Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтерактивної мапи населеного пункту з використанням технологій Leaflet та React

Виконав: студент IV курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Мартинів О.С.
(підпис) (прізвище та ініціали)

Керівник _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2023

АНОТАЦІЯ

Кваліфікаційна робота бакалавра за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42, 2023 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: 61 с., 22 рис., 1 табл., 3 додатки, презентація.

Тема: Розробка інтерактивної мапи населеного пункту з використанням технологій Leaflet та React.

В атестаційній роботі бакалавра висвітлено ключові елементи в розробці інтерактивних веб мап, із використанням як вже наявних інструментів розробки у вигляді пакету Leaflet та React так і створених власноруч на основі мови програмування Java Script, HTML та CSS. В процесі було створено продукт, що на своїй меті демонструє роботу інтерактивної веб мапи, в основі якої є простий інтуїтивно зрозумілий дизайн інтерфейсів, оптимізоване завантаження веб сторінки, інтеграція під основні електронні пристрої, що можуть працювати у веб просторі, можливість гнучкого налаштування під потреби користувача, елементи управління для створення і ведення організаційних робіт компаній в межах використання інтерактивної мапи. Зображено можливості використання шарованих веб мап для відтворення тих чи інших сценаріїв подій, що будуть відображатися у електронному пристрої користувача інтерактивної мапи.

Ключові слова: веб технології, інтерактивна веб мапа, React JS, Leaflet, модель клієнт сервер, віртуальна об’єктна модель документа.

ANNOTATION

Bachelor's qualification work in the specialty 121 - Software Engineering. Ternopil National Technical University named after Ivan Puluj, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-42, 2023. Explanatory note to the qualification work for the bachelor's degree contains: 61 p., 22 figures, 1 table, 3 appendices, presentation.

Topic: Development of an interactive map of a settlement using Leaflet and React technologies.

The bachelor's attestation work highlights the key elements in the development of interactive web maps, using both existing development tools in the form of the Leaflet and React packages and those created on the Java Script, HTML and CSS programming language. In the process, a product was created that demonstrates the operation of an interactive web map, based on a simple intuitive interface design, optimized web page loading, integration for major electronic devices that can work in the web space, the ability to flexibly customize to the user's needs, controls for creating and managing organizational work of companies within the framework of using an interactive map. The article shows the possibilities of using layered web maps to recreate certain scenarios of events that will be displayed in the electronic device of the interactive map user.

Keywords: web technologies, interactive web map, React JS, Leaflet, client-server model, virtual document object model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS (Java Script) – мова програмування, використовується для створення інтерактивних та динамічних веб-сторінок.

API (Application Programming Interface) – прикладний програмний інтерфейс.

HTML (HyperText Markup Language) – мова розмітки гіпертексту.

CSS (Cascading Style Sheets) – спеціальна мова стилю сторінок.

UML (Unified Modeling Language) – уніфікована мова моделювання.

DOM (Document Object Model) – об’єктна модель веб документа.

Плагін (Plug-in) – незалежно скомпільований програмний модуль.

Тайли (Tiles) - зображення, які використовується для створення текстур шляхом укладання копій цього зображення на зразок «черепиці», тобто розміщуючи копії текстури пліч-о-пліч так, щоб місця стику були непомітні.

ВСТУП

Розробка інтерактивних мап певних населених пунктів являє собою створення координаційної системи задля забезпечення користувачеві можливості орієнтації у просторі за допомогою відповідних інтерфейсних рішень. Важливим є також інтеграція у інтерактивну мапу основних картографічних інструментів для орієнтації у просторі таких як компас, лінійка та інших додаткових інструментів для прокладання маршрутів чи виділення необхідних користувачеві областей.

У загальній картографії розуміють такі поняття як масштаб, деталізація, позначення, легенда мапи. Для реалізації відповідних значень у веб мапах використовують відповідні кодові структури – алгоритми. До прикладу реалізація масштабування за своїми властивостями схожа до альбому фотографій, де кожна наступна фотографія розбивається на декілька, але більш детальніших. В матеріальній картографії довільне масштабування було б доволі ємнісно реалізувати, але для створення такого довільного масштабування на веб сторінці достатньо кількох стрічок коду, що будуть реалізовувати цей механізм.

Щодо мною поставлених завдань є реалізація складної картографічної системи, яка буде виконувати не тільки функції простої веб мапи, а також буде певним веб довідником місцевості, надасть користувачеві можливість довідатись необхідну інформацію, до прикладу про час роботи бібліотеки, а додавши модуль для взаємодії між користувачами – записатись на відвідування чи провести консультацією із працівником бібліотеки, який буде відповідати за корпоративне використання веб мапи і тим самим за чат із користувачами.

Для незареєстрованих користувачів функціонал мапи буде обмежений базовими картографічними інструментами і спрощеним варіантом взаємодії із даними про певні локації відображені на веб мапі.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд конкурентів

Серед лідерів картографічних веб сервісів можна виділити продукт корпорації Google – Google Maps.

Рисунок 1.1 – Логотип Google Maps

Продукт від Google містить у собі достатній спектр інструментів для зручного користування їх мапою, таких як лінійка для вимірювання відстаней і площ, при цьому будувати можна лише довільною ламаною лінією, можливість змінювати масштаб мапи, а також її стиль відображення, прокладати шлях, визначати власну геолокацію за наявності у відповідного пристрою такої можливості, а також компас [1]. Додатково їх мапи містять у собі елементи взаємодії між користувачами такі як відмітки про заклади, із профільними сторінками у яких користувачі можуть додавати коментарі, а також ставити оцінку. Одним із найбільш цікавих і найменш розвинутих сервісів саме в Україні є Google Street View – або ж панорамний огляд місцевості, що уявляє собою спеціальний вид зображень, які дозволяють користувачеві оглянути фото місцевості від точки з якої була сфотографована місцевість на всі 360 градусів.

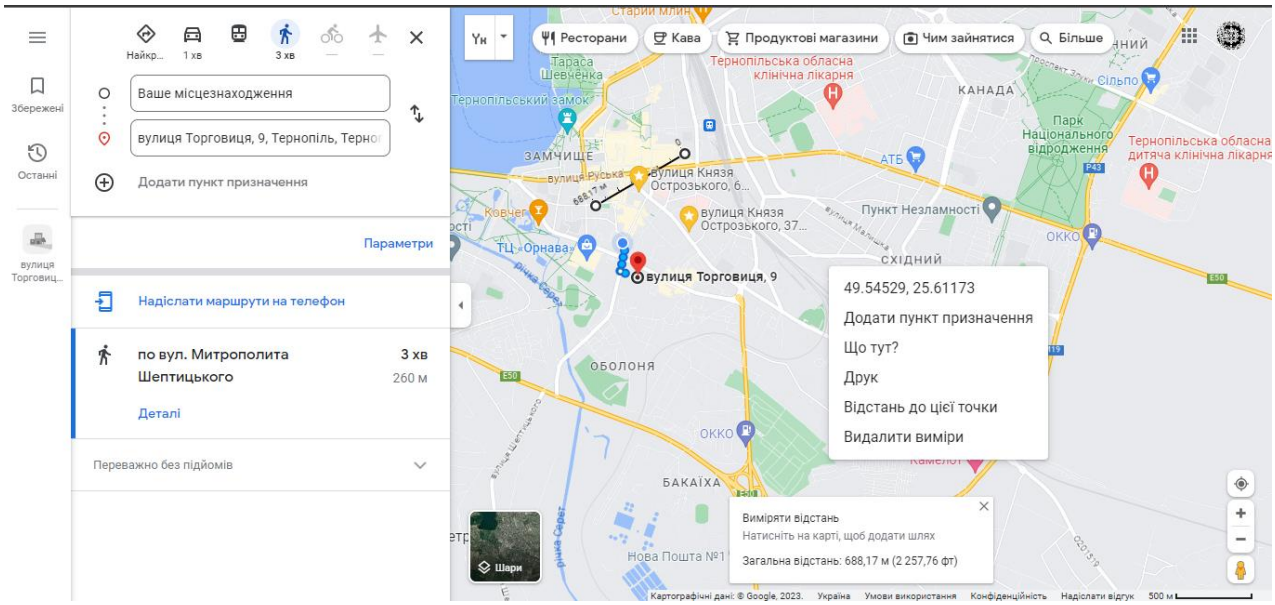

Рисунок 1.2 – Зображення інтерфесу веб мапи від корпорації Google

Загалом картографічна система від компанії Google є гарним інструментом для орієнтування користувача у просторі, але має свої обмеження. Основний з них це направлений вектор на користування мапою саме туристами.

Серед великих картографічних сервісів також можна згадати про Bing Maps від корпорації Microsoft. Сервіс собою демонструє високо продуктивну картографічну систему, яка відразу інтегрована у сучасні версії програмного забезпечення Microsoft Windows починаючи із версії 8, а також доступної широкій масі користувачів різних пристроїв - веб версії [2]. Продуктивність і є ключовою особливістю цієї мапи. Загалом функціонально вона абсолютно подібна до її ключового конкурента від Google. Проте веб мапа від Microsoft має детальнішу інформацію про назви вулиць та населених пунктів і має ширші можливості в інтерфейсних налаштуваннях, а також має власного постачальника картографічних даних.

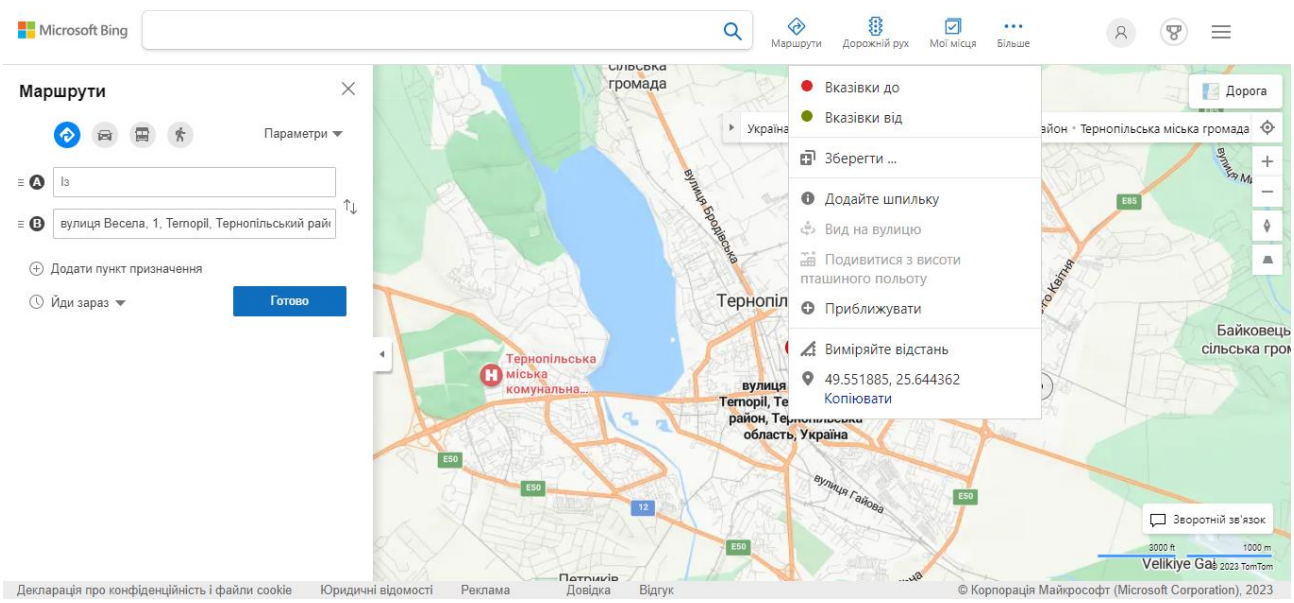

Рисунок 1.3 – Вигляд інтерфейсу мапи від Bing

У вибірку конкурентів варто додати зовсім нову українську картографічну систему під назвою DeepState Map. Цей сервіс якраз таки має реалізацію на інструментах від Leaflet та React JS, проте заточений він під умовне користування для моніторингу військової ситуації в Україні. Має локальну популярність саме в Україні, але присутня також і міжнародна підтримка.

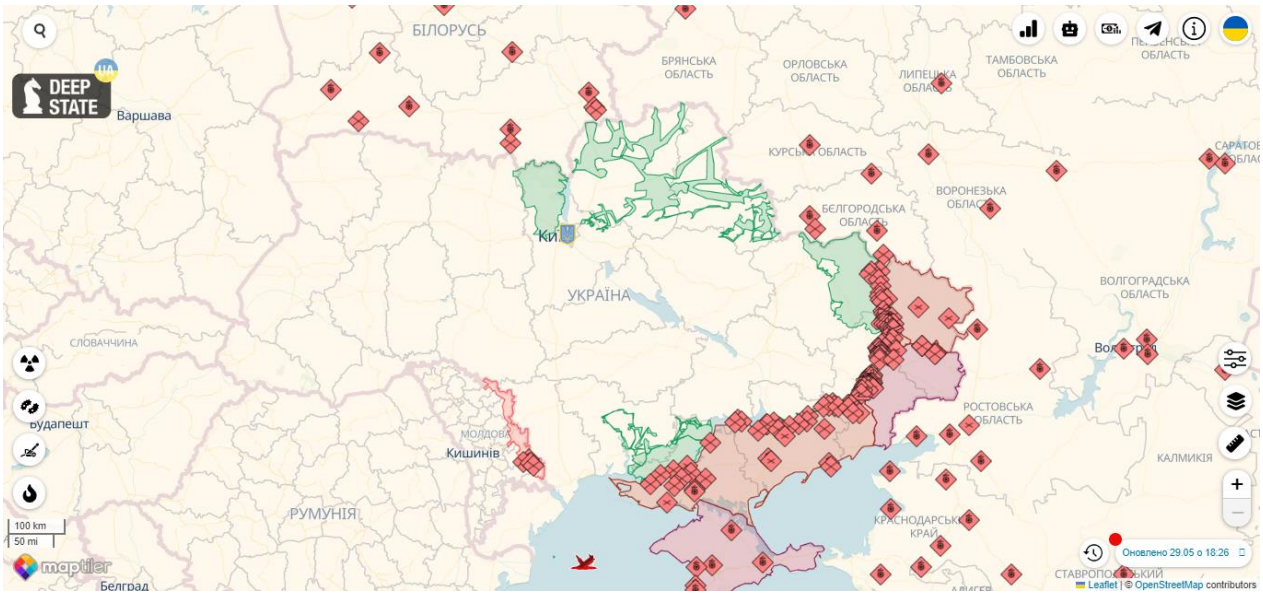

Рисунок 1.4 – Вигляд інтерфейсу мапи від DeepState Map

Що хочеться виділити про DeepState Map, це реалізація системи шарів, можливість налаштовувати відображення на мапі саме тих даних, що потрібні користувачеві.

1.2 Обґрунтування вибору напрямку дослідження

На роль середовища розробки було обрано веб простір, оскільки розробка такого роду проєктів є найбільш ефективною і функціонально необмеженою саме у веб просторі. Зручність у використанні веб сайтів для простих користувачів практично немає конкурентів, оскільки від користувача потребується лише пристрій, що матиме можливість виходу у мережу інтернет та веб браузер із підтримкою актуальних інструментів відображення веб вмісту та виконання скриптів. На розробника покладено вирішення більшості проблем із оптимізацією під різні пристрої, адже ефективніше оптимізувувати проєкт під рамки різних пристроїв, а ніж дробити проєкт на підпроєкти, що будуть розроблені під певну операційну систему, пристрій чи систему пристроїв.

Детальніше про самі інструменти розробки:

Для створення коду проєкту було обрано рішення від Microsoft, а саме Microsoft Visual Studio Code.

