НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 122 Комп’ютерні науки та інформаційні технології

на тему Автоматизація інтерактивної карти зменшення викидів вуглекислого газу при освоєнні потенціалу сонячної енергії по областях України

Виконав (-ла): студент (-ка) 4 курсу, групи ТМ_62

Мєхов Ігор Вадимович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент кафедри, к.в.н. доцент Онисько А.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Мєхов Ігор Вадимович
(прізвище, ім’я, по батькові)

1. Тема роботи Автоматизація інтерактивної карти зменшення викидів вуглекислого газу при освоєнні потенціалу сонячної енергії по областях України

керівник роботи доцент кафедри, к.в.н. доцент Онисько А.І.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи React.js проект

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити проаналізувати існуючі програмні рішення та засоби аналізу викидів вуглекислого газу при освоєнні потенціалу сонячної енергії, спроектувати архітектуру системи аналізу викидів CO2, розробити програмне забезпечення, розробити інтерфейс користувача

5. Перелік ілюстративного матеріалу
1. Актуальність 2. Мета та завдання роботи 3. Функції додатку 4. Опис функціональності системи 5. Архітектура програмного комплексу 6. Опис структури інтерфейсу 7. Використані програмні засоби 8. Інтерфейс головної сторінки 8. Калькулятор викидів вуглекислого газу 9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Матях С.В.		

7. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	25.02.2020	
3.	Розробка архітектури та загальної структури системи	10.04.2020	
4.	Розробка структур окремих підсистем	15.04.2020	
5.	Програмна реалізація системи	25.04.2020	
6.	Оформлення пояснювальної записки	25.05.2020	
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було розробити серверну частину та адміністративну панелі для інтерактивної карти альтернативних джерел енергії України. Програма дозволяє дослідити показники викидів вуглекислого газу при освоєнні потенціалу сонячної енергії на території України. Завдяки цьому користувач має можливість провести аналіз ефективності встановлення сонячних систем та їх екологічний вплив по всім регіонам України. Адміністративна панель, дозволяє виводити та розраховувати данні викидів CO2 а також візуалізувати їх на інтерактивній карті України.

Записка містить 72 сторінки, 10 картинок та 10 посилань

ABSTRACT

The aim of the work was to develop a server part and an administrative panel for an interactive map of alternative energy sources in Ukraine. The program allows to study the indicators of carbon dioxide emissions during the development of the potential of solar energy in Ukraine. Due to this, the user has the opportunity to analyze the effectiveness of the installation of solar systems and their environmental impact in all regions of Ukraine. The administrative panel allows you to display and calculate data on CO2 emissions and visualize them on an interactive map of Ukraine.

The note contains 72 pages, 10 pictures and 10 links

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних;

MS (англ. Microsoft) — багатонаціональна корпорація комп'ютерних технологій.

ВСТУП

Однією з найперспективніших сфер розвитку глобальної енергетики сьогодні є використання відновлюваних джерел енергії, що усуває низку проблем, пов'язаних із використанням традиційних видів палива. Кінець другого тисячоліття характеризується інтенсивним зростанням використання відновлюваних джерел енергії у світі. Враховуючи виклики глобального потепління, все більше людей розглядають можливість переходу на екологічно чисте поновлюване джерело енергії. Ця система має на меті допомогти людям, зацікавленим у цьому

В результаті обробки статистичних метеорологічних даних щодо надходження сонячної радіації визначено конкретні енергетичні показники сонячної енергії та розподіл енергетичного потенціалу сонячної радіації для кожного з регіонів України. Ці енергетичні показники для сонячного випромінювання є основою для обрахунку екологічності сонячних систем.

Тому було розроблено веб-додаток, в якому за допомогою даних метеостанцій буде проаналізовано екологічний вплив при використання сонячних систем в регіонах України для моніторингу стану навколишнього середовища, а точніше викидів вуглекислого газу. За допомогою супутників ми можемо отримати дані про склад атмосфери та рельєфу в регіонах України та автоматизувати створення та контроль сонячних електростанцій.

Користувачем системи буде кожен відвідувач сайту, який цікавиться альтернативними джерелами енергії, але не знає деталей клімату у своєму регіоні. Він отримує можливість вибрати будь-яку точку на карті України та завдяки метеорологічним даним цього регіону обчислити викиди CO_2, при встановленні відновлювальних систем.

Записка містить 7 розділів.

У першому розділі описується проблема відображення викидів CO_2 при використанні відновлювальних джерел енергії по областям України.

У другому розділі описані характеристики сонячної радіації.

У третьому розділі описані підходи до використання сонячної енергії.

У четвертому розділі описані методи аналізу викидів вуглекислого газу.

. У п’ятому розділі вказані основні інструменти розробки цієї системи.

Шостий розділ описує реалізований програмний продукт та його архітектуру.

У сьомому розділі описано роботу користувача з системою.

1.ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ ІНТЕРАКТИВНОЇ КАРТИ СОНЯЧНИХ ЕНЕРГОДЖЕРЕЛ УКРАЇНИ

З темпом розвитку сучасного світу та технологій важливо ефективно систематизувати та аналізувати дані. Кількість даних досить велика, тому для швидкого отримання корисної інформації з них зручно використовувати засоби візуалізації. Прикладом таких даних є повідомлення про сонячну активність у різних куточках світу.

Сонячна енергія - одна з перспективних сфер використання відновлюваної енергії, яка швидко розвивається. З усіх відновлюваних джерел сонячна енергія є найбільш містким і доступним природним енергетичним ресурсом; у його використанні накопичено багатовіковий історичний досвід. На сучасному етапі розвитку сонячної енергії на першому місці стоять проблеми екологічного впливу відновлювальних джерел енергії.. Перевагами практично невичерпного джерела енергії сонячного випромінювання при використанні в якості основного локального енергетичного ресурсу є можливість використання джерела тепла на більшій частині поверхні Землі та можливість безпосередньо перетворювати енергію сонячного випромінювання в електроенергію.

Світова стратегія розвитку енергетики передбачає, що до 2020 року 20% електроенергії буде вироблятися за рахунок використання відновлюваних джерел енергії (ВДЕ) та альтернативних видів палива (АВП), у 2040 р. - 50%, а в кінці XXI століття частка електроенергія, вироблена з ВДЕ та АВП, може перевищувати 85%. До 2050 року планується збільшити частку ВДЕ та АВП у загальному паливно-енергетичному балансі Євросоюзу до 50%.

Цілі України як однієї з учасників Паризької кліматичної угоди - скоротити викиди CO2 на 25% до 2020 року, на 40% до 2030 року та запропоновану амбітну мету до 2050 року на 70%.

З результатами розвитку зеленної енергетики бізнес в цій сфері стає достатньо актуальним але йому досить важно розвиватися через наступні фактори:

- Інформаційна база
- Статистичні ресурси
- Застосування існуючих інтерактивних карт

Тому з’явилась ідея дослідження зменшення викидів CO_2 при створенні відновлювальних джерел на території України .

Зібравши дані, необхідно подати їх у формі, зрозумілій кожному користувачеві і поєднати їх в простому та загальнодоступному веб-додатку, основним функціоналом якого є:

- Відображення інтуітивно зрозумілої карти викидів вуглекислого газу по України
- Калькулятор розрахунку викидів з використанням індивідуальних даних по різним типам систем
- Мітки на карті вже існуючих систем
- Таблиці з дослідженими даними по областям України для різних типів систем
- Можливість додавати нові системи на карту та їх данні в таблицю

2 СОНЯЧНА ЕНЕРГЕТИКА

У даному розділі було проведено аналіз актуальності сонячної енергетики та існуючих програмних рішень.

2.1. Потенціал сонячної енергетики

Використання відновлюваних джерел енергії є одним із найперспективніших напрямків світового розвитку енергетики сьогодні, що усуває низку проблем, що виникають у функціонуванні традиційної енергетики, включаючи шкідливий вплив на довкілля.

Необхідність докорінних змін у світовій енергетиці пов'язана з виснаженням традиційних викопних видів палива, енергетичними кризами та катастрофами на атомних електростанціях. Виробництво енергії з використанням викопного палива та ядерної енергії супроводжується забрудненням навколишнього середовища та негативним впливом на тепловий баланс планети, що може призвести до глобальних незворотних змін клімату.

Розвиток відновлюваної енергетики в різних країнах є нагальним завданням, незалежно від стану їх промислового розвитку. Таким чином, якщо для промислово розвинених країн, які не мають власних традиційних енергетичних ресурсів, енергетична безпека є першорядною, то для промислово розвинених країн забезпечені власними традиційними енергетичними ресурсами, екологічна безпека, збереження органічних енергетичних ресурсів для майбутніх поколінь та для неенергетичного використання , а також доступ до світу. ринки обладнання для відновлюваної енергії. Країни, що розвиваються, мають можливість покращити соціальні та побутові умови населення та рівень промислового розвитку за допомогою нових екологічно чистих технологій.

Необхідність використання відновлюваних джерел енергії в економіках розвинених країн обумовлена не тільки обмеженими запасами викопних палив, але і вимогами скорочення викидів парникових газів, особливо вуглекислого газу. Парниковий ефект - це екологічна проблема, що визначається як підвищення температури та вологості земної атмосфери за рахунок викиду вуглекислого газу в атмосферу та його поглинання інфрачервоним випромінюванням. Розширення споживання відновлюваної енергії, враховуючи той факт, що використання майже кожного з цих джерел не супроводжується викидами CO_2, не тільки зменшить глобальні викиди, але й наддасть найближчим часом можливість збільшити виробництво енергії, оскільки використання відновлюваної енергії як первинних джерел енергії практично не впливає на тепловий баланс планети.

Загальний щорічний приплив сонячної радіації на територію України оцінюється у 720 · 1012 кВт · год, що еквівалентно 88,4 млрд т п.п. Територіальний розподіл теоретично можливого та технічно досяжного потенціалу сонячної енергії на території України наведено в таблиці 2.1 та на рис.2.1

Таблиця 2.1 Потенціал сонячної України

№ з/п	Області	Потенціал сонячної енергії, т у.п./рік		№ з/п	Області	Потенціал сонячної енергії, т у.п./рік	
		Теоретично-можливий потенціал ($\times 10^9$)	Технічно-досяжний Потенціал ($\times 10^5$)			Теоретично-можливий потенціал ($\times 10^9$)	Технічно-досяжний потенціал ($\times 10^5$)
1.	АР Крим	4,5	3,8	14.	Миколаївська	4,0	2,6
2.	Вінницька	3,7	2,5	15.	Одеська	5,6	3,7
3.	Волинська	2,6	1,8	16.	Полтавська	3,8	2,6
4.	Дніпропетровська	4,5	3,2	17.	Рівненська	2,6	1,7
5.	Донецька	4,1	2,7	18.	Сумська	3,2	2,2
6.	Житомирська	4,0	2,6	19.	Тернопільська	2,0	1,5
7.	Закарпатська	1,8	1,4	20.	Харківська	4,3	2,9
8.	Запорізька	4,3	2,8	21.	Херсонська	4,7	3,1
9.	Івано-Франківська	2,0	1,3	22.	Хмельницька	3,0	2,0
10.	Київська	3,8	2,6	23.	Черкаська	3,8	2,1
11.	Кіровоградська	3,4	2,3	24.	Чернівецька	1,2	0,9
12.	Луганська	4,2	2,7	25.	Чернігівська	4,2	2,8
13.	Львівська	3,1	2,2		**ВСЬОГО**	**88,4**	**60,0**

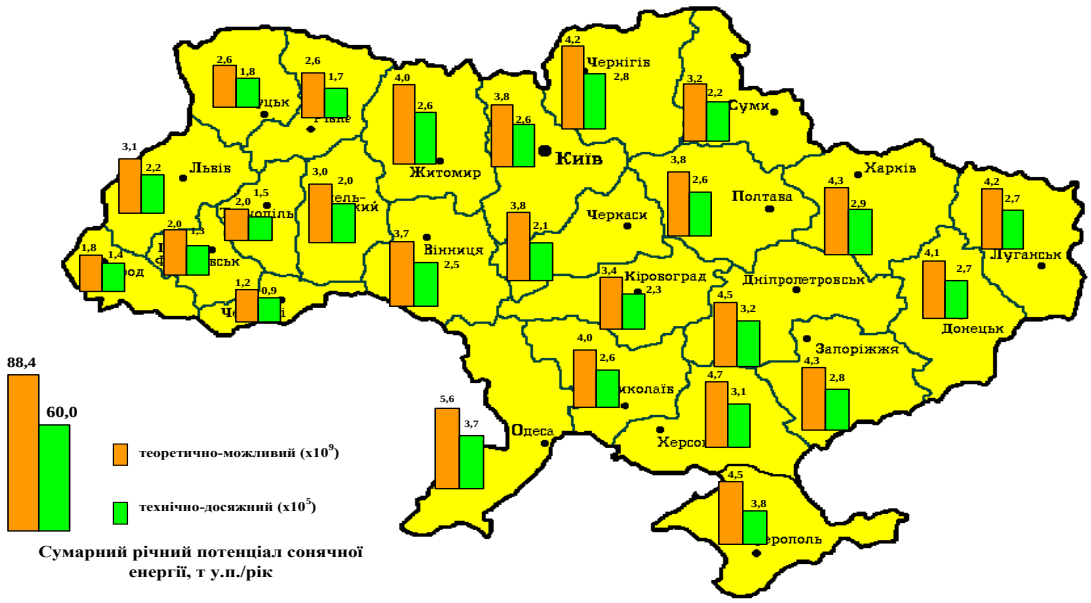

Рисунок 2.1 Потенціал сонячної України

Щорічно технічно досяжний енергетичний потенціал сонячної енергії в Україні еквівалентний 6 мільйонам тонн, а його використання заощаджує близько 5 млрд. $М^3$ природного газу.

