Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ РОЗРАХУНКУ ТА ЗМЕНШЕННЯ ВИКИДУ CO2 В АТМОСФЕРУ

DEVELOPMENT OF A WEB SERVICE FOR CALCULATING AND REDUCING CO2 EMISSIONS IN THE ATMOSPHERE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗз-41
Кухарчук Давід Сергійович
_____
(підпис)

Керівник:
асистент
Потейчук Михайло Іванович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М. Лінчук
«2» 02 202 4 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кухарчуку Давіду Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка вебсервісу для розрахунку та зменшення викиду CO2 в атмосферу»
Керівник роботи: Потейчук М.І.
затверджені наказом закладу вищої освіти від «*30» грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи:

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):

Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Потейчук М. І.*		
Специфікація вимог до розробленої системи	*Потейчук М. І.*		
Розробка об'єкта проектування	*Потейчук М. І.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	9,1%		
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*до 16.02.2024 р.*	вик.
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*до 01.03.2024 р.*	вик.
3	*Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання*	*до 15.03.2024 р.*	вик.
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*до 29.03.2024 р.*	вик.
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*до 26.04.2024 р.*	вик.
6	*Тестування та налагодження об'єкта проектування*	*до 03.05.2024 р.*	вик.
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*до 5.06.2024 р.*	вик.

Здобувач вищої освіти _____ Бухарцун Д.С.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Потейчук М.І.
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: Кухарчук Давід Сергійович

Тема: Розробка вебсервісу для розрахунку та зменшення викиду CO2 в атмосферу.

Коротка характеристика кваліфікаційної роботи: Ця робота присвячена розробці інноваційного вебсервісу для автоматизованого розрахунку та зменшення викидів CO2 в атмосферу. Основним завданням було створення інструменту, який не тільки надає точні розрахунки викидів CO2, але й пропонує ефективні стратегії для їх зменшення. Для реалізації цього проєкту використовувались сучасні технології веброзробки, такі як Node.js з фреймворком Express для бекенду, React.js для фронтенду та AWS Lambda для серверних функцій.

Самостійні розробки і пропозиції автора: У роботі були розроблені та впроваджені інноваційні підходи до створення вебсервісу для розрахунку та зменшення викидів CO2. Особлива увага була приділена вибору технологій і структурних рішень, які забезпечують високу продуктивність і надійність системи. Впроваджено інтеграцію з AWS Lambda, що дозволяє забезпечити безперервне масштабування та ефективну обробку даних у реальному часі. Це рішення значно зменшує витрати на інфраструктуру та підвищує доступність сервісу.

Практичне значення роботи: Запропоновані методи і підходи були успішно застосовані для розробки та впровадження вебсервісу, який розраховує та допомагає зменшити викиди CO2 в атмосферу. Результати роботи мають практичне значення, підтверджене їх використанням у реальних умовах, зокрема, для моніторингу та оптимізації викидів у різних секторах. Вебсервіс забезпечує інтуїтивно зрозумілий інтерфейс для користувачів, високий рівень точності розрахунків і пропонує ефективні стратегії для зниження викидів, що робить його цінним інструментом для екологічного моніторингу та управління.

Недоліки: відсутні.

Загальний висновок: Робота виконана на високому науковому рівні, має логічну та послідовну структуру, відповідає встановленим вимогам, може бути представлена до захисту на державній екзаменаційній комісії та заслуговує високої позитивної оцінки.

Рецензент: Здолбіцька Н.В., доцент кафедри КН
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«07» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Здобувач вищої освіти *Кухарчук Давід Сергійович*
група *ІПЗс-41*

Тема кваліфікаційної роботи: Розробка вебсервісу для розрахунку та зменшення викиду CO2 в атмосферу.

Керівник *Ліщина Наталія Миколаївна*

Актуальність теми: сьогодні, в умовах глобальних кліматичних змін, питання викидів CO2 стає все більш критичним. Розробка ефективних інструментів для моніторингу та зменшення викидів CO2 є ключовою для забезпечення екологічної стійкості та збереження нашої планети. Вебсервіс, який дозволяє розраховувати викиди CO2 та пропонувати стратегії для їх зменшення, є вкрай необхідним у контексті сучасних екологічних викликів.

