Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Кафедра математики та інформатики

ДИПЛОМНА РОБОТА

на здобуття другого (магістерського) рівня вищої освіти

на тему «Розробка веб-додатка для моніторингу та управління природними ресурсами на основі ГІС»

Виконав: студент _____ курсу, групи_____
спеціальності Комп'ютерні науки
(шифр і назва спеціальності)
Чекан В.І.
(прізвище та ініціали студента)
Керівник _____
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент _____
(науковий ступінь, вчене звання, посада, прізвище та ініціали)

Івано-Франківськ – 2024 р.

АНОТАЦІЯ

Дипломна робота складається зі вступу, двох розділів з підрозділами, висновків до кожного розділу, загальних висновків списку використаних джерел, додатків. Дипломна робота налічує 65 сторінок, 29 рисунків, 10 таблиць, 7 додатків. Список використаних джерел налічує 56 позицій.

У першому розділі здійснено аналіз предметної області. В даному напрямку розглянуто управління природними ресурсами за допомогою ГІС, оглянуто існуючі програмні рішення. Обґрунтовано необхідність розробки веб-додатка для моніторингу та управління природними ресурсами, здійснено огляд засобів для розробки веб-додатків.

У другому розділі показано особливості розробки веб-додатка. Було наведено вимоги до веб-додатку, його архітектуру. Показано розробку бази даних, frontend та backend, інтерактивності з геоданими, а також розгортання та підтримку. Сформульовано оцінку ефективності розробки.

Ключові слова: ГІС, природні ресури, веб-додаток, геопросторові дані, екологія.

ANNOTATION

The thesis consists of an introduction, two sections with subsections, conclusions to each section, general conclusions, a list of used sources and appendices. The thesis has 65 pages, 29 figures, 10 tables, 7 appendices. The list of used sources includes 56 items.

In the first section, we carried out the analysis of the subject area. In this direction, we considered the management of natural resources with the help of GIS; also, there are a review of the existing software solutions. In this section, we substantiated the need to develop a web application for monitoring and management of natural resources, and reviewed tools for developing web applications.

The second section shows the features of web application development. In this direction, we gave the requirements for the web application and its architecture. There are shown database development, frontend and backend, interactivity with geodata, as well as deployment and support. In addition, an assessment of the effectiveness of the development has been formulated.

Keywords: GIS, natural resources, web application, geospatial data, ecology.

ЗМІСТ

ВСТУП 5
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ДОЦІЛЬНІСТЬ РОЗРОБКИ ВЕБ-ДОДАТКА 7
1.1. Управління природними ресурсами за допомогою ГІС 7
1.2. Огляд існуючих програмних рішень 11
1.3. Обґрунтування розробки веб-додатка для моніторингу та управління природними ресурсами 19
1.4. Огляд засобів для розробки веб-додатків 24
Висновки до розділу 1 35
РОЗДІЛ 2. ОСОБЛИВОСТІ РОЗРОБКИ ВЕБ-ДОДАТКА 37
2.1. Вимоги до веб-додатку 38
2.2. Архітектура веб-додатка 41
2.3. Розробка бази даних 43
2.4. Розробка frontend та backend 45
2.5. Інтерактивність з геоданими 50
2.6. Розгортання та підтримка 55
2.7. Оцінка ефективності розробки 58
Висновки до розділу 2 63
ВИСНОВКИ 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
ДОДАТКИ 70

ВСТУП

Актуальність. Розробка веб-додатка для моніторингу та управління природними ресурсами включає в себе збір інформації про кліматичні зміни, динаміку використання ґрунтів, лісових масивів, водних ресурсів тощо. Інформація обробляється та аналізується з використанням різноманітних методів, що дозволяє виявляти тенденції та визначати оптимальні стратегії управління ресурсами. Це досягається за умови використання ГІС, адже за допомогою ГІС можна створювати картографічні представлення, які надають користувачам зрозумілий спосіб перегляду та аналізу інформації, такі як різноманітні шари, що показують стан різних ресурсів, їх зміни з часом, прогнози тощо.

Такий веб-додаток зможе надавати користувачам можливість взаємодії з даними, встановлення параметрів аналізу, створення власних шарів, а також приймання рішень на основі аналізу доступної інформації, це спонукає нас до дослідження даної теми.

Мета дослідження – напрацювання теоретичних аспектів поставленого питання, а також демонстрація розробки веб-додатка, який буде забезпечувати ефективний моніторинг та управління природними ресурсами з використанням інструментів геоінформаційних систем.

Задля досягнення мети було поставлено наступні **завдання**:

1. Дослідити особливості управління природними ресурсами за допомогою ГІС.

2. Здійснити огляд існуючих програмних рішень.

3. Обґрунтувати необхідність розробки веб-додатка для моніторингу та управління природними ресурсами.

4. Показати особливості розробки веб-додатку.

5. Оцінити очікувану ефективність.

Об’єкт – процес моніторингу та управління природними ресурсами.

Предметом дослідження є розробка веб-додатка, який базується на геоінформаційних технологіях та призначений для моніторингу і управління різноманітними природними ресурсами, такими як ліси, водні та ґрунтові ресурси.

Методи дослідження. В роботі було використано методи аналізу літературних джерел для обробки теоретичної інформації, порівняння та систематизація виокремлених даних, методи системного аналізу, програмування веб-додатків, а також аналізу та обробки геопросторових даних.

Практичне значення. Результати дослідження мають велике практичне значення для різноманітних сфер, від екологічного управління до сільського господарства. Розробка веб-додатку сприятиме оптимізації використання природних ресурсів, покращенню екологічної ситуації. Дипломна робота може бути використана в якості посібника по розробці такого веб-додатка.

Структура дипломної роботи обумовлена метою та поставленими завданнями. Робота складається зі вступу, двох розділів з підрозділами та висновками до кожного, загальних висновків, списку використаних джерел та додатків. В дипломній роботі наведено 10 таблиць, 29 рисунків. Список використаних джерел містить 56 позицій. До роботи долучено 7 додатків на 15 сторінках.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ДОЦІЛЬНІСТЬ РОЗРОБКИ ВЕБ-ДОДАТКА

1.1. Управління природними ресурсами за допомогою ГІС

За останні десятиліття географічні інформаційні системи (ГІС) стали потужним інструментом для збору, аналізу та використання даних про природні ресурси. У цьому підрозділі буде розглянуто, сутність ГІС та як вони сприяють ефективному управлінню природними ресурсами.

ГІС – це сучасна комп’ютерна технологія для картування і аналізу об’єктів реального світу, також подій, які відбуваються на нашій планеті. Ця технологія об’єднує традиційні операції роботи з базами даних, такими як запит і статистичний аналіз, з перевагами повноцінної візуалізації та географічного (просторового) аналізу, які надає карта [6].

За словами О.О. Світличного та С.В. Плотницького, ГІС – це інтегрована сукупність апаратних, програмних і інформаційних засобів, що забезпечують введення, збереження, обробку, маніпулювання, аналіз і відображення (представлення) просторово-координованих даних [7, с. 17].

В.Д. Шипулін та Є.І. Кучеренко стверджують, що ГІС – це складна система, в яку входять взаємозв'язані шість основних компонентів: інформаційні продукти (бажані вихідні матеріали, одержані за допомогою ГІС), програмне забезпечення (комп'ютерні програми, що забезпечують функції, необхідні для виконання аналізу і створення бажаних інформаційних продуктів), дані, апаратне забезпечення, процедури, люди [8, с. 14].

І.В. Пітак та ін. дають розширене визначення: ГІС – це інформаційна система, тобто «система обробки даних, що має засоби накопичення, збереження, відновлення, пошуку і видачі даних»; ця інформаційна система належить до категорії автоматизованих інформаційних систем, «що використовують ЕОМ на всіх етапах обробки інформації»; ця інформаційна

система надає можливості маніпулювання і обробки просторової (просторово–розподіленої, просторово–координованої) інформації [3, с. 24].

В.І. Зацерковний та ін., навпаки, пропонують найкоротше визначення: «ГІС – це просторовоорієнтована база даних» [2, с. 38].

К. Пілі в своїй публікації використовує наступне визначення ГІС: це комп'ютерна інформаційна система, яка призначена для роботи з даними або інформацією, тобто з просторовими чи географічними координатами [42, с. 1515].

За визначенням Р. Аблера ГІС – це комплекс апаратно-програмних засобів і діяльності людини зі збереження, маніпулювання та відображення географічних (просторово-співвіднесених) даних [9].

А. Дегані зазначає, що ГІС – це динамічно організована множина даних, поєднана з множиною моделей, реалізованих на ЕОМ для розрахункових, графічних і картографічних перетворень цих даних у просторову інформацію з метою задоволення специфічних потреб певних користувачів у межах структури точно визначених концепції і технологій [18].

Найбільш повне визначення, запропоновано фахівцями Інституту дослідження систем навколишнього середовища (ESRI) у посібнику користувача системи ARC / INFO: «Географічна інформаційна система – це організований набір апаратних і програмних засобів, географічних даних і персоналу, призначений для ефективного отримання, збереження, відновлення, обробки, аналізу й одержання зображення всіх видів географічно прив'язаної інформації. За допомогою ГІС можуть бути виконані реальні складні просторові операції, які за інших умов були б дуже складними, тривалими в часі або непрактичними» [53].

Втім, в зарубіжних літературних джерел зустрічаються і вузькопрофільні визначення ГІС. Наприклад, географічна інформаційна система (ГІС) – це технологія, яка забезпечує засоби для збору та використання географічних даних для допомоги в розвитку сільського господарства [27].

Загалом, ГІС є потужним інструментом для збору, аналізу та використання географічної інформації в різних галузях, включаючи управління природними ресурсами, містобудування, транспорт, геологію, екологію та інші. З наведених визначень можна виокремити особливості ГІС (табл. 1.1).

Таблиця 1.1

Особливості ГІС

Особливість	Характеристика
Географічна спрямованість	здатність працювати з географічною інформацією, такою як координати, форми земельних ділянок, висоти та інші географічні атрибути
Інтеграція даних	взаємодія з різними джерелами, такими як супутникові знімки, аерофотознімки, цифрові карти, бази даних тощо, для створення комплексної географічної бази даних
Аналіз просторових взаємозв'язків	здатність аналізувати взаємозв'язки між різними географічними об'єктами та явищами, що дозволяє виявляти та прогнозувати певні закономірності
Візуалізація	використання карт та графіків, що допомагає зрозуміти та представити інформацію зрозумілим та доступним способом
Моделювання і прогнозування	можливість використання для моделювання різних географічних процесів та подій, таких як розповсюдження забруднень, зміни клімату, розробка містобудівних проектів тощо, що дозволяє робити прогнози та оцінювати наслідки різних сценаріїв
Підтримка прийняття рішень	є інструменти для прийняття обґрунтованих рішень в географічно орієнтованих ситуаціях, шляхом аналізу даних та врахування різних факторів

Модель управління природними ресурсами за допомогою ГІС відображено на рисунку 1.1.

Рис. 1.1. Управління природними ресурсами за допомогою ГІС

Так, ГІС дозволяють збирати та організовувати велику кількість геопросторових даних про природні ресурси, такі як ліси, водні ресурси, ґрунти, кліматичні умови тощо. Це дає змогу управлінцям отримувати повну картину про стан природних ресурсів на конкретній території. ГІС також дозволяють проводити аналіз цих даних шляхом застосування різноманітних методів та моделей. Наприклад, за допомогою ГІС можна визначити оптимальні місця для розміщення нових лісів або встановити області з високим ризиком ерозії ґрунтів. Такий аналіз допомагає приймати обґрунтовані рішення щодо використання природних ресурсів.

В той же час, ГІС сприяють моніторингу та контролю за використанням природних ресурсів. За допомогою спеціальних програм можна вести постійний моніторинг змін в ландшафті, рівні води в річках та інших параметрів середовища. Це дозволяє оперативно реагувати на негативні зміни та вчасно приймати заходи щодо їх запобігання.

Не можна також не зазначити, що ГІС полегшують співпрацю між різними зацікавленими сторонами. Вони надають зручний інтерфейс для обміну даними та спільної роботи над проектами, пов'язаними з управлінням природними ресурсами.

Отже, можна зробити висновок, що використання ГІС для управління природними ресурсами дозволяє ефективно використовувати ці ресурси, зменшувати негативний вплив на довкілля та забезпечувати сталість їх використання для майбутніх поколінь. Важливо продовжувати розвивати цей напрямок інформаційних технологій для забезпечення сталого розвитку суспільства.

1.2. Огляд існуючих програмних рішень

На сьогоднішній день функціонує декілька веб-додатків та платформ, які пропонують моніторинг та управління природними ресурсами на основі ГІС. Деякі з них використовуються для конкретних сфер, таких як лісове господарство, екологічний моніторинг, сільське господарство тощо. Ці програмні рішення надають користувачам можливість моніторити стан природних ресурсів, а також використовувати ці дані для прийняття рішень щодо їх управління та охорони. В даному підрозділі будуть розглянуті міжнародні та українські програмні рішення.

Для аналізу було обрано наступні веб-додатки:

1. Global Forest Watch. Ця платформа надає інформацію про стан лісових масивів у всьому світі на основі аналізу супутникових даних та інших джерел інформації [24]. Вона дозволяє користувачам моніторити зміни в лісовому

покриві, виявляти випадки вирубки лісів, а також стежити за реалізацією зон охорони.

Веб-додаток має сучасний та інтуїтивно зрозумілий інтерфейс користувача. Структура сайту включає в себе наступні сторінки: карта, панель приладів, довідка, про сайт, блог та інші інструменти (рис. 1.2).

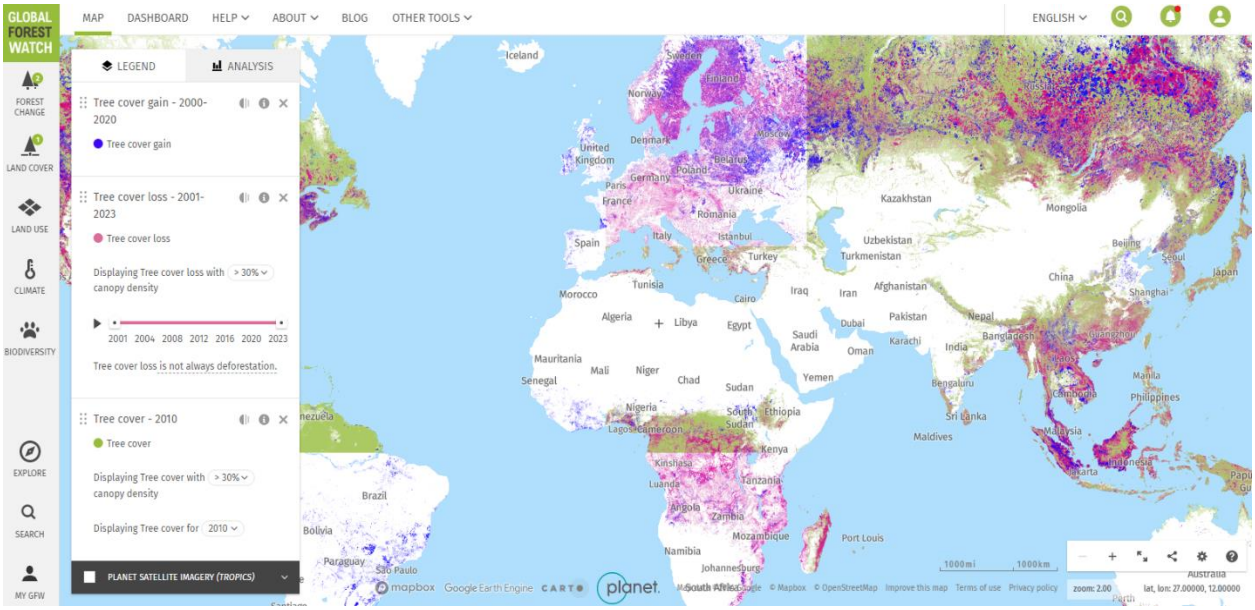

Рис. 1.2. Веб-додаток Global Forest Watch [24]

Переваги: надає доступ до глобальних даних про лісові ресурси; дозволяє виявляти випадки незаконної вирубки лісів та інші загрози довкіллю; забезпечує можливість моніторингу змін в лісовому покриві на різних масштабах. Недоліки: обмежена доступність деяких функцій безкоштовно; деякі дані можуть бути застарілими.

2. Google Earth Engine. Розроблений Google, цей веб-додаток надає доступ до обширного архіву географічних даних та інструментів для їх обробки та аналізу. Earth Engine використовується для різних задач, включаючи моніторинг змін клімату, розрахунок показників рослинності та визначення зон ризику екологічних катастроф [30].

Google Earth Engine має великий обсяг супутникових зображень від різних супутникових місій, таких як Landsat, Sentinel, MODIS та інших. Ці дані охоплюють багато років спостережень і дозволяють виконувати аналіз змін на земній поверхні протягом тривалого часу. Також платформа надає широкий

набір інструментів та бібліотек для обробки та аналізу географічних даних. Платформа має API, що дозволяє розробникам створювати власні програми та інструменти на базі Google Earth Engine. Це дає можливість інтегрувати функціонал Earth Engine у власні веб-додатки та проекти.

Структура сайту включає в себе наступні сторінки: платформа, набори даих, таймлапс, тематичні досліджння, поширені запитання, початок роботи, комерційний та некомерційни доступ. Веб-додаток також містить підрозділи: огляд, редактор коду, документація. Широкий функціонал зумовлює складності у взаємодії, тому веб-додаток може бути незрозумілим в користуванні для пересічного користувача (рис. 1.3).

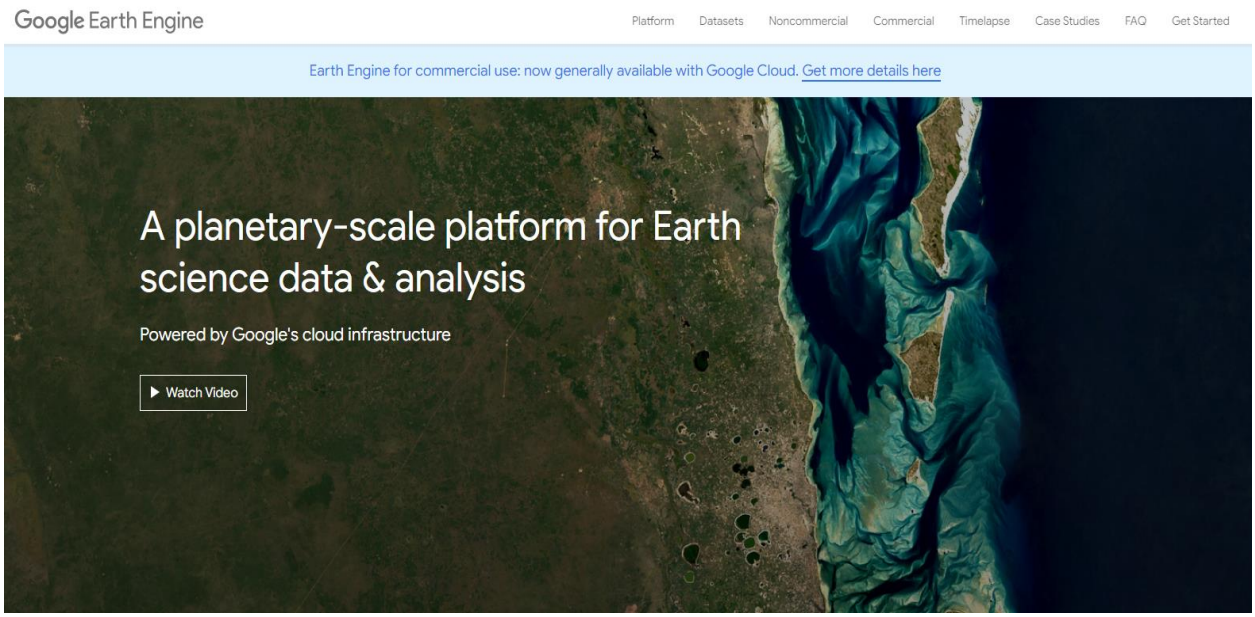

Рис. 1.3. Веб-додаток Google Earth Engine [30]

Переваги: має великий обсяг географічних даних, включаючи супутникові зображення; широкі можливості для аналізу даних та використання інструментів штучного інтелекту та машинного навчання; легка інтеграція з іншими сервісами Google. Недоліки: складність використання для неспеціалістів; обмежені можливості доступу до певних функцій безкоштовно.

