

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему: РОЗРОБКА ПРОГРАМИ ВЕБЗАСТОСУНКУ

«ТУРИСТИЧНА ОНЛАЙН КАРТА»

SOFTWARE DEVELOPMENT OF A TOURIST ONLINE MAP
APPLICATION

Виконав: студент(ка) 4 курсу, групи КНТ-128

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Кравченко В.М.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Касьян М.М.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кравченко Владислава Миколайовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програми вебзастосунку «Туристична онлайн карта». Software Development of a Tourist Online Map Appplication
 керівник проєкту (роботи) Сердюк Сергій Микитович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання. 2. Розробка архітектури програми. 3. Розробка програмного забезпечення. 4. Керівництво програ міста. 5. Керівництво оператора.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтроль	Белова А.В., асистент		

7. Дата видачі завдання “28” лютого 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програмного забезпечення	4– 5 тижні	Розділ 3
5	Керівництво програміста	6 тиждень	Розділ 4
6	Керівництво оператора	7 тиждень	Розділ 5
8	Оформлення пояснювальної записки.Нормоконтроль та рецензування	8 тиждень	Додатки
9	Захист роботи	9 тиждень	

Студент(ка)

Кравченко В.М.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи)

Сердюк С.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавру:
110 с., 3 табл., 52 рис., 2 дод., 17 джерел.

ТУРИСТИЧНА ОНЛАЙН КАРТА, ВЕБЗАСТОСУНОК, ПОДОРОЖІ,
REACT.

Об'єкт дослідження – процеси обробки інформації.

Предмет дослідження – методи обробки інформації у сфері туризму.

Мета роботи – збільшити поінформованість користувача з приводу туристичних об'єктів у світі, покращити юзабіліті взаємодії користувача з інформаційними ресурсами у сфері туризму.

Матеріали, методи та технічні засоби – функціональна мова програмування, мова програмування JavaScript, технологія розробки для клієнтської частини React.js, технологія для розробки серверної частини Node.js, персональний комп'ютер з процесором IntelCore i5-3350 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний вебзастосунок у вигляді онлайн карти, який дозволяє проглядати користувацькі відгуки щодо різних туристичних об'єктів у світі, та залишати свої точки на карті з відгуками щодо тих чи інших туристичних об'єктів.

Висновки. Розроблено програмну систему для отримання релевантної інформації з приводу різних туристичних об'єктів у світі.

Галузь використання – туризм.

ABSTRACT

Explanatory note to the diploma of the bachelor's qualification work: 110 p., 3 tables, 52 figures, 2 reports, 17 references.

TOURIST ONLINE MAP, WEB PAGE, SUBROADS, REACT.

The subject of the research - information processing processes.

Subject of research - methods of information processing in the field of tourism.

The goal of the work is to increase the knowledge of the user about tourist objects in the world, to increase the user's interaction with information resources in the field of tourism.

Materials, methods and tools - functional programming language, JavaScript programming language, development technology for the client's part React.js, technology for development of the server part Node.js, personal computer with an Intel Core i5-3350 processor under control of the Microsoft Windows 10 operating system.

Results. A Web-based online chart viewer was created which allows users to view reviews of various tourist destinations around the world, and leave their points on the map with reviews of certain tourist destinations.

Conclusions. Developed a program system for obtaining relevant information on various tourist sites in the world.

The sphere of use is tourism.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	8
ВСТУП.....	9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	11
1.1 Огляд інформаційних потреб у туристичної області	11
1.2 Аналіз існуючих аналогів.....	14
1.3 Технічне завдання	20
1.3.1 Підстава для розробки	20
1.3.2 Мета розробки та її призначення.....	20
1.3.3 Основні вимоги до програмного продукту	20
1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування	21
1.3.5 Вимоги до інформаційної та програмної сумісності.....	22
1.3.6 Вимоги до програмної документації.....	22
1.3.7 Порядок приймання і тестування	22
1.4 Висновки за розділом 1.....	22
2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ	24
2.1 Моделювання програмного забезпечення	24
2.2 Розробка архітектури програмного забезпечення	26
2.2.1 Опис модулів системи	27
2.3 Обґрунтування вибору інструментів розробки.....	28
2.3.1 Обґрунтування вибору мови програмування	28
2.3.2 Обґрунтування вибору фреймворків та бібліотек	28
2.3.3 Обґрунтування вибору бази даних	30
2.3.4 Обґрунтування вибору IDE.....	30
2.4 Висновки за розділом 2	31
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	32
3.1 Розробка компонентів системи.....	32
3.1.1 Розробка компонентів контролера	33

3.1.2 Розробка компонентів моделі	40
3.1.3 Розробка компонентів представлення.....	41
3.2 Розробка програмного інтерфейсу	56
3.3 Тестування застосунку.....	65
3.4 Висновки за розділом 3	71
4 КЕРІВНИЦТВО ПРОГРАМІСТА	73
4.1 Призначення та умови застосування.....	73
4.1.1 Призначення програми	73
4.1.2 Умови застосування програми.....	73
4.2 Характеристики програми.....	73
4.3 Звертання до програми	74
4.4 Початкові та вихідні дані	74
4.5 Повідомлення програмісту.....	74
5 КЕРІВНИЦТВО ОПЕРАТОРА	75
5.1 Призначення програми	75
5.2 Умови виконання програми	75
5.3 Виконання програми.....	75
5.4 Повідомлення.....	78
ВИСНОВКИ.....	79
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	81
ДОДАТОК А.....	83
ДОДАТОК Б	101

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

UI – користувацький інтерфейс;
E-MAIL – адреса онлайн поштової скриньки;
ID – унікальний ідентифікатор;
SR – принцип єдиної відповідальності;
MVC – модель, представлення, контролер;
REST - передача репрезентативного стану;
DOM – документна об'єктна модель сторінки браузера;
HTML – мова гіпертекстової розмітки сторінки браузера;
API – програмний інтерфейс програми;
URL – уніфікований локатор ресурсів, адресна строка у браузері;
HTTP – протокол передачі даних у інтернеті;
IDE – інтегрована середовище розробки;
UML – уніфікована мова моделювання;
JSON – JavaScript об'єктна нотація документа;
JSX – розширення мови JavaScript;
SEO – пошукова оптимізація.

ВСТУП

Після того як пройшла основна хвиля коронавірусу, і кордони держав почали знов відкриватися, людство знову починають мандрувати, їздити у відпустки та на курорти. Кожна подорож, не важливо від того яка у неї мета, починається з планування, де можна зняти житло, де можливо зупинитися поїсти або на яку архітектурну чи історичну споруду відвідати. Планування невід’ємна частина будь-якої подорожі, але для планування, людина повинна мати інформацію з приводу тієї країни чи тієї місцевості куди вона їде. Звичайно, у вік високих технологій у нас є інтернет, і всю інформацію можливо знайти за хвилину у ньому, але будь-яка інформація ціниться не тільки обсягом та корисним змістом, але й її структурованістю. Також людина це істота соціальна, її хочеться ділитись враженнями з іншими людьми, передавати важливу інформацію. Це і розповсюджується на подорожі, наприклад поділитися враженнями від побачених архітектурних споруд, чи порадити певний ресторан, якщо людина захоче приїхати у ту ж країну, або навіть порадити яке блюдо більш сподобалось у цьому ресторані, певні аспекти місцевої кухні про які не напишуть у багатьох гідах по країнах і так далі.

Саме тому необхідно мати таке джерело інформації, яка буде містити не тільки корисний зміст, але й мати певну структурованість, щоб користувач зміг швидко дізнатися інформацію про певний об’єкт на карті, не витрачаючи на це багато часу та дізнавшись думку інших людей з приводу цього об’єкту.

Звичайно є альтернативи статичній публіцистичній інформації, наприклад ті ж гугл-карти. Але у цих альтернатив є спільна проблема – багато інформації, яка може знадобитися далеко не кожному користувачу, і ця інформація дуже захаращує інтерфейс, не завжди інтуїтивно зрозуміло, де подивитись відгуки та певну інформацію, тому що людині важливо насамперед дізнатися думку інших користувачів. Тому важливо мати такий ресурс, у якому користувачі можуть ділитись між собою власними думками з приводу того чи іншого місця, без надлишкової інформації, яка часто є не зовсім дійсною.

Об'єктом дослідження є процеси отримання інформації, що релевантна потребам користувача.

Предметом дослідження є методи отримання релевантної інформації з приводу подорожі світом.

Мета роботи – збільшити поінформованість користувача з приводу туристичних об'єктів у світі, покращити процес отримання релевантної та структурованої інформації для користувача у предметній області.

Завдання дослідження:

- аналіз предметної області, аналіз проблем предметної області;
- аналіз існуючих аналогів;
- розробка функціональних вимог програми;
- розробка архітектури програми, та обрання інструментів для розробки;
- розробка моделей програмного забезпечення;
- провести тестування програмного застосунку;

Практичним застосуванням програмного застосунку є отримання користувачем релевантної та структурованої інформації з приводу того чи іншого туристичного об'єкта у світі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд інформаційних потреб у туристичній області

На даний момент, всі люди які вимушено перенесли свої туристичні та ділові подорожі з-за пандемії коронавіруси, вони починають знов планувати свої поїздки, але хоч кордони і відкриті, деякі місця можуть працювати з обмеженнями або не працювати зовсім.

Якщо казати у цілому, то постраждала вся розважальна галузь у світі, як для туристів так і для внутрішнього користувача, це і різноманітні розважальні комплекси, і місця масового відпочинку, і різноманітні культурні заходи, наприклад музичні концерти. Кожний елемент моделі розважальної галузі зазнали втрат, у тому числі великі грошові втрати зазнала і туристична галузь.

На даний момент туристична галузь починає відновлюватися після пандемії коронавіруси, на час якої майже усі країни закрили свої кордони для іноземних туристів та для заробітчан. Пандемія коронавірусу нанесла значну шкоду світовій галузі туризму, що на 2020 рік складає скорочення відвідування іноземними туристами країн світу більш ніж на 72 відсотки, що складає приблизно 1 мільярд людей [1].

Туризм, у даний час являє собою не тільки елемент розважальної галузі, але є і самостійною системою, зі своїми елементами. До цих елементів ми можемо віднести:

- туристичний продукт – це те що ми можемо назвати туристичними послугами, тобто це туристичні тури, маршрути, які відповідають потребам користувача, це критерій зазвичай є найбільш вагомим у подорожі;
- суб'єкти туризму – це внутрішні або іноземні туристи, громадські, державні і приватний підприємства і організації туризму і подорожей;
- учасник туризму – це людина, яка користується продуктами та послугами, які створює система туризму, тобто будь який турист, внутрішній чи іноземний;

- об'єкт туризму – уся сфера пов'язаний з туризмом послуг, тобто харчування, житло яке надається користувачу туристичних послуг;
- організатор – це юридичне або приватний підприємець, який забезпечує планування і здійснення заходів, програм, створення і поширення туристичного продукту [10].

Всі ці елементи і створюють велику туристичну систему, у якій звичайно всі елементи однаково важливі, всі елементи тісно пов'язані між собою, але все ж головним елементом у цій системі звичайно є учасник туризму. Всі елементи працюють лише на потреби користувача, і якщо кожна певна система не впорюється зі своїми завданнями, з приводу обслуговування головного елементу системи, то система може вважатися недієздатною. Учасник туризму вносить необхідні ресурси, тобто грошові одиниці, задля того щоб система змогла існувати у сучасних реаліях, змогла поповнювати власні системні ресурси, які система витратила на обслуговування учасника туризму, покращувати якість функціонування інших елементів системи, збільшувати цінність самої системи. Система, задля того щоб забезпечити потреби свого головного елемента системи, повинна виконувати певні завдання. Задачі можуть відрізнятися від певного типу туристичної системи, але можливо виділити дві задачі, які будуть спільними, незважаючи на контекст туристичної системи.

Першорядна задача туристичної галузі, ввести людину у стан так званого «Вау» ефекту, що збільшить шанс на те що турист повернеться наступного разу у це ж місце, скористається послугами саме цієї туристичної компанії, та розповість про свої враження знайомим чи друзям, порекомендує відвідати саме цей туристичний об'єкт та саме цю туристичну компанію. Далі по важливості вже іде комфорт туриста, що частково є суб'єктивною річчю, але є загальноприйняті стандарти та сертифікації туристичних послуг [2] які повинні надавати особи які беруть відповідальність за розміщення туристів.

Ці два фактори є ключові у плануванні подорожі. Але як часто можливо побачити об'єктивну правду, наприклад з приводу того чи іншого туристичного

готелю, або з той чи іншої бази відпочинку? Як правило все обмежується інформацією з головної сторінки сайту туристичного об'єкту, яким зацікавлений користувач, якщо такий ресурс у туристичного об'єкту є взагалі. А як скажемо дізнатися перелік тих чи інших місць відпочинку, скажемо у іншій країні, порівняти їх, продивитись відгуки користувачів які були у цьому місці? Як правило, користувач зустрічається з проблемою у пошуку релевантної для нього інформації у інтернеті, натикаючись у інтернеті на статичну інформацію яка вигідна власнику туристичного закладу, а не туристу.

Користувачу необхідно мати такі ресурси, які будуть давати об'єктивну інформацію від інших туристів, стислу та релевантну, у користувачів які планують ту чи іншу вид подорожі не повинна виникати проблема з пошуку інформації. Тому на зараз велика актуальність таких ресурсів, які будуть задовольняти такі потреби у пошуку інформації з приводу туристичних об'єктів.

Мета даної роботи, розробити веб-застосунок, який буде дозволяти користувачам обмінюватися враженнями та оцінками з приводу того чи іншого туристичного об'єкту у світі, без зайвої статичної інформації від власники цих туристичних об'єктів. У веб-застосунку, який буде розроблений у ході виконання роботи, буде присутні два типи користувачів, які будуть чітко розділяти функціонал майбутнього застосунку та заохочити людей становитись частиною системи та розширювати користувацьку базу у системі запропонованими перевагами для певного типу користувача,. Необхідно вести у майбутню систему можливість залишати відгуки з приводу тих чи інших туристичних об'єктів тільки у випадку якщо користувач є зареєстрованим, тобто частиною системи. Тим часом, в незалежності від того чи є людина частиною системи, він може ознайомитись з інформацією на сайті, може ознайомитись з відгуками користувачів. Більш докладнішу інформацію з приводу можливостей різних типів користувачів та функціональних можливостей системи, буде зазначено у документи технічного завдання.

Для початку необхідно провести аналіз вже існуючих аналогів, щоб виділити основні недоліки та переваги цих аналогів, на основі яких буде побудований документ з основними функціональними можливостями веб-застосунку.

1.1 Аналіз існуючих аналогів

У вигляді існуючих аналогів було обрано три веб-застосунки, які представляють собою туристичну онлайн карту. У ході аналізу цих веб-застосунків, необхідно дати короткий опис застосунку, виділити основні переваги та недоліки.

Discover.ua.

Discover.ua – це туристична онлайн карта України представлена на рис. 1.1 [3], у якій кожний бажаючий користувач зможе ознайомитись з різноманітними туристичними точками всієї України, від архітектурних та історичних споруд, до прокату лиж та ресторанів у певній місцевості. Застосунок також надає можливості записатися до гідів на екскурсію, надає можливості самостійно обрати та подивитись вже створений маршрут. Проект було розроблено у співпраці з Google у рамках компанії «Цифрове перетворення України» [4]. Веб-застосунок розробила українська компанія «JamDigital» [5] у період з 2017-2021 рік.

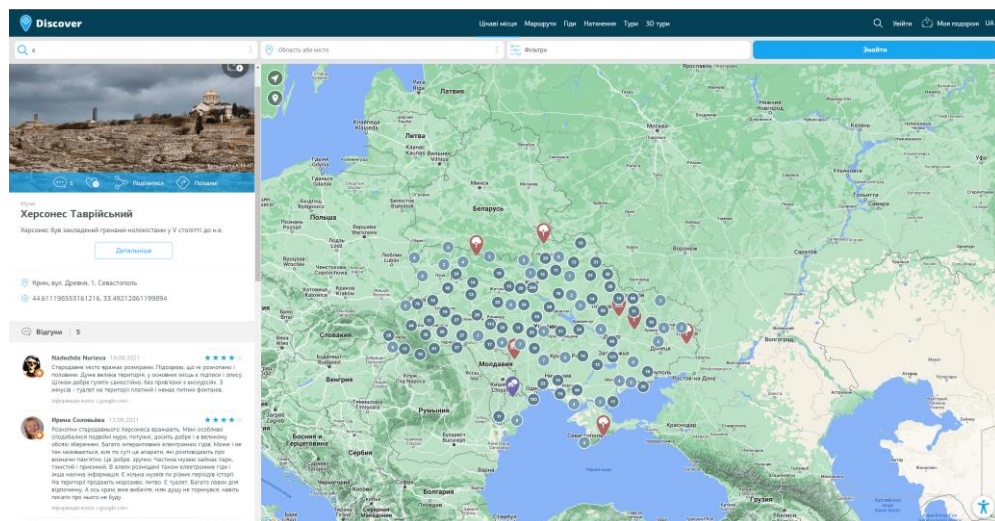


Рисунок 1.1 – Discover.ua – туристична онлайн карта України [3]

Переваги:

- багатий функціонал. Є фільтри пошуку для як для типу потреб вашого туристичного запиту, так і фільтр туристичних потреб для людей з вадами здоров'я, фільтр пошуку представлений на рис 1.2 [3].
- можливість залишати коментарі з приводу тієї чи іншої архітектурної споруди;
- плавні приємні анімації при взаємодії з вебзастосунком.

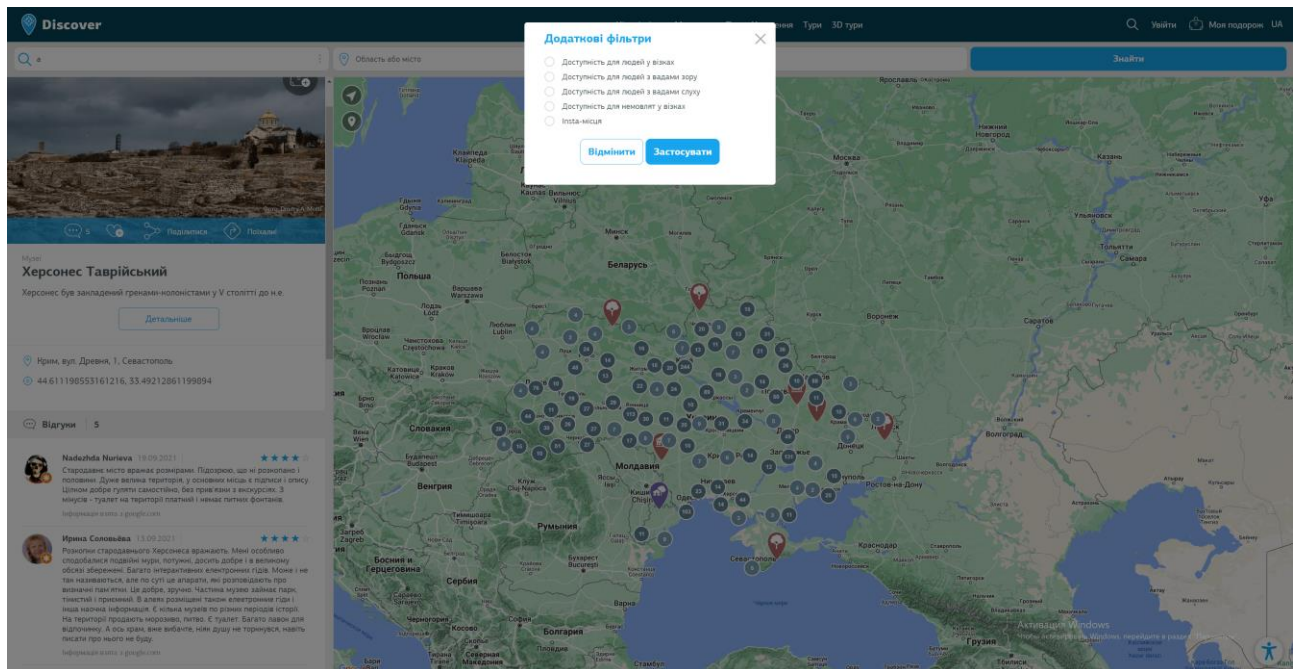


Рисунок 1.2 – Фільтр туристичних вподобань для людей з вадами здоров'я [3]

Недоліки:

- карта обмежена лише територією України;
- не зовсім зрозумілий UI;
- деякі проблеми з працездатності застосунку, не повторному переході на одні і ті ж крапки, перехід не працює;
- сайт виключно українською мовою, не адаптовано для використання іноземним користувачем;
- схожі між собою деякі іконки, що не дає зрозумілості з приводу типу архітектури даної точки.

Opentripmap.com.

Opentripmap.com – це туристична світова онлайн представлена на рис 1.3 [6], яка представляє собою мапу світу, з точками по у всіх країнах світу, які представляють собою збірну інформацію як з туристичних пам'яток так і з готелей та ресторанів. Реалізований функціонал фільтрацій різних потреб, від цікавих місць і готелів, до спортивних об'єктів, наприклад тренувальних залів, стадіонів, басейну та дайвінгу. Є також можливість прокладати маршрути із точки А в точку Б, також є реалізований функціонал термомапи, що показує наскільки високий показник плюсової температури у світі. Веб-застосунок було розроблено на основі програмного інтерфейсу іншого програмного застосунку з відкрити вихідним кодом Openmaptiles [7], представлений на рис 1.4.

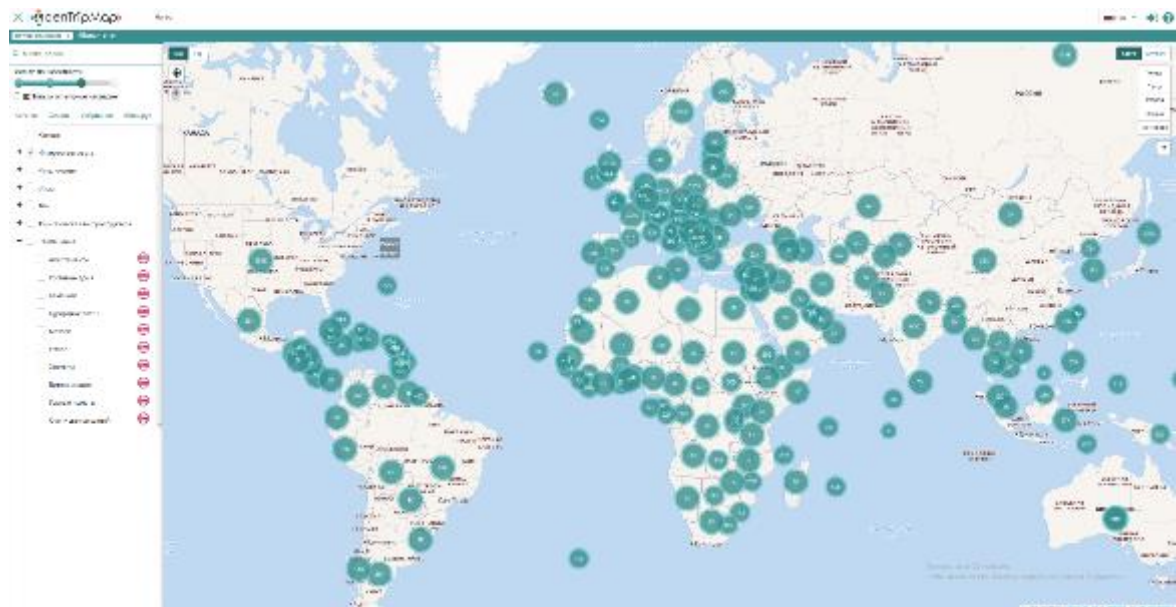


Рисунок 1.3 – Opentripmap.com [6]



Рисунок 1.4 – Openmaptiles.org [7]

Переваги:

- є на вибір декілька мов інтерфейсу, російська та англійська;
- мапа відображає точки по усьому світу, де є туристичні пам'ятки;
- є різноманітні фільтри для обрання типу об'єкта який вас цікавить;
- зрозумілий інтерфейс;
- послідовне відображення інтерфейсу.

