

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Гео-дані як інтерактивний елемент UI в сучасних вебзастосунках.

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-20-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Ісікова Микити
Олександровича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Омельчук Антон
Анатолійович, к.т.н.
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Омельчук Антон
Анатолійович, к.т.н.
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «22» травня 2024 № 15

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Ірпінь – 2024

Реєстрація № 34

27.05.2024

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

22.05.2024

(дата)



(підпис)

Результат захисту

5 (відмінно)
(оцінка)

06.06.2024

(дата)



(підпис)

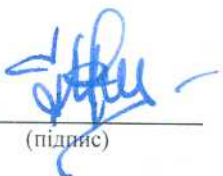
Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

С.М. Грищенко



(підпис)

В.О. Ніжегородцев

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		06.06.2024 Франчук В.М. <i>[підпис]</i> Філоненко М.М. <i>[підпис]</i> Грищенко С.М. <i>[підпис]</i> Ніжегородцев В.О. <i>[підпис]</i> Суховерхівка Л.П. <i>[підпис]</i>
2. Дотримання вимог щодо оформлення	від 0 до 10	8		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником - 46 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
91	A	5 (відмінно)	06.06.2024	Франчук В.М. <i>[підпис]</i> Філоненко М.М. <i>[підпис]</i> Грищенко С.М. <i>[підпис]</i> Ніжегородцев В.О. <i>[підпис]</i> Суховерхівка Л.П. <i>[підпис]</i>

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри  Філоненко
М.М.

«23» жовтня 2023

**Завдання на кваліфікаційну роботу
здобувачу вищої освіти**

Ісікову Микиті Олександровичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Гео-дані, як інтерактивний елемент UI в сучасних вебзастосунках.

Затверджена наказом ректора від «20» жовтня 2023 № 1411

2. Строк здачі закінченої роботи «20» травня 2024

3. Мета роботи Розробка програмного продукту з інтеграцією геоінформаційних систем та виконанням низькорівневих налаштувань візуалізації мап та їх окремих елементів.

Об'єкт дослідження Геоінформаційні системи.

Предмет дослідження Процеси візуалізації гео-даних та способи впливу на них.

Орієнтовна база дослідження Вебзастосунок для перегляду подій на території університету

4. Вихідні дані до роботи Гео-дані будівель університету, документація ГІС та фронт-енд фреймворків.

5. Перелік графічного матеріалу Блок-схеми, знімки екрану взаємодії з геоінформаційними системами, знімки екрану користувацького інтерфейсу вебзастосунку.

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ ПІДГРУНТЯ ЗАСТОСУВАННЯ КАРТОГРАФІЧНИХ БІБЛІОТЕК

1.1. Картографічні бібліотеки в контексті геоінформаційних систем

1.2. Процеси візуалізації інформації картографічних бібліотек

РОЗДІЛ 2. ВИКОРИСТАННЯ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ У ВЕБЗАСТОСУНКАХ

2.1. Інструменти візуалізації гео-даних та взаємодія з ними

2.2. Екстракція та попередні перетворення гео-даних

2.3. Розробка вебзастосунку

РОЗДІЛ 3. РЕТРОСПЕКТИВНИЙ АНАЛІЗ ТА ПРОПОЗИЦІЇ З ПОКРАЩЕННЯ ПРОГРАМНОГО ПРОДУКТУ

ВИСНОВОК.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактичн о
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.10.2023	До 15.10.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	29.12.2023	29.12.2023
3	II розділ (аналіз експериментальної частини про методики досліджень)	15.02.2024	15.02.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	15.04.2024	15.04.2024
5	Висновки	01.05.2024	01.05.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	06.05.2024	06.05.2024
7	Отримання відгуку керівника	28.05.2024	28.05.2024
8	Отримання зовнішньої рецензії	30.05.2024	30.05.2024

8. Дата видачі завдання «23» жовтня 2023

9. Керівник кваліфікаційної роботи


(підпис)

Омельчук А.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Омельчук А.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Ісіков М.О.
(прізвище, ініціали)

АНОТАЦІЯ

Гео-дані як інтерактивний елемент UI в сучасних вебзастосунках.

(назва кваліфікаційної (дипломної) роботи)

Мета роботи полягає в розробці програмного продукту з інтеграцією геоінформаційних систем та виконанням низькорівневих налаштувань візуалізації мап та їх окремих елементів.

Завданням роботи є:

- Класифікувати і визначити зони відповідальності різних типів геоінформаційних систем.
- Охарактеризувати процеси візуалізації інтерактивних мап у контексті вебзастосунків.
- Розглянути доступні методи реалізації інтерактивних елементів на мапах в рамках картографічних бібліотек.
- Розробити вебзастосунок з активним використанням гео-даних як частиною динамічного користувацького інтерфейсу.
- Обґрунтувати прийняті підходи реалізації та виконати ретроспективний аналіз процесу інтеграції з точки зору розробника.

Предметом дослідження є процеси візуалізації гео-даних та способи впливу на них.

Об'єктом дослідження є геоінформаційні системи.

Базою дослідження виступає вебзастосунок для перегляду подій на території університету

За результатами дослідження сформульовані настанови щодо планування та реалізації інтерактивних мап в вебзастосунках на базі бібліотеки Mapbox GL JS.

Одержані результати можуть бути використані розробниками та підприємствами для розробки програмних продуктів з застосуванням геоінформаційних систем.

Кваліфікаційна робота містить 44 сторінки, 1 таблицю, 11 рисунків, список використаних джерел містить 37 найменувань, 2 додатки.

Ключові слова: гео-дані, геоінформаційні системи, веб-розробка, просторові дані, користувацький інтерфейс.

Рік виконання кваліфікаційної роботи 2024

Рік захисту роботи 2024

ANNOTATION

Geo-data as an Interactive Element of UI in Modern Web Applications.

(title of the qualification (diploma) work)

The goal of the work is to develop a web applicatoion with integrated geoinformation system and to perform low-level changes to visualization settings for maps and their elements.

The objectives of the work are:

- To classify and define the areas of responsibility of different types of geoinformation systems..
- To characterize the processes of visualizing interactive maps in the context of web applications.
- To consider available methods for implementing interactive elements on maps within cartographic libraries.
- To develop a web application with active use of geo-data as part of a dynamic user interface.
- To substantiate the chosen implementation approaches and conduct a retrospective analysis of the integration process from a developer's perspective.

The subject of the research are the processes of geo-data visualization and the methods influencing them.

The object of the research is geoinformational systems.

The basis of the research is a web application for viewing events on the territory of the university.

Based on the research results, guidelines for planning and implementing interactive maps in web applications based on the Mapbox GL JS library have been formulated.

The obtained results can be used by developers and companies to create software products using geoinformation systems.

The qualification work contains 44 pages, 1 table, 11 figures, a list of sources containing 37 titles, and 2 appendices.

Keywords: geo-data, geoinformation systems, web development, spatial data, user interface.

Year of completion of the qualification work: 2024

Year of work defense: 2024

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ПІДГРУНТЯ ЗАСТОСУВАННЯ КАРТОГРАФІЧНИХ БІБЛІОТЕК.....	5
1.1. Картографічні бібліотеки в контексті геоінформаційних систем.....	5
1.2. Процеси візуалізації інформації картографічних бібліотек.	8
РОЗДІЛ 2. ВИКОРИСТАННЯ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ У ВЕБЗАСТОСУНКАХ.....	19
2.1. Інструменти візуалізації гео-даних та взаємодія з ними.....	19
2.2. Екстракція та попередні перетворення гео-даних.	25
2.3. Розробка вебзастосунку.....	29
РОЗДІЛ 3. РЕТРОСПЕКТИВНИЙ АНАЛІЗ ТА ПРОПОЗИЦІЇ З ПОКРАЩЕННЯ ПРОГРАМНОГО ПРОДУКТУ.....	40
ВИСНОВКИ.....	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	47
Додаток А.....	51
Додаток Б.	55

ВСТУП

Сучасні вебзастосунки дедалі активніше використовують інтерактивні мапи для різноманітних цілей: від полегшення навігації та покращеного пошуку на базі гео-даних до відображення та обробки великих масивів даних. Веб-розробник, перед яким стоїть задача інтеграції інтерактивної мапи у вебсайт, має розуміти всі особливості візуалізації та обробки гео-даних для об'єктивного сприйняття процесу розробки відповідно до поставлених задач та повинен мати розуміння процесів для правильного вибору інструментів для вирішення конкретної проблеми.

Гео-дані та геоінформаційні системи активно розглядаються в науковій літературі, пов'язаній з агропромисловістю або маркетингом. Але в обох випадках розглядається більше аналіз цих даних, ніж їх динамічна візуалізація та покращення досвіду користувача в процесі взаємодії з вебзастосунками.

Мета: Розробка програмного продукту з інтеграцією геоінформаційних систем та виконанням низькорівневих налаштувань візуалізації мап та їх окремих елементів.

Завдання кваліфікаційної роботи. Класифікувати і визначити зони відповідальності різних типів геоінформаційних систем. Охарактеризувати процеси візуалізації інтерактивних мап у контексті вебзастосунків. Розглянути доступні методи реалізації інтерактивних елементів на мапах в рамках картографічних бібліотек. Розробити вебзастосунок з активним використанням гео-даних як частиною динамічного користувацького інтерфейсу. Обґрунтувати прийняті підходи реалізації та виконати ретроспективний аналіз процесу інтеграції з точки зору розробника.

Об'єктом дослідження є бібліотеки, що виконують функції візуалізації гео-даних. Предметом дослідження є процеси візуалізації гео-даних та способи впливу на ці процеси.

Дослідження здійснювалось в декілька способів. Для коректної класифікації

геоінформаційних систем здійснювалось ознайомлення з офіційними документаціями низки найбільш розповсюджених ГІС. Розглядались статті, монографії та інші засоби інформації, які описували технічні аспекти створення та відтворення гео-даних. Спроектовано та реалізовано програмний продукт, який висвітлює своїм прикладом сильні і слабкі сторони дослідження та дозволяє провести конструктивний аналіз.

Враховуючи невеликий масштаб побудованого програмного продукту, досвід практичної частини дослідження доречно буде застосовувати до таких сфер діяльності як цифрова комерція або вебсайти окремих організацій. Реалізований програмний продукт вирішує поставлену перед ним задачу, семантично реагує на вхідні сигнали користувача та підтримує зв'язок між інтерактивною мапою та потоками даних застосунку в цілому. Його практична цінність полягає насамперед в роз'ясненні механізмів впливу інструкцій розробника на візуалізовані гео-дані. Проведене дослідження не включає використання таких просунутих функцій як навігація в реальному часі, динамічне оновлення інформації в залежності від розташування користувача або методи пошуку гео-даних через інтерфейс вебзастосунку.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ПІДГРУНТЯ ЗАСТОСУВАННЯ КАРТОГРАФІЧНИХ БІБЛІОТЕК.

1.1. Картографічні бібліотеки в контексті геоінформаційних систем.

Картографічні платформи та бібліотеки є частиною більших інформаційних систем – так званих Геоінформаційних систем (ГІС). Для того щоб правильно класифікувати картографічні платформи та бібліотеки необхідно чітко визначити функції географічних інформаційних систем та вказати сфери застосування програмного забезпечення згідно зазначених парадигм.

Географічна інформаційна система – це система збору, зберігання, аналізу й графічної візуалізації просторових даних і пов’язаної з ними інформації про об’єкти, розташовані на місцевості. Географічна інформаційна система у більш вузькому змісті – як інструмент у вигляді програмного продукту, що дозволяє користувачам шукати, аналізувати й редагувати цифрові карти, а також одержувати додаткову інформацію про об’єкти[1].

До найважливіших функцій геоінформаційних систем відносять отримання даних, зберігання даних, керування даними, аналіз даних та візуалізацію даних. Прикладами ГІС, що підтримують всі 5 з вищеперелічених функцій є ArcGIS, Quantum GIS, Google Earth Engine. Такі системи ставлять за мету перш за все надання інструментарію для комплексного аналізу великих масивів гео-даних та їх подальшої обробки.

Розглядаючи ГІС в більш широкому сенсі, яскравим прикладом їх застосування в сільському господарстві. Зокрема геоінформаційні системи допомагають в вирішенні таких управлінських завдань, як інвентаризація сільськогосподарських ділянок із формуванням кадастрів; охорона ґрунтових ресурсів, оцінка земель та планування землекористування; оцінка можливих наслідків стихійних лих (повені, посухи, пожежі, зсуви тощо); просторове цифрове управління підприємствами та угіддями; точне землеробство,

інтелектуальне землеробство та збирання даних за допомогою просторових цифрових технологій; моніторинг функціонування підприємств та агроєкосистем; забезпечення надходження інформації, пов'язаної із захистом рослин, боротьби з бур'янами, шкідниками та хворобами; визначення наявності та достатності поживних речовин ґрунту та управлінню родючістю; управління тваринництвом та пасовищами; картування земель за даними моніторингу[2].