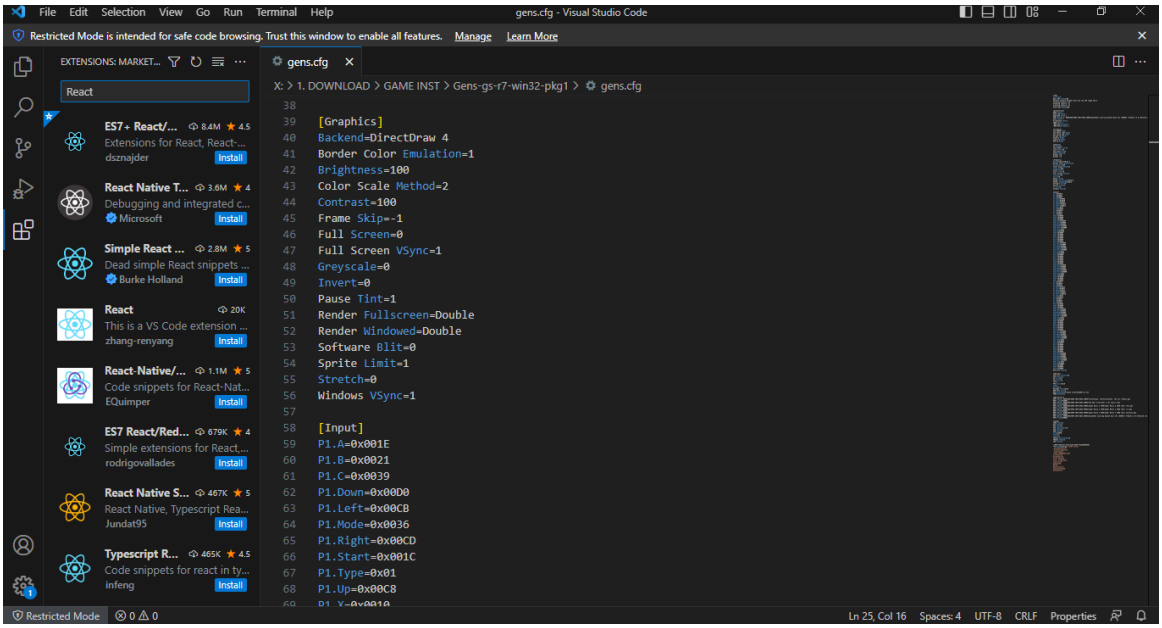

Рисунок 1.5 – Відображення інтерфейсу Microsoft Visual Studio Code

Це програмне рішення для розробки було обрано через можливість модифікації його інструментарію під конкретну задачу завантаженням додаткових пакетів, які направлені саме у веб розробку. А також завдяки цим доповненням розробник має можливість із легкістю протестувати написаний ним код, або ж моніторити зміни у реальному часі під час написання тих чи інших частин коду веб сторінки.

Для створення інтерактивної мапи необхідні джерела інформації, із цим можна впоратись використовуючи загально доступні джерела, які не накладають заборону на зберігання та використання їх даних у створюваних продуктах. Також для веб мапи необхідна сама мапа, у нашому випадку це має бути система зображень, такі послуги у безкоштовному форматі надають різні картографічні сервіси через API підключення у створюваних веб сторінках. Серед доступних було проаналізовано їх варіації мап для використання та виділено сервіс MapTiler, як найдоступніший і зручний до інтеграції у створюваний нами проєкт.

Для тестування будуть використанні системні інструменти тестування та наявні у браузері базованому на Chromium ядрі.

1.2.1 Використання мови програмування JavaScript

JavaScript – це універсальне рішення серед мов програмування, яка широко використовується для створення інтерактивних веб сторінок. Саме за рахунок її розгалуженості і розвитку як зі сторони головних розробників так і користувацьких екосистем бібліотек, таких як Leaflet і React, вона собою являє чудовий інструмент для розробки проєктів у напрямку веб, включно із розробкою інтерактивних, динамічних веб мап.

Через хорошу гнучкість у роботі з об’єктною моделлю документів, JavaScript дозволяє ефективно інтегрувати до веб сторінок інтерактивні елементи, ефективні скрипти, а конкретно у розробці веб мап можливість будувати ключові алгоритми такі як розвертання карти, масштабування, маркери, спливаючі вікна для підказок чи тому подібних інтерактивних елементів, модулів.

Завдяки JavaScript можна реалізувати систему багатошарових мап. Де кожен шар як похідний клас від базового буде реалізовувати певну інформаційну систему (згуртовану інформацію певного характеру, до прикладу шар що відображатиме інформацію про залізничні колії, зупинки потягів, маршрути та прибуття відбуття транспорту).

Ще однією ключовою особливість саме JS, за рахунок якої її дуже вигідно використовувати у розробці таких чи подібних інформаційних систем є можливість підключення через API додаткових джерел інформації, інформаційних баз, які використовуючи наявні дані зі сторонніх систем будуть надавати їх у нашу систему, не використовуючи користувацького та серверного ресурсу [3].

Важливо зазначити і кросплатформові властивості мови програмування JavaScript, це важливо при розробці систем що будуть адаптовані відразу на широку кількість пристроїв.

1.2.2 Використання мови гіпертекстової розмітки HTML

HTML (Hypertext Markup Language - мова гіпертекстової розмітки) - це фундамент майже будь-якої веб сторінки, і її використання пов’язано із наступних вагомих причин:

1) Контроль та гнучкість над структурою та поведінкою веб сторінки – мова HTML організовує саме представлення контенту у найбільшому відсотку веб сторінок, вона дозволяє створити ієрархічний макет, який визначає кожен елемент веб сторінки від заголовку і текстового абзацу до розміщення зображення та таблиць, а легкість в інтеграції з тим ж JavaScript дозволить це все привести у цільну динамічну систему, як того забажає розробник за його власним шаблоном.
2) Широка сумісність – HTML як один з основних стовпів веб простору є інтегрований у кожний веб браузер (за виключення експериментальних що використовують за базу веб сторінки іншу мову програмування), що робить його вседоступним для широкого кола користувачів.

1.2.3 Використання мови стилізації CSS

CSS (каскадні таблиці стилів) використовуються для дизайн частини створюваного веб сайту. Актуальною версією є CSS3, яка дозволяє максимально детально налаштувати візуальних вигляд сторінки відповідно до потреб розробників, у поєднанні із JavaScript дозволить створювати інтерактивні дизайн рішення, як то до прикладу персоналізоване завантаження сторінки, чи зміна кольорової гами відповідно до часу у користувача. Загалом немає обмежень щодо використання комбінацій.

Візуальна складова веб сторінки особливо важлива для інтерактивних веб мап, бо ключовий функціонал веб сайту випадає саме на візуальне сприйняття. І використовуючи CSS, розробники можуть визначати зовнішній вигляд, макет і поведінку елементів карти, таких як маркери, полігони і підписи. CSS дозволяє точно керувати такими їх властивостями як колір, розмір, прозорість та анімація, що дає змогу створювати візуально привабливі та інтуїтивно зрозумілі інтерфейси мап.

1.3 Технічний аспект проблеми

1.3.1 Використання Leaflet для розробки інтерактивних веб мап

Використання бібліотеки Leaflet дозволить в рази спростити розробку інтерактивних веб мап, завдяки великій кількості актуальних інструментів розробки саме у сфері картографії [4]. Ключовою перевагою над іншими інструментами саме цієї бібліотеки є широка екосистема плагінів [5], що розроблені і підтримуються спільнотою розробників, які розширюють функціонал створюваного проєкту у відповідності із напрямком розробки, як то до прикладу надання можливості інтеграції із зовнішніми сервісами систем геолокації чи отримання даних від стороніх джерел, до прикладу актуальних даних про магнітні поля у тій чи іншій зоні мапи.

Бібліотека Leaflet підтримує широкий спектр постачальників мап, що дозволяє розробникам налаштувати власний проєкт відповідно до поставленої задачі. Також бібліотека вирізняється й тим, що активно підтримує міжплатформову розробку, тобто розроблена мапа буде із легкістю інтегрована для різноманітних електронних пристроїв, що є надзвичайно важливою рисою, яка вирізняє саме Leaflet серед бібліотек що спеціалізуються на картографічних задачах.

2 ПРОЄКТУВАННЯ ІНТЕРАКТИВНОЇ МАПИ

2.1 Розробка моделі предметної області

При створенні моделі інтерактивної веб мапи весь функціонал та можливості було поділено на 10 функціональних груп взаємодій, що визначаються для певного рангу користувача, серед користувачів виділено наступних 3 ранги: «Простий користувач», «Зареєстрований користувач» та «Адміністратор». Стисло інформацію про розподіл подано у таблиці 2.1.

Отже базовий ранг – «Простий користувач» цього рангу набуває будь-який користувач що завітає на веб сторінку, йому будуть доступні наступні групи функціоналу: «Перегляд мапи», «Зареєструватися», «Пошук та навігація», «Відображення довідкової інформації», «Вимірювальні роботи», «Технічна підтримка».

Далі розглянемо детальніше кожну групу функціоналу:

«Перегляд мапи»:

- Користувач може відкрити та переглядати географічні об’єкти місцевості у веб мапі;
- Користувач має можливість використовувати інструменти для зміни масштабу мапи;
- Користувач може пересувати мапу задля відображення інших географічних об’єктів на мапі.

«Зареєструватися»:

- Користувачеві доступна функція реєстрації, що дозволить отримати ранг «Зареєстрований користувач»;
- Користувачеві доступна функція відновлення облікового запису у випадках втрати даних для входу.

«Пошук та навігація»:

- Користувач може використовувати вікно пошуку, для знаходження необхідного географічного об’єкту відповідно до введених даних;

- Користувач може отримати у відповідь на пошуковий запит запропонований алгоритмом маршрут до шуканої точки.

«Відображення довідкової інформації»:

- Користувач може отримати додаткову інформації про обраний ним географічний об’єкт;
- Користувач може відкривати сповіщення що містять додаткову інформацію;
- Користувач може переглядати додатковий фото та відео матеріал про обраний географічний об’єкт.

«Вимірювальні роботи»:

- Користувач може використовувати картографічні інструменти для вимірювання відстаней та площі;
- Користувач може використовувати картографічні інструменти для орієнтації у просторі, визначення сторон світу.

«Технічна підтримка»:

- Користувач може створювати форми із запитами до адміністрації.

Ранг «Зареєстрований користувач» є похідним від рангу «Простий користувач», а отже повністю наслідує його функціонал, а також має додаткові функціональні групи: «Взаємодія із користувачами», «Запропонувати зміни», «Зміни в обліковому записі».

Відповідно огляд можливостей груп функціоналу:

«Взаємодія із користувачами»:

- Користувач має можливість залишати відгуки під певними географічними об’єктами, а також відповідати на відгуки інших користувачів;
- Користувач може вказувати персональну оцінку якомусь географічному об’єкту;
- Користувач може завантажити фото або відео матеріали для обраного географічного об’єкта.

«Запропонувати зміни»:

- Користувач може надіслати на розгляд адміністрації пропозицію про внесення зміни інформації до певного географічного об’єкта;
- Користувач може надіслати на розгляд адміністрації пропозицію про додавання або видалення певного географічного об’єкта.

«Зміни в обліковому записі»:

- Користувач має можливість вийти із облікового запису отримавши ранг «Простий користувач»;
- Користувачеві доступні поля для внесення персональних даних їх зміни;
- Користувач може завантажити персональні світлини для облікового запису;
- Користувач може видалити обліковий запис, автоматично після цього здобуваючи ранг «Простий користувач».

Користувачу із рангом «Адміністратор» доступні функціональні групи рангів нижче, але із виключенням груп, що стосуються запитів до адміністрації, а саме «Технічна підтримка» та «Запропонувати зміни», вони заміщені розширеними варіантами у додатковій групі функціоналу під назвою «Адміністрування мапи» і відповідно до вимог обліковий запис рангу «Адміністратор» створюється при розробці інтерактивної веб мапи, а отже додавати додаткових облікових записів такого рангу можливо лише через звернення до розробника, тому також відсутній функціонал групи «Зареєструватися».

«Адміністрування мапи» ця група функціоналу містить наступні можливості:

- Адміністратор може увійти у відповідного рангу обліковий запис;
- Адміністратор може розглядати і виносити вердикти по справам щодо змін, додавання чи видалення географічних об’єктів чи інформації про них запропонованих зареєстрованими користувачами;
- Адміністратор може самостійно додавати відсутні географічні об’єкти;
- Адміністратор може самостійно змінювати вже існуючі географічні об’єкти;

- Адміністратор може внести обмеження для обраних облікових записів зареєстрованих користувачів;
- Адміністратор може видалити обліковий запис певного користувача;
- Адміністратор може відповідати на запити користувачів;
- Адміністратор має доступ до відповідної панелі інструментів для адміністрування інтерактивної веб мапи.

Таблиця 2.1 – Розподіл функціональних груп між рангами користувачів

Функціональна група / Ранг	**Простий користувач**	**Зареєстрований користувач**	**Адміністратор**
Перегляд мапи	+	+	+
Зареєструватися	+	+	
Пошук та навігація	+	+	+
Відображення довідкової інформації	+	+	+
Вимірювальні роботи	+	+	+
Технічна підтримка	+	+	
Взаємодія із користувачами		+	+
Запропонувати зміни		+	
Зміни в обліковому записі		+	+
Адміністрування мапи			+

Відповідно до отриманої нами задачі до проєкту буде можливість із легкістю інтегрувати різноманітні модулі через спеціальний функціональний буфер підключення модулів для відображення чи взаємодії із картографічними даними, що надасть у подальшому розширювати, або обмежувати функціонал

відповідно до потреб замовника чи модифікації з ціллю зміни вектора використання мапи.

2.1.1 Інструменти пакету React JS

Рішення про вибір React JS в якості ядра розробки виникло одразу ж після дослідження актуальних інструментів розробки веб мап, оскільки попередньо описана бібліотека Leaflet має повну підтримку і операційну інтеграцію із інструментами бібліотеки React JS, утворюючи єдину систему – React Leaflet.

Загалом ядро React JS має цілий спектр переваг:

1) Це його висока популярність серед розробників, а отже підтримка всіх актуальних інструментів розробки і загальна підтримка функціоналу, додатково це об’єднання і форуми розробників, що використовують цю бібліотеку, які допомагають один одному у вирішенні специфічних проблем при розробці проєктів із використанням цієї технології;
2) Модульність коду – дозволяє не тільки розбивати проєкт на окремо робочі кластери коду, але і об’єднувати весь проєкт у високопродуктивний механізм, деталі якого можна модифікувати не порушуючи загальну функціональність;
3) Багаторазове використання – це саме те, що робить код ефективним, і бібліотека ReactJS одна з найкращих, що активно використовує це. Завдяки такому методу, можна оптимізувати сам процес розробки, а також заощадити використовуваний проєктний ресурс;
4) Віртуальна реалізація DOM (Document Object Model) – це останній пункт, але найбільш суттєвий, реалізація такої моделі в React JS дозволила створювати повністю динамічну веб сторінку [6], при цьому виконання кожного інтерактивного елемента буде орієнтоване на увазі користувача, тобто частковий рендеринг того, що потрібно саме тут і

зараз, це в рази підвищує ефективність коду у порівнянні зі стандартною моделлю надання інтерактивного доступу до структури веб сторінки і роботу цієї системи наведено на наступній ілюстрації:

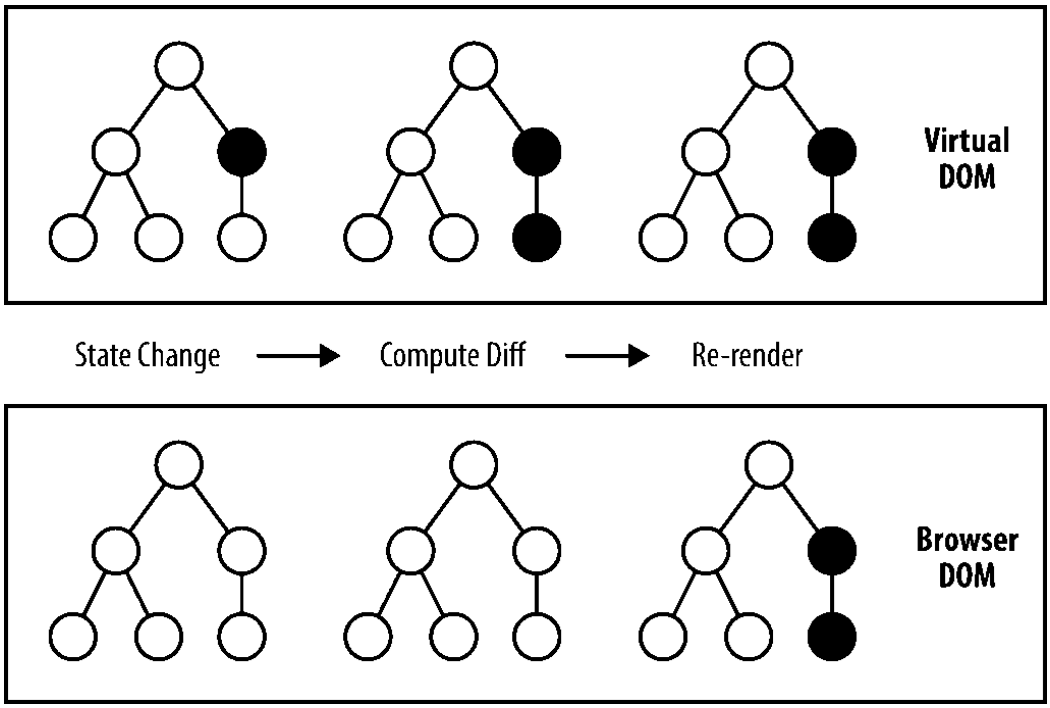

Рисунок 2.1 – Порівняння схеми рендерингу VDOM ReactJS та класичної DOM

Загалом усе це робить ReactJS найбільш привабливим та ефективним інструментом при розробці динамічної бази для картографічного веб проєкту, для реалізації міжплатформових інтерфейсів, що однаково швидко і якісно будуть виконувати свою функціональну частину.