Недоліки:

- інформація статична, нема можливості залишити коментар з приводу того чи іншого об'єкта, та не можливості ставити свої точки;
- деякі недоліки з приводу UI, не завжди спрацьовує клік по елементам інтерфейсу.

Google Maps.

Набір програм, побудованих на основі безкоштовного картографічного сервісу та технології, що надаються компанією Google. Створено 2005 року.

Сервіс являє собою карту та супутникові знімки планети Земля. Для багатьох регіонів доступні високодеталізовані аерофотознімки (зняті з висоти 250-500 м), для деяких – з можливістю перегляду під кутом 45° з чотирьох сторін світла.

Із сервісом інтегрований бізнес-довідник та карта автомобільних доріг з пошуком маршрутів, що охоплюють США, Канаду, Японію, Росію, Гонконг, Китай, Велику Британію, Ірландію (тільки центри міст) та деякі райони Європи [8].

Переваги:

- стабільність роботи застосунку;
- багатофункціональність, починаючи від огляду карточки об'єкту на карті який цікавить користувача, закінчуючи прокладанням маршруту від точки А до точки Б, з витратами часу, транспортом який ходить по цьому маршруту, та інше;

- мультимовний, так як переводить мову інтерфейсу в залежності від мови системи користувача;

- можливість залишати свої точки на карті;
- можливість коментувати точки.

Недоліки:

- для того щоб залишити, змінити, або видалити свою точку на карті, доведеться написати до тех-підтримки гугл, де вже вони будуть приймати рішення про додання чи не додання точки на карті;

- дуже захаращений інтерфейс, не дуже зрозумілий інтуїтивно;
- застосунок не призначений для знаходження інформації з приводу туристичних подорожей.

На основі аналізу вже існуючих програмних рішень, визначення їх основних переваг та недоліків, були виділені певні критерії, а саме:

- стабільність роботи застосунку. Першочерговий критерій у розробці будь якого програмного забезпечення, є повна стабільність роботи, відсутність критичних помилок, працюючі анімації та обробка помилок, які можуть виникнути під час користування;

- безкоштовність. Цей критерій є виключно суб'єктивним, і залишається на вибір розробника або замовника, але наявність даного критерію у програмі, або часткова безкоштовність, з можливістю додаткових функцій за підписку/донат, є явним плюсом;

- актуальність інформації. Даний критерій являє собою ключовий критерій у розробці застосунку теми туризму, тому його наявність є головним аспектом у додатку;

- можливість залишати відгуки – це елемент критерію «Актуальність інформації», тому що найбільш актуальна інформація для користувачів, це враження та думка інших людей з приводу того чи іншого туристичного об'єкту, тому що офіційна інформація на сайті, або у брошурі, може не відображати дійсності;

– зручний та інтуїтивно зрозумілий UI – цей критерій також є одним з ключових при розробці будь якого продукту з користувацьким інтерфейсом, він потрібний для того, щоб під час використання вашого додатку, користувач зміг отримати всю необхідну інформацію швидко та зручно;

– анімація та інтерактивність елементів інтерфейсу - цей критерій менш вагомий ніж зручний UI, але потрібен для того, щоб більше зацікавити користувача користуватися саме вашим додатком, та зробити, та навіть у деяких випадках, визначити що саме з цим елементом інтерфейсу можливо провести взаємодію, що робить UI більше зручним та зрозумілим.

У даному розділі, було проведено аналіз вже існуючих програмних аналогів, та були виділені їхні загальні критерії. Було визначено та обґрунтовано їхню важливість при реалізації програмного продукту. На основі цих критеріїв будуть побудовані функціональні вимоги до програми у технічному завданні.

Критерії наведені порівняльній таблиці критеріїв табл. 1.1.

Таблиця 1.1 – Порівняльна таблиця критеріїв

Критерії	Результати порівняння		
	Google Maps	Opentripmap	Discover.ua
Стабільність роботи	+	+	–
Безкоштовність	+	+	+
Актуальність інформації	+	–	+–
Можливість залишати свої відгуки	+–	–	–
Зручний, та інтуїтивно зрозумілий UI	–	+	–
Анімації та інтерактивність елементів інтерфейсу	–	+	–

1.3 Технічне завдання

1.3.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу бакалавра на тему Розробка вебзастосунку «Туристична онлайн карта», затверджене наказом № 102 від 27 квітня 2022 р. по НУ «Запорізька політехніка».

1.3.2 Мета розробки та її призначення

Метою є розробка веб-застосунку, який призначений для отримання релевантної інформації у сфері туризму, та надає можливість поділитися з іншими користувачами своєю інформацією з приводу туристичних об'єктів.

1.3.3 Основні вимоги до програмного продукту

Як було зазначено у розділі 1.1, вебзастосунок мусить мати декілька типів користувачів, а саме два: зареєстрований користувач та незареєстрований. У різних типів користувачів, будуть відрізнятись функціональні можливості додатку, але зареєстрований користувач, має мати усі функціональні можливості незареєстрованого користувача.

Програмне забезпечення, що розроблюється має надавати користувачеві наступні можливості:

а) для не зареєстрованих користувачів:

- 1) можливість масштабувати розміри та здійснювати переміщення по карті світу завдяки жестам на телефонах, та колесу миші на комп'ютерах;
- 2) можливість перегляду інформації з приводу туристичних об'єктів, яку залишили інші користувачі програми;
- 3) можливість обрати тип архітектурної пам'ятки для перегляду;

- 4) можливість зареєструватися у системі;
- 5) можливість зайти у зареєстрований аккаунт у системі;
- 6) валідація даних системою при реєстрації на предмет існуючих у системі даних схожих логінів та поштових адрес, та валідація даних при вході у систему;
- 7) зберігання у базі даних системи, данні створеного аккаунту користувача;

б) для зареєстрованих користувачів:

- 1) зберігання системою даних логіну користувача у локальному сховищі браузера, завдяки чому користувач не розлогінюється при закритті сайту, або його перезавантаження;
- 2) можливість користувача створити крапку на карті натискання подвійного тапу на мобільних пристроях, та подвійного кліку на миші;
- 3) можливість користувача при створенні крапки вказати заголовок для крапки, опис крапки, рейтинг крапки, тип архітектури крапки;
- 4) можливість зберігання створеної крапки у базу даних системи;
- 5) можливість видалити створену з бази даних системи крапку користувачем, але якщо вона належить до крапок, який створив саме цей користувач.

1.3.4 Вимоги до технічних засобів, операційної системи і мови програмування

Програма, що розробляється у вигляді вебзастосунку, має бути написана на мові JavaScript у середовищі розробки VisualStudioCode. Програма кросплатформена, та підтримує будь-яку операційну систему та пристрій, який підтримує веббраузер. Вимоги до технічних засобів, мають складати вимоги до

операційної системи на якій запускається веббраузер, та вимоги для запуску веббраузера, також вимогою є наявність стабільного інтернет зв'язку.

1.3.5 Вимоги до інформаційної та програмної сумісності

Вебзастосунок що розроблюється, має бути кросбраузерним та кросплатформним, та має мати змогу працювати на будь якому пристрої, які підтримують операційну систему та веббраузер. Програма повинна бути підготовлена до можливої модернізації.

1.3.6 Вимоги до програмної документації

Програмна документація має включати: керівництво програміста, керівництво оператора, опис програми та тексти основних програмних модулів.

1.3.7 Порядок приймання і тестування

Тестування даного ПЗ проводиться у вигляді виконання усіх можливих сценаріїв роботи оператора.

Дане ПЗ вважається прийнятним, якщо воно задовольняє усім висунутим вимогам до нього і успішно пройшло етап тестування.

1.4 Висновки зарозділом1

У даному розділі було досліджено актуальну тему пошуку користувачем релевантної інформації у туристичній галузі, з ціллю побудову плану своєї подорожі. Було визначені втрати туристичної галузі, були виділені елементи системи туристичної галузі такі як: туристичний продукт, суб'єкти туризму, учасник туризму, об'єкт туризму, організатор. Було надано визначення цим елементам системи, визначена їх роль у системи. Були виділені дві головні

задача туристичної галузі, та доведена їх важливість для функціонування туристичної системи, а саме це задачі: надання позитивних враження людині від туристичної поїздки, та надання комфорту перебування під час поїздки. Була розроблена концепція програми, яка буде вирішувати задачі даного дослідження, а також вимоги до наявності типів користувачів у програмі, була доказана актуальність даної вимоги

Було виділено існуючі аналоги веб-застосунків, які вирішують проблему пошуку релевантної інформації у галузі туризму, були визначенні їх недоліки та переваги.

Було розроблене документ технічне завдання, в якому визначені всі функціональні характеристики програми, чітко розділені ролі користувачів у системі, та описані функціональні характеристики програми для кожної з ролей користувачів у системі. Також були встановлені вимоги до технічних засобів, операційної системи і мови програмування, вимоги до інформаційної та програмної сумісності та вимоги до програмної документації.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Моделювання програмного забезпечення

У системі користувачі поділяються на дві ролі – зареєстровані та незареєстровані, ці дві ролі мають схожий функціонал, але у зареєстрованого користувача доступний більш розширений функціонал. Зареєстрованим, користувач вважається якщо він зайшов у свій аккаунт у системі, тим часом незареєстрований користувач вважається користувач якого немає у системі, або який вийшов з системи.

Функціонал незареєстрованого користувача:

- користувач може зареєструвати аккаунт у системі;
- користувач може увійти до аккаунту у системі.

Функціонал зареєстрованого користувача:

- користувач може залишати крапок на мапі з коментарем та оцінкою з приводу того чи іншого туристичного об'єкту;
- користувач може видалити свої точки які він залишив на мапі;
- сесія з аккаунтом користувача залишається і після закриття браузера.

Ключовими процесами у системі є функціональні можливості реєстрації користувача, перегляд інформації з певного туристичного об'єкту, процес створення крапок на мапі веб-застосунку з коментарем та оцінкою того чи іншого туристичного об'єкту, видалення крапок з коментарем на мапі, фільтрація за типом архітектурної пам'ятки для відображення їх на карті.

Опис процесу реєстрації:

- заповнення полів для реєстрації;
- пошук збігів введених даних у системі;
- перевірка на валідність введених даних;
- у разі успішної перевірки системою введених даних, дані хешуються системою та вносяться у базу даних;

– у разі введення у полях не валідних даних, система виведе повідомлення про помилку.

Перегляд інформації з певної крапки на карті, відбувається після натискання лівою кнопкою миші, чи тапу по кнопці на мобільних пристроях.

Опис процесу створення крапок на мапі:

- перевірка на наявність користувача у системі;
- виклик вікна створення крапки;
- заповнення полів заголовку, опису, рейтингу, типу архітектури;
- перевірка на валідність введених даних;
- у разі успіху перевірка системою на валідність введених даних, система зберігає крапку користувача у базу даних;
- у разі введення не валідних даних, система виведе повідомлення про помилку.

Опис видалення крапки з мапи:

- вибір користувачем крапки;
- перевірка системою, на те чи належить дана крапка користувачу;
- у разі успіху перевірки системою на належність крапки до користувача, виведення у вікні з інформацією о крапці, кнопки «Видалити»;
- у разі не належності крапки до користувача, кнопка «Видалити» не виводиться;
- виклик вікна видалення крапки;
- отримання системою id крапки;
- видалення системою за отриманим id крапки за бази даних.

Фільтрація крапок за типом архітектурної пам'ятки, відбувається після обрання користувачем у відповідній секції типу архітектурної пам'ятки, після цього з мапи зникають усі крапки, які не відносяться до обраного типу архітектурної пам'ятки. Для більш чіткого розуміння процесів та сценаріїв програми, було розроблено діаграму прецедентів (рис 2.1), яка показує зв'язаність окремих процесів

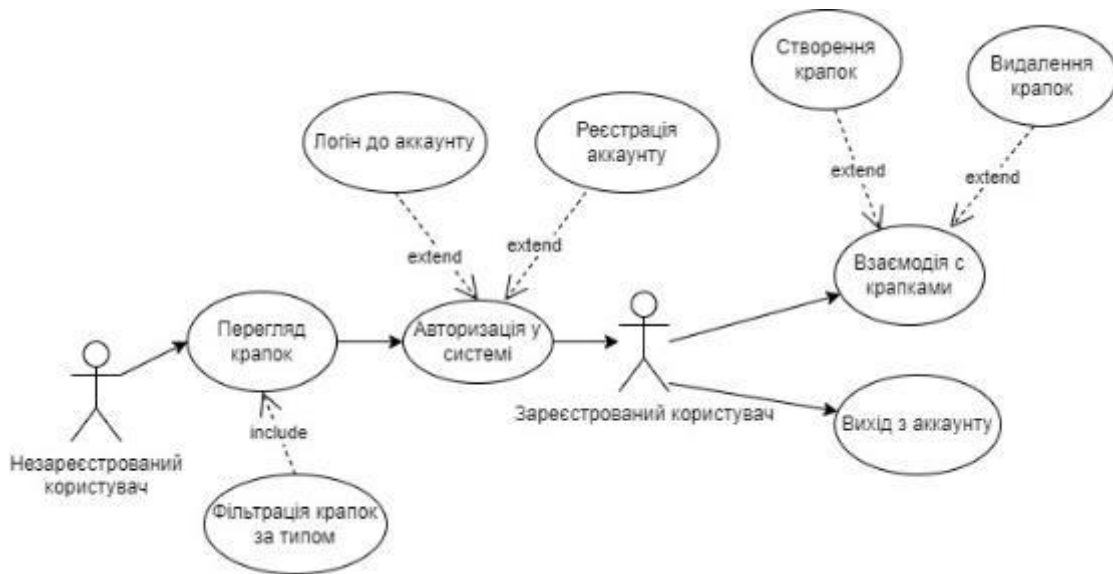


Рисунок 2.1 – Діаграма прецендентів

2.2 Розробка архітектури програмного забезпечення

Вебзастосунок що розроблюється, буде складатися з п'яти основних модулів, а саме:

- модуль користувацького інтерфейсу взаємодії;
- модуль авторизації користувача;
- модуль роботи з крапками на мапі;
- модуль серверної частини;
- модуль бази даних.

Архітектурним патерном для розробки було обрано MVC. Цей патерн використовують при розробці майже усіх сайтів, так як MVC являє собою взаємодію клієнт – сервер – база даних.

Моделлю у даній системі буде слугувати база даних, яка буде надавати дані серверу та змінювати стан даних, коли від сервера будуть приходити запити. Дана частина, взаємодіє лише с сервером, та реагує на запити і змінює стан даних лише від сервера.

Контролером у даній системі буде слугувати сервер, який реагує на запити з клієнтської частини застосунку, та на основі цих запитів робить свої

запити до моделі. Дана частина система взаємодіє і з базою даних, і з клієнтською частиною.

Представлення у даній системі буде слугувати клієнтська частина, а саме модуль авторизації, модуль користувацького інтерфейсу та модуль роботи з точками. Дана частина робить запити до контролеру, для отримання даних або їх стану з моделі

2.2.1 Опис модулів системи

Модуль користувацького інтерфейсу взаємодії – цей модуль забезпечує графічний інтерфейс для керування можливостями програмного інтерфейсу, надає можливість передання іншим компонентам системи даних, необхідних для коректної роботи цих систем. Цей модуль являє собою весь графічний інтерфейс програми, та працює з такими компонентами системи як: модуль авторизації користувача, модуль роботи з крапками на мапі, модуль серверної частини.

Модуль авторизації користувача – цей модуль забезпечує можливість користувача авторизуватись в системі. Цей модуль являє собою частину графічного інтерфейсу програми, який відповідає за авторизацію, та працює з такими компонентами як: модуль роботи з крапками на мапі, модуль серверної частини, модуль користувацького інтерфейсу взаємодії.

Модуль роботи з крапками на мапі – цей модуль забезпечує можливість створення крапок на мапі та їх видалення. Даний модуль доступний лише зареєстрованим користувачам. Цей модуль являє собою частину графічного інтерфейсу, який відповідає за створення та видалення крапок з системи, та працює з такими модулями як: модуль серверної частини, модуль користувацького інтерфейсу взаємодії.

Модуль серверної частини – цей модуль відповідає за обробку даних, які прийшли з усіх інших модулів програми, та віддає у модуль користувацького інтерфейсу взаємодії оброблені дані. Цей модуль являє собою частину

програмного інтерфейсу, який відповідає за обробку та повернення оброблених даних, які прийшли з клієнтської частини, працює з усіма компонентами системи.

Модуль бази даних – цей модуль відповідає за зберігання даних о користувачах та о крапках на мапі, які цей модуль отримує з модулю серверної частини. Цей модуль являє собою частину програмного інтерфейсу, який відповідає за збереження даних о системі, та працює лише з модулем серверної частини.

2.3 Обґрунтування вибору інструментів розробки

2.3.1 Обґрунтування вибору мови програмування

Мовою програмування для розробки веб-застосунку було обрано JavaScript, як єдина мова програмування, інтерпретатор якої підтримують усі браузерні рушії світу. JavaScript на даний момент являє собою не тільки браузерну мову програмування, завдяки платформі NodeJS, є можливість також програмувати і серверну частину застосунку, що дає перевагу у тому, що знаючи одну мову програмування, можливо розробляти як серверну так і клієнтську частину, реалізація клієнтської або серверної частини є тільки у інструментарії, який був обраний для розробки, мається на увазі фреймворки та бібліотеки.

2.3.2 Обґрунтування вибору фреймворків та бібліотек

Для розробки клієнтської частини застосунку, було обрано фреймворк React.js. Цей фреймворк дозволяє ефективно та швидко малювати UI на клієнтській стороні веб-застосунку, завдяки технології VirtualDOM, що перед малюванням UI на сторінці сайту, порівнює внесені зміни у код з звичайним DOM деревом на сторінці, що дозволяє не впливати на DOM безпосередньо. Операції з маніпулюванням DOM затратні за ресурсами та часом, тож ця

технологія дозволить нам оптимізувати роботу нашого веб-застосунка. React.js для створення UI інтерфейсу який буде відображений на сторінці браузеру, використовує технологію JSX, що являє собою набір JavaScript функцій, які надають нам синтаксис ідентичний HTML, що дозволяє використовувати свій досвід у верстці сайтів на HTML, для створення UI інтерфейсу веб-застосунку, але окрім того надає зручний спосіб звернення до API

Окрім фреймворку, необхідні ще додаткові бібліотеки які оптимізують роботу з задачами при розробці, а саме:

- Antd – бібліотека, яка містить у собі вже готові компоненти UI інтерфейсу, які можливо легко кастомізувати завдяки пропсам (вхідні дані функції);
- Axios – бібліотека, яка оптимізує взаємодію між серверною частиною застосунку, та клієнтською;
- MapGLBox – бібліотека, яка малює точну мапу світу у нашому застосунку, та надає API для взаємодії застосунку з мапою.

Для розробки серверної частини застосунку, було обрано фреймворк Express на платформі NodeJS. Express — програмний каркас розробки серверної частини вебзастосунків для NodeJS, реалізований як вільне і відкрите програмне забезпечення. Він спроектований для створення вебзастосунків і API [12]. Express містить у собі всі необхідні функціональні можливості для обробки HTTP запитів, створення серверного порту, роботи з файловою системою будь якої операційної системи.

Окрім Express також необхідні ще додаткові бібліотеки які оптимізують роботу з задачами при розробці, а саме:

- Nodemon – це бібліотека, яка дозволяє виконувати зміни на сервері без перезапуску скрипту з цими змінами у режимі реального часу;
- Mongoose – це бібліотека яка розроблена спеціально під Express, яка надає необхідне API для розробки моделей бази даних MongoDB;
- Bcrypt – це бібліотека яка дозволяє хешувати дані.

2.3.3 Обґрунтування вибору бази даних

Для розробки бази даних, та моделей даних, було обрано базу даних MongoDB. Ця база даних представляє моделі даних не у звичному форматі. Моделі у базі даних, являють собою JavaScript об'єкти, і дані у них зберігаються у форматі «ключ-значення». Це по-перше, дозволяє більш легше розроблювати схеми баз даних, так як між моделями не потрібно будувати зв'язок, і кожна з моделей це окремий об'єкт, який зберігає дані у базі даних у форматі «ключ-значення», і якщо необхідно створити зв'язок, то потрібно лише вказати у полі посилання на іншу модель, з якою необхідно створити зв'язок, по-друге, синтаксис для створення моделей JavaScript-подібний, що дозволяє при знанні лише одно JavaScript розробляти і серверну частину застосунку, і клієнтську, і моделі бази даних. Також MongoDB працює у зв'язку з фреймворком Express, який був обраний для розробки серверної частини застосунку, цей зв'язок забезпечую бібліотека Mongoose, яка надає необхідне API для розробки моделей бази даних MongoDB.

2.3.4 Обґрунтування вибору IDE

Для розробки застосунку, була обрана IDE Microsoft Visual Studio Code. Дана IDE є безкоштовною, доступна широка можливість кастомізації плагінами як для налаштування програмного оточення, наприклад плагін LiveServer, який дозволяє запустити застосунку на сервері для застосування внесених змін в режимі реального часу, так і для кастомізації вигляду самої IDE. Microsoft Visual Studio Code має легке налаштування оточення, можливість створювати код сніпети (одразу заготовлені куски коду), та влаштований термінал, що дозволяє виконувати системні функції операційної системи безпосередньо у самій IDE, наприклад створення React.js налаштованого шаблонного проекту, або встановлення бібліотек та пакетів. Підсумовуючи,

можна сказати що ця IDE не потребує довгого налаштування, і після установки вже готова для використання.

2.4 Висновки за розділом 2

У даному розділі були визначенні функціональні можливості застосунку, та сценарій за якими процеси повинні виконуватися, визначенні ролі користувачів, та функціональні можливості системи для кожної з ролі користувачів, була побудована UML-діаграму прецедентів, для чіткого розуміння процесів та сценаріїв програми.

Були визначені задачі кожного компонента та визначена їх взаємодія між собою, був обраний архітектурний патерн MVC розробки програмного забезпечення.

Були обрані інструменти для розробки програмного забезпечення, а саме була обрана мова програмування JavaScript, були обрані фреймворки для клієнтської частини за стосунку (React.js) та для серверної частини (Express.js), було також обрані бібліотеки для оптимізації робочого процесу над застосунком як для клієнтської і для серверної, було обрано базу даних MongoDB та обрано IDE Microsoft Visual Studio.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка компонентів системи

Згідно з обраною архітектурою MVC, модулі веб-застосунку поділяються на представлення, контролер та модель.