З боку веб технологій Географічні інформаційні системи виконують широкий спектр завдань. в рамках навігаційних додатків широко використовуються, як частини служб Google Maps, Mapbox, Waze. Вони забезпечують навігацію в реальному часі, планування маршрутів та оновлення трафіку. Також багато компаній застосовують навігаційні функції в сфері громадського транспорту.

Підприємства в сфері громадського транспорту використовують геоінформаційні системи для пропонування розкладів в реальному часі, для оптимізації маршрутів і відстеження місцезнаходження.

Також існують т. зв. Location-Based Services (LBS), тобто такі, що базуються на місцезнаходженні. Вони поділяються на два основних напрямки. Геокодування та зворотне геокодування. Він полягає у перетворенні адрес у географічні координати та навпаки – для пошуку на основі місцезнаходження. Прикладом цього є служба доставки або служба таксі. Другий тип – пошук за близькістю. До таких послуг належить пошук найближчих бізнесів, точок інтересу і тому подібного на основі поточного місцезнаходження користувача.

До інших сфер застосування ГІС в вебзастосунках належать: додатки для клімату та прогнозів погоди, попередження про катастрофи, сервіси пов'язані з нерухомістю та управлінням нерухомого майна та ін.

Переважає більшість вебзастосунків не потребує такого широкого інструментарію зберігання та аналізу даних, які надають географічні інформаційні системи. Замість цього такі вебзастосунки використовують два

типи інструментів, а саме картографічні платформи та картографічні бібліотеки. Картографічні бібліотеки певною мірою можна вважати частиною картографічних платформ, але головними відмінностями картографічних бібліотек є швидка інтеграція, модульність та універсальність.

На основі порівняння таких картографічних платформ, картографічних бібліотек та геоінформаційних систем, як Mapbox[3], Google Maps Platform[4], Leaflet[5], Mapbox GL JS, ArcGIS[6], QGIS[7] було створено порівняльну таблицю (Додаток 1), на основі якої можна сформулювати чіткі визначення картографічних платформ та картографічних бібліотек.

Картографічні платформи – це комплексні служби, які пропонують розробникам набір інструментів і API для створення власних карт і інтеграції геопросторових даних у свої програми. Ці платформи часто надають хмарні послуги для розміщення, обробки та візуалізації даних. Вони можуть містити різні функції, такі як геокодування, маршрутизація, інтеграція даних у реальному часі тощо.

Картографічні бібліотеки — це легкі модульні рішення, призначені для вбудовування інтерактивних карт у вебсторінки чи програми. Вони надають основні функції для візуалізації карти, панорамування, масштабування та додавання шарів. Ці бібліотеки часто розширюються за допомогою плагінів або додаткових модулів, що дозволяє розробникам розширювати свої можливості за потреби.

Враховуючи що в даній роботі гео-дані розглядаються саме з точки зору користувацького інтерфейсу, то найбільша увага приділятиметься саме службам типу картографічних бібліотек, адже за візуалізацію даних для користувача використовуються саме вони.

1.2. Процеси візуалізації інформації картографічних бібліотек.

Дані, якими оперують ГІС можна розділити на дві категорії: дані з просторовою прив'язкою, які представлені векторними та растровими формами (включаючи зображення), і таблиці атрибутів, які представлені в табличному форматі.

У групі просторових прив'язаних даних дані ГІС можна додатково класифікувати на два різні типи: векторні та растрові. Більшість програмного забезпечення ГІС зосереджено на використанні та маніпулюванні векторними базами гео-даних із доданими компонентами для роботи з растровими базами гео-даних.

Растрові зображення часто використовуються для картографування поверхонь, наприклад топографії, і можуть використовуватися для кількісних змін ландшафту. Ще одне поширене використання растрових даних як фоновий шар. В якості базового шару при створенні карт часто використовуються аерофотозйомки, супутникові знімки та растрові зображення рельєфу.

Ця категорія даних також включає аерофотознімки та супутникові знімки. Типи растрових зображень можуть бути у форматах geoTIFF, TIFF, JPG, PNG, GIF, BMP та інших форматах файлів зображень. Ці формати файлів часто мають другий файл, який називається «файлом світу», який містить інформацію про проекцію растрового зображення[8].

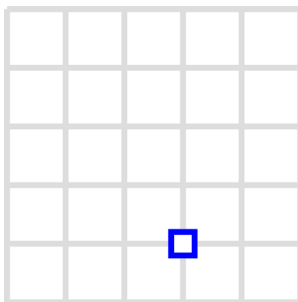
GeoJSON — це формат обміну геопросторовими даними на основі нотації об'єктів JavaScript (JSON). Він визначає кілька типів об'єктів JSON і спосіб їх поєднання для представлення даних про географічні об'єкти, їхні властивості та просторові екстенти. GeoJSON використовує географічну систему координат Всесвітньої геодезичної системи 1984 року та одиниці десяткових градусів[9].

GeoJSON підтримує такі геометричні типи (Рис. 1.1):

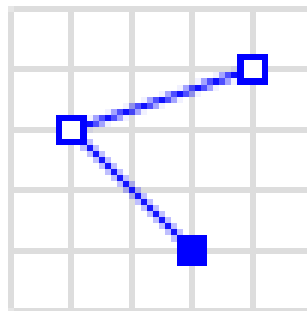
- Point

- MultiPoint
- LineString
- MultiLineString
- Polygon
- MultiPolygon

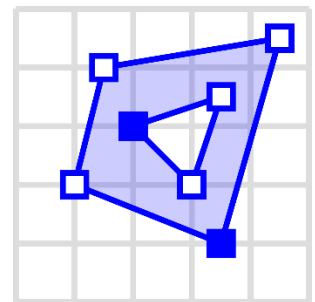
Ці геометричні типи об'єднуються в геометричний конструкт Position.



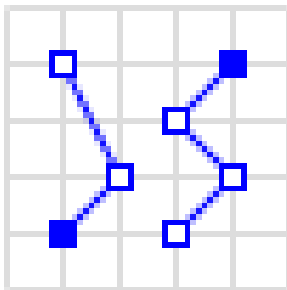
а) Point



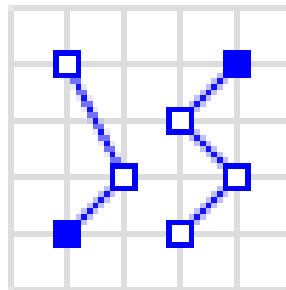
б) LineString



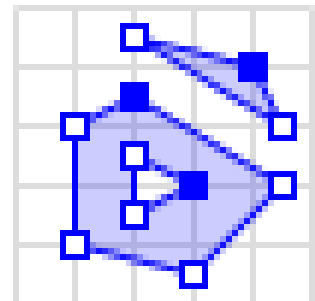
в) Polygon



г) MultiPoint



д) MultiLineString



е) MultiPolygon

Рис. 1.1 Візуальне представлення основних геометричних типів GeoJSON.

Окремим геометричним типом варто зазначити т.зв. GeometryCollection. Його особливість полягає в тому що він може бути різномірною композицією менших геометричних об'єктів. Наприклад, геометричний об'єкт у формі малої літери «і» може складатися з одного Point та одного LineString. Хоча об'єкт GeometryCollection не має члена «coordinates», він має координати: координати

всіх його частин належать колекції.

Альтернативою GeoJSON є стандарт KML. KML — це мова XML, орієнтована на географічну візуалізацію, включаючи анотації карт і зображень. Географічна візуалізація включає не тільки представлення графічних даних на земній кулі, а й керування навігацією користувача в сенсі того, куди йти і куди дивитися[10].

Підбиваючи підсумки стандартів зберігання гео-даних, важливо розуміти що обидва стандарти KML та GeoJSON підтримуються більшістю сучасних GIS. Але менший розмір файлів GeoJSON та більш вільна структура робить цей стандарт більш оптимальним для використання у вебзастосунках. В свою чергу KML спроектований перш за все для використання з Google Earth, що в контексті інтеграції інтерактивних карт в вебзастосунки не є актуальним.

В основі процесів візуалізації у картографічних бібліотеках лежить розділення всього вмісту карти на окремі шари або т. зв. Layers. Кожна бібліотека має власний підхід для розподілу та категоризації вмісту на шари, але існуює певний порядок якого певною мірою притримується більшість бібліотек.

Розглядаючи бібліотеки Leaflet, Mapbox GL JS та OpenLayers[11] можна дійти висновку щодо цієї структури – першим шаром майже завжди виступає растрові (фонові) тайли. За ним іде шар, який включає мітки на мапі, які також називають «Marker» або «Symbol».

Після шару з мітками відображається шар з векторними даними. Зазвичай це комплексні об'єкти згідно стандарту GeoJSON, наприклад LineString, Polygon та ін. На цьому етапі відображення спостерігаються суттєві відмінності між розглянутими бібліотеками. Прикладом цього є те, що у Mapbox GL JS є окремі типи шарів для векторних ліній та об'ємних об'єктів. Натомість Leaflet має інші типи шарів, що відповідно може викликати розбіжності у відображенні даних якщо в рамках одного проекту будуть застосовуватись різні

бібліотеки.

Шари, які відображаються наступними – зазвичай додаються вже на етапі розробки і не включені в базову конфігурацію бібліотеки. Кожна бібліотека надає можливість створювати, додавати та видаляти шари до кореневого об'єкту мапи за допомогою відповідних методів. Такі шари завжди додаються поверх всіх попередніх шарів та можуть містити GeoJSON дані.

Варто також зазначити що в певних бібліотеках з більш просунутим функціоналом є можливість створювати особливий шар типу HeatMap (Теплові мапи). Такі теплові мапи дозволяють переглядати концентрацію великих масивів даних на площині. І це певною мірою схоже на кластеризацію. Але, на відміну від кластеризації, теплова карта не може бути використана для категоризації одиничних даних і слугує в основному для представлення глобальної картини замість конкретної та чіткої інформації в певному секторі мапи.

При використанні теплових мап можна обмежити їх відображення на певному значенні наближення. Після цього значення можна відобразити шар з одиничними елементами або звернутися до методу кластеризації.

Одним з найбільш яскравих прикладів растрової візуалізації гео-даних є т. зв. Теплові мапи (Heat maps). В класичному розумінні теплові мапи являють собою двовимірні таблиці чисел у вигляді відтінків кольорів. Це популярна техніка побудови графіків у біології, яка використовується для зображення експресії генів та інших багатовимірних даних. Завдяки щільному та інтуїтивно зрозумілому відображенню теплові карти добре підходять для представлення високопродуктивних даних. На екрані можуть бути відображені сотні рядків і стовпців. Теплові карти в основному покладаються на кодування кольорів і зміну порядку рядків і стовпців. Коли будь-який із цих компонентів скомпрометовано, корисність візуалізації страждає[12].

Теплові мапи є в переважній більшості ГІС, наприклад у QGIS для таких

цілей є спеціалізований плагін «Heatmap Plugin»[13]. В тому числі теплові мапи можна створювати за допомогою картографічних платформ Mapbox та OpenLayers.

Відображення різних шарів на інтерактивній мапі в залежності від рівню наближення теж може зіграти важливу роль в оптимізації обчислювальних ресурсів на стороні користувача. Правильне комбінування різних шарів відображення даних на різних рівнях наближення може заощадити обчислювальні ресурси, що особливо важливо для вебзастосунків на мобільних або застарілих платформах. Як розробник, важливо розуміти в якій послідовності відбувається візуалізація шарів і яка бібліотека може виконати поставлені задачі ефективніше.

Для того, щоб використовувати растрові фотографії для побудови інтерактивних мап, необхідно використовувати певні картографічні проекції. В залежності від типу проекції мапа буде виглядати по-різному. Також проекція буде впливати на правильність відображення координат, форм та розмірів географічних об'єктів.

Найбільш розповсюдженою проекцією для карт в вебзастосунках є Web Mercator. Ця проекція є адаптацією однієї з найвідоміших географічних проекцій Меркатора[12]. Така проекція використовується в переважній більшості картографічних бібліотек і є оптимізованою саме для використання у браузерях та веб-середовищі в цілому.

Mapbox GL JS, Leaflet та OpenLayers підтримують різні типи проекцій для картографічних проекцій. Mapbox GL JS здебільшого використовує Web Mercator, але також дозволяє налаштовувати проекції через API. Leaflet використовує Web Mercator і може налаштовувати інші проекції за допомогою плагінів. OpenLayers підтримує найширший спектр проекцій, включаючи стандартні та спеціальні проекції, які можуть бути налаштовані за допомогою Proj4js. Ці можливості дозволяють обирати відповідну проекцію для кожного конкретного завдання, забезпечуючи точність та відповідність вимогам

користувачів.