В результаті обробки статистичних метеорологічних даних про надходження сонячної радіації визначено конкретні енергетичні показники сонячної енергії та розподіл енергетичного потенціалу сонячної радіації для кожного з регіонів України. Щорічно технічно досяжний енергетичний потенціал сонячної енергії в Україні еквівалентний 4,2 мільйона тонн н.е., а його використання заощаджує близько 5 млрд. М3 природного газу.

Ці енергетичні показники сонячної радіації є основою для вибору обладнання для сонячної енергії та встановлення його оптимальної потужності, а розподіл середньорічної загальної сонячної радіації дає загальне уявлення про доцільність розміщення обладнання сонячної енергії у певній місцевості.

Середньорічна кількість загальної сонячної радіації на 1 м2 поверхні в Україні знаходиться в межах від 1070 кВт / год на м2 у північній частині України до 1400 кВт / год в м2 в Автономній Республіці Крим.

Потенціал сонячної енергії в Україні досить високий для широкого впровадження як фотоелектричного, так і теплоенергетичного обладнання майже у всіх районах.

Фотоелектричне обладнання можна експлуатувати досить ефективно протягом року, а термін ефективної роботи сонячної енергетичної апаратури у південних регіонах України - 7 місяців (з квітня по жовтень), у північних регіонах - 5 місяців (з травня по вересень) .

У кліматичних та метеорологічних умовах України ефективним є використання плоских сонячних колекторів, які використовують як пряму, так і розсіяну сонячну радіацію.

Для регіонів з безсніжною зимою та великою кількістю сонячних днів протягом року (Крим, Причорномор’я та Приазов'я тощо) ефективно використовувати сонячні колектори з вакуумними трубами.

Перетворення сонячної енергії в електроенергію в умовах України має бути зосереджено насамперед на використанні фотоелектричних пристроїв.

Основними перевагами використання сонячної енергії є її довговічність та екологічність в порівнянні з іншими енергетичними джерелами. Сонячна енергія доступна всюди на планеті, крім полюсів. Таким чином, лише ніч і хмари на всій земній кулі можуть перешкоджати її використанню. Через це цей вид енергії неможливо монополізувати, на відміну від нафти та газу. На жаль, вартість 1 кВт / год. сонячна енергія перевищує витрати енергії, отримані традиційними методами. Лише п'ята частина сонячного світла перетворюється на електроенергію, але ця частина продовжує зростати завдяки роботі вчених та інженерів по всьому світу.

Потенціал сонячної енергії в Україні досить високий для широкого впровадження як теплового, так і фотоелектричного обладнання майже у всіх районах. Термін ефективної роботи сонячних водонагрівачів у південних регіонах України становить 7 місяців (з квітня по жовтень), у північних регіонах - 5 місяців (з травня по вересень). В кліматичних і богословських умовах України ефективним є використання як плоских сонячних колекторів, так і надалі концентраторів, які використовують пряме та розсіяне сонячне випромінювання. Фотоелектричне обладнання можна експлуатувати досить ефективно протягом року.

2.2 Сфери використання сонячної енергетики

Основними параметрами, які використовуються при визначенні ефективності сонячних електростанцій, є інтенсивність сонячної радіації та зовнішня температура. Сонячне випромінювання, що надходить на будь-яку поверхню, складається з прямого та дифузного сонячного випромінювання та випромінювання, відбитого від поверхні Землі та різних об'єктів, розташованих поблизу цієї поверхні.

Кількість енергії сонячної радіації багато в чому залежить від астрономічних та метеорологічних факторів - висоти Сонця над горизонтом та тривалості дня, хмар, вологості та прозорості атмосфери. Кількість від загальної сонячної радіації змінюються протягом дня, року та року за роком, але її середньорічна цінність у довгостроковій перспективі є досить стабільною.

Сонячні технології можуть бути як пасивними, так і активними залежно від способу їх використання, перетворення та розподілу сонячного світла, а також дозволяють використовувати сонячну енергію на різних рівнях світу, залежно від відстані до екватора.

Сонячні системи гарячого водопостачання використовують сонячне світло для нагрівання води. На низьких широтах (нижче 40 градусів) сонячні системи опалення можуть забезпечувати від 60 до 70% споживання гарячої води вдома з температурою до 60 ° С.

Велика кількість промислово розвинених країн використовують значну кількість сонячної енергії у своїх електромережах для доповнення або надання альтернативи традиційним джерелам енергії, тоді як менш розвинені країни звертаються до сонячної енергії, щоб зменшити залежність від дорогого імпортного палива. Це дозволяє в значній мірі витіснити споживання викопного палива.

2.3. Способи вимірювання та реєстрації сонячного потенціалу на поверхні землі

При освоєнні потенціалу сонячної енергії з’являється потреба підрахунку її ефективності та умовної кількості на поверхні землі на різних її ділянках.

Основними підходами до отримання даних про сонячні ресурси є наземні датчики, встановлені на професійних метеорологічних станціях та супутникові метеорологічні моделі. Ці два методи доповнюють один одного. Високоточні наземні сонячні датчики забезпечують високочастотні вимірювання з високою

точністю (якщо професійно встановлені та обслуговуються). Зазвичай дані високої точності та надійні дані сонячного датчика доступні лише для обмеженої кількості конкретних місць, і вони представляють лише конкретні та обмежені періоди часу. Супутникові сонячні моделі здатні постійно контролювати великі географічні райони в режимі реального часу.

Перевага супутникових моделей полягає в тому, що вони представляють історію від 10 до більше 20 років (залежно від регіону). Обмежувальною особливістю даних, отриманих із супутникових джерел, є їх часова та просторова роздільна здатність та менша точність (порівняно з високоякісними наземними вимірюваннями). Вимірювання, отримані наземними датчиками, допомагають зменшити невизначеність імітованих значень, тоді як моделі сонячного випромінювання взаємно забезпечують економічний спосіб контролю якості вимірювань, проведених наземними метеостанціями. При плануванні та експлуатації проектів сонячної енергії рекомендованим підходом є використання супутникових моделей для задоволення потреб у сонячних ресурсах на всіх етапах життєвого циклу проекту та одночасно встановлення наземних сонячних та метеорологічних датчиків (метеорологічних станцій) на існуючих або запланованих масштабні проекти та інші стратегічні проекти. місця. Якість сонячних даних має вирішальне значення для економічної та технічної оцінки сонячних електростанцій. Розуміння невизначеності та управління погодними ризиками важливо для успішного планування та експлуатації енергії сонячної енергії.

2.4. Готові програмні рішення пов’язані з сонячною енергетикою

Був проведений ретельний пошук та аналіз існуючих програмних рішень, було виявлено, що на даний момент більшість з них мають складний, нелогічний інтерфейс, неповний технічний функціонал, та нажаль не одна з них не має можливості підрахунку викидів CO_2 при освоєнні потенціалу сонячної енергії

- - "atmosfera.ua" - це веб-сайт з калькулятором енерго-ефективності сонячної електростанції, на якому ви можете вибрати регіон країни та потужність електростанції
- “Solar-battery.com.ua” - це веб-сайт, який надає дані про сонячну активність у різних регіонах України.
- "Enertime.com.ua" - це веб-сайт, який має калькулятор сонячної електростанції, на якому ви можете вибрати потужність і тип електростанції, але не можете вибрати регіон країни.

Отже, створення системи контролю за викидами вуглекислого газу є в своєму роді необхідністю для подальшого розвитку відновлювальної енергетики по областям України.

2.5.Висновки до розділу

У розділі було розглянуто актуальність використання сонячних енергоносіїв на сьогоднішній день, як висновок її використання стає достатньо актуальним за низкою факторів як в тепловій так і в електроенергетиці, при аналізі існуючих програмних рішень, було виявлено відсутність сервісів для підрахунку екологічності освоєння даних систем.

3 ВИКИДИ CO_2 ПРИ ОСВОЄННІ ПОТЕНЦІАЛУ СОНЯЧНОЇ ЕНЕРГІЇ

3.1. Актуальні способи боротьби

Світ все більше бореться із забрудненням навколишнього середовища. Основна увага приділяється викидам парникових газів та пов'язаному з цим катастрофічному зростанню глобальної температури. 12 грудня 2015 року в Парижі було підписано нову угоду в рамках Рамкової конвенції ООН про зміну клімату (РКЗК ООН) щодо регулювання заходів щодо зменшення викидів вуглекислого газу, яка має на меті обмежити зусилля щодо обмеження температури до 1,5 ° С від допромислових рівні. На відміну від Кіотського протоколу, Паризька кліматична угода передбачає, що зобов'язання щодо зменшення шкідливих викидів беруть на себе всі держави, незалежно від рівня їх економічного розвитку.

22 квітня 2016 року на урочистій церемонії Паризьку угоду підписали 175 країн, в тому числі і Україна. 14 липня 2016 року Верховна Рада України ратифікувала Паризьку угоду.

Метою Угоди є посилення імплементації Рамкової конвенції про зміну клімату шляхом:

- утримання середньосвітового зростання температури значно нижче + 2 ° С від доіндустріального рівня та орієнтація на обмеження підвищення температури до +1,5 ° С від доіндустріального рівня, оскільки це значно зменшить ризики та наслідки зміни клімату;

- підвищити здатність адаптуватися до негативних наслідків зміни клімату, підтримувати боротьбу зі зміною клімату, розвиток із низькими викидами парникових газів таким чином, що не загрожує виробництву продуктів харчування;

- узгодження фінансових потоків із шляхом розвитку з протидією змінам клімату та низьким викидом парникових газів ".

Крім того, країни будуть прагнути досягти «поворотного пункту викидів парникових газів якнайшвидше» (оскільки викиди парникових газів довго затримуються в атмосфері, а розвиток та зростання добробуту бідних та найбідніших країн рівень накопичення парникових газів буде досягти поворотної точки - пік з подальшим падінням, тим більший негативний вплив на клімат, а, отже, більш складні та дорогі зусилля для мінімізації.

Передбачається, що такі цілі будуть досягнуті за рахунок субсидій країнам, що розвивають використання низьковуглецевих відновлюваних енергетичних технологій та зменшують виробництво та використання викопних видів палива.

У 2017 році була розроблена стратегія низьковуглецевого розвитку України до 2050 року, цілі якої представлені нижче: [Стратегія розвитку низьковуглецевого рівня України до 2050 р. [Електронний ресурс] // - Режим доступу: https: // menr.gov.ua/news/31815.html].

- Завдання I - перехід до енергосистеми, що передбачає використання низьковуглецевих джерел енергії, розвиток чистої електроенергії та тепла, підвищення енергоефективності та енергозбереження у всіх галузях економіки та житлово-комунальної інфраструктури, стимулювання використання альтернативних автомобільні нафтопродукти та перехід вантажних та пасажирських перевезень за рахунок більш чистих видів транспорту;

- мета II - збільшити захоплення та утримання вуглецю шляхом застосування передового досвіду сільського господарства та лісового господарства, адаптованого до зміни клімату;

- Завдання III - зменшити викиди ПГ, такі як метан та оксид азоту, пов'язані головним чином з виробництвом викопного палива, сільського господарства та відходів.

Стратегія розвитку України з низьким вмістом вуглецю до 2050 року зосереджена на політиці та заходах, які будуть поступово впроваджуватися до середини поточного століття, та передбачає її періодичний огляд та коригування.

Відповідно до Постанови Ради Міністрів Енергетичного Співтовариства "Про імплементацію Директиви 2009/28 / ЄС та внесення змін до статті 20 Договору про створення Енергетичного Співтовариства", яка встановлює обов'язкові національні цілі щодо відновлюваної енергії, насамперед для забезпечення певних гарантує інвесторам та заохочує розвиток нових технологій та інновацій у цій галузі. Україна взяла на себе зобов'язання досягти 11% енергії, виробленої з відновлюваних джерел енергії, у загальному кінцевому споживанні енергії країни до 2020 року, що послужить потужним стимулом для подальшого розвитку відновлюваних джерел енергії в Україні.