3. USGS Earth Explorer. Веб-додаток, який надає доступ до супутникових зображень, літакових знімків та інших географічних даних, зібраних

американським Управлінням з геологічних досліджень (USGS) [23]. Ця платформа широко використовується для моніторингу змін природних ресурсів, таких як водні та лісові ресурси, а також геологічні утворення.

Структура веб-додатку включає в себе наступні сторінки: довідка, зворотній зв'язок, реєстрація, критерії пошуку, набори даних, результати (рис. 1.4). Слід заначити, що даний портал більш інформативний щодо моніторингу ресурсів США.

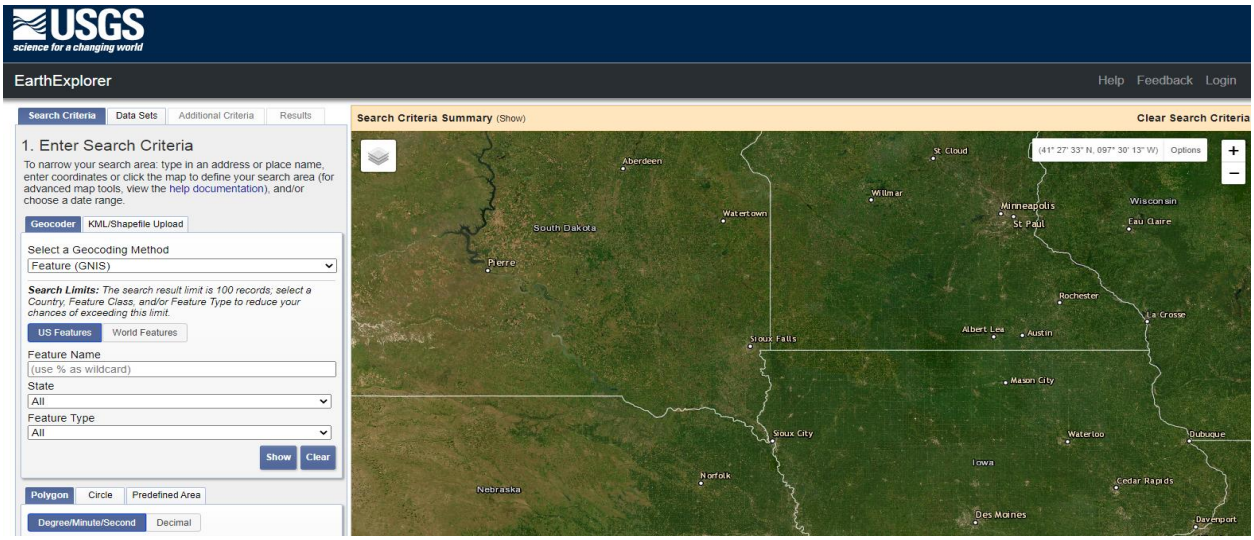

Рис. 1.4. Веб-додаток USGS Earth Explorer [23]

Переваги: надає доступ до широкого спектру географічних даних, включаючи супутникові зображення; легка навігація та пошук даних. Недоліки: обмежений функціонал порівняно з іншими платформами; деякі дані можуть бути застарілими або менш точними.

4. Global Fishing Watch. Цей веб-додаток використовує дані з супутникових систем відслідковування рибальських суден для моніторингу рибальних зон та виявлення незаконного, неповідомленого та нерегульованого рибальства [51]. Структура веб-додатку включає в себе наступні сторінки: теми (промислове рибальство, морські природоохоронні території, перевалка), карта та дані (карта, API, портал суден-носіїв, портал морського менеджера, перегляд суден, набори даних і код), програми (аналіз, дослідження, прозорість), новини, блог, про веб-додаток, довідка тощо (рис. 1.5). Веб-додаток наповнений великою кількістю інформації.

Рис. 1.5. Веб-додаток Global Fishing Watch [51]

Переваги: надає можливість моніторингу не лише відкритого океану, але і діяльності рибальських суден та виявлення незаконного рибальства; використовує дані з супутникових систем відслідковування суден. Недоліки: обмежений фокус на лише одному аспекті природних ресурсів - морських ресурсах.

5. Sentinel Hub (by Planet Labs). Веб-додаток, який надає доступ до даних від супутників Sentinel Європейського космічного агентства [47]. Його можна використовувати для моніторингу змін на земній поверхні, включаючи визначення впливу людської діяльності на природні ресурси. Структура веб-додатку включає в себе наступні сторінки: дослідження (браузер EO, дані, API, галузі та вітрини, екосистема, освіта, програми та утиліти), розвиток (панель приладів, спеціальні сценарії, інтеграція, API, спільнота, запитання), про веб-додаток, ціни, блог (рис. 1.6).

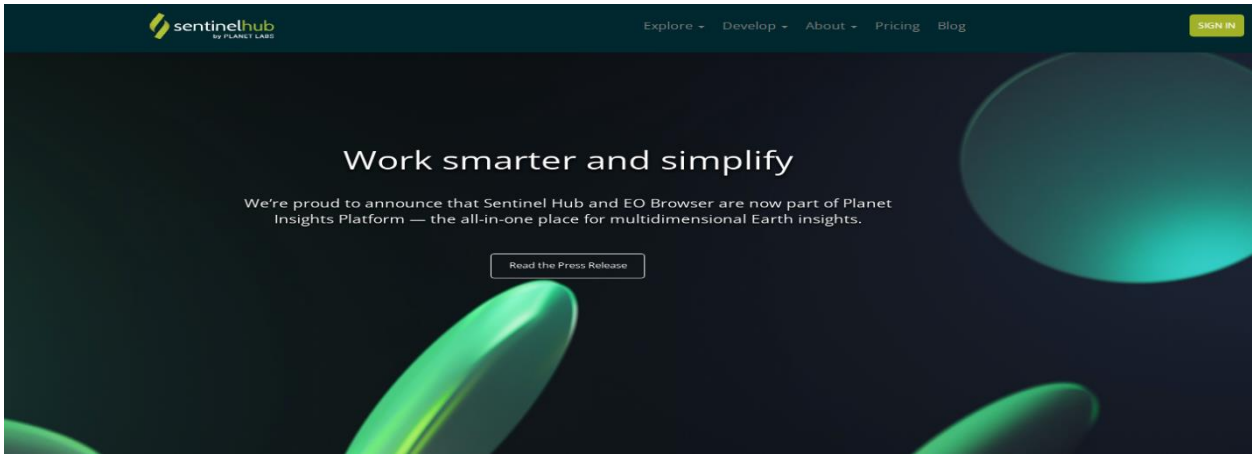

Рис. 1.6. Веб-додаток Sentinel Hub [47]

Переваги: надає доступ до даних супутникових систем, включаючи данні від Sentinel; має простий та інтуїтивно зрозумілий інтерфейс. Недоліки: обмежений функціонал порівняно з більш розгалуженими платформами, такими як Google Earth Engine.

6. Земельний кадастр України – це єдина геоінформаційна система землевпорядкування та земельних ресурсів України (ЄГІС ЗЗР). Ця система створена для забезпечення доступу до інформації про земельні ресурси та їх використання з використанням OpenStreetMap [1]. Веб-додаток містить дані про кадастрові та технічні характеристики земельних ділянок, а також дозволяє виконувати аналіз та моніторинг змін у земельному фонді. Структура веб-додатку дуже проста, включає в себе одну сторінку з мапою України з можливістю застосування фільтрів: форми власності, категорія земель, довкілля (заповідний фонд, річкова мережа, басейни річок тощо), адміністративний устрій (рис. 1.7).

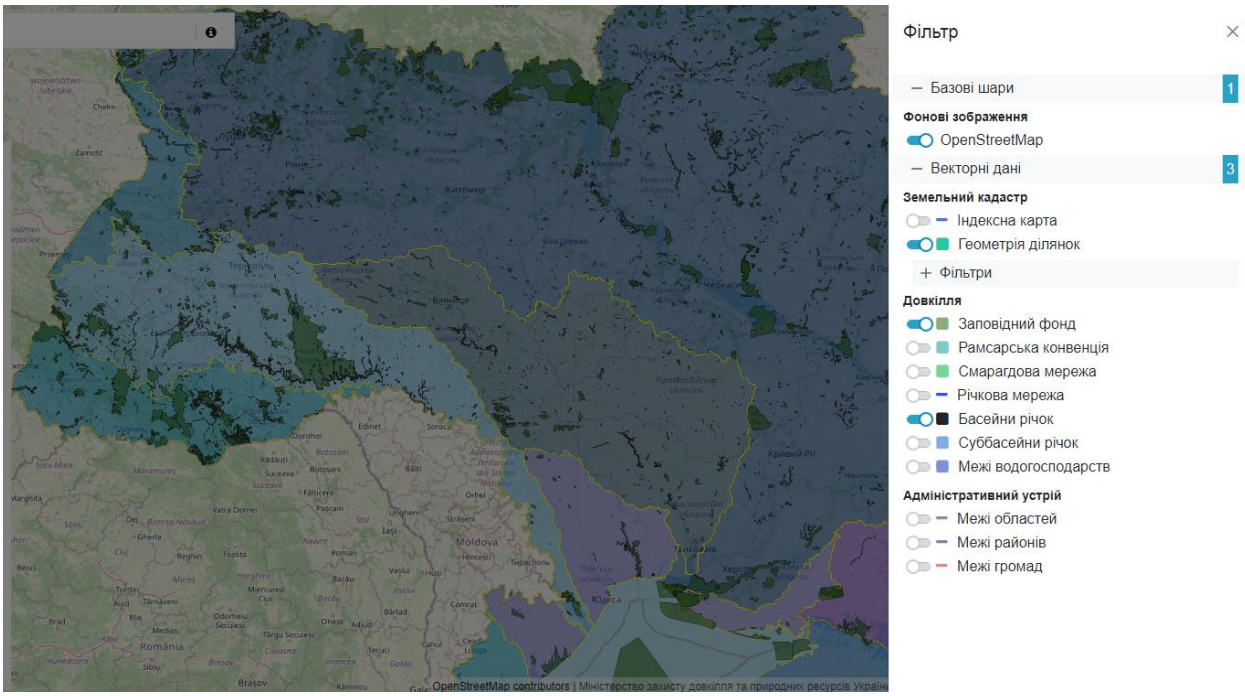

Рис. 1.7. Веб-додаток Земельний кадастр України [1]

Переваги: надає доступ до офіційних даних про земельні ділянки України; дозволяє отримати інформацію про власників, призначення земель та інші параметри. Недоліки: обмежений в доступі до деяких даних для

загального користувача; обмежена кількість природних ресурсів для моніторингу; обмежений фокус лише на Україні.

7. Геопортал «Ліси України» [4]. Цей веб-додаток містить інформацію про лісові ресурси України, включаючи дані про лісорубні роботи, відтворення лісів, мапи лісових масивів тощо. Слід зазначити, що веб-додаток потребує багато оперативної пам’яті, працює з помилками, малоінформативний (рис. 1.8).

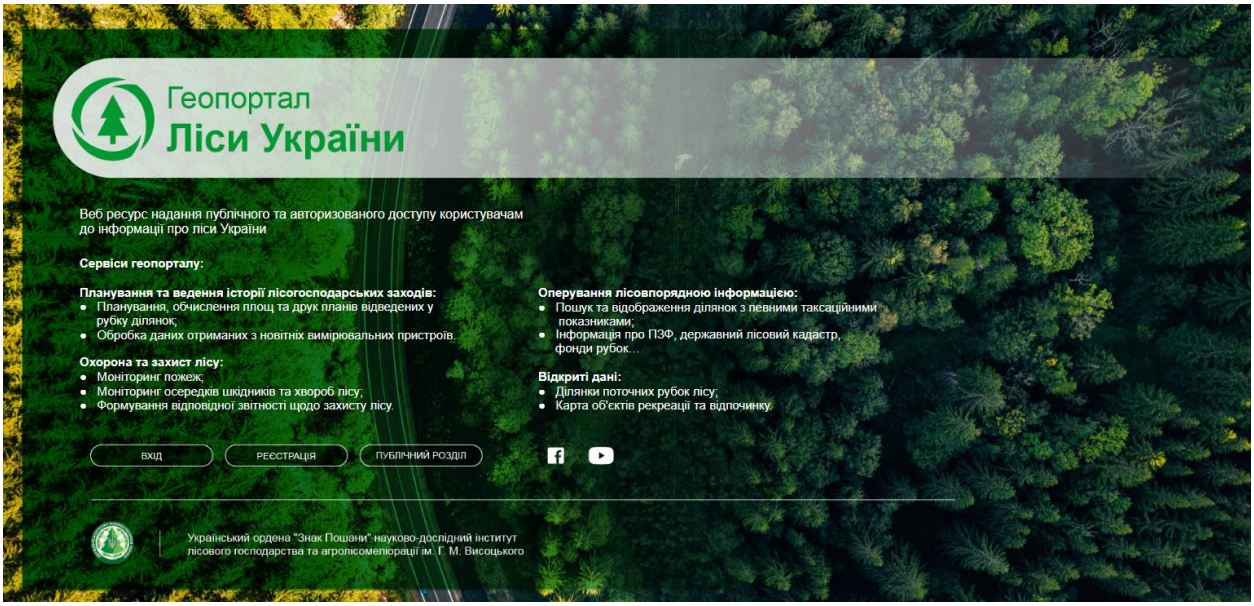

Рис. 1.8. Веб-додаток «Ліси України» [4]

Переваги: надає доступ до даних про лісові масиви та ресурси України; дозволяє вести моніторинг змін у лісовому покриві та проводити аналіз стану лісів. Недоліки: обмежений в доступі до деяких даних або функціоналу для загального користувача; обмежена кількість природних ресурсів для моніторингу; обмежений фокус лише на Україні.

В Україні було створено й інші веб-додатки, такі як український екологічний моніторинг (УЕМ), геоінформаційна система «Відкритий ліс», геопортал Державної служби України з надзвичайних ситуацій. Втім, на момент написання дипломної роботи (весна 2024 року) ці веб-додатки недоступні для публічного використання, а ГІС «Відкритий ліс» пропонується у використання за умови благодійного внеску.

Оцінити розглянуті веб-додатки можна за декількома критеріями методом експертних оцінок: зручність інтерфейсу, зрозумілість інтерфейсу, простота дизайну, інформативність, влучність вибору дизайну, загальне враження від веб-додатку, швидкодія (табл. 1.2). Шкала оцінювання – 5 балів, де 5 – відмінно, 4 – дуже добре, 3 – добре, 2 – погано, 1 – веб-додаток потребує змін.

Таблиця 1.2

Оцінка існуючих програмних рішень

	GFW	GEE	USGS EE	GFW	SH	ЗКУ	ГЛУ
Зручність інтерфейсу	5	5	5	4	4	4	4
Зрозумілість інтерфейсу	5	3	5	5	5	5	4
Простота дизайну	5	5	5	4	4	5	4
Інформативність	5	4	3	4	4	3	4
Влучність вибору дизайну	5	5	4	5	5	4	4
Загальне враження від веб-додатку	5	4	4	4	4	3	3
Швидкодія	5	4	4	4	4	5	2
Загальна оцінка	35	30	30	30	30	29	25

З табл. 1.2 бачимо, що Global Forest Watch має найвищу загальну оцінку – 35 балів. Sentinel Hub, USGS Earth Explorer, Google Earth Engine також мають високі загальні оцінки (30 балів), що робить їх конкурентоспроможними альтернативами. Земельний кадастр України та Геопортал «Ліси України» мають нижчі загальні оцінки, що свідчить про менш задовільний досвід користування веб-додатками. Загальні проблеми, які потребують вирішення – недостатня інтеграція та аналіз географічних даних для ефективного моніторингу та управління природними ресурсами; проблеми зі швидкодією веб-додатку.

Отже, існує велика кількість програмних рішень для моніторингу та управління природними ресурсами на основі ГІС. Ці програмні рішення

можуть бути використані для різноманітних завдань, пов'язаних з моніторингом та управлінням природними ресурсами, включаючи картографію, аналіз змін середовища, прогнозування, планування та прийняття рішень. Проведене дослідження показало, що кожен із розглянутих веб-додатків має свої переваги та обмеження, тому вибір між ними повинен залежати від конкретних потреб та завдань користувача. Наприклад, для глобального моніторингу лісів найбільш підходить Global Forest Watch, тоді як для аналізу географічних даних на рівні великомасштабних проектів може бути корисним використання Google Earth Engine.

1.3. Обґрунтування розробки веб-додатка для моніторингу та управління природними ресурсами

Розробка власного веб-додатка для моніторингу та управління природними ресурсами стане корисною для невеликих організацій, урядових установ, наукових дослідників та громадських організацій України, які займаються охороною природи та сталим розвитком.

Розробка власного веб-додатка для моніторингу та управління природними ресурсами має як переваги, так і недоліки.

До переваг відносимо наступне:

1. Власний веб-додаток може бути розроблений з урахуванням конкретних потреб і вимог управління природними ресурсами, що дозволяє точно відповідати вимогам користувачів.

2. Після розробки можна налаштувати додаток так, як це необхідно для конкретної організації або проєкту, що забезпечить максимальну ефективність у використанні.

3. Власний додаток дає контроль над даними, які будуть збиратись, оброблятись та зберігатись.

4. Можливість розширити функціонал додатка у майбутньому, додаючи нові функції або інтегруючи з іншими системами та сервісами.

Слід також звернути увагу і на недоліки:

1. Розробка власного веб-додатка може бути витратною за часом, а у випадку, якщо потрібно буде найняти додаткових помічників або консультантів, і у грошових витратах.

2. Після випуску додатку в експлуатацію потрібно буде забезпечувати його підтримку, виправлення помилок та оновлення, що може вимагати додаткових ресурсів та фінансових витрат.

3. Підвищення ризиків відносно безпеки та конфіденційності даних.

Звісно, переваг розробки веб-додатка більше. В той же час, Україна стикається зі складними проблемами у сфері охорони природи та управління природними ресурсами. Розробка веб-додатка для моніторингу та управління цими ресурсами може бути важливою і необхідною для декількох причин (рис. 1.9).

Рис. 1.9. Необхідність розробки веб-додатка для моніторингу та управління природними ресурсами

Відтак, розробка веб-додатка для моніторингу та управління природними ресурсами в Україні є важливою і може сприяти досягненню цілей сталого розвитку, збереженню біорізноманіття та забезпеченню екологічної безпеки країни. В той же час така розробка покращить ефективність управління цими ресурсами, сприятиме їх бережливому використанню та збереженню для майбутніх поколінь.

На даний момент, наприклад, практичну необхідність створення такого додатка вбачаємо для державного підприємства (ДП) «Ліси України», яке є одним з ключових учасників управління та збереження лісових ресурсів з великою кількістю філій по всій країні. Підприємство налічує 10 офісів та 147 філій на території України (рис. 1.10).