Всі провайдери гео-даних спираються на тайли певних розмірів та масштабів, в залежності від значення параметру «zoom». Цей параметр відповідає за ступінь наближення і має початкове значення – 0, де при значенні 0 один тайл відображає всю поверхню Землі. Всі наступні рівні зуму містять кількість тайлів що дорівнює 2^z , де z – рівень зуму (Рис. 1.2)

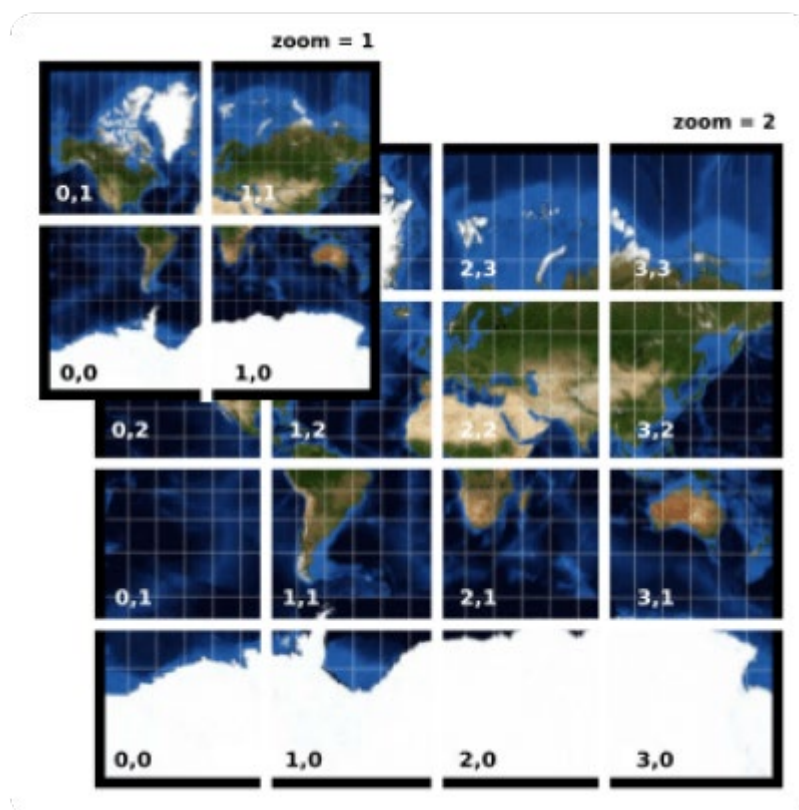


Рис. 1.2. Демонстрація тайлів на рівнях зуму 1 та 2.

Стандартно для розкладу растрових файлів на тайли використовується проекція Веб Меркатор. Для перетворення растрових файлів використовують програмний продукт PROJ, джерело якого є відкритим. PROJ представляє собою загальне програмне забезпечення для перетворення координат, яке перетворює геопросторові координати з однієї системи відліку координат (CRS) в іншу. Це включає в себе картографічні проекції, а також геодезичні перетворення[14].

Mapbox на відміну від Leaflet або Google Maps Platform, які є сучасними та широко застосовуваними сервісами, що потенційно можуть виконувати такі ж самі задачі, мають певні недоліки, і ці недоліки виражені в основному специфікою їх використання. Спершу розглянемо Leaflet. Leaflet є виключно картографічною бібліотекою і не має власної бекенд інфраструктури або не може виконувати функції картографічних платформ.

Це призводить до того, що всі операції з даними, а також оптимізація візуалізації карт має відбуватись виключно на стороні клієнта, що особливо в контексті мобільних пристроїв не завжди є бажаним шляхом вирішення поставлених задач. Натомість перевагою Leaflet є відкрита структура самої бібліотеки, а також широка підтримка спільноти, велика кількість навчальних матеріалів та широке описання API бібліотеки. До переваг також слід віднести можливість створювати негеографічні карти і не прив'язуватись до істинної сферичної меркатор проєкції.

Певною мірою недоліком, але в той же час і перевагою Leaflet є візуалізація карт за допомогою переважно HTML-блоків та за допомогою SVG. З одного боку, це дозволяє менше покладатися на зв'язок з інтернетом і не покладатися на реальність. Картографічні платформи для того, щоб візуалізувати складний контент.

Але з іншого боку, це є недоліком, особливо в ситуаціях, коли інтернет-трафік не є проблемою і якщо на інтерактивній мапі одночасно знаходиться знадто багато шарів, або елементів. Напротив цього, деякі конкуренти застосовують візуалізацію через Canvas, який в цій ситуації є більш легким. З точки зору використання пам'яті, але потребує більш ретельного догляду за оптимізацією непотрібних оновлень вмісту інтерактивної карти.

Велика перевага Leaflet є його незалежність від картографічних платформ та легкість у порівнянні з іншими схожими бібліотеками. Зокрема, в порівнянні з Mapbox або OpenLayers. Загалом, Leaflet підтримує широкий спектр функцій взаємодії з інтерактивною картою. Тобто це такі дії як Hover, Click, Pan, Zoom

та інші. Leaflet підтримує всі базові типи GeoJSON даних і має вичерпну документацію, хоча і не є порівняно крупною бібліотекою сам по собі. Також його великою перевагою є можливість створення плагінів, які спрощують персоналізацію таких карт.

Розглянувши Leaflet, можна зробити висновок, що він має достатній функціонал для використання в кваліфікаційному проєкті. Зокрема, в відношенні до персоналізації шарів та інших елементів інтерактивної мапи за допомогою хуків. Але суттєвим недоліком, який значно може ускладнити роботу по мірі росту проєкту, є проблема з тим, що у Leaflet немає вбудованих засобів для взаємодії з картографічними платформами.

В довгостроковій та середньостроковій перспективі це призведе до того, що розробнику потрібно буде покладатись або на власні розробки з інтеграції інтерактивних мап, або спиратись на готові open-source-плагіни, які були створені іншими розробниками. На іншій стороні спектру геоінформаційних систем, які можуть бути використані для виконання кваліфікаційного проєкту, стоїть Google Maps Platform, або скорочено GMP. GMP має абсолютно протилежну до Leaflet методологію, яка полягає в всіх налаштуваннях в веб-клієнті Google Maps Platform.

Це означає, що втручання розробника в код візуалізації карт від Google є мінімальним, що є недоліком в разі потреби створювання специфічних карт, але є великою перевагою, особливо в таких сферах, як e-commerce, та в ситуаціях, коли час розробника є обмеженим, або підприємству необхідно якісно та швидко рішення. Для використання з іншою enterprise інфраструктури та легкої інтеграції. Враховуючи все вище сказане, Google Maps Platform є гарним варіантом для інтеграції інтерактивних мап, але одним з найбільших її недоліків є висока цінова політика в порівнянні з конкурентами.

Окремо варто звернути увагу на те, що високі ціни перекриваються найбільшим рівнем підтримки з усіх конкурентів та Якісно описано документацію і широкий інструментарій no-code модифікацій тем і інтеграції

карт. Також GMP використовує систему оплати pay-as-you-go, що означає залежність вартості послуг від кількості унікальних користувачів та кількості індивідуальних запитів до API Google Maps. Крім того, Google Maps Platform акцентують увагу на тому, що серед конкурентів GMP має передові інструменти для обробки великих даних та їх аналізу і візуалізацію.

Для деяких ситуацій Google Maps Platform є вирішальним чинником переваги в сторону цієї геоінформаційної системи. Підсумувавши, ця платформа має широкий функціонал персоналізації інтерактивних карт. Зокрема, одна з найбільших її переваг є широкий інструментарій з обробки та візуалізації великих масивів даних, безпрецедентне лідерство в сфері швидкої інтеграції карт та вичерпної документації.

З недоліків можна виділити сувору ціннову політику, а також велике обмеження власній персоналізації в плані візуалізації інтерактивних карт. Тобто вся інформація, яка стосується Google Maps Platform має проходити обробку на сервісах Google і надходити клієнтові виключно для візуалізації без подальшої обробки отриманих даних для зміни в візуальному результаті.

Розглянувши Leaflet та Google Maps Platform, можна зробити висновок, що обидві платформи виконують кардинально різні ролі, мають різну цільову аудиторію та кардинально різні випадки використання. Хоча обидві геоінформаційні системи з точки зору кінцевого користувача є дуже схожі між собою і іноді використовують ідентичні гео-дані для візуалізації. Повертаючись до Leaflet, основним його недоліком є його обмеженість до виключно візуалізації гео-даних і відсутність будь-якого додаткового інструментарію для їх обробки.

Натомість GMP має протилежну проблему, яка полягає в потужних інструментах обробки гео-даних, але відносно обмеженим інструментарієм для персоналізації візуалізації цих даних.

Тепер розглянемо інфраструктуру Mapbox. Mapbox надає широкий

інструментарій продуктів, пов'язаних з обробкою візуалізацією та персоналізацією гео-даних. А також виступає провайдером гео-даних, які можуть використовувати інші геоінформаційні системи. Mapbox надає відносно малий опис документації та малу кількість прикладів для інтеграції всіх візуальних ефектів. Це є одним з найбільших його недоліків. В порівнянні з Leaflet, Mapbox втрачає достатньо мало переваг модульності. В основному, зберігаючи її. Але на відміну від Leaflet, Mapbox необхідне постійне з'єднання з його картографічною платформою та онлайн-сервісами. Так само, як в GMP, це робиться за допомогою персоналізованого API-ключа.

Mapbox має майже ідентичну до GMP систему оплати Pay-As-You-Go. Але цінова політика Mapbox є більш ощадливою. Великим недоліком Mapbox є обмежена офіційна документація. Але це пов'язано з швидким розвитком інфраструктури Mapbox, великою кількістю пропонованих продуктів та орієнтація на внутрішні сервіси з обробки гео-даних. Крім того Mapbox заохочує інтеграцію в існуючий API, на відміну від використання як окремої бібліотеки з можливістю надбудови. Іншими словами, недоліком Mapbox є обмежені методи створення персоналізованого контенту на низькому рівні, тобто саме зі сторони розробників.

На відміну від Leaflet не дозволяється створювати плагіни, які могли б напряму взаємодіяти з інфраструктурою Mapbox. Якщо GMP ставить за мету аналіз великих даних, то Mapbox як основний пріоритет ставить надання рішень для комерційних підприємств. Зокрема в таких сферах як логістика, погода, e-commerce, нерухомість та навігація.

Плюсом Mapbox є його універсальність та легкість взаємодії між внутрішніми продуктами Mapbox між собою. Взявши до уваги вищезазначені недоліки та переваги трьох розглянутих геоінформаційних систем, було зроблено рішення взяти за основу саме Mapbox, а конкретніше інструмент Mapbox для візуалізації гео-даних під назвою Mapbox GL JS.

Очевидно, що таким чином вони докладають зусиль, щоб клієнти

використовували існуючі інфраструктуру і не використовували власні інструменти персоналізації та використовували готові рішення для бізнесів, які пропонує цей сервіс.

РОЗДІЛ 2. ВИКОРИСТАННЯ ГЕОІНФОРМАЦІЙНИХ СИСТЕМ У ВЕБЗАСТОСУНКАХ.

2.1. Інструменти візуалізації гео-даних та взаємодія з ними.

Переважна більшість сучасних вебзастосунків розробляється за допомогою веб-фреймворків. Прикладами таких веб-фреймворків є React, Angular, Vue.js та інші. Наприклад, Svelte, Solid і так далі. Але перша трійка є найбільш затребуваною в індустрії і вважається еталоном серед застосування у веб-розробці[15].

Зокрема, перевагою React є легкий порт вебзастосунків на мобільні платформи за допомогою React Native. Всі три з вищеперелічених бібліотек і фреймворків мають в своїй основі модульний підхід до виконання коду. І найголовніше, надають можливість створювати окремі компоненти і організовувати структуру файлів відповідно до цих окремих компонентів, які можуть функціонувати незалежно один від одного. Але в той самий час використовувати одні і ті ж самі джерела даних і взаємодіяти між собою за допомогою специфічних методів обміну даних. Для вибору потрібного фреймворку одним з ключових факторів є визначення цілей застосунку, визначення розмірів застосунку або кількості сторінок та компонентів в ньому.

Крім того, одним з найважливіших моментів в цьому процесі є вибір відповідно до потоків даних в рамках вебзастосунку. Зокрема в React немає нативної стандартизованої та налагодженої системи обміну даних між компонентами. Натомість кожна окрема команда розробників застосовує свої власні, застосовує різні стейт-менеджери в залежності від преференцій команди та поставлених задач.