Для досягнення цих цілей Кабінет Міністрів України наказом від 03.09.14 р. № 791-р затвердив План дій щодо імплементації Директиви 2009/28 / ЄС Європейського Парламенту та Ради та наказом від 01.10.2014 № 902-і затвердили Національний план дій з відновлюваної енергії на період до 2020 року.

3.2. Підрахунок викидів сонячних електростанцій

Умовне паливо (ПМ) - це одиниця обліку палива, яка використовується для порівняння різних його видів і використовується як енергетичний еквівалент, який харатеризує потенційне споживання енергії або розмір запасів відповідного джерела енергії. В одиницях маси 1 кг п.п. = 29,3 МДж = 7000 ккал = 8,14 кВт · рік = 0,7 кг н.е.

Нафтовий еквівалент - це стандартизована одиниця енергії, яка використовується для порівняння використання великої кількості енергії з різних джерел.

1 т н.е. (ніг) еквівалентний кількості енергії, що виділяється спалюванням однієї тонни сирої нафти, приблизно 41,868 ГДж або 11,63 МВт-год. енергії.

Коефіцієнт 1428 використовується для перетворення однієї тонни нафти в еквівалент однієї тонни вугільного еквівалента.

За допомогою показника "умовне паливо" або "нафтовий еквівалент" формуються залишки палива або загальні енергетичні баланси галузей, країн та світу в цілому.

Екологічна ефективність, тобто заощаджена кількість викидів CO_2, становить:

де k - коефіцієнт викидів CO_2 (1,32 т CO_2 на 1 МВт · год).

1 т AD = 11,63 МВт · год. енергія, відповідно, скорочення викидів вуглекислого газу за рахунок енергетичного потенціалу, що відповідає 1 т н.е., становить (11,63 х 1,32 т) близько 15,4 т.

3.3. Висновки до розділу

У даному розділі було розглянуто актуальні способи боротьби с вуглецевим забрудненням у всьому світі, приведена низка законопроектів пов’язаних з цією проблемою. Приведені способи підрахунку кількості викидів CO_2 при освоєнні енергетичного потенціалу сонячних енергоносіїв.

4 ЗАСОБИ РОЗРОБКИ

Важливо вибрати правильні та зручні інструменти при розробці. Одним із таких інструментів є середовище розробки та бібліотека. Основним середовищем було обрано Visual Studio Code.

Фреймворк React з мовою програмування TypeScript було використано для створення структури інтерфейсу користувача.

Для підключення інтерактивної карти було обрано бібліотеку Leaflet.js та постачальник відкритих карт OpenStreetMap.

4.1. Середовище розробки Visual Studio Code

Visual Studio Code - це редактор вихідного коду, розроблений Microsoft для Windows, Linux та macOS. Він включає підтримку налагодження, вбудовані елементи управління Git та GitHub, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти коду та рефакторинг коду. Він легко налаштовується, що дозволяє користувачам змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, що додають додаткові функції. Вихідний код є безкоштовним та відкритим кодом та випускається за окремою ліцензією MIT. Скомпільовані виконувані файли є безкоштовними для приватного чи комерційного використання.

Visual Studio Code базується на Electron, що використовується для розгортання настільних додатків Node.js, що працюють на механізмі компонування Blink. Хоча воно використовує платформу Electron, програмне забезпечення не використовує Atom, а натомість використовує той самий компонент редактора (кодова назва "Монако"), який використовується в DevOps Azure (раніше Visual Studio Online і Visual Studio Team Services).

У опитуванні розробників Stack Overflow 2019 код Visual Studio був визнаний найпопулярнішим інструментом у середовищі розробки, 50,7% із 87 337 респондентів заявили, що використовують його.

4.2. Фреймворк React

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript для створення інтерфейсів користувача. Його підтримують Facebook та спільнота окремих розробників та компаній.

React може бути використаний в якості основи при розробці односторінкових або мобільних додатків, оскільки це оптимально для вилучення швидкозмінних даних, які потрібно записати. Однак вибірка даних - це лише початок того, що відбувається на веб-сторінці, тому складні програми React зазвичай вимагають використання додаткових бібліотек для управління, маршрутизації та взаємодії з API.

Всі реактивні програми базуються на компонентах. Компонент - це окремий модуль, який видає деякий вихід. Ми можемо записати елементи інтерфейсу, як кнопку або поле введення, як компонент React. Компоненти складаються. Компонент може включати один або більше інших компонентів.

У широкому розумінні для написання програм React ми пишемо React-компоненти, які відповідають різним елементам інтерфейсу. Потім ми організовуємо ці компоненти в межах компонентів вищого рівня, які визначають структуру нашої програми.

Наприклад, візьміть форму. Форма може складатися з багатьох елементів інтерфейсу, таких як поля введення, мітки або кнопки. Кожен елемент всередині форми може бути записаний як компонент React. Потім записуємо компонент вищого рівня, сам компонент форми. Компонент форми визначатиме структуру форми і включає кожен з цих елементів інтерфейсу.

Важливо зазначити, що кожен компонент програми React дотримується принципів суворого управління даними. Складний, інтерактивний інтерфейс користувача часто включає складні дані та стан програми. Площа поверхні React обмежена і має на меті дати нам інструменти, щоб ми могли передбачити, як буде виглядати наша програма за певного набору обставин. Пізніше ми поглибимося в ці принципи.

На відміну від багатьох попередників, React працює не на об'єктній моделі документів браузера (DOM), а на віртуальній DOM. Тобто, замість того, щоб маніпулювати документом у браузері після зміни наших даних (що може бути досить повільним), він вирішує зміни до вбудованої DOM і працює повністю в пам'яті. Після оновлення віртуального DOM React обґрунтовано визначає, які зміни потрібно внести у фактичний браузер DOM.

React Virtual DOM повністю існує в пам'яті і є репрезентацією веб-браузера DOM. Через це, коли ми пишемо компонент React, ми не пишемо безпосередньо до DOM, а пишемо віртуальний компонент, який перетворить React у DOM.

React використовується для створення веб-сайтів додатків на одній сторінці.

Односторінковий додаток (SPA) - це веб-додаток або веб-сайт, який взаємодіє з користувачем, динамічно переписуючи поточну сторінку, а не завантажуючи всі нові сторінки з сервера. Такий підхід дозволяє уникнути перешкод для користувальницької роботи між послідовними сторінками, зробивши програму схожішою на настільну програму. В SPA або весь необхідний код - HTML, JavaScript та CSS - витягується з завантаження однієї сторінки, або відповідні ресурси динамічно завантажуються та додаються на сторінку за потребою, як правило, у відповідь на дії користувача. Сторінка не перезавантажується в будь-який момент процесу і не контролює перенесення на іншу сторінку, хоча хеш-код API або API історії HTML5 можна використовувати для забезпечення сприйняття та навігації окремих логічних сторінок програми.

Взаємодія з односторінковою програмою часто включає динамічне спілкування з веб-сервером за кадром.

Оскільки SPA - це еволюція від моделі перемальовування сторінок, для якої браузери спочатку були розроблені, з'явилися нові проблеми. Кожна з цих проблем має ефективне рішення:

• Бібліотеки JavaScript на стороні клієнта, які вирішують різні проблеми.

• Серверні веб-рамки, що спеціалізуються на SPA-моделі.

• Еволюція браузера та специфікація HTML5, розроблені для моделі SPA.

Через відсутність JavaScript на сканерах деяких популярних веб-пошукових систем, SEO (оптимізація пошукових систем) історично була проблемою для публічних веб-сайтів, які хочуть прийняти модель SPA.

У період з 2009 по 2015 рік Google Webmaster Central запропонував, а потім рекомендував "схему сканування AJAX", використовуючи початковий знак оклику в ідентифікаторах фрагментів для сторінок AJAX. На веб-сайті SPA повинна бути реалізована особлива поведінка, щоб дозволити сканування відповідних метаданих сканером пошукової системи. Для пошукових систем, які не підтримують цю схему хешування URL-адрес, хешовані SPA-адреси залишаються невидимими. Багато авторів вважають такі URI-банди, які забороняють хеш-бани, включаючи Джені Теннісон у W3C, оскільки вони роблять сторінки недоступними для тих, хто не включив JavaScript у своєму браузері. Вони також порушують заголовки HTTP Referrer, оскільки браузерам заборонено надсилати ідентифікатор фрагмента в заголовку Referer. У 2015 році Google проігнорував її скануючу пропозицію AJAX.

Крім того, програми можуть відтворювати завантаження першої сторінки на сервер та подальші оновлення сторінки на клієнті. Це традиційно важко, тому що код візуалізації, можливо, потрібно буде записати іншою мовою або рамкою на сервері та клієнті. Використання шаблонів без логіки, перехресне компілювання з однієї мови на іншу або використання однієї мови на сервері та клієнті може допомогти збільшити кількість коду, яким ви можете поділитися.

Оскільки сумісність SEO не є тривіальною у SPA-режимах, слід зазначити, що SPZ зазвичай не використовуються в контексті, коли індексація пошукових систем є необхідною або бажаною. Випадки використання включають програми, які повертають приватні дані, приховані за системою аутентифікації. У випадках, коли ці програми є споживчими продуктами, часто використовується класична модель "перемальовування сторінок" для цільової сторінки додатка та маркетингового сайту, яка забезпечує достатню кількість метаданих, щоб програма відображалась як запит у пошуковому запиті. Блоги, форуми підтримки та інші традиційні артефакти для перемальовування сторінок часто сидять навколо СПА, що дозволяє насіння пошукових систем з відповідними термінами.

Інший підхід, використовуваний серверно-орієнтованими веб-рамками, такими як ItsNat, заснований на Java, полягає у відтворенні будь-якого гіпертексту на сервері, використовуючи ту саму мову шаблону та технології. При такому підході сервер точно знає стан DOM на клієнті, будь-які основні або незначні оновлення сторінки, необхідні, генеруються на сервері та передаються Ajax, точний код JavaScript, щоб привести сторінку клієнта до нового стану, який виконує Методи DOM. Розробники можуть вирішити, які сторінки сторінок слід сканувати веб-павуками для SEO, і мати можливість генерувати необхідний стан при завантаженні, генеруючи звичайний HTML замість JavaScript. У випадку з FrameworkNatNat це автоматично, тому що ItsNat зберігає клієнтське дерево DOM на сервері як дерево W3C Java DOM; надання цього дерева DOM на сервері генерує звичайний HTML під час завантаження та роботи JavaScript DOM для запитів Ajax. Ця подвійність дуже важлива для SEO, оскільки розробники можуть створювати за допомогою того самого коду Java та чистого шаблону HTML бажаний стан DOM на сервері; під час завантаження сторінки звичайний HTML створюється за допомогою ItsNat, що робить цей стан DOM сумісним із SEO. Починаючи з версії 1.3, ItsNat забезпечує новий режим без стану, а клієнт DOM не зберігається на сервері, оскільки з клієнтом режиму, який не бере участь, стан DOM частково або повністю реконструюється на сервері під час обробки будь-

якого запиту Ajax на основі необхідних даних надісланий клієнтом. інформування сервера про поточний стан DOM; Режим "без статусу" також може бути SEO-сумісним, оскільки SEO-сумісність виникає при завантаженні домашньої сторінки, що не залежить від режиму статусу або без громадянства.

Існує кілька способів вирішити те, як веб-сайт можна сканувати. Обидві передбачають створення окремих HTML-сторінок, які відображають зміст SPA. Сервер може створити HTML-версію сайту та доставити її до сканерів, або ви можете використовувати безголовий браузер, наприклад PhantomJS, для запуску JavaScript та виведення отриманого HTML.

Обидва вони вимагають досить великих зусиль і можуть врешті-решт принести головний біль великим складним предметам. Існують також потенційні підводні камені SEO. Якщо сервер, створений HTML, вважається занадто відмінним від контенту SPA, сайт буде накладений штраф. Запуск PhantomJS для виведення HTML може уповільнити частоту відповідей сторінок, через що пошукові системи, особливо Google, знижують рейтинг.

Одним із способів збільшити кількість коду, яким можна ділитися між серверами та клієнтами, є використання мови шаблонів без логіки, наприклад, вуса або ручки. Такі шаблони можуть відображатися з різних мов, наприклад, Ruby на сервері та JavaScript у клієнті. Однак просто обмін шаблонами зазвичай вимагає дублювання бізнес-логіки, що використовується для вибору правильних шаблонів і заповнення їх даними. Візуалізація шаблону може мати негативні ефекти на ефективність лише при оновленні невеликої частини сторінки - наприклад, значення введення тексту у великий шаблон. Заміна всього шаблону також може порушити вибір користувача або положення курсору, де оновлення лише зміненого значення може не відбуватися. Щоб уникнути цих проблем, програми можуть використовувати прив’язки даних інтерфейсу користувача або детальну маніпуляцію з DOM, щоб оновити лише відповідні частини сторінки замість повторного відображення цілих шаблонів.