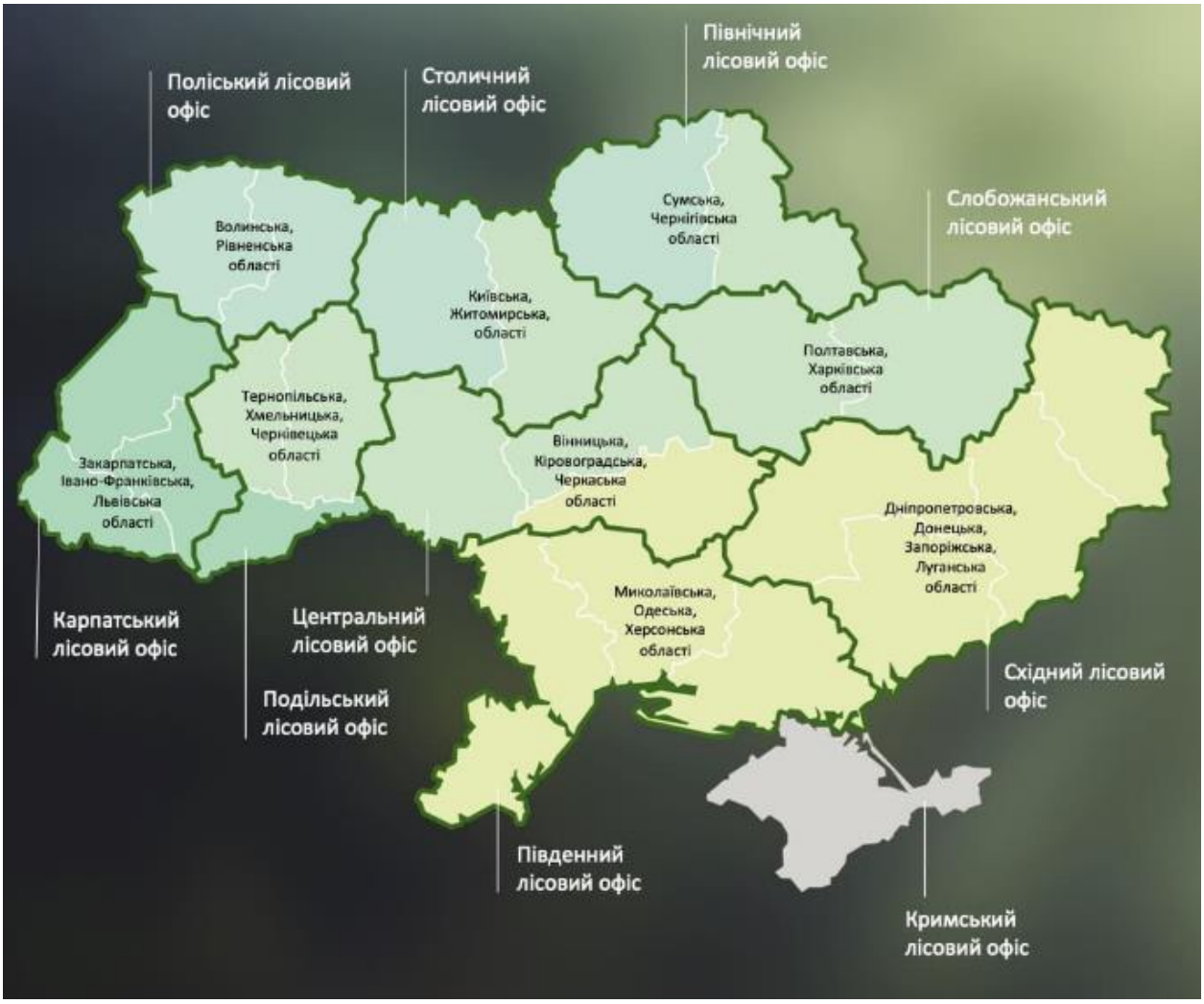

Рис. 1.10. Структура ДП «Ліси України» у 2024 році [5]

ДП «Ліси України» володіє значними територіями лісових масивів по всій території України. Ці масиви включають як ліси, що підлягають промисловому використанню, так і ліси, які мають важливе значення для охорони природи та біорізноманіття [5].

Основними завданнями даного підприємства є збереження, відтворення та раціональне використання лісових ресурсів, а також охорона та відновлення лісових екосистем. Кожна філія також відповідає за власне територіально-обмежене лісове господарство, а також здійснює контроль за дотриманням законодавства в галузі лісового господарства.

Підприємство здійснює науково-дослідну роботу у сфері лісового господарства та охорони природи, співпрацюючи з відомствами та науковими установами для вдосконалення методів управління лісовими ресурсами [5]. ДП «Ліси України» також взаємодіє з громадськістю та громадськими організаціями для забезпечення відкритості та прозорості управління лісовими ресурсами, а також для залучення громадськості до заходів з охорони лісів.

До викликів, з якими стикається дане підприємство, входять лісові пожежі, незаконні вирубки, збереження біорізноманіття та сталого використання лісових ресурсів у контексті зміни клімату та економічних труднощів.

З огляду на те, що державне підприємство «Ліси України» відіграє важливу роль у збереженні та управлінні лісовими ресурсами країни, розробка веб-додатка для моніторингу та управління лісовими ресурсами допоможе вирішити кілька ключових проблем, а саме:

1. Додаток дозволить здійснювати постійний моніторинг і контроль за станом лісових масивів, включаючи визначення рівня рубок, стану лісового покриву та прогресу відновлення лісів після рубок або природних катастроф.

2. Якщо в додатку будуть додані аналітичні інструменти, можна буде здійснювати оптимальне планування рубок, враховуючи природні особливості та сталий розвиток лісових ресурсів в кожному лісовому офісі країни.

3. При інтеграції додатку з IoT можна додати датчики пожеж для попередження та боротьби з лісовими пожежами. В результаті веб-додаток допоможе вчасно виявляти потенційно небезпечні області для виникнення пожеж та швидкого реагування на них.

4. Створення веб-додатка для моніторингу лісових ресурсів дозволить підвищити рівень відкритості та доступності інформації про управління лісами, що сприятиме підвищенню довіри громадськості та зменшенню корупції. Адже, як було зазначено раніше, сьогодні в Україні немає повністю відкритого та доступного веб-додатку з такими функціями.

5. Додаток може допомогти у плануванні і розподілі бюджетних коштів для управління лісовими ресурсами, забезпечуючи їх ефективне використання та враховуючи потреби у збереженні лісових екосистем.

Отже, розробка веб-додатка для моніторингу та управління природними ресурсами допоможе у зборі, аналізі та використанні даних про природні ресурси, такі як ліси, водні джерела, ґрунти тощо. В даному підрозділі було розглянуто необхідність розробки такого додатку для моніторингу та управління природними ресурсами України з метою вирішити ключові проблеми, з якими стикається державне підприємство «Ліси України», сприяючи кращому управлінню та збереженню лісових екосистем країни. В цьому веб-додатку пропонується не лише моніторинг лісових масивів, але і моніторинг водних джерел, ґрунтів, клімату, екосистем та біорізноманіття.

1.4. Огляд засобів для розробки веб-додатків

Для розробки веб-додатка для моніторингу та управління природними ресурсами на основі ГІС потрібно використовувати власне геоінформаційну систему та бібліотеки, веб-фреймворк, технології фронтенд розробки, базу даних, API, різноманітні розробницькі інструменти тощо. Розглянемо ці засоби та технології більш детально в даному підрозділі.

По-перше, обов’язкове використання геоінформаційної системи і бібліотек, адже ці системи надають можливості для роботи з географічними даними та їх візуалізації на веб-картах.

Найвідомішими ГІС і бібліотеками є наступні:

1. ArcGIS, розроблений компанією Esri, є одним з найпоширеніших інструментів для створення, аналізу та візуалізації геоінформації [12].

2. OpenLayers – це відкрита JavaScript бібліотека для роботи з інтерактивними картами і геопросторовими даними веб-додатків. Вона надає інструменти для візуалізації різних типів географічних даних, таких як точки, лінії, полігональні області, векторні та растрові шари, а також можливості для взаємодії з користувачем, включаючи маршрутизацію, маркери, інформаційні вікна та інше [40].

3. QGIS – це безкоштовна та відкрита ГІС-програма з розширеними можливостями для аналізу геоданих [45].

4. GRASS GIS – це інший відкритий проєкт, який пропонує багато інструментів для обробки та аналізу геопросторових даних [31].

5. Google Earth Engine – це хмарна платформа для аналізу географічних даних, яка надає доступ до великих обсягів супутникових даних і інструментів для їх обробки [30]. Google Earth Engine можна використовувати не лише в якості програмного рішення, але і як інструмент розробки.

6. Leaflet – це бібліотека JavaScript для створення інтерактивних карт у веб-додатках. Слід зазначити, що Leaflet був створений 11 років тому українцем Володимиром Агафонкіним, який живе в Києві [35].

7. GDAL (Geospatial Data Abstraction Library) – це бібліотека з відкритим вихідним кодом для обробки різних форматів геопросторових даних [26].

8. PostGIS – це розширення для бази даних PostgreSQL, яке дозволяє зберігати та оброблювати геодані з використанням стандартних SQL-запитів [43].

Ці інструменти та бібліотеки надають широкі можливості для роботи з геоданими та створення геоінформаційних систем з різними функціональними потребами. Переваги та недоліки розглянутих ГІС і бібліотек наведено в таблиці 1.3.

Таблиця 1.3

Порівняння найвідоміших ГІС і бібліотек

ГІС та бібліотеки	Переваги	Недоліки
ArcGIS	Широкий спектр функцій для аналізу, візуалізації та обробки геоданих; інтеграція з іншими продуктами Esri; можливості співпраці з геоданими та геопроцесінгом	Високі вартості ліцензій і обслуговування; потребує додаткового часу на навчання та освоєння
OpenLayers	Відкрите програмне забезпечення з великою активною спільнотою; простий у використанні та налаштуванні; підтримка великої кількості географічних форматів та проекцій	Складність при розгортанні на власних серверах; можливі проблеми з продуктивністю для великих обсягів даних
QGIS	Безкоштовне програмне забезпечення з великим функціоналом; можливість розширення за допомогою плагінів; спрощений інтерфейс користувача	Деякі функції можуть бути менш потужними порівняно з комерційними ГІС; обмежені можливості в роботі з великими обсягами даних
GRASS GIS	Вільне програмне забезпечення з багатьма функціями геопроцесінгу; підтримка широкого спектру географічних форматів; активна спільнота користувачів	Складність у використанні; обмежені можливості для створення інтерактивних веб-додатків
Google Earth Engine	Можливість роботи з великими обсягами супутникових даних; інтегрована платформа для аналізу та візуалізації геоданих; можливості машинного навчання та аналізу великих даних	Обмежена підтримка великих обсягів даних без платної підписки
Leaflet	Простота використання та інтеграції у веб-додатки; легкість у використанні та налаштуванні; можливість розробки інтерактивних карт	Вимагає високого рівня знань програмування та роботи з геоданими; складність налаштування
GDAL	Бібліотека для роботи з різними географічними форматами даних; широкі можливості обробки геоданих; підтримка різноманітних операцій з геоданими	Вимагає навичок роботи з командним рядком
PostGIS	Безкоштовне програмне забезпечення; можливість розширення за допомогою плагінів; інтеграція з PostgreSQL для роботи з геоданими; підтримка стандарту SQL	Можливі проблеми з продуктивністю для великих обсягів даних

На основі наданих переваг і недоліків, найкращим вибором для розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС є Google Earth Engine, а також PostGIS для інтеграції з базою даних.

Найвідоміші фреймворки для розробки веб-додатків включають Django (Python), Ruby on Rails (Ruby), Laravel (PHP), або Express.js (JavaScript). Найбільш використовувані:

1. Django – це високорівневий веб-фреймворк для розробки веб-додатків на мові Python. Django пропонує швидкий розвиток та вбудовану адміністративну панель [20].

2. Ruby on Rails – це фреймворк на мові програмування Ruby, який надає зручність та швидкість розробки завдяки конвенціям над конфігурацією [15].

3. Express.js – це легковаговий фреймворк для створення веб-додатків на мові JavaScript з використанням середовища виконання Node.js [16].

4. React.js / Vue.js / Angular. Ці фреймворки для фронтенду дозволяють створювати користувацький інтерфейс веб-додатків. React.js і Vue.js надають гнучкість та простоту використання, тоді як Angular пропонує широкі можливості для великих проектів [25].

5. Flask – це легковаговий веб-фреймворк для розробки веб-додатків на мові Python. Flask забезпечує гнучкість та простоту у використанні [22].

6. Spring Boot – це фреймворк на Java для швидкої розробки веб-додатків з використанням вбудованого сервера та інших зручних інструментів [54].

7. Laravel – це фреймворк на PHP, який пропонує простий та ефективний спосіб для розробки веб-додатків [49].

Ці фреймворки різняться за мовами програмування, підходами до розробки та функціональністю, і вибір залежить від конкретних потреб та вподобань розробника, тому порівняємо їх в таблиці 1.4.

Таблиця 1.4

Порівняння найвідоміших фреймворків

Веб-фреймворки	Переваги	Недоліки
Django	Швидке розгортання і простота використання; велика кількість готових модулів та пакетів для розширення функціоналу; вбудована підтримка аутентифікації, адміністративного інтерфейсу та інших основних функцій	Складнощі при масштабуванні на великі обсяги даних або велику кількість користувачів; складнощі при роботі з асинхронними операціями
Ruby on Rails	Швидкість розробки завдяки високій продуктивності та готовим рішенням; конвенції перед налаштуваннями для швидкого старту проекту; вбудовані інструменти для тестування та відлагодження	Складнощі при масштабуванні на великі обсяги даних; можливість зменшення продуктивності для складних додатків
Express.js	Мінімалістичний фреймворк, що надає швидкість розробки та гнучкість; простий у використанні та налаштуванні; активна спільнота розробників та розширення для підтримки	Менше підходить для великих та складних проектів у порівнянні з іншими фреймворками; потребує більшого обсягу ручного коду для складних операцій
React.js / Vue.js / Angular	Реактивність та ефективне управління станом додатку; модульна структура для простоти розширення та підтримки; широкий вибір компонентів та бібліотек для побудови інтерфейсу	Велика кількість варіантів та підходів при виборі між React.js, Vue.js та Angular
Flask	Легкий у використанні та налаштуванні; можливість інтеграції з іншими інструментами та бібліотеками Python; гнучкість у виборі баз даних та інших компонентів	Обмежена функціональність порівняно з іншими фреймворками; потребує додаткових бібліотек для реалізації деяких функцій; складність у використанні для великих проектів
Spring Boot	Висока продуктивність та швидкість розробки завдяки конвенціям і автоматизації; велика спільнота та підтримка від корпорації Spring; широкий вибір сторонніх бібліотек і модулів	Велика кількість конфігурацій та налаштувань, що може призвести до складнощів; складність у використанні для невеликих проектів; потребує додаткового часу для освоєння важливих концепцій
Laravel	Ефективне використання пам'яті та обробка запитів; велика кількість готових рішень та пакетів для розширення функціоналу; зручна підтримка веб-додатків з великим обсягом даних	Потребує додаткового часу на налаштування і знайомство з концепціями фреймворку; можливі проблеми з продуктивністю для великих обсягів даних та складних додатків

На основі наданих переваг і недоліків, для розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС найкращим вибором буде фреймворк React.js / Vue.js / Angular.

Frontend технології будуть використовуватися для створення інтерфейсу користувача веб-додатка. Щоб створити функціональний інтерфейс веб-додатків, розробники найчастіше використовують наступні frontend технології:

1. HTML (HyperText Markup Language) використовується для створення структури веб-сторінок та визначення їхнього змісту [21].

2. CSS (Cascading Style Sheets) використовується для візуального оформлення веб-сторінок, включаючи кольори, шрифти, розміри, макети та інше [38].

3. JavaScript – це мова програмування, яка використовується для додавання інтерактивності до веб-сторінок, маніпулювання DOM, взаємодії з користувачем та виконання різних завдань на клієнтському боці [38].

4. Sass/SCSS (Syntactically Awesome Stylesheets) – це розширення CSS, яке дозволяє використовувати змінні, вкладені стилі, міксіни та інші продуктивні можливості [36].

5. Bootstrap – це frontend фреймворк, який містить набір готових компонентів та стилів для швидкого створення відзначеного веб-дизайну [48].

Ці технології допомагають створювати зручні та ефективні веб-додатки з різноманітними функціональними можливостями, їх переваги та недоліки наведено в таблиці 1.5.

Таблиця 1.5

Порівняння найвідоміших frontend технологій

frontend технології	Переваги	Недоліки
HTML	Легко вивчити та використовувати; стандартна мова для створення веб-сторінок; ідеальний для структурування контенту	Обмежена функціональність без CSS та JavaScript; неможливість реалізації складних стилів без використання CSS
CSS	Легко вивчити та використовувати; дозволяє змінювати вигляд та стилізацію веб-сторінок; дозволяє розділити зовнішній вигляд від HTML-структури	Складно керувати стилями для складних макетів; ризик перевантаження стилів та змішування властивостей

Закінчення таблиці 1.5

frontend технології	Переваги	Недоліки
JavaScript	Мова програмування високого рівня; дозволяє зробити веб-сторінку інтерактивною; може взаємодіяти з DOM-деревом	Потребує знань програмування для розробки складних функцій
Sass/SCSS	Розширює можливості CSS за допомогою змінних, вкладених стилів, міксінів тощо; підтримка зручного синтаксису для розробників	Потребує додаткового часу для оволодіння синтаксисом та особливостями
Bootstrap	Містить готові компоненти та шаблони, що полегшує розробку інтерфейсу; адаптивний та мобільний дизайн; широкий вибір тем і зручна настройка	Може вимагати додаткового часу на кастомізацію для відповідності унікальним дизайнам; може виникнути збиток продуктивності через велику кількість непотрібних стилів; ризик перевантаження CSS-коду

З табл. 1.5 бачимо, що найбільш оптимальними фронтенд технологіями є CSS і JavaScript.

Для зберігання географічних даних та інформації про ресурси в розробці знадобиться база даних, яка забезпечить зберігання та операції з геопросторовою інформацією. До найбільш використовуваних баз даних відносяться наступні:

1. PostgreSQL – система управління базами даних з багатьма функціями та можливостями, яка підходить для широкого спектру застосувань, від невеликих веб-додатків до великих корпоративних систем [44]. Для PostgreSQL є геопросторове розширення PostGIS, що надає різноманітні функції для роботи з геоданими, такі як зберігання, об'єднання, аналіз та відображення.

2. MySQL – інша відома СУБД, в якій існує розширення з підтримкою геопросторових даних Spatial Extensions [38].

3. SQLite/SpatiaLite – легкий вбудовуваний двіжок баз даних, який також має розширення SpatiaLite для роботи з геоданими [38].

4. Oracle Spatial – розширення для бази даних Oracle, яке дозволяє зберігати та опрацьовувати геодані на платформі Oracle [33].

5. MongoDB – це NoSQL база даних, яка також має геопросторовий індекс і деякі геопросторові операції для роботи з геоданими [46].

6. CouchDB – ще одна NoSQL база даних, яка може використовуватися для зберігання геоданих та інших типів даних [11].

Ці бази даних надають різні можливості для зберігання та операцій з геоданими, порівняння за перевагами та недоліками наведено в таблиці 1.6.