В рамках кваліфікаційної роботи буде достатньо звичайного паттерна Pub/Sub[16], який надає можливість підписуватись на події та повідомляти компоненти-підписники про зміни даних. В рамках React реалізувати це просто,

адже є відповідні до цього хуки, які допомагають в реалізації подібної архітектури. Альтернативою до цього первісного підходу є React Redux[17], в той же час React Redux вважається гарною практикою в рамках розробки на фреймворку React особливо через те, що він вже інтегрований в великі кількості вебзастосунків та має постійні оновлення і велику спільноту, яка підтримує цей проєкт.

З більш нових альтернатив можна виділити Zustand[18]. Це теж є бібліотека зі стейт-менеджменту для React, яка висвітлює більш передовий підхід і є перспективною технологією. Мінус цієї технології є в основному в тому, що по ній немає такої великої кількості документацій та прикладів робіт.

В той же час React не дозволяє робити проєкти великого масштабу без додаткового використання TypeScript та інших бібліотек, що покращують мейнтейнабіліті для розробників. Для більших проєктів доречніше буде використовувати Angular через його пряму підтримку TypeScript та готової інфраструктури, яка саме націлена на великі проєкти, до яких потрібно багато підтримки та догляду з боку розробників.

Великою перевагою Angular є його система стейт-менеджменту[19], адже вона вбудована напрямку в фреймворк Angular і до цієї системи є багато документації. В мережі інтернет можна знайти реалізацію майже будь-якого кейсу з передачі та обробки даних в рамках фронт-енд частини застосунку. Крім того TypeScript сильно полегшує роботу з великою кількістю змінних за рахунок своєї системи типів, інтерфейсів і класів. Найголовнішою проблемою Angular в контексті завдання кваліфікаційної роботи є його відносно негнучка структура, тобто спроби внесення будь-яких змін в архітектуру застосунку або зміни в потоці даних супроводжуються великою кількістю супутніх процесів, зокрема узгодження нових типів та інтерфейсів, що негативно впливатиме на потребу в швидких змінах. На противагу цьому, варто зауважити, що цей недолік дійсно потребує багато часу для внесення змін, але команда розробників завжди розуміє, що код є добре документованим, описаним та

приведений до стандартної форми.

Альтернативним підходом до React міг би стати Vue.js, який як за швидкодією, так і за організацією файлів та процесів в проєкті є дуже схожим на нього. В контексті застосування Vue.js в проєкті кваліфікаційної роботи є мінімальними, і отже використання React відображатиме більш загальну та часто зустрічний процес розробки і зробить дослідження більш об'єктивним та практикоорієнтованим. Адже Vue.js використовується порівняно в меншій кількості вебзастосунків, ніж React. Також в контексті інтеграції геоінформаційних систем вебзастосунки важливим фактором є гнучкість Frontend-фреймворку та підтримка бібліотеки, яка складається з двох основних факторів.

Фактор 1 полягає в підтримці бібліотеки, як бібліотеки виключно для JS. Мається на увазі сумісність цієї бібліотеки візуалізації гео-даних зі стандартним JavaScript, що дозволяє розробникам, не створюючи абстракцій для веб-фреймворків, маніпулювати даними на інтерактивній мапі. Другим фактором є безпосередня бібліотека для фреймворку.

Тобто, беручи до уваги Mapbox GL JS, існують окремі бібліотеки для React, Angular і Vue JS, у яких є окремі класи і методи, що дозволяють більш швидко інтеграцію в такі вебзастосунки. Але, як показала практика, ці бібліотеки мають порівняно обмежений функціонал в порівнянні з стандартними JS бібліотеками. Адже такі спеціалізовані бібліотеки є абстракціями над стандартними бібліотеками. Вони не надають такої широкої і низькорівневої кастомізації процесу візуалізації, як альтернативний підхід зі створенням власних методів зі звичайною бібліотекою.

Плюсом спеціалізованих бібліотек є швидкість інтеграції та налагоджений «з коробки» процесу передачі даних, але в той же час недалеко є мінімальні можливості персоналізації таких інтерактивних мап.

Беручи до уваги опис картографічних бібліотек, який був зроблений в

теоретико-методологічному розділі, можна зробити висновок, що для виконання задач дипломного проєкту Mapbox і Leaflet є оптимальними варіантами. Але треба враховувати, яка з цих технологій ймовірніше буде використовуватись на практиці або в індустрії веб-розробки. Звернувшись до офіційних документацій, можна зробити висновок про те, що Leaflet є виключно бібліотекою для візуалізації. Тобто Leaflet не надає жодної інфраструктури для обробки гео-даних і діє виключно як бібліотека візуалізації. Якщо ж звертатись до Mapbox, то в їх інфраструктурі є як інструменти візуалізації, зокрема Mapbox GL JS, але крім цього можна знайти інструменти з обробки гео-даних, таких як Mapbox Search JS, Directions API, Optimization API, Geocoding API та ін. Варто зазначити, що Mapbox не надає збереження гео-даних, але надає потужні обчислювальні інструменти, які злагоджено взаємодіють одне з одним.

Найкращою практикою для збереження гео-даних є використання власної бек-енд інфраструктури та використовувати бази даних як сховище. В багатьох випадках NoSQL бази даних були б більш доречні в використанні за рахунок того, що гео-дані складно атомізувати і звести до третьої нормальної форми[20] в середовищі SQL баз, хоча існують спеціалізовані рішення для гео-даних. Прикладом таких баз даних є PostGIS[21], яка є надбудовою над популярною open-source БД PostgreSQL. В цьому контексті NoSQL бази даних, такі як MongoDB[22], мають переваги, особливо через те, що вони підтримують нестінг даних, тобто їх вкладення, а також дані, які отримані з NoSQL баз даних, в абсолютній більшості випадків, зчитуються у JSON форматі.

Для окремих випадків варто звернути увагу на бази даних, які можуть зберігати та експортувати дані в XML форматі, який теж підтримується багатьма геоінформаційними системами. Але конкретно в випадку проєктної роботи, функціонал пов'язаний з XML та застосування такого підходу є нерентабельним в плані часу його реалізації.

Ініціалізація інтерактивної мапи за допомогою Mapbox, GL JS та Leaflet є

дуже схожою, синтаксично вони є майже ідентичними. Але, як можна побачити з параметрів налаштування об'єктів мапи, Leaflet більш спирається на локальні дані, в той час як Mapbox отримує інформацію про тему з мережі інтернет. Як нижній (фоновий) шар, в обидвох випадках використовується API від Open Source бібліотеки під назвою OpenStreetMap[23].

Всі гео-дані відображаються поверх шару з геоданими OpenStreetMap. Крім того, OpenStreetMap надає тайли в растровому форматі, хоча на офіційному сайті всі елементи, наприклад, будівлі, дороги і так далі, можна експортувати в XML форматі, просто натиснувши на відповідний елемент на мапі на їх вебсайті.

Провівши порівняння на макрорівні, варто зануритись саме в технічну реалізацію, з точки зору обох бібліотек. Умови порівняння мають бути максимально близькими, тому початкові умови визначаються наступним чином: фронтенд-фреймворк – React, мова програмування - JavaScript es6+, Компонент з інтеракивною мапою є окремим (ізолюваним) і підключається як дочірній елемент кореневого компоненту App[24].

Порівняння починається з написання коду з використанням картографічної бібліотеки Leaflet (Додаток Б). Компонент імпортує React та хуки з react, а також компоненти з react-leaflet. Імпортується також файл стилів Leaflet для забезпечення коректного відображення мапи. Гео-дані імпортуються з локального файлу GeoJSON. У реальному проекті ці дані можуть бути отримані з зовнішнього API або сервера. Використовується компонент MapContainer з react-leaflet для створення контейнера мапи. Властивість center задає початкову координату центру мапи, а zoom визначає початковий рівень збільшення. Компонент TileLayer додає шар тайлів OpenStreetMap до мапи, забезпечуючи базову картографічну інформацію. Компонент GeoJSON використовується для відображення гео-даних з файлу GeoJSON. Властивість data приймає імпортовані гео-дані.

У випадку Mapbox процес відбувається схожим чином, але є певні

відмінності (Додаток Б). `MapComponent` приймає функцію `onMarkerClick` через `Props`. Використовується хук `useState` для управління станом `viewport` мапи. Використовується `useRef` для отримання посилання на екземпляр мапи. Функція `handleMapClick` використовується для обробки подій кліку на мапі. Вона використовує метод `queryRenderedFeatures` для отримання міток, на які клікнули, і викликає `onMarkerClick` з `id` мітки. Компонент `MapGL` візуалізує мапу, додаючи шар з мітками з файлу `GeoJSON`.

2.2. Екстракція та попередні перетворення гео-даних.

Для пошуку та як джерело гео-даних в проєкті застосовувався сервіс OpenStreetMap. OpenStreetMap (OSM) є вільним, відкритим і колективним проєктом, що надає можливість створення та використання детальних географічних даних. Заснований у 2004 році Стівом Коастом, OSM став одним з найважливіших джерел геопросторової інформації, доступним для широкого загалу. Основною метою проєкту є забезпечення відкритих картографічних даних, які можуть бути вільно використані, модифіковані та розповсюджені будь-ким[25]. OpenStreetMap базується на принципі краудсорсингу, що передбачає активну участь волонтерів з усього світу в створенні та оновленні картографічних даних. Ці дані включають в себе різноманітні географічні об'єкти, такі як дороги, будівлі, річки, залізничні лінії, парки та інші елементи інфраструктури. Користувачі можуть додавати нові об'єкти, редагувати існуючі, а також виправляти помилки, що дозволяє постійно оновлювати та покращувати якість картографічних даних.

Однією з ключових особливостей OSM є відкритість даних. Всі дані, зібрані в рамках проєкту, доступні під ліцензією Open Database License (ODbL)[26], що дозволяє їх вільне використання, модифікацію та розповсюдження. Це забезпечує широкий спектр можливостей для їх застосування у різних галузях. Відкриті дані OSM використовуються в геоінформаційних системах для створення карт, аналізу просторових даних та моделювання. Багато програмного забезпечення для ГІС, такі як QGIS та ArcGIS, підтримують імпорт та використання даних OSM, що дозволяє користувачам створювати деталізовані карти та проводити різноманітні просторові аналізи. Крім геоінформаційних систем, дані OSM використовують для мобільних додатків, урбаністики, гуманітарної допомоги, наукових досліджень та комерційних проєктів.

Для отримання гео-даних про будівлі університету потрібно скористатись

OpenStreetMap, а саме функцією експорту (Рис. 2.1). В панелі управління по лівій стороні екрану запропоновано декілька способів експорту. Варто зазначити Overpass API, який дозволяє виконувати до 10000 запитів на добу або отримувати до 1 ГБ даних на добу. Цей API дозволяє напряму отримувати дані з OSM в форматі JSON, що в свою чергу дає розробнику можливість автоматизувати отримання гео-даних.

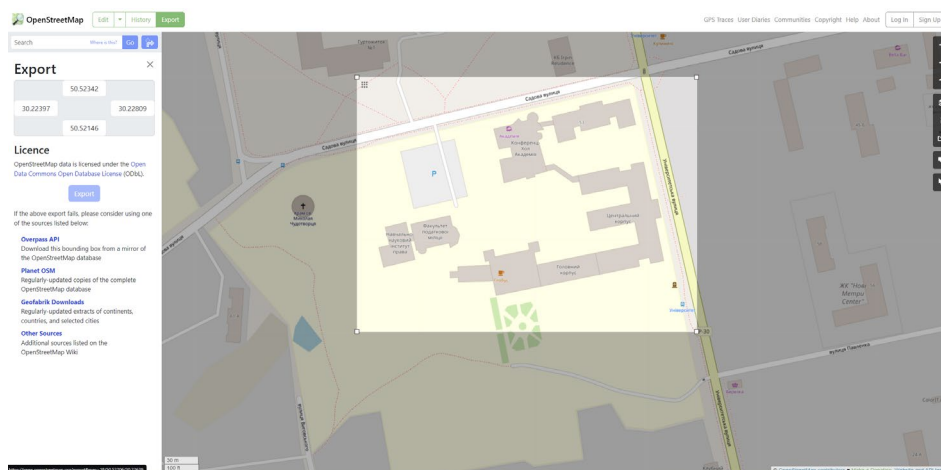


Рис. 2.1. Експорт гео-даних в обраному секторі на сайті OpenStreetMap.

У випадку з проектом дипломної роботи, потрібно лише отримати дані про завчасно визначені споруди на території Державного податкового університету. Для цього було обрано відповідну зону на мапі і натиснуто кнопку «експорт». В результаті на ПК завантажено файл формату .osm, що можна відразу використовувати в геоінформаційних системах. Враховуючи особливості програмного продукту, отримані дані потрібно спершу конвертувати в GeoJSON формат та обрати набір цільових полігонів (полігонів будівель).