2.2 Розробка бізнес моделі

Обов’язково варто виділити опис деталей про бізнес складову та варіанти використання для нашого проєкту, оскільки це є важливим етапом для розробки та підтримки проекту.

Основне направлення розробки картографія обране із огляду на конкурентне середовище, яке стисло можна назвати монополізованим гігантами ІТ-індустрії та активним розвитком цифровізації держав [7], коли сервіси та послуги активно перетікають у веб середовища, як то до прикладу реалізація в Україні масштабного проєкту «Дія», яка є безперечно інноваційним рішенням у світі, тому ідея реалізації системи інтерактивної мапи виникла одразу з огляду на такі світові тенденції.

Кому буде потрібна ця інтерактивна веб мапа? – Відповідь на таке запитання, яке обов’язково мало б виникнути, ще до етапу реалізації в коді, доволі не просте, оскільки по своїй суті проєкт створює не готовий бізнес продукт, а чудову базу для реалізації власних потреб замовника, або в іншому варіанті реалізацію потреб групи замовників, по своїй суті інтерактивна мапа для бізнес рішень може використовуватись як супермаркет для різноманітних магазинів та рекламодавців, які будуть акумулювати власними вкладами підтримку та покращення продукту, а для них середовище нагороджуватиме новими клієнтами.

З іншої сторони проєкт системи інтерактивної мапи можна реалізовувати як чисто картографічний продукт, який буде розповсюджуватись шляхом купівлі прав на використання та технічної підтримки, бо завдяки моделі коли є користувач і є адміністратор який може вносити правки на мапу не змінюючи нічого глобально, проєкт створює чудові умови для високої стійкості до можливих помилок, більшість помилок виникатиме на етапі конструювання відповідно до потреб замовника і будуть усунені там ж.

Ми будемо спиратись на варіант бізнес реалізації саме як чисто картографічний інструмент тому відповідно до згенерованих нами функціональних груп у попередньому етапі розробки цієї інформаційної системи, можна реалізувати діаграми варіантів використання для некомерційної версії нашої системи. Одразу варто виділити такий аспект, що попри те, що функціоналом ми визначаємо три ранги користувачів по факту для відображення у діаграми варіантів використання нам необхідно лише два актори: «Користувач» та «Адміністратор» - оскільки актор «Користувач» може самовільно змінювати

власний ранг до «Зареєстрований користувач» використовуючи групу функціоналу «Зареєструватися».

Отже перейдемо до реалізації діаграми використання для цього будемо використовувати внутрішній функціонал програмного забезпечення для легкої та продуктивної розробки UML-діаграм – IBM Rational Software Architect [8].

2.2.1 Діаграма варіантів використання для актора «Користувач»

Згенеруємо нашу першу діаграму для актора «Користувач», після чого надамо опис кожного варіанту використання:

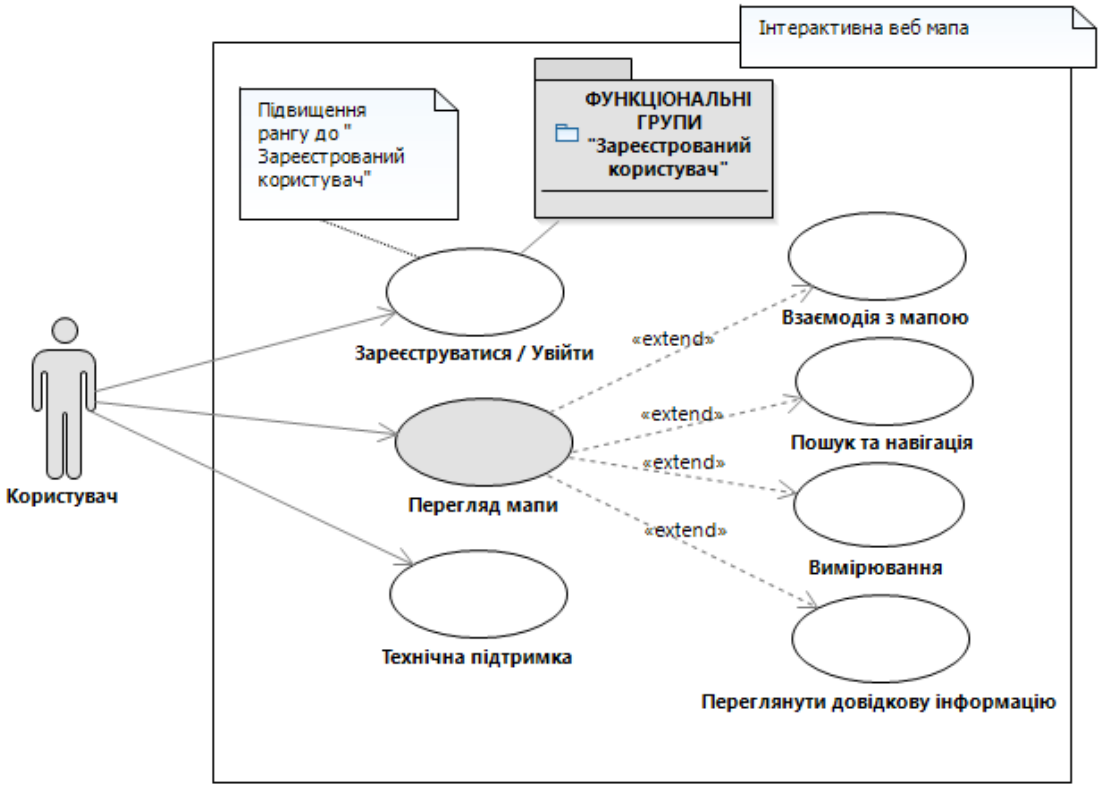

Рисунок 2.2 – Діаграма варіантів використання для актора «Користувач»

Опис варіантів використання:

Варіант використання «Зареєструватися / Увійти»

Короткий опис:

Цей варіант використання дозволяє створити обліковий запис рангу «Зареєстрований користувач» для отримання доступу користувачеві до розширених можливостей функціоналу функціональної групи відповідного рангу.

Основний потік подій:

1. Користувач успішно відкриває веб сторінку інтерактивної мапи;
2. Після чого у області управління акаунтом обирає та натискає кнопку «Зареєструватись» або «Увійти»;
3. У результаті отримує інтерфейс для введення персональних даних;
4. Вводить персональні дані;
5. Натискає кнопку «Зареєструватися» або «Увійти»;
6. Отримує ранг «Зареєстрований користувач».

Альтернативний потік подій:

1. Користувач вводить неправильні персональні дані;
2. Отримує сповіщення про невідповідність введених даних;
3. Повертається до кроку введення персональних даних для входу чи реєстрації.

Передумови:

Користувач має ранг «Користувач», система стабільна у роботі.

Післяумови:

При виконанні цього варіанту використання користувач успішно набуває рангу «Зареєстрований користувач» та отримує розширений функціонал взаємодії із мапою.

Варіант використання «Перегляд мапи»

Короткий опис:

Надає можливість користувачеві переглядати мапу.

Основний потік подій:

1. Користувач успішно заходить на веб сторінку.

Альтернативний потік подій:

2. Користувач успішно заходить на веб сторінку, але мапа не відображається;
3. Система видає причини.

Передумови:

Система у стабільному стані, у користувача наявне підключення до мережі інтернет.

Післяумови:

Можливість використовувати додаткові інструменти на веб сторінці.

Варіант використання «Взаємодія з мапою»

Короткий опис:

Дозвіл користувачеві масштабувати та рухатись по мапі.

Основний потік подій:

1. Користувач обирає інструменти управління мапою;
2. Користувач працює з інструментами.

Альтернативний потік подій:

1. Користувач не може працювати із інструментами;
2. Система видає причини.

Передумови:

Система у стабільному стані, мапа та інтерфейс відображається коректно.

Післяумови:

Збереження результату взаємодії з мапою, для подальшої навігації, успішне завершення роботи з інструментом.

Варіант використання «Пошук та навігація»

Короткий опис:

Користувач отримує можливість знайти необхідний йому географічний об’єкт на мапі.

Основний потік подій:

1. Користувач обирає інструмент для пошуку на мапі;
2. Користувач у вікні пошуку вводить необхідний запит;
3. Користувач натискає кнопку пошуку;

4. Користувач отримує відповідні дані та розташування шуканого об’єкта.

Альтернативний потік подій:

1. Користувач вводить дані некоректно;
2. Користувач отримує відповідь із вказаною помилкою.

Передумови:

Система у стабільному стані, мапа та інтерфейс відображається коректно.

Післяумови:

Шуканий об’єкт виділено, система працює стабільно, дані збережено, успішне завершення роботи з інструментом.

Варіант використання «Вимірювання»

Короткий опис:

Користувач використовує інструменти для вимірювання відстаней, площі та визначення сторони світу.

Основний потік подій:

1. Користувач обирає інструмент для виконання вимірювань;
2. Користувач використовує інструмент на мапі;
3. Користувач отримує відповідні розрахунки у вікні інструменту.

Альтернативний потік подій*:*

1. Користувач не може обрати інструмент;
2. Система видає причини.

Передумови:

Система у стабільному стані, мапа та інтерфейс відображається коректно.

Післяумови:

Успішне завершення роботи користувача з інструментом.

Варіант використання «Переглянути довідкову інформацію»

Короткий опис:

Користувач може переглянути довідкову інформацію до географічних об’єктів на мапі.

Основний потік подій:

1. Користувач знаходить на мапі географічний об’єкт;

2. Користувач виділяє відповідний об’єкт;
3. Користувач отримує віконце, що містить певну інформацію про географічний об’єкт, його медіа дані;

Альтернативний потік подій*:*

1. Користувач не може виділити необхідний об’єкт;
2. Система видає причини.

Передумови:

Система працює стабільно, мапа та інтерфейс відображаються коректно, на мапі внесено інформацію до географічних об’єктів.

Післяумови:

Користувач успішно завершує роботу із інструментом.

Варіант використання «Технічна підтримка»

Короткий опис:

Користувач звертається за технічною підтримкою до адміністрації веб сайту.

Основний потік подій:

1. Користувач завітав на сайт;
2. Користувач знайшов якусь помилку чи недолік;
3. Користувач натискає кнопку «Технічна підтримка»;
4. Користувач заповнює форму звернення вказуючи персональні дані;
5. Користувач натискає кнопку «Надіслати».

Альтернативний потік подій:

1. Користувач не може натиснути кнопку «Технічна підтримка»;
2. Система видає причини.

Передумови:

Система працює стабільно, користувачеві завантажився мінімальний інтерфейс із кнопкою «Технічна підтримка».

Післяумови:

Система надсилає адміністрації лист із відповідним зверненням.

На моделі варіантів використання відображено перехід до функціональної групи «Зареєстрований користувач», тому наведемоо також відповідний розширений функціонал в цій групі для актора «Користувач». Зображено в додатку В.

2.2.2 Діаграма варіантів використання для актора «Адміністратор»

Після реалізації діаграм варіантів використання для актора «Користувач», необхідно реалізувати діаграму варіантів використання також для актора «Адміністратор»:

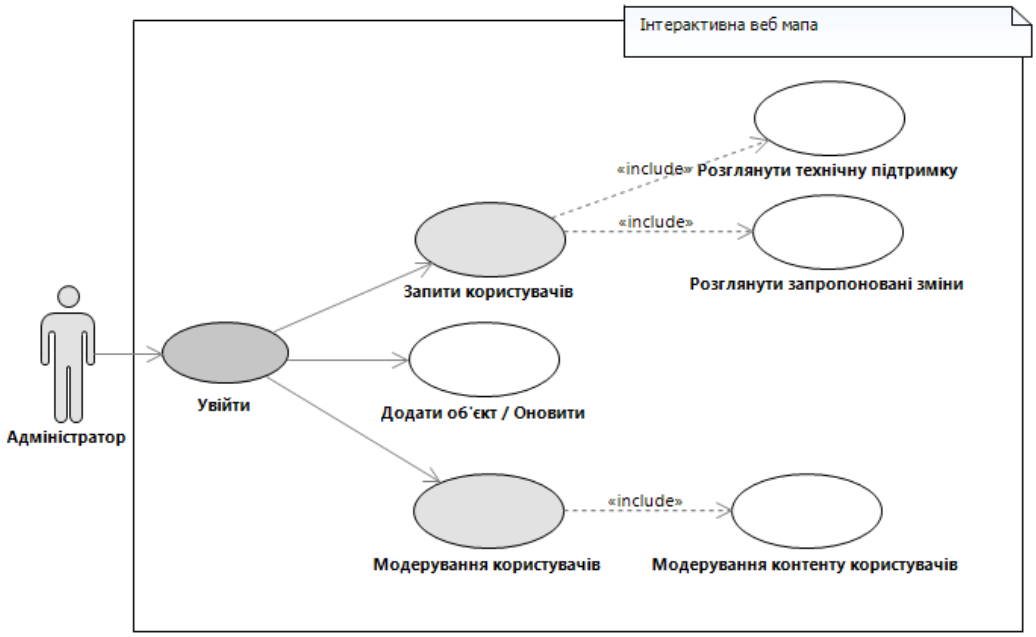

Рисунок 2.3 – Діаграма варіантів використання для актора «Адміністратор»

Опис варіантів використання:

Варіант використання «Увійти»

Короткий опис:

Адміністратор може увійти у акаунт відповідного рангу «Адміністратор».

Основний потік подій:

1. Адміністратор як і Користувач знаходить на веб сторінці системи, область авторизації;
2. Адміністратор вводить надані розробником дані для входу в систему;
3. Система надає Адміністратору ранг «Адміністратор» та відповідні інструменти;

Альтернативний потік подій:

1. Адміністратор вводить неправильні авторизаційні дані;
2. Якщо адміністратор немає авторизаційних даних, він має звернутися до розробника для створення таких.

Передумови:

Система працює, у Адміністратора є актуальні авторизаційні дані.

Післяумови:

Надання функціоналу для адміністрування інтерактивної веб мапи.

Варіант використання «Запити користувачів»

Короткий опис:

Адміністратору доступна панель інструментів для перегляду запитів користувачів.

Основний потік подій:

1. Адміністратор натискає на відповідний пункт у панелі інструментів.

Альтернативний потік подій:

1. Система сповіщає про відсутність запитів.

Передумови:

Система працює, Адміністратор має відповідний ранг, є хоча б один запит від Користувача.

Післяумови:

Адміністратор отримує доступ до наступних варіантів використання: «Розглянути технічну підтримку» та «Розглянути запропоновані зміни».

Варіант використання «Розглянути технічну підтримку»

Короткий опис:

Адміністратор має можливість розглянути користувацьке звернення до технічної підтримки, для подальшого контакту із відповідним користувачем.

Основний потік подій:

1. Адміністратор натискає на кнопку «Запити до Технічної підтримки»;
2. Адміністратор обирає відповідне звернення;
3. Адміністратор обробляє вміст звернення;
4. Адміністратор вирішує питання запиту;
5. Адміністратор відмічає звернення як завершене натиснувши кнопку «Запит опрацьовано Успішно»;
6. Система зберігає зміни.

Альтернативний потік подій:

1. Адміністратор не може вирішити питання запиту;
2. Адміністратор вибирає кнопку «Запити не опрацьовано»;
3. Адміністратор у сповіщенні вказує на причину;
4. Система відмічає запит незавершеним та зберігає дані.

Передумови:

Система працює, інтерфейс відображається коректно, адміністратор має відповідний ранг, є хоча б один запит від Користувача.

Післяумови:

Система зберігає статус запиту.

Варіант використання «Розглянути запропоновані зміни»

Короткий опис:

Адміністратору розглядає запропоновані користувачами зміни до мапи та підтверджує або відхиляє запропоновану зміну.

Основний потік подій:

1. Адміністратор натискає кнопку «Запропоновані зміни»;
2. Адміністратор обирає зі списку якийсь користувацький запит;
3. Адміністратор оглядає форму запиту;
4. Адміністратор підтверджує зміни кнопкою «Прийняти»;
5. Система відзначає запит виконаним;

6. Система вносить зміни до мапи;
7. Система зберігає дані.