Таблиця 1.6

Порівняння СУБД та розширень

СУБД та розширення	Переваги	Недоліки
PostgreSQL PostGIS	Повний функціонал геопросторових операцій та запитів; інтеграція з геоданими	Потребує встановлення PostgreSQL; складність в налаштуванні
MySQL Spatial Extensions	Нативна підтримка геопросторових операцій; покращена швидкість в порівнянні з попередніми версіями MySQL	Обмежена підтримка геопросторових операцій порівняно з іншими рішеннями; поганий підтримки індексів для геоданих
SQLite/SpatiaLite	Легкий у використанні та налаштуванні; можливість використання у вбудованому режимі	Не підходить для великих обсягів даних та високої навантаженості; обмежена функціональність порівняно з іншими
Oracle Spatial	Розширена підтримка геопросторових даних та операцій; інтеграція з іншими інструментами Oracle	Високі вартості ліцензії та підтримки Oracle; складність в інтеграції з відкритими системами
MongoDB	Гнучка схема без джерел даних; швидкий доступ до даних за допомогою NoSQL підходу	Менш підходить для складних геопросторових операцій порівняно з реляційними базами; недостатність підтримки транзакцій та змін у вибірках
CouchDB	Розподілена архітектура, що дозволяє масштабуватися горизонтально; можливість синхронізації даних на рівні документів	Обмежена функціональність геопросторових операцій та запитів; можливість втрати даних під час асинхронних записів

З табл. 1.6 бачимо, що на основі наданих переваг і недоліків, найкращою базою даних для розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС буде зв’язка PostgreSQL і PostGIS.

З огляду на те, що веб-додатку потрібен доступ до великої кількості географічних даних або сервісів, необхідно налатувати API для геоданих. Існує велика кількість API для роботи з геоданими, які надають доступ до різноманітної географічної інформації та сервісів:

1. Google Maps API надає доступ до широкого спектру функцій, включаючи карти, маршрутизацію, геокодування, відображення маркерів та багато іншого [32].

2. Mapbox API – це набір інструментів для створення інтерактивних карт, маршрутів та векторної графіки. Mapbox також надає можливості створення кастомних шарів та стилів [10].

3. OpenStreetMap API надає вільний доступ до геоданих та сервісів, таких як геокодування, маршрутизація та відображення карт [41].

4. HERE Maps API – набір інструментів для роботи з геоданими, включаючи карти, маршрутизацію, геокодування, трафік та багато іншого [37].

5. Leaflet API – це відкрита бібліотека JavaScript для роботи з інтерактивними картами у веб-додатках, вона підтримує різноманітні тайлові провайдери та може бути інтегрована з різними сервісами геоданих [35].

6. Esri ArcGIS API for JavaScript надає доступ до розширених геоінформаційних можливостей, таких як відображення шарів даних, аналіз геоданих та взаємодія з картою [52].

7. Weather API надає доступ до метеорологічної інформації, такої як прогнози погоди, температура, вітер та багато іншого [56].

Так, API надають різноманітні можливості для роботи з геоданими та інтеграції географічної інформації у веб-додатки, порівняння переваг та недоліків використання кожного з них наведено в таблиці 1.7.

Таблиця 1.7

Порівняння API

API	Переваги	Недоліки
Google Maps API	Простий у використанні та інтеграції; багато документації та прикладів	Обмежена кількість безкоштовних запитів; вартість використання зростає зі збільшенням обсягу трафіку
Mapbox API	Зручний API з широкими можливостями кастомізації; інтуїтивний та добре документований	Обмеження на безкоштовні плани
OpenStreetMap API	Безкоштовне та відкрите джерело; має широкий функціонал та добре документований API	Обмежені функціональні можливості порівняно з іншими API; недостатність підтримки та обмежена кількість безкоштовних запитів
HERE Maps API	Велика кількість готових функцій та послуг; висока швидкість та точність даних; підтримується велика кількість мов програмування	Обмеження на безкоштовний доступ; вартість може бути високою для великих обсягів трафіку

Продовження таблиці 1.7

API	Переваги	Недоліки
Leaflet API	Легкий та швидкий у використанні; має гнучкий API та можливості кастомізації; безкоштовне та відкрите джерело	Потребує додаткових плагінів для деяких функцій; може бути складно розгорнути на великих проєктах
Esri ArcGIS API for JavaScript	Має великий функціонал для роботи з геоданими; інтеграція з іншими продуктами Esri; підтримується велика корпорація та спільнота	Вартість ліцензії та підтримки може бути високою для підприємств; обмеження на безкоштовні плани
Weather API	Надає широкий спектр метеоданих та прогнозів; доступний безкоштовно або за невелику плату; легко інтегрується з іншими сервісами та додатками	Не завжди доступні всі необхідні метеодані; немає доступу до інших природніх реурсів; обмеження доступу до прогнозів на безкоштовному плані

На основі наданих переваг і недоліків, найкращим вибором для розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС є Mapbox API.

Не варто забувати про те, що для розробки знадобляться IDE (інструменти для розробки) – редактори коду, системи контролю версій та засоби для тестування.

До редакторів коду / інтегрованих середовищ розробки (IDEs) відносять наступні:

1. Visual Studio Code – безкоштовний редактор коду, який підтримує багато мов програмування та має розширення для підтримки різних технологій [19].

2. Sublime Text – легкий редактор коду з багатьма корисними функціями та плагінами [50].

3. Atom – безкоштовний та відкритий редактор коду, розроблений GitHub, з підтримкою плагінів [13].

Переваги та недоліки цих інструментів розробки наведено в таблиці 1.8.

Таблиця 1.8

Порівняння IDE

Параметр	Visual Studio Code	Sublime Text	Atom
Переваги			
Доступність	Безкоштовне та відкрите джерело; підтримка розширень та налаштувань	Швидкий та легкий; підтримка розширень та налаштувань; велика кількість додаткових пакетів	Безкоштовне та відкрите джерело; підтримка розширень та налаштувань; інтеграція з GitHub
Швидкодія	Висока швидкодія та продуктивність; широкий функціонал та можливості налаштувань; підтримка великої кількості мов програмування	Дуже швидкий та ефективний у використанні; необмежені можливості налаштувань та розширень	Легкий та швидкий; підтримка багатьох мов програмування; інтеграція з GitHub
Розширення	Широкий вибір розширень для роботи з різними мовами програмування, фреймворками та інструментами; велика кількість тем оформлення	Великий вибір додаткових пакетів та розширень; підтримка користувацьких розширень	Широкий вибір розширень та тем оформлення; інтеграція з іншими інструментами та сервісами
Недоліки			
Великі ресурси	Використання великої кількості ресурсів пам'яті та процесора, особливо для великих проєктів	Платний застарілий редактор, який може вимагати значних ресурсів; з’являється мало нових функцій	Високе споживання пам'яті, особливо при великих обсягах даних; можливі проблеми з продуктивністю на старих комп'ютерах

На основі переваг і недоліків в табл. 1.8, для розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС найкращим вибором буде Visual Studio Code.

Іншими інструментами розробки є інструменти для тестування та налагодження, для керування версіями та співпраці, системи керування пакетами.

До інструментів для тестування та налагодження відносять наступні:

1. Jest – фреймворк для тестування JavaScript, який забезпечує простий та потужний інтерфейс для тестування коду [34].

2. Chrome DevTools – набір інструментів для розробників у браузері Google Chrome, який дозволяє аналізувати, налагоджувати та профілювати веб-додатки [17].

Інструменти керування версіями включать в себе наступні:

1. Git – система керування версіями для відстеження змін у коді та спільної роботи над проектами [28].

2. GitHub / GitLab / Bitbucket – хмарні платформи для спільної роботи над проєктами, що базуються на Git, які надають інструменти для керування проєктами, проблемами та спільної роботи над кодом [14], [29].

Найбільш оптимальною системою керування пакетами є npm (Node Package Manager) – система керування пакетами для JavaScript та Node.js, яка дозволяє легко встановлювати, оновлювати та управляти залежностями проекту [39].

Переваги та недоліки цих інструментів розробки наведено в таблиці 1.9.

Таблиця 1.9

Порівняння інструментів розробки

Параметр	Jest	Chrome DevTools	Git	GitHub / GitLab / Bitbucket	npm (Node Package Manager)
Переваги					
Тестування	Простий у використанні та налаштуванні; широкий функціонал тестування JavaScript, включаючи підтримку React та інших фреймворків	-	-	-	-
Відлагодження	-	Має інтерфейс для відлагодження JavaScript та CSS; дозволяє відстежувати різні події та маніпулювати DOM	-	-	-
Контроль версій	-	-	Потужний та популярний інструмент для керування версіями файлів; дозволяє створювати гілки, коміти, злиття тощо	Надають хмарні сервіси для зберігання коду, співпраці, перегляду та керування версіями.	-

Продовження таблиці 1.9

Параметр	Jest	Chrome DevTools	Git	GitHub / GitLab / Bitbucket	npm (Node Package Manager)
Пакування та залежності	-	-	-	-	Дозволяє легко управляти залежностями проєкту, встановлювати та оновлювати пакети з npm репозиторію
Недоліки					
Складність	Складне налаштування для великих проєктів; вимагає додаткового часу на вивчення та розуміння синтаксису	-	Вимагає вивчення та розуміння командного рядка Git; може виникати конфлікт між гілками при злитті змін	Часто виникає конфлікт між розробниками при злитті змін; деякі функції доступні тільки на платних тарифах	Може виникати проблеми з розробленими версіями пакетів, несумісними з іншими пакетами
Ресурсозатратність	Вимагає великої кількості ресурсів для виконання тестів, особливо на великих проєктах	Може вимагати великої кількості пам'яті та процесорних ресурсів для виконання відлагодження великих або складних проєктів	Може виникати конфлікт між різними версіями пакетів та розгалуженням и проєкту в гіт-репозиторії	Потребує доступу до Інтернету для синхронізації змін з хмарним репозиторієм	Може виникати збій пакування або встановлення пакетів через проблеми з інфраструктурою npm

Інструменти, наведені в табл. 1.8 та табл. 1.9, забезпечать повний цикл розробки веб-додатку для моніторингу та управління природними ресурсами на основі ГІС, починаючи з тестування та відлагодження коду, через керування версіями та спільну роботу, до управління залежностями та пакетами.

Висновки до розділу 1

В першому розділі було здійснено аналіз предметної області, показано доцільність розробки веб-додатка.

Визначено, що використання геоінформаційних систем (ГІС) дозволяє ефективно управляти природними ресурсами, аналізувати їх стан та вплив на навколишнє середовище, в результаті чого приймати обґрунтовані рішення щодо їх використання та охорони.

Проведено огляд існуючих програмних рішень, щоб визначити їхні переваги, недоліки та відповідність потребам та вимогам проблематики даного дипломного проєкту. Порівняння цих веб-додатків показало, що міжнародні програмні рішення направлені на використання розробниками (Sentinel Hub, USGS Earth Explorer, Google Earth Engine), інші зосереджені на конкретному напрямку, наприклад, лісові ресурси (Global Forest Watch) та морські ресурси (Global Fishing Watch). Українські веб-додатки мають невелику географічну базу даних (Земельний кадастр України) та недостатність інструментів для моніторингу і аналізу (Геопортал «Ліси України»). Загальні проблеми, які потребують вирішення – недостатня інтеграція та аналіз географічних даних для ефективного моніторингу та управління природними ресурсами; проблеми зі швидкодією веб-додатку.

Розробка веб-додатка для моніторингу та управління природними ресурсами допоможе у зборі, аналізі та використанні даних про природні ресурси, такі як ліси, водні джерела, ґрунти тощо. Це дозволить оптимізувати їх використання, зменшуючи втрати та сприяючи сталому розвитку. В той же час, веб-додаток служитиме інструментом для збору та аналізу даних про зміни клімату, такі як температура, опади, рівень води тощо. Це допоможе у розробці ефективних стратегій адаптації до змін клімату та зменшення їх негативного впливу на природу.

Здійснено огляд засобів для розробки веб-додатків, такі як як JavaScript-фреймворки (наприклад, React, Angular, Vue.js), серверні технології (Node.js, Python, Ruby on Rails), бази даних (PostgreSQL, MongoDB) та інші. Виділено переваги та недоліки для кожної розглянутої технології.

РОЗДІЛ 2
ОСОБЛИВОСТІ РОЗРОБКИ ВЕБ-ДОДАТКА

Враховуючи природу проекту для моніторингу та управління природними ресурсами на основі ГІС, оптимальною є комбінація Google Earth Engine та JavaScript для інтерактивності та маніпуляції з геоданими, React.js / Vue.js / Angular для створення потужних користувацьких інтерфейсів, Mapbox API та PostGIS як базу даних. Кроки розробки відображено на рисунку 2.1.

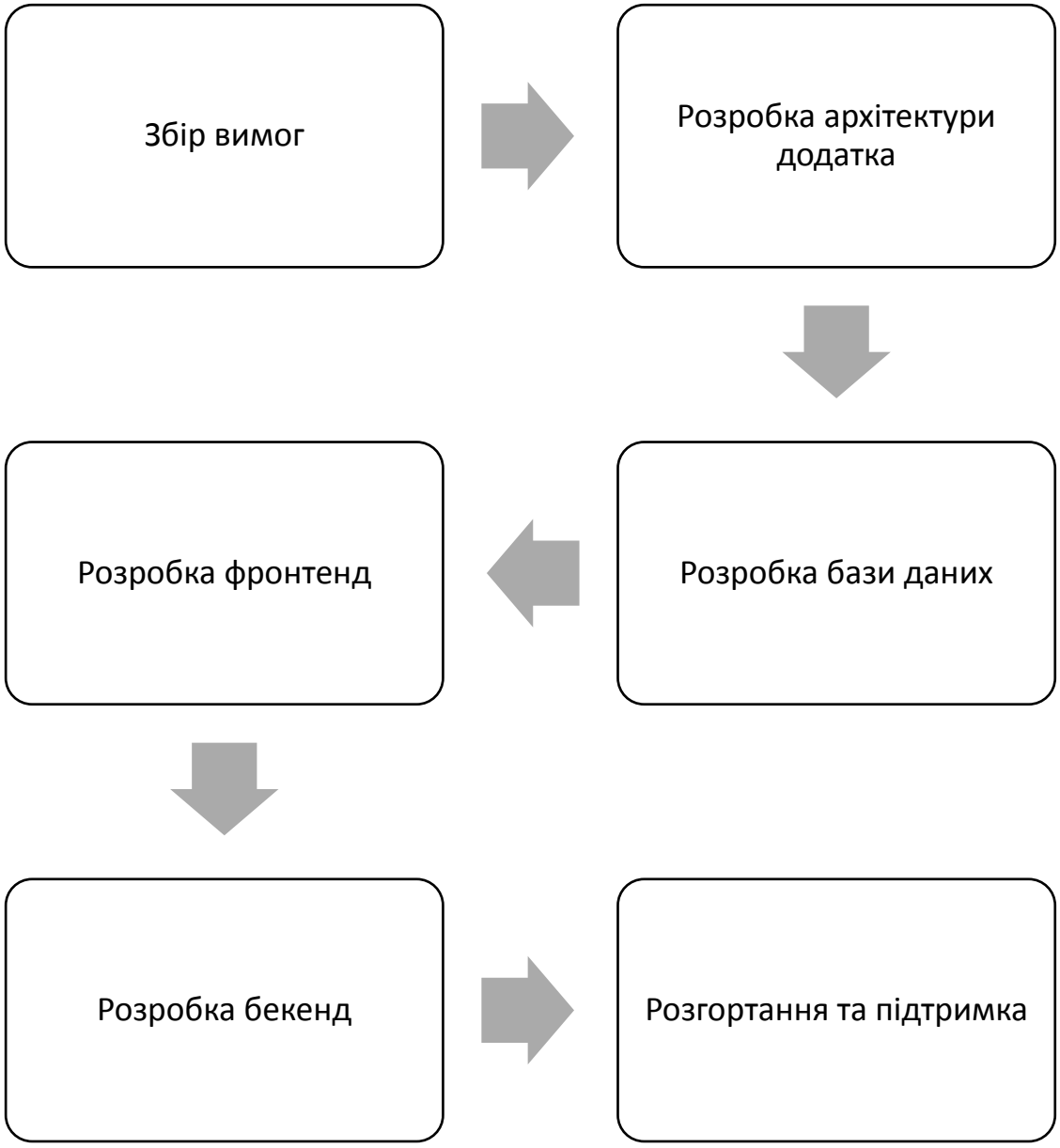

Рис. 2.1. Етапність розробки веб-додатка для моніторингу та управління природними ресурсами

2.1. Вимоги до веб-додатку

Першим кроком потрібно ретельно зібрати вимоги до додатка. Це включає уточнення функціональних вимог, технічних вимог та інтерфейсних вимог.

2.1.1. Функціональні вимоги

Веб-додаток має забезпечувати можливість моніторингу різних типів природних ресурсів, включаючи ліси, водні джерела, ґрунти, клімат, екосистеми та біорізноманіття. Користувач повинен мати змогу відображати дані про ресурси на мапі та отримувати актуальну інформацію про стан цих ресурсів.

Також веб-додаток повинен дозволяти користувачам додавати нові дані про ресурси, вносити зміни в існуючі дані та видаляти непотрібні записи. Користувач повинен мати можливість створювати та зберігати звіти про ресурси, включаючи аналіз динаміки змін, прогнозування тощо.

Розроблювана система повинна мати функціонал пошуку та фільтрації даних про ресурси для швидкого доступу до необхідної інформації. Користувач повинен мати можливість застосовувати різні фільтри, наприклад, за типом ресурсу, географічною областю, статусом тощо.

2.1.2. Технічні вимоги

Система повинна бути адаптована для роботи на різних типах пристроїв, включаючи мобільні телефони та планшети. Веб-додаток повинен мати респонсивний дизайн, що забезпечує коректне відображення та функціонування на різних екранах.

Має бути забезпечена масштабованість, що передбачає здатність обробляти великі обсяги даних та багатокористувацьке навантаження. Додаток повинен мати ефективну систему кешування та оптимізації запитів для забезпечення швидкої реакції на запити користувачів.

З метою забезпечення безпеки веб-додаток повинен мати механізми автентифікації та авторизації користувачів для захисту конфіденційності

даних та запобігання несанкціонованому доступу. Додаток повинен забезпечувати захист від атак, таких як сторонні SQL-запити, перехоплення сесій тощо.

2.1.3. Інтерфейсні вимоги

Користувачам повинно бути легко зрозуміти, як взаємодіяти з додатком та як шукати потрібну інформацію, що забезпечується зручним та інтуїтивно зрозумілим інтерфейсом. При цьому інтерфейс повинен мати чітку та логічну структуру, яка дозволяє ефективно виконувати різні завдання.

Веб-додаток повинен забезпечувати інтерактивні можливості, такі як можливість взаємодії з мапою, фільтрація даних за допомогою різних критеріїв тощо. Користувачам повинно бути надано можливість взаємодії з даними та виконання різних операцій без зайвих перешкод.

Для візуалізації роботи веб-додатка побудуємо діаграму прецедентів Use-case diagram (UML-діаграма варіантів використання) на рисунку 2.2.