Для конвертування даних було використано онлайн-сервіс Aspose[27], який надає послуги з конвертування .osm в GeoJSON. Після завантаження .osm-файлу відбувається перетворення даних та з'являється можливість завантажити файл в цільовому форматі. В результаті отриманий файл можна відкрити за допомогою блокноту (Рис. 2.2) або іншого текстового редактора і перенести

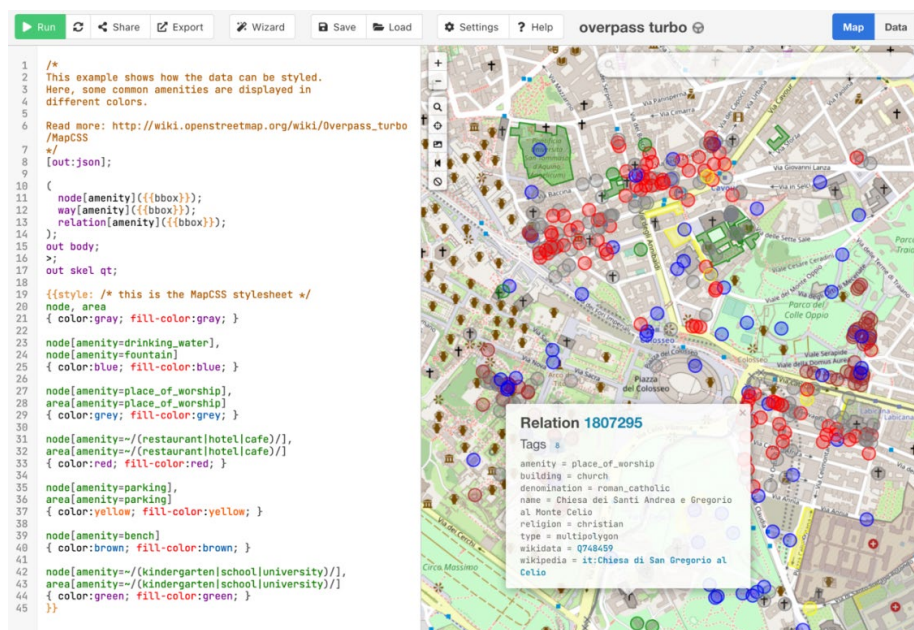


Рис. 2.3. Інтерфейс вебзастосунку Overpass Turbo.

Загалом Overpass Turbo є потужним та гнучким інструментом для роботи з даними OpenStreetMap, який надає широкі можливості для веб-розробників. Він дозволяє легко формулювати складні запити, візуалізувати результати на інтерактивній карті та експортувати дані у різні формати. Завдяки своїй функціональності та простоті використання, Overpass Turbo стає незамінним інструментом для розробників, які працюють з геопросторовими даними та інтеграцією карт у свої веб-додатки.

2.3. Розробка вебзастосунку.

Проект дипломної роботи полягає в розробці вебзастосунку, який є централізованою платформою для відображення заходів на території університету. Для цього потрібно створити інтерактивну мапу, яка відображатиме окремі будівлі, при натисканні на які інтерактивна мапа реагуватиме на ввід від користувача. Окремий компонент має відображати детальну інформацію про події в обраній будівлі. Цей компонент має оновлюватись відповідно до взаємодії користувача з мапою. Додатково потрібно створити 2 шари з гео-даними. Шар 1 має містити набір точок що позначають кожену будівлю. Такі точки, маркери, також повинні містити інформацію про кількість запланованих заходів в цьому місці. Шар 2 буде складатись з полігонів з динамічною візуалізацією. Ці полігони містять інформацію про координати кожної будівлі і візуалізуються динамічно, в залежності від обраного маркера на шарі 1.

Для реалізації вищеописаних елементів першим кроком в побудові фронт-енд вебзастосунку є планування структури компонентів React. Для базового функціонування такого програмного продукту знадобиться 4 компоненти, які зможуть виконувати вищезазначені задачі (Рис. 2.4): App, Map, InfoPanel, EventItem. Стили до цих компонентів мають міститись в окремих файлах формату .css, розташовані разом з відповідними файлами компонентів формату .jsx. Глобальні стилі містяться в стандартному файлі index.css.

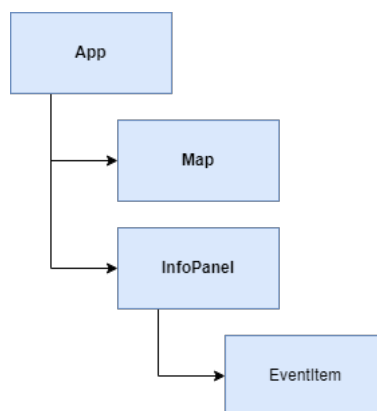


Рис. 2.4. Діаграма ієрархії компонентів програмного продукту.

Процеси обміну даними в рамках застосунку є відносно простим. Основним провайдером даних є об'єкт класу `MockDataController`, який створюється в `App` компоненті, і посилання на нього передається нижче по ієрархії компонентів у вигляді `Props`[30]. Провайдер бере всі необхідні дані з файлів формату `.json`, що згодом розробник може замінити на повноцінну інтеграцію бази даних або відповідною бек-енд інфраструктурою. `MockDataController` містить методи стандартного набору `CRUD`[31] і відповідає одразу за 2 типи даних – гео-дані та заходи. При ініціалізації об'єкту в кожен окремий елемент гео-даних додається поле «`id`» в категорії «`properties`». Це дозволяє індексувати всю гео-інформацію і робити їх прив'язку до інтерактивних елементів або до подій.

Другим важливим елементом в процесі обміну даними виступає запроваджений в застосунку архітектурний шаблон `Pub/Sub`. Якщо `MockDataController` є «пасивним» провайдером даних, тобто дані від нього передаються лише тоді, коли відповідний елемент робить запит до цього об'єкта, то `pub/sub` дозволяє створити івенти і мати так званих «підписників» на ці івенти, тобто будь-який елемент може реагувати на івент з інших елементів, при цьому ієрархія компонентів в такому випадку не грає жодної ролі. Найбільшою перевагою такого підходу є можливість створення майже необмеженої кількості івентів та додавання до кожного будь-якої кількості

підписників, з динамічним оновленням даних в випадку їх змін.

Комбінація з двох вищезазначених елементів фактично задовольняє більшість потреб програмного продукту. Альтернативно для стейт-менеджменту можна застосовувати спеціалізовані бібліотеки[32], але їх переваги стають явними тільки в процесі збільшення масштабів застосунку та наростання додаткових змінних, стан яких потрібно відслідковувати.

Дані, які використовує та модифікує програмний продукт, зберігаються в окремих файлах формату .json. Для простоти процесу тестування було запроваджено попереднє завантаження до об'єкта MockDataController даних про події та гео-дані міток будівель. Вони проходять індексацію і необхідні перетворення для подальшого відображення під час створення об'єкта. Не менш важливим є підрахунок кількості подій та занесення цієї інформації в properties геомітки. Зібравши та наповнивши масив даних під назвою data в об'єкті MockDataController, будь-який компонент в який імпортовано посилання на цей об'єкт, може використовувати методи для отримання та зміни як інформації про події, так і інформації про геомітки (Рис. 2.5).

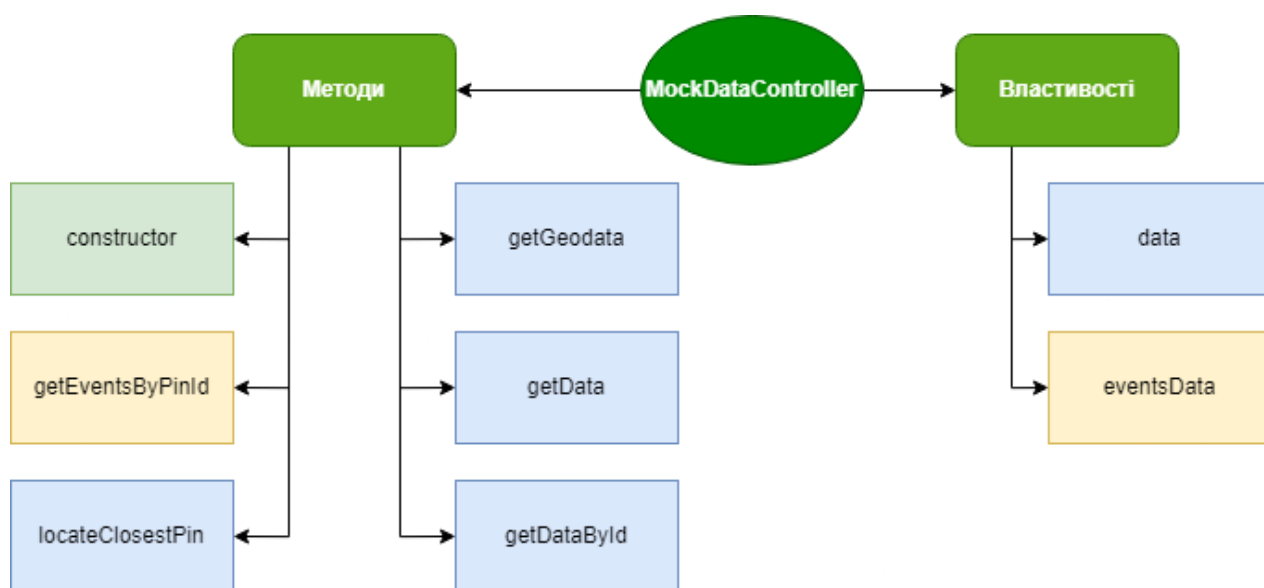


Рис. 2.5. Структура класу MockDataController.

Використання картографічної бібліотеки Mapbox починається з надання ключа доступу[33]. Це здійснюється за допомогою змінної Access Token, яка застосовується відразу після імпортування бібліотеки. Ініціалізувати об'єкт мапи в фреймворку React рекомендується після ініціалізації компоненту, тобто після івенту `ComponentDidMount`, що в сучасних версіях фреймворку здійснюється за допомогою `useEffect`[34].

Після цього створюється новий елемент класу `MapboxGL.map`, в який передається об'єкт з параметрами. Ці параметри включають наступні елементи: `Container`, `Style`, `Center`, `Zoom`, `Pitch`, `Bearing`.

Контейнер, призначений в першу чергу для ідентифікації об'єкта в розмітці React, до якого буде прикріплена мапа. В коді HTML це призначається атрибутом `ref`, до якого призначається відповідна змінна в коді JSX. Наступним пунктом є `Style`.

`Style` використовує посилання на тему Mapbox. Теми можна використовувати як завчасно заготовлені, тобто стандартні від Mapbox, або є можливість створювати власні теми з великою кількістю налаштувань та імпортувати ідентичним чином в новий об'єкт класу `Map`. Імпорт здійснюється як HTTP-посилання, яке оброблюється в об'єкті без участі в цьому процесі розробника та без можливості впливу на цей процес.

Наступні 4 параметри `Center`, `Zoom`, `Pitch` та `Bearing` відповідають за конкретну точку та параметри візуалізації на віртуальній мапі. Зокрема, властивість `Zoom` відповідає за наближення мапи та відповідно до цього має вплив на ті тайли, які будуть візуалізовуватись в залежності від цього параметру. Властивість `Pitch` відповідає за кут нахилу мапи відповідно до поверхні.

Таким чином, якщо існують 3D елементи, які візуалізуються за допомогою Mapbox, існує можливість підкреслити цей 3D-ефект за рахунок використання цього параметру. Альтернативними використаннями цього

параметру є здійснення анімацій або більш якісне відображення певної площини відповідно до вимог та розмірів контейнера інтерактивної мапи. Властивість `Bearing` вказує кут від півночі.

Тобто при значенні 0 карта відображається у форматі північ вгорі. В той час, якщо значення `Bearing` буде 180, то північ для користувача буде знизу екрану. Після створення об'єкту цього класу та присвоєння цьому об'єкту змінно, можна застосовувати нові методи.

Використовуючи метод `Map.on('move', ...)` State компонента оновлюється з новими значеннями довготи, широти та зуму. Оновлення цих даних важливо в першу чергу для можливості подальшого їх застосування в супутніх функціях що викликаються як реакція на ввід від користувача та для усунення багів. У інтерактивній мапі було відключено можливість переміщуватись та наближати/віддаляти мапу. Це зроблено тому що фокус користувача під час використання вебзастосунку має бути консолідованим на університеті, і для запобігання небажаних переміщень при натисканні на маркери. За переключення таких параметрів відповідають методи `.disable()` та `.enable()`. Їх перевагою є перш за все те, що активувати їх можна динамічно, з будь-якої точки коду, де є доступ до об'єкту `map`. Це вказує що застосовувати такі налаштування можна в залежності від режиму роботи карти. Потенційно для кожної ситуації вводу користувача, може бути окремий профіль налаштувань, що в свою чергу значно покращує досвід користувача і надає розробнику зручний API для швидкої модифікації подібних налаштувань.