Використовуючи SPA, за визначенням "одну сторінку", модель розбиває дизайн браузера для навігації по історії сторінок за допомогою кнопок Вперед / Назад. Це заважає використанню, коли користувач натискає кнопку "назад", очікуючи попереднього стану екрану в SPA, але замість цього одна сторінка сторінки програми завантажується і подається попередня сторінка історії браузера.

Традиційним рішенням SPA є зміна ідентифікатора хеш-фрагменту URL-адреси браузера відповідно до поточного стану екрана. Це може бути здійснено за допомогою JavaScript і запускає події історії URL-адрес у браузері. Поки SPA здатний відновити той самий стан екрану з інформації, що міститься в хеші URL-адреси, очікувана поведінка кнопки "назад" зберігається.

Для подальшого вирішення цієї проблеми специфікація HTML5 запровадила pushState та замінила стан, що забезпечує програмне забезпечення доступу до фактичної URL-адреси та історії браузера.

4.3. Фреймворк Leaflet

Leaflet - це широко використовувана бібліотека JavaScript з відкритим кодом, що використовується для створення веб-додатків для картографування. Вперше випущений у 2011 році, він підтримує більшість мобільних та настільних платформ, підтримуючи HTML5 та CSS3. Поряд з OpenLayers та API Google Maps, це одна з найпопулярніших бібліотек для відображення JavaScript і використовується великими веб-сайтами, такими як FourSquare, Pinterest та Flickr.

Leaflet дозволяє розробникам без фону ГІС легко відображати плиткові веб-карти, розміщені на загальнодоступному сервері, з додатковими накладками плиток. Він може завантажувати дані функції з файлів GeoJSON, стилювати їх та створювати інтерактивні шари, такі як спливаючі маркери.

Leaflet підтримує шари веб-служб (WMS), шари GeoJSON, векторні шари та шари плитки. Багато інших типів шарів підтримуються плагінами.

Як і інші бібліотеки веб-карт, основна модель відображення, реалізована Leaflet, є єдиною базовою картою, плюс нуль або більш прозорі накладки, з нулем або більше векторних об'єктів, що відображаються вгорі.

Елементи

Основними типами об'єктів є: растрові типи (TileLayer та ImageOverlay), векторні типи (Шлях, Полігон та конкретні типи, такі як коло), змішані типи (LayerGroup, FeatureGroup та GeoJSON) та компоненти управління (масштабування, шари тощо).

Існує також ряд корисних класів, таких як інтерфейси для управління проекціями, перетвореннями та взаємодія з DOM.

Leaflet прямо порівнянна з OpenLayers, оскільки вони обидва є відкритими, лише клієнтськими бібліотеками JavaScript. Бібліотека в цілому набагато менша, приблизно 7000 рядків коду порівняно з 230 000 OpenLayers (станом на 2015 рік). Він має менший розмір коду, ніж OpenLayers (близько 123 KB проти 423 KB), що частково пояснюється його модульною структурою. Кодова база є новішою та використовує найновіші функції JavaScript, а також HTML5 та CSS3. Однак Leaflet не має таких можливостей, які підтримує OpenLayers, таких як Служба веб-функцій (WFS) та вбудована підтримка проекцій, яка відрізняється від Google Web Mercator (EPSG 3857).

Його також можна порівняти із закритим вихідним кодом API Google Maps (дебютував у 2005 році) та API Bing Maps, обидва з яких мають важливий серверний компонент для надання таких послуг, як геокодування, маршрутизація, пошук та інтеграція з такими функціями, як Google Maps Google API забезпечує швидкість і простоту, але не є гнучким і може використовуватися лише для доступу до послуг Google Maps. Однак нова частина даних API DataLayer Google дозволяє відображати зовнішні джерела даних.

Для роботи з даними деяких регіонів на карті потрібно використовувати формат GeoJson. GeoJSON - це відкритий стандартний формат, призначений для

відображення простих географічних особливостей, а також їх непросторових атрибутів. Він заснований на JSON, позначенні об’єктів JavaScript.

Основним елементом географічних даних є координати. Це єдине число, яке представляє один вимір: зазвичай розміри - це довгота та широта. Іноді також існує координата по висоті. Час - це вимір, але зазвичай не відображається в координаті, оскільки він занадто складний, щоб вмістити число.

Координати в GeoJSON відформатовані як числа в JSON: у простому десятковому форматі. На відміну від географічних даних для споживання людиною, формати даних ніколи не використовують кодування, що не стосується бази даних, наприклад, сексуальне. Холодно, як 8 ° 10 '23', це просто не дуже вдалий спосіб виявити цифри на комп’ютерах.

Позиція - це масив координат: це найменша одиниця, яку ми можемо реально вважати "місцем", оскільки вона може представляти точку на землі. GeoJSON описує порядок координат: вони повинні йти у такій послідовності: довгота, широта, висота.

Цей порядок може бути дивним. Історично порядок координат зазвичай "широта, довгота", і багато людей вважають, що цей випадок є універсальним. Питання, які краще обговорювались протягом багатьох годин, але для цього обговорення я підсумую наступне:

• Довгота, широта відповідає порядку X, Y математики

• Формати даних зазвичай використовують порядок довготи, широти

• Програми, як правило, використовують широту, довготу

Перш ніж випускати поточну специфікацію, GeoJSON дозволяв зберігати більше 3 координат на кожну позицію, а іноді люди використовували її для зберігання часу, частоти серцевих скорочень тощо. Цей інструмент належним чином не підтримується в інструментах GeoJSON і заборонений новою специфікацією.

З точки зору продуктивності вам потрібно відразу зрозуміти, де у вашій системі є вузьке місце. Наприклад, якщо у вас є класичні проблеми з конфігурацією та працездатністю GeoJSON Leaflet, вузьке місце майже завжди є мережею або SVG.

Якщо це продуктивність SVG - вартість листівки для малювання полігонів та ліній у вашому браузері - тоді формат файлу не має значення. Передайте ті самі дані в ультраефективному форматі, і ви все одно отримаєте повільну карту.

Припустимо, що час у мережі є вузьким місцем: файл GeoJSON становить 20 Мб і завантажується за 20 секунд. Підходи до вирішення таких питань частіше, ніж будь-який формат файлу: стиснення втрат, завантаження підмножини, потокове передавання.

4.4. Карта OpenStreetMap

OpenStreetMap (OSM) - спільний проект зі створення вільної карти для редагування світу. Дані, згенеровані проектом, замість самої карти, вважаються його первинним результатом. Створення та зростання ОСМ мотивовано обмеженнями використання або доступності картографічної інформації в більшості країн світу, а також появою недорогих портативних супутникових навігаційних пристроїв. OSM вважається яскравим прикладом добровільної географічної інформації.

Створений Стівом Коуст у Великобританії у 2004 році, він надихнувся успіхом Вікіпедії та переважанням власних даних карт у Великобританії та інших країнах. З тих пір вона зросла до понад двох мільйонів зареєстрованих користувачів, які можуть збирати дані за допомогою ручних опитувань, GPS-пристроїв, аерофотозйомки та інших безкоштовних джерел. Дані, надані через посередника, передаються за ліцензією Open Database. Сайт підтримує Фонд OpenStreetMap, неприбуткова організація, зареєстрована в Англії та Уельсі.

Дані OSM доступні для використання як у традиційних програмах, таких як Facebook, Craigslist, OsmAnd, Geocaching, MapQuest Open, програмне забезпечення статистики JMP та Foursquare для заміни Google Maps, так і в більш незвичних ролях, таких як заміна даних за замовчуванням. GPS приймачі. Дані OpenStreetMap позитивно порівнюються з власними джерелами даних, хоча в 2009 році якість даних відрізнялася у всьому світі

Ці карти складаються з нуля добровольцями, які проводять систематичні опитування на місцях за допомогою таких інструментів, як ручний GPS-пристрій, ноутбук, цифрова камера чи диктофон. Дані вводяться в базу даних OpenStreetMap. Марафони також проводяться командою OpenStreetMap та некомерційними організаціями та органами місцевого самоврядування, щоб відобразити конкретну область.

Наявність аерофотозйомки та інших даних з комерційних та державних джерел додала важливих джерел даних для ручного редагування та автоматизованого імпорту. Існують спеціальні процеси для автоматичного імпорту та уникнення юридичних та технічних проблем.

Редагування карт можна здійснити за допомогою редактора браузера за замовчуванням під назвою iD, програми HTML5, яка використовує D3.js, і письмового Mapbox, який спочатку фінансувався Knight Foundation. Для проміжних користувачів зберігається раніше використовувана програма Flashlat Potlatch. JOSM та Merkaartor - це більш потужні програми для редагування настільних ПК, які краще підходять для досвідчених користувачів.

Vespucci - перший повнофункціональний редактор для Android; це було випущено в 2009 році. StreetComplete - це нова, зручна для користувачів програма Android, запущена в 2016 році, яка дозволяє користувачам без будь-яких знань OpenStreetMap відповідати на прості квести наявних даних у OpenStreetMap і таким чином полегшувати передачу даних. Maps.me - це мобільний додаток (працює на платформах Android та iOS) і пропонує окремі карти, які також містять обмежений редактор даних OSM. Перейдіть на карту !! це програма iOS, яка дозволяє створювати та редагувати інформацію в OpenStreetMap. Pushpin - ще одна програма для iOS, яка дозволяє додавати POI в дорозі.

Проект має географічно різноманітну базу користувачів, завдяки акцентуванню на місцевих знаннях та основних правах у процесі збору даних. Багато ранніх учасників були велосипедистами, які оглядали велосипедистів та переглядали велосипедні маршрути та судна. Інші - професіонали ГІС, які надають

дані за допомогою інструментів Esri. Співробітники - це переважно чоловіки, і лише 3-5% - жінки.

До серпня 2008 року, незабаром після другої Державної конференції на карті, було зареєстровано понад 50 000 учасників; до березня 2009 року їх було 100 000, а до кінця 2009 року ця цифра становила майже 200 000. У квітні 2012 року OpenStreetMap звільнив 600 000 зареєстрованих членів. 6 січня 2013 року OpenStreetMap досяг мільйона зареєстрованих користувачів. Близько 30% користувачів внесли хоча б один елемент у базу даних OpenStreetMap.

Наземні огляди проводяться картографом, пішки, велосипедом, автомобілем, мотоциклом чи човном. Карти даних збираються зазвичай за допомогою GPS-пристрою, хоча це не обов'язково, якщо область вже відстежується із супутникових знімків.

Після збору даних вони вводяться в базу даних, завантажуючи їх на веб-сайт проекту разом з відповідними даними атрибутів. Оскільки збір та завантаження даних можуть бути відокремлені від редагування об'єктів, внесок у проект можливий без використання пристрою GPS.

Деякі зацікавлені сторони зобов'язуються порівнювати цілі міста та міста або організовувати картографічні картки для отримання підтримки інших для складання карти. Велика кількість менш активних користувачів вносить коригування та невеликі доповнення до карти.

На додаток до декількох різних наборів фонів супутникових зображень, доступних редакторам OSM, дані з декількох платформ зображення на вулиці доступні у вигляді даних накладання фотографій: доріжки зображення Bing Streetside 360º, а також відкриті та краудсорсингові платформи Mapillary і OpenStreetCam, як правило, смартфони та інші зображення, встановлені на камеру. Крім того, може бути включений шар даних дорожніх знаків Mapillary; є продуктом поданих користувачем зображень.

Примітивні дані OSM зберігаються та обробляються в різних форматах. Резервна копія даних OSM зберігається в основній базі даних OSM. Основна база

даних - це база даних PostgreSQL з розширенням PostGIS, в якій є одна таблиця для кожного примітиву даних, окремі об'єкти зберігаються як рядки. Усі редагування відбуваються в цій базі даних, а всі інші формати створюються з неї.

Для передачі даних було створено декілька скидів у базу даних, які доступні для завантаження. Повне сміттєзвалище називається planet.osm. Ці демпінги бувають у двох форматах: один використовує XML, а другий використовує двійковий формат буфера протоколу (PBF).

Дані LinkedGeoData використовують словники GeoSPARQL та відомий текстовий (WKT) RDF для представлення даних OpenStreetMap. Це робота дослідницької групи Agile Knowledge Engineering and Semantic Web (AKSW) в Лейпцизькому університеті, групи, яка в основному відома DBpedia.

4.5. Мова TypeScript

TypeScript - мова програмування з відкритим кодом, розроблена та підтримувана Microsoft. Це суворий синтаксичний набір JavaScript і додає мові додаткове статичне введення тексту.

TypeScript призначений для розробки великих програм та перекомпільований у JavaScript. Оскільки TypeScript - це набір JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript. TypeScript може використовуватися для розробки програм JavaScript на стороні клієнта та сервера (Node.js).