Альтернативний потік подій:

1. Адміністратор відмовляється від змін кнопкою «Відхилити»;
2. Система відзначає рішення відхилення;
3. Система не вносить змін до мапи;
4. Система зберігає дані.

Передумови:

Система працює, інтерфейс відображається коректно, адміністратор має відповідний ранг, є хоча б один запит від Користувача.

Післяумови:

Система відповідно відзначає рішення по запитах, а також відповідно до рішення вносить чи відхиляє запропоновані зміни.

Варіант використання «Додати об’єкт / Оновити»

Короткий опис:

Система дозволяє Адміністратору вносити зміни до мапи без затверджень.

Основний потік подій:

1. Адміністратор обирає на мапі локацію;
2. Адміністратор з використанням інструменту додавання додає новий об’єкт на мапу;
3. Адміністратор вказує відомі дані про об’єкт;
4. Адміністратор натискає на кнопку «Додати»
5. Система створює відповідний об’єкт.
6. Система зберігає дані.

Альтернативний потік подій:

1. Адміністратор обирає вже існуючий географічний об’єкт на мапі;
2. За допомогою інструменту оновлення, Адміністратор змінює дані про об’єкт, або видаляє його;
3. Адміністратор натискає на кнопку «Зберегти»;
4. Система зберігає дані.

Передумови:

Система працює стабільно, інтерфейс та мапа відображаються коректно, адміністратор має відповідний ранг.

Післяумови:

Система зберігає внесені зміни до бази даних.

Варіант використання «Модерування користувачів»

Короткий опис:

Система дозволяє Адміністратору модерувати користувачів, відповідно блокувати чи вносити обмеження на використання функціоналу, або знімати ці обмеження, блокування.

Основний потік подій:

1. Адміністратор на панелі адміністрування обирає відповідний інструмент;
2. Адміністратор у списку користувачів знаходить потрібного;
3. Адміністратор за допомогою кнопки «Заблокувати» чи «Обмежити» вносить зміни до можливостей конкретного користувача;
4. Система зберігає зміни.

Альтернативний потік подій:

1. Адміністратор знаходить серед списку заблокованих чи обмежених користувачів;
2. Адміністратор обирає такого користувача;
3. За допомогою кнопок «Розблокувати» або «Обмежити» вносить відповідні зміни;
4. Система зберігає зміни.

Передумови:

Система працює, інтерфейс відображається коректно, у системі зареєстрований хоча б один користувач рангу «Зареєстрований користувач».

Післяумови:

Система вносить змінити і відповідно обмежує користувачів відповідно до рішення Адміністратора, також система надає Адміністратору доступ до варіанту використання «Модерування контенту користувачів».

Варіант використання «Модерування контенту користувачів»

Короткий опис:

Система дозволяє Адміністратору видаляти створений користувачами контент, що не відповідає умовам погодженим із замовником мапи.

Основний потік подій:

1. Адміністратор у списку користувачів вибирає необхідного користувача;
2. Адміністратор натискає кнопку «Перегляд контенту користувача»;
3. Адміністратор виділяє контент, який необхідно видалити;
4. Адміністратор натискає кнопку «Видалити»;
5. Система зберігає зміни та відображає їх на мапі.

Альтернативний потік подій:

1. Адміністратор натискає кнопку «Надіслати попередження»;
2. Система генерує сповіщення, яке надходить на акаунт користувача;
3. Система зберігає дані.

Передумови:

Система працює, інтерфейс відображається коректно, у системі зареєстрований хоча б один користувач рангу «Зареєстрований користувач», який розмістив хоча один з варіантів контенту: медіа, відгук, коментар.

Післяумови:

Система виконує рішення Адміністратора, вносить відповідні зміни та зберігає стан.

2.3 Проєктування архітектури

Проєктування архітектури інтерактивної карти, яка містить у собі картографічне відображення вибраного населеного пункту з додатковими інтерактивними шарами, вимагає ретельного підходу для забезпечення надійності та масштабованості рішення, як про це було сказано раніше, оскільки для

підтримки актуальності і робочого стану необхідно дотримуватись якісних рішень в процесі розробки.

Вибраний нами підхід передбачає використання клієнт-серверної моделі архітектури. Архітектура складатиметься з трьох основних компонентів: клієнтської частини, серверної частини та сховища даних. За виглядом фізична система демонструє в роботі модель файлового сервера:

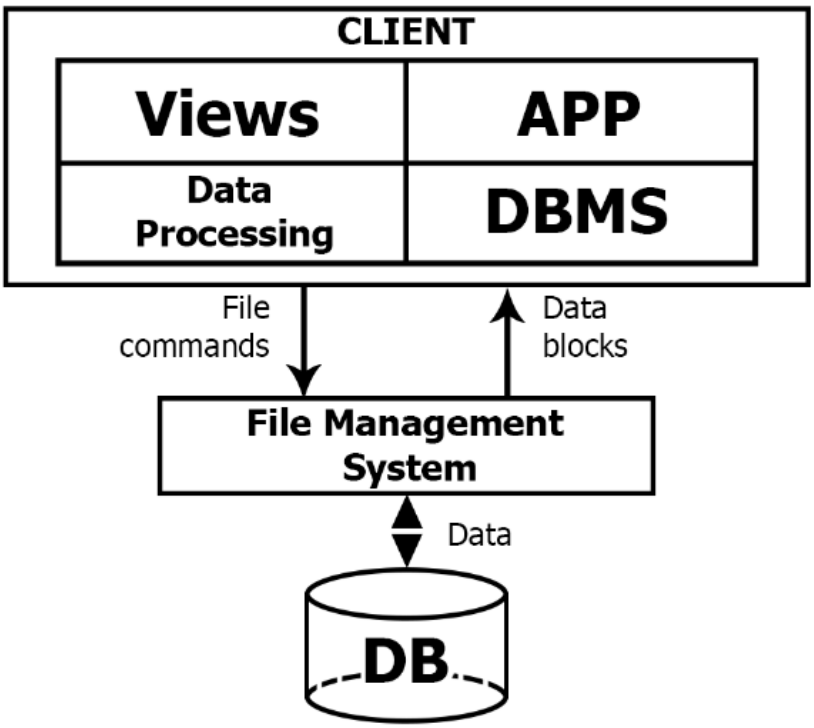

Рисунок 2.4 – Модель файлового сервера

На стороні клієнта React JS слугуватиме фреймворком для побудови інтерактивного інтерфейсу мапи. React-компоненти будуть розроблені та організовані для роботи із шарами та функціоналом мапи. Кожен шар, наприклад, інформація про будівлі, буде реалізовано як окремі React-компоненти, які можна буде динамічно завантажувати та оновлювати на основі віртуальної об’єктної моделі документа. Інструменти бібліотеки Leaflet буде інтегровано у React-компоненти для забезпечення картографічної функціональності, що дозволить рендерити та взаємодіяти з плитками карти, маркерами та елементами візуальної складової функціональних шарів.

2.3.1 Діаграма ієрархії класів

У відповідності до попередніх даних, діаграм варіантів використання, моделі системи було створено відповідно діаграму класів для системи:

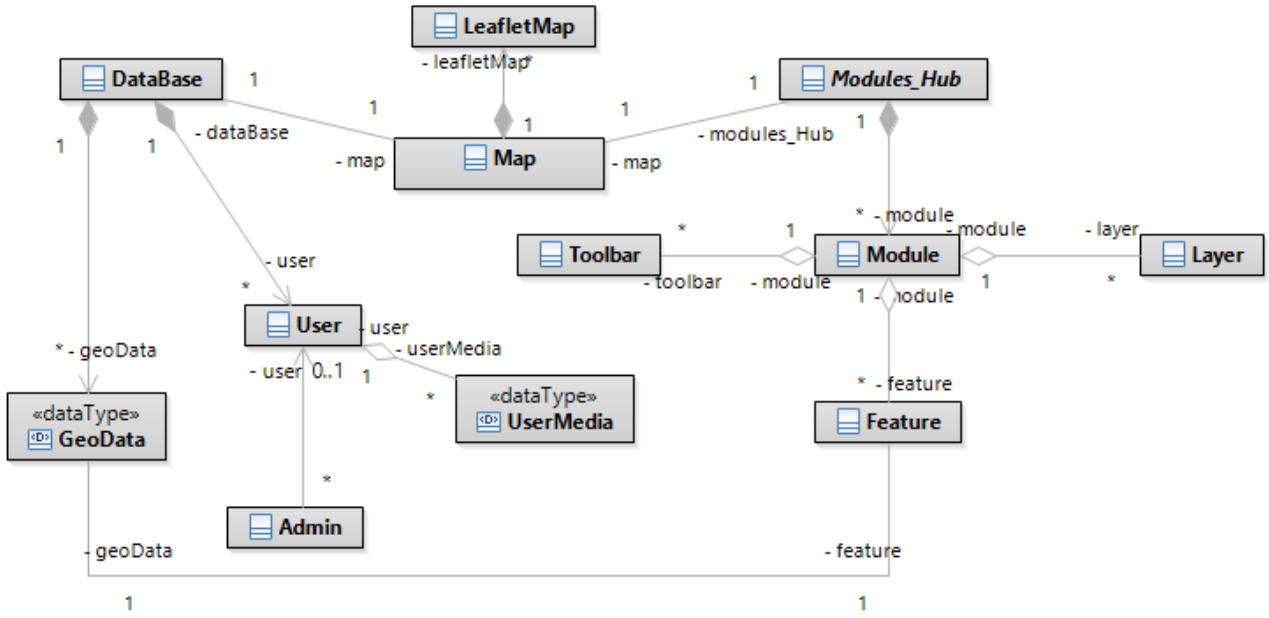

Рисунок 2.5 – Діаграма ієрархії класів для реалізованої інформаційної системи

Детальніше про складові, опис класів:

В центрі системи знаходиться клас “Map”, який представляє головний компонент карти. Клас “Map” має композиційну асоціацію з базовим класом бібліотеки React Leaflet – “LeafletMap”, що відповідає за відображення веб мапи, а отже має містити у собі методи для відображення складових мапи, які наслідує з материнського класу, а сама метод “MapContainer” - для побудови основи під мапу, метод “TileLayer”, який відображає зображення що передаються з бази даних в упорядковану систему для зображення мапи, також клас “Map” містить у собі методи для взаємодії із класом “DataBase”, щоб отримувати географічні дані “GetGeo” та дані користувачів “GetUser”, їх назначення “SetGeo” та “SetUser”. Для взаємодії з класом “Modules_Hub” використовується метод “ShowModules” що надає можливість генерувати на мапі модулі із відповідним функціоналом.

Клас “DataBase” включає композиційну асоціацію з класом “GeoData”, який зберігає географічні дані, а також клас “DataBase” має агрегаційну асоціацію з класом “User”, який представляє користувачів системи для взаємодії з цим класом використовуються методи “Login”, “Registration” та “Delete”, що відповідно до назв українською відповідають за відповідний функціонал що до екземпляру класу.

Клас “User” включає агрегаційну асоціацію з класом “UserMedia”, що дозволяє користувачам додавати медіафайли до своїх профілів викликом методу “AddUserMedia” та перевіряти і виводити через метод “CheckUserMedia”. Крім того, від класу “User” походить похідний клас “Admin”, який розширює функціональність користувача та надає адміністративні права взаємодії з мапою, клас адміністратора отримує у розпорядження доступ до полів екземплярів класу “User” через метод “ConfigureUserRules” для можливі адміністрування кожного з користувачів, а також метод “ModUserContent” для модерації екземплярів класу “UserMedia”.

Абстрактний клас “Modules_Hub” має композиційну асоціація з класом “Module”, який представляє модулі, що можуть бути додані до системи методом “GetModule”.

Клас “Module” включає агрегаційні асоціації з класами “Toolbar”, “Layer” та “Feature” та дозволяє включати та налаштовувати кожен з тих класів відповідно викликаючи “Геттери” та “Сеттери” для відповідних класів: “GetToolbar”, “SetToolbar” - для встановлення і модифікації елементів інтерфейсу модуля; “GetLayer”, “SetLayer” - для встановлення і модифікації об’єкта екземпляра класа що створює інтерактивний шар для модуля, поверх якого відображаються інтерфейсні рішення та функціонал; “GetFeature”, “SetFeature” - для взаємодії з екземпляром класу «Feature», що виводить певні обчилення географічних даних для модуля.

Клас “Feature” має двосторонню асоціацію з класом “GeoData”, що дозволяє взаємодіяти з географічними об'єктами на карті через метод “ChangeGeo”.

Кожен клас в системі виконує конкретні функції та має свої власні відповідальності, що сприяє структурованості та модульності системи, в подальшому легкої модифікації.

2.3.2 Діаграми послідовностей

Згенеруємо діаграму послідовності для демонстрації послідовності взаємодії між елементами системи, до прикладу для сценарія авторизації користувача:

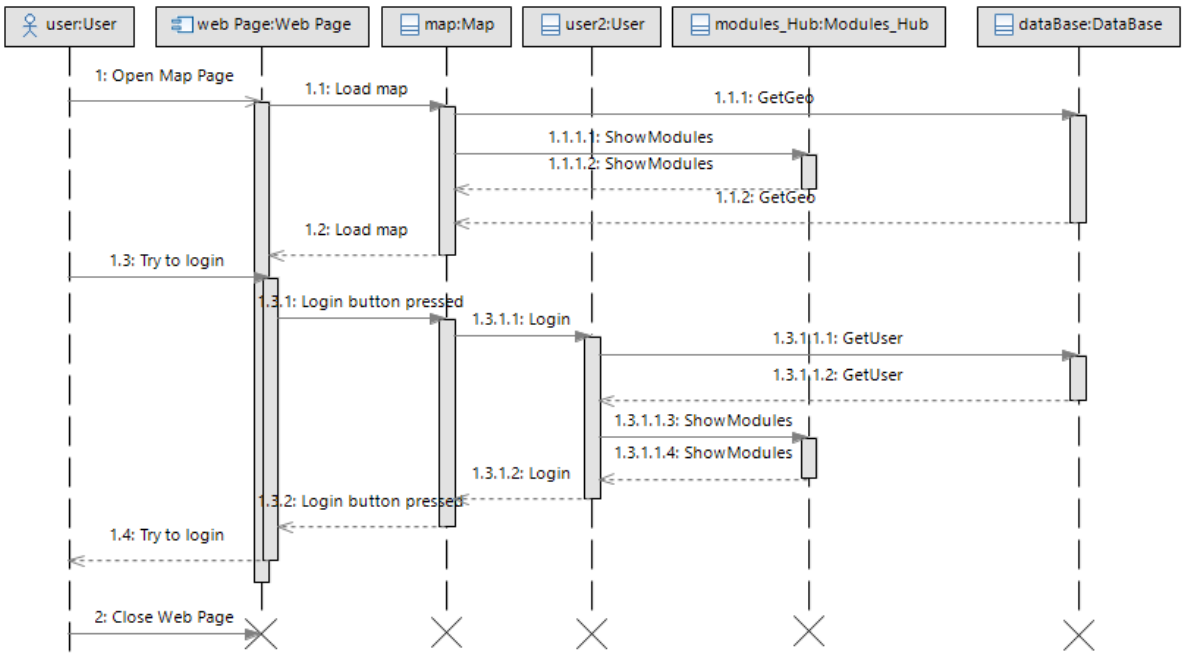

Рисунок 2.6 – Діаграма послідовності для авторизації користувача

Як можна помітити на діаграмі система активно взаємодіє із базою даних для проведення авторизаційних процедур, при цьому при першому заході на веб сторінку, мапа підвантажує модулі для функціональної взаємодії на базовому рівні із мапою, а після виконання авторизації додатково подає запит на підвантаження модулів відповідно до підвищення користувачем рангу.