Важливим етапом для оптимізації продуктивності програмного продукту є запобігання витокам пам'яті. Через це при виконанні хука `React ComponentDidUnmount` застосовується метод `Map.remove()`, який завершує процеси візуалізації мапи та очищує раніше зайняту нею оперативну пам'ять.

Серед найбільш використовуваних методів класу `Map` є такі методи як `On`, `LoadImage`, `AddImage`, `AddSource`, `AddLayer` та інші. Розглядаючи кожен з методів окремо, описати метод `On` можна як універсальний обробник подій, які

відбуваються на мапі. До таких подій відносяться Load, Click, MouseMove та інші.

Зокрема, випадком використання параметру Load може слугувати додавання на мапу шарів або завантаження та підключення додаткових HTTP запитів та наповнення мапи даними. Адже всі ці операції мають проводитись після ініціалізації об'єкту. Зокрема, існує синхронний метод loadImage.

Його задача завантажити растрову картинку, яка в подальшому може використовуватись на мапі як персоналізований маркер. Наприклад, в проєкті кваліфікаційної роботи було створено локальний backend сервер, який надає frontend частині застосунку посилання на png-файл унікальної геометрії. Недоліком цього методу є те, що завантаження картинок може здійснюватись виключно через HTTP запит.

Тобто, зі сторони фронтенду неможливо додавати власні медіафайли, які могли б використовуватись в інтерактивній мапі. Ще є недоліком, особливо якщо мапа використовується в офлайн режимі, що негативно впливає на досвід користувача. Разом з методом loadImage часто використовується метод addImage.

Фактично, addImage присвоює отриманному медіафайлу певний текстовий ідентифікатор, який пізніше можуть отримувати і використовувати інші методи об'єкту map для застосування у відповідних шарах[35]. Метод addSource є одним з найважливіших методів, адже він відповідає за додавання до мапи гео-даних, які в подальшому використовують шари, посиляючись саме на значення, задані цим методом. Важливо зауважити, що з використанням addSource потрібно чітко і явно визначити параметри та типи даних для того, щоб гарантувати коректну інтерпретацію даних бібліотекою.

Важливими полями є назва та в об'єкті налаштувань type та datatype. В застосунку кваліфікаційного проєкту використовувалась нотація GeoJSON, і відповідно до цього типи вказувалися відповідно до нотації. Прикладом

використаних типів є `FeatureCollection`, в який можуть включатись всі базові типи GeoJSON, наприклад, `Points`, `Lines` та `Polygons`.

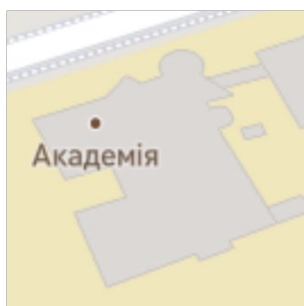
За допомогою `addSource` було додано два джерела гео-даних, один під назвою `Points`, який відповідав за візуалізацію маркерів, і другий під назвою `BuildingPolygons`, що відповідає за полігони окремих будівель, які візуалізуються в залежності від значення фільтру. Переходячи до методу `addLayer`, `addLayer` відповідає конкретно за візуалізацію або конкретніше відображення шарів на інтерактивній мапі. При створенні шару цим методом, зокрема, найважливішим параметром є поле `ID`, в якому розробник має ввести унікальний текстовий ідентифікатор шару.

Також існує поле `Type`, в якому можна вказати такі значення як `Symbol`, `Fill`, `Line` і так далі. В об'єктах `Layout` та `Paint` вказуються унікальні параметри, які відповідають за унікальні процеси візуалізації та залежать від типу шару.

В окремому `.json` файлі зберігаються додаткові дані з полігонами окремих будівель університету. Згідно стандартам GeoJSON, ці дані заберігаються у спеціалізованому форматі, який бібліотека `Mapbox GL JS` може інтерпретувати для відображення без додаткових перетворень. Навідміну від попередньо розглянутих типів даних, немає необхідності в їх зміні, а отже можна напрямку імпортувати їх в компонент `Map` і використовувати для візуалізації в залежності від обраного на мапі маркера. В `properties` кожного полігона мною було завчасно додано унікальні `id` для зв'язку між гео-даними маркерів та полігонів.

Візуалізація полігонів будівель складається з 2 шарів – «`building-fills-hover`» та «`building-borders-hover`» (Рис. 2.6). Обидва шари описують один і той самий набір гео-даних, тобто полігони будівель. «`building-fills-hover`» відповідає за заповнення полігону кольором. Конкретно в цьому шарі використано колір «`627BC1`» зі значенням прозорості 0,5. Для вказування цих властивостей згідно прикладу «`data-overlay-hover`»[36] в об'єкті властивостей шару передається об'єкт «`paint`» з унікальними властивостями «`fill-color`» та «`fill-opacity`» відповідно. В свою чергу «`building-borders-hover`» відповідає за візуалізацію

кордонів полігонів. Значення налаштувань візуалізації передається, як у попередньому випадку, властивістю «paint», але для лінії існують унікальні властивості, такі як «line-color» та «line-width», що містять значення «627BC1» та «2» відповідно. Потрібно зауважити, що окрім вищезазначених унікальних візуальних властивостей, розробнику необхідно явно зазначити тип шару, прикладом цього є значення «fill» та «line».



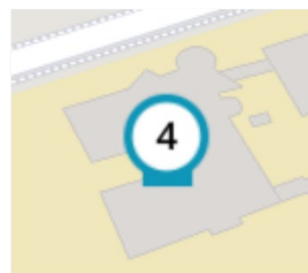
а) Растровий OSM



б) Заповнені полігони



в) Кордони полігонів



г) Маркери

Рис. 2.6. Візуалізація різних шарів окремо одне від одного.

Для цього зв'язку було використано проста передача параметру фільтру шару в рамках обробника івенту «click» (Рис. 2.7). Таким чином фільтр проходили тільки ті елементи, id якого дорівнює знайденому, тобто одночасно може бути візуалізовано лише одну будівлю.

```

52
51     map.on('click', 'points', (e) => {
50         const foundId = dataController.locateClosestPin(e.lngLat);
49         publish('pinIdUpdate', foundId);
48         map.setFilter('building-fills-hover', ['==', 'pinId', foundId]);
47         map.setFilter('building-borders-hover', ['==', 'pinId', foundId]);
46
45     });
44

```

Рис. 2.7. Демонстрація коду обробника івенту «click».

Основним інструментом взаємодії користувача з інтерактивною мапою є набір маркерів, які відповідають кожній окремій будівлі на інтерактивній мапі. Розглядаючи стандартні маркери бібліотеки Mapbox, можна зробити висновок, що вони візуально не відповідають задачам, які вони повинні виконувати в рамках проектної роботи. Перш за все, маркер повинен відображати кількість подій, які плануються найближчим часом.

Стандартний маркер Mapbox дозволяє відобразити текст фактично тільки поряд з маркером (Рис. 2.8.а), не маючи при цьому відведеного на самому маркері місця для розміщення тексту. Згідно завдання, для відображення на інтерактивній мапі достатньо місця для 2-3 символів (цифер) для відображення кількості запланованих до проведення в цій будівлі подій.

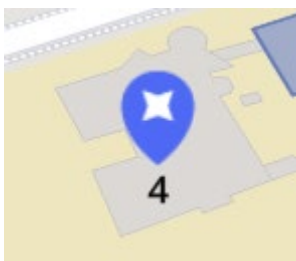
Для цього можна зробити маркер з центральним монохромним елементом у формі кола, в центрі якого надалі розмістити напис з кількістю подій. Для створення маркеру було використано програму Photoshop. В результаті було створено фігуру в ресторовому форматі .png, яку пізніше було зменшено до розмірів 30x30 пікселів та використано в вебзастосунку (Рис. 2.8.б).

Як було зазначено вище, метод Map.loadImage(), який необхідний для додавання маркера, працює виключно з гіперпосиланнями і не дозволяє отримати дані картинки з локального середовища. Саме тому було створено примітивний HTTP сервер за допомогою Node.js та Express.js. Єдина функція, яку виконує цей сервер - надання даних медіафайлу на кореновому URL-посиланні. Тобто, при запиті на посилання localhost:4000 можна отримати

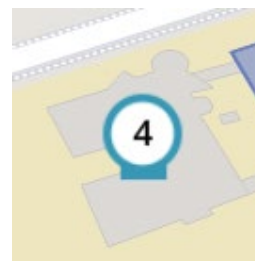
картинку маркера в будь-якому середовищі, яке пристосоване до обробки HTTP запитів.

Перевірити працездатність HTTP сервера можна за допомогою програмного застосунку Postman, браузера, або інтегрувати це посилання напряму в front-end вебзастосунок та перевірити візуальну складову.

Наступним етапом медіафайл у випадку успішного завантаження додається до мапи за допомогою метода `Map.addImage()` і в контексті мапи йому привласнюється ім'я `customMarker`. Після виконаних процедур маркер може використовуватися для відображення шарів типу «symbol» на інтерактивній мапі.



а) Стандартний маркер



б) Видозмінений маркер

Рис. 2.8. Візуалізація маркерів за допомогою Mapbox GL JS.

До додаткових налаштувань інтерактивної мапи відноситься, зокрема, зміна курсору миші при наведенні на маркери. Це зроблено перш за все для того, щоб показати користувачеві, що на маркер можна натискати і покращує досвід користувача. При натисканні на маркер застосовується новий фільтр до шарів, які відповідають за візуалізацію контурів будівель, що теж несе роль семантичної реакції на дії користувача.

В момент натискання в інформаційній панелі відображаються заходи, які заплановані до проведення в обраній будівлі. Заходи відображаються в форматі списку, і цей список складається з одиничних елементів з описами кожного

окремого заходу. З технічної точки зору це реалізовано за допомогою JSX кода, який з використанням циклу візуалізує кожен елемент списку з даними масива.

Кожна подія в списку містить наступні поля — заголовок, дата проведення, час початку та кінця проведення заходу, місце проведення та опис. В залежності від обраної будівлі заголовок списку також змінюються (Рис. 2.9).

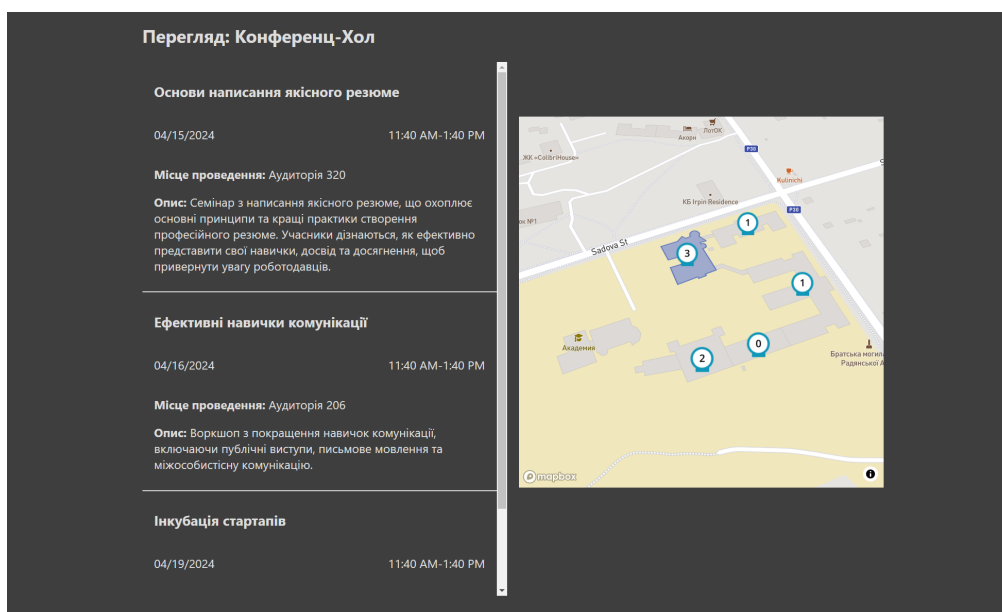


Рис. 2.9. Користувацький інтерфейс розробленого вебзастосунку.

Таким чином, зокрема, можна додати функціонал, який підтримує програми Screen Reader, які допомагають людям з вадами зору орієнтуватись в вебзастосунку.

РОЗДІЛ 3. РЕТРОСПЕКТИВНИЙ АНАЛІЗ ТА ПРОПОЗИЦІЇ З ПОКРАЩЕННЯ ПРОГРАМНОГО ПРОДУКТУ.