До частини представлення належать модулі, які забезпечують користувача взаємодією з функціоналом інших частин системи, а саме:

- модуль користувацького інтерфейсу взаємодії;
- модуль роботи з крапками;
- модуль авторизації.

До частини контролера належить модуль серверної частини, який виконує обробку даних застосунку, до частини моделі належить модуль бази даних, який зберігає у собі дані до програми.

Модулі поділені на компоненти системи, які містять у собі реалізацію певних функціональних можливостей програмного застосунку.

Опис модулів та їх компонентів наведено у табл. 3.1.

Таблиця 3.1 – Модулі та їх компоненти

Назва модулю	Назва компоненту	Опис компонента
1	2	3
Модуль користувацького інтерфейсу взаємодії	Mappin	Даний компонент системи містить у собі весь графічний інтерфейс застосунку, відображає усі елементи керування у системі, містить у собі функціонал зв'язаний з обробкою виняткових подій, помилок у програмі
	App	Даний компонент є стандартним для усіх додатків які розроблюються за допомогою React.js, реалізується відображення компоненту завантаження сторінки веб-додатку, може містити у собі навігацію по сайту, та інші системні компоненти які не відносяться до графічного інтерфейсу
	Index	Даний компонент містить у собі вузол у додатку, та відмальовує весь веб-застосунок в DOM-дереві

Продовження таблиці 3.1

1	2	3
Модуль авторизації користувача	Auth	Даний компонент містить у собі функціонал логіну користувача до свого аккаунту у системі, відповідає за валідацію даних та перевірку їх наявності у базі даних
	Register	Даний компонент містить у собі функціонал реєстрації аккаунту користувача до системи, перевірку введених користувачем даних, та перевірку цих даних у базі даних на наявність збігів
Модуль серверної частини	Pins	Даний компонент містить у собі функціонал створення у базі даних крапки, видалення крапки, та отримання всіх крапок з бази даних
	Users	Даний компонент містить у собі функціонал реєстрації користувача у системі, та перевірки даних на валідність для авторизації користувача у системі
	Index	Даний компонент містить у собі реалізацію запуску серверу на робочій машині, та реалізацію втілення змін під час розробки серверу у режимі реального часу, без необхідності перезапуску сервера
Модуль роботи з крапками на мапі	Createpin	Даний компонент містить у собі функціонал створення крапок на мапі, компонент приймає дані, які користувач ввів у полях, проводить чи є всі поля заповнені, та чи є користувач зареєстрований, та надсилає усі наявні дані на сервер, для створення крапки
	Deletepin	Даний компонент містить у собі функціонал видалення крапок на мапі, компонент перевіряє, чи належить ця точка до користувача який хоче її видалити, перевіряє чи є користувач зареєстрованим, та приймає ідкрапки і надсилає дані до серверу для видалення крапки
Модуль бази даних	Pin	Даний компонент описує модель крапки у базі даних
	User	Даний компонент описує модель користувача у базі даних

3.1.1 Розробка компонентів контролера

До компонентів контролеру відносять компоненти Pins, Users, та Index.

Index.js є корневим компонентом, містить у собі функцію запуску серверу, та підключення серверу до бази даних ініціалізацію серверу. Код компонента Index.js:

```
const express = require("express");
const mongoose = require("mongoose");
const dotenv = require("dotenv");
```

```
const PINROUTES = require("./routes/pins");
const USERSROUTES = require("./routes/users");

dotenv.config();

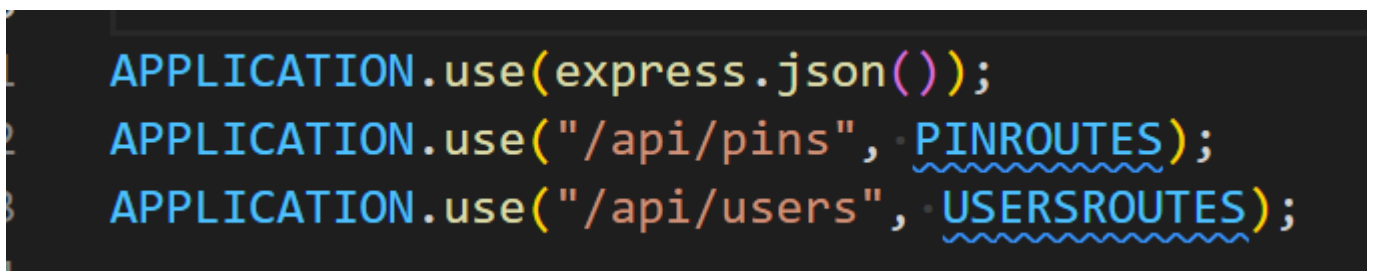
const APPLICATION = express();

APPLICATION.use(express.json());
APPLICATION.use("/api/pins", PINROUTES);
APPLICATION.use("/api/users", USERSROUTES);

mongoose
  .connect(process.env.MONGODB_URL)
  .then(() => console.log("Database Connect"))
  .catch((err) => console.log(err));

APPLICATION.listen(8800, () => {
  console.log("Back start");
});
```

Містить у собі виклик компонентів `Pins`, `Users`, які є обробниками. Обробники – це компоненти які містять у собі логіку обробки даних, які прийшли з клієнтської частини, на URL-адресу за якою знаходиться обробник. Виклик обробників у `Index.js`, та привласнювання URL-адреси відбувається завдяки синтаксису фреймворку Express (рис 3.1).

A screenshot of a code editor with a dark background. It shows three lines of JavaScript code for registering routes in an Express application. The first line is `APPLICATION.use(express.json());`. The second line is `APPLICATION.use("/api/pins", PINROUTES);`. The third line is `APPLICATION.use("/api/users", USERSROUTES);`. The variable names `PINROUTES` and `USERSROUTES` are underlined with blue wavy lines, indicating they are likely imported from another file. Line numbers 1, 2, and 3 are visible on the left side of the code block.

```
1 APPLICATION.use(express.json());
2 APPLICATION.use("/api/pins", PINROUTES);
3 APPLICATION.use("/api/users", USERSROUTES);
```

Рисунок 3.1 – Виклик обробників у `Index.js`

Компонент Pins.js містить у собі реалізацію обробника даних крапки, який знаходиться за URLадресою `/api/pins`. Містить у собі реалізацію функціоналу отримання усіх крапок, видалення та створення крапок. Приймає на вхід дані з модулю роботи з крапками з клієнту в форматі JSON, викликає функцію запиту до бази даних, та у випадку валідного запиту відправляє на клієнтську частину змінені дані. Функції для запиту до бази даних:

- Save – зберігає дані у базі даних;
- Find – знаходить всі документи у базі даних які відповідають заданому критерію;
- FindOne - знаходить один документ у базі даних який відповідає заданому критерію;
- Remove – видаляє документ з бази даних.

Реалізація компонента Pins.js:

```
const ROUTER = require("express").Router();
const PIN = require("../models/Pin");
```

```
//createpin
```

```
ROUTER.post("/", async (req, res) => {
  const NEWPIN = new PIN(req.body);
  try {
    const SAVEPIN = await NEWPIN.save();
    res.status(200).json(SAVEPIN);
  } catch (err) {
    res.status(500).json(err);
  }
});
```

```
//getallpin
```

```
ROUTER.get("/", async (req, res) => {
```

```

try {
  const PINS = awaitPIN.find();
  res.status(200).json(PINS);
} catch (err) {
  res.status(500).json(err);
}
});

//deletepin

ROUTER.delete("/delete/", async (req, res) => {
  try {
    const DELETEDPIN = awaitPIN.findOne({ _id: req.body._id });
    if (!DELETEDPIN) {
      return res.status(400).json({ message: "Pinnotfound" });
    }
    awaitDELETEDPIN.remove();
    res.status(200).json(DELETEDPIN);
  } catch (err) {
    res.status(500).json(err);
  }
});

module.exports = ROUTER;

```

Компонент Users.js містить у собі реалізацію обробника даних користувача, який знаходиться за URL адресою /api/users. Містить у собі реалізацію функціоналу реєстрації користувача, логіну користувача. Приймає на вхіддані модулі реєстрації користувачів з клієнту в форматі JSON, викликає функцію запити да бази даних даних, та у випадку валідного запити відправляє на клієнтську частину змінені дані. Функції для запити до бази даних використовуються ті ж самі що і у компоненті Pins.js для збереження даних у

базі даних користувачів(Save), та для логіну, система робить запит для знаходження конкретного користувача (FindOne).

Обробник реєстрації приймає об'єкт з даними(логін,email, пароль) у заголовку запиту з клієнту, та перед внесенням даних у базу даних, хешує пароль за допомогою функції hashбібліотеки Bcrypt, яка являє собою реалізацію алгоритму хешування Bcrypt -адаптивна криптографічна хеш-функція формування ключа, що використовується для захищеного зберігання паролів. Функція заснована на шифрі Blowfish [13], для захисту від атак за допомогою райдужних таблиць bcrypt, крім того, функція є адаптивною, час роботи легко налаштовується і її можна уповільнити, щоб ускладнити атаку перебором, задаючи значення від 8 до 31 [14].

Обробник логіну приймає запит на спробу входу користувача до системи з клієнтської частини, та робить запит до бази даних для знайдення схожого логін,email, також приймає пароль, який функція compare, яка приймає у себе пароль який потрібно дешифрувати, та пароль з яким потрібно порівняти, дешифрує пароль та перевіряє на валідність, тобто наявність паролю у системі.

Реалізація компоненту Users.js:

```
const ROUTER = require("express").Router();
const USERS = require("../models/User");
const BCRYPT = require("bcrypt");

ROUTER.post("/register", async (req, res) => {
  try {
    //generate password
    const SALT = await BCRYPT.genSalt(10);
    const HASHPASSWORD = await BCRYPT.hash(req.body.password,
SALT);

    //create new user
    const NEWUSER = new USERS({
```

```

        username: req.body.username,
        email: req.body.email,
        password: HASHPASSWORD,
    });

    //save user and send response
    const USER = await NEWUSER.save();
    res.status(200).json(USER._id); //it makes no sense to save the
entire user, just his id is enough
    } catch (err) {
    res.status(500).json(err);
    }
    });

ROUTER.post("/login", async (req, res) => {
    try {
        //find user and email
        const USER = await USERS.findOne({ username: req.body.username
    });

        const EMAIL = await USERS.findOne({ email: req.body.email });

        //validate password
        const VALIDATE_PASSWORD = await BCRYPT.compare(
req.body.password,
USER.password
        );

        if (!USER || !EMAIL || (!VALIDATE_PASSWORD &&res.status(400)))
    {
        return res.status(400).json("Wrong username or email or
password");
    } else {
        //send response

```

```

        return res.status(200).json({ _id: USER._id, username:
USER.username });
    }
    } catch (err) {
        res.status(500).json(err);
    }
});

module.exports = ROUTER;

```

Контролер містить у собі головний компонент `Index.js`, який приймає запити на URLадресу обробників `Pins.js` та `User.js`, та після того як обробники виконають логіку по додаванню даних до бази даних, відає відповідь на клієнтську частину.

Для більш кращого представлення взаємодії компонентів контролеру, було розроблено діаграму компонентів (рис 3.2). На діаграмі показаний інтерфейс взаємодії між компонентами контролеру, а саме дані які приймають Pins.js та User.js, та передачу результату оброблених даних у головний компонент контролеру Index.js.

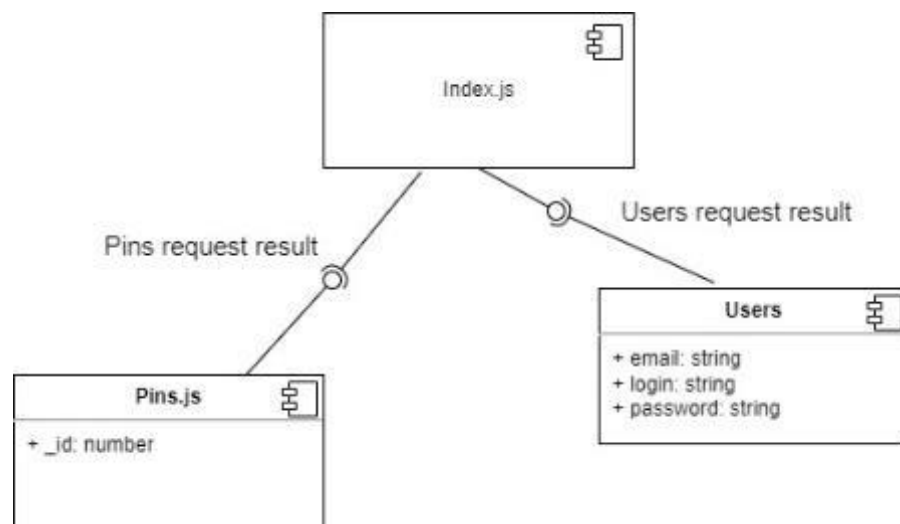


Рисунок 3.2 – Діаграма компонентів контролеру

3.1.2 Розробка компонентів моделі

До компонентів моделі, відносяться Pin.js та User.js.

Pin.js являє собою JavaScript об'єкт з даними о крапках на мапі, всі поля у якому є обов'язкові. Містить у собі такі поля:

а) Username – містить у собі ім'я користувача, якому належить ця крапка;

б) _id – унікальний ідентифікатор, який генерується MongoDB автоматично для кожного об'єкта моделі;

в) Title – містить у собі заголовок крапки;

г) Description – містить у собі опис крапки;

д) Type – містить у собі архітектурний тип крапки, має діапазон вхідних даних від 1 до 5 де:

1) архітектурний тип;

2) кулінарний тип;

3) спортивний тип;

4) історичний тип;

5) інше. Під типом інше мається на увазі що це будь-який тип якого нема у переліку типів;

е) Rating – містить у собі рейтинг крапки, має діапазон вхідних даних від 1 до 5;

є) Lat – містить у собі географічну координату ширини міста, на якому створена точка;

ж) Long – містить у собі географічну координату довготи міста, на якому створена точка.

User.js являє собою JavaScript об'єкт з авторизаційними даними користувача, всі поля у якому є обов'язкові. Містить у собі такі поля:

– username – містить у собі ім'я користувача;

– email – містить у собі поштову скриньку користувача;

– password – містить у собі пароль користувача;

Діаграму розроблених компонентів моделі, наведено на рис 3.3. На діаграмі показані поля, які у собі містять моделі Pin та User.

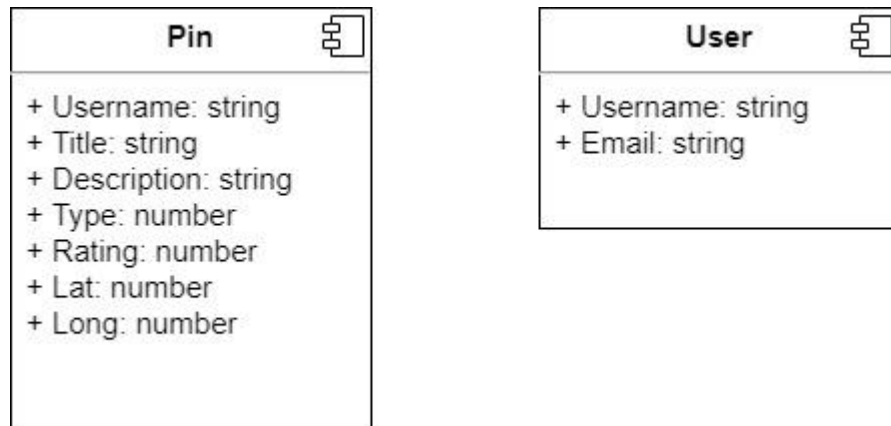


Рисунок 3.3 – Діаграма компонентів моделі

3.1.3 Розробка компонентів представлення

До компонентів представлення відноситься Index.js, App.js, Mappin.js, DeleteBtn.js, DeleteModal.js, CreatePin.js, Auth.js, Register.js.

Index.js являє собою корневий компонент застосунку, містить у собі кореневий вузол, який відповідає за рендер всіх компонентів додатку у VirtualDOM. Код компоненту Index.js:

```

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
ReactDOM.render(
  <React.StrictMode>
  <App />
</React.StrictMode>,
  document.getElementById('root')
)
  
```

Містить виклик у собі компонента App.js, якій являє собою компонент для відмальовкимапи світуMapPin.js, та містить у собі системний компонент Suspense, який відповідає за відмальовку запасного UI, у випадку коли сторінка у стані загрузки. Код компонента App.js:

```
import React, { Suspense } from 'react';

import MapPin from './component/map/mappin';

const App = () => {
  return (
    <Suspense fallback={<div>Loading...</div>}>
    <MapPin />
    </Suspense>
  );
};

export default App;
```

3.2 Розробка програмного інтерфейсу

MapPin.js являє собою компонент UI інтерфейсу, який містить у собі відмальовку компонентів DeleteBtn.js, CreatePin.js, Auth.js, Register.js, та містить відмальовку мапи світу та всіх інтерфейсних елементів на мапі. За допомогою бібліотеки Axios, який надає застосунку можливість робити запити до серверної частини застосунку, отримує дані користувачів, та дані крапок, та розміщує дані по змінам у компоненті. У якості змінних, у програмі використовується нативні можливості React.js – веб-хуки, а саме хук useState. Цей хук представляє собою масив з двох елементів, перший елемент є стан змінної, другий елемент функція, яка цей стан змінює, у офіційній документації React.js, перший елемент прийнято називати іменником, а другий елемент який

є функцією, прийнято іменувати також само, але перед цим дописати Set (рис. 3.4). Серед змінних UseState, які містять дані з серверу відносяться:

- Title–містить у собі інформацію з приводу заголовку крапки, є типом string, початкове значення Null;
- Description–містить у собі інформацію з приводу опису крапки, є типом string, початкове значення Null;
- Type–містить у собі інформацію з приводу архітектурного типу крапки, є типом number, початкове значення число 1, що відповідає у системі архітектурній пам'ятці типу «Архітектурна споруда» ;
- Rating–містить у собі інформацію з приводу рейтингу крапки, є типом number, початкове значення число 1;
- Pins–містить у собі масив крапок на мапі, початкове значення пустий масив;
- NewPlacePin–містить у собі координати крапки, яка створюється, є типом number, початкове Null;
- CurrentPlaceID –містить у собі id крапки, на яку натиснули, є типом number, початкове Null;
- CurrentUser–містить у собі інформацію про сесію зареєстрованого користувача, є типом string, початкове значення приймає у себе строку токена з браузерного сховища LocalStorage.

```
const [title, setTitle] = useState(null);
```

Рисунок 3.4 – Приклад найменування змінних хука UseState

У Mappin.js також є зміні UseState, які відповідають за зміну станів UI інтерфейсу, а саме:

- SelectedOptions–містить у собі тип крапок, який потрібно відобразити на мапі;

- `SelectedType`—містить у собі обраний користувачем тип фільтру крапки, який обрав користувач;
- `ShowRegister`—містить у собі стан відкриття модального вікна реєстрації, є типом `boolean`, початкове значення `false`;
- `ShowLogin`—містить у собі стан відкриття модального вікна логіну, є типом `boolean`, початкове значення `false`;
- `OpenDeleteModal`—містить у собі стан відкриття модального вікна видалення крапки, є типом `boolean`, початкове значення `false`.

Окрім змінних `useState`, у `Mappin.js` є звичайні константи, які зберігають у собі статичну інформацію, а саме змінна `Options`, яка зберігає список фільтрів крапок за типом, та змінна `Storage`, яка отримує токен сесії користувача з локального сховища у браузері.

Весь функціонал взаємодії з елементами користувача, реалізований у функціях хендлерів — функціях які оброблюють введені дані, якщо був викликаний зовнішній тригер, тобто натискання кнопки, введення тексту у `input`, та інше. У `Mappin.js` реалізовані такі функції-хендлери:

- `HandlerMarkerClick`—відстежує клік по мапі, та записує у `SetCurrentPlaceId` крапки на яку натиснули;
- `HandleAddPopUpClick`—відстежує подвійний клік по мапі, та викликає модальне вікно створення крапки, записує у `SetNewPlacePin` координати широти та довготи де було викликано модальне вікно створення;
- `HandleSubmit`—відстежує підтвердження натискання на кнопку «Створити» у модальному вікні створення крапки, записує у `NewPin` об'єкт за даними які користувач ввів, та робить запит на сервер, для внесення даних крапки у базу даних;
- `HandleLogout`—відстежує натискання кнопки `Logout`, та видаляє дані токена з локального сховища у браузері, та записує у `SetCurrentUser`;
- `HandleChange`—відстежує обраний фільтр архітектурної споруди, та записує у `SelectedType` номер архітектурний споруди, від 1 до 5.

Перед тим, як компонент Mappin.js відмалюється на екрані пристрою користувача, викликається React.jsхук UseEffects. Він робить запит на сервер перед відмалюванням компонента Mappin.js на екрані, та отримує усі крапки які є у базі даних, та інформацію о цих крапках, та за допомогою функції SetPins, записує всі дані у масив крапок Pins.

Компонент DeleteBtn.js являє собою кнопку видалення крапки, та приймає у себе аргументи з Mappin.js, а саме:

- setIsOpenDeleteModal– функція яка визначає стан змінної OpenDeleteModal, яка відповідає за те відкрите модальне вікно видалення крапки;
- Pin–ідкрапки, у якої було визвано модальне вікно з інформацією;
- OpenDeleteModal–функція, яка зберігає стан чи відкрите модальне вікно.

Даний компонент реалізує у собі логіку виклику модального вікна видалення крапки, та містить у собі компонент DeleteModal.js.

Код компонента DeleteBtn.js:

```
import React, { useState } from 'react';
import DeleteModal from './DeleteModal';

import style from './style.module.css';

const DeleteBtn = ({ setIsOpenDeleteModal, pin, openDeleteModal })
=> {
  return (
    <div>
      <p className={style.deleteBtn} onClick={()
=>setIsOpenDeleteModal(true)}>
        Delete
      </p>
      {openDeleteModal ? (
```

```

        <DeleteModal setIsOpenDeleteModal={setIsOpenDeleteModal}
        pinId={pin} />
      ) : null}
    </div>
  );
};

export default DeleteBtn;

```

Компонент DeleteModal.js, являє собою модальне вікно видалення крапки, та приймає у себе аргументи з DeleteBtn.js, а саме:

- SetIsOpenDeleteModal – функція яка визначає стан змінної OpenDeleteModal, яка відповідає за те відкрите модальне вікно видалення крапки.

Компонент DeleteModal.js, являє собою модальне вікно видалення крапки, та приймає у себе аргументи з DeleteBtn.js, а саме:

- SetIsOpenDeleteModal – функція яка визначає стан змінної OpenDeleteModal, яка відповідає за те відкрите модальне вікно видалення крапки;

- Pin – id крапки, у якої було визвано модальне вікно з інформацією.

Функціонал видалення крапки, реалізований у функцій-хендлері HandleDeleteSubmit, яка відстежує натискання на кнопку «Видалити», приймає id крапки, робить запит до серверу на видалення документу крапки за переданим id, та після успішного видалення перезавантажує сторінку. Якщо було натиснуто кнопку «Закрити», то модальне вікно закривається, і функція видалення не спрацьовує.