Розглянемо також детальнішу діаграму послідовності згенеровану для внутрішнього сценарію відображення модуля:

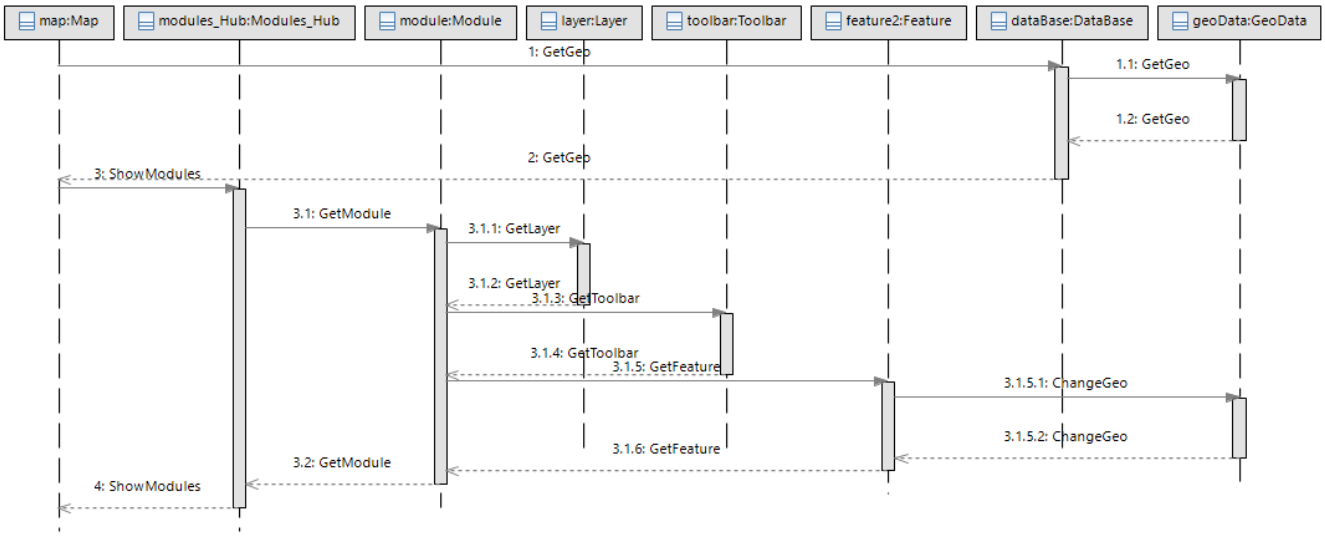

Рисунок 2.7 – Діаграма послідовності для показу модулів

Відразу можна помітити що мапа спершу запитує дані в бази даних, а база даних в свою чергу дістає ці дані та надає мапі, для попередньої генерації, після чого відбувається наступний запит до модульного хабу який у свою чергу дає команду зібрати модуль, після чого йде процес збірки із запитом інформації для кожного елементу, де екземпляр класу “Feature” знову запитується в бази даних, безпосередньо до її даних для подальшої їх модифікації для відображення необхідних обчислень відповідно до зібраного модуля.

3 КОНСТРУЮВАННЯ ІНТЕРАКТИВНОЇ МАПИ

3.1 Реалізація ключових частин коду

Перехід до реалізації ключових елементів коду передбачає розробку набору класів, які легко інтегруються, щоб забезпечити динамічний та керований обмін даними між системою.

Головним класом системи довкола якої будуть збиратись дані з інших класів в єдине ціле це клас “Map”, він взаємодіє з класом “DataBase”, який слугує сховищем для зберігання та пошуку відповідних картографічних даних, а також класом “Modules_Hub”, який інкапсулює різні модулі та функції, що покращують інтерактивність мапи. А також клас “Map” співпрацює з класом “LeafletMap”, який використовує бібліотеку Leaflet для відображення власне інтерфейсу мапи та обробки взаємодії з користувачем.

Завдяки цим взаємопов'язаним класам інтерактивна веб мапа легко поєднує пошук, обробку та візуалізацію даних, створюючи цікавий і зручний картографічний досвід, тому розпочати реалізацію проекту варто із написання його базової структури, для цього відповідно до створених нами у діаграмі ієрархії класів (див. рис. 2.5) напишемо, згідно стандарту написання мовою програмування Java Script, основу та взаємодію між класами, будемо використовувати метод “Import” для включення заголовкових файлів основних елементів системи:

```
import { Component } from 'react';
import LeafletMap from './LeafletMap';
import Modules_Hub from './Modules_Hub';
import DataBase from './DataBase';

export default class Map extends Component{
  render(){
      return (
    <div className='Map'>
        <div className='background'>
            <body>
//class association
            <Modules_Hub/>
            <LeafletMap/>
            <DataBase/>
            </body>
        </div>
    </div>
  );}}
```

Рисунок 3.1 – Код до класу “Map”

У класі явно визначено вкладання функціоналу зі складових класів “LealetMap”, “Modules_Hub” та “DataBase” для формування загальної картини проекту. Вкладаються вони як експортовані дані в “Div” блоки структури HTML для подальшої їх кастомізації, кожен блок “Div” отримує власний “className” атрибут для взаємодії саме із цим елементом у CSS.

Наступним кроком варто перейти до реалізації кожного з класів, для створення основи для подальшої реалізації системи, тож на базовому рівні кожен з класів які будуть інтегровані один в одного будуть мати типову класову структуру, що реалізується у JS:

```
import React, { Component } from 'react'

export default class Layer extends Component {
  render() {
    return (
      <div className='Layer'>
        <h2>Layer</h2>
      </div>
    )
  }
}
```

Рисунок 3.2 – Базова структура кожного класу

Детальніше розглянемо клас “LeafletMap” оскільки він відповідальний за ініціалізацію власне мапи в проекті:

Лістинг 3.1 – Реалізація класу “LeafletMap”

```
//import of main components
import React, { Component, useEffect } from 'react'
import './leaflet.css';
import './App.css';
//import basic Leaflet Components
import {
    MapContainer,
    Popup,
    TileLayer,
    Circle, Polygon, Rectangle
    } from 'react-leaflet';

//Map bounds (with sides)
const boundslim = [
  [48.683, 26.5159],// NW
  [48.725, 26.5159],// NE
  [48.725, 26.3836],// SW
  [48.683, 26.3836] // SE
]
//Map Central point
const      map_center      =      [48.70337856246677,
26.4507881504109 93]; // Координати центру мапи

export default class LeafletMap extends Component {
  render() {
    return (
      <MapContainer
        bounds={boundslim}
        center={map_center}
        zoom={14} minZoom={14}
        style={{width:'100vw', height:'100vh'}}>
        <TileLayer bounds={boundslim}
        url="https://api.maptiler.com/maps/toner-
v2/256/{z}/{x}/{y}.png?key=Plhoe3EFjXOiZpbkM4SK"
        attribution='<a
href="https://www.maptiler.com/copyright/"
target="_blank">&copy;            MapTiler            <a
href="https://t.me/inertiared"      target="_blank">&copy;
```

```
IV:XXXVPM</a>, Contains OS data © Crown copyright and
database right 2023'
            ></TileLayer>
        </MapContainer>
        )
      }
    }
```

У першій частині коду імпортуються необхідні компоненти з бібліотеки React та Leaflet, а також додаткові стилі. Потім визначаються різні компоненти та функції, які дозволяють взаємодіяти з мапою.

Компонент “CoordinatePopup” відповідає за відображення координат покажчика миші на мапі у вигляді спливаючого вікна. Він також форматує координати, щоб вони відображалися у зручному форматі.

Компонент “CompassControl” відповідає за додавання компасу на мапу, що дозволяє визначити орієнтацію. Він інтегрує функціонал компасу з бібліотекою Leaflet.

Клас “LeafletMap” є основним компонентом, який відображає мапу. Він використовує “MapContainer” для відображення мапи з заданими межами, центром та масштабом. В компоненті також використовуються “CoordinatePopup” для відображення координат, “CompassControl” для додавання компасу, а також “TileLayer” для відображення “тайлів” мапи – шматочків з якої мапа утворюється.

Загалом через наведений код у проекті реалізується базовий функціонал взаємодії та відображення мапи, який буде модифіковуватись накладанням функціональних шарів через клас “Modules_Hub”. Їх взаємодія визначатиметься спілкуванням відповідних компонентів класів.

Після написання базової структури коду варто розпочати детальніше налаштовувати кожен компонент, а для цього необхідно спершу розробити інтерфейсну складову, оскільки обчислювальна складова в цій системі є скоріше додатковим функціоналом ніж основним.

3.2 Розробка інтерфейсу користувача

Візуальне оформлення це саме те із чим користувачі будуть найбільше взаємодіяти, а отже необхідний чіткий і збалансований підхід у розробці та реалізації цієї частини системи.

Ще на етапі огляду конкурентів, можна було помітити спільні риси у інтерфейсів, будемо відштовхуватись від них, оскільки інтерфейсні рішення компаній гігантів в плані дизайну не просто дизайнерський хід, це десятки, а то і сотні годин досліджень, порівнянь ефективності та користувацької взаємодії для виведення оптимального рішення.

Тому спершу розробимо концептуальні шаблони для інтерфейсу, щоб мати на що орієнтуватись при втілення рішень у коді.

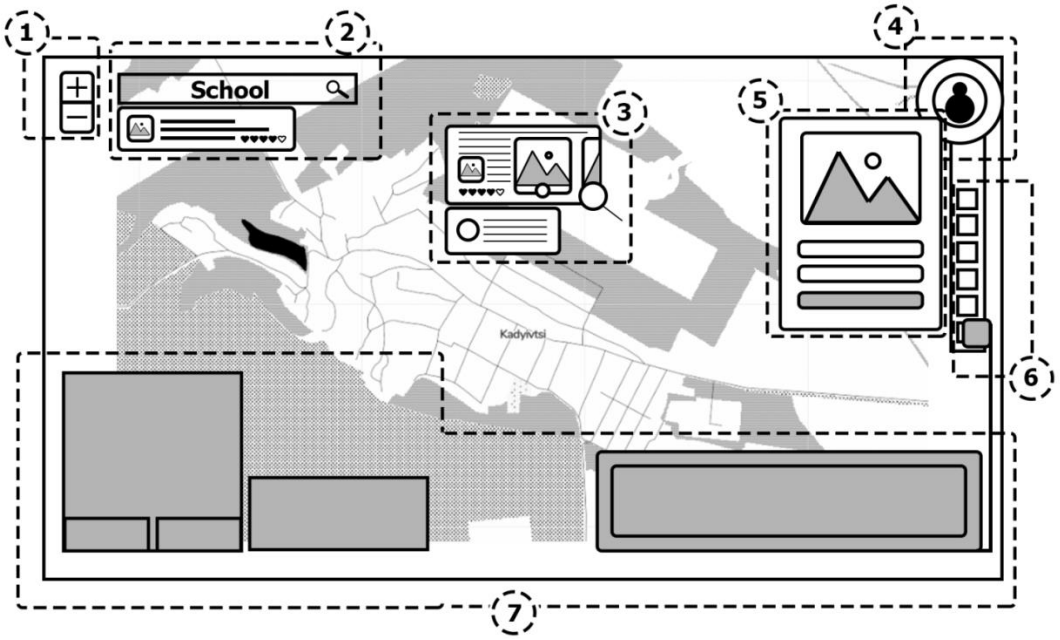

Рисунок 3.3 – Шаблон інтерфейсу

На створеному шаблоні для подальшого втілення інтерфейсу нашої інформаційної системи можна помітити виділені штрихованої лінією зони, це зроблено для чіткішого ознайомлення з реалізацією певної області:

1. Область з інструментом для зміни масштабу;

2. Зона пошуку географічного об’єкта, при пошуку може видавати виринаючи вікно у якому буде зазначено спів падіння з короткою про них інформацією та картинкою і рейтингом;
3. Орієнтовний вигляд інтерфейсу для виділеного географічного об’єкта, що містить у собі комбіновану область відгуків та коментарів, стрічку із медіа контентом користувачів та зоною оцінки і опису об’єкта;
4. Зона з іконкою для авторизації, натискаючи на яку користувач переходить до наступного пункту - авторизації;
5. Виринаюче вікно форма для авторизації користувача, адміністратора;
6. Зона вибору інструментів та модулів, які присутні на мапі;
7. Зона можливого відображення інтерфейсних складових модулів.

Отже розпочнемо реалізацію шаблона в коді, для початку зорієнтуємося, який саме елемент ми будемо розробляти для прикладу, від цього залежатиме який клас має бути задіяний під час реалізації цього елементу, а оскільки в нас за архітектурою функціональні складові реалізовуються системою модулів, то відповідно будемо працювати у класах “Modules_Hub” та “Module”. Оберемо для розробки групу 5, 4 та 6 зображених на малюнку шаблону (див. рис. 3.3), їх можна вмістити в один інтерфейсний блок – що назвемо бічною панеллю. Реалізація коду доступна в додатку А (лістинг 2а).

В результаті чого отримаємо наступну картинку:

Рисунок 3.4 – Сира версія бічної панелі

Через відсутність стилізації наша система не може потішити зручним інтерфейсом, щоб це виправити необхідно стилізувати кожен елемент що ми використовуємо, підігнати до відповідного розміру, накласти правильні відношення між елементами, групування, для цього і є CSS, тому реалізовуємо його, щоб можна було перейти до наступного пункту – результату розробки.

Повну версію коду, та стилізацію інтерактивної веб мапи наведено в додатку А.

3.3 Результати розробки

У результаті плідної праці по аналізу предметної області наявних конкурентів, визначення вимог, генерації діаграм та створенні робочого коду, ми отримуємо нашу робочу систему:

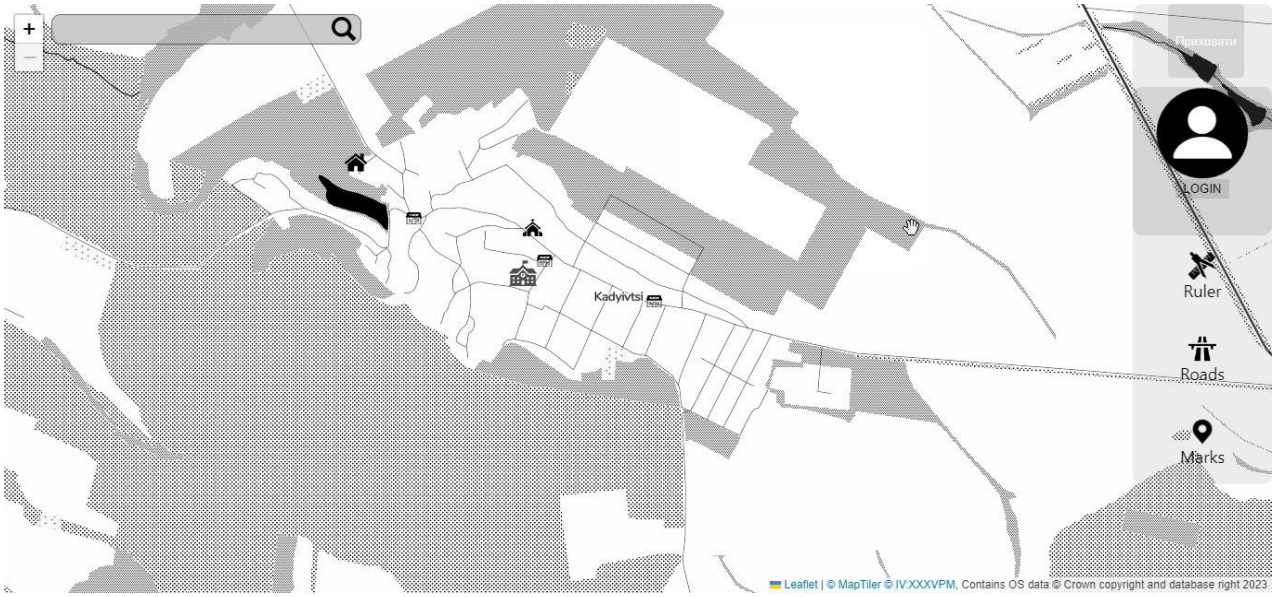

Рисунок 3.5 – Результат розробки

Та в процесі створення продукту готова робоча система не є фіналом роботи на нею, важливо також провести частину із тестуванням та оцінкою ефективності, перевірити чи все коректно працює та відображається.