Використання різних картографічних бібліотек, таких як Mapbox і Leaflet, має свої особливості, переваги та виклики, що потребують врахування при розробці та вдосконаленні картографічних рішень. Цей розділ присвячений аналізу поточних недоліків та пропозиціям щодо їх усунення.

Низькорівнева персоналізація, яка включає детальне налаштування окремих елементів інтерактивної мапи, є більш складною при використанні Mapbox у порівнянні з Leaflet. Leaflet надає розробникам більш універсальні класи та методи, що дозволяють легко маніпулювати індивідуальними елементами карти. Наприклад, у Leaflet можна з легкістю створювати та кастомізувати маркери, полігони та інші об'єкти, використовуючи вбудовані методи та події.

Mapbox, з іншого боку, часто потребує створення додаткового коду для реалізації персоналізації. Хоча Mapbox GL JS надає потужні інструменти для роботи з картами, такі як стильові правила та обробка подій, розробникам може знадобитися більше часу та зусиль для досягнення бажаного рівня кастомізації. Таким чином, для проектів, де важливо забезпечити високу ступінь персоналізації карт, може бути доцільніше використовувати Leaflet.

Варто звернути увагу на принципово різні підходи до візуалізації мапи та взаємодії розробника з її елементами. Mapbox GL JS виконує всю візуалізацію за допомогою HTML-елементу Canvas, на чому і базуються всі подальші дії. Leaflet, з іншого боку, використовує елемент-орієнтований підхід[37].

Створення інтерактивної мапи за допомогою Mapbox GL JS вимагає готовності до розробки власної backend інфраструктури. Це обумовлено тим, що Mapbox надає широкий спектр інструментів та сервісів для обробки та візуалізації гео-даних, які можуть вимагати інтеграції з іншими системами.

Наприклад, для зберігання великих обсягів даних або забезпечення високої продуктивності системи може знадобитися налаштування власного сервера з базою даних.

Це пасує для великих проектів з високими вимогами до масштабованості та продуктивності, але може стати викликом для невеликих проектів або команд з обмеженими ресурсами. В таких випадках, використання більш легких бібліотек, таких як Leaflet, може бути більш доцільним, оскільки вони дозволяють зосередитися на розробці клієнтської частини без необхідності створення складної backend інфраструктури.

Використання шарів для відображення колекцій гео-даних є ефективним з точки зору продуктивності, але накладає певні обмеження на розробника. Використання шарів дозволяє значно підвищити продуктивність за рахунок зменшення кількості рендерингів окремих об'єктів, особливо при роботі з великими наборами даних. Однак, цей підхід обмежує можливості індивідуального редагування об'єктів та надання їм унікальних рис.

Наприклад, при відображенні великої кількості маркерів або полігонів за допомогою Mapbox GL JS, використання шарів дозволяє значно зменшити навантаження на систему. Але якщо виникає потреба у наданні окремим об'єктам унікальних властивостей, таких як різний колір або поведінка при взаємодії з користувачем, це може вимагати додаткових зусиль та складних рішень. Для покращення цього аспекту, розробникам слід розглянути можливість комбінування підходів, використовуючи шари для загального відображення даних та індивідуальні елементи для специфічних випадків. Це дозволить зберегти високу продуктивність системи, не обмежуючи можливості кастомізації.

Сучасною тенденцією у світі геоінформаційних систем є розвиток 3D візуалізації та створення 3D оточень. Використання 3D моделей для відображення будівель, ландшафтів та інших об'єктів на інтерактивних картах дозволяє надавати користувачам більш детальну та реалістичну інформацію.

Інструменти, такі як OpenDroneMap[38], вже сьогодні дозволяють створювати 3D моделі з аерофотознімків, що відкриває нові можливості для візуалізації та аналізу гео-даних.

Ще одним важливим аспектом, що потребує уваги, є оптимізація використання трафіку шляхом застосування технологій кешування. Використання Redis[39] або інших кешуючих механізмів дозволяє зберігати часто використовувані дані у тимчасовому сховищі, що знижує навантаження на сервер та прискорює доступ до інформації. Це особливо актуально для вебзастосунків з великим обсягом користувацьких запитів або при роботі з великими наборами гео-даних.

Проект кваліфікаційної роботи може значно виграти від застосування TypeScript як основної мови програмування. TypeScript є надмножиною JavaScript, яка додає статичну типізацію до мови[40], що забезпечує ряд переваг у розробці та підтримці програмних продуктів. У цьому розділі розглянемо, як саме використання TypeScript може позитивно вплинути на реалізацію та якість проекту кваліфікаційної роботи.

Статична типізація дозволяє виявляти помилки типів ще до запуску коду, що значно знижує ризик виникнення помилок під час виконання програми. Це особливо важливо у складних проектах, де некоректне передавання типів може призвести до серйозних багів і некоректної роботи застосунку. Явне визначення типів робить код більш зрозумілим та само-документованим. Розробники можуть швидко зрозуміти, які типи даних очікуються та передаються між функціями та компонентами. Це особливо корисно при роботі у великих командах або при підтримці проекту іншими розробниками.

TypeScript забезпечує додатковий рівень безпеки, запобігаючи поширеним помилкам, таким як неявні перетворення типів або використання невизначених змінних. Це підвищує надійність коду та забезпечує його коректну роботу в різних сценаріях використання.

Не менш важливою перевагою застосування TS є розширені можливості автодоповнення та інтелектуального аналізу коду в сучасних редакторах коду та IDE, таких як Visual Studio Code. Це прискорює процес розробки, знижує кількість помилок та підвищує продуктивність розробників.

З іншого боку, покращення вебзастосунку можна зробити шляхом заміни паттерну Pub/Sub, що відповідає за стейт-менеджмент, на більш розповсюджені та універсальні альтернативи, такі як Redux або Zustand. Redux забезпечує централізоване управління станом, передбачуваність та підтримку потужних інструментів для розробки, що є важливим для великих проектів. Zustand, зі свого боку, пропонує простоту, гнучкість та мінімальну кількість коду, що робить його ідеальним вибором для швидкого впровадження та підтримки. Вибір між цими бібліотеками залежить від конкретних вимог проекту та його масштабу, однак обидві бібліотеки можуть значно покращити реалізацію паттерну Pub/Sub та забезпечити ефективне управління станом у вебзастосунках.

Redux надає єдине сховище для всіх даних додатку, що забезпечує централізоване управління станом. Це спрощує відстеження змін та полегшує діагностику проблем, оскільки весь стан додатку зберігається в одному місці. Використання редукторів (reducers) у Redux дозволяє чітко визначити, як змінюється стан у відповідь на дії (actions). Це робить поведінку додатку передбачуваною та спрощує тестування, оскільки кожна дія має визначений результат.

Крім того, Redux має потужні інструменти для розробки, такі як Redux DevTools, що дозволяють відстежувати зміни стану у реальному часі, відкатувати стан та аналізувати дії. Це значно підвищує ефективність розробки та дебагінгу.

До недоліків Redux в свою чергу відносять те, що він може призвести до написання великої кількості коду для визначення дій, редукторів та типів. Це може збільшити складність коду та потребувати додаткових зусиль для підтримки. Також особливості бібліотеки можуть збільшити час на навчання та

впровадження Redux у проект.

В свою чергу Zustand є більш легкою та сучасною бібліотекою для управління станом у React-застосунках. Вона використовує менш вербозний підхід до управління станом та забезпечує простоту у використанні. У порівнянні з Redux, Zustand вимагає значно менше коду для реалізації управління станом. Це знижує складність коду та полегшує підтримку проекту. Вона легко інтегрується з існуючими React-застосунками без необхідності значних змін у коді. Це дозволяє швидко впровадити Zustand у поточний проект та отримати переваги від його використання.

На відміну від Redux, Zustand не має таких потужних інструментів для розробки та дебагінгу. Це може ускладнити відстеження змін стану у великих проектах. Також вона має меншу спільноту розробників у порівнянні з Redux, що може обмежити доступ до ресурсів та підтримки. Хоча використання патерну Pub/Sub є традиційним підходом, бібліотеки Redux та Zustand можуть значно покращити цей процес.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою було продемонстровано важливість інтеграції геоінформаційних систем у сучасні вебзастосунки та показано можливості покращення користувацького досвіду за допомогою інтерактивних картографічних рішень. Використання сучасних технологій, таких як TypeScript, React, Mapbox GL JS, а також впровадження ефективних методів управління станом та оптимізації продуктивності, що дозволяє створювати високоякісні та надійні вебзастосунки.

Отримані результати та розроблений вебзастосунок мають практичну цінність і можуть бути використані для створення інтерактивних картографічних рішень у різних сферах, таких як освіта, логістика, управління нерухомістю та інші. Подальший розвиток та вдосконалення інтеграції геоінформаційних систем у вебзастосунки сприятиме підвищенню ефективності та зручності роботи з гео-даними, забезпечуючи користувачам нові можливості для аналізу та візуалізації просторової інформації.

У процесі виконання кваліфікаційної роботи було здійснено глибоке ознайомлення з офіційною документацією та науковими публікаціями, що стосуються геоінформаційних систем та їх компонентів. Розглянуто картографічні бібліотеки та платформи, такі як Mapbox, Google Maps, Leaflet, ArcGIS, QGIS, і їх функціональні можливості. Було визначено переваги та недоліки кожної з них у контексті використання для інтерактивної візуалізації гео-даних.

В рамках практичної частини роботи був спроектований та реалізований вебзастосунок, який демонструє можливості інтеграції геоінформаційних систем у користувацький інтерфейс. Проект включав створення інтерактивної мапи, яка дозволяє користувачам взаємодіяти з гео-даними, переглядати інформацію про події у вибраних будівлях та здійснювати навігацію по території університету.

Для реалізації проекту були застосовані сучасні технології, зокрема, бібліотеки React та Mapbox GL JS. Використання React дозволило створити модульний та легко масштабований застосунок, а Mapbox GL JS забезпечила високу продуктивність та можливості кастомізації інтерактивної мапи. Було також розглянуто альтернативні бібліотеки, такі як Leaflet та Google Maps Platform, і зроблено висновок про їхні можливості та обмеження.

На основі проведених досліджень та практичних експериментів були розроблені пропозиції щодо покращення інтеграції геоінформаційних систем у вебзастосунки. Рекомендовано зокрема використовувати TypeScript для підвищення надійності та безпеки коду, зниження кількості помилок та полегшення підтримки проекту; розглядати можливість використання сучасних бібліотек для управління станом, таких як Redux та Zustand, для покращення обміну даними між компонентами та забезпечення передбачуваності поведінки застосунку; застосовувати методи кешування для зниження навантаження на сервер та прискорення доступу до даних; мати чітке розуміння сфер використання різних геоінформаційних систем, особливості візуалізації та парадигми їх модифікації.

Результати дослідження було апробовано на Всеукраїнській конференції молодих вчених «Математичні методи та цифрові технології в економіці». На ній розглядались та порівнювались між собою картографічні бібліотеки, адаптовані для роботи з веб-фреймворками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петлюк, І., Рижов, Є., Гелета, С., & Істомін, К. ІНФОРМАЦІЙНІ ТА ГЕОГРАФІЧНІ ІНФОРМАЦІЙНІ СИСТЕМИ. Матеріали Міжнародної науково-технічної конференції «Геофорум-2022», 6–8 квітня 2022 р., Львів–Яворів–Брюховичі.–Львів: Видавництво Львівської політехніки, 2022.–Режим доступу: http://zgt.com.ua/wp-content/uploads/2022/06/ТЕЗИ_ГЕОФОРУМ_2022.pdf вільний.–, 48.
2. Козенкова, В. Д. ГЕОГРАФІЧНІ ІНФОРМАЦІЙНІ СИСТЕМИ ЯК МЕХАНІЗМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В АГРАРНОМУ СЕКТОРІ.
3. Mapbox. Офіційна документація. URL: <https://docs.mapbox.com/> (дата звернення: 28.11.2023).
4. Google Maps Platform. Офіційна документація. URL: <https://developers.google.com/maps/documentation/> (дата звернення: 28.11.2023).
5. Leaflet. Офіційна документація. URL: <https://leafletjs.com/reference.html> (дата звернення: 02.12.2023).
6. ArcGIS. Офіційна документація. URL: <https://www.esri.com/en-us/arcgis/products/arcgis-pro/resources> (дата звернення: 03.12.2023).
7. QGIS. Офіційна документація. URL: <https://qgis.org/en/docs/index.html> (дата звернення: 03.12.2023).
8. С. Dempsey. Types of GIS Data Explored: Vector and Raster. Стаття. 2024. URL: <https://www.geographyrealm.com/geodatabases-explored-vector-and-raster-data/> (дата звернення: 06.12.2023).