Код компоненту DeleteModal.js:

```

import React from 'react';
import style from './modalDelete.module.css';
import axios from 'axios';

```

```

constDeleteModal = ({ setIsOpenDeleteModal, pinId }) => {
  consthandleDeleteSubmit = async (e) => {
    e.preventDefault();
    try {
      awaitaxios
        .delete(`/pins/delete/`, { data: { _id: pinId } })
        .then(setIsOpenDeleteModal(false))
        .then(document.location.reload());
    } catch (err) {
      console.log(err);
    }
  };
  return (
    <div className={style.modalBackground}>
      <div className={style.modalContainer}>
        <div className={style.modalCloseBtn}>
          <buttononClick={() =>setIsOpenDeleteModal(false)}> X </button>
        </div>
        <div className={style.modalTitle}>
          <h2 style={{ fontWeight: '900' }}>DeletePin</h2>
        </div>
        <div className={style.modalBody}>
          <p>Doyousure, youwan`tdeletepin?</p>
        </div>
        <div className={style.modalFooter}>
          <buttononClick={() =>setIsOpenDeleteModal(false)}>Cancel</button>
          <buttononClick={handleDeleteSubmit}>Delete</button>
        </div>
      </div>
    </div>
  );
};

```

```
export default DeleteModal;
```

Компонент CreatePin.js, являє собою модальне вікно для створення крапки, та приймає у себе аргументи з MapPin.js, а саме:

- setTitle;
- setDescription;
- setType;
- setRating;
- newPlacePin;
- setNewPlacePin;
- handleSubmit.

Функціонал створення крапки, реалізований у функцій-хендлері handleSubmit, яка відстежує натискання на кнопку «Створити», , робить запит до серверу на створення документу крапки за даними які вказав користувач, та після успішного створення крапка заноситься до масиву Pins.

Код компоненту CreatePin.js:

```
import style from './style.module.css';
import { Popup } from 'react-map-gl';
import { Close } from '@material-ui/icons';

const CreatePin = ({ newPlacePin, setNewPlacePin, handleSubmit, setTitle, setDescription, setType, setRating }) => {
  return (
    <Popup
      longitude={newPlacePin.lng}
      latitude={newPlacePin.lat}
      anchor="left"
      closeOnClick={false}
      onClose={() => setNewPlacePin(null)}
    >
    <div>
```

```

<formclassName={style.card} onSubmit={handleSubmit}>
  <label>Place</label>
  <input
    placeholder="Title"
    onChange={(e) =>setTitle(e.target.value)}
  />
  <label>Description</label>
  <textarea
    placeholder="Plese, writeyou`redescription"
    onChange={(e) =>setDecription(e.target.value)}
  />
  <label>Type</label>
  <selectonChange={(e) =>setType(e.target.value)}>
    <optionvalue="1">Architectural</option>
    <optionvalue="2">Culinary</option>
    <optionvalue="3">Sports</option>
    <optionvalue="4">Historical</option>
    <optionvalue="5">Other</option>
  </select>
  <label>Rating</label>
  <selectonChange={(e) =>setRating(e.target.value)}>
    <optionvalue="1">1</option>
    <optionvalue="2">2</option>
    <optionvalue="3">3</option>
    <optionvalue="4">4</option>
    <optionvalue="5">5</option>
  </select>
  <buttonclassName={style.Create__Pin} type="submit">
    AddPin
  </button>
  <Close
    className={style.closePopUp}
    onClick={() =>setNewPlacePin(null)}
  />

```

```

</form>
</div>
</Popup>
    );
};

export default CreatePin;

```

Компонент Auth.js, являє собою модальне вікно логіну користувача, та приймає у себе аргументи з MapPin.js, а саме:

- Storage;
- SetCurrentUser.

Компонент Auth.js має у собі змінну UseState, а саме, Fail – містить у собі стан помилки невірно вказаних даних у процесі логіну, якщо має значення true, відображає у модальному вікні текст помилки, є типом boolean, початкове значення false.

Також має виклик хуку UseRef, змінні цього хуку, містять у собі значення яке записано у середину DOM-елементу, та дозволяє маніпулювати змінною їх станом, без перемальовування DOM-елемента. До цих змінних відносяться UserNameRef, EmailRef, PasswordRef, які містять значення відповідних полів у формі реєстрації.

Функціонал логіну користувача, реалізований у функції-хендлеру HandleSubmit, яка відстежує натискання на кнопку «Увійти», створює у середині об'єкт User, та записує у об'єкт дані усіх змінних UseRef, після чого робить запит на сервер, де іде перевірка валідності даних, які ввів користувач, якщо дані валідні, сервер відправляє токен користувача, де в змінну Storage записується цей токен, функція SetCurrentUser записує у себе цей логін користувача, та модальне вікно закривається, якщо дані не валідні, система виводить у вікно повідомлення про помилку.

Код компоненту Auth.js:

```

import { Cancel } from '@material-ui/icons';
import React, { useRef, useState } from 'react';

import AuthIcon from '../assets/auth.png';
import axios from 'axios';

import style from './auth.module.css';

const Auth = ({ setShowLogin, storage, setCurrentUser }) => {
  const [fail, setFail] = useState(false);
  const usernameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();

  const handleSubmit = async (e) => {
    e.preventDefault();
    const user = {
      username: usernameRef.current.value,
      email: emailRef.current.value,
      password: passwordRef.current.value,
    };

    try {
      const response = await axios.post('/users/login', user);
      storage.setItem('token', response.data.username);
      setCurrentUser(response.data.username);
      setShowLogin(false);
      setFail(false);
    } catch (err) {
      setFail(true);
    }
  };

  return (

```

```

    <div className={style.auth__container}>
      <form onSubmit={handleSubmit}>
        <div className={style.logo}>
          <img src={AuthIcon} alt="register" />
          SignInForm
        </div>
        <input type="text" placeholder="Username" ref={userNameRef} />
        <input type="email" placeholder="Email" ref={emailRef} />
        <input type="password" placeholder="Password" ref={passwordRef} />
        <button>SignIn</button>

        {fail && <span className={style.failure}>Something wrong</span>}

      </form>
      <Cancel
        className={style.auth_cancel}
        onClick={() => setShowLogin(false)}
      />
    </div>
  );
};

export default Auth;

```

Компонент Register.js являє собою модальне вікно реєстрації користувача.

Компонент має у собі ідентичний за призначенням змінні як і у Auth.js, а саме UserNameRef, EmailRef, PasswordRef, та Fail, також містить змінну useStateSuccess –містить у собі стан успішного процесі реєстрації, якщо має значення true, відображає у модальному вікні текст про успішну реєстрацію, є типом boolean, початкове значення false.

Функціонал реєстрації користувача, реалізований у функції-хендлеру handleSubmit, яка відстежує натискання на кнопку «Зареєструватись», створює у середині об'єкт User, та записує у об'єкт дані усіх змінних useRef,

після чого робить запит на сервер, де іде перевірка валідності даних, які ввів користувач, якщо дані валідні, то сервер створює документ користувача у базі даних, та SetSuccess записує у себе true, і у модальному вікні виводиться повідомлення про успішну реєстрацію, у випадку якщо дані введені користувачем не валідні, система виводить повідомлення про невдалу реєстрацію.

Код компонента Register.js:

```
import React, { useRef, useState } from 'react';
import axios from 'axios';

import RegIcon from '../assets/register.png';

import style from './register.module.css';
import { Cancel } from '@material-ui/icons';

const Register = ({ setShowRegister }) => {
  const [success, setSuccess] = useState(false);
  const [fail, setFail] = useState(false);
  const userNameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();

  const handleSubmit = async (e) => {
    e.preventDefault();
    const newUser = {
      username: userNameRef.current.value,
      email: emailRef.current.value,
      password: passwordRef.current.value,
    };

    try {
      await axios.post('/users/register', newUser);
```

```

    setFail(false);
    setSuccess(true);
    } catch (err) {
    setFail(true);
    }
  };

  return (
    <div className={style.register__container}>
      <form onSubmit={handleSubmit}>
        <div className={style.logo}>
          <img src={RegIcon} alt="register" />
          RegisterForm
        </div>
        <input type="text" placeholder="Username" ref={userNameRef} />
        <input type="email" placeholder="Email" ref={emailRef} />
        <input type="password" placeholder="Password" ref={passwordRef} />
        <button>Register</button>
        {success && (
          <span className={style.successfull}>Registersuccessfull</span>
        )}
        {fail && <span className={style.failure}>Somethingwrong</span>}
      </form>
      <Cancel
        className={style.register_cancel}
        onClick={() => setShowRegister(false)}
      />
    </div>
  );
};

export default Register;

```

Представлення являє собою батьківський компонент Mappin.js який містить у собі дочірні UIкомпоненти, такі як наприклад модальне вікно реєстрації. Mappin.js містить у собі дані, які передаються дочірнім компонентам для обробки, у той час як дочірні компоненти оброблюють дані надані Mappin.js, та відправляють оброблені дані на сервер. Для більш кращого уявлення взаємодії компонентів між собою, було розроблено діаграму компонентів представлення (рис 3.5). На діаграмі зображено ієрархію компонентів представлення, які дані у собі містять компоненти, та які дані вони надають дочірнім компонентам.

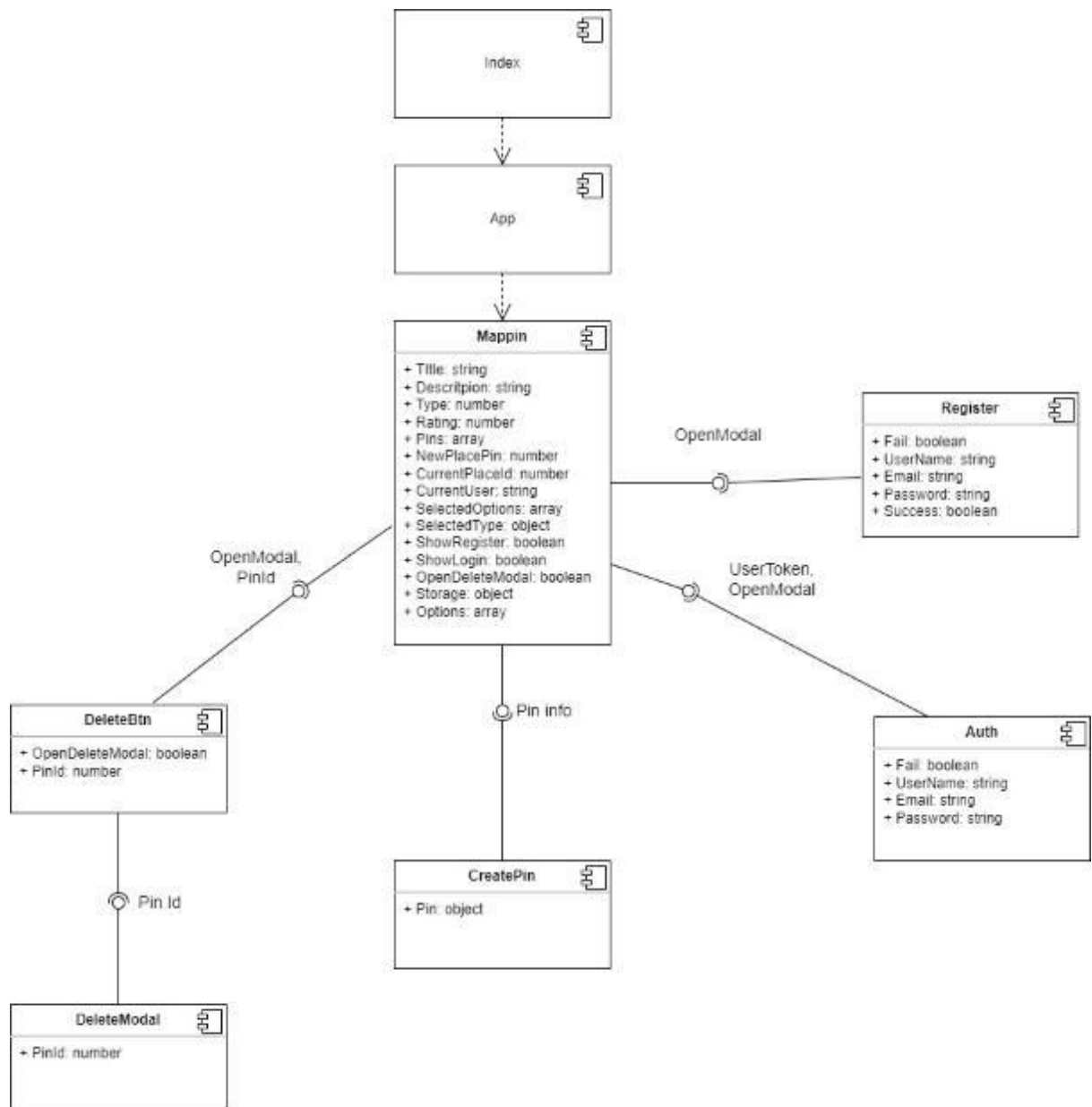


Рисунок 3.5 – Діаграма компонентів представлення

3.2 Розробка програмного інтерфейсу

Програмний інтерфейс застосунку був розроблений за допомогою фреймворку React.js, та технологій module.css, яка генерує для селекторів унікальні імена класів, що дозволяє більш предметно називати класи JSX-елементів, без можливості того, що назви класів будуть повторюватись, що у випадку звичайного CSS призведе до копіювання стилів.

У додатку одна головна сторінка, яка містить у собі мапу світу, та на якій знаходяться елементи керування інтерфейсом. Головна сторінка зображена на рис 3.6.



Рисунок 3.6 – Головна сторінка застосунку

Реалізацією мапи є бібліотека Mapbox [15] з відкритим вихідним кодом, яка надає компонент з вже готовою мапою світу, у якій початково реалізований функціонал переміщення по мапі, та можливість приближати та віддаляти мапу.

Реалізація функціоналу взаємодії з мапою міститься у файлі MapPin.js, а саме:

– функціонал відображення крапок на мапі. Умова відображення працює так: Якщо тип крапки дорівнює обрану типу у фільтрі архітектурних типів, то відобразити крапку, якщо не дорівнює, не відобразити нічого;

– визначення кольору крапки в залежності від рейтингу якщо користувача не зареєстрований, або визначення кольору крапок які належать або не належать користувачу. Умова визначення кольору працює так: Якщо рейтинг крапки дорівнює числу від 1 до 5 і не дорівнює зареєстрованому користувачу, то крапки повинні мати колір який відносить до її рейтингу, у іншому випадку крапки які належать користувачу який зареєструвався у системі повинні мати червоний колір, всі інші синій.

Код відображення крапок на мапі:

```
{pin.type === selectType ? (
  <Marker
    longitude={pin.long}
    latitude={pin.lat}
    key={pin.type}
    anchor="bottom"
  >
    <PersonPinCircle
      style={{
        fontSize: '38',
        onClick={() =>handleMarkerClick(pin._id)}
      }}
    />
  </Marker>
) : null}
```

Код визначення кольорів:

```
pin.rating === 1 && !currentUser
? '#FF1800'
: pin.rating === 2 && !currentUser
```

```

? '#FF8700'
: pin.rating === 3 && !currentUser
? '#1DD300'
: pin.rating === 4 && !currentUser
? '#05819E'
: pin.rating === 5 && !currentUser
? '#7F07A9'
: pin.username === currentUser
? 'red'
: 'blue',

```

У верхньому лівому куті розташований фільтр точок за типом архітектури, являє собою випадаюче меню, елементи якого призначені для обрання типу архітектурної крапки, яка потрібна користувача. Випадаюче меню зображено на рис 3.7.

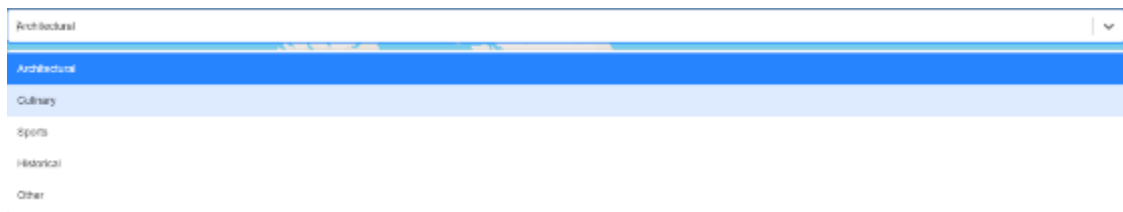
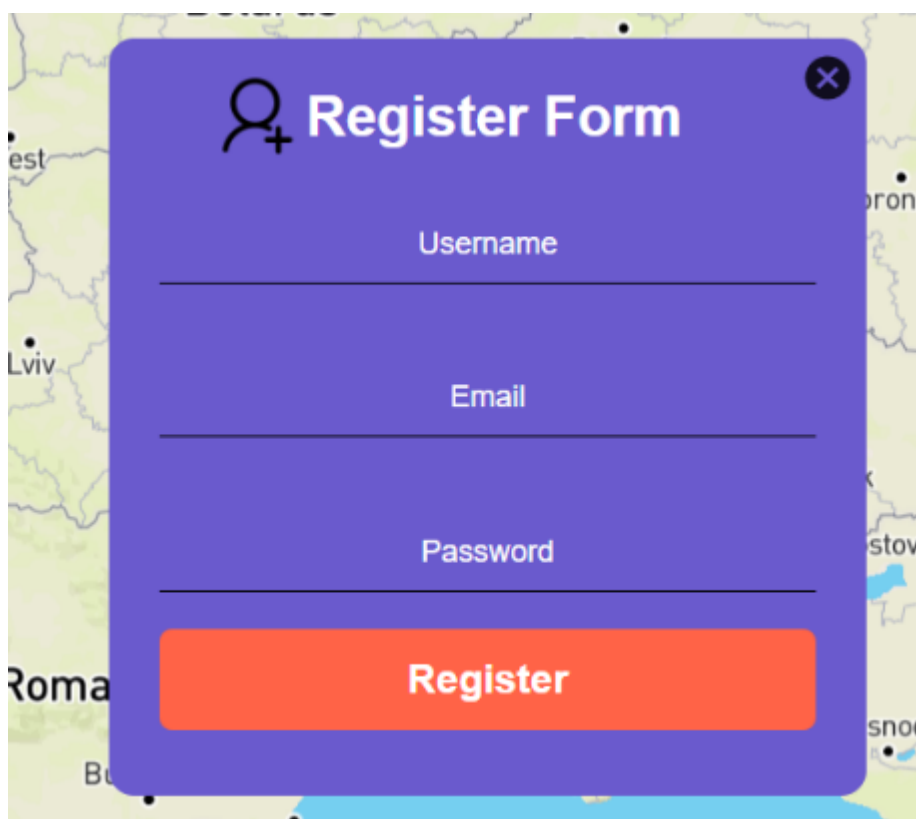


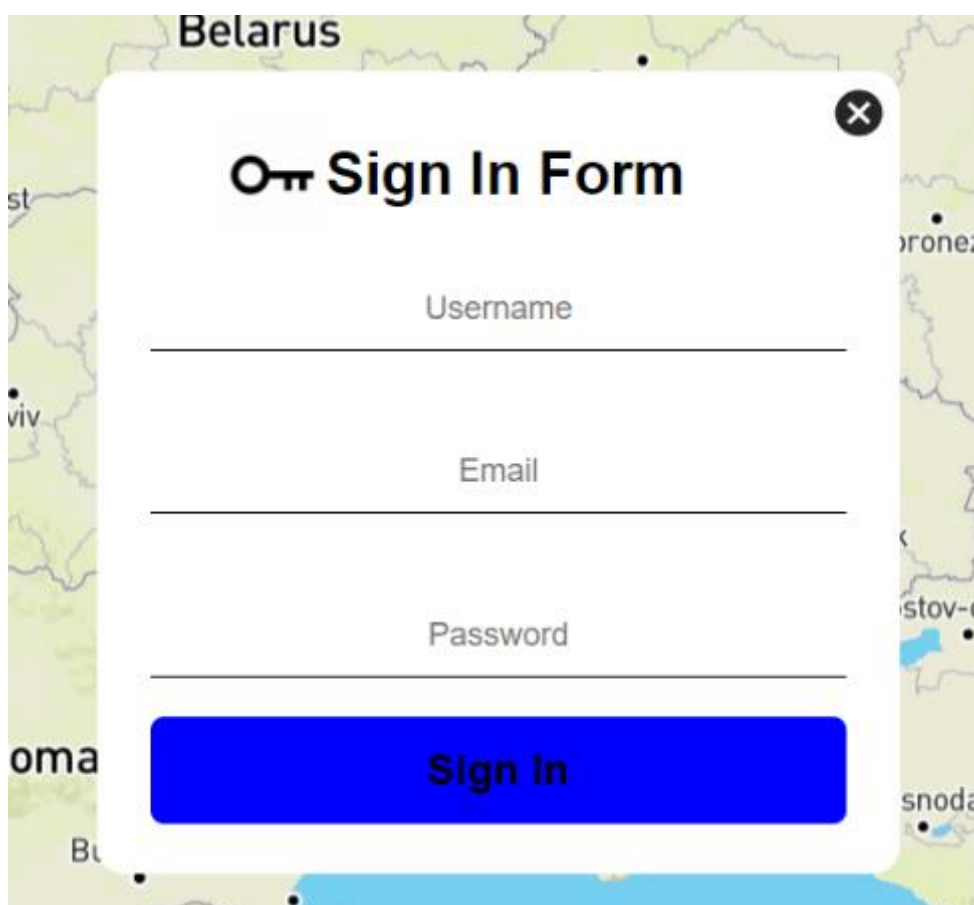
Рисунок 3.7 – Випадаюче меню фільтрації

У верхньому правому куті розташовані форми реєстрації (рис 3.8) та логіну (рис 3.9) користувача що містяться у файлах Auth.js та Register.js, являють собою кнопки, при натисканні на які з'являються модальні вікна. Кнопки з формами зображені на рис 3.10.



The image shows a purple rectangular overlay titled "Register Form" with a person icon and a close button (X) in the top right corner. It contains three input fields labeled "Username", "Email", and "Password", each with a horizontal line for text entry. At the bottom is a large orange button labeled "Register". The background is a map showing parts of Eastern Europe, including labels like "Lviv", "Roma", "Bu", "stov-", and "snod".

Рисунок 3.8 – Форма реєстрації



The image shows a white rectangular overlay titled "Sign In Form" with a key icon and a close button (X) in the top right corner. It contains three input fields labeled "Username", "Email", and "Password", each with a horizontal line for text entry. At the bottom is a large blue button labeled "Sign In". The background is a map showing parts of Eastern Europe, including labels like "Belarus", "st", "vive", "oma", "Bu", "stov-", and "snoda".