Об'єкт дослідження – Процес розробки вебсервісу для розрахунку та зменшення викидів CO2 в атмосферу.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи: розроблений вебсервіс є потужною платформою для моніторингу викидів CO2, що дозволяє користувачам вводити дані про різні види палива і автоматично розраховувати викиди. Система інтегрує сучасні технології, такі як Node.js для бекенду, React.js для фронтенду, і AWS Lambda для серверних функцій, що забезпечує високу продуктивність, масштабованість та надійність.

Зауваження та недоліки: істотних недоліків не виявлено.

Загальний висновок робота виконана на високому рівні та заслуговує оцінки «відмінно» за умови успішного захисту.

Керівник _____ (Потапчук М. Ч.)
(підпис) (прізвище, ім'я, по батькові)

«7» 06 2024 р.

АНОТАЦІЯ

Кухарчук Д. С. Розробка вебсервісу для розрахунку та зменшення викиду CO_2 в атмосферу. Рукопис.

Кваліфікаційна робота бакалавра. Луцький національний технічний університет, 2024.

У першому розділі роботи проаналізовано основні принципи розрахунку викидів CO2, їх структуру та вплив на навколишнє середовище. У другому розділі досліджено методологію розробки вебсервісу з використанням фреймворку NodeJS Express з використання AWS Lambda, EC2, S3, що забезпечує гнучкість та масштабованість системи. У практичній частині роботи детально описаний повний процес розробки та тестування вебсервісу на реальних даних. В роботі проаналізовано економічну сторону проекту: розраховано витрати на розробку програмного забезпечення, експлуатаційні витрати, ціну споживання проектного рішення та визначено показники економічної ефективності, які свідчать про вигідність запровадження цього вебсервісу для широкого кола користувачів.

Ключові слова: розробка вебсервісів, розрахунок викидів CO_2, зменшення викидів, екологічні технології, NodeJS, AWS, React, обробка даних, аналіз вуглецевого сліду.

ABSTRACT

Kukharchuk David. Development of a web service for calculating and reducing CO2 emissions into the atmosphere. Bachelor qualification work. Lutsk National Technical University, 2024. Manuscript.

Climate change and the increase in CO_2 concentration in the atmosphere have become one of the most pressing issues of the modern world. The first chapter of the thesis analyzes the basic principles of CO_2 emission calculations, their structure, and their impact on the environment. The second chapter explores the methodology of developing the web service using the NodeJS Express framework with AWS Lambda, EC2, S3 integration, providing flexibility and scalability to the system. The practical part of the thesis describes in detail the complete process of developing and testing the web service using real data. The economic aspect of the project is analyzed: the costs of software development, operational expenses, the price of using the project solution, and indicators of economic efficiency are calculated, indicating the profitability of implementing this web service for a wide range of users.

Keywords: web service development, CO_2 emissions calculation, emissions reduction, environmental technologies, NodeJS, AWS, React, data processing, carbon footprint analysis.

ЗМІСТ

ВСТУП

Сьогодення ставить перед собою нагальне завдання зменшення викидів CO2, що є однією з ключових причин зміни клімату. Штучні нейронні мережі виявляються невід'ємною частиною цього процесу, допомагаючи вирішувати проблеми в різних сферах, зокрема в енергетиці, транспорті, технологічному виробництві та інших.

У даній роботі ми зосереджуємо увагу на розробці вебсервісу, спрямованого на розрахунок та зменшення викидів CO2 в атмосферу. Метою є створення інструменту, який допоможе розрахувати викиди та запропонувати ефективні стратегії для їх зменшення.

Метою дослідження є розробка вебсервісу, який дозволяє точно розраховувати викиди CO2 та надавати рекомендації для їх зменшення, використовуючи сучасні технології обробки даних та аналітичні методи.

Об'єктом дослідження є процеси викидів CO2 в різних секторах економіки, таких як промисловість, транспорт та енергетика, а також засоби їх моніторингу та управління.