9. The GeoJSON Format. Стандарт. URL: <https://datatracker.ietf.org/doc/html/rfc7946> (дата звернення: 07.12.2023).
10. KML. Стандарт. URL: <https://www.ogc.org/standard/kml/> (дата звернення: 07.12.2023).
11. OpenLayers. Офіційна документація. URL: <https://openlayers.org/en/latest/apidoc/> (дата звернення: 08.12.2023).
12. Vis, M. J. P. (2018). History of the Mercator projection.
13. Gehlenborg, N., & Wong, B. (2012). Heat maps. Nature Methods, 9(3), 213.
14. Proj. Офіційний сайт. URL: <https://proj.org/en/9.4/about.html> (дата звернення: 17.12.2023).
15. State of JavaScript 2022: Front-end Frameworks. URL: <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/> (дата звернення: 18.12.2023).
16. A. Kumar. Design patterns in Javascript: Publish-Subscribe or PubSub. Стаття. 2021. URL: <https://dev.to/anishkumar/design-patterns-in-javascript-publish-subscribe-or-pubsub-20gf> (дата звернення: 18.12.2023).
17. React Redux. Офіційна документація. URL: <https://react-redux.js.org/> (дата звернення: 25.12.2023).
18. Zustand. Introduction. Офіційна документація. URL: <https://docs.pmnd.rs/zustand/getting-started/introduction> (дата звернення: 25.12.2023).
19. What is NgRx? Офіційна документація. URL: <https://ngrx.io/docs> (дата звернення: 16.01.2024).
20. L. Heeseok. Justifying database normalization: a cost/benefit model. Information processing & management. 1995. С. 59-67.
21. PostGIS. Офіційна документація. URL: https://postgis.net/documentation/getting_started/ (дата звернення: 22.01.2024).

22. MongoDB. Query on Embedded/Nested Documents. Офіційна документація.
URL: <https://www.mongodb.com/docs/manual/tutorial/query-embedded-documents/> (дата звернення: 23.01.2024).
23. Leaflet API Reference. Map usage example. Офіційна документація. URL:
<https://leafletjs.com/reference.html#map-example> (дата звернення: 25.01.2024).
24. A Comprehensive Guide to Structuring Your React App Project. Стаття. 2024.
URL: https://dev.to/dhanasai_tholeti/a-comprehensive-guide-to-structuring-your-react-app-project-2a9d (дата звернення: 26.01.2024).
25. OpenStreetMap. Офіційний сайт. URL:
<https://www.openstreetmap.org/#map=18/50.52236/30.22580> (дата звернення: 28.01.2024).
26. Open Data Commons Open Database License (ODbL). Електронний ресурс.
URL: <https://opendatacommons.org/licenses/odbl/> (дата звернення: 28.01.2024).
27. Aspose GIS Instruments. Conversion to GeoJSON. Електронний ресурс.
URL: <https://products.aspose.app/gis/conversion/convert-to-geojson> (дата звернення: 30.01.2024).
28. Overpass Turbo. Електронний ресурс. URL: <https://overpass-turbo.eu/> (дата звернення: 01.02.2024).
29. Overpass QL. Офіційна документація. URL:
https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL (дата звернення: 01.02.2024).
30. React. Passing Props to a Component. Офіційна документація. URL:
<https://react.dev/learn/passing-props-to-a-component> (дата звернення: 03.02.2024).
31. Truica, C. O., Boicea, A., & Trifan, I. (2013, August). CRUD operations in MongoDB. In 2013 international conference on advanced computer science and electronics information (ICACSEI 2013) (pp. 347-350). Atlantis Press.
32. Tran, K. (2023). State Management in React. С. 47-84.

- 33.Mapbox. Glossary: access token. Офіційна документація. URL: <https://docs.mapbox.com/help/glossary/access-token/> (дата звернення: 06.02.2024).
- 34.React. useEffect. Офіційна документація. URL: <https://react.dev/reference/react/useEffect> (дата звернення: 06.02.2024).
- 35.Mapbox. API Reference: addImage. Офіційна документація. URL: <https://docs.mapbox.com/mapbox-gl-js/api/map/#map#addimage> (дата звернення: 06.02.2024).
- 36.GitHub. Mapbox react examples: data-overlay-hover. URL: <https://github.com/mapbox/mapbox-react-examples/tree/master/data-overlay-hover> (дата звернення: 07.02.2024).
- 37.Leaflet. Extending Leaflet. Офіційна документація. URL: <https://leafletjs.com/examples/extending/extending-1-classes.html> (дата звернення: 11.02.2024).
- 38.OpenDroneMap. Електронний ресурс. URL: <https://opendronemap.org/> (дата звернення: 12.02.2024).
- 39.Redis. Client-side caching in Redis. Офіційна документація. URL: <https://redis.io/docs/latest/develop/use/client-side-caching/> (дата звернення: 18.03.2024).
- 40.TypeScript. Everyday Types. Офіційна документація. URL: <https://www.typescriptlang.org/docs/handbook/2/everyday-types.html#object-types> (дата звернення: 18.03.2024).

Додаток А.

Порівняльна таблиця геоінформаційних систем

Функція	Геоінформаційні системи	Картографічні платформи	Картографічні бібліотеки
Охоплення	Повний набір інструментів для управління просторовими даними, аналізу та візуалізації	Комплексний набір інструментів та сервісів для веб- та мобільних додатків	Легкі, модульні рішення для вбудовування карт у веб-сторінки або додатки
Зберігання даних	Інтегровані можливості зберігання даних, включаючи просторові бази даних (наприклад, PostGIS)	Хмарне зберігання даних (наприклад, Mapbox, Google Maps Platform)	Зазвичай не включає зберігання даних; покладається на зовнішні джерела даних
Дані в реальному часі	Розширені можливості інтеграції та аналізу даних в реальному часі	Підтримка інтеграції даних в реальному часі (наприклад, оновлення трафіку, живе відстеження)	Обмежені можливості для даних в реальному часі
Геокодування та зворотне геокодування	Інтегровані інструменти геокодування та зворотного геокодування	Вбудовані сервіси геокодування та зворотного геокодування (наприклад, Google Maps API)	Вимагає сторонніх сервісів для геокодування
Маршрутизація та навігація	Розширені інструменти аналізу мереж та маршрутизації	Забезпечує покрокову навігацію та оптимізацію маршрутів	Базові можливості маршрутизації через плагіни
Просторовий аналіз	Повний просторовий аналіз, включаючи растровий і векторний аналіз, просторову статистику	Доступні базові інструменти просторового аналізу	Базовий просторовий аналіз через плагіни та розширення
Управління	Розширене	Базові можливості	Обмежене або

Функція	Геоінформаційні системи	Картографічні платформи	Картографічні бібліотеки
даними	управління даними, включаючи версіонування, редагування та обробку метаданих	управління даними	відсутнє управління даними; покладається на зовнішні інструменти
Налаштування	Розширені можливості налаштування через скрипти (наприклад, Python, ModelBuilder)	Високий рівень налаштування через розширені API (наприклад, Mapbox Studio)	Налаштовується через плагіни та модулі
Візуалізація та стилізація	Розширені картографічні інструменти для створення та експорту високоякісних карт	Високоякісні, налаштовувані стилі карт та теми	Базові можливості візуалізації; розширюється за допомогою плагінів
Управління користувачами	Розширене управління доступом на основі ролей, інструменти управління користувачами та співпраці	Обмежені можливості управління користувачами; часто прив'язані до системи облікових записів платформи	Зазвичай відсутнє управління користувачами
Інтеграція із зовнішніми даними	Пряма інтеграція з широким спектром зовнішніх джерел даних та стандартів	Підтримка інтеграції з різними зовнішніми джерелами даних та API	Покладається на плагіни для інтеграції зовнішніх даних
Карти, готові до друку	Розширені інструменти для створення та експорту карт, готових до друку	Базові можливості експорту карт для друку	Обмежена або відсутня підтримка карт, готових до друку
3D візуалізація	Повні інструменти для 3D аналізу та візуалізації просторових даних	Базові до розширених можливості 3D візуалізації (наприклад, Mapbox	Обмежена 3D візуалізація

Функція	Геоінформаційні системи	Картографічні платформи	Картографічні бібліотеки
		GL JS)	
Скрипти та автоматизація	Розширені можливості скриптів та автоматизації (наприклад, Python, R, ModelBuilder)	Базові можливості скриптів через API	Обмежені скрипти; покладається на JavaScript
Функції співпраці	Розширені інструменти співпраці для багатокористувацького редагування та обміну даними	Обмежені функції співпраці	Зазвичай відсутні функції співпраці
Розглянуті платформи	ArcGIS, QGIS	Mapbox, Google Maps Platform	Leaflet, Mapbox GL JS

Пояснення ознак порівняння:

- Охоплення: Загальний діапазон та комплексність наданих інструментів та сервісів.
- Зберігання даних: Здатність зберігати та керувати геопросторовими даними.
- Дані в реальному часі: Інтеграція та обробка живих потоків даних.
- Геокодування та зворотне геокодування: Перетворення адрес у координати та навпаки.
- Маршрутизація та навігація: Можливості надання навігації та планування маршрутів.
- Просторовий аналіз: Інструменти та можливості для виконання просторового аналізу даних.
- Управління даними: Обробка, редагування та підтримка геопросторових даних.

- Налаштування: Гнучкість та інструменти для налаштування карт та функціональності.
- Візуалізація та стилізація: Якість та опції для візуалізації та стилізації карт.
- Управління користувачами: Функції для управління ролями користувачів та дозволами.
- Інтеграція із зовнішніми даними: Здатність інтегрувати та використовувати дані із зовнішніх джерел.
- Карти, готові до друку: Можливості створення карт, придатних для друку.
- 3D візуалізація: Інструменти для 3D представлення та аналізу просторових даних.
- Скрипти та автоматизація: Підтримка автоматизації завдань та розширення функціональності через скрипти.
- Функції співпраці: Інструменти та функції, що підтримують багатокористувацьку співпрацю.

Додаток Б.

Лістинг 1. Код React-комоненту для візуалізації мапи Leaflet.

```
import React from 'react';
import { MapContainer, TileLayer, GeoJSON } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';
import geoData from './data/yourGeoData.geojson'; // Імпорт файлу GeoJSON

const MapComponent = ({ onMarkerClick }) => {

  const onEachFeature = (feature, layer) => {
    layer.on({
      click: () => {
        onMarkerClick(feature.properties.id);
      },
      mouseover: (event) => {
        event.target.setStyle({
          color: 'blue',
          weight: 2,
        });
      },
      mouseout: (event) => {
        event.target.setStyle({
          color: 'black',
```

```

        weight: 1,

    });

},

});

};

return (

    <MapContainer center={[51.505, -0.09]} zoom={13} style={{ height: "100vh",
width: "100%" }}>

        <TileLayer

            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

            attribution='&copy;                                <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

            />

            <GeoJSON data={geoData} onEachFeature={onEachFeature} />

        </MapContainer>

    );

};

export default MapComponent;

```

Лістинг 2. Код React-компоненту для візуалізації мапи Mapbox GL JS.

```
import React, { useEffect, useState, useRef } from 'react';
import MapGL, { Source, Layer } from 'react-map-gl';
import 'mapbox-gl/dist/mapbox-gl.css';
import geoData from './data/yourGeoData.geojson'; // Імпорт файлу GeoJSON

const MAPBOX_TOKEN = 'your_mapbox_token_here'; // Замініть на свій Mapbox
Access Token

const MapComponent = ({ onMarkerClick }) => {
  const [viewport, setViewport] = useState({
    latitude: 51.505,
    longitude: -0.09,
    zoom: 13,
    bearing: 0,
    pitch: 0,
    width: '100vw',
    height: '100vh'
  });

  const mapRef = useRef();

  const handleMapClick = (event) => {
```

```

const features = mapRef.current.queryRenderedFeatures(event.point, {
  layers: ['points']
});

if (features.length) {
  const feature = features[0];
  onMarkerClick(feature.properties.id);
}
};

return (
  <MapGL
    {...viewport}
    mapStyle="mapbox://styles/mapbox/streets-v11"
    onViewportChange={nextViewport => setViewport(nextViewport)}
    mapboxApiAccessToken={MAPBOX_TOKEN}
    ref={mapRef}
    onClick={handleMapClick}
  >
    <Source id="geojson-data" type="geojson" data={geoData}>
      <Layer
        id="points"
        type="circle"
        paint={{

```

```
      'circle-radius': 5,  
      'circle-color': '#007cbf'  
    }  
  />  
  
  </Source>  
  
  </MapGL>  
  
);  
};  
  
export default MapComponent;
```