3.3.1 Тестування системи та оцінка якості

Перейдемо до пункту тестування успішно реалізованих елементів, для початку перевіримо чи працює функціонал масштабування та разом із ним модуль для відображення додаткової інформації про відмічений географічний об’єкт. А для цього наведемо курсором мишки на обраний нами географічний об’єкт та натиснемо на нього. У відповідь система має показати змінити масштаб та фокус на виділений нами об’єкт, після чого показати віконце із інформацією про нього:

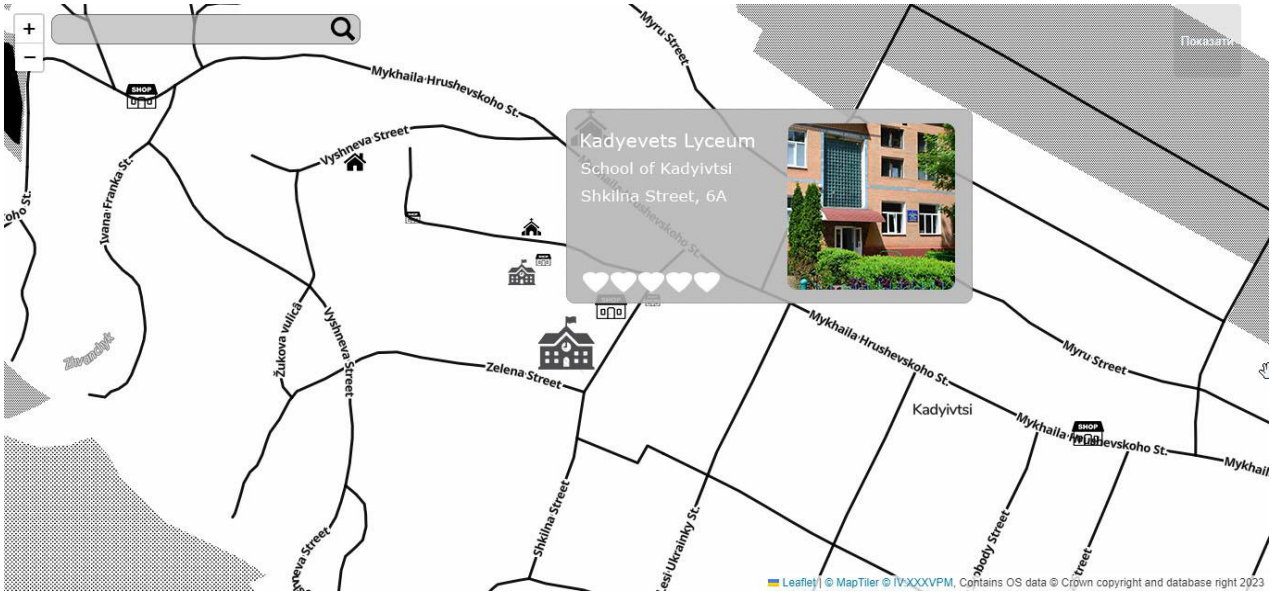

Рисунок 3.6 – Тестування відображення інформації про географічний об’єкт

Як можна помітити на зображенні система успішно справилась із цією задачею.

Наступним тестовим «кейсом» буде перевірка функціонування модулю із пошуку відповідного географічного об’єкта на мапі, разом із тим можна буде перевірити чи добре взаємодіють класи “Map” та “DataBase”. Для цього необхідно натиснути на відповідну область на мапі, що має стрічку для вводу запиту та кнопку у вигляді збільшую чого скельця, після чого вводимо наш запит:

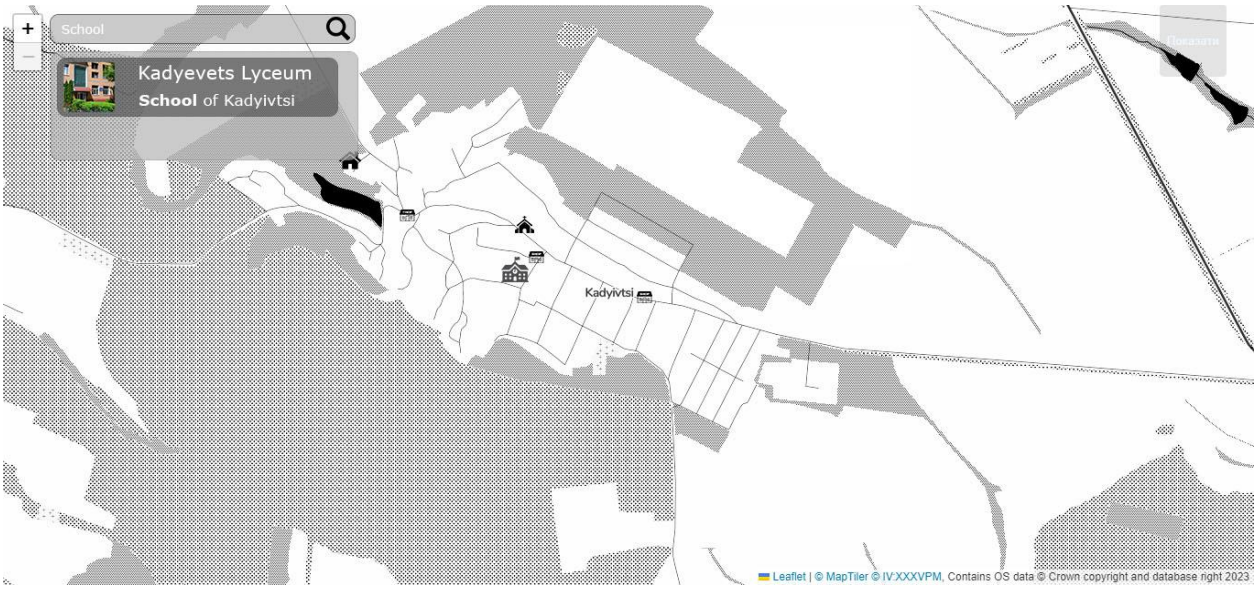

Рисунок 3.7 – Тестування пошуку на мапі

У результаті тестування, при введенні нашого запиту, система одразу підказує про наявність відповідного географічного об’єкта на мапі у випадаючому списку об’єктів пошуку, а відповідно пошук виконує свій функціонал.

Тестування підтвердило, що мапа успішно отримує дані з бази даних, інтегрує різні модулі та відображає інтерфейс карти за допомогою бібліотеки Leaflet. Загалом, етап тестування підтвердив функціональність та зручність інтерактивної веб мапи населеного пункту.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Моделювання та прогнозування небезпечних ситуацій

При вирішенні питання моделювання та прогнозування небезпечних ситуацій мною було проведено ряд досліджень створюваної системи, щоб виявити потенційні небезпеки, змоделювати рішення, які необхідно інтегрувати у систему щоб забезпечити максимально можливе уникнення небезпечних ситуацій.

Хід дослідження було поділено на відповідні кроки, що містять детальну інформацію щодо вирішеного питання.

Перш за все необхідно провести дослідження самої системи [9], для цього створено стислий опис системи:

Створюваний програмний продукт є інтерактивною веб мапою, що уявляє собою онлайн картографічний інструмент для орієнтування користувачів у просторі, а також надання відповідних довідкових даних що до певних географічних об’єктів. Ключовими елементами є саме використання зображень мапи, які в відносній точності відображають існуючі географічні елементи. До інтерфейсу користувача входять блоки інструментів для орієнтування, такі як компас, а також визначення його геолокації в реальному часі. Збір даних відбувається за допомогою взаємодії користувачів між собою, та підтвердженням даних відповідальними особами – адміністраторами веб мапи, а також додатково із наявних актуальних джерел географічної інформації, мережі інтернет.

Наступним кроком є розробка моделі для аналізу потенційних небезпек:

Аналіз небезпек починається із загального дослідження, яке дозволяє виявити основні джерела ризику. За потреби, дослідження можуть бути узагальнені та проведений детальний якісний аналіз. Ці аналізи використовують різні методи та прийоми, які відомі під різними назвами. Нижче перераховані основні з цих загальних інструментів [10].

Типи аналізу:

- попередній аналіз небезпек (ПАН)

- системний аналіз небезпек (САН)
- підсистемний аналіз небезпек (ПСАН)
- аналіз небезпеки робіт та обслуговування (АНРО).

Методи та прийоми, що використовуються при аналізах:

- аналіз пошкоджень та викликаного ними ефекту (АПВЕ)
- аналіз дерева помилок (АДП)
- аналіз ризику помилок (АРП)
- прорахунки менеджменту та дерево ризику (ПМДР)
- аналіз потоків та перешкод енергії (АППЕ)
- аналіз поетапного наближення (АПН)
- програмний аналіз небезпек (ПрАН)
- аналіз загальних причин поломки (АЗПП)
- причинно-наслідковий аналіз (ПНА)
- аналіз дерева подій (АДП)

Попередній аналіз небезпек (ПАН) – це аналіз загальних груп небезпек, присутніх в системі, їх розвитку та рекомендації щодо контролю. ПАН є першою спробою в процесі безпеки систем визначити та класифікувати небезпеки, які мають місце в системі. Проте в багатьох випадках цьому аналізу може передувати підготовка попереднього переліку небезпек, який відображає можливі загрози і потенційні ризики для системи. Цей попередній аналіз небезпек допомагає зрозуміти основні вектори небезпек та встановити початкові заходи контролю, які необхідно вжити.

ПАН звичайно виконується у такому порядку:

- вивчають технічні характеристики об'єкта, системи чи процесу, а також джерела енергії, що використовуються, робоче середовище, матеріали; встановлюють їхні небезпечні та шкідливі властивості;
- визначають закони, стандарти, правила, дія яких розповсюджується на даний об'єкт, систему чи процес;

- перевіряють технічну документацію на її відповідність законам, правилам, принципам і нормам безпеки;
- складають перелік небезпек, в якому зазначають ідентифіковані джерела небезпек (системи, підсистеми, компоненти), чинники, що викликають шкоду, потенційні небезпечні ситуації, виявлені недоліки.

При проведенні ПАН особливу увагу приділяють наявності вибухо-пожежонебезпечних та токсичних речовин, виявленню компонентів об'єкта, в яких можлива їх присутність, потенційна небезпечна ситуація від неконтрольованих реакцій чи при перевищенні тиску. Після того, коли виявлені крупні системи об'єкта, які є джерелами небезпеки, їх можна розглядати окремо і досліджувати більш детально за допомогою інших методів аналізу, перелік яких наведено вище. Існують базові запитання, на які обов'язково необхідно відповісти, коли проводять ПАН, незважаючи на те, що деякі з них можуть здаватися занадто простими, якщо ці запитання не розглянути, то існує ризик неповного аналізу безпеки системи. Вся простота чи очевидність має схильність приховувати деякий рівень прихованої небезпеки.

Проведення ПАН може бути спрощено і формалізовано завдяки використанню визначенню попередніх небезпечних ситуацій.

У процесі користування на користувача покладається основна маса вирішення питань уникнення небезпечних ситуації, таких як відвідання потенційно небезпечних географічних об’єктів, як то до прикладу круті схили, місця зсуву ґрунту, які практично неможливо відображати у статичних елементах веб мапи, оскільки такі дані потребують постійної актуалізації відповідним персоналом і не можуть бути доцільно автоматизованими.

Проте в процесі розробки було вирішено інтегрувати модуль сповіщення про потенційну небезпеку від природніх та техногенних процесів. У відповідності щодо природніх небезпек користувач буде отримувати сповіщення про: посилення швидкості вітру, насування грози чи туману, дані про який будуть надходити із перевірених метеодослідницьких станцій, а також сповіщення про катастрофічні природні явища від державних служб надзвичайних ситуацій.

Щодо техногенних небезпек, було інтегровано модуль сповіщення про загрозу ракетних ударів дані яких беруться із локальних систем сповіщення населення.

У ході роботи постійно необхідно актуалізовувати дані для цього потрібно проводити збір та аналіз даних [11].

Збір та аналіз даних відбувається за допомогою соціальних медіа, а також джерел метеорологічних прогнозувань і відповідних органів що відповідають за сповіщення про надзвичайні ситуації. За процеси аналізу даних відповідає внутрішній механізм відповідного модуля, а рішення про сповіщення на інтерактивній веб мапі приймається автоматично, але може бути скасоване відповідальною за адміністрацію мапи особою.

Для кращого розуміння системи сповіщень було створено стислу візуалізацію сповіщень.

Відповідно до ступеня загрози сповіщення що відображаються у середовищі інтерактивної веб мапи мають наступні категорії:

1) Повідомлення застереження – використовується для сповіщення про незначні погодні негаразди, ускладнення у вигляді туману, ожеледі чи тому подібних погодних явищ.
2) Повідомлення небезпеки – використовується при необхідності сповістити, що наступні природні чи техногенні явища, що відбуватимуться у певній географічній області несуть потенційно небезпечний характер.
3) Екстрені сповіщення – використовуються тільки при виникненні ситуацій максимально небезпечного характеру, як то руйнівні землетруси, повені, лісні пожежі, загрози ракетних та терористичних атак.

Для тестування системи сповіщень було використано моделювання потенційно небезпечних ситуацій за допомогою активації модулів сповіщення, надходженням на них відповідних повідомлень про певні загрози. У ході тестування було виявлені деякі неточності спрацювання, та було усунуто їх

налаштуванням чутливості [12] щодо сповіщень, а також додатково інтегровано в модуль звернення до адміністратора значень про небезпечні ділянки мапи.

Після тестування та корекції відповіді модуля, було досягнуто необхідної точності реагування на потенційно небезпечні ситуації і виклику сповіщення для користувачів, для уникнення цих небезпечних ситуацій.

Висновки щодо проведеної роботи:

У ході роботи над питанням було змодельовано і створено систему для прогнозування небезпечних ситуацій, яку відповідно було інтегровано у проєкт. Під час тестування система показала хороші результати відгуку на потенційні загрози, а після виконання робіт по налаштуванню чутливості модулів – результати відгуку на загрози набули необхідних значень для успішного сповіщення користувачів про загрози.

Питання затримок сповіщення для створюваної інформаційної системи неможливо усунути в повній мірі, оскільки затримка надходження відповідних даних існує у джерел фіксації загрози [12]. Для можливого покращення і зменшення затримок перед сповіщеннями, необхідно створювати системи що будуть з кращою продуктивністю і якістю фіксувати потенційні природні та техногенні явища небезпеки, а дані від цих систем будуть доступні для розробників у повній мірі і точності.

4.2 Вимоги до виробничого освітлення та його нормування в офісному приміщенні.

У відповідності до статті 4, розділу І про загальні положення в законі України про охорону праці для робочих місць встановлюються єдині вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності [13], тому вирішувати питання вимог до виробничого освітлення та його нормування варто спираючись на

затверджені вимоги для нормування природного освітлення, яким являється коефіцієнт природного освітлення (КПО) згідно ДБН В 2.5-28:2018 [14].

КПО встановлюється в залежності від розряду виконуваних зорових робіт. Робота оператора ПК відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5 – 1,0мм), для яких при використанні бокового освітлення КПО = 1,5%.

Для штучного освітлення нормованим параметром виступає $E_{мін}$ – мінімальний рівень освітленості, та $К_п$ – коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%.

Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряда зорових робіт вона складає 300-500 лк.

Перевіримо освітленість робочого місця користувача ПК у середньостатистичному офісному приміщенні на відповідність розряду зорової роботи. За даними вимірювань рівень природної освітленості поверхні, де розташований ПК, в середньому складає 200 лк при освітленості тієї же поверхні відкритим небосхилом в 20000 лк, тобто КПО = 1%, що не відповідає нормативному КПО.

Для штучного освітлення у приміщенні використовуються будуть використовуватись люмінесцентні лампи, які в порівнянні з лампами розжарювання мають ряд істотних переваг:

- за спектральним складом світла вони близькі до природного світла;
- мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання);
- мають триваліший термін служби (до 10 тис. часів).

Розрахунок штучного освітлення проведемо для кімнати площею 20 $м^2$, ширина якої складає 5 м, довжина – 4 м, висота – 3 м. Скористаємося методом використання світлового потоку [15].

Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \text{де}$$

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк*;

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5*, прирівнюючи офіс до приміщення громадських і житлових будівель);

S – площа освітлюваного приміщення (у нашому випадку $S = 20\ м^2$);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1,1 – 1,2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (виражається відношенням світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп, і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін}$) і стелі ($\rho_{стелі}$)), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 40% і $\rho_{стелі}$ = 60%.

Обчислимо індекс приміщення за формулою:

$$I = \frac{S}{h(A + B)}, \text{де}$$

S – площа приміщення, $S = 20\ м^2$;

h – розрахункова висота підвісу, *h* = 2,9 м;

A – ширина приміщення, *A* = 4 м;

B – довжина приміщення, *B* = 5 м.

Підставивши значення отримаємо:

$$I = \frac{20}{2{,}9 \cdot (4 + 5)} = 0{,}77$$

Знаючи індекс приміщення *I* за таблицею 4 [ДБН В.2.5-28:2018] у нашому випадку знаходимо значення n = 0,22. Підставимо всі значення у формулу для визначення світлового потоку *F*:

$$F = \frac{300 \cdot 1{,}5 \cdot 20 \cdot 1{,}1}{0{,}22} = 45000 \text{ Лм}$$

Для освітлення використані люмінесцентні лампи типу ЛБ 40-1, світловий потік яких *F* = 4320 Лм.

Розрахуємо необхідну кількість ламп у світильниках за формулою:

$$N = \frac{F}{F\text{л}} \text{, де}$$

N – необхідне число ламп;

F – світловий потік, *F* = 45000 Лм;

*F*л – світловий потік лампи, *F*л = 4320 Лм.