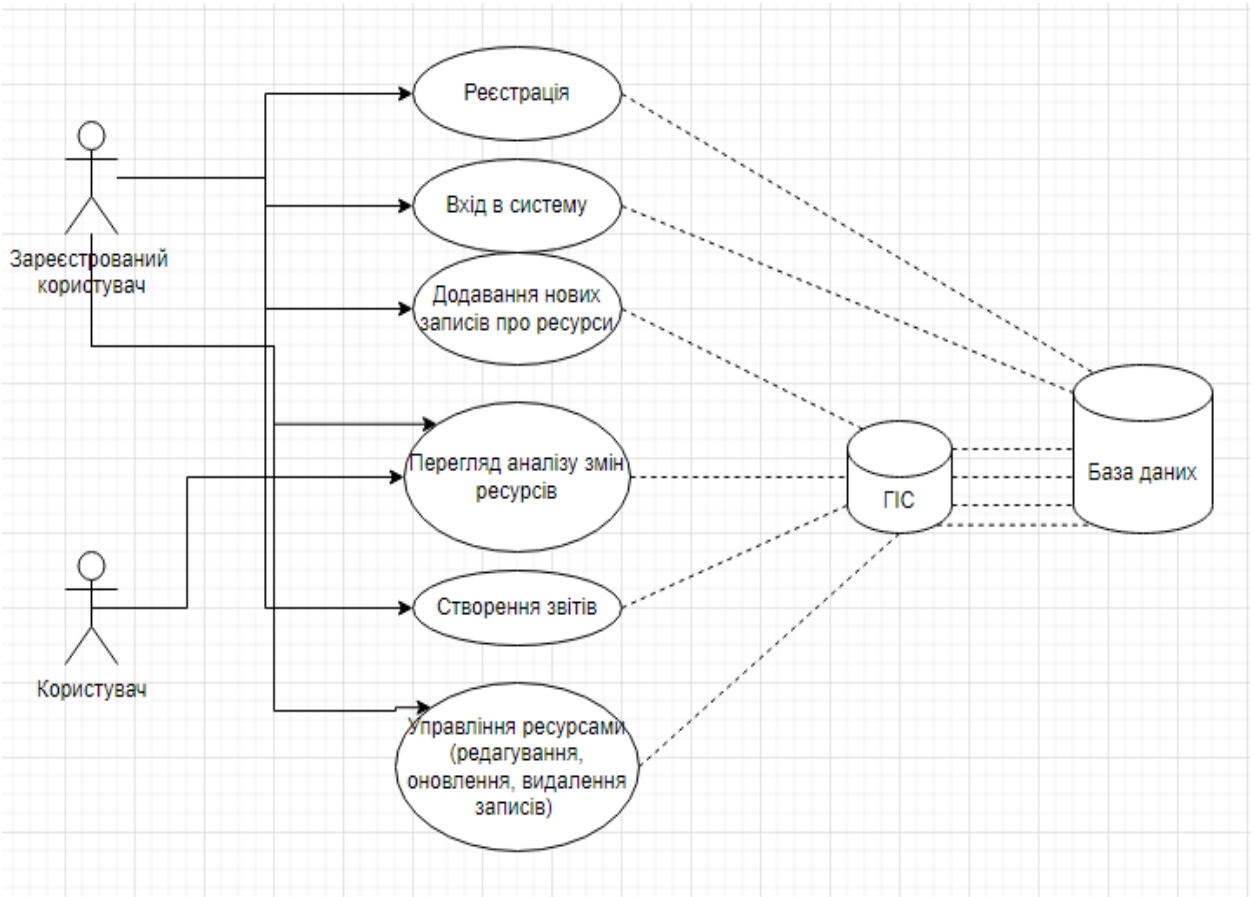

Рис. 2.2. UML-діаграма варіантів використання веб-додатка для моніторингу та управління природними ресурсами

В діаграмі на рис. 2.2 показані основні функції, доступні для користувачів веб-додатку. Актори – зареєстрований та незареєстрований користувачі. Зареєстрований користувач взаємодіє з системою шляхом реєстрації, виконання входу в систему, додаванням нового запису про ресурси, переглядом змін у ресурсах за певний період часу, створенням звітів на основі даних про ресурси та редагуванням, оновленням та видаленням записів про ресурси. Незареєстрований користувач може переглядати аналіз змін ресурсів.

Інформація про користувачів, така як ім'я, електронна пошта та пароль, буде збережена в базі даних. Це допомагає забезпечити безпеку та аутентифікацію користувачів. Дані про авторизованих користувачів перевіряються в базі даних для забезпечення доступу до системи. Після входу у систему користувач може бачити та маніпулювати геоданими, які зберігаються в базі даних. Нові дані про ресурси, такі як ліси, водні джерела тощо, будуть збережені в базі даних для подальшого використання та аналізу. Географічні координати та інші важливі атрибути ресурсу також будуть збережені в базі даних для подальшого відображення на карті в системі ГІС. Аналітичні дані про зміни у ресурсах будуть збережені в базі даних для подальшого відображення на графіках або діаграмах. Зміни у ресурсах відображатимуться на мапі в системі ГІС за допомогою геоданих, що також зберігаються в базі даних. Дані для створення звітів мають бути витягнуті з бази даних та оброблені для створення звітів. Результати аналізу ресурсів та інші важливі дані мають бути представлені у вигляді звітів та збережені в базі даних. Існуючі дані про ресурси мають бути змінені або видалені з бази даних в результаті дій користувача. Оновлені дані будуть відображені на мапі в системі ГІС для відображення оновленої інформації.

2.2. Архітектура веб-додатка

Для розробки веб-додатка з використанням Google Earth Engine, JavaScript, React.js / Vue.js / Angular, Mapbox API та PostGIS потрібно налаштувати середовище, яке включатиме наступне програмне забезпечення та сервіси:

1. Редактор коду або інтегроване середовище розробки (IDE) Visual Studio Code для написання, редагування та відлагодження коду.

2. Node.js для виконання JavaScript поза браузером, а також Node Package Manager для управління залежностями проєкту та установку пакетів.

3. React.js для розробки користувацького інтерфейсу веб-додатка.

4. Google Earth Engine JavaScript API для взаємодії з геоданими та виконання геопроцесів у веб-додатку.

5. Mapbox API для відображення геоданих на мапі (інструменти для створення та керування інтерактивною картографією).

6. Розширення бази даних PostgreSQL для обробки геоданих – PostGIS.

7. База даних PostgreSQL для зберігання геоданих та інших даних веб-додатку.

8. Середовище для розгортання веб-додатка.

Розробка архітектури веб-додатка для моніторингу та управління природними ресурсами проводиться на основі патерну MVC (Model-View-Controller) (рис. 2.3).

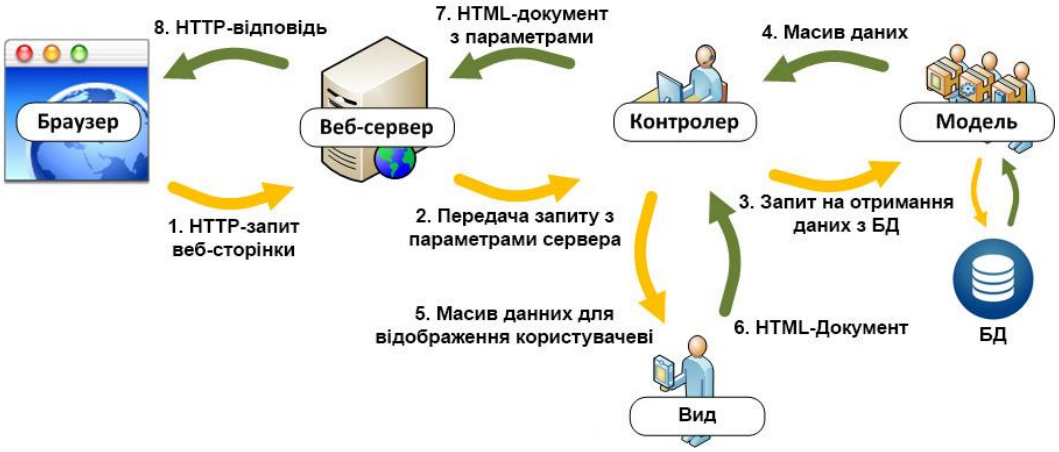

Рис. 2.3. Архітектура веб-додатка

Модель відповідає за управління даними та бізнес-логікою додатка. Вона відділяє дані від представлення та управляє всіма операціями з даними.

Модель включає серверну сторону (Back-end) додатка, яка забезпечує доступ до бази даних, взаємодію з іншими API та виконання бізнес-логіки. В даній дипломній роботі буде показано розробку серверної частини за допомогою Node.js, Express.js.

Модель також включає схеми даних, які представляють різні типи ресурсів, їх відносини та властивості. Вона використовує систему керування базами даних (в даному випадку PostgreSQL) для зберігання та управління даними.

Представлення (View) відповідає за відображення даних користувачеві та інтерфейс взаємодії. Воно отримує дані від моделі та відображає їх користувачеві у зручному для сприйняття вигляді.

В представлення включена клієнтська частина (Front-end), яка відповідає за відображення інтерфейсу користувача. В даній дипломній роботі буде показана реалізація за допомогою React.js.

Контролер (Controller) відповідає за приймання вхідних запитів від користувача, обробку їх та ініціювання відповідних дій, таких як виклик методів моделі або відображення відповідного представлення. Контролер включає маршрутизатор, який визначає, який контролер або дія повинна бути викликана для кожної вхідної URL-адреси (реалізація здійснюється за допомогою бібліотек). Для забезпечення зв’язку між компонентами контролер отримує запит від користувача та викликає відповідний метод моделі для обробки цього запиту. Після обробки запиту модель повертає дані контролеру, а контролер вибирає відповідне представлення та передає дані для відображення користувачеві.

На даному етапі також можна реалізувати API для взаємодії з додатком через HTTP-запити, що дозволить використовувати додаток як із клієнтського, так і з серверного боку.

2.3. Розробка бази даних

В даному проєкті передбачено базу даних для зберігання геоданих та їх обробки на бекенді. Це означає, що потрібно налаштувати з'єднання з базою даних, обробку фільтрів та звітів, а також реалізацію компонентів для відображення даних на мапі.

Базу даних створюємо за допомогою PostgreSQL. Вибір PostgreSQL обумовлено тим, що він є більш повнофункціональним та строгим щодо дотримання стандартів SQL, що робить його більш гнучким у використанні для складних операцій з базою даних. Також PostgreSQL має розширення PostGIS, яке надає повноцінну підтримку геоданих і геопросторових операцій, що робить його більш підходящим для даного проєкту.

Створюємо таблиці resources, users, reports та resource_history (рис. 2.4).

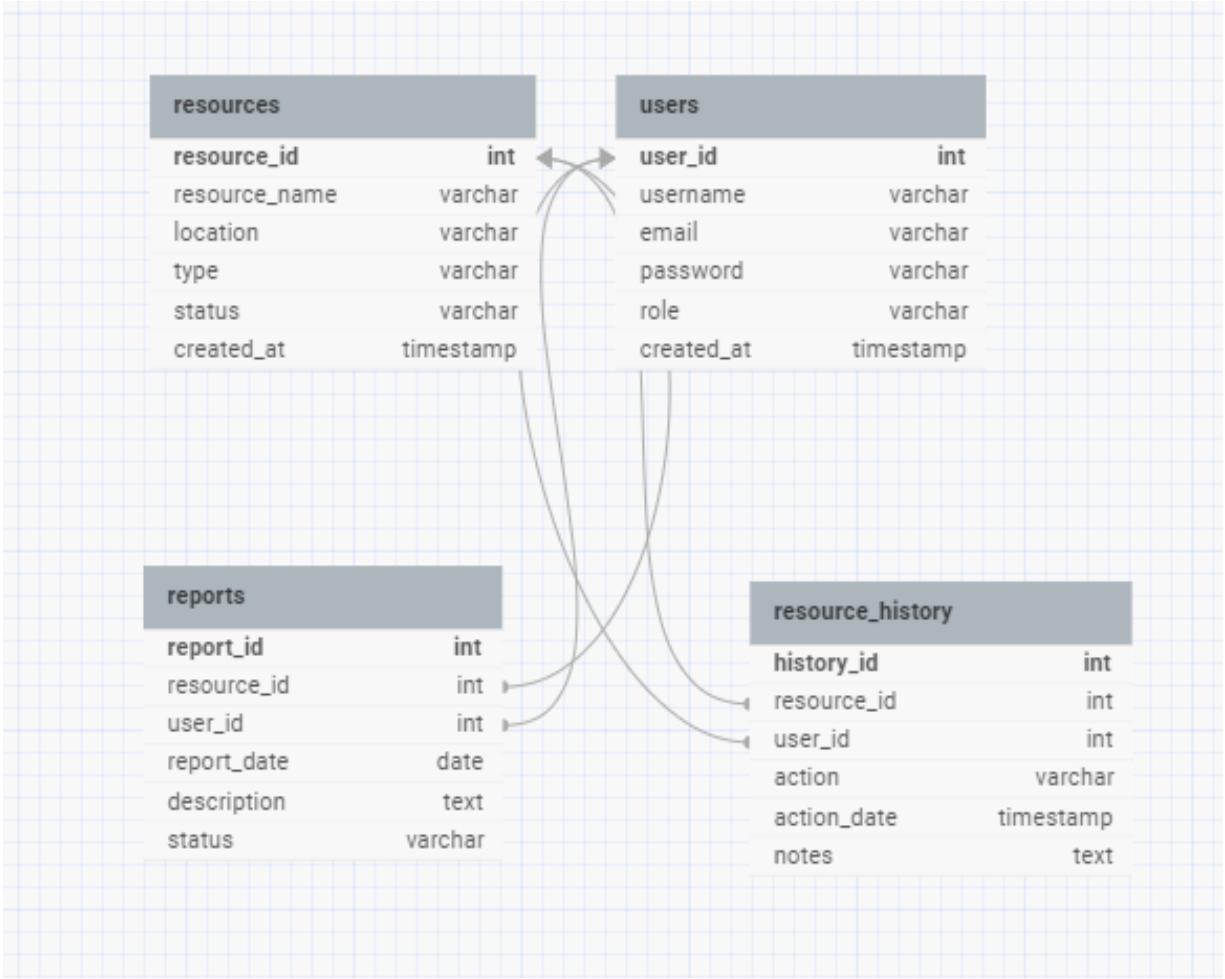

Рис. 2.4. ER діаграма бази даних

В базі даних, представленій на рис. 2.4, буде зберігатись інформація про ресурси, користувачів, звіти про ресурси та історію змін у ресурсах для подальшого аналізу та використання.

Таблиця resources призначена для зберігання інформації про ресурси. У ній є наступні стовпці:

1. resource_id: унікальний ідентифікатор ресурсу.
2. resource_name: назва ресурсу.
3. location: місце розташування ресурсу.
4. type: тип ресурсу.
5. status: статус ресурсу.
6. created_at: дата та час створення ресурсу.

Таблиця users містить інформацію про користувачів системи. У ній є такі стовпці:

1. user_id: унікальний ідентифікатор користувача.
2. username: ім'я користувача.
3. email: адреса електронної пошти користувача.
4. password: пароль користувача.
5. role: роль користувача.
6. created_at: дата та час створення користувача.

У таблиці reports зберігається інформація про звіти, які створюються користувачами про ресурси. Її структура включає:

1. report_id: унікальний ідентифікатор звіту.
2. resource_id: зовнішній ключ, що посилається на resources.resource_id.
3. user_id: зовнішній ключ, що посилається на users.user_id.
4. report_date: дата звіту.
5. description: опис звіту.
6. status: статус звіту.

Таблиця resource_history призначена для зберігання історії змін у ресурсів. Вона містить:

1. history_id: унікальний ідентифікатор історії ресурсу.

2. resource_id: зовнішній ключ, що посилається на resources.resource_id.

3. user_id: зовнішній ключ, що посилається на users.user_id.

4. action: дія, яка була виконана над ресурсом.

5. action_date: дата та час виконання дії.

6. notes: додаткові примітки щодо дії.

У цих описах використовуються стандартні типи даних бази даних, такі як INT, VARCHAR, TIMESTAMP, DATE та TEXT, разом із обмеженнями, такими як PRIMARY KEY та зовнішні ключі (FOREIGN KEY), щоб забезпечити цілісність даних та зв'язки між таблицями.

2.4. Розробка frontend та backend

Розробка frontend (фронтенд) проводиться за допомогою React.js, оскільки React.js є одним з найпопулярніших JavaScript-фреймворків для створення користувацьких інтерфейсів.

Імпорт необхідних компонентів та служби:

import React, { useState, useEffect } from 'react';

import Map from './components/Map';

import ResourceList from './components/ResourceList';

import ReportGenerator from './components/ReportGenerator';

import FilterPanel from './components/FilterPanel';

import ApiService from './services/ApiService';

Повний лістинг наведено в додатку А.

Після цього переходимо до створення компонентів «Map», «ResourceList», «ReportGenerator» та «FilterPanel».

Компонент «Map» буде відповідати за відображення мапи та ресурсів на ній. Буде використовуватись бібліотека React для створення компонента, який відображає мапу з ресурсами (рис. 2.5).

```
import React from 'react';
import { MapboxMap } from './MapboxMap';

const Map = ({ resources, selectedResource }) => {
  return (
    <div className="map-container">
      <MapboxMap resources={resources} selectedResource={selectedResource} />
    </div>
  );
};

export default Map;
```

Рис. 2.5. Програмний код компонента «Map»

Компонент «ResourceList» буде відображати список ресурсів, якими можна управляти (рис. 2.6). Для реалізації даного компоненту імпортується бібліотека React, яка дозволяє використовувати React-специфічні функції та компоненти у файлі. Після чого оголошується функціональний компонент ResourceList, в який передаються два параметри resources і onSelectResource, які використовуються для відображення списку ресурсів та обробки вибору ресурсу відповідно. Використовується метод map, щоб пройтися по кожному елементу масиву resources і створити відповідний елемент для кожного ресурсу. Кожен ресурс відображається у вигляді , де key встановлюється на унікальне значення id ресурсу, а також додається обробник подій onClick, який викликає функцію onSelectResource при кліці на ресурс. Назва ресурсу відображається всередині елементу.

```
import React from 'react';

const ResourceList = ({ resources, onSelectResource }) => {
  return (
    <div className="resource-list">
      {resources.map((resource) => (
        <div key={resource.id} onClick={() => onSelectResource(resource)}>
          {resource.name}
        </div>
      ))}
    </div>
  );
};

export default ResourceList;
```

Рис. 2.6. Програмний код компонента «ResourceList»

Компонент «ReportGenerator» призначений для генерації звітів для веб-додатка (рис. 2.7). Ця конструкція оголошує функціональний компонент ReportGenerator. У компонент передається один параметр selectedResource, який містить інформацію про обраний ресурс, для якого генерується звіт. В коді також відображено функцію generateReport, яка буде викликатися при натисканні на кнопку «Generate Report».

```
import React from 'react';

const ReportGenerator = ({ selectedResource }) => {
  const generateReport = () => {
    const report = generateReportFunction(selectedResource);
  };

  return (
    <div className="report-generator">
      {selectedResource && (
        <div>
          <h2>Report Generator</h2>
          <button onClick={generateReport}>Generate Report</button>
        </div>
      )}
    </div>
  );
};

export default ReportGenerator;
```

Рис. 2.7. Програмний код компонента «ReportGenerator»

Компонент «FilterPanel» відображає панель фільтрації та надає можливість користувачеві вибрати фільтри та застосувати їх (рис. 2.8).

У компонент передається один параметр onFilter, який є функцією зовнішнього коду і викликається для застосування фільтрів. Використовуємо хук useState для створення стану filters, який буде містити об'єкт з встановленими фільтрами. В результаті відбувається відображення панелі фільтрації. Вона містить форму, яка містить елементи фільтрації, і кнопку «Фільтрувати», що викликає функцію handleSubmit при натисканні.

