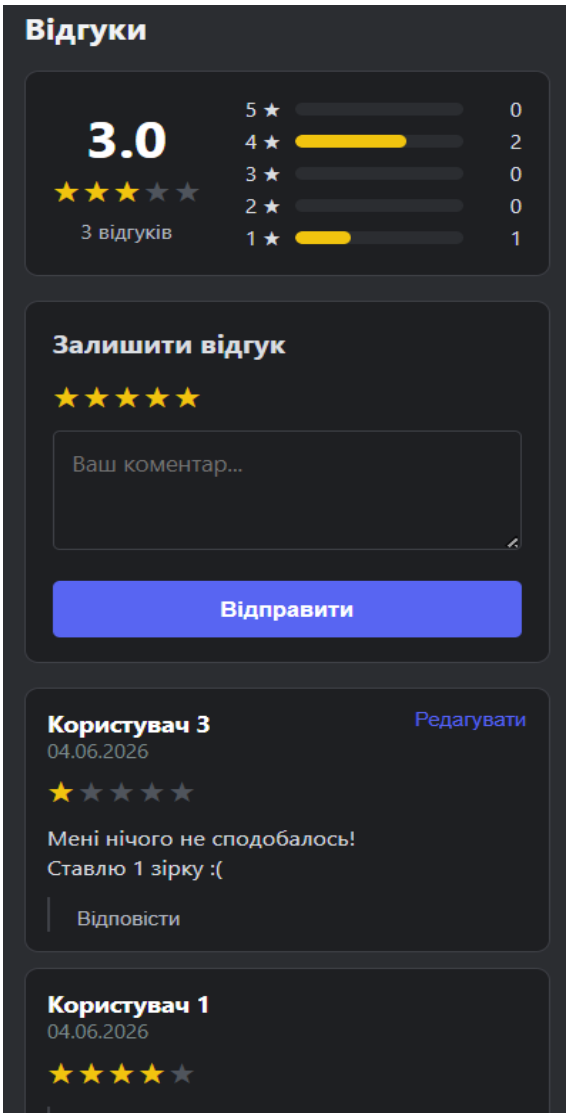


Рисунок 3.11 – Загальна статистика рейтингу та стрічка користувацьких відгуків

З метою створення повноцінної дискусійної платформи, до системи відгуків було інтегровано можливість багаторівневого коментування. Кожен залишений відгук супроводжується кнопкою «Відповісти». Після її натискання розгортається додаткове текстове поле для введення відповіді. Ця функція є критично важливою для власників закладів або адміністраторів, оскільки дозволяє їм комунікувати з відвідувачами, реагувати на критику або дякувати за позитивний зворотний зв’язок.

Відповіді на відгуки візуально зміщені вправо (nested replies), що створює чітку ієрархічну структуру діалогу і дозволяє легко відрізнити головний відгук від обговорення під ним. Процес взаємодії з системою відповідей наведено на рисунку 3.12.

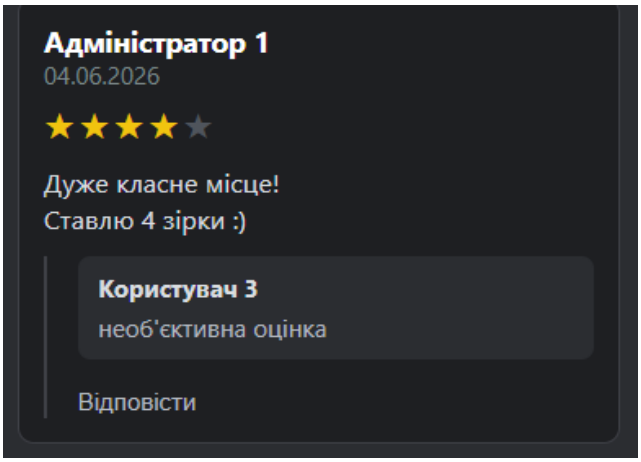


Рисунок 3.12 – Відображення ієрархічних відповідей на відгуки

Таким чином, реалізовано систему оцінювання та соціальної взаємодії користувачів.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне інженерне завдання – розроблено легкий та інтуїтивно зрозумілий вебзастосунок з інтерактивною картою для обміну рекомендаціями щодо міських локацій «RecoMaps». В результаті виконання роботи було пройдено всі етапи життєвого циклу створення програмного забезпечення: від детального аналізу предметної області до розгортання готового продукту в хмарному середовищі. Підсумовуючи результати проведених досліджень та розробок, можна зробити наступні висновки.