Тоді отримаємо відповідний результат:

$$N = \frac{45000}{4320} = 11$$

В приміщенні використовуються світильники типу ОД. Кожен світильник комплектується двома лампами. Тобто необхідно використовувати 6 світильників із 12 працюючими лампами в них відповідно створено схему розташування цих світильників в офісному приміщенні:

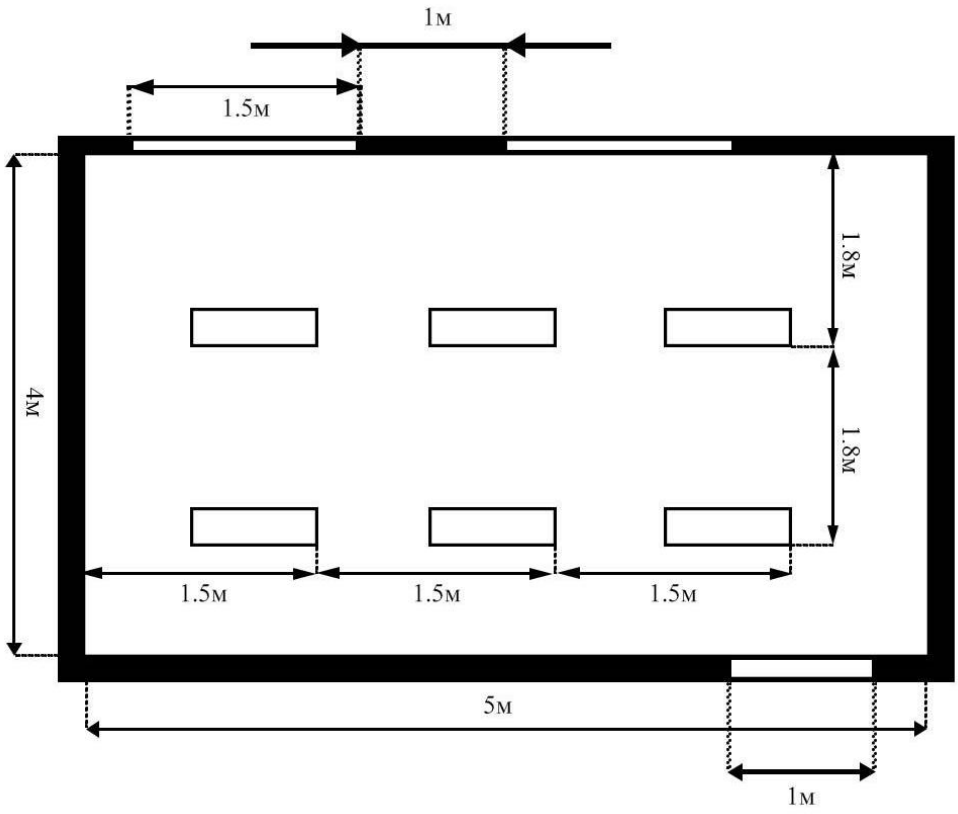

Рисунок 4.1 – Схема розташування світильників

Використовуючи наведену схему можна досягнути необхідних світлових умов для безпечної роботи з персональними комп’ютерами в офісному приміщенні, що має площину 20 м^2, є прямокутної форми із внутрішніми сторонами 4 м та 5 м відповідно.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра на тему "Розробка інтерактивної мапи населеного пункту з використанням технологій Leaflet та React" було здійснено ретельний аналіз предметної області. Були вивчені конкуренти, їх переваги та недоліки, що сприяло ідентифікації ключових аспектів для успішної розробки інтерактивної мапи. В процесі роботи була проведена значна кількість досліджень у галузі картографії з метою визначення найбільш ефективних інструментів для розробки подібних систем та що ці системи мають містити у собі.

Далі було розроблено модель системи, яка включала діаграми використання, класів і послідовностей. Цей крок дозволив уявити та чітко визначити функціональність та взаємодію компонентів системи перед самим процесом реалізації.

Також під у ході розробки була розглянута бізнес-модель проєкту. Проєктування бізнес-моделі врахувало потенційних користувачів, можливі шляхи отримання монетизації проєкту. Це допомогло створити фундаментальний план для комерціалізації розробленої системи.

Останнім ж етапом була реалізація в коді та тестування інтерактивної мапи. У результаті розробки була успішно реалізована система, яка забезпечує зручну взаємодію з користувачем та надійно відображає інформацію про населені пункти. Під час тестування було перевірено правильність роботу системи та її відповідність вимогам проєкту.

В результаті успішного виконання цієї кваліфікаційної роботи бакалавра була розроблена та протестована інтерактивна мапа населеного пункту з використанням технологій Leaflet та React, яка цілком реалізовує поставлені до роботи завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Maps Platform Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation
2. Bing Map Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/bingmaps/
3. JavaScript Підручник. Основи веб-програмування [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/js/
4. Кравців С.С., Войтків П.С., Кобелька М.В. Картографія: навч. посіб. Львів: ЛНУ імені Івана Франка, 2020. 191 с.
5. Leaflet Documentation [Електронний ресурс] – Режим доступу до ресурсу https://leafletjs.com/reference.html
6. React Learn [Електронний ресурс] – Режим доступу до ресурсу https://react.dev/learn
7. Крижановський Є. М., Ящолт А.Р., Жуков С.О., Козачко О.М. Моделювання бізнес-процесів та управління ІТ-проектами: електронний навч. посіб. Вінниця: ВНТУ, 2018. 91 с.
8. Essentials of modeling with Rational Software Architect Designer - Self-paced training [Електронний ресурс] – Режим доступу до ресурсу https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=overview-essentials-modeling-rational-software-architect-designer-self-paced-training
9. Желібо Є.П., Зацарний В.В. Безпека життєдіяльності. Підручник. – К.: Каравела, 2006. – 288 с.
10. Коростіль Ю.М. Метод побудови моделей для непроектних несправностей./ Збірник наукових праць. Інститут проблем моделювання в енергетиці. Вип.. 33, Україна, Київ, 2006.
11. Давиденко А.М., Попенко А.С., Сторчак А.С. Візуалізація елементів моделі загроз за допомогою COM – сервера./ Моделювання та інформаційні технології. ІПМЕ НАН України, вип.. 46, 2008.

12. Давиденко А.М., Головань С.М., Щербак Т.Л. Аналіз дій загроз у автоматизованих системах обробки інформації./ Моделювання та інформаційні технології. ІПМЕ НАН України, вип.. 37, 2006.
13. Закон України «Про охорону праці». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/2694-12
14. ДБН В.2.5-28:2018. «Природне і штучне освітлення» – К.: Мінрегіон України, 2018. 133 с.
15. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці. Підручник – К.: Каравела, 2007. 408 с.

ДОДАТКИ

ДОДАТОК А

Лістинг коду розробленої системи

Лістинг 1а – Файл Map.js

```
// Importing necessary modules and components
import { Component } from 'react';
import LeafletMap from './LeafletMap';
import Modules_Hub from './Modules_Hub';
import DataBase from './DataBase';
import { useState } from 'react';

// Component for a hideable element
const HideableElement = () => {
  // Using the state hook to manage visibility
  const [isVisible, setIsVisible] = useState(true);

  // Function to toggle the visibility
  const toggleVisibility = () => {
    setIsVisible(!isVisible);
  };

  // Rendering the hideable element
  return (
    <div>
      {/* Button to toggle visibility */}
      <button id="HideButton" onClick={toggleVisibility}>
        {isVisible ? 'Приховати' : 'Показати'}
      </button>

      {/* Conditionally rendering the Modules_Hub component based on visibility */}
      {isVisible && <div><Modules_Hub/></div>}
    </div>
  );
};
```

```
// Exporting the Map component
export default class Map extends Component {
  render() {
    return (
      <div className='Map'>
        <div className='background'>
          <body>
            {/* Rendering components */}
            <HideableElement/>
            <LeafletMap/>
            <DataBase/>
            <Modules_Hub/>
          </body>
        </div>
      </div>
    );
  }
}
```

Лістинг 2а – Файл Modules_Hub.js

```
// Importing necessary modules and components
import React, { Component } from 'react';
import { useState } from 'react';
import 'boxicons';
import profile from './images/profile.png';
import ruler from './images/ruler.png';
import road from './images/road.png';
import marks from './images/marker-icon.png';
import Module from './Module';

// Component for a hideable element
const HideableElement = () => {
    const [isVisible, setIsVisible] = useState(true);

    // Function to toggle the visibility
    const toggleVisibility = () => {
        setIsVisible(!isVisible);
    };
```

```
        return (
            <div>
                {/* Button to toggle visibility */}
                <button id="HideButton2" onClick={toggleVisibility}>
                    {isVisible ? 'Приховати' : 'Показати'}
                </button>

                {/* Conditionally rendering the Module component based on
visibility */}
                {isVisible && <div><Module></Module></div>}

                {/* Another hideable element */}
                <div>
                    <button id="HideButton3" onClick={toggleVisibility}>
                        {isVisible ? 'Приховати' : 'Показати'}
                    </button>
                    {isVisible && <div><Module></Module></div>}
                </div>
            </div>
        );
    };

    // Exporting the Modules_Hub component
    export default class Modules_Hub extends Component {
        render() {
            return (
                <div className='sidebar'>
                    <ul className='nav-links'>
                        {/* List item for hiding/showing */}
                        <li id='hide'>
                            <span className='link_name' id='hide'>
                                <span></span>
                            </span>
                        </li>
                        <li>
                            {/* Profile section */}
                            <span className='link_name' id='profile'>
                                <img src={profile} alt='nothing:('></img>
```

```
                            <button id='Login'>LOGIN</button>
                        </span>
                    </li>
                    <li>
                        {/* Ruler section */}
                        <span className='link_name'>
                            <img src={ruler} alt='Ruler'></img>
                            <span id='obj'>Ruler</span>
                        </span>
                    </li>
                    <li>
                        {/* Roads section */}
                        <span className='link_name'>
                            <img src={road} alt='Road'></img>
                            <span id='obj'>Roads</span>
                        </span>
                    </li>
                    <li>
                        {/* Marks section */}
                        <span className='link_name'>
                            <img src={marks} alt='Marks'></img>
                            <span id='obj'>Marks</span>
                        </span>
                    </li>
                </ul>
            </div>
        );
    }
}
```

Лістинг 3а – Файл LeafletMap.js

```
// Importing necessary modules and components
import React, { Component } from 'react';
import './leaflet.css';
import './App.css';
import Modules_Hub from './Modules_Hub';
import {
    LayersControl,
    MapContainer,
    Marker,
    Popup,
    TileLayer,
```

```
        useMap,
        useMapEvents,
        useMapEvent,
        Circle, Polygon, Rectangle
    } from 'react-leaflet';

    // Component to display a popup with coordinates of the pointer
    const CoordinatePopup = () => {
        const [position, setPosition] = React.useState(null);

        useMapEvent('mousedown', (e) => {
            if (e.originalEvent.button === 1) {
                setPosition(e.latlng);
            }
        });

        // Format coordinates of the pointer
        const formatCoordinate = (coordinate) => {
            return                          `[${coordinate.lat.toFixed(9)},
${coordinate.lng.toFixed(9)}]`;
        };

        // Handle center click event
        const handleCenterClick = (event) => {
            if (event.target.className === 'center-button') {
                useMap.setView = { map_center };
            }
        };

        return (
            <>
                {/* Display the popup with coordinates */}
                {position && (
                    <Popup position={position}>
                        <span>{`${position.lat.toFixed(9)},
${position.lng.toFixed(9)}`}</span>
                    </Popup>
                )}
            </>
        );
    };

    // Map bounds
    const boundslim = [
        [48.683, 26.5159], // Top left corner (northwest) NW
        [48.725, 26.5159], // Top right corner (northeast) NE
        [48.725, 26.3836], // Bottom right corner (southeast) SE
        [48.683, 26.3836] // Bottom left corner (southwest) SW
    ];

    // Map central point
    const  map_center  =  [48.70337856246677,  26.450788150410993];  //
Coordinates of the map center

    export default class LeafletMap extends Component {
        render() {
            return (
                <MapContainer
```

```
                    bounds={boundslim}
                    center={map_center}
                    zoom={14}
                    minZoom={14}
                    style={{ width: '100vw', height: '100vh' }}
                >
                    {/* Render the CoordinatePopup component */}
                    <CoordinatePopup />

                    {/* Render the TileLayer with the map tiles */}
                    <TileLayer
                        bounds={boundslim}
                        url="https://api.maptiler.com/maps/toner-
v2/256/{z}/{x}/{y}.png?key=Plhoe3EFjXOiZpbkM4SK"
                        attribution='<a
href="https://www.maptiler.com/copyright/" target="_blank">&copy; MapTiler
<a href="https://t.me/inertiared" target="_blank">&copy; IV:XXXVPM</a>,
Contains OS data © Crown copyright and database right 2023'
                    ></TileLayer>
                </MapContainer>
            );
        }
    }
```

Лістинг 4а – Файл DataBase.js

```
// Importing necessary modules and components
import React, { Component } from 'react';
import GeoData from './GeoData';
import User from './User';

export default class DataBase extends Component {
    render() {
        return (
            // Render the GeoData and User components
            <>
                <GeoData />
                <User />
            </>
        );
    }
}
```

Лістинг 5а – Файл GeoData.js

```
// File: GeoData.js
import React, { Component } from 'react';
```

```
class DataBase {
    constructor() {
        this.geoData = new GeoData();
    }

    fetchData() {
        // Retrieve the actual map data from the database or an external source

        // Assume the fetched data is an object containing information about geographic objects
        const data = {
            locations: [
                { name: { name }, lat: { lat }, lng: { lng } },
            ],
        };

        this.geoData.update(data.locations);
    }

    getMapData() {
        return this.geoData.getData();
    }
}

class GeoData {
    constructor() {
        this.data = [];
    }

    update(locations) {
        // Update the map data with the provided locations
        this.data = locations;
    }

    getData() {
        return this.data;
    }
```

```
}

// Example usage

const database = new DataBase();

// Fetch actual map data
database.fetchData();

// Get the map data
const mapData = database.getMapData();
console.log(mapData);
```

Лістинг 6а – Файл User.js

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

// Define the Login component
const Login = () => {
  // State variables for email, password, and error
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  // Event handler for email input change
  const handleEmailChange = (event) => {
    setEmail(event.target.value);
  };

  // Event handler for password input change
  const handlePasswordChange = (event) => {
    setPassword(event.target.value);
  };

  // Event handler for form submission
  const handleSubmit = (event) => {
    event.preventDefault();

    // Check entered data and perform authentication
```

```
      if (email === 'example@example.com' && password === 'password') {
        // Successful authentication
        setError('');
        console.log('Успішна авторизація');
        // Additional code for redirecting to another page or
performing other actions after authentication
      } else {
        // Invalid email or password
        setError('Невірний email або пароль');
      }
    };

    // Render the Login component
    return (
      <div>
        <h2>Вхід</h2>
        {error && <p>{error}</p>}
        <form onSubmit={handleSubmit}>
          <div>
            <label htmlFor="email">Email:</label>
            <input
              type="email"
              id="email"
              value={email}
              onChange={handleEmailChange}
              required
            />
          </div>
          <div>
            <label htmlFor="password">Пароль:</label>
            <input
              type="password"
              id="password"
              value={password}
              onChange={handlePasswordChange}
              required
            />
          </div>
          <button type="submit">Увійти</button>
```

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

export default Login;
```

Лістинг 7а – Файл UserMedia.js

```
class UserMedia {
  constructor() {
    this.images = [];
  }

  addImage(image) {
    this.images.push(image);
  }

  removeImage(index) {
    if (index >= 0 && index < this.images.length) {
      this.images.splice(index, 1);
    }
  }

  getImages() {
    return this.images;
  }
}

// Create an instance of UserMedia
const userMedia = new UserMedia();

// Add images
const image = { id: 1, url: {url} };
userMedia.addImage(image);

// Remove an image by index
userMedia.removeImage(0);
```

```
// Get all images
const images = userMedia.getImages();
console.log(images);
```

Лістинг 8а – Файл Feature.js

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

export default class Feature extends Component {
  render() {
    return (
      <div className='Feature'>
        const MapInfo = ({ location }) => {
          return (
            <div>
              <h3>{location.name}</h3>
              <p>Координати: {location.lat}, {location.lng}</p>
              <p>Тип: {location.type}</p>
              {/*  Place  for  extend  other  data,  or  make  it  more
tolerance */}
            </div>
          );
        };
        export default MapInfo;
      </div>
    );
  }
}
```

Лістинг 9а – Файл Layer.js

```
import React, { Component } from 'react';
import Leaflet from Leaflet;
import {
  LayersControl,
} from 'react-leaflet';

export default class Layer extends Component {
  render() {
    return (
      <div className='Layer'>
        // Maplayers manipulation
        class MapLayer {
```