Існує кілька варіантів транскопіляції. Ви можете використовувати типовий перевіряючий засіб TypeScript або зателефонувати на компілятор Babel для перетворення TypeScript у JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про типи існуючих бібліотек JavaScript так само, як файли заголовків C ++ можуть описувати структуру існуючих об'єктних файлів. Це дозволяє іншим програмам використовувати значення, визначені у файлах, як ніби вони були статично типізованими сутностями TypeScript. Існують сторонні файли заголовків для

популярних бібліотек, таких як jQuery, MongoDB і D3.js. Заголовки TypeScript також доступні для базових модулів Node.js, які дозволяють розробляти додатки Node.js у TypeScript.

Сам компілятор TypeScript написаний у TypeScript та компільований у JavaScript. Він ліцензований під ліцензією Apache 2.0. TypeScript включений як основна мова програмування в Microsoft Visual Studio 2013, оновлення 2 та новіших версій, разом із C # та іншими мовами Microsoft. Офіційне розширення дозволяє Visual Studio 2012 також підтримувати TypeScript.

TypeScript випливає з недоліків JavaScript для розробки масштабних додатків як від Microsoft, так і від їх зовнішніх клієнтів. Проблеми зі складним кодом JavaScript призвели до необхідності користувацьких інструментів для полегшення розробки мовних компонентів.

Розробники TypeScript шукали рішення, яке не порушило б сумісність зі стандартом та його підтримкою між платформами. Знаючи, що поточна стандартна пропозиція ECMAScript обіцяє майбутню підтримку програмування на основі класу, TypeScript покладався на цю пропозицію. Це призвело до компілятора JavaScript з набором синтаксичних розширень на основі речень, який перетворює розширення у звичайний JavaScript. У цьому сенсі TypeScript був попереднім переглядом того, чого слід очікувати від ECMAScript 2015. Унікальний аспект, який не включений у пропозицію, а доданий до TypeScript, - це додаткове статичне введення тексту, що дозволяє статичний аналіз мови, що полегшує інструменти та підтримку IDE.

Компілятор TypeScript під назвою tsc записаний у TypeScript. Як результат, його можна скласти в звичайний JavaScript і потім виконати в будь-якому механізмі JavaScript (наприклад, у браузері). Пакет компілятора постачається із хостом сценарію, який може виконати компілятор. Він також доступний як пакет Node.js, який використовує Node.js як хост.

Існує також альфа-версія компілятора на стороні клієнта в JavaScript, який виконує код TypeScript під час руху під час завантаження сторінки.

Поточна версія компілятора за замовчуванням підтримує ECMAScript 5. Ви можете націлити на ECMAScript 2015, щоб використовувати мовні функції, що ексклюзивні для цієї версії (наприклад, генератори). Класи, хоча є частиною стандарту ECMAScript 2015, доступні в обох режимах.

TypeScript - це доповнення до ECMAScript 2015 із суворим введенням тексту, яке саме по собі є набором ECMAScript 5, який зазвичай називають JavaScript. Таким чином, JavaScript також є дійсною програмою TypeScript, а TypeScript може вільно споживати JavaScript. За замовчуванням компілятор націлений на ECMAScript 5, поточний стандарт, який переважає, але також може генерувати конструкції, використовувані в ECMAScript 3 або 2015.

За допомогою TypeScript ви можете використовувати існуючий код JavaScript, включати популярні бібліотеки JavaScript та код виклику, згенерований за допомогою TypeScript з іншого JavaScript. Декларації типів для цих бібліотек забезпечуються вихідним кодом. Анотації для примітивних типів - це числа, булеві та рядки. Слабо або динамічно типізовані структури мають будь-який тип.

Типи анотацій можна експортувати в окремий файл декларації для надання інформації про тип скриптам TypeScript, використовуючи типи, вже зібрані в JavaScript. Анотації можна оголосити для існуючої бібліотеки JavaScript, як це було зроблено для Node.js та jQuery.

Компілятор TypeScript використовує тип виводу для типу, коли не передбачено типів. Наприклад, метод додавання у наведеному вище коді буде визначений як номер повернення, навіть якщо анотація типу повернення не надана. Він заснований на статичних типах лівого і правого чисел, а також на знаннях укладача, що результатом додавання двох чисел завжди є число. Однак явна декларація типу повернення дозволяє компілятору переконатися, що вона правильна.

Якщо жоден тип не може бути виведений через відсутність декларацій, то за замовчуванням він буде динамічним для будь-якого типу. Значення будь-якого типу

підтримує ті самі операції, що і значення JavaScript, і мінімальна статична перевірка типу виконується для операцій з будь-яким значенням.

Під час компіляції сценарію TypeScript можна створити файл декларації (з розширенням .d.ts), який функціонує як інтерфейс для компонентів у складеному JavaScript. Під час роботи компілятор видаляє всі елементи функцій і методів і зберігає лише підписи тих типів, які експортуються. Отриманий файл декларації може бути використаний для опису експортованих типів віртуальної бібліотеки TypeScript або модуля JavaScript, коли сторонній розробник споживає його разом з TypeScript.

Поняття файлів декларації схоже на концепцію файлу заголовка, знайденого в C / C ++.

Файли типу оголошень можна записати вручну для існуючих бібліотек JavaScript, як це було у випадку з jQuery та Node.js.

TypeScript розрізняє модулі та простори імен. Обидві функції в TypeScript підтримують інкапсуляцію класів, інтерфейсів, функцій та змінних в контейнери.

4.6. Система керування версіями Git

Дуже зручно використовувати системи управління версіями, у цій роботі була використана система Git. Це дозволяє покращити швидкість, цілісність даних та підтримку розподілених нелінійних робочих процесів.

Git - це система управління розподіленою версією для відстеження змін у вихідному коді під час розробки програмного забезпечення. Він використовується для координації роботи між програмістами, але може використовуватися для відстеження змін у будь-якій групі файлів. Git був розроблений в 2005 році для розробки ядра Linux, а інші розробники ядра внесли свій внесок у його початкову розробку.

Як і більшість інших розподілених систем управління версіями та на відміну від більшості клієнтсько-серверних систем, кожен каталог Git на кожному

комп’ютері є повноцінним сховищем з повною історією та функцією відстеження версій, незалежно від доступу до мережі чи центрального сервера.

Git є цілком безкоштовним та поширюється на умовах Загальної публічної ліцензії GNU.

Основні елементи Git насправді не є системою управління джерелами. Багато в чому ви можете бачити git лише як файлову систему - це адресація вмісту та концепція версії. Git розробив низку функцій, які очікуються від традиційних SCM. Ці функції зазвичай створюються за потребою, а потім вдосконалюються та розширюються з часом.

Git має дві структури даних: змінний індекс (також відомий як рівень або кеш), який кешує інформацію про робочий каталог.

Індекс служить ланкою між об’єктною базою даних та робочим середовищем.

Об'єктна база даних містить чотири типи об'єктів: файли (blolb), дерево (дерево), коміти та теги.

Кожен об'єкт ідентифікується хешем його вмісту SHA-1. Git обчислює хеш і використовує це значення для імені об'єкта. Об'єкт зберігається в каталозі, який відповідає першим двом символам його хешу. Решта хешу використовується як ім'я файлу для цього об’єкта.

Git зберігає кожну версію файлу як унікальний блок. Нещодавно додані об'єкти повністю зберігаються за допомогою стиснення zlib. Це може швидко зберегти багато місця, тому об’єкти можна об’єднати в пакети, які використовують дельта-стиснення для економії місця.

4.7. Висновки до розділу

У цьому розділі розглядаються основні засоби розробки для написання веб-інтерфейсу користувача та взаємодії з інтерактивною веб-картою. Був проведених аналіз та їх порівняння з аналогічними бібліотеками та структурами, і їх використання в цій роботі було обґрунтовано.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі описується архітектура програмної реалізації. Спосіб взаємодії з сторонніми сервісами та методика роботи користувача з системою.

5.1. Опис архітектури системи

Додаток буде складатися з веб-сторінки, де користувач матиме доступ до інтерактивної карти та панелі обчислень. Користувач може масштабувати карту. Усі елементи керування будуть на бічній панелі. Такий підхід зробить інтерфейс максимально інтуїтивним (Рисунок 5.1).

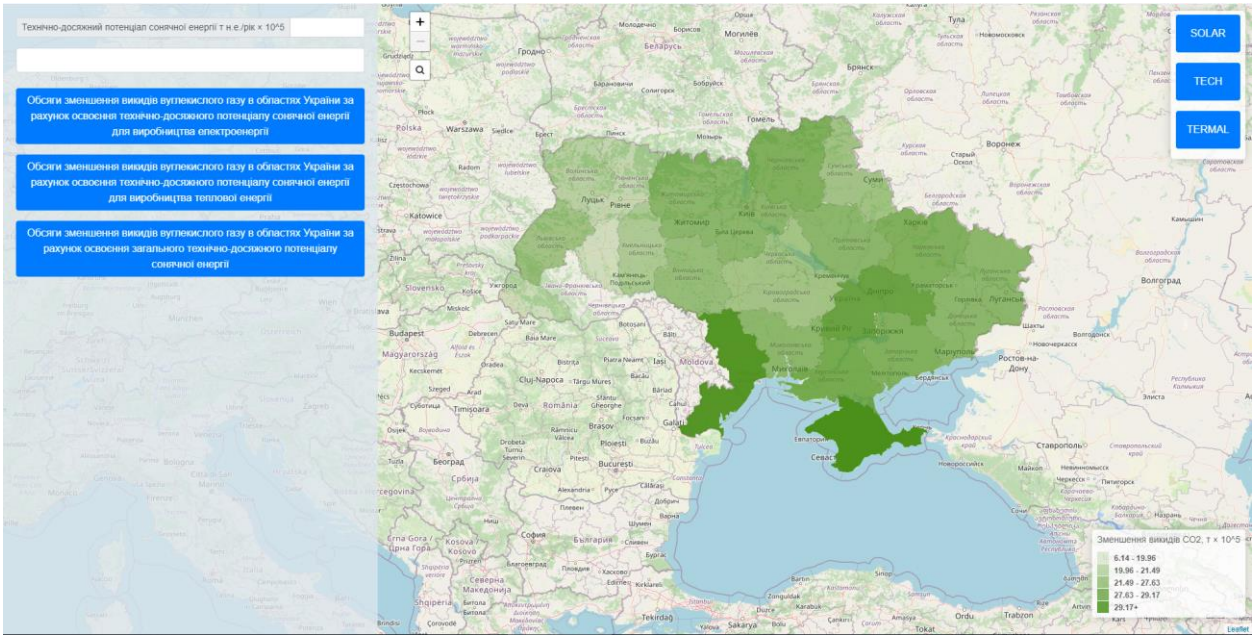

Рисунок 5.1 — Основна сторінка додатку

Система складається з одного актора, який має можливість використовувати веб-карту, аналізувати регіони та потенціал сонячної енергії в Україні (рис. 5.2).

Рисунок 5.2 — Діаграма прецедентів системи

На карті розміщені три кнопки (Рисунок 5.3) що дозволяють інтерактивно відображати обсяги зменшення викидів вуглекислого газу в областях України за рахунок відповідного технічно-досяжного потенціалу. Solar відповідає потенціалу сонячної енергії для виробництва електроенергії, Tech відповідає освоєнню загального технічно-досяжного потенціалу сонячної енергії, Termal - освоєння технічно-досяжного потенціалу сонячної енергії для виробництва теплової енергії.

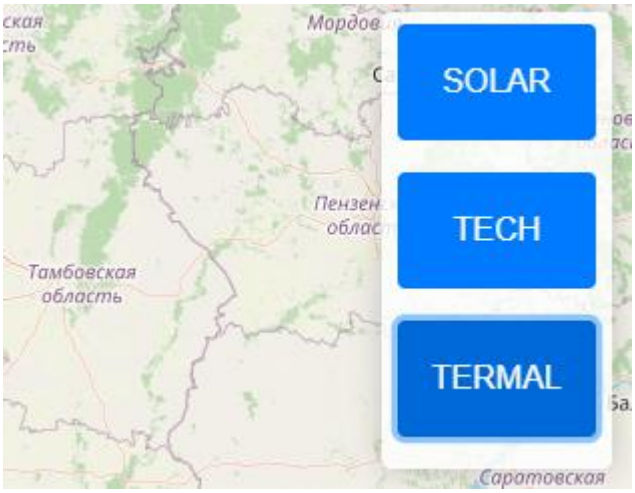

Рисунок 5.3 — Кнопки карти

Натискання відповідних кнопок оновлює інформацію на карті що свідчить про обсяги зменшення викидів вуглекислого газу різних типів (Рисунок 5.4).