На початковому етапі було проведено аналіз предметної області та існуючих ГІС-рішень на ринку. Виявлено проблему перенасиченості універсальних картографічних гігантів, що ускладнює користувацьку навігацію. На основі цього сформовано вимоги до системи та обґрунтовано вибір передового технологічного стека: Node.js (Express) для обробки серверних запитів, PostgreSQL із розширенням PostGIS для ефективного управління просторовими даними та бібліотеку React для побудови динамічного інтерфейсу.

Під час проєктування розроблено нормалізовану реляційну схему бази даних, яка забезпечує збереження та обробку об’єктів із географічними координатами, а також інформації про облікові записи, мультимедійні файли та відгуки. За допомогою уніфікованої мови моделювання (UML) спроєктовано клієнт-серверну архітектуру. Також створено адаптивний користувацький інтерфейс у темній колірній гамі, який забезпечує коректне відображення мапи та зручну навігацію на різних типах пристроїв.

У ході програмної реалізації серверної частини створено масштабований REST API. Інтегровано модуль для збереження мультимедійних файлів та налаштовано механізми безпеки: паролі облікових записів захищені криптографічним алгоритмом хешування, а для автентифікації користувачів

реалізовано безсесійний підхід на базі JSON Web Tokens (JWT). Це забезпечило надійний захист даних та стабільну роботу бекенду.

Логіку взаємодії з інтерактивною картою на стороні клієнта реалізовано у вигляді односторінкового вебзастосунку (SPA). Завдяки інтеграції картографічної платформи Mapbox GL JS та технології WebGL створено високопродуктивну мапу, що дозволило суттєво зменшити витрати апаратних ресурсів при рендерингу великої кількості маркерів. Впровадження динамічної фільтрації за категоріями, текстового пошуку та інтуїтивної системи соціальної взаємодії (відгуки, рейтинги) забезпечило значне скорочення часу користувача на пошук і обмін рекомендаціями.

На завершальному етапі розроблений програмний продукт успішно протестовано та розгорнуто у хмарному середовищі Render. Налаштовано процес безперервної інтеграції (CI/CD) та підключення до бази даних через захищений протокол SSL/TLS, що підтвердило його готовність до використання цільовою аудиторією.

Таким чином, мета кваліфікаційної роботи – скорочення часу користувача на пошук і обмін рекомендаціями щодо міських локацій та зменшення витрат апаратних ресурсів при рендерингу маркерів – досягнута повною мірою. Усі поставлені завдання виконано, а розроблений продукт має високе практичне значення та потенціал для подальшого масштабування.

ПЕРЕЛІК ПОСИЛАНЬ

1. React – A JavaScript library for building user interfaces. URL: https://react.dev/ (дата звернення: 10.03.2026).

2. Node.js is a JavaScript runtime built on Chrome’s V8 JavaScript engine. URL: https://nodejs.org/ (дата звернення: 10.03.2026).

3. Express – Node.js web application framework. URL: https://expressjs.com/ (дата звернення: 10.03.2026).

4. PostgreSQL: The World’s Most Advanced Open Source Relational Database. URL: https://www.postgresql.org/ (дата звернення: 10.03.2026).

5. PostGIS – Spatial and Geographic objects for PostgreSQL. URL: https://postgis.net/ (дата звернення: 10.03.2026).

6. Mapbox GL JS – Interactive, thoroughly customizable maps in the browser. URL: https://docs.mapbox.com/mapbox-gl-js/ (дата звернення: 17.03.2026).

7. JSON Web Tokens – An open, industry standard RFC 7519 method for representing claims securely between two parties. URL: https://jwt.io/ (дата звернення: 17.03.2026).

8. Render – Cloud Hosting for Developers. URL: https://render.com/ (дата звернення: 17.03.2026).

9. Multer – Node.js middleware for handling multipart/form-data. URL: https://www.npmjs.com/package/multer (дата звернення: 17.03.2026).