```
            constructor(map) {
              this.map = map;
              this.layers = [];
            }

            addLayer(layer) {
              this.layers.push(layer);
              this.map.addLayer(layer);
            }

            removeLayer(layer) {
              const index = this.layers.indexOf(layer);
              if (index !== -1) {
                this.layers.splice(index, 1);
                this.map.removeLayer(layer);
              }
            }

            removeAllLayers() {
              this.layers.forEach((layer) => {
                this.map.removeLayer(layer);
              });
              this.layers = [];
            }
          }
        </div>
      );
    }
  }
```

Лістинг 10а – Файл Toolbar.js

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

export default class Toolbar extends Component {
  constructor(props) {
    super(props);
    // Initialize the component state
    this.state = {
      searchQuery: '',      // Stores the search query entered by the
user
      searchResults: [],    // Stores the results of the search
      selectedObject: null, // Stores the currently selected object
    };
  }

  // Event handler for search input change
  handleSearchInputChange = (event) => {
    this.setState({ searchQuery: event.target.value });
  };

  // Event handler for search form submission
  handleSearchSubmit = (event) => {
    event.preventDefault();
    const { searchQuery } = this.state;
    // Perform the search using the GeoData module
```

```
        const searchResults = GeoData.search(searchQuery);
        this.setState({ searchResults, selectedObject: null });
      };

      // Event handler for clicking on an object
      handleObjectClick = (object) => {
        this.setState({ selectedObject: object });
      };

      render() {
        const { searchQuery, searchResults, selectedObject } =
this.state;

        return (
          <div className='Toolbar'>
            <div>
              {/* Search Form */}
              <form onSubmit={this.handleSearchSubmit}>
                <input
                  type="text"
                  value={searchQuery}
                  onChange={this.handleSearchInputChange}
                  placeholder=""
                />
                <button type="submit">Search</button>
              </form>
              <div>
                {/* Display search results */}
                {searchResults.map((result) => (
                  <div       key={result.name}       onClick={()       =>
this.handleObjectClick(result)}>
                    <h3>{result.name}</h3>
                    <p>Широта: {result.lat}</p>
                    <p>Довгота: {result.lng}</p>
                  </div>
                ))}
              </div>
            </div>
            <div>
              {/* Display selected object details */}
              {selectedObject && (
                <div>
                  <h3>{selectedObject.name}</h3>
                  <p>Широта: {selectedObject.lat}</p>
                  <p>Довгота: {selectedObject.lng}</p>
                  {/* Display additional object data */}
                </div>
              )}
            </div>
          </div>
        );
      }
    }
```

ДОДАТОК Б

Диск з розробленою програмою

ДОДАТОК В

Розширена діаграма варіантів використання для актора «Користувач»

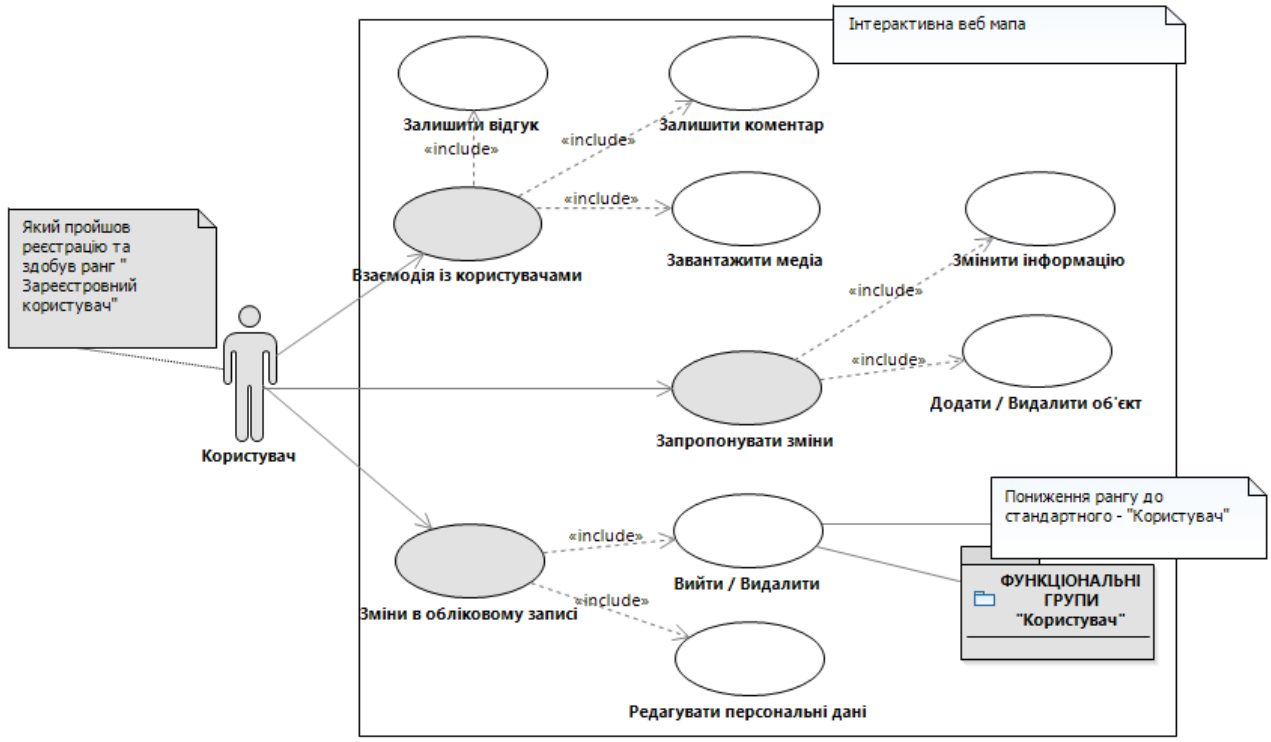

Рисунок В.1 – Розширення варіантів використання для актора «Користувач»

Опис варіантів використання:

Варіант використання «Взаємодія із користувачами»

Короткий опис:

Користувач отримує доступ до функціоналу для взаємодії із іншими зареєстрованими користувачами.

Основний потік подій:

1. Користувач знаходить географічний об’єкт;
2. Користувач обирає цей об’єкт;
3. Користувач отримує доступ до меню взаємодії;

Альтернативний потік подій:

1. Користувачеві недоступне меню взаємодій;
2. Система видає причини.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач».

Післяумови:

Користувач успішно завершує роботу із цим інструментом, дані зберігаються.

Варіант використання «Залишити відгук»

Короткий опис:

Користувач має можливість залишити відгук про географічний об’єкт.

Основний потік подій:

1. Користувач у вибраному об’єкті знаходить пункт залишити відгук.
2. Користувач пише відгук;
3. Користувач виставляє оцінку об’єкту;
4. Користувач натискає кнопку «Опублікувати».

Альтернативний потік подій:

1. Користувач вводить недопустимі значення;
2. Відгук неможливо опублікувати;
3. Система видає причини.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», користувач обрав відповідний географічний об’єкт.

Післяумови:

Система зберігає внесений відгук.

Варіант використання «Залишити коментар»

Короткий опис:

Користувачеві доступний інструмент щоб залишити коментар на відгук іншого користувача.

Основний потік подій:

1. Користувач знаходить географічний об’єкт на якому є відгуки користувачів;

2. Користувач обирає відгук до якого хоче написати коментар;
3. Користувач натискає на кнопку «Коментувати»;
4. Користувач пише текст коментару та за бажанням прикріпляє медіа контент;
5. Користувач натискає кнопку «Коментувати».

Альтернативний потік подій:

1. Користувач не може прокоментував відгук;
2. Система видає причини.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», на мапі доступний хоча б один об’єкт, у якого є відгук.

Післяумови:

Система зберігає дані про коментар.

Варіант використання «Завантажити медіа»

Короткий опис:

Користувач може завантажувати медіа контент.

Основний потік подій:

1. Користувач знаходить на мапі географічний об’єкт до якого хоче додати медіа контент;
2. Користувач обирає пункт «Завантажити медіа»;
3. Користувач обирає медіа контент;
4. Користувач натискає «Опублікувати».

Альтернативний потік подій:

1. Користувач не може завантажувати медіа контент;
2. Система видає причину.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», на мапі є хоча б один географічний об’єкт до якого можна додати медіа контент.

Післяумови:

Система зберігає дані.

Варіант використання «Запропонувати зміни»

Короткий опис:

Користувачеві доступний функціонал для внесення зміни до даних об’єкта.

Основний потік подій:

1. Користувач знаходить відповідне меню інструментів - «Запропонувати зміни»;
2. Користувач відкриває це меню натиснувши на кнопку;

Альтернативний потік подій:

1. Користувачеві недоступне внесення змін;
2. Система видає причини.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», мапа відображається коректно.

Післяумови:

Користувачеві доступні наступні варіанти використання «Змінити інформацію» та «Додати / Видалит об’єкт».

Варіант використання «Змінити інформацію»

Короткий опис:

Користувачеві доступна можливість редагувати інформацію що до обраного ним географічного об’єкта.

Основний потік подій:

1. Користувач обирає пункт «Редагувати»;
2. Користувач обирає необхідний географічний об’єкт;
3. Користувач вносить зміни у відповідні поля;
4. Користувач натискає кнопку «Запропонувати зміни».

Альтернативний потік подій:

1. Користувач не може використовувати пункт «Радагувати»;
2. Система видає причини.

Передумови:

Система працює стабільно, мапа відображається коректно, на мапі є хоча б один географічний об’єкт, користувач має ранг «Зареєстрований користувач».

Післяумови:

Система не вносить зміни в географічний об’єкт та надсилає форму запиту адміністрації.

Варіант використання «Додати / Видалити об’єкт»

Короткий опис:

Користувачеві доступний функціонал запропонувати додати об’єкт на мапу чи видалити існуючий.

Основний потік подій:

1. Користувач обирає пункт «Додати / Видалити» у меню;
2. Користувач точкою на мапі визначає куда хоче запропонувати додати об’єкт;
3. Користувач у поле введення даних про об’єкт вводить дані що йому відомі;
4. Користувач натискає на існуючий об’єкт на мапі, який хоче запропонувати видалити;
5. Користувач натискає кнопку «Запропонувати зміни»;

Альтернативний потік подій:

1. Користувач не може використовувати якийсь з пунктів;
2. Система видає причину.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», мапа відображається коректно, на мапі є хоча б один географічний об’єкт, що задовольнить умову для видалення його.

Післяумови:

Система не вносить зміни до мапи та надсилає форму запиту адміністрації.

Варіант використання «Зміни в обліковому записі»

Короткий опис:

Користувач може змінювати та додавати дані про власний профіль в особистому обліковому записі, а також видалити чи вийти з акаунта.

Основний потік подій:

1. Користувач знаходить на веб сторінці область ведення акаунтів;
2. Користувач знаходить та натискає на іконку власного профілю;
3. Система надає меню для налаштування акаунта

Альтернативний потік подій:

1. Користувач не може зайти у відповідне меню;
2. Система пояснює причини.

Передумови:

Система працює стабільно, користувач має ранг «Зареєстрований користувач», інтерфейс відображається коректно.

Післяумови:

Система надає користувачеві доступ до таких варіантів використання як «Вийти / Видалити» та «Редагувати персональні дані»

Варіант використання «Вийти / Видалити»

Короткий опис:

Користувач може видалити свій акаунт, або вийти з існуючого понизивши ранг до стандартного - «Користувач».

Основний потік подій:

1. Користувач обирає пункт видалити акаунт;
2. Користувач натискає кнопку «Видалити акаунт»;
3. Користувач підтверджує видаленням натисканням кнопки «Підтвердити» у сповіщенні про видалення акаунта;
4. Система понижає ранг користувача до стандартного - «Користувач»;
5. Система видаляє дані облікового запису.

Альтернативний потік подій:

1. Користувач обирає пункт вийти з акаунта;
2. Користувач натискає на кнопку «Вийти»;
3. Система понижає ранг користувача до стандартного - «Користувач»;

Передумови:

Система працює стабільно, інтерфейс відображається коректно, користувач має ранг «Зареєстрований користувач».

Післяумови:

Система працює відповідно до потоку подій, в будь-якому випадку вона понижає ранг до стандартного.

Варіант використання «Редагувати персональні дані»

Короткий опис:

Система дозволяє користувачеві змінювати власні персональні дані.

Основний потік подій:

1. Користувач знаходить та натискає кнопку «Редагувати»;
2. Користувач вносить зміни;
3. Користувач настискає кнопку «Зберегти» якщо зміни задовольняють;
4. Користувач натискає кнопку зберегти;
5. Система зберігає дані користувача;

Альтернативний потік подій:

1. Користувач натискає кнопку «Скасувати», оскільки зміни не задовольняють його;
2. Система повертає попередні значення.

Передумови:

Система працює стабільно, інтерфейс відображається коректно, користувач має ранг «Зареєстрований користувач».

Післяумови:

Система вносить відповідні зміни до бази даних щодо інформації про відповідного користувача.

Для повноти картини розробки додамо теоретичну діаграму варіантів використання для акторів «Розробник» та «Замовник», що показати процес отримання та реалізації замовлення, тим самим проілюструвати можливу комерційну бізнес модель.

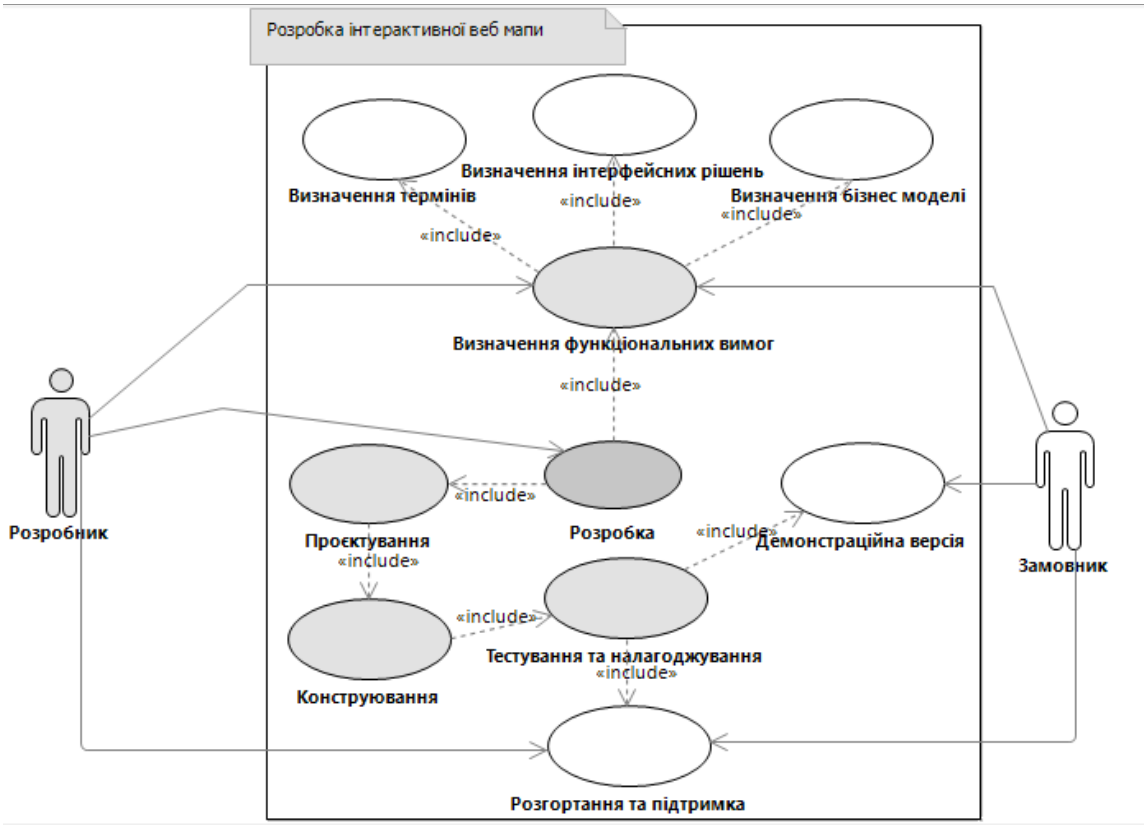

Рисунок В.2 – Діаграма варіантів використання для акторів «Розробник» та «Замовник»

Така діаграма варіантів використання може допомогти візуалізувати взаємодію і взаємозв'язки в рамках розробки інтерактивних веб мап. На діаграмі відображений процес розробки інтерактивної мапи із відповідними акторам як «Розробник» та «Замовник», діаграма демонструє потік діяльності та взаємодії між замовником і розробником, забезпечуючи огляд процесу розробки та цінності, що надається потенційним замовникам.