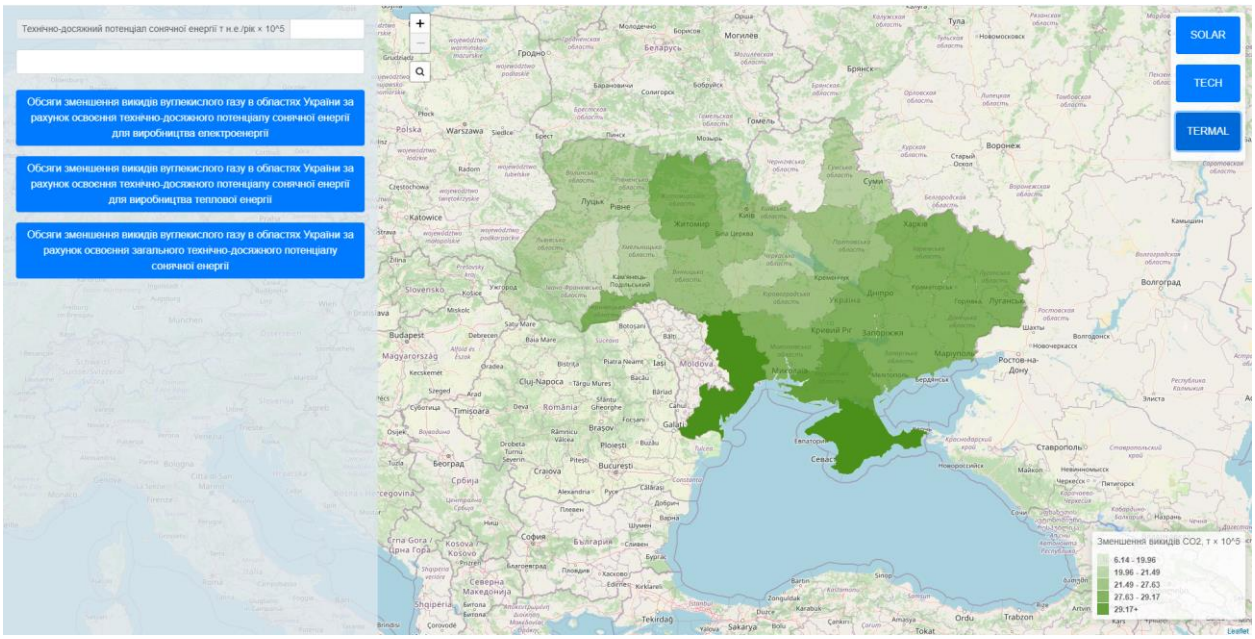

Рисунок 5.4 — Оновлена карта

Бокова панель містить кнопки, що забезпечують вивід інформації про обсяги зменшення викидів вуглекислого газу у вигляді таблиць (Рисунок 5.5)

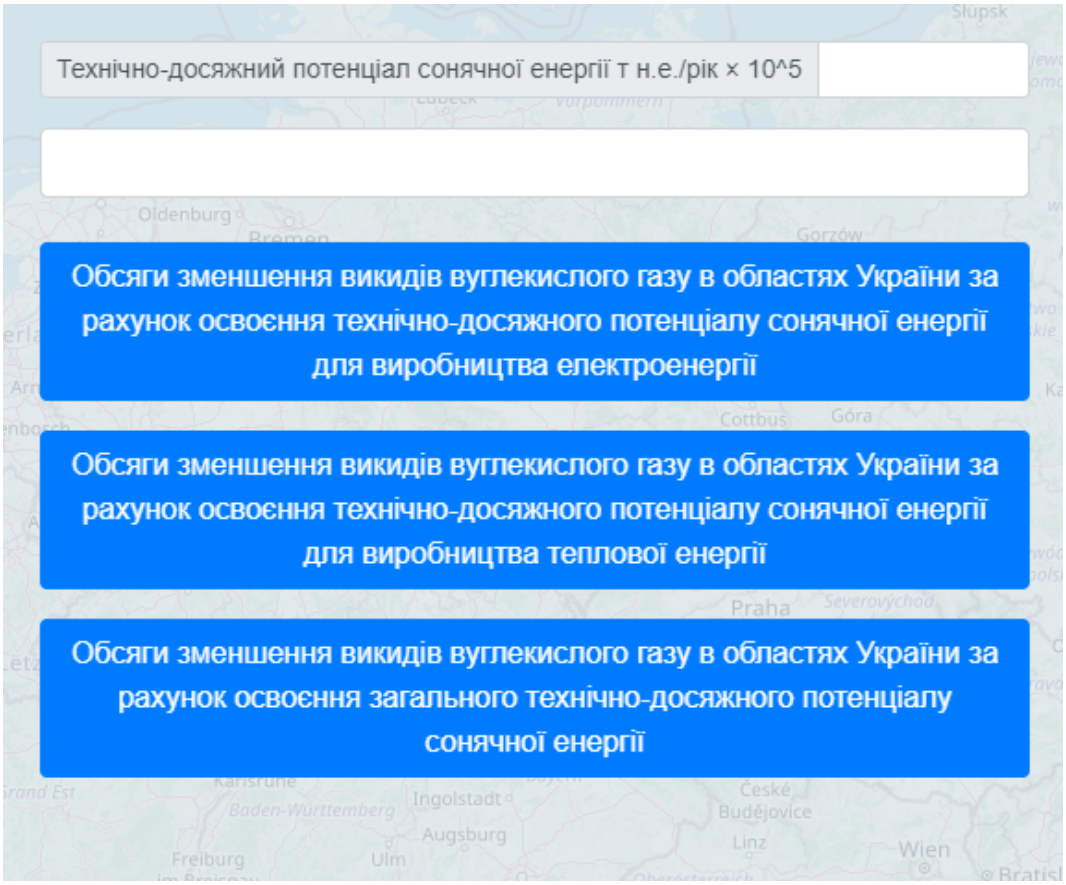

Рисунок 5.5 — Бокова панель

Вигляд таблиці для “Обсяги зменшення викидів вуглекислого газу в областях України за рахунок освоєння технічно-досяжного потенціалу сонячної енергії для виробництва електроенергії” (Рисунок 5.6).

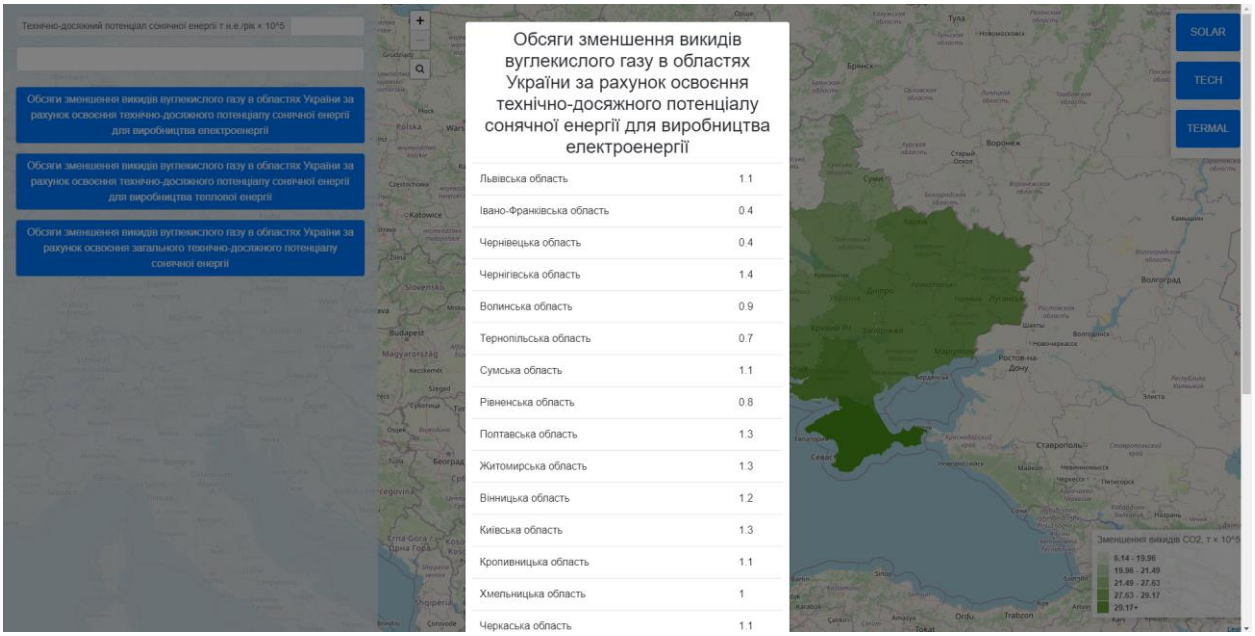

Рисунок 5.6 — Приклад таблиці

Також користувач може бачити шкалу зафарбованості (Рисунок 5.7).

Рисунок 5.7 — Зменшення викидів CO2, т × 10^5

5.2. Розробка сторінки інтерактивної веб-карти

Цей модуль дозволяє користувачеві контролювати карту і робити розрахунки.

Перш за все, було створено файл JSON, який відповідає специфікації GeoJson. Він позначає координати кожного регіону України. Назва регіону визначається стандартом ISO3166-2.

Наступним кроком було ініціалізація інтерактивної карти, з'єднавши рамку lefleat.js та карту OpenStreetMap. Назва області також відповідає специфікації IchSO3166-2. Об'єкту карти було надано список областей у форматі GeoJson.

За допомогою іншого компонента було створено бічну панель, яка дозволяє розрахувати кількість скорочення викидів вуглекислого газу в регіонах України за рахунок відповідного технічно досяжного потенціалу.

Реалізація складається з клієнтської та серверної частини. Для їх взаємодії було створено веб-службу. Було реалізовано взаємодію з сервером через API (інтерфейс прикладної програми). Це дає можливість отримати дані про регіони України. Ці дані поєднувались із форматом області та додаються до інтерактивної карти.

При ініціалізації карти робиться запит за допомогою API, який повертає дані за регіонами (рис. 5.8).

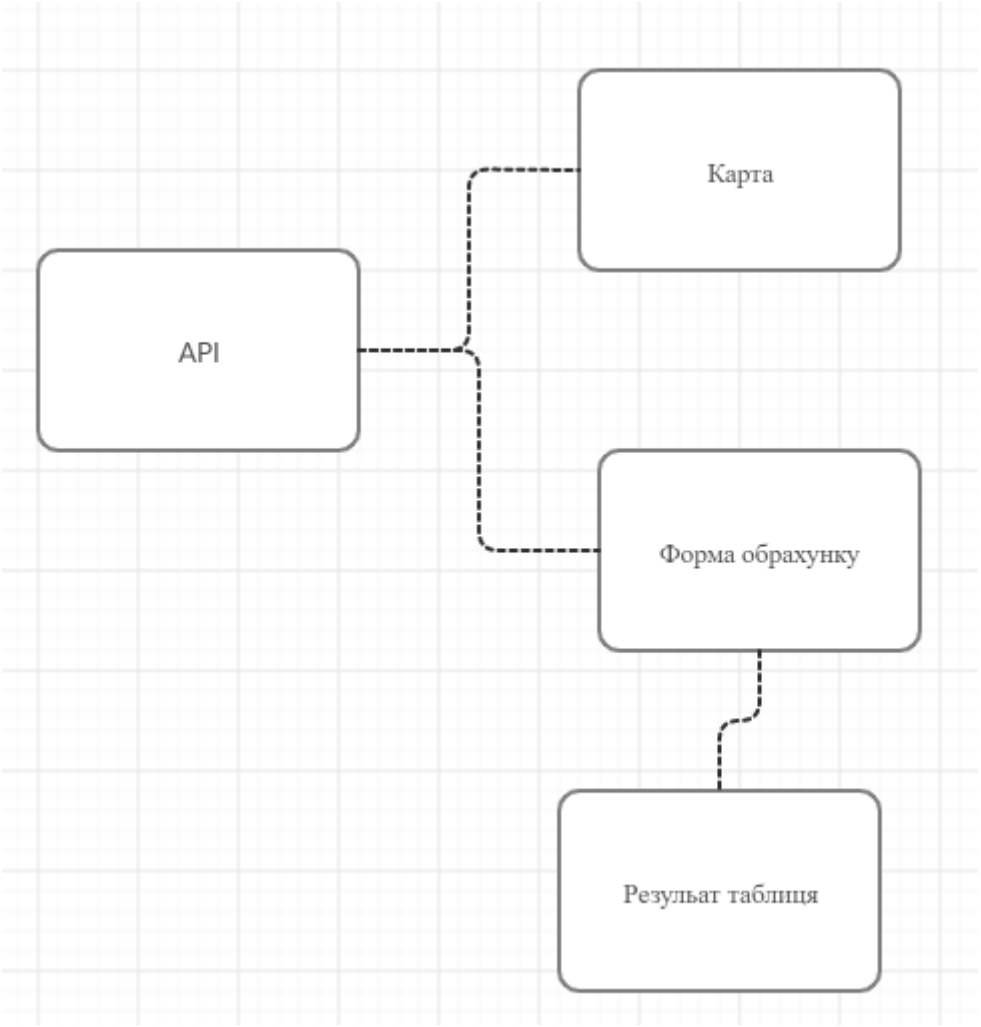

Рисунок 5.8 — Схема структури системи

Головна сторінка надає користувачеві всі можливості проекту та складається з декількох компонентів. Компоненти - це заняття. Їх взаємодія зроблена так, щоб було легко додати нові типи станцій та розробити нову логіку обчислення даних (рис. 5.9).