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

const FilterPanel = ({ onFilter }) => {
  const [filters, setFilters] = useState({});

  const handleFilterChange = (e) => {
    const { name, value } = e.target;
    setFilters((prevFilters) => ({ ...prevFilters, [name]: value }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    onFilter(filters);
  };

  return (
    <div className="filter-panel">
      <form onSubmit={handleSubmit}>
        <button type="submit">Фільтрувати</button>
      </form>
    </div>
  );
};

export default FilterPanel;
```

Рис. 2.8. Програмний код компонента «FilterPanel»

Наступним кроком налаштовуємо комунікацію між компонентами за допомогою передачі пропсів (props) від батьківського компонента до дочірніх. Батьківський компонент, якій з’єднує компоненти «Map», «ResourceList», «ReportGenerator» та «FilterPanel», відображено на рисунку 2.9. Програмний код для кожного з цих компонентів наведено в додатку Б.

```
import React, { useState } from 'react';
import Map from './Map';
import ResourceList from './ResourceList';
import ReportGenerator from './ReportGenerator';
import FilterPanel from './FilterPanel';

const MainComponent = () => {
  const [selectedResource, setSelectedResource] = useState(null);
  const [filteredData, setFilteredData] = useState([]);

  const handleResourceSelect = (resource) => {
    setSelectedResource(resource.name);
  };

  const handleFilterChange = (value) => {
    setFilteredData([...]);
  };

  return (
    <div>
      <Map selectedResource={selectedResource} />
      <ResourceList resources={filteredData} onResourceSelect={handleResourceSelect} />
      <ReportGenerator selectedResource={selectedResource} />
      <FilterPanel onFilterChange={handleFilterChange} />
    </div>
  );
};

export default MainComponent;
```

Рис. 2.9. Програмний код батьківського компонента, який з'єднує всі компоненти

Після цього реалізовується backend (бекенд) за допомогою Node.js, Express.js, а також ApiService (клієнтська сторона). Лістинг коду наведено в додатку В.

Для реалізації бекенду створюється простий веб-сервер за допомогою фреймворка Express.js для веб-розробки на Node.js. Для цього підключаємо фреймворк Express.js до проєкту. Він використовує require() для завантаження модуля Express з пакету, який встановлений у проєкті. Створюємо екземпляр додатку Express, який буде використовуватися для налаштування маршрутів та обробки запитів. Обов’язково потрібно становити GET-маршрут /api/resources, який обробляється колбек-функцією. Наприкінці викликається метод listen(), який запускає сервер на вказаному порті. Після запуску сервера виводиться повідомлення про те, що сервер запущений та доступний за адресою http://localhost:3000.

Для реалізації API використовуємо статичний метод getResources. Клас ApiService оголошується за допомогою ключового слова class. Цей клас призначений для забезпечення доступу до API сервера. Обов’язково виконується запит до сервера за допомогою функції fetch(). Виконується GET-запит за адресою /api/resources. Після отримання відповіді в форматі JSON, вона розкодовується за допомогою методу json().

Також необхідно налаштувати з'єднання з базою даних за допомогою Node.js, Express.js, PostGIS. Для цього створюється сервер за допомогою Express.js. Також використовується пакет pg для з'єднання з базою даних PostgreSQL через об'єкт Pool, який використовується для підключення до бази даних PostgreSQL. Налаштовується роутер для отримання даних про ресурси з бази даних. Цей роутер слухає GET-запити на шлях /api/resources. При отриманні такого запиту він виконує запит до бази даних за допомогою pool, отримує дані та відправляє їх у відповідь у форматі JSON. У випадку помилки він надсилає статус 500 та повідомлення про внутрішню помилку сервера. Повний лістинг розміщено в додатку Г.

Логіку обробки фільтрів та звітів на бекенді реалізуємо шляхом визначення маршрута /api/filter, який приймає POST-запити (див. додаток Д). Запит POST передає дані у тілі запиту (req.body). Тут ми витягуємо дані з тіла запиту, конкретно об'єкт filters. Забезпечуємо виконання SQL-запиту до бази даних. Для цього використовуємо параметризований запит, передаючи значення type та region з об'єкта filters. В результаті виконання запиту отримуємо дані, які задовольняють умовам фільтрації. Після успішного виконання запиту відправляємо відфільтровані дані у форматі JSON у відповідь. Якщо під час виконання запиту виникає помилка, вона обробляється у блоку catch. У цьому випадку відправляється відповідь зі статусом 500 та повідомленням про внутрішню помилку сервера.

2.5. Інтерактивність з геоданими

Для реалізації інтерактивності з геоданими на мапі за допомогою Google Earth Engine та JavaScript для початку потрібно налаштувати середовище для роботи з Google Earth Engine та інтегрувати його з веб-додатком.

Для налаштування Google Earth Engine необхідно зареєструватись на Google Earth Engine в якості розробника (рис. 2.10).

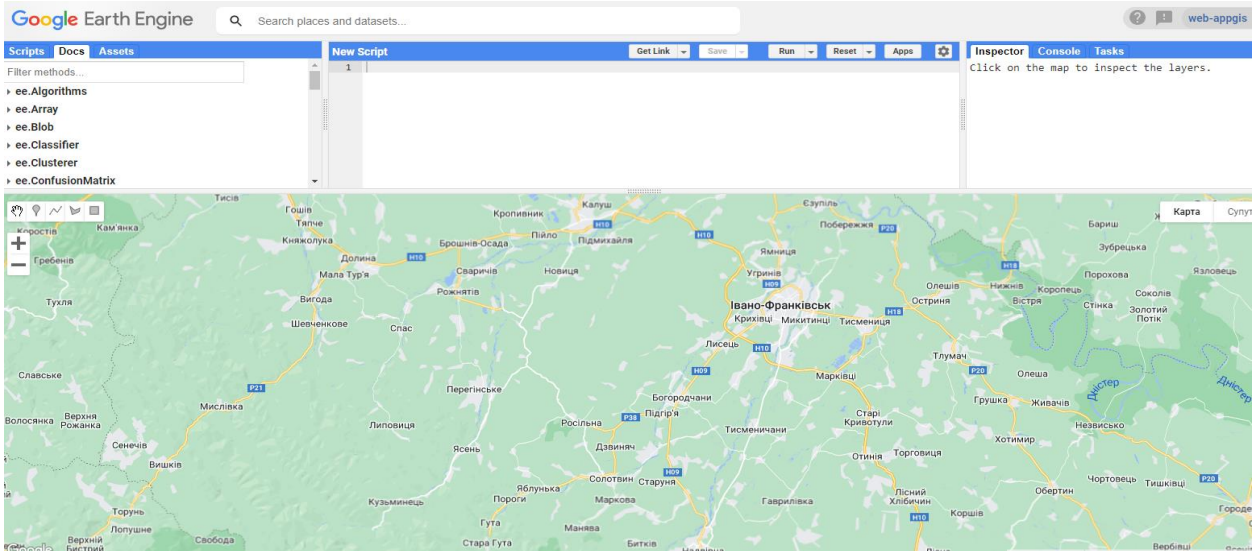

Рис. 2.10. Середовище розробки в Google Earth Engine

Після цього отримуємо доступ до Google Earth Engine API та налаштовуємо середовище розробки. Для інтеграції з веб-додатком інсталюємо необхідні пакети та бібліотеки для роботи з Google Earth Engine, а також підключаємо Google Earth Engine API до веб-додатка за допомогою ключа API. Подальша реалізація інтерактивності відбувається в декілька етапів.

Перший етап – відображення шарів ресурсів. Алгоритм виконання даного етапу, закладений в програмному коді представлено на рисунку 2.11.

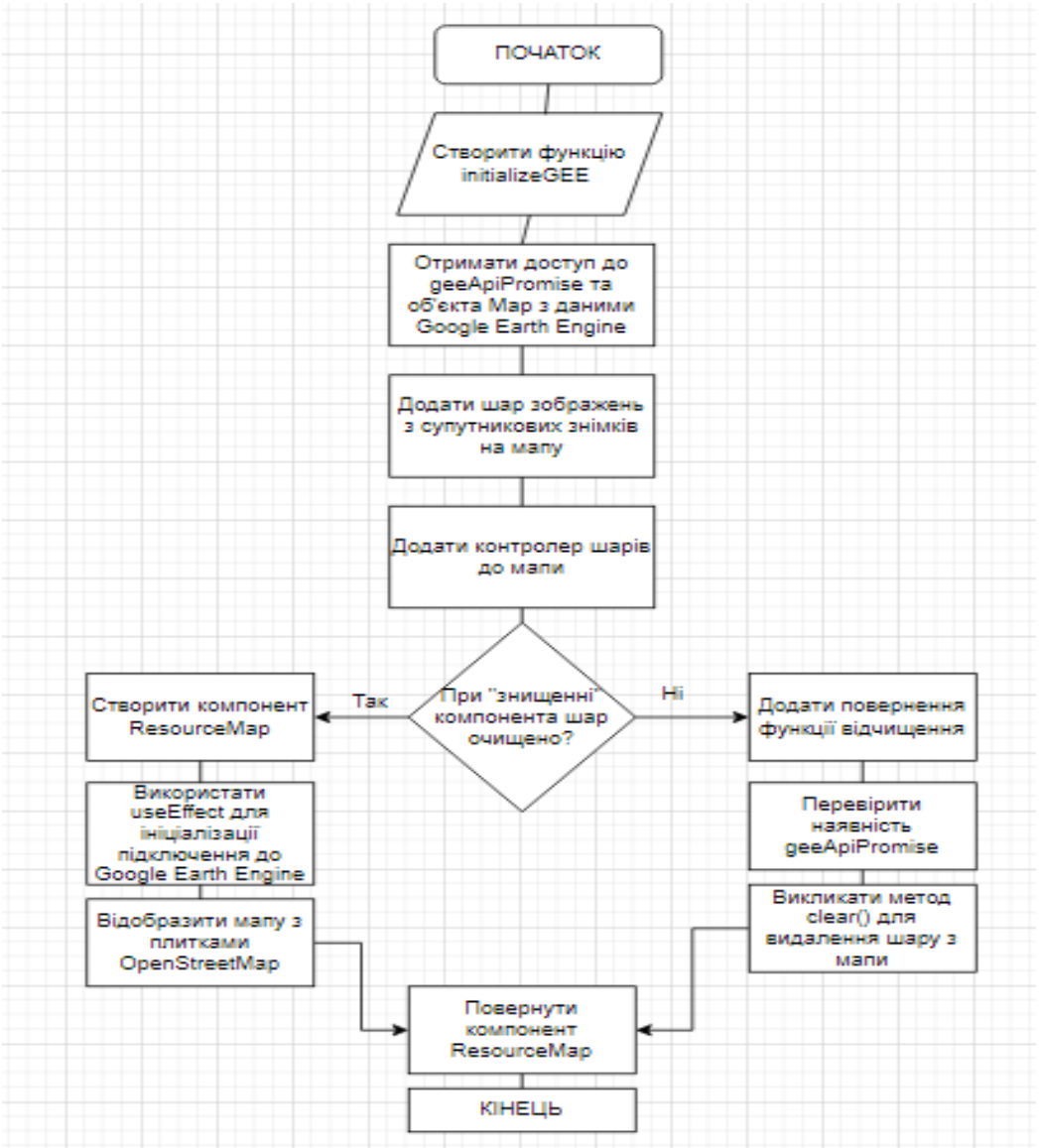

Рис. 2.11. Алгоритм розробки відображення шарів ресурсів

На даному етапі використовуємо Google Earth Engine для завантаження та відображення шарів різних ресурсів на мапі. Наприклад, можна відобразити

шари зображень з супутникових знімків або дані про використання землі. Для цього потрібно спочатку завантажити дані з GEE та потім використовувати їх у веб-додатку. Програмний код включає в себе ініціалізацію підключення до Google Earth Engine, очищення шару при «знищенні» компонента, а також відображення мапи. Для цього використовуємо бібліотеку Leaflet для відображення мапи та підключаємо до неї шари зображень з супутникових знімків з Google Earth Engine (лістинг наведено в додатку Е).

Другий етап – навігація по мапі, алгоритм виконання представлено на рисунку 2.12.

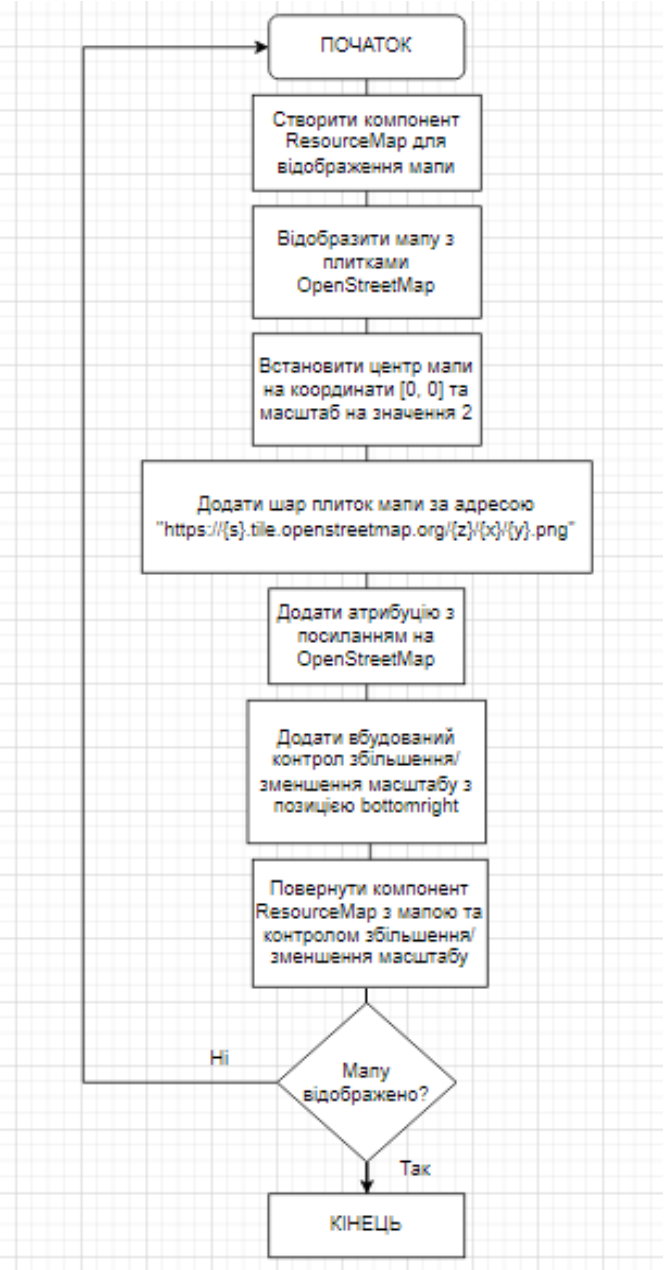

Рис. 2.12. Алгоритм розробки навігації по мапі

На даному етапі додаємо функціонал навігації по мапі, такий як збільшення та зменшення за допомогою вбудованих контролів Leaflet. Додатково використовуємо компонент ZoomControl з бібліотеки Leaflet, щоб додати вбудований контрол збільшення/зменшення масштабу на мапі. Повний лістинг наведено в додатку Е.

Третій етап – вибір конкретних областей, алгоритм, закладений логікою програмного коду представлено на рисунку 2.13.

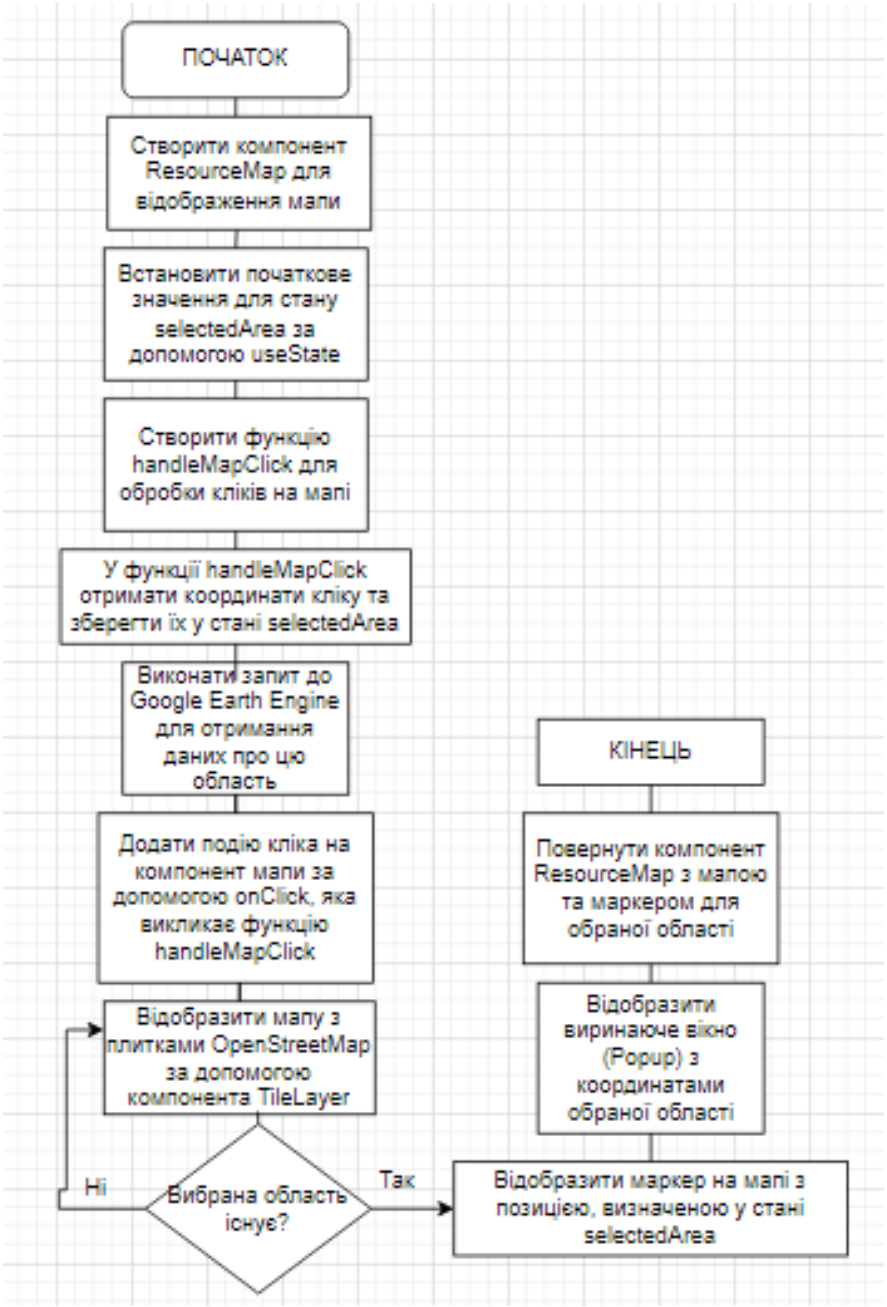

Рис. 2.13. Алгоритм розробки вибору конкретних областей

Слід зазначити, що в Google Earth Engine можна займатись розробкою скриптів, які потім будуть додані до веб-додатку за допомогою Google Earth Engine API. Такі скрипти можуть виконувати різні завдання з аналізу та обробки геоданих. Приклад розробленого скрипту для обробки зображень зі супутників показано на рисунку 2.14.

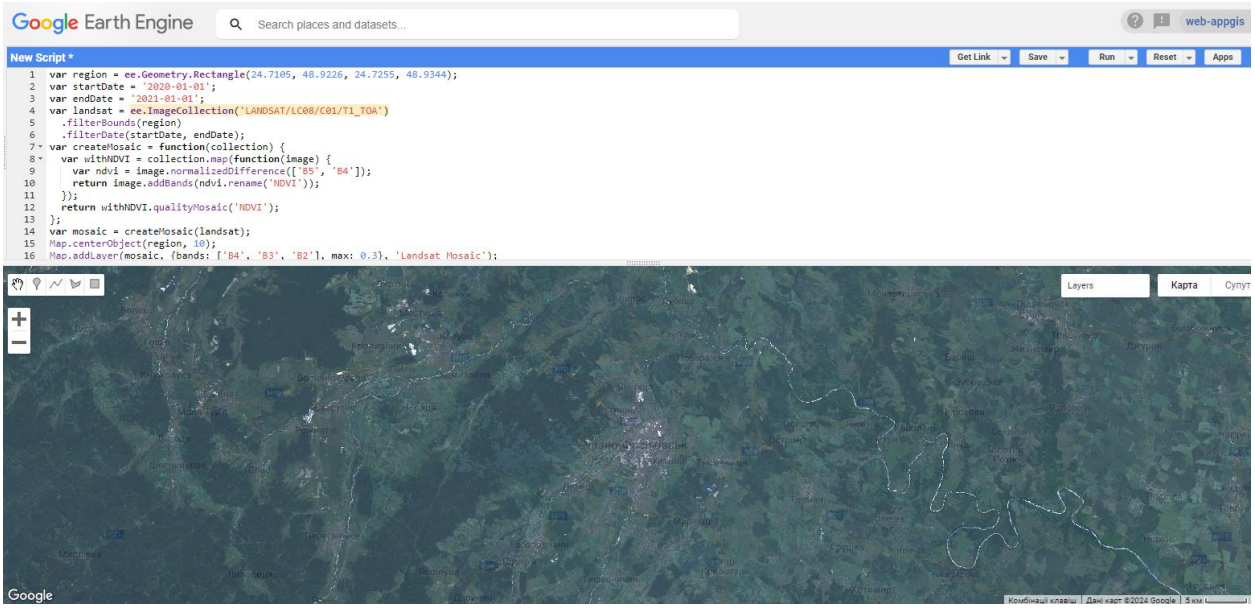

Рис. 2.14. Реалізація скрипту для обробки зображень зі супутників

На рис. 2.14 показано код скрипту та результат його впровадження в Івано-Франківській області (використано координати 24.7105, 48.9226, 24.7255, 48.9344 для центральної частини міста Івано-Франківська) – створення додаткового «пласту» на карті, що підтягує зображення зі супутників.

На даному етапі додаємо можливість вибору конкретних областей на мапі, наприклад, за допомогою маркерів, полігонів або прямокутників. Після вибору області виконується запит до Google Earth Engine для отримання даних про цю область та відображення їх на мапі. Для цього використовуємо компонент Marker з бібліотеки Leaflet для відображення маркера на мапі, коли користувач клікає на неї. Після вибору буде виконано запит до Google Earth Engine для отримання даних про цю область. Результати запиту можна

відобразити на мапі або використати у веб-додатку для подальшої обробки. Повний лістинг наведено в додатку Е.

2.6. Розгортання та підтримка

Після успішного тестування розробленого веб-додатку і перед його розгортанням на веб-сервері, виконуються наступні кроки:

1. Встановлення серверного середовища Node.js на своєму сервері, а також встановлення PostgreSQL для використання як бази даних (рис. 2.15).