Предметом дослідження є методи та інструменти для розрахунку та зменшення викидів CO2, зокрема розробка вебсервісу, який автоматизує ці процеси та дозволяє користувачам ефективно управляти викидами.

Для досягнення поставленої мети були поставлені наступні завдання:

– аналіз методів розрахунку викидів CO2;

– проектування архітектури вебсервісу з використанням сучасних технологій програмування та баз даних;

– розробка вебсервісу для розрахунку викидів CO2;

– реалізація функціональності для введення даних про споживання ресурсів, розрахунку викидів CO2 та генерування звітів.

– проведення тестування та аналізу ефективності вебсервісу.

У рамках роботи будуть розглянуті різні аспекти розробки вебсервісу, включаючи:

– аналіз методів розрахунку викидів CO2. Огляд різних методів та підходів до вимірювання та оцінки викидів CO2, враховуючи їхні впливи на навколишнє середовище та клімат. Враховуючи різні джерела викидів, такі як промисловість, транспорт та сільське господарство, ми будемо досліджувати оптимальні підходи до їх зменшення та контролю;

– розробка програмного забезпечення. Проектування та реалізація вебсервісу з використанням сучасних технологій програмування та баз даних для забезпечення ефективного функціонування та збереження даних. Розроблений вебсервіс буде надійним та зручним у використанні, що дозволить користувачам ефективно керувати викидами CO2;

– оцінка ефективності стратегій зменшення викидів CO2. Проведення аналізу та оцінка різних стратегій та заходів для зменшення викидів CO2, включаючи їхні витрати та вплив на довкілля. Ми будемо досліджувати ефективність різних заходів, таких як використання відновлюваних джерел енергії, енергоефективність та карбонізація промисловості, для зменшення загальних викидів CO2.

Наукова новизна цього дослідження полягає у використанні сучасних технологій, включаючи штучний інтелект та аналітичні методи, для розробки інструменту, який може значно покращити управління викидами CO2 та сприяти збереженню навколишнього середовища.

Далі будемо розглядати докладніше основні аспекти розробки вебсервісу та методів зменшення викидів CO2.

РОЗДІЛ 1
ОГЛЯД ПРОБЛЕМИ З ВИКИДОМ CO2 ТА МЕТОДІВ РОЗРАХУНКУ

1.1 Поняття викидів CO2 та їх вплив на навколишнє середовище

Викиди CO2 (вуглекислого газу) є серйозною проблемою, яка загрожує екологічній стійкості нашої планети. Вуглекислий газ утворюється під час згоряння вуглеводнів, таких як вугілля, нафта та природний газ, що використовуються у великій кількості в промисловості, енергетиці, транспорті та інших галузях [1]. Основні джерела викидів CO2 включають автомобільний транспорт, електростанції, заводи та інші джерела забруднення [2]. Надмірні викиди CO2 призводять до зміни клімату, що має різноманітні негативні наслідки для навколишнього середовища та людського здоров'я. Один із основних ефектів – глобальне потепління, що призводить до танення льодовиків, підвищення рівня морів, зміни кліматичних зон та інших змін у природних процесах. Крім того, викиди CO2 також сприяють забрудненню повітря, що може призводити до захворювань дихальних шляхів та інших проблем зі здоров'ям.

Опишемо основні джерела викидів CO2. Автомобільний транспорт. Транспортні засоби, що використовують бензин та дизельне паливо, є одними з найбільших джерел викидів CO2. За даними Агентства з охорони навколишнього середовища (EPA), транспортний сектор у США відповідає за приблизно 29 % загальних викидів парникових газів, з яких більша частина припадає на легкові автомобілі та вантажівки [3].

Електростанції. Вугільні електростанції є ще одним значним джерелом викидів CO2. Згідно з Міжнародним енергетичним агентством (IEA), виробництво електроенергії генерує близько 42 % глобальних викидів CO2 [4]. Це пов'язано з тим, що вугілля має високу вуглецеву інтенсивність, що означає, що його згоряння виділяє велику кількість CO2.

Промисловість. Промислові