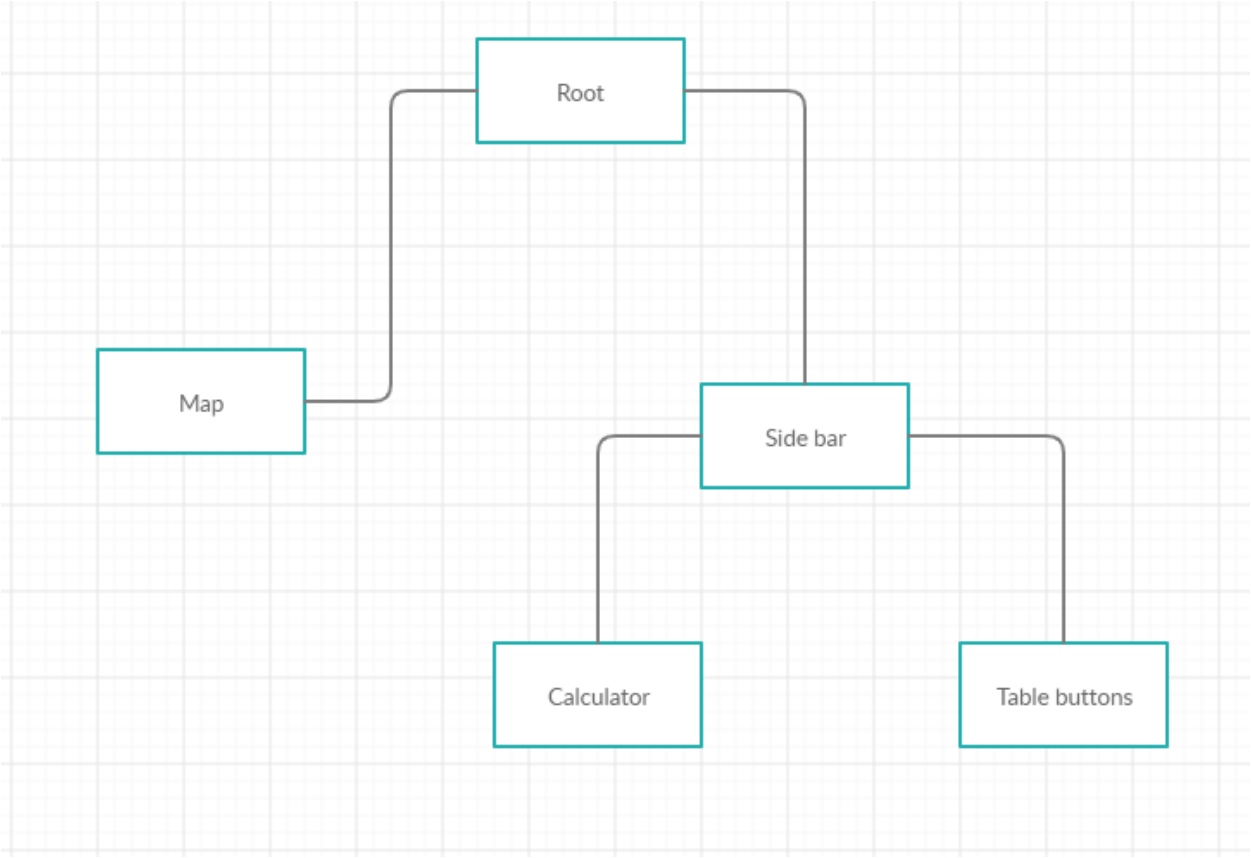

Рисунок 5.9 — Компоненти системи

5.3. Висновки до розділу

У цьому розділі розглядаються деталі впровадження системи, способи взаємодії між різними службами та компонентами, а також показано, як користувач повинен працювати з додатком.

ВИСНОВКИ

В ході аналізу існуючого програмного забезпечення для аналізу та визначення кількості викидів вуглекислого газу при освоєнні енергетичного потенціалу сонячної активності були визначено, що існуючі програмні комплекси незручні у використанні, не вирішують проблему в повному обсязі або не призначені для вирішення поставленої задачі.

Розроблено веб-додаток, який дозволить користувачеві взаємодіяти з інтерактивною картою в Інтернеті, ставити маркери, додавати інформацію. Крім того, користувач має можливість використовувати онлайн-калькулятор для підрахунку кількості викидів за довільними даними. Проведено аналіз методів та засобів розробки програмних систем. Вибір створення програмної системи на основі веб-технологій, а також вибір трирівневої архітектури, дозволив підвищити інтелектуальність та зручність використання системи, як при розробці, так і у використанні.

За результатами тестувань було підтверджено коректність отриманих даних, тому система відповідає вимогам. Користувачами системи може бути будь хто зацікавлений в енергетичних системах сонячного типу. Програмне забезпечення можна використовувати на будь-якій операційній системі, де встановлений браузер, який підтримує сучасні веб-стандарти та має доступ до Інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Павловская Т. А. C#. Программирование на языке высокого уровня / Т. А. Павловская., 2014. – 432 с

2. ESRI. ArcGIS [Електронний ресурс] / ESRI – Режим доступу до ресурсу: https://www.arcgis.com/features/features.html.

3. Ukrstar. Державна служба статистики України [Електронний ресурс] / Ukrstar – Режим доступу до ресурсу: http://www.ukrstat.gov.ua.

4.Атлас енергетичного потенціалу нетрадиційних та відновлюваних джерел енергії. Під ред. Н.М. Мхітаряна. – Київ: ТОВ «Видавництво Вікторія»,-2012. − 61с.

5.Кудря С. О. Нетрадиційні та відновлювані джерела енергії . Підручник. Національний технічний університет України («КПІ»). Київ. 2012. − 495 с.

6.Адам Фримен — ASP.NET Core MVC с примерами на C# для профессионалов [Електронний ресурс]. — 2017. — Режим доступу: https://bit.ly/2JbSgHe.

7. React в действии - Марк Тиленс Томас [Книга]

8. Рєзцов В.Ф., Матях С.В., Кудреватих О.О. Інтерактивна карта потенціалу сонячної енергії України /Відновлювана енергетика. – 2018.

9. Болье А. — Learning SQL [Електронний ресурс]. — 2005. — Режим доступу: http://shop.oreilly.com/product/9780596007270.do.

10. Твайделл Дж., Уэйр А. Возобновляемые источники энергии. – М.: Энергоатомиздат. 1990. – 344 с.

11. Автоматизація бізнес процесів…. // Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського, 2020. – Т. 2. – 160 с.

ДОДАТОК А

Інтерактивна веб-карта для представлення викидів вуглекислого газу при освоєнні потенціалу сонячної енергії по областям України (розробка веб-інтерфейсу користувача).

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62205_20Б-1

Аркушів 1

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62205_20Б-1	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ 62205_20Б-2	solarMap.jsx solarCalculator.jsx	Основні компоненти
УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ 62205_20Б-3	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Інтерактивна веб-карта для представлення викидів вуглекислого газу при освоєнні потенціалу сонячної енергії по областям України (розробка веб-інтерфейсу користувача).

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ62205_20Б-2

Аркушів 10

Київ 2020

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using AutoMapper;
using Bs.Cms.Core.Localization.Routing;
using MapUA.Core.EntityFramework;
using MapUA.Core.EntityFramework.Models;
using MapUA.Core.Repo;
using MapUA.Core.ViewModels;
using MapUA.WebApp.ViewModels;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace MapUA.WebApp.Controllers
{
    [Produces("application/json")]
    [Route("marks")]
    [ApiController]
    public class MarksController : ControllerBase
    {
        private readonly IMapRepo _mapRepo;
        private readonly IMapper _mapper;

        public MarksController(IMapRepo mapRepo, IMapper mapper)
        {
            _mapRepo = mapRepo;
            _mapper = mapper;
        }

        [HttpGet]
        [Route("getAll")]
        public async Task<MapVM> GetAll()
        {
            var          regions          =          _mapper.Map<IEnumerable<RegionVM>>(await
_mapRepo.GetAllRegions());
            var markers = _mapper.Map<IEnumerable<MarkVM>>(await _mapRepo.GetAllMarks());
            var angles = _mapper.Map<IEnumerable<AngleVM>>(await _mapRepo.GetAllAngles());
            var         solar         =         _mapper.Map<IEnumerable<SolarStationVM>>(await
_mapRepo.GetSolarStations());
            var         helio         =         _mapper.Map<IEnumerable<HelioStationVM>>(await
_mapRepo.GetHelioStations());



            var minSolar = regions.Min(x => x.SolarPotential) * 11.63m * 1.32m;
            var maxSolar = regions.Max(x => x.SolarPotential) * 11.63m * 1.32m;

            var minTech = regions.Min(x => x.TechPotential) * 11.63m * 1.32m;
            var maxTech = regions.Max(x => x.TechPotential) * 11.63m * 1.32m;

            var minTermal = regions.Min(x => x.TermalPotential) * 11.63m * 1.32m;
            var maxTermal = regions.Max(x => x.TermalPotential) * 11.63m * 1.32m;


            regions = regions.Select(c =>
            {
                c.SolarPotentialPercentage = CalculatePercentage(c.SolarPotential, minSolar,
maxSolar);
                c.TechPotentialPercentage  =  CalculatePercentage(c.TechPotential,  minTech,
maxTech);
```

```
                c.TermalPotentialPercentage     =     CalculatePercentage(c.TermalPotential,
minTermal, maxTermal);

                return c;

            }).ToList();


            return new MapVM
            {
                Markers = markers,
                Regions = regions,
                Angles = angles,
                SolarStations = solar,
                HelioStations = helio,
            };
        }

        private decimal? CalculatePercentage(decimal? SolarPotential, decimal? min, decimal?
max)
        {
            var Co2 = SolarPotential * 11.63m * 1.32m;

            return (Co2 - min) / (max - min);
        }

        [HttpPost]
        [Route("addMark")]
        public async Task<ActionResult> AddMark(MarkVM model)
        {
            if (await _mapRepo.AddMark(_mapper.Map<Mark>(model)) == null)
            {
                ModelState.AddModelError("", "Mark with these coordinates already exist");
                return BadRequest(ModelState);
            }
            return Ok();
        }
    }
}
```

```
import React, { Component } from "react";
import ReactDOM from "react-dom";
import Control from "react-leaflet-control";
import { Map, TileLayer, Marker, Popup, GeoJSON } from "react-leaflet";
import getLocation from "../services/geolocation.js";
import L from "leaflet";
import ukraineGeo from "../geoJson/Ukraine.json";
import otherCountriesGeo from "../geoJson/countriesArroundUkraine.json";
import "../assets/map.css";
import $ from "jquery";

export default class SolarMap extends Component {
  constructor(props) {
    super(props);
    this.state = {
      userMarker: props.userMarker,
      markers: [],
      regions: [],
      angles: [],
      solarStations: [],
      helioStations: [],
```

```
    colorStep: 0,
    lat: 48.55,
    lng: 31,
    zoom: 6,
    components: null,
    displayPreloader: true,
    mapElemnt: null
  };

  this.setUkraineStyle = this.setUkraineStyle.bind(this);
  this.onEachFeature = this.onEachFeature.bind(this);
  this.setColorSolar = this.setColorSolar.bind(this);
  this.setColorTech = this.setColorTech.bind(this);
  this.setColorTermal = this.setColorTermal.bind(this);
  this.onEachFeatureSolar = this.onEachFeatureSolar.bind(this);
  this.onEachFeatureTech = this.onEachFeatureTech.bind(this);
  this.onEachFeatureTermal = this.onEachFeatureTermal.bind(this);