```
sudo apt update
sudo apt install nodejs npm

sudo apt update
sudo apt install postgresql postgresql-contrib

sudo systemctl start postgresql
sudo systemctl enable postgresql
sudo su - postgres
initdb -D /var/lib/postgresql/data
exit

sudo -u postgres psql
\password postgres

sudo -u postgres createuser --interactive
sudo -u postgres createdb mydatabase

npm install express nodemon pg
```

Рис. 2.15. Налаштування серверного середовища Node.js на Debian

2. Налаштування серверної частини веб-додатка (ініціалізація нового проєкту Node.js та налаштування залежностей; встановлення пакетів для взаємодії з базою даних PostgreSQL (в нашому випадку, pg); створення серверної структури проекту, включаючи файли маршрутизації, контролери, сервіси тощо). Лістинг коду розміщено в додатку Ж, частина коду представлена на рисунку 2.16.

```
router.get('/users', getAllUsers);
router.get('/users/:id', getUserById);
router.post('/users', createUser);
router.put('/users/:id', updateUser);
router.delete('/users/:id', deleteUser);

module.exports = router;

const { Pool } = require('pg');

const pool = new Pool({
  user: 'ChekanV',
  host: '3000',
  database: 'ChekanGIS',
  password: '1111',
  port: 5432,
});

const getAllUsers = async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM users');
```

Рис. 2.16. Частина лістингу налаштування серверної частини веб-додатка

3. Підключення до бази даних (налаштування підключення до бази даних PostgreSQL у своєму серверному додатку, використовуючи налаштування, отримані від хостинг-провайдер; встановлення драйвера PostgreSQL для Node.js і використання його для взаємодії з базою даних). Лістинг коду розміщено в додатку Ж, частина коду представлена на рис. 2.17.

```
pool.connect((err, client, release) => {
  if (err) {
    return console.error('Error acquiring client', err.stack);
  }
  console.log('Connected to PostgreSQL database');
  client.release();
});

module.exports = pool;

pool.query('SELECT NOW()', (err, res) => {
  if (err) {
    console.error('Error executing query', err.stack);
  } else {
    console.log('Current date in the database:', res.rows[0].now);
  }
});
```

Рис. 2.17. Код перевірки підключення до бази даних PostgreSQL

4. Наштування API для взаємодії з даними (створення маршрутів для обробки запитів клієнта; створення контролерів, які обробляють ці запити та взаємодіють з базою даних; забезпечення API для отримання, оновлення, створення та видалення даних в базі даних). Код структури для створення API з CRUD-операціями для взаємодії з базою даних PostgreSQL за допомогою Node.js та Express.js наведено в додатку Ж.

5. Настройка сервера та розгортання додатка (налаштування веб-сервера для обробки запитів; налаштування середовища виконання для сервера; розгортання серверного додатку на вибраному хостинг-провайдері). Код настройки сервера та розгортання веб-додатка за допомогою Express.js показано на рисунку 2.18.

```
const express = require('express');
const bodyParser = require('body-parser');
const routes = require('./routes');

const app = express();
const port = process.env.PORT || 3000;
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use('/api', routes);

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Рис. 2.18. Настройка сервера за допомогою Express.js

6. Тестування та налагодження (перевірка правильності роботи веб-додатку, виправлення помилок; налаштування журналювання для відстеження діяльність сервера та виявлення проблем).

2.7. Оцінка ефективності розробки

Проведене дослідження теоретичних джерел та існуючих програмних рішень дає змогу стверджувати, що ефективність роботи веб-додатка вимірюється його продуктивністю та веб-оптимізацією.

З огляду на те, що веб-додаток для моніторингу та управління природними ресурсами на основі ГІС передбачає активне навантаження від користувачів, оцінка ефективності розробленого веб-додатку має включати декілька аспектів, які оцінюються як технічно, так і відносно користувачів.

Тому, доцільним для оцінки ефективності вбачаємо впровадження розробленої методики порівняння різних метрик, які визначають якість та продуктивність роботи веб-додатку. Пропонується використовувати наступні метрики для порівняння ефективності роботи веб-додатків:

1. Час відгуку сервера (ЧВС) – час, який потрібний серверу для обробки запиту від клієнта. Чим менше цей час, тим ефективніше працює сервер. Математично це можна виміряти як середнє значення часу відгуку сервера за певний період часу. Формула для розрахунку:

$$ЧВС = T_{end} – T_{start}, \quad (2.1)$$

де T_{end} – час, коли сервер закінчив відповідь на запит, T_{start} – час, коли сервер отримав запит.

2. Час завантаження сторінок (ЧЗС) – час, який потрібен для завантаження сторінки веб-додатку користувачем. Швидке завантаження покращує користувацький досвід. Цей час можна виміряти як середнє значення часу завантаження сторінки для різних сторінок веб-додатку. Формула для розрахунку:

$$ЧЗС = T_{load} – T_{navig}, \quad (2.2)$$

де T_{load} – час завантаження сторінки, T_{navig} – час початку навігації на сторінку.

3. Продуктивність за розміром сторінок (ПРС) – обсяг даних, які передаються користувачеві для завантаження сторінки. Менший розмір сторінок дозволяє ефективніше використовувати пропускну здатність мережі. Можна виміряти середній розмір сторінок додатку та встановити цільові значення для оптимізації. Формула для розрахунку:

$$ПРС = \frac{РС}{ЧЗС}, \quad (2.3)$$

де РС – розмір сторінки в байтах, ЧЗС – час завантаження сторінок.

4. Використання кешу (ВК). Ефективне використання кешу може зменшити час завантаження сторінок та обсяг мережевого трафіку. Показник можна отримати шляхом виміру відсотку запитів, які успішно обслуговуються з кешу, а також часу існування кешованих даних. Формула для розрахунку:

$$ВК = \frac{КЗв}{КЗз}, \quad (2.4)$$

де КЗв – кількість запитів, що були видачами, КЗз – загальна кількість запитів.

5. Час виконання клієнтського коду (ЧВКК) – це час, який потрібний браузеру для виконання JavaScript-коду на стороні клієнта. Можна виміряти середній час виконання клієнтського коду та шукати можливості для його оптимізації. Формула для розрахунку:

$$ЧВКК = T_{end} – T_{start}, \quad (2.5)$$

де Tend – час, коли весь клієнтський код виконався, Tstart – час початку виконання клієнтського коду.

6. Частота помилок (ЧП) вимірюється шляхом визначення кількості помилок, які виникають під час роботи додатку. Висока частота помилок може вказувати на проблеми з якістю коду. Можна виміряти кількість помилок на певну кількість запитів або користувачів. Формула для розрахунку:

$$ЧП = \frac{КП}{КЗз}, \quad (2.6)$$

де КП – кількість помилок, КЗз – загальна кількість запитів.

Середні значення цих метрик для веб-додатка для моніторингу та управління природними ресурсами на основі ГІС розміщено на рисунку 2.19.

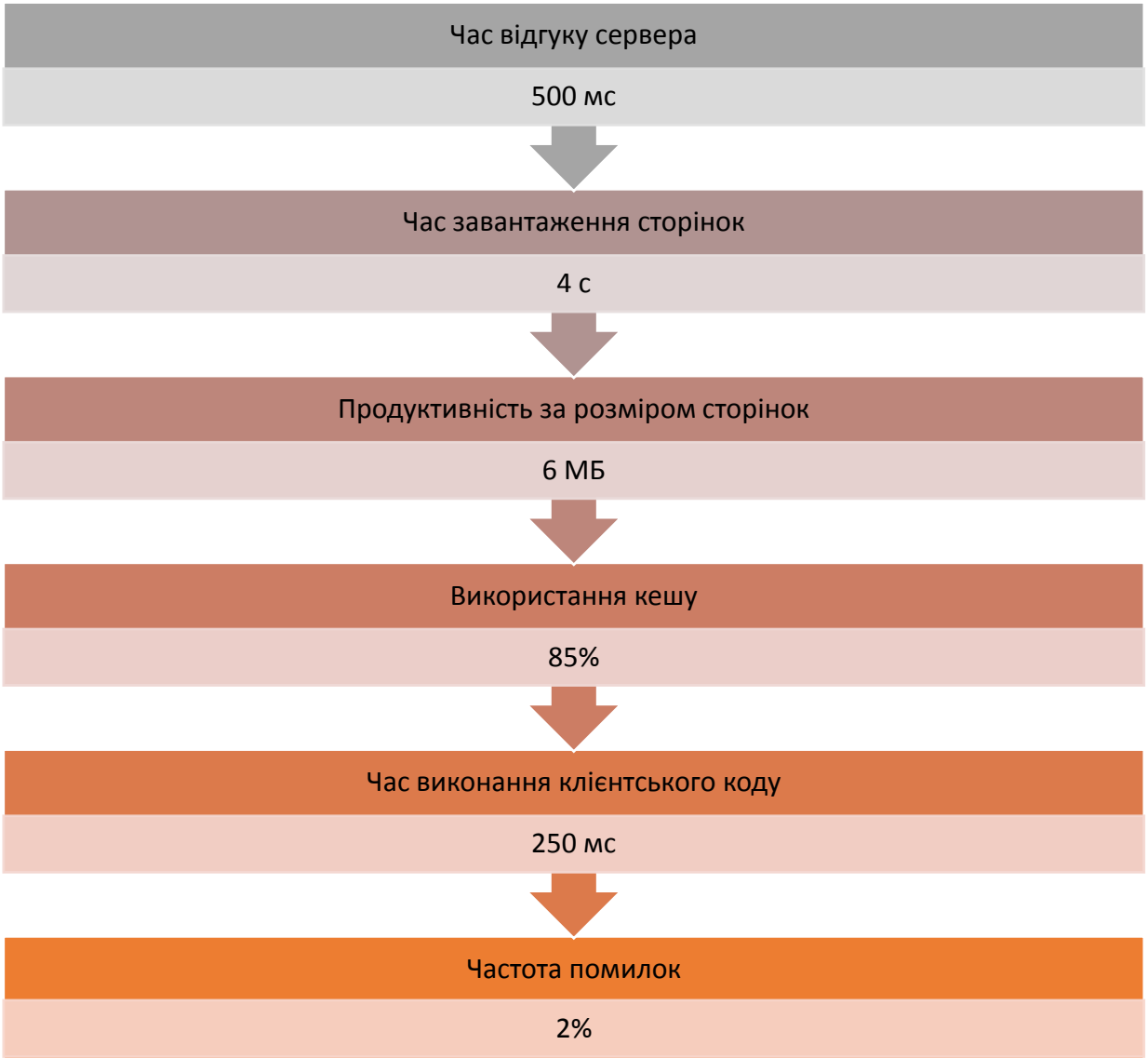

Рис. 2.19. Середні значення роботи веб-додатка для моніторингу та управління природними ресурсами на основі ГІС

Порівняння метрик з рис. 2.19 з існуючими програмними рішеннями, які розглядались в роботі (див. підрозділ 1.2) наведено в таблиці 2.1.

Таблиця 2.1

Метрики роботи веб-додатка для моніторингу та управління природними ресурсами на основі ГІС

Метрика	Час відгуку сервера	Час завантаження сторінок	Продуктивність за розміром сторінок	Використання кешу	Час виконання клієнтського коду	Частота помилок
Веб-додаток власної розробки	500 мс	4 с	6 MB	80%	250 мс	2%
Global Forest Watch	300 мс	3 с	5 MB	70%	150 мс	0,5%
Google Earth Engine	300 мс	3 с	5 MB	70%	150 мс	0,5%
USGS Earth Explorer	400 мс	4 с	4 MB	75%	180 мс	1%
Global Fishing Watch	500 мс	5 с	7 MB	80%	250 мс	2%
Sentinel Hub (by Planet Labs)	400 мс	4 с	6 MB	75%	180 мс	1%
Земельний кадастр України	400 мс	3 с	4 MB	75%	180 мс	1%
Геопортал «Ліси України»	500 мс	4 с	4 MB	75%	180 мс	2%

З табл. 2.1 бачимо, що час відгуку сервера для всіх додатків знаходиться у межах від 300 мс до 500 мс, при цьому веб-додаток власної розробки та

геопортал «Ліси України» мають час відгуку 500 мс, що трохи вище середнього. Розглянуті веб-додатки мають час завантаження сторінок у межах від 3 до 5 секунд, тому час завантаження 4 секунди є середнім значенням. Розмір сторінок для всіх веб-додатків знаходиться у межах від 4 до 7 МБ. Додатки Global Fishing Watch та веб-додаток власної розробки мають найбільший розмір сторінок (7 МБ), що негативно впливає на їх швидкодію. Розглянуті веб-додатки мають ефективне використання кешу, яке становить 70-80%. Час виконання клієнтського коду для всіх додатків становить близько 150-250 мс, що є прийнятним. Розглянуті веб-додатки мають низьку частоту помилок (0,5-2%), що свідчить про стабільність їх функціонування. Відтак, можна стверджувати, що веб-додаток власної розробки має певні напрямки для покращення, зокрема, в області розміру сторінок та частоти помилок.

Для покращення ефективності роботи веб-додатку власної розробки рекомендуємо наступні заходи: оптимізація розміру сторінок (аналіз коду та ресурсів, щоб зменшити їхній обсяг; мініфікація та компресія зображень, об'єднання та мінімізація CSS та JavaScript файлів, а також використання CDN для швидкого завантаження статичних ресурсів); підвищення ефективності кешування (ревізія заголовків кешування для статичних файлів, таких як зображення та JavaScript, а також розглянути можливості використання серверного кешування); моніторинг та виправлення помилок (моніторинг та логування помилок для ідентифікації та вирішення проблем); оптимізація серверної відповіді (використання кешування на рівні сервера, а також масштабування серверної інфраструктури); перевірка швидкості завантаження сторінок (використання інструментів профілювання швидкості завантаження сторінок, таких як Google PageSpeed Insights, для виявлення можливостей оптимізації швидкості завантаження).

Висновки до розділу 2

В другому розділі було описано особливості розробки веб-додатка для моніторингу та управління природними ресурсами на основі ГІС.

Сформульовано вимоги до веб-додатку, які включають в себе функціональні, технічні та інтерфейсні вимоги. Згідно з визначеними вимогами розробка веб-додатка надає користувачу можливість моніторингу та управління природними ресурсами на основі геоданих, що передбачає швидкий відгук сервера, велику продуктивність та інтерактивний інтерфейс.

Визначено оптимальний стак для розробки: використання Google Earth Engine та JavaScript для маніпуляції з геоданими, React.js / Vue.js / Angular для створення користувацьких інтерфейсів. Mapbox API використовується для роботи з геоданими, а PostGIS як база даних.

Встановлено та налаштовано pgAdmin для розробки бази даних PostgreSQL для зберігання геоданих та іншої інформації, необхідної для веб-додатку.

Продемонстровано як frontend, так і backend частини веб-додатка. Реалізацію frontend показано з використанням React.js / Vue.js / Angular, а backend – з використанням Node.js та Express.js.

Показано забезпечення інтерактивності з геоданими за допомогою Google Earth Engine та інших інструментів для роботи з геоданими, що дозволяє користувачам взаємодіяти з даними та проводити аналіз.

Описано процедуру розгортання веб-додатку, показано настройку середовища виконання, встановлення необхідних залежностей та забезпечення підтримки серверної і клієнтської частин.

Оцінка ефективності показує, що розробка веб-додатку надає продукт, що володіє швидким часом відгуку сервера, адекватним часом завантаження сторінок та ефективним використанням кешу. Однак існують певні області для покращення, такі як розмір сторінок та частота помилок, які можуть бути покращені для забезпечення ще більшої ефективності веб-додатку.

ВИСНОВКИ

В результаті написання дипломної роботи було зроблено наступні висновки:

1. В ході дослідження було встановлено, що використання геоінформаційних систем для управління природними ресурсами дозволяє здійснювати комплексний моніторинг, аналіз та прийняття рішень на основі геопросторових даних. ГІС дозволяють враховувати географічні аспекти в управлінських процесах, що робить їх незамінним інструментом для збереження та ефективного використання природних ресурсів.

2. Проведений огляд існуючих програмних рішень показав, що існують деякі веб-додатки для моніторингу та управління природними ресурсами на основі ГІС. Було розглянуто такі програмні рішення, як Global Forest Watch, Sentinel Hub, Global Fishing Watch, USGS Earth Explorer, Google Earth Engine, Земельний кадастр України та геопортал «Ліси України». Порівняння цих веб-додатків показало, загальні проблеми, які потребують вирішення – недостатня інтеграція та аналіз географічних даних для ефективного моніторингу та управління природними ресурсами; проблеми зі швидкодією веб-додатку.