Рисунок 3.9 – Форма логіну

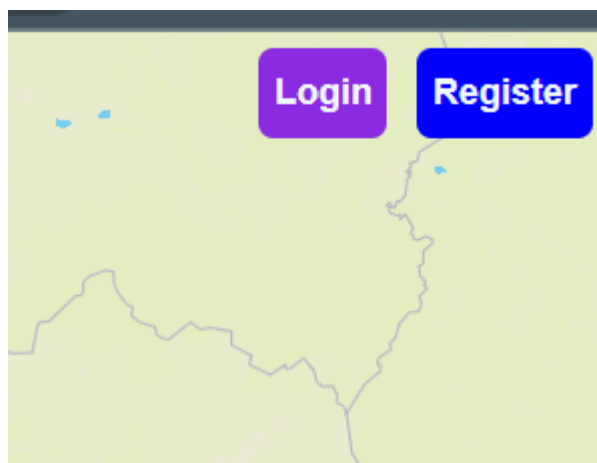


Рисунок 3.10 – Кнопки форм реєстрації та логіну

На мапі розташовані крапки (рис 3.11) створені користувачами, при натисканні на які викликається модульне вікно з інформацією (рис 3.12). Крапки візуально відрізняються за кольором (рис 3.13) який прив'язаний до рейтингу крапки, який обирає користувач при створенні, і система надає колір крапки у в залежності від рейтингу а саме: 1 – червоний, 2 – помаранчевий, 3 – зелений, 4 – бірюзовий, 5 – фіолетовий. Така залежність працює лише у випадку коли користувач ще не зайшов у систем, у іншому випадку система відображає крапки користувача червоним кольором, крапки які не належать синім (рис 3.14). Крапки на карті відображаються в залежності від типу який був обраний при фільтрації, тому якщо було обрано Історичний тип пам'ятки, то на карті буде відображатися крапки лише з Історичним типом пам'ятки, як показано на рис 3.15 – 3.16.



Рисунок 3.11 – Крапки на мапі

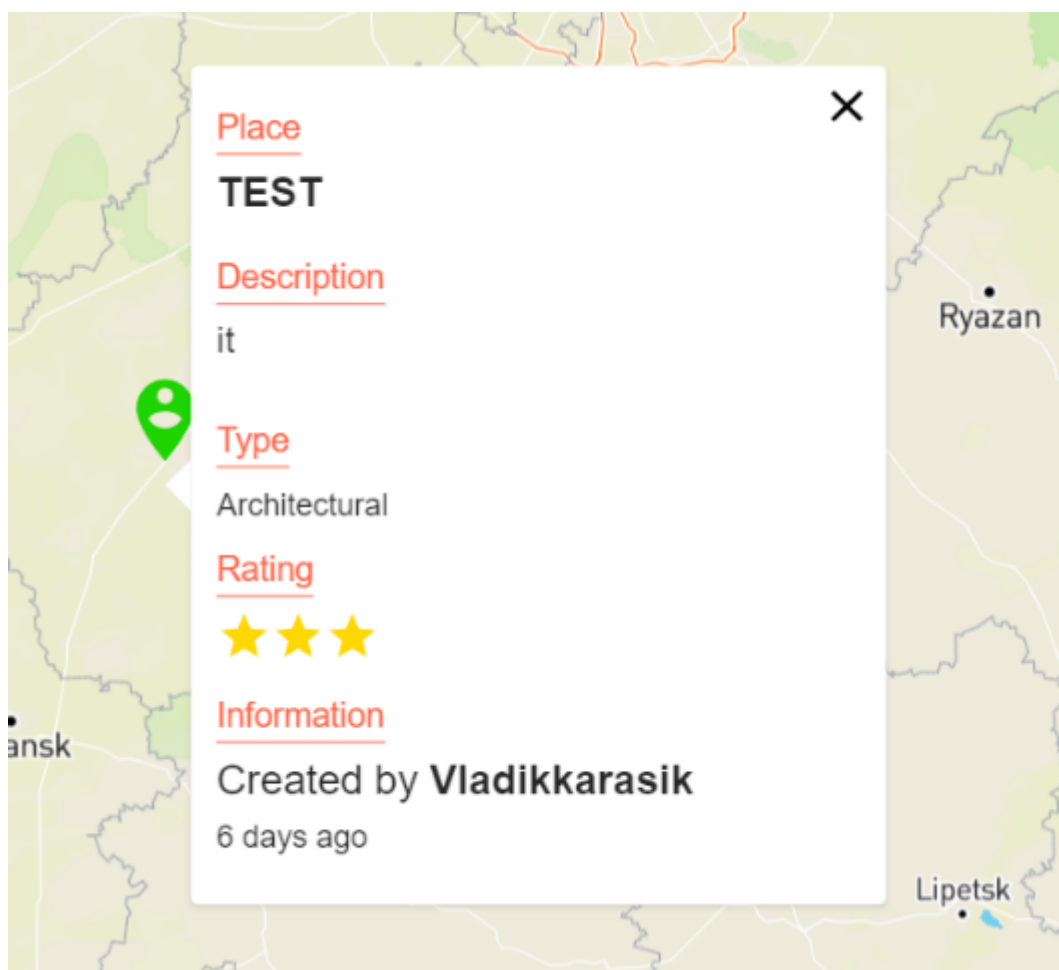


Рисунок 3.12– Модульне вікно з інформацією

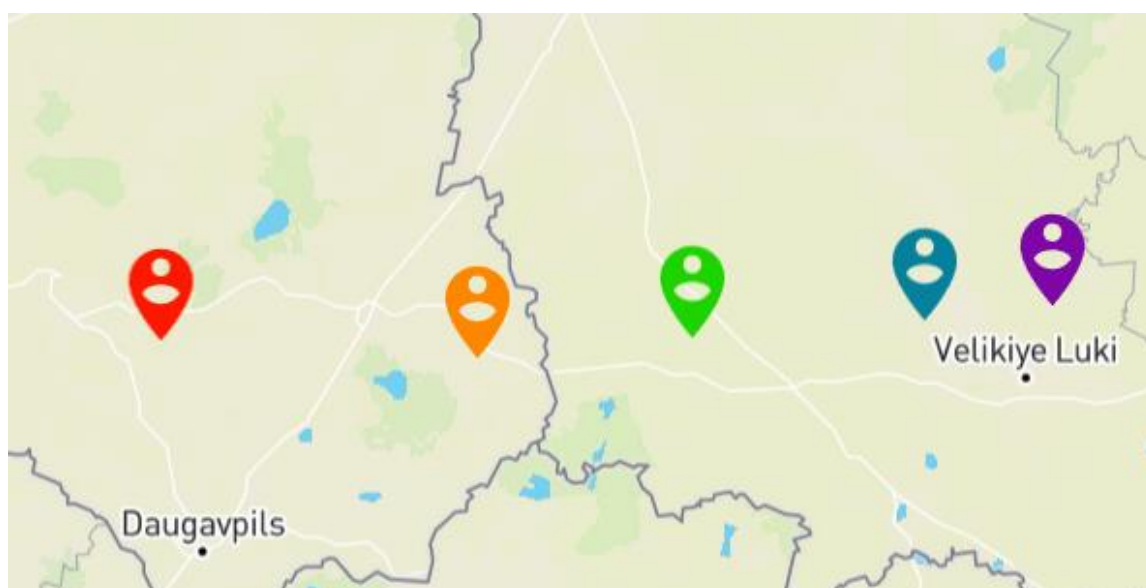


Рисунок 3.13 – Відображення рейтингової системи кольором

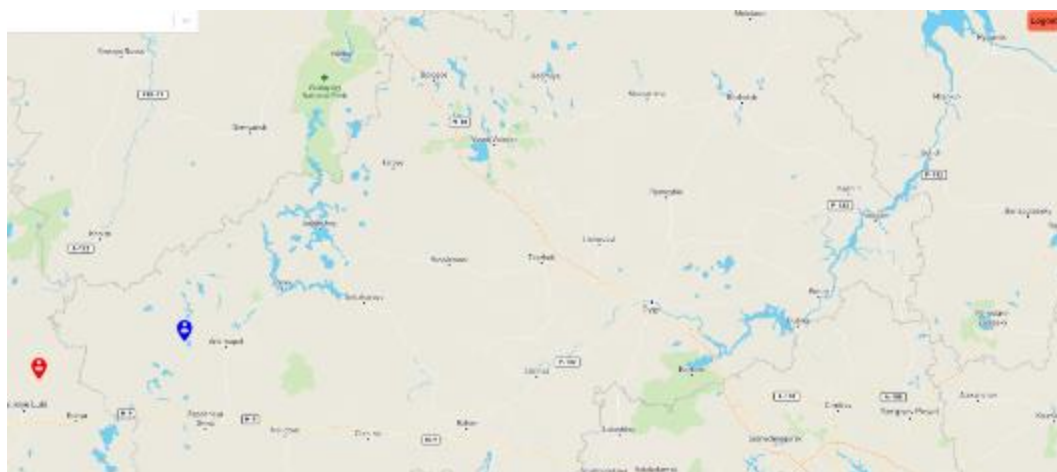


Рисунок 3.14 – Відображення кольору крапок зареєстрованого користувача

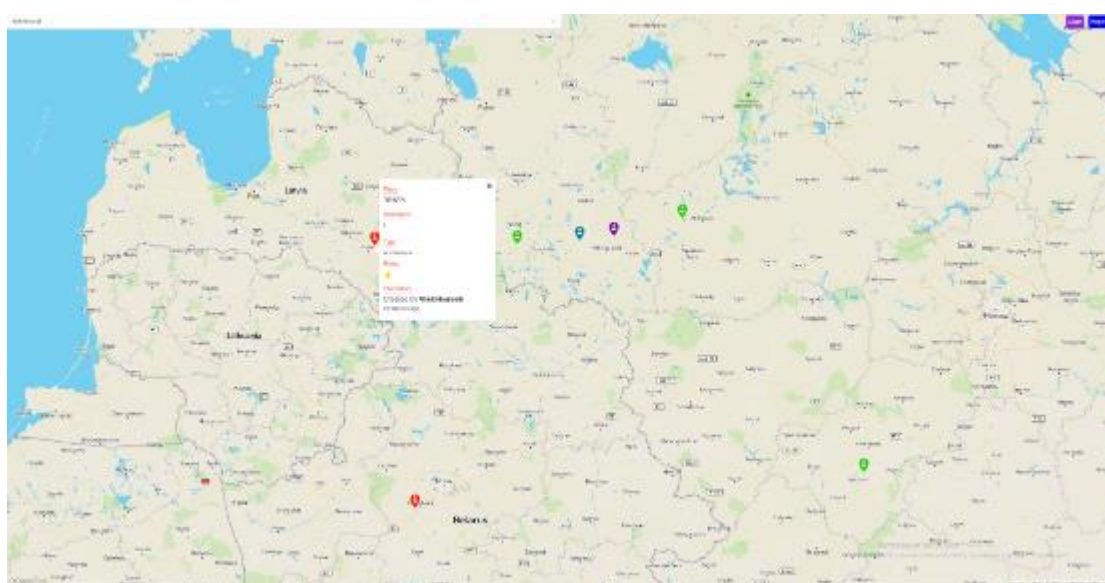


Рисунок 3.15 – Приклад фільтрації

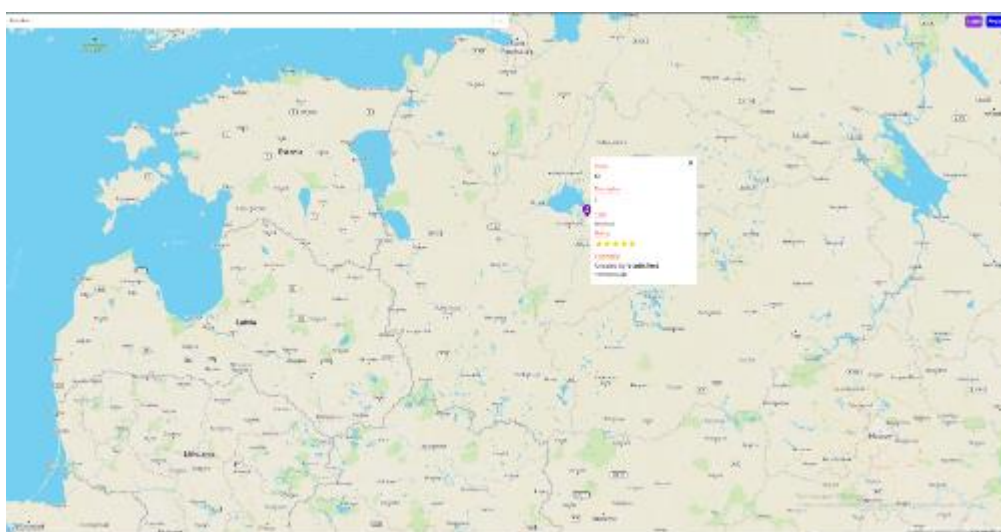


Рисунок 3.16 – Приклад фільтрації

Зареєстровані користувачі можуть створювати крапки на карті, для цього їм необхідно викликати модальне вікно створення крапки, двійним кліком миші або двійним тапом на мобільних пристроях, реалізація цього функціоналу знаходиться у файлі Createpin.js. Якщо крапка належить користувачу, при перегляді модального вікна крапки, буде доступна кнопка «Видалити» (рис 3.17), при натисканні на яку буде викликано модальне вікно видалення крапки, ралізація цього функціоналу знаходиться у файлі DeleteModal.js. Модальні вікна створення та видалення крапки наведено на рис 3.18 – 3.19.

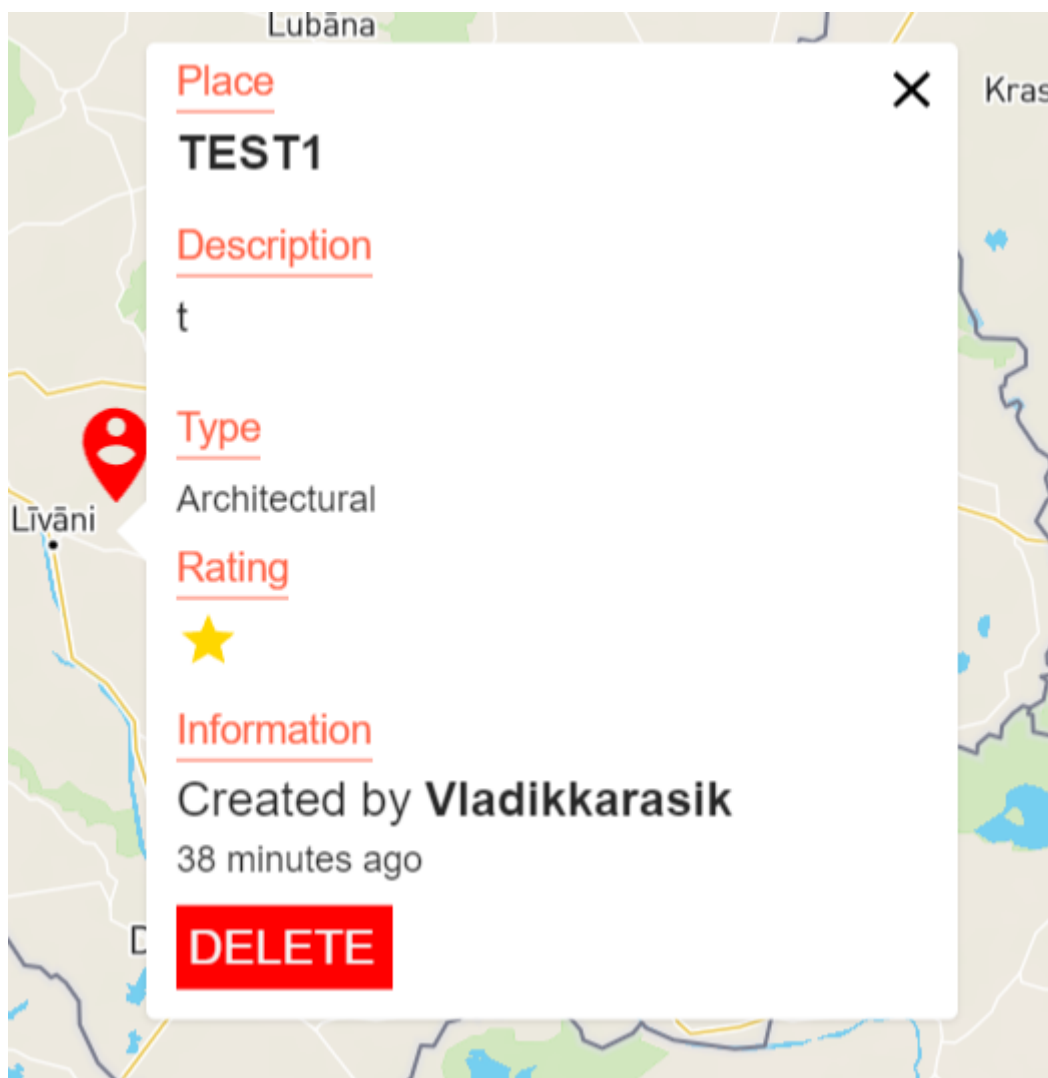
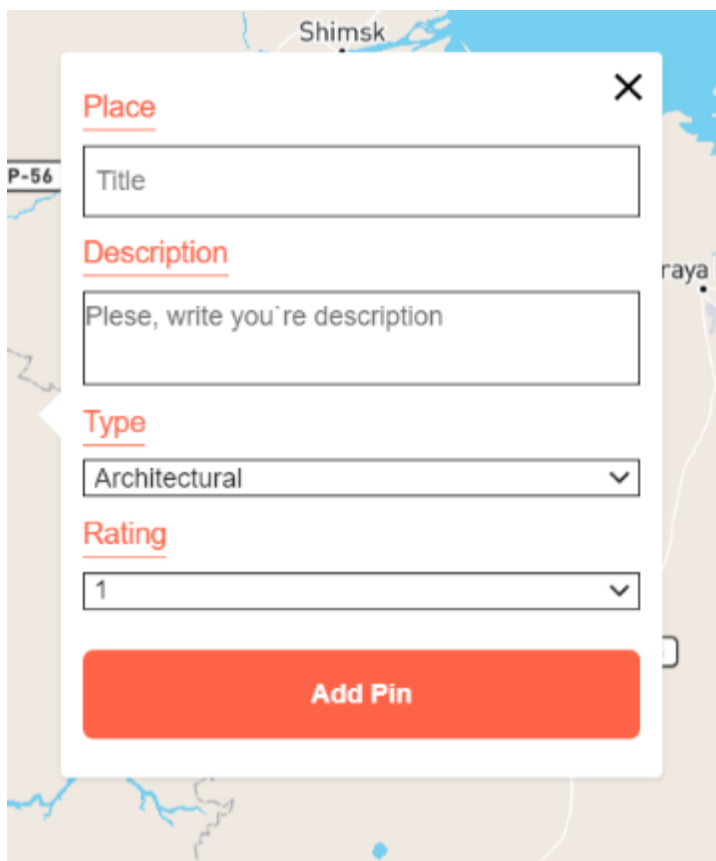
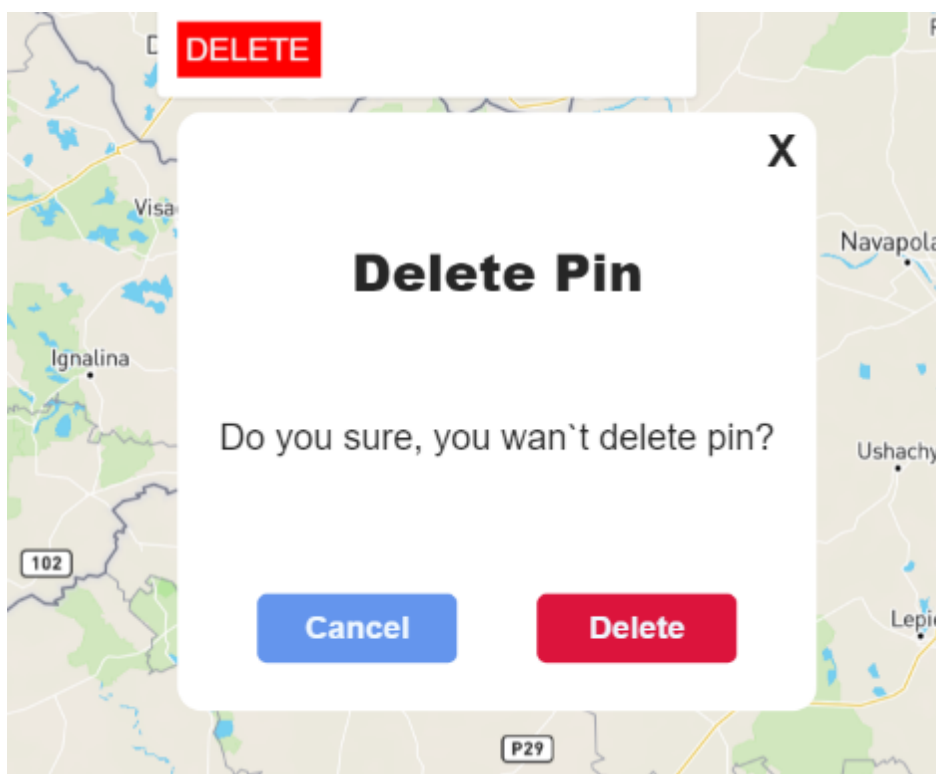


Рисунок 3.17 – Кнопка «Видалити»



A modal window titled "Place" with a close button (X) in the top right corner. It contains four sections: "Title" with a text input field, "Description" with a text input field containing the placeholder "Plese, write you`re description", "Type" with a dropdown menu showing "Architectural", and "Rating" with a dropdown menu showing "1". At the bottom is a large orange button labeled "Add Pin". The background shows a map with labels like "Shimsk" and "P-56".

Рисунок 3.18 – Модальне вікно створення крапки



A modal window titled "Delete Pin" with a close button (X) in the top right corner. It contains the text "Do you sure, you wan`t delete pin?". At the bottom are two buttons: a blue "Cancel" button and a red "Delete" button. Above the modal, a red button labeled "DELETE" is visible on the map background. The map background shows locations like "Visa", "Ignalina", "Navapola", "Ushachy", and "Lepie".

Рисунок 3.19 – Модальне вікно видалення крапки

3.3 Тестування застосунку

Тестування застосунку проводилось згідно з розроблених вимог у технічному завданні, а саме функціональних вимог, та вимог до кросбраузерності та кросплатформеності. Тестування функціональних вимог, проводилось згідно сценаріїв дії користувача, за технологією Дія користувача – Очікуваний результат – Успішний сценарій – Неуспішний сценарій, тестування відбувається з розрахунку на наявність у користувача інтернет зв'язку, та справності пристрою, на якому запущено веб-застосунку. Тестування кросплатформеності та кросбраузерності, проводилось на двох пристроях, перший мобільний пристрій на операційній системі IOS, у браузері Safari, з роздільною здатністю екрану 414x896px, розробка проводилась на стаціонарному комп'ютері, на операційній системі Windows 10 у браузері GoogleChrome, з роздільною здатністю екрану 2560x1440px, результати вигляду застосунку на стаціонарному комп'ютері наведено у розділі 3.2.

Тестування функціональних вимог наведено у табл 3.2.

Таблиця 3.2 – Тестування функціональних вимог

Функціональна вимоги	Сценарії			
	Дія	Очікуваний результат	Успішний сценарій	Неуспішний сценарій
1	2	3	4	5
Можливість перегляду інформації на крапках	Клік користувача по крапці	Виклик модального вікна з інформацією	Відбувається виклик модального вікна	Неуспішного сценарію не було виявлено
Можливість обрати тип архітектурної пам'ятки для перегляду	Обрання фільтру архітектурної пам'ятки	Виведення крапок на мапі які відповідають фільтру	Користувач бачить на мапі крапки за обраним фільтром	Якщо у системі немає ні однієї крапки, то мапа залишається порожньою

Продовження таблиці 3.2

1	2	3	4	5
Можливість зареєструватися у системі	Клік по кнопці «Зареєструватись»	Реєстрація користувача у системі	Користувач ввів коректні дані у відповідні поля, система створила документ користувача у базі даних, система вивела повідомлення про успішну реєстрацію у модальному вікні	Користувач ввів не коректні дані, система вивела повідомлення про помилку реєстрації
Можливість логіну у систему	Клік по кнопці «Увійти»	Користувач залогінувся у системі	Користувач ввів коректні дані у відповідні поля, система створила токен для його та занесла у браузерне сховище системи	Користувач ввів невірні дані, система вивела повідомлення про помилку логіну
Можливість логіну у систему	Клік по кнопці «Увійти»	Користувач залогінувся у сесії,	Користувач ввів коректні дані у відповідні поля, система створила токен для його сесії, та занесла у браузерне сховище	Користувач ввів невірні дані, система вивела повідомлення про помилку логіну
Можливість користувача створити крапку на мапі	Подвійний клік по мапі	Користувач створив крапку	Користувач заповнив усі поля у модальному вікні створення, натиснув кнопку «Створити», крапка внеслась до системи, та відмалювалась на мапі	Користувач заповнив не усі поля у модальному вікні створення, натиснув кнопку «Створити», крапка не внеслась до системи, та модальне вікно закрилось
Вихід користувача з аккаунту	Клік по кнопці «Вийти»	Користувач вийшов з системи	Користувач вийшов з системи, його токен з сесію видалився з сховища у браузері	Користувач власноруч видалив токен зі сховища у браузері, система не реагує на натискання кнопки, потрібно перезапустити сторінку
Можливість видалити крапку	Клік по кнопці «Видалити»	Користувач видалив крапку, кнопку,	Користувач клікає по крапці на мапі, яка належить йому, присутня кнопка «Видалити». Користувач натискає на та підтверджує видалення	Користувач клікає по крапці на мапі, яка йому не належить, кнопка видалити відсутня у модальному вікні

Тестування на кросплатформенність та кросбраузерність, під собою перш за все має на увазі, те що мобільні пристрої, або інші пристрої, мають різну пропускну можливість прийняття інтернет зв'язку, та різні процесорні ресурси, тому сайт повинен бути оптимізованим, та не потребувати великих системних ресурсів. Для тестування на продуктивність(performance) застосунку, було обрано вбудовану у консоль розробника GoogleChrome засіб LightHouse [16]. LightHouse аналізує по 4 показникам веб сайт, та видає звіт, з оцінкою по кожному з пункту від 1 до 100.

Перший показник продуктивність, його оцінка показує те, наскільки добре оптимізований сайт, та на скільки він буде завантажувати систему, чим менша оцінка, тим більш погано сайт оптимізований

Другий показник це доступність, це те наскільки сайт придатний для використання спеціальними апаратами для людей з вадами здоров'я, чи менша оцінка, тим більш сайт не придатний для взаємодії з спеціальними пристроями

Третій показник це найкращі практики, відноситься до програмної реалізації застосунку, показує наскільки програма відповідає сьогоdnішнім стандартам захисту, наскільки стабільний зв'язок з сервером, тощо.

Четвертий показник це показник SEO, його оцінка показує те, наскільки сайт оптимізований для пошукових ботів у інтернеті.

Звіт LightHouse наведено на рис 3.20.

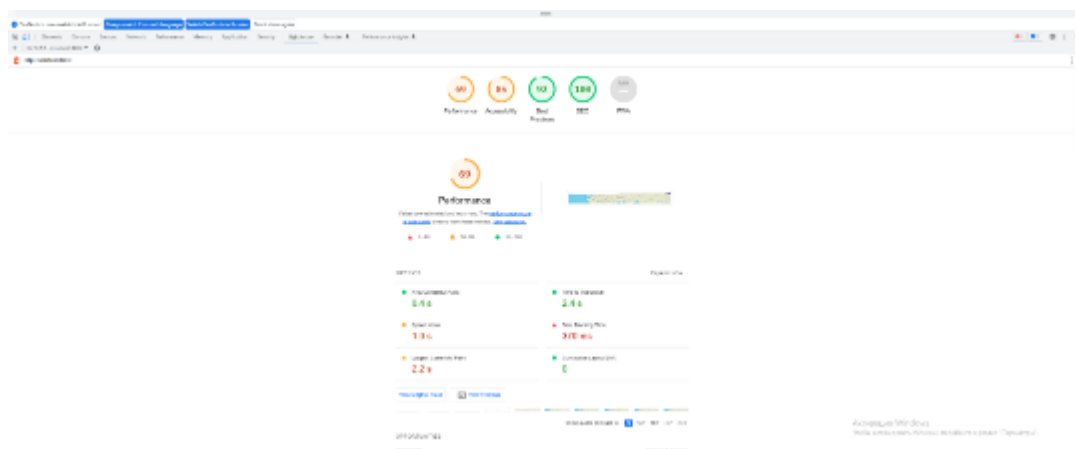


Рисунок 3.20 – Звіт LightHouse

Звіт надає нам показники за продуктивність 69, за доступність 85, за найкращі практики 92, та заSEO 100. З цих показників важливі для кросплатформеності та кросбраузерності лише продуктивність та доступність, що мають показники вище середнього, що є гарним результатом, і являється доказом оптимізованості програми.

Для тестування на час завантаження сайту в залежності від інтернет з'єднання, було обрано вкладку Network, у консолі розробника GoogleChrome. Дана вкладка, дозволяє виставити швидкість інтернету, та проаналізувати час за який сторінка веб-застосунку завантажується на девайсі користувача.

Результат перевірки часу завантаження наведено на рисунках 3.21 – 3.23



Рисунок 3.21 – Швидкість 4Гінтернету

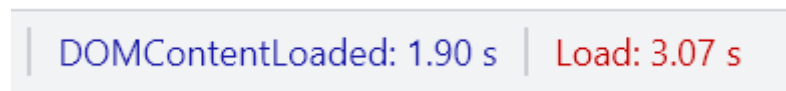


Рисунок 3.22 – Швидкість швидкого 3Гінтернету



Рисунок 3.23 – Швидкість повільного 3Гінтернету

Подія DOMContentLoaded запускається, коли всі вузли на сторінці створені в дереві DOM. Подія Load запускається, коли всі ресурси, такі як зображення та підкадри, завантажуються повністю [Error! Unknown switch argument.].

Результати тестування вигляду застосунку на мобільному пристрої, наведено у рис. 3.24– 3.27.

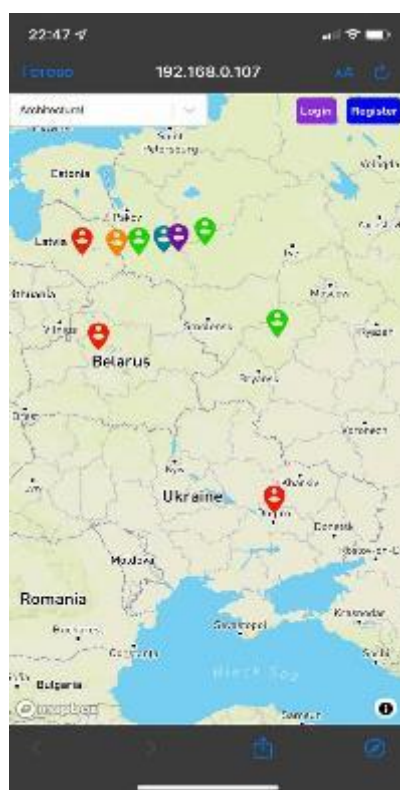


Рисунок 3.24 – Вигляд застосунку на мобільному пристрої



Рисунок 3.25 – Вигляд застосунку на мобільному пристрої

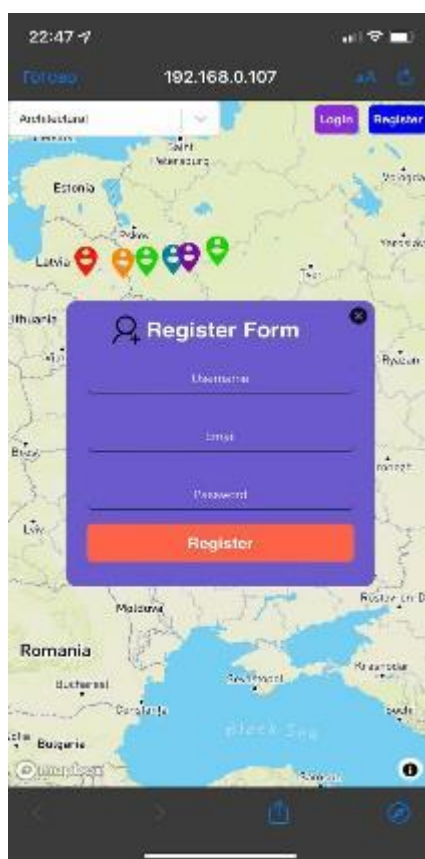


Рисунок 3.26 – Вигляд застосунку на мобільному пристрої

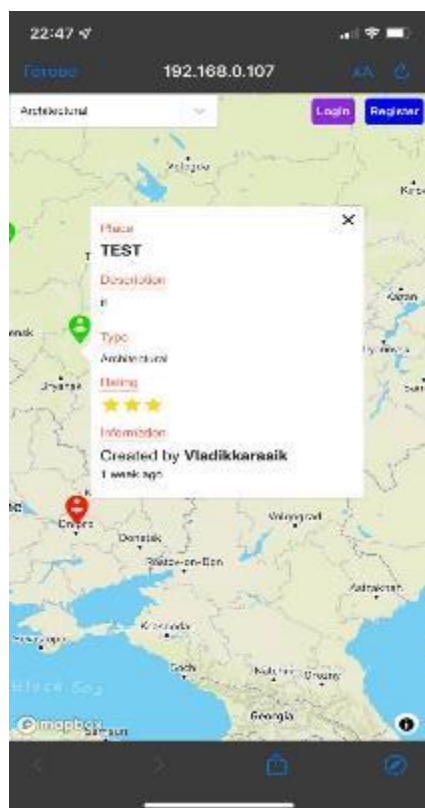


Рисунок 3.27 – Вигляд застосунку на мобільному пристрої

3.4 Висновки за розділом 3

У даному розділі було розроблено веб-застосунок за допомогою архітектурного патерну MVC. Були виділенні компоненти контролеру, моделі та представлення, були описанні вхідні та вихідні дані компонентів, було описано функції, завдяки яким іде обробка та візуалізація даних у системі. Згідно концепції модульності, кожен із компонентів MVC був розроблений таким чином, що зміни у одному компоненті MVC, не призведуть до зміни функціональності іншого компонента, що надає можливість більш швидко та легко розширювати функціонал програми у майбутньому. Були побудовані діаграми компонентів для візуалізацій, як програма обмінюється даними між собою.

Розроблено графічний інтерфейс веб-застосунку, зверстані та стилізовані елементи керування в застосунку, реалізована логіка взаємодії користувача з графічним інтерфейсом.

Згідно з функціональними вимогами до програмного застосунку, які описані у документі технічного завдання, розроблено основні сценарії роботи користувача з системою, та проведені тести роботи програмного застосунку на основі цих сценаріїв. Описані вхідні дані, очікуваний результат, та вдалі та невдалі сценарії роботи користувача, описано реагування програми на вдалий та невдалий сценарій.

Проведено тестування застосунку за допомогою інструменту LightHouse, за такими критеріями: продуктивність, доступність, захист системи та SEO показників. Результати тестування: продуктивність – 69 з 100, доступність – 85 з 100, захист систем – 92 з 100, SEO показник 100 з 100. Показники отримані в результаті аналізу системи за допомогою LightHouse вище за середні, це означає що застосунок придатний для використання у реальних умовах в інтернеті.

Проведено тестування часу першого завантаження застосунку, тестування проводилося на швидкостях 4G, 3G та повільному 3G, результати отриманого часу: 4G – 1.2 секунди, 3G – 3.07 секунди, повільний 3G – 7.06 секунди. Результати є значно вищі за середні, і це означає що додатком можливо зручно користуватись навіть при повільному інтернет з'єднанні.

Тестування на кроссбраузерність та кроссплатформеність, а саме вигляд графічного інтерфейсу та роботу функціоналу застосунку, проведено у браузерах Safari та GoogleChrome, та на операційних системах Windows 10 та IOS 15. Результати тестування показали, що функціонал системи працює однаково на різних системах та у різних браузерах, та вигляд елементів графічного інтерфейсу не відрізняється на різних системах.

Проведено тест вигляду графічного інтерфейсу на різних роздільних здатностях екрану, а саме на стаціонарному комп'ютері з роздільною здатністю екрану 2560x1440 пікселів, та на мобільному телефоні з роздільною здатністю екрану 414x896 пікселів. Результати тестування показали що графічний інтерфейс адаптивний, та зручний у використанні в незалежності від роздільної здатності екрану девайсу.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування

4.1.1 Призначення програми

Дане програмне забезпечення призначення для отримання користувачем релевантної та структурованої інформації з приводу того чи іншого туристичного об'єкта у світі, у форматі взаємодії з інтерактивною мапою світу.

До функцій програми відносяться:

- реєстрація користувача у системі;
- створення крапок з інформацією з приводу туристичних об'єктів;
- видалення крапок;
- логін користувача до системи;
- перегляд інформації яка містить у модальному вікні крапки;
- фільтрація крапок за певним типом архітектурної пам'ятки.

4.1.2 Умови застосування програми

Дане програмне забезпечення підтримує будь-яку операційну систему та пристрій, який підтримує веб-браузер, програма має адаптивний інтерфейс, що дозволяє користуватися нею на екранах з роздільною здатністю від 3840x2160 пікселей до 320x568 пікселей. Для функціонування застосунку необхідно мати пристрій який має можливість запустити веб-браузер, та наявність стабільного інтернет зв'язку.

4.2 Характеристики програми

Програмне забезпечення реагує на помилки у системі, які можуть бути спричинені як введенням користувачем невірних даних, як наприклад не заповнення полів вводу при реєстрації аккаунту, так і системні помилки, наприклад відсутність інтернет зв'язку.

Для коректної роботи застосунку, необхідно мати постійно працюючий сервер.

4.3 Звертання до програми

Звертання до програми виконується завдяки мережевим запитам протоколу HTTP, а саме типи запиту «GET», «POST», «DELETE». Запити здійснюються на сервер, за допомогою мережі інтернет.

4.4 Початкові та вихідні дані

Початковими даними у застосунку є інтерактивна мапа світу, крапки на мапі з інформацією.

Вихідними даними є модальні вікна з інформацією той чи іншої крапки.

4.5 Повідомлення програмісту

Помилки які можуть виникнути при запиті на сервер:

- 500 internalserver – виникає у разі коли запит на сервері не був оброблений через невірно переданий URL-адрес;
- 400 badrequest– виникає у разі надіслання невірних даних.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Призначення програми наведено у розділі 4.1.1.

5.2 Умови виконання програми

Умови виконання програми наведено у розділі 4.1.2.

5.3 Виконання програми

Для початку роботи з веб-застосунком, необхідно перейти на URL-адресу, за якою буде знаходитись веб-застосунок.

Після того, як користувач опинився на сторінці веб-застосунку, йому буде доступна можливість зареєструватись на сайті, обрати у фільтрі архітектурний тип крапок, який йому потрібен, та натиснувши на будь яку крапку, користувач може отримати інформацію у модальному вікні крапки, а саме:

- рейтинг крапки;
- заголовок крапки;
- опис крапки;
- архітектурний тип крапки;
- дату та час створення крапки;
- ім'я користувача, хто створив цю крапку;

Модальне вікно з інформацією наведено у рис 5.1.

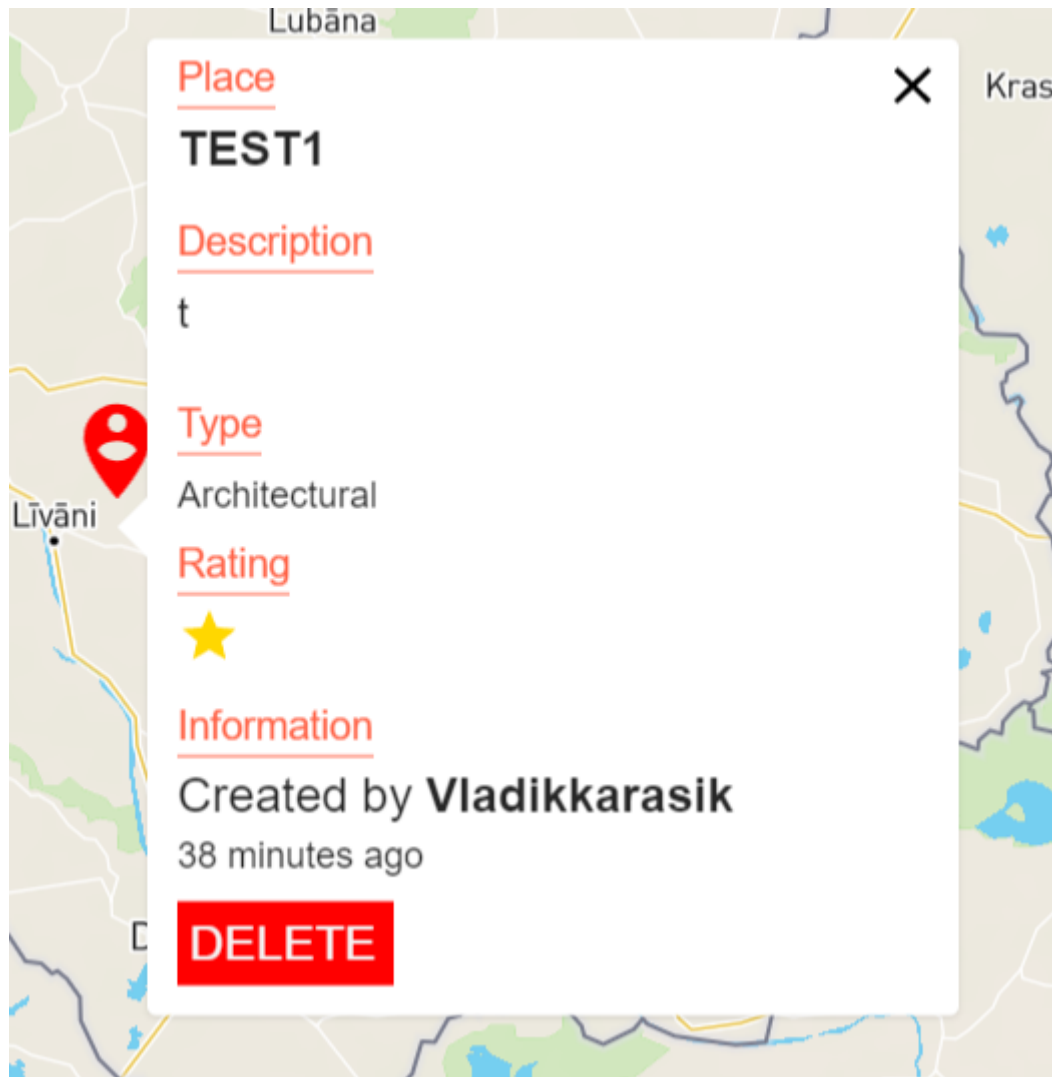


Рисунок 5.1 – Модальне вікно з інформацією

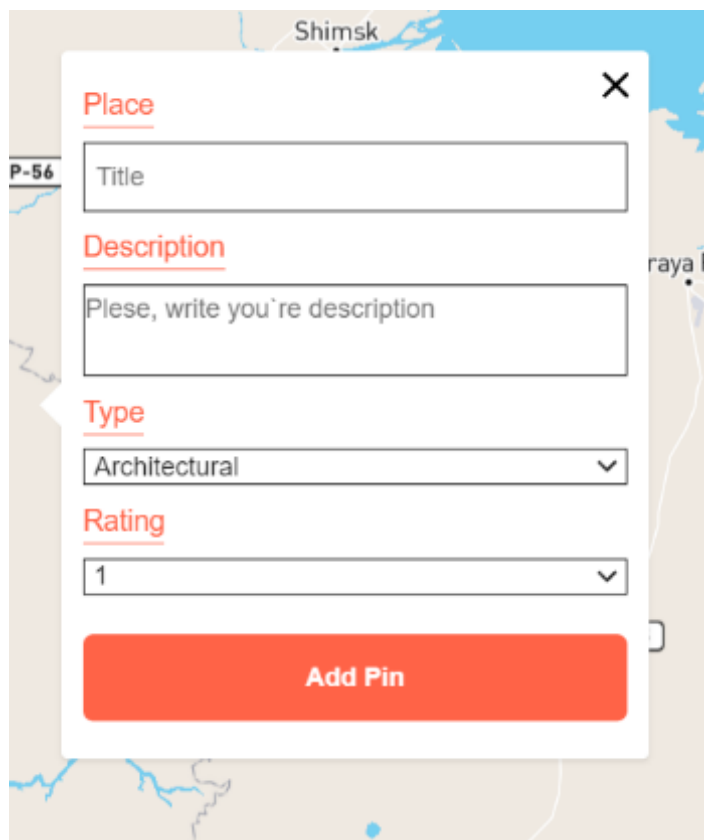
Після того, як користувач зареєструвався, йому буде доступна можливість створення крапок, та видалення крапок яких він створив.

У модальному вікні створення крапки вказати такі дані:

- рейтинг крапки;
- заголовок крапки;
- опис крапки;
- архітектурний тип крапки.

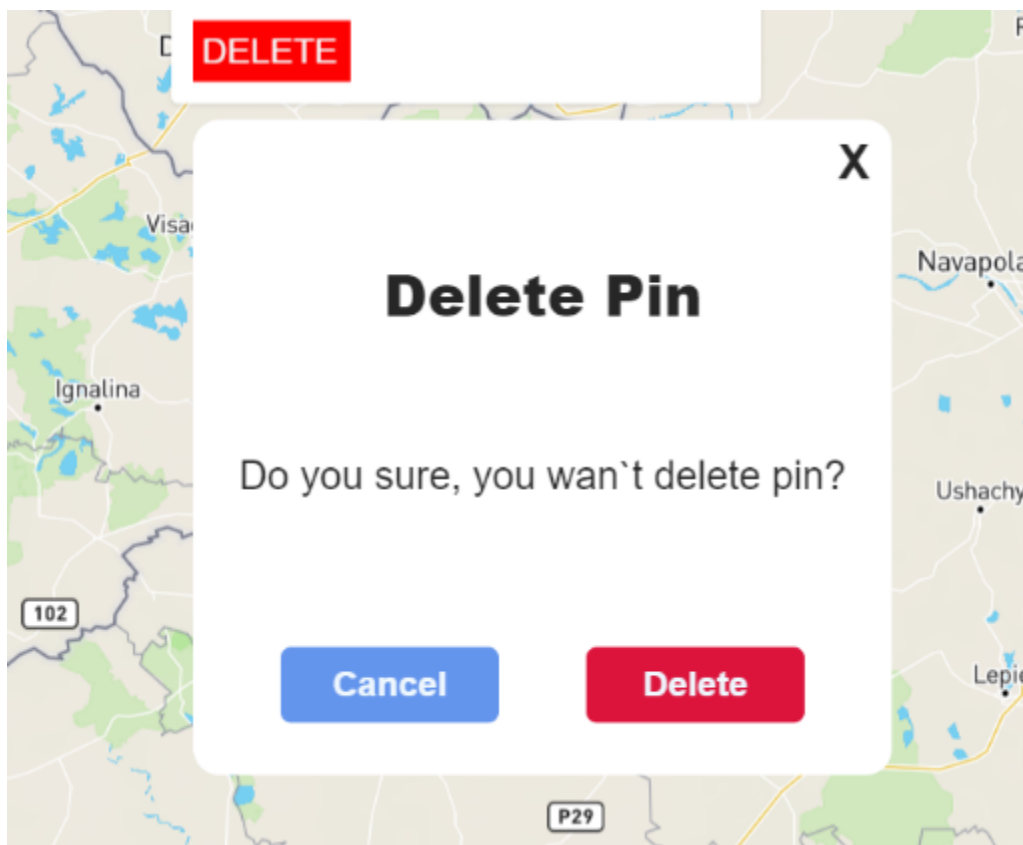
Кнопка видалення у модальному вікні крапки, з'являється тільки у випадку, коли крапка належить користувачу, який відкрив це модальне вікно.

Модальні вікна створення та видалення наведено у рис. 5.2 – 5.3.



A modal window for creating a pin, overlaid on a map. The window has a close button (X) in the top right corner. It contains four sections: 'Place' with a 'Title' input field; 'Description' with a text area containing the placeholder 'Plese, write you`re description'; 'Type' with a dropdown menu showing 'Architectural'; and 'Rating' with a dropdown menu showing '1'. At the bottom is a large orange button labeled 'Add Pin'.

Рисунок 5.2 – Модальне вікно створення крапки



A modal window for deleting a pin, overlaid on a map. The window has a close button (X) in the top right corner. At the top left, there is a red button labeled 'DELETE'. The main text reads 'Delete Pin' in bold, followed by the question 'Do you sure, you wan`t delete pin?'. At the bottom are two buttons: a blue 'Cancel' button and a red 'Delete' button.

Рис 5.3 – Модальне вікно видалення крапки

5.4 Повідомлення

Повідомлення, які система може видати користувачу:

- адреса електронної пошти повинна містити символ @. Ця помилка видається при реєстрації, у випадку коли користувач ввів у полі електронна пошта не вірні дані. У випадку цієї помилку користувачу необхідно у відповідному полі правильно вказати адресу електронної пошти;
- somethingwrong. Ця помилка видається при реєстрації та при логіну у систему, коли користувач ввів не валідні дані. У випадку цієї помилки, користувач повинен ввести інші дані при реєстрації, так як такий email або логін вже існує, або у випадку логіну до системи, вказати вірні дані свого аккаунту;
- registersuccessfull. Це повідомлення виводиться у випадку коли користувач успішно пройшов реєстрацію.

ВИСНОВКИ

У результаті виконання дипломної роботи, було розроблено веб-застосунок «Туристична онлайн мапа» для отримання релевантної інформації у галузі туризму.

Досліджено актуальність теми пошуку користувачем релевантної інформації у туристичній галузі, проаналізовані існуючі аналоги а саме GoogleMaps, OpendripMap та Discover.ua, виділені їх недоліки та переваги. По результатам аналізу, розроблені функціональні можливості програми, та технічне завдання.

Так як програмне забезпечення яке розроблене у ході виконання дипломної роботи являє з себе веб-застосунок, то архітектурним патерном обрано MVC, що є стандартом для розроблення веб-сайтів та веб-застосунків. Для розробки компонентів представлення було обрано мову програмування JavaScript та фреймворк React, для розробки компонентів контролеру було обрано мову програмування JavaScript та фреймворк Express. У якості моделі, обрано базу даних MongoDB, у якій таблиці з даними представленні у вигляді JavaScript об'єктів, що не потребує від розробника розробляти зв'язки між таблицями.

Були розроблені компоненти контролеру, представлення та моделі, описані зв'язки між компонентами, описані функції та змінні які реалізують функціональність компонентів. Розроблено графічний інтерфейс користувача та проведено тестування застосунку на швидкодію першого завантаження сайту. Отримані результати є значно вищі за середні, і це означає що застосунком можливо зручно користуватись навіть при повільному інтернет з'єднанні.

Результатом дипломної роботи є веб-застосунок «Туристична онлайн мапа» який являє собою мапу світу, що надає користувачу можливість отримання релевантної та структурованої інформації у туристичній галузі.

Застосунок поділений на окремі модулі, що у майбутньому допоможе більш ефективно та швидко розширювати його функціонал.

В подальшому можливо додати можливість редагувати крапки та розробити особистий кабінет для зареєстрованого користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Пандемія та туризм: глобальна економіка може втратити понад 4 трлн доларів [Електронний ресурс]. – 2021. – Режим доступу: <https://news.un.org/ru/story/2021/06/1405562>.
2. Стандартизація та сертифікація туристичних послуг [Електронний ресурс]. – Режим доступу: https://studbooks.net/65070/turizm/standartizatsiya_sertifikatsiya_turisticheskikh_uslug.
3. Discover.ua [Електронний ресурс]. – Режим доступу: <https://discover.ua/locations>.
4. Інформація з приводу розробки [Електронний ресурс]. – Режим доступу: <https://discover.ua/>.
5. Jam Digital [Електронний ресурс]. – Режим доступу: <https://jam.in.ua/>.
6. Opentripmap.com [Електронний ресурс]. – Режим доступу : <https://opentripmap.com/ru/#3.25/-9.60/42.02>.
7. OpenTilesMap [Електронний ресурс]. – Режим доступу: <https://openmaptiles.org/>.
8. Google Maps [Електронний ресурс]. // Wikipedia. – 2020. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D1%82%D1%8B_Google.
9. MongoDB [Електронний ресурс]. – Режим доступу: <https://www.mongodb.com/>.
10. Склад системи туризму [Електронний ресурс]. – Режим доступу: https://tourlib.net/books_tourism/izotova1-4.htm.
11. REST [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/REST>.
12. Express [Електронний ресурс]. – Режим доступу:

<https://uk.wikipedia.org/wiki/Express.js>.

13. Blowfish [Электронный ресурс]. – Режим доступа:
<https://ru.wikipedia.org/wiki/Blowfish>.

14. Bcrypt [Электронный ресурс]. – Режим доступа:
<https://ru.wikipedia.org/wiki/Bcrypt>.

15. MapBox [Электронный ресурс]. – Режим доступа:
<https://www.mapbox.com/about/maps/>.

16. LightHouse [Электронный ресурс]. – Режим доступа:
<https://developer.chrome.com/docs/lighthouse/overview/>.

17. DOMContentLoadedvsload [Электронный ресурс]. – Режим доступа:
<https://thisthat.dev/dom-content-loaded-vs-load/>.

ДОДАТОК А**Код програми**

Серверна частина:

Index.js

```
const express = require("express");
const mongoose = require("mongoose");
const dotenv = require("dotenv");
const PINROUTES = require("./routes/pins");
const USERSROUTES = require("./routes/users");

dotenv.config();

const APPLICATION = express();

APPLICATION.use(express.json());
APPLICATION.use("/api/pins", PINROUTES);
APPLICATION.use("/api/users", USERSROUTES);

mongoose
  .connect(process.env.MONGODB_URL)
  .then(() => console.log("Database Connect"))
  .catch((err) => console.log(err));

APPLICATION.listen(8800, () => {
  console.log("Back start");
});
```

Pins.js

```
const ROUTER = require("express").Router();
const PIN = require("../models/Pin");

//create pin

ROUTER.post("/", async (req, res) => {
  const NEWPIN = new PIN(req.body);
  try {
    const SAVEPIN = await NEWPIN.save();
    res.status(200).json(SAVEPIN);
  } catch (err) {
```

```

res.status(500).json(err);
  }
});

//get all pin

ROUTER.get("/", async (req, res) => {
  try {
    const PINS = await PIN.find();
    res.status(200).json(PINS);
  } catch (err) {
    res.status(500).json(err);
  }
});

//delete pin

ROUTER.delete("/delete/", async (req, res) => {
  try {
    const DELETEPIN = await PIN.findOne({ _id: req.body._id });
    if (!DELETEPIN) {
      return res.status(400).json({ message: "Pin not found" });
    }
    await DELETEPIN.remove();
    res.status(200).json(DELETEPIN);
  } catch (err) {
    res.status(500).json(err);
  }
});

module.exports = ROUTER;

```

Users.js

```

const ROUTER = require("express").Router();
const USERS = require("../models/User");
const BCRYPT = require("bcrypt");

ROUTER.post("/register", async (req, res) => {
  try {

```

```

//generate password
const SALT = await BCRYPT.genSalt(10);
const HASHPASSWORD = await BCRYPT.hash(req.body.password, SALT);

//create new user
const NEWUSER = new USERS({
  username: req.body.username,
  email: req.body.email,
  password: HASHPASSWORD,
});

//save user and send response
const USER = await NEWUSER.save();
res.status(200).json(USER._id); //it makes no sense to save the entire user,
just his id is enough
} catch (err) {
res.status(500).json(err);
}
});

ROUTER.post("/login", async (req, res) => {
  try {
    //find user and email
    const USER = await USERS.findOne({ username: req.body.username });
    const EMAIL = await USERS.findOne({ email: req.body.email });

    //validate password
    const VALIDATE_PASSWORD = await BCRYPT.compare(
req.body.password,
USER.password
    );

    if (!USER || !EMAIL || (!VALIDATE_PASSWORD &&res.status(400))) {
      return res.status(400).json("Wrong username or email or password");
    } else {
      //send response
      return res.status(200).json({ _id: USER._id, username: USER.username });
    }
  } catch (err) {
res.status(500).json(err);
  }
}

```

```
});
```

```
module.exports = ROUTER;
```

Клієнтська частина:

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
ReactDOM.render(
  <React.StrictMode>
  <App />
</React.StrictMode>,
document.getElementById('root')
);
```

App.js

```
import React, { Suspense } from 'react';

import MapPin from './component/map/mappin';

const App = () => {
  return (
    <Suspense fallback={<div>Loading...</div>}>
    <MapPin />
    </Suspense>
  );
};

export default App;
```

Mappin.js

```
import React, { useEffect, useState } from 'react';
import Map, { Marker, Popup } from 'react-map-gl';
import Select from 'react-select';
```

```

import 'mapbox-gl/dist/mapbox-gl.css';

import { webToken } from '../constant/const';

import { Close, PersonPinCircle, Star } from '@material-ui/icons';

import axios from 'axios';
import { format } from 'timeago.js';

import style from './style.module.css';
import 'antd/dist/antd.css';

import Register from '../authAndRegister/Register';
import Auth from '../authAndRegister/Auth';
import DeleteBtn from '../deleteBtn/DeleteBtn';
import CreatePin from '../createPin/createpin';

const MapPin = () => {
  const storage = window.localStorage;

  const options = [
    { value: 'Architectural', label: 'Architectural' },
    { value: 'Culinary', label: 'Culinary' },
    { value: 'Sports', label: 'Sports' },
    { value: 'Historical', label: 'Historical' },
    { value: 'Other', label: 'Other' },
  ];
  const [selectedOption, setSelectedOption] = useState(options[0]);

  let [selectType, setSelectType] = useState(1);

  const [currentUser, setCurrentUser] = useState(storage.getItem('token'));
  const [pins, setPins] = useState([]);
  const [newPlacePin, setNewPlacePin] = useState(null);
  const [currentPlaceId, setCurrentPlaceId] = useState(null);

  const [title, setTitle] = useState(null);
  const [description, setDescription] = useState(null);
  const [type, setType] = useState(1);
  const [rating, setRating] = useState(1);

```

```

const [showRegister, setShowRegister] = useState(false);
const [showLogin, setShowLogin] = useState(false);

const [openDeleteModal, setIsOpenDeleteModal] = useState(false);

const handleMarkerClick = (id) => {
  setCurrentPlaceId(id);
};

const handleAddPopUpClick = (e) => {
  const { lat, lng } = e.lngLat;
  setNewPlacePin({
    lat,
    lng,
  });
};

const handleSubmit = async (e) => {
  e.preventDefault();
  const newPin = {
    username: currentUser,
    title,
    description,
    type,
    rating,
    lat: newPlacePin.lat,
    long: newPlacePin.lng,
  };

  try {
    const response = await axios.post('/pins', newPin);
    setPins([...pins, response.data]);
    setNewPlacePin(null);
  } catch (err) {
    console.log(err);
  }
};

const handleLogout = () => {
  storage.clear('token');

```

```

setCurrentUser(null);
  };

  const handleChange = (selectedOption) => {
    setSelectedOption({ selectedOption });
    selectType =
    selectedOption === options[0]
      ? setSelectedType(1)
      : selectedOption === options[1]
      ? setSelectedType(2)
      : selectedOption === options[2]
      ? setSelectedType(3)
      : selectedOption === options[3]
      ? setSelectedType(4)
      : selectedOption === options[4]
      ? setSelectedType(5)
      : null;
  };

  useEffect(() => {
    const getPins = async () => {
      try {
        const response = await axios.get('/pins');
        setPins(response.data);
      } catch (err) {
        console.log(err);
      }
    };
    getPins();
  }, []);

  return (
    <div>
      <Map
        initialState={{
          longitude: 32,
          latitude: 49,
          zoom: 4,
        }}
        style={{ width: '100vw', height: '100vh' }}

```

```

mapStyle="mapbox://styles/mapbox/streets-v9"
mapboxAccessToken={webToken}
onDbClick={handleAddPopUpClick}
>
<div className={`select-price`} style={{ width: '50%' }}>
<Select
  defaultValue={selectedOption}
  onChange={handleChange}
    options={options}
  />
</div>

    {pins.map((pin) => (
<>
      {pin.type === selectType ? (
<Marker
      longitude={pin.long}
      latitude={pin.lat}
      key={pin.type}
      anchor="bottom"
    >
<PersonPinCircle
      style={{
fontSize: '38',
      color:
pin.rating === 1 && !currentUser
      ? '#FF1800'
      : pin.rating === 2 && !currentUser
      ? '#FF8700'
      : pin.rating === 3 && !currentUser
      ? '#1DD300'
      : pin.rating === 4 && !currentUser
      ? '#05819E'
      : pin.rating === 5 && !currentUser
      ? '#7F07A9'
      : pin.username === currentUser
      ? 'red'
      : 'blue',
      }}
onClick={() => handleMarkerClick(pin._id)}
    />

```

```

</Marker>
) : null}
    {pin._id === currentPlaceId&& (
<Popup
    key={pin._id}
    longitude={pin.long}
    latitude={pin.lat}
    anchor="left"
    closeOnClick={false}
    onClose={() =>setCurrentPlaceId(null)}
  >
    <div className={style.card}>
      <label>Place</label>
      <h4 className={style.place}>{pin.title}</h4>
      <label>Description</label>
      <p className={style.review}>{pin.description}</p>
      <label>Type</label>
      <div className={style.type}>
        {pin.type === 1
          ? 'Architectural'
          : pin.type === 2
          ? 'Culinary'
          : pin.type === 3
          ? 'Sports'
          : pin.type === 4
          ? 'Historical'
          : pin.type === 5
          ? 'Other'
          : pin.type}
      </div>
      <label>Rating</label>
      <div className={style.rating}>
        {Array(pin.rating).fill(<Star className={style.star} />)}
      </div>
      <label>Information</label>
      <span className={style.username}>
        Created by <b>{pin.username}</b>
      </span>
      <span className={style.date}>{format(pin.createdAt)}</span>
      {pin.username === currentUser ? (
        <DeleteBtn

```

```

openDeleteModal={openDeleteModal}
setIsOpenDeleteModal={setIsOpenDeleteModal}
        pin={pin._id}
    />
) : null}
<Close
className={style.closePopUp}
onClick={() =>setCurrentPlaceId(null)}
    />
</div>
</Popup>
    )}
</>
    )}}
    {newPlacePin&&currentUser ? (
    <CreatePin
newPlacePin={newPlacePin}
setNewPlacePin={setNewPlacePin}
handleSubmit={handleSubmit}
setTitle={setTitle}
setDescription={setDescription}
setRating={setRating}
setType={setType}
    />
) : null}
    {currentUser ? (
    <button className={style.logout} onClick={handleLogout}>
        Logout
    </button>
    ) : (
    <div className={style.button__container}>
    <button className={style.login} onClick={() =>setShowLogin(true)}>
        Login
    </button>
    <button
className={style.register}
onClick={() =>setShowRegister(true)}
    >
        Register
    </button>
    </div>

```

```

    })
    {showRegister&&<Register setShowRegister={setShowRegister} />}
    {showLogin&& (
<Auth
setShowLogin={setShowLogin}
      storage={storage}
setCurrentUser={setCurrentUser}
      />
    )}
</Map>
</div>
  );
};

export default MapPin;

```

DeleteBtn.js

```

import React, { useState } from 'react';
import DeleteModal from './DeleteModal';

import style from './style.module.css';

const DeleteBtn = ({ setIsOpenDeleteModal, pin, openDeleteModal }) => {
  return (
    <div>
      <p className={style.deleteBtn} onClick={() =>setIsOpenDeleteModal(true)}>
        Delete
      </p>
      {openDeleteModal ? (
        <DeleteModalsetIsOpenDeleteModal={setIsOpenDeleteModal} pinId={pin} />
      ) : null}
    </div>
  );
};

export default DeleteBtn;

```

DeleteModal.js

```

import React from 'react';
import style from './modalDelete.module.css';
import axios from 'axios';

const DeleteModal = ({ setIsOpenDeleteModal, pinId }) => {
  const handleDeleteSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios
        .delete(`/pins/delete/`, { data: { _id: pinId } })
        .then(setIsOpenDeleteModal(false))
        .then(document.location.reload());
    } catch (err) {
      console.log(err);
    }
  };
  return (
    <div className={style.modalBackground}>
      <div className={style.modalContainer}>
        <div className={style.modalCloseBtn}>
          <button onClick={() =>setIsOpenDeleteModal(false)}> X </button>
        </div>
        <div className={style.modalTitle}>
          <h2 style={{ fontWeight: '900' }}>Delete Pin</h2>
        </div>
        <div className={style.modalBody}>
          <p>Do you sure, you wan't delete pin?</p>
        </div>
        <div className={style.modalFooter}>
          <button onClick={() =>setIsOpenDeleteModal(false)}>Cancel</button>
          <button onClick={handleDeleteSubmit}>Delete</button>
        </div>
      </div>
    </div>
  );
};

export default DeleteModal;

```

CreatePin.js

```

import style from './style.module.css';
import { Popup } from 'react-map-gl';
import { Close } from '@material-ui/icons';

const CreatePin = ({newPlacePin, setNewPlacePin, handleSubmit, setTitle, setDescription, setType, setRating}) =>
{
  return (
    <Popup
      longitude={newPlacePin.lng}
      latitude={newPlacePin.lat}
      anchor="left"
      closeOnClick={false}
      onClose={() =>setNewPlacePin(null)}
    >
      <div>
        <form className={style.card} onSubmit={handleSubmit}>
          <label>Place</label>
          <input
            placeholder="Title"
            onChange={(e) =>setTitle(e.target.value)}
          />
          <label>Description</label>
          <textarea
            placeholder="Plese, write you`re description"
            onChange={(e) =>setDescription(e.target.value)}
          />
          <label>Type</label>
          <select onChange={(e) =>setType(e.target.value)}>
            <option value="1">Architectural</option>
            <option value="2">Culinary</option>
            <option value="3">Sports</option>
            <option value="4">Historical</option>
            <option value="5">Other</option>
          </select>
          <label>Rating</label>
          <select onChange={(e) =>setRating(e.target.value)}>
            <option value="1">1</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5</option>
          </select>
        </form>
      </div>
    </Popup>
  );
}

```

```

</select>
<button className={style.Create__Pin} type="submit">
    Add Pin
</button>
<Close
  className={style.closePopUp}
  onClick={() =>setNewPlacePin(null)}
  />
</form>
</div>
</Popup>
  );
};

export default CreatePin;

```

Auth.js

```

import { Cancel } from '@material-ui/icons';
import React, { useRef, useState } from 'react';

import AuthIcon from '../assets/auth.png';
import axios from 'axios';

import style from './auth.module.css';

const Auth = ({ setShowLogin, storage, setCurrentUser }) => {
  const [fail, setFail] = useState(false);
  const userNameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();

  const handleSubmit = async (e) => {
    e.preventDefault();

    const user = {
      username: userNameRef.current.value,
      email: emailRef.current.value,
      password: passwordRef.current.value,
    };

    try {

```

```

        const response = await axios.post('/users/login', user);
storage.setItem('token', response.data.username);
setCurrentUser(response.data.username);
setShowLogin(false);
setFail(false);
    } catch (err) {
setFail(true);
    }
};

return (
<div className={style.auth__container}>
<form onSubmit={handleSubmit}>
<div className={style.logo}>
<imgsrc={AuthIcon} alt="register" />
        Sign In Form
</div>
<input type="text" placeholder="Username" ref={userNameRef} />
<input type="email" placeholder="Email" ref={emailRef} />
<input type="password" placeholder="Password" ref={passwordRef} />
<button>Sign In</button>
        {fail &&<span className={style.failure}>Something wrong</span>}
</form>
<Cancel
className={style.auth_cancel}
onClick={() =>setShowLogin(false)}
/>
</div>
);
};

export default Auth;

```

Register.js

```

import React, { useRef, useState } from 'react';
import axios from 'axios';

import RegIcon from '../assets/register.png';

import style from './register.module.css';

```

```

import { Cancel } from '@material-ui/icons';

const Register = ({ setShowRegister }) => {
  const [success, setSuccess] = useState(false);
  const [fail, setFail] = useState(false);
  const userNameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();

  const handleSubmit = async (e) => {
    e.preventDefault();
    const newUser = {
      username: userNameRef.current.value,
      email: emailRef.current.value,
      password: passwordRef.current.value,
    };

    try {
      await axios.post('/users/register', newUser);
    } catch (err) {
    }

    setFail(false);
    setSuccess(true);
    setFail(true);
  };

  return (
    <div className={style.register__container}>
      <form onSubmit={handleSubmit}>
        <div className={style.logo}>
          <img src={RegIcon} alt="register" />
          Register Form
        </div>
        <input type="text" placeholder="Username" ref={userNameRef} />
        <input type="email" placeholder="Email" ref={emailRef} />
        <input type="password" placeholder="Password" ref={passwordRef} />
        <button>Register</button>
        {success && (
          <span className={style.successfull}>Register successfull</span>
        )}
        {fail && <span className={style.failure}>Something wrong</span>}
      </form>
    </div>
  );
};

```

```
</form>
<Cancel
  className={style.register_cancel}
  onClick={() =>setShowRegister(false)}
    />
</div>
  );
};

export default Register;
```

ДОДАТОК Б
Слайди презентації

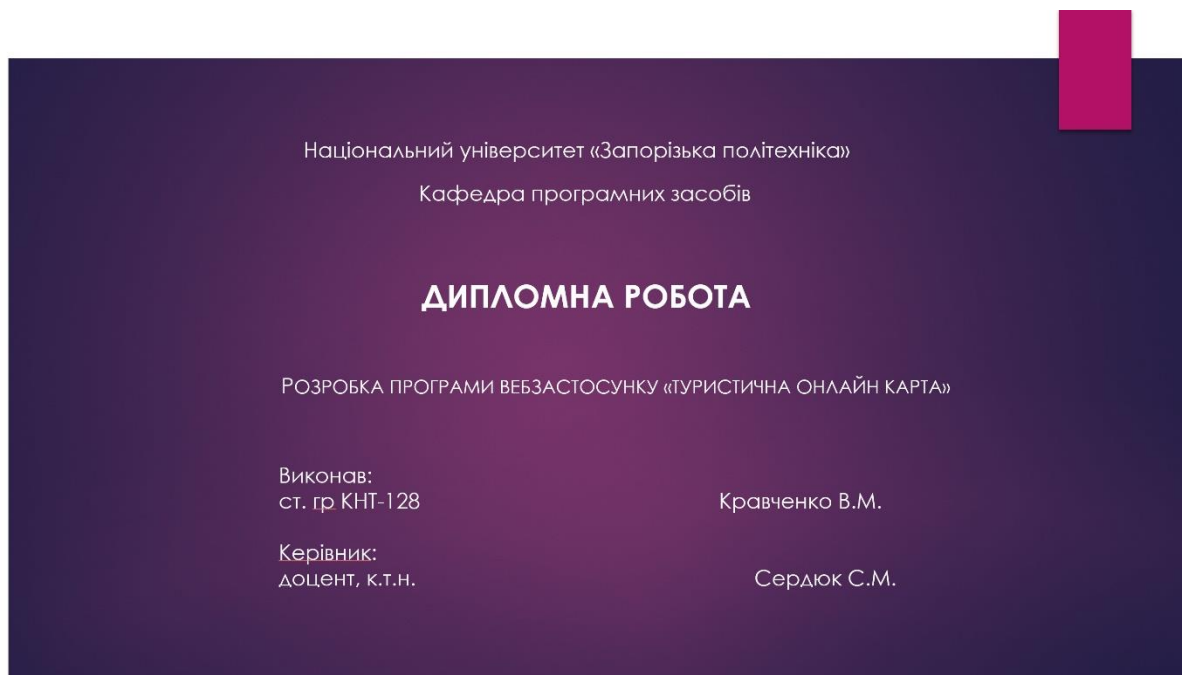


Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2

3

ЗАДАЧІ НА ДИПЛОМНУ РОБОТУ

- Провести аналіз предметної області;
- Проаналізувати існуючі програмні рішення;
- Визначити функціональні та технічні вимоги до розроблюваного застосунку;
- Розробка архітектури програми;
- Вибір інструментів розробки;
- Розробка програмного застосунку;
- Тестування розробленого застосунку;
- Розробка документації для програміста;
- Розробка документації для оператора;

Активуйте Windows
Щоб активувати Windows, перейдіть в розділ "Параметри".

Рисунок Б.3 – Слайд 3

4

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

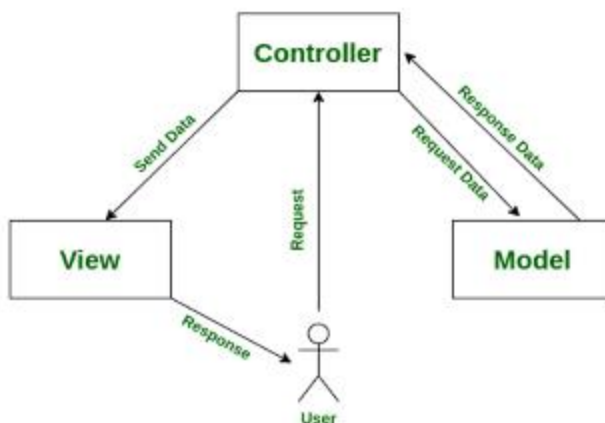
Критерії	Discovery.UA	Opentrip Map	Google Map
Стабільність роботи	+	+	-
Безкоштовність	+	+	+
Актуальність інформації	+	-	+-
Можливість залишати відгуки	+-	-	-
Зручний та зрозумілий UI	-	+	-
Анімації та інтерактивність елементів інтерфейсу	-	+	-

Активуйте Windows
Щоб активувати Windows, перейдіть в розділ "Параметри".

Рисунок Б.4 – Слайд 4

5

АРХІТЕКТУРА ПРОГРАМИ



Активація Windows
Щоб активувати Windows, перейдіть в розділ "Власність".

Рисунок Б.5 – Слайд 5

6

ДІАГРАМА ПРЕЦЕДЕНТІВ

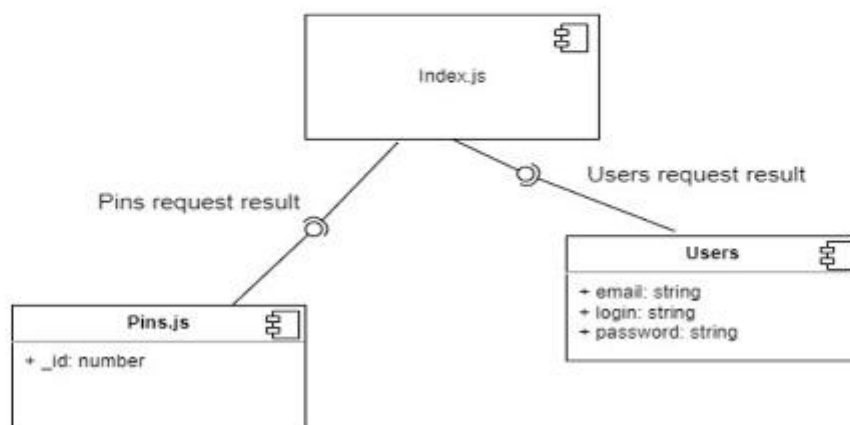


Активація Windows
Щоб активувати Windows, перейдіть в розділ "Власність".

Рисунок Б.6 – Слайд 6

7

ДІАГРАМА КОМПОНЕНТІВ КОНТРОЛЕРУ

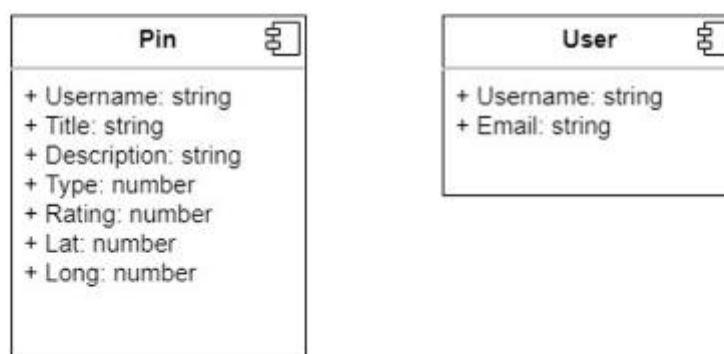


Автоматизація Windows
 "Немає автоматизації Windows, проблема в режимі "Панорама".

Рисунок Б.7 – Слайд 7

8

ДІАГРАМА КОМПОНЕНТІВ МОДЕЛІ



Автоматизація Windows
 "Немає автоматизації Windows, проблема в режимі "Панорама".

Рисунок Б.8 – Слайд 8

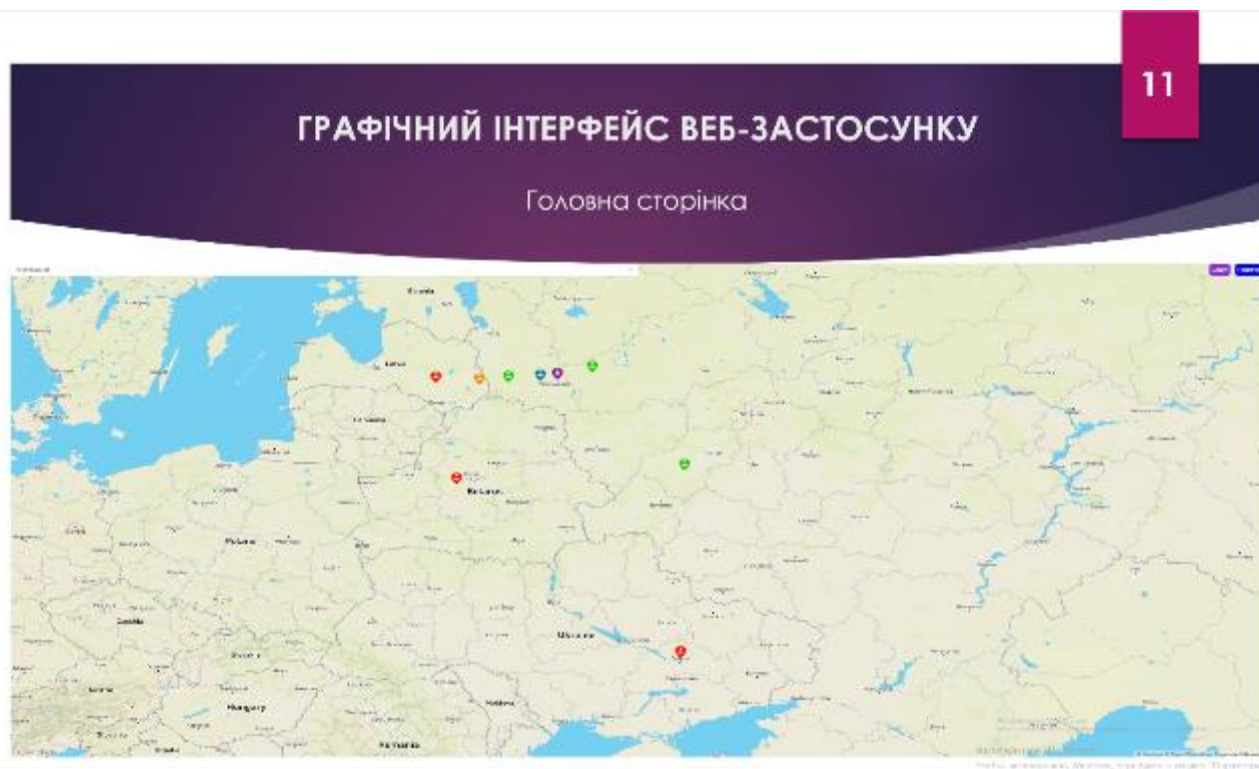


Рисунок Б.11 – Слайд 11

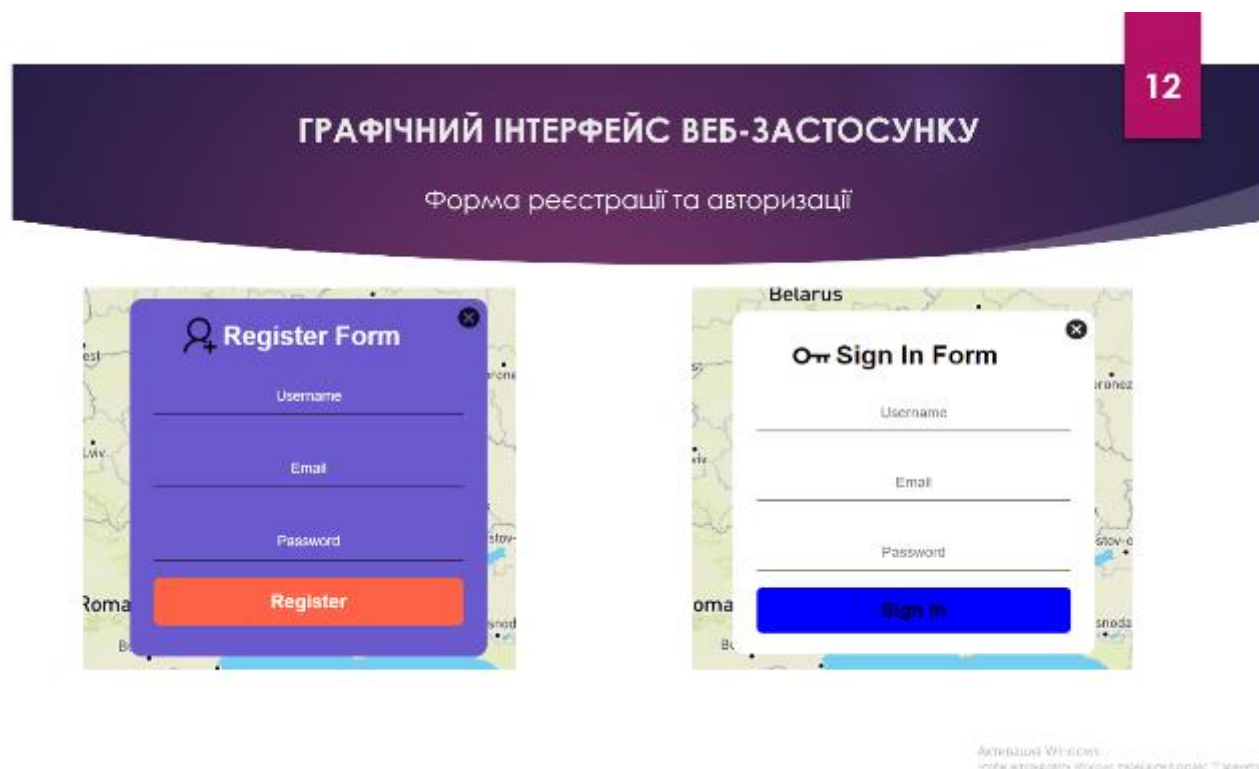
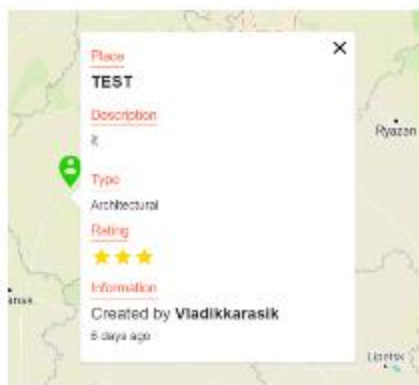


Рисунок Б.12 – Слайд 12

13

ГРАФІЧНИЙ ІНТЕРФЕЙС ВЕБ-ЗАСТОСУНКУ

Модальне вікно з інформацією о крапці



ArchiMagnum Windows
 Windows, which can be used to create, edit and delete places.

Рисунок Б.13 – Слайд 13

14

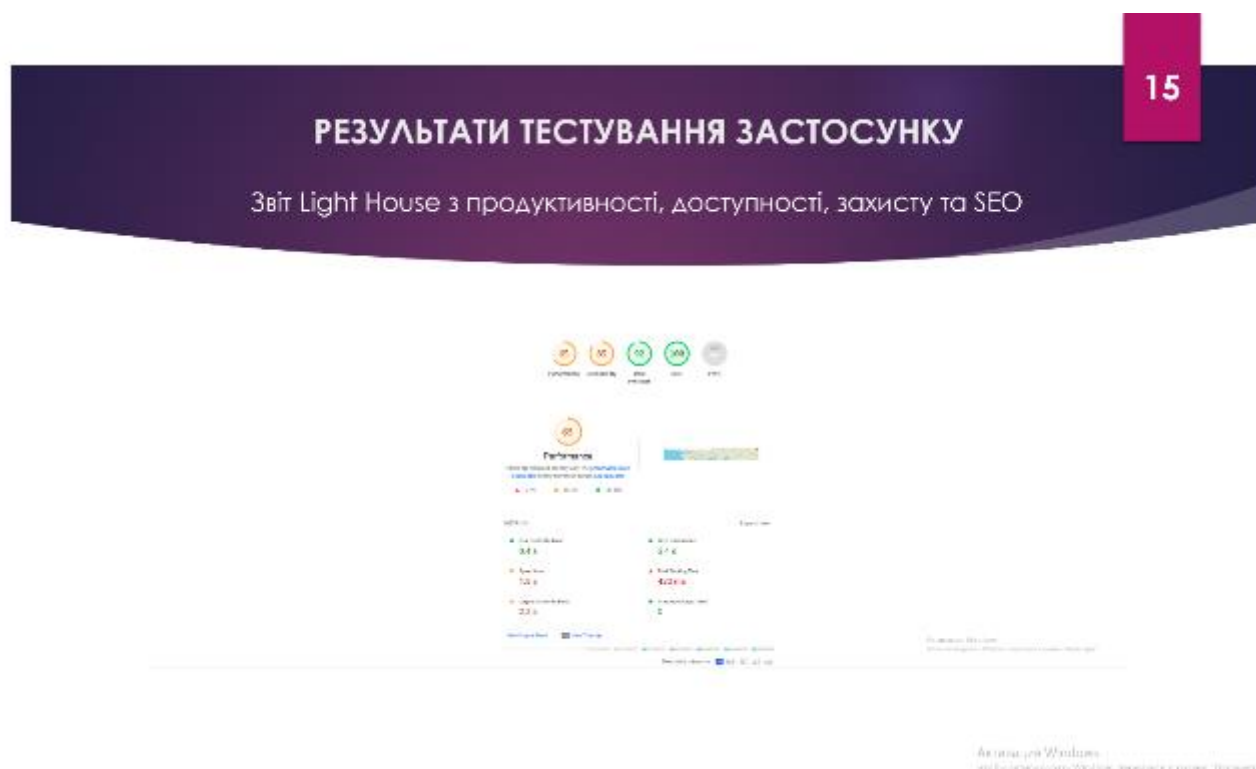
ГРАФІЧНИЙ ІНТЕРФЕЙС ВЕБ-ЗАСТОСУНКУ

Модальне вікно створення крапки



ArchiMagnum Windows
 Windows, which can be used to create, edit and delete places.

Рисунок Б.14 – Слайд 14



РисунокБ.15 – Слайд 15

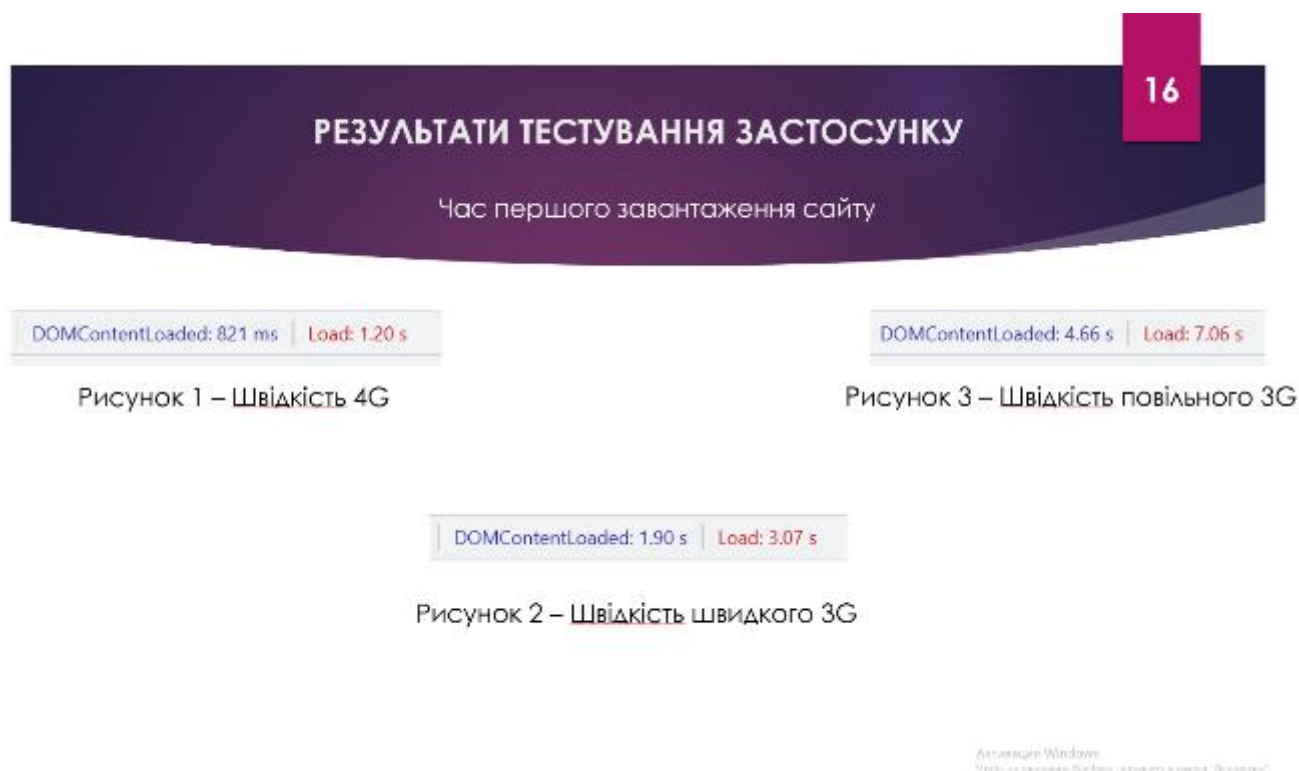


Рисунок Б.16 – Слайд 16

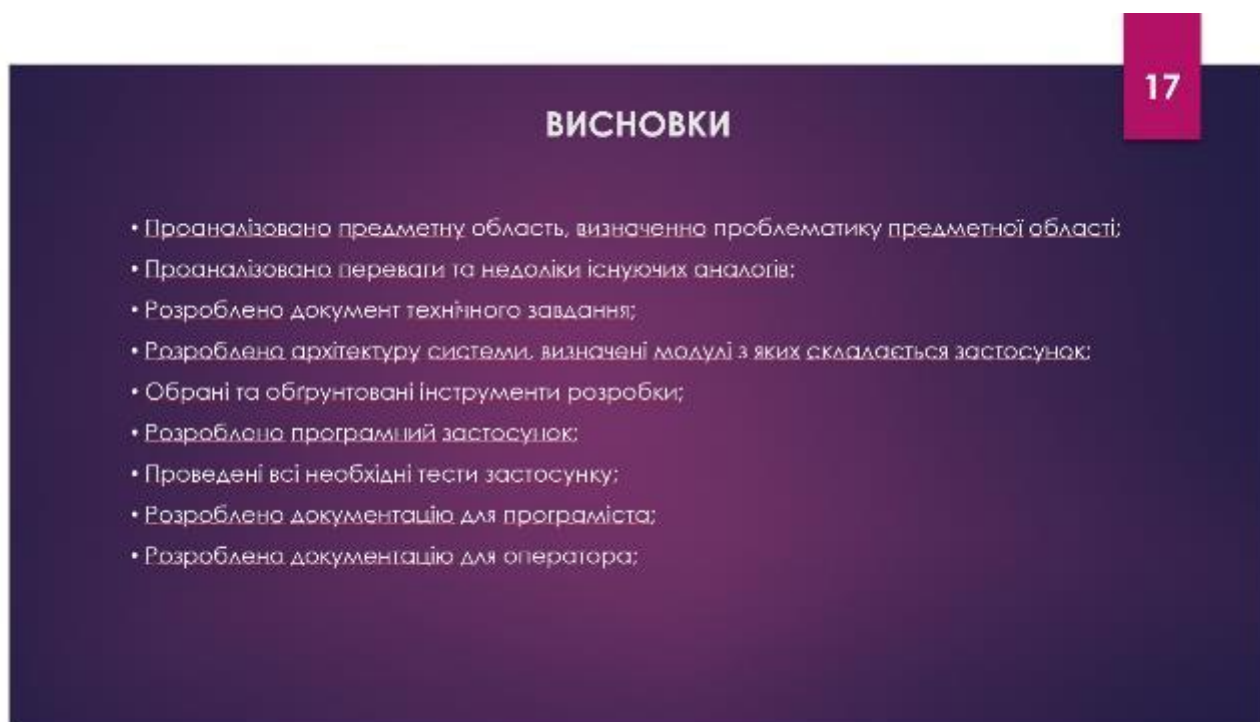


Рисунок Б.17 – Слайд 17



Рисунок Б.18 – Слайд 18