}

componentDidMount() {
  const { withLeaflet } = require("react-leaflet");
  const { ReactLeafletSearch } = require("react-leaflet-search");
  this.setState({
    components: { ReactLeafletSearch: withLeaflet(ReactLeafletSearch) }
  });

  fetch("http://localhost:17725/marks/getAll")
    .then(data => data.json())
    .then(map => {
      this.setState({
        markers: [],
        regions: map.regions,
        angles: map.angles,
        solarStations: map.solarStations,
        helioStations: map.helioStations,
        colorStep: 1.0 / (map.regions.length + 1)
      });

      getLocation().then(location => {
        this.setState({
          userMarker: {
            region: this.state.regions.find(el => el.iso === "UA-32"),
            coordinates: location,
            isNew: true,
            systemStationId: 1,
            angles: map.angles,
            solarStations: map.solarStations,
            helioStations: map.helioStations
          },
          displayPreloader: false
        });
        this.props.locationCallBack(this.state.userMarker);
      });
    });
}


setColorSolar() {

  var map = this.state.mapElemnt.leafletElement;
```

```
  map.eachLayer(function (layer) {
    if (!!layer.toGeoJSON) {
      map.removeLayer(layer);
    }
  });

  var stl = {
    color: "#4A901A",
    opacity: 0,
    fillOpacity: 0.5
  };

  var geojson = L.geoJson(ukraineGeo, {
    style: stl,
    onEachFeature: this.onEachFeatureSolar

  }).addTo(map);

}


setColorTech() {

  var map = this.state.mapElemnt.leafletElement;

  map.eachLayer(function (layer) {
    if (!!layer.toGeoJSON) {
      map.removeLayer(layer);
    }
  });

  var stl = {
    color: "#4A901A",
    opacity: 0,
    fillOpacity: 0.5
  };

  var geojson = L.geoJson(ukraineGeo, {
    style: stl,
    onEachFeature: this.onEachFeatureTech

  }).addTo(map);

}



setColorTermal() {

  var map = this.state.mapElemnt.leafletElement;

  map.eachLayer(function (layer) {
    if (!!layer.toGeoJSON) {
      map.removeLayer(layer);
    }
  });

  var stl = {
    color: "#4A901A",
    opacity: 0,
```

```
      fillOpacity: 0.5
    };

    var geojson = L.geoJson(ukraineGeo, {
      style: stl,
      onEachFeature: this.onEachFeatureTermal

    }).addTo(map);

  }

  setUkraineStyle(feature) {
    let colorOpac = 0;
    let     regionIndex     =     this.state.regions.findIndex(el     =>     el.iso     ===
feature.properties["iso3166-2"]);
    if (regionIndex != -1) {
      colorOpac = this.state.regions[regionIndex].solarPotentialPercentage;
    }

    return {
      color: "#4A901A",
      opacity: 0,
      fillOpacity: colorOpac
    };
  }



  onEachFeatureSolar(feature, layer) {

    let colorOpac = 0;
    let     regionIndex     =     this.state.regions.findIndex(el     =>     el.iso     ===
feature.properties["iso3166-2"]);
    if (regionIndex != -1) {
      colorOpac = this.state.regions[regionIndex].solarPotentialPercentage;
    }


    //var map = this.state.mapElemnt.leafletElement;

        layer.setStyle({color: "#4A901A",
        opacity: 0,
        fillOpacity: colorOpac})
  }


  onEachFeatureTech(feature, layer) {

    let colorOpac = 0;
    let     regionIndex     =     this.state.regions.findIndex(el     =>     el.iso     ===
feature.properties["iso3166-2"]);
    if (regionIndex != -1) {
      colorOpac = this.state.regions[regionIndex].techPotentialPercentage;
    }


    //var map = this.state.mapElemnt.leafletElement;

        layer.setStyle({color: "#4A901A",
        opacity: 0,
        fillOpacity: colorOpac})
```

```
  }

  onEachFeatureTermal(feature, layer) {

    let colorOpac = 0;
    let     regionIndex     =     this.state.regions.findIndex(el     =>     el.iso     ===
feature.properties["iso3166-2"]);
    if (regionIndex != -1) {
      colorOpac = this.state.regions[regionIndex].termalPotentialPercentage;
    }


    //var map = this.state.mapElemnt.leafletElement;

        layer.setStyle({color: "#4A901A",
        opacity: 0,
        fillOpacity: colorOpac})
  }

  onEachFeature(feature, layer) {
    var that = this;
    layer.on("click", function (e) {
      that.markerClick(e, feature);
    });
  }

  markerClick(e, feature) {
    if (
      e.target &&
      e.target.options &&
      e.target.options.position &&
      e.target.options.position.id
    ) {
      var marker = this.state.markers.find(
        element => element.id === e.target.options.position.id
      );
      var region = this.state.regions.find(
        element => element.id === marker.regionId
      );
      this.setState({
        userMarker: {
          id: marker.id,
          systemStationId: marker.systemStationId,
          area: marker.area,
          region: region,
          regionId: marker.regionId,
          coordinates: e.latlng,
          isNew: false,
          angles: this.state.angles,
          solarStations: this.state.solarStations,
          helioStations: this.state.helioStations
        }
      });
    } else {
      region = this.state.regions.find(
        element => element.iso === feature.properties["iso3166-2"]
      );
      this.setState({
        userMarker: {
          id: null,
          systemStationId: this.state.solarStations[0].id,
```

```
          area: 0,
          region: region,
          regionId: region.id,
          coordinates: e.latlng,
          isNew: true,
          angles: this.state.angles,
          solarStations: this.state.solarStations,
          helioStations: this.state.helioStations
        }
      });
      fetch('http://localhost:17725/marks/addMark', {
        method: 'POST',
        headers: {
          'Accept': 'application/json',
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          Lat: e.latlng.lat,
          Lng: e.latlng.lng,
          Text: 'Created by user',
          RegionId: region.id,
          Area: 0,
        })
      })
        .then(() => {
          this.componentDidMount();
        });
    }
    this.props.locationCallBack(this.state.userMarker);

    //ReactDOM.unmountComponentAtNode(document.getElementById("results"));
  }

  render() {
    if (!this.state.components) {
      return null;
    }
    const { ReactLeafletSearch } = this.state.components;
    const yellowIcon = L.icon({
      iconUrl:
        "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-
yellow.png",
      shadowUrl:
        "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    const blueIcon = L.icon({
      iconUrl:
        "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-
blue.png",
      shadowUrl:
        "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });
```

```
    const greenIcon = L.icon({
      iconUrl:
        "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-
green.png",
      shadowUrl:
        "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
      iconSize: [25, 41],
      iconAnchor: [12, 41],
      popupAnchor: [1, -34],
      shadowSize: [41, 41]
    });

    var corner1 = L.latLng(55, 13);
    var corner2 = L.latLng(40, 40);
    var bounds = L.latLngBounds(corner1, corner2);

    let newMark = null;
    if (
      this.state.userMarker.coordinates.lat &&
      this.state.userMarker.coordinates.lng
    ) {
      newMark = (
        <Marker
          position={[
            this.state.userMarker.coordinates.lat,
            this.state.userMarker.coordinates.lng
          ]}
          icon={yellowIcon}
        />
      );
    }

    let solarPotentialArray = [];
    this.state.regions.forEach(element => {
      solarPotentialArray.push(element.solarPotential * 11.63 * 1.32);
    });

    let maxSolarPotential = Math.max.apply(Math, solarPotentialArray);
    let minSolarPotential = Math.min.apply(Math, solarPotentialArray);

    var grades = [roundTwo(minSolarPotential),
    roundTwo(solarPotentialArray[Math.round(3 / 10 * solarPotentialArray.length)]),
    roundTwo(solarPotentialArray[Math.round(6 / 10 * solarPotentialArray.length)]),
    roundTwo(solarPotentialArray[Math.round(9 / 10 * solarPotentialArray.length)]),
    roundTwo(maxSolarPotential)
    ];

    var legendColorStep = 1.0 / (grades.length + 1);

    return (
      <React.Fragment>
        {this.state.displayPreloader ? (
          <div class="preloader-wr">
            <div class="lds-ring">
              <div />
              <div />
              <div />
              <div />
            </div>
          </div>
```

```
) : null}
<Map
  ref={(ref) => { this.state.mapElemnt = ref; }}
  className="simpleMap"
  center={[this.state.lat, this.state.lng]}
  zoom={this.state.zoom}
  minZoom={this.state.zoom}
  maxBounds={bounds}
  maxBoundsViscosity="0"
>
  <ReactLeafletSearch
    position="topleft"
    zoom={9}
    showMarker={false}
    showPopup={true}
    openSearchOnLoad={false}
    closeResultsOnClick={false}
    customProvider={false}
    provider="OpenStreetMap"
    providerOptions={{ region: "ua" }}
  />
  <TileLayer url="http://{s}.tile.osm.org/{z}/{x}/{y}.png" />
  <GeoJSON
    data={ukraineGeo}
    style={this.setUkraineStyle}
    onEachFeature={this.onEachFeature}
  />
  {/* <GeoJSON
    data={otherCountriesGeo}
    style={{
      fillColor: "grey",
      weight: 1,
      opacity: 0,
      color: "white",
      dashArray: "3",
      fillOpacity: 0.3
    }}
  /> */}
  {this.state.markers.map((position, idx) => (
    <Marker
      key={"marker " + idx}
      position={position}
      icon={
        this.state.solarStations.find(
          x => x.id === position.systemStationId
        )
          ? blueIcon
          : greenIcon
      }
      onClick={this.markerClick.bind(this)}
    >
      <Popup className="request-popup">
        <p>{position.text}</p>
        <img width='200px' src={'' + position.imageBase64} />
      </Popup>
    </Marker>
  ))}

  {newMark}

  <Control position="bottomright" className="legend">
```

```
            <div class="legend-header">
              <span>Зменшення викидів CO2, т × 10^5</span>
            </div>
            {grades.map((item, i) => {
              return (
                <div>
                  <i
                    style={{
                      background: "#4A901A",
                      opacity: legendColorStep * (i + 1)
                    }}
                  />
                  <span class="legend-text">
                    {item + (grades[i + 1] ? " - " + grades[i + 1] : "+")}
                  </span>
                  <br />
                </div>
              );
            })}
          </Control>



          <Control position="topright" className="legend">
            <div class="legend-header">

              <button      class="btn      btn-primary      w-100      p-3"      type="button"
onClick={this.setColorSolar}>
                SOLAR
              </button>
<p></p>
              <button      class="btn      btn-primary      w-100      p-3"      type="button"
onClick={this.setColorTech}>
                TECH
              </button>
<p></p>
              <button      class="btn      btn-primary      w-100      p-3"      type="button"
onClick={this.setColorTermal}>
                TERMAL
              </button>

            </div>
          </Control>
        </Map>
      </React.Fragment>
    );
  }
}

function roundTwo(num) {
  return Math.round(num * 100) / 100;
}
```

ДОДАТОК В

Інтерактивна веб-карта для представлення зменшення викидів вуглекислого газу при освоєнні потенціалу сонячної енергії по областям України (розробка веб-інтерфейсу користувача).

Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ62205_20Б-3

Аркушів

Київ 2019

АНОТАЦІЯ

У цьому розділі описано частину, необхідну для роботи з веб-картою, яка є базовою одиницею програмного продукту, та подано поєднання всіх інших компонентів для обчислення викидів вуглекислого газу в Україні на одній веб-сторінці. Вказано принцип роботи з API, який служить для отримання необхідних даних з сервера та бази даних. Модуль написаний мовою програмування C #, використовуючи EntityFramework.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглянуто програмний модуль системи який відповідає за роботу веб-сторінкою з кодом UKR.NTUU "KPI" _TEF_APEPS_TI51172_19B 12-1, що міститься у файлі solarMap.jsx. Модуль створено за допомогою бібліотек React і Leaflet. Модуль призначений для управління системою, яка контролює данні, що надходять з API. Користувач має можливість вибрати наявні записи або зробити новий розрахунок, використовуючи власні дані.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Мета модуля для роботи з веб-карткою - відображення інформації з сервера та бази даних та безпосередньо прийняття вхідних даних та обмін інформацією з клієнтським інтерфейсом. Використання такого шаблону дозволяє створити програмне забезпечення, де інтерфейс та логіка модуля для бази даних та сервера є незалежними компонентами, що дозволяє використовувати його для зменшення навантаження на клієнтську частину системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Моніторинг змін інформації підсистем є основним завданням модуля. Коли система запускається, модуль повертає всю інформацію, що міститься в базі даних, для відображення даних в інтерфейсі користувача. Модуль також обробляє інформацію, дозволяючи вводити необхідні дані та виконувати розрахунки та відображати готові результати.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблений у середовищі розробки коду Microsoft Visual Studio Code, яка забезпечує набір функцій обслуговування та графічний діалог з користувачем. Для реалізації клієнтської частини були використані бібліотеки React та Leaflet. За допомогою елементів React клієнтська частина може підключитися до сервера за допомогою API.

Постачальником найбільш інтерактивної карти було обрано сервіс OpenStreetMap.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль реалізований як окремий клас, він дозволяє клієнтській частині та модулю бізнес-логіки існувати окремо один від одного, але запуск обох компонентів відбувається одночасно.

Щоб використовувати цей модуль, вам необхідно завантажити веб-сторінку, яка зробить необхідні запити та відобразить результат.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є інформація, яку користувач вводить в додатку.

Вихідними даними програмного модуля є результати обчислення.