3. Існуючі програмні рішення не завжди відповідають потребам користувачів, тому створення нового веб-додатка є важливим кроком у розвитку цієї області. Розробка веб-додатка для моніторингу та управління природними ресурсами допоможе у зборі, аналізі та використанні даних про природні ресурси, такі як ліси, водні джерела, ґрунти тощо. Це дозволить оптимізувати їх використання, зменшуючи втрати та сприяючи сталому розвитку. В той же час, веб-додаток служитиме інструментом для збору та аналізу даних про зміни клімату, такі як температура, опади, рівень води тощо. Це допоможе у розробці ефективних стратегій адаптації до змін клімату та зменшення їх негативного впливу на природу.

4. Було показано розробку веб-додатка для моніторингу та управління природними ресурсами на основі ГІС на основі комбінації технологій: Google

Earth Engine та JavaScript для інтерактивності та маніпуляції з геоданими, React.js / Vue.js / Angular для створення потужних користувацьких інтерфейсів, Mapbox API та PostGIS як базу даних. У ході розробки веб-додатку були враховані особливості геоінформаційних технологій, а також вимоги до зручності використання і функціональності. Особлива увага була приділена візуалізації даних, інтерактивним можливостям та можливостям управління.

5. Очікується, що розробка такого веб-додатку значно полегшить процес моніторингу та управління природними ресурсами для зацікавлених сторін в Україні, сприяючи збереженню ресурсів та покращенню екологічної ситуації. Забезпечуючи користувачам з України зручний доступ до геопросторової інформації та інструментів аналізу, веб-додаток допоможе приймати обґрунтовані рішення з управління природними ресурсами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Відкриті дані земельного кадастру України. Відкриті дані земельного кадастру України. URL: https://kadastr.live (дата звернення: 11.04.2024).

2. Геоінформаційні системи і бази даних : монографія / В. І. Зацерковний та ін. Ніжин : НДУ ім. М. Гоголя, 2014. 492 с.

3. Геоінформаційні технології в екології / І. В. Пітак та ін. Чернівці, 2012. 273 с.

4. Геопортал:Ліси України|forestry.org.ua. Геопортал:Ліси України|forestry.org.ua. URL: https://forestry.org.ua/ (дата звернення: 11.04.2024).

5. ДП «Ліси України». e-forest. URL: https://e-forest.gov.ua/ (дата звернення: 13.04.2024).

6. Лекція 9 «Геоінформаційні системи та технології та їх використання в туризмі». Державний університет «Житомирська політехніка». URL: https://learn.ztu.edu.ua/pluginfile.php/265129/mod_resource/content/1/L9_GIS.pdf (дата звернення: 06.04.2024).

7. Світличний О. О., Плотницький С. В. Основи геоінформатики. Суми : ВТД «Унів. кн.», 2006. 295 с.

8. Шипулін В. Д., Кучеренко Є. І. Планування і управління гіс-проектами. Харків : ХНМАГ, ХНУРЕ, 2009. 158 с.

9. Abler R. The National Science Foundation National Center for Geographic Information and Analisis. Int. J. of Geographical Information Systems. 1987. No. 4. P. 302–306.

10. Ajayi A. D., Boiarskii B. Utilizing MapBox API, Java, and ICT in the creation of agricultural interactive maps for improved farm management and decision-making. AIMS agriculture and food. 2024. Vol. 9, № 2. P. 393–410.

11. Apache CouchDB. 2024. 697 p. URL: https://docs.couchdb.org/_/downloads/en/latest/pdf/ (date of access: 22.04.2024).

12. ArcGIS. URL: https://www.arcgis.com/index.html (date of access: 16.04.2024).

13. Atom-editor. Stack Overflow. URL: https://riptutorial.com/Download/atom-editor.pdf (date of access: 22.04.2024).

14. Bitbucket. URL: https://bitbucket.org/ (date of access: 22.04.2024).

15. Black D. A. Ruby for rails. Manning Publications Co., 2006. 529 p.

16. Brown E. Web development with Node and Express. 2nd ed. O'Reilly.

17. Chrome DevTools. Chrome for Developers. URL: https://developer.chrome.com/docs/devtools (date of access: 22.04.2024).

18. Degani A. Methodological observation on the state of geocartographic analisis in the context of automated spatial information systems. Map Data Process. Proc. NATO. Acad. Press. 1980. P. 207–220.

19. Del Sole A. Visual studio code distilled: evolved code editing for windows, macos, and linux. Cremona : Apress, 2019. 221 p.

20. Django documentation. Django Software Foundation, 2022. 2060 p.

21. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis: «John Wiley & Sons, Inc.», 2011. 490 p.

22. Dwyer G. Flask by example. Birmingham - Mumbai : Packt publishing, 2016. 277 p.

23. EarthExplorer. EarthExplorer. URL: https://earthexplorer.usgs.gov/ (date of access: 08.04.2024).

24. Forest monitoring, land use & deforestation trends | global forest watch. Forest Monitoring, Land Use & Deforestation Trends | Global Forest Watch. URL: https://www.globalforestwatch.org/ (date of access: 08.04.2024).

25. From angular to Vue: a cross-language comparative survey of web frameworks / T. K. Mohd et al. Conference: international conference on intelligent systems and new applicationsat: liverpool, UK. 2023.

26. GDAL. GDAL documentation. URL: https://gdal.org/index.html (date of access: 16.04.2024).

27. Geographical Information Systems (GIS). MANAGE. URL: https://www.manage.gov.in/studymaterial/GIS.pdf (date of access: 06.04.2024).

28. Git. URL: https://git-scm.com/ (date of access: 22.04.2024).

29. GitHub. URL: https://github.com/ (date of access: 22.04.2024).

30. Google Earth Engine. Google Earth Engine. URL: https://earthengine.google.com/ (date of access: 08.04.2024).

31. GRASS GIS. Bringing advanced geospatial technologies to the world. URL: https://grass.osgeo.org/ (date of access: 16.04.2024).

32. Hu S., Dai T. Online map application development using google maps API, SQL database, and ASP.NET. International journal of information and communication technology research. 2023. No. 3. P. 102–110.

33. Jayapalan L. Oracle Spatial and Graph Developer's Guide. Oracle, 2023. 1119 p.

34. Jest. Delightful JavaScript Testing. URL: https://jestjs.io/uk/ (date of access: 22.04.2024).

35. Leaflet. URL: https://leafletjs.com/ (date of access: 16.04.2024).

36. Learning SASS. riptutorial. URL: https://riptutorial.com/Download/sass.pdf (date of access: 19.04.2024).

37. Maps API for javascript. developer's guide. 2023. URL: https://www.here.com/docs/bundle/maps-api-for-javascript-developer-guide-3.1.48.0/resource/Maps_API_for_JavaScript_HLP_v3.1.48.0_Developer_Guide.pdf (date of access: 22.04.2024).

38. Nixon R. Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites. O’Reilly Media, 2014. 786 p.

39. Npm. URL: https://www.npmjs.com/ (date of access: 22.04.2024).

40. OpenLayers. URL: https://openlayers.org/ (date of access: 16.04.2024).

41. OpenStreetMap API – Public APIs. Public APIs. URL: https://publicapis.io/open-street-map-api (date of access: 22.04.2024).

42. Pilli K. Concept of GIS and its applications in natural resource management. International Journal of Chemical Studies. 2019. No. 7(5). P. 1515–1524.

43. PostGIS. URL: https://postgis.net/ (date of access: 16.04.2024).

44. PostgreSQL. URL: https://www.postgresql.org/ (date of access: 19.04.2024).

45. QGIS. URL: https://www.qgis.org/uk/site/ (date of access: 16.04.2024).

46. Seguin K. The Little MongoDB Book. 66 p.

47. Sentinel Hub. Sentinel Hub. URL: https://www.sentinel-hub.com/ (date of access: 09.04.2024).

48. Spurlock J. Bootstrap. O'Reilly, 2013. 126 p.

49. Stauffe M. Laravel up & running. A framework for building modern PHP apps. 2nd ed. O'Reilly, 2019. 544 p.

50. Sublime Text. Tutorials Point, 2015. 87 p.

51. Transparency for a sustainable ocean | global fishing watch. Global Fishing Watch. URL: https://globalfishingwatch.org/ (date of access: 09.04.2024).

52. Tutorials | Overview | ArcGIS Maps SDK for JavaScript 4.29. ArcGIS Developers. URL: https://developers.arcgis.com/javascript/latest/tutorials/ (date of access: 22.04.2024).

53. Understanding Gis: The Arc Info Method : Self-Study Workbook. Environmental Systems Research, 1997. 560 p.

54. Walls C. Spring in action. SHELTER ISLAND : Manning Publications Co., 2022. 520 p.

55. Waters N. GIS: History. The International Encyclopedia of Geography. 2017.

56. Weather API. Current weather and forecast - OpenWeatherMap. URL: https://openweathermap.org/api (date of access: 22.04.2024).

ДОДАТОК А

Лістинг коду розробки фронтенд за допомогою React.js

```
import React, { useState, useEffect } from 'react';
import Map from './components/Map';
import ResourceList from './components/ResourceList';
import ReportGenerator from './components/ReportGenerator';
import FilterPanel from './components/FilterPanel';
import ApiService from './services/ApiService';
function App() {
  const [resources, setResources] = useState([]);
  const [filteredResources, setFilteredResources] = useState([]);
  const [selectedResource, setSelectedResource] = useState(null);
  useEffect(() => {
    const fetchData = async () => {
      const data = await ApiService.getResources();
      setResources(data);
      setFilteredResources(data);
    };
    fetchData();
  }, []);
  const handleFilter = (filters) => {
  };
  const handleSelectResource = (resource) => {
    setSelectedResource(resource);
  };
  return (
    <div className="app">
      <FilterPanel onFilter={handleFilter} />
      <div className="content">
        <ResourceList resources={filteredResources} onSelectResource={handleSelectResource} />
        <Map resources={filteredResources} selectedResource={selectedResource} />
        <ReportGenerator selectedResource={selectedResource} />
      </div>
    </div>
  );
}

export default App;
```

ДОДАТОК Б

Лістинг коду комунікації між компонентами

```
import React from 'react';
const Map = ({ selectedResource }) => {
  return (
    <div className="map">
      <p>Selected Resource: {selectedResource}</p>
    </div>
  );
};
export default Map;
```

```
import React from 'react';
const ResourceList = ({ resources, onResourceSelect }) => {
  return (
    <div className="resource-list">
      <h2>Resource List</h2>
      <ul>
        {resources.map((resource) => (
          <li key={resource.id} onClick={() => onResourceSelect(resource)}>
            {resource.name}
          </li>
        ))}
      </ul>
    </div>
  );
};
export default ResourceList;
```

```
import React from 'react';
const ResourceList = ({ resources, onResourceSelect }) => {
  return (
    <div className="resource-list">
      <h2>Resource List</h2>
      <ul>
        {resources.map((resource) => (
          <li key={resource.id} onClick={() => onResourceSelect(resource)}>
            {resource.name}
```

```
        </li>
      ))}
    </ul>
  </div>
 );
};
export default ResourceList;


import React from 'react';
const ReportGenerator = ({ selectedResource }) => {
 return (
  <div className="report-generator">
   <h2>Report Generator</h2>
   <p>Selected Resource: {selectedResource}</p>
   {/* Форма для створення звіту */}
  </div>
 );
};
export default ReportGenerator;


import React from 'react';
const FilterPanel = ({ onFilterChange }) => {
 const handleFilterChange = (event) => {
  const value = event.target.value;
  onFilterChange(value);
 };
 return (
  <div className="filter-panel">
   <h2>Filter Panel</h2>
   {/* Фільтри для вибору критеріїв */}
   <input type="text" placeholder="Filter" onChange={handleFilterChange} />
  </div>
 );
};
export default FilterPanel;


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

```
import ReportGenerator from './ReportGenerator';
import FilterPanel from './FilterPanel';
const MainComponent = () => {
  const [selectedResource, setSelectedResource] = useState(null);
  const [filteredData, setFilteredData] = useState([]);
  const handleResourceSelect = (resource) => {
    setSelectedResource(resource.name);
  };
  const handleFilterChange = (value) => {
    setFilteredData([...]);
  };
  return (
    <div>
      <Map selectedResource={selectedResource} />
      <ResourceList resources={filteredData} onResourceSelect={handleResourceSelect} />
      <ReportGenerator selectedResource={selectedResource} />
      <FilterPanel onFilterChange={handleFilterChange} />
    </div>
  );
};
export default MainComponent;
```

ДОДАТОК В

Лістинг коду реалізації бекенду та ApiService

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/api/resources', (req, res) => {
  res.json(resources);
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});

class ApiService {
  static getResources() {
    return fetch('/api/resources')
      .then((response) => response.json())
      .catch((error) => console.error('Error fetching resources:', error));
  }
}

export default ApiService;
```

ДОДАТОК Г

Налаштування з'єднання з базою даних

```
const express = require('express');
const app = express();
const port = 3000;
const { Pool } = require('pg');

const pool = new Pool({
  user: 'your_username',
  host: 'localhost',
  database: 'your_database',
  password: 'your_password',
  port: 5432,
});

app.get('/api/resources', async (req, res) => {
  try {
    const client = await pool.connect();
    const result = await client.query('SELECT * FROM resources');
    const resources = result.rows;
    client.release();
    res.json(resources);
  } catch (error) {
    console.error('Error fetching resources:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

ДОДАТОК Д

Логіка обробки фільтрів та звітів на бекенді

```
app.post('/api/filter', async (req, res) => {
  const { filters } = req.body;
  try {
    const client = await pool.connect();
    SELECT * FROM resources WHERE type = $1 AND region = $2;
    const result = await client.query('SELECT * FROM resources WHERE type = $1 AND region = $2', [filters.type, filters.region]);
    const filteredResources = result.rows;
    client.release();
    res.json(filteredResources);
  } catch (error) {
    console.error('Error filtering resources:', error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
});
```

ДОДАТОК Е

Інтерактивність з геоданими

1. Відображення шарів ресурсів

```
import React, { useEffect } from 'react';
import { Map, TileLayer } from 'react-leaflet';
import L from 'leaflet';
import 'leaflet/dist/leaflet.css';

const ResourceMap = () => {
  useEffect(() => {
    const initializeGEE = async () => {
      const { data } = await window.geeApiPromise;
      const { Map } = data;

      const imageLayer = Map.addLayer({
        eeObject: ee.ImageCollection("COPERNICUS/S2")
          .filterDate('2022-01-01', '2022-12-31')
          .median()
          .clipToCollection(geometry),
        visParams: {
          bands: ['B4', 'B3', 'B2'],
          min: 0,
          max: 3000,
        },
        name: 'Sentinel-2 Imagery',
      });

      L.control.layers(null, { 'Sentinel-2 Imagery': imageLayer }).addTo(Map);
    };

    initializeGEE();

    return () => {
      if (window.geeApiPromise) {
        const { data } = window.geeApiPromise;
        data.Map.clear();
      }
    };
  }, []);
```

```
  return (
    <div style={{ height: '400px' }}>
      <Map center={[0, 0]} zoom={2}>
        <TileLayer
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        />
      </Map>
    </div>
  );
};

export default ResourceMap;
```

2. Навігація по мапі

```
import React from 'react';
import { Map, TileLayer, ZoomControl } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';

const ResourceMap = () => {
  return (
    <div style={{ height: '400px' }}>
      <Map center={[0, 0]} zoom={2} zoomControl={false}>
        <TileLayer
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        />
        <ZoomControl position="bottomright" />
      </Map>
    </div>
  );
};

export default ResourceMap;
```

3. Вибір конкретних областей

```
import React, { useState } from 'react';
import { Map, TileLayer, Marker, Popup } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';

const ResourceMap = () => {
```

```
  const [selectedArea, setSelectedArea] = useState(null);

  const handleMapClick = (event) => {
    const { latlng } = event;
    setSelectedArea(latlng);
  };

  return (
    <div style={{ height: '400px' }}>
      <Map center={[0, 0]} zoom={2} onClick={handleMapClick}>
        <TileLayer
          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        />
        {selectedArea && (
          <Marker position={selectedArea}>
            <Popup>
              Selected Area: <br /> {selectedArea.lat.toFixed(2)}, {selectedArea.lng.toFixed(2)}
            </Popup>
          </Marker>
        )}
      </Map>
    </div>
  );
};

export default ResourceMap;
```

ДОДАТОК Ж

Розгортання та підтримка веб-додатка

1. Створення серверної частини веб-додатка

```
mkdir my-app
cd my-app
npm init -y

npm install pg

mkdir src
cd src
mkdir routes controllers services
touch index.js

const express = require('express');
const router = express.Router();

const { getAllUsers, getUserById, createUser, updateUser, deleteUser } = require('../controllers/usersController');

router.get('/users', getAllUsers);
router.get('/users/:id', getUserById);
router.post('/users', createUser);
router.put('/users/:id', updateUser);
router.delete('/users/:id', deleteUser);

module.exports = router;

const { Pool } = require('pg');

const pool = new Pool({
  user: 'ChekanV',
  host: '3000',
  database: 'ChekanGIS',
  password: '1111',
  port: 5432,
});

const getAllUsers = async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM users');
```

```
    res.json(result.rows);
  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  }
};

module.exports = {
  getAllUsers,
};

const express = require('express');
const routes = require('./routes/routes');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(express.json());
app.use('/api', routes);

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

2. Підключення до бази даних

```
npm install pg

const { Pool } = require('pg');

const pool = new Pool({
  user: 'ChekanV',
  host: 'localhost',
  database: 'ChekanGIS',
  password: '1111',
  port: 5432,
});

pool.connect((err, client, release) => {
  if (err) {
    return console.error('Error acquiring client', err.stack);
  }
  console.log('Connected to PostgreSQL database');
```

```
  client.release();
});

module.exports = pool;

pool.query('SELECT NOW()', (err, res) => {
  if (err) {
    console.error('Error executing query', err.stack);
  } else {
    console.log('Current date in the database:', res.rows[0].now);
  }
});
```

3. Налаштування API для взаємодії з даними

```
const express = require('express');
const router = express.Router();
const controller = require('./controller');

router.get('/data', controller.getData);
router.post('/data', controller.createData);
router.put('/data/:id', controller.updateData);
router.delete('/data/:id', controller.deleteData);

module.exports = router;

const pool = require('./db');

const getData = (req, res) => {
  pool.query('SELECT * FROM your_table', (err, results) => {
    if (err) {
      throw err;
    }
    res.status(200).json(results.rows);
  });
};

const createData = (req, res) => {
  const { field1, field2 } = req.body;
  pool.query(
    'INSERT INTO your_table (field1, field2) VALUES ($1, $2)',
    [field1, field2],
    (err) => {
```

```
    if (err) {
      throw err;
    }
    res.status(201).send('Data created successfully');
  }
 );
};

const updateData = (req, res) => {
 const id = req.params.id;
 const { field1, field2 } = req.body;
 pool.query(
  'UPDATE your_table SET field1 = $1, field2 = $2 WHERE id = $3',
  [field1, field2, id],
  (err) => {
    if (err) {
      throw err;
    }
    res.status(200).send(`Data with ID ${id} updated successfully`);
  }
 );
};

const deleteData = (req, res) => {
 const id = req.params.id;
 pool.query('DELETE FROM your_table WHERE id = $1', [id], (err) => {
  if (err) {
    throw err;
  }
  res.status(200).send(`Data with ID ${id} deleted successfully`);
 });
};

module.exports = {
 getData,
 createData,
 updateData,
 deleteData,
};

const express = require('express');
const routes = require('./routes');
```

```
const app = express();

app.use(express.json());

app.use('/api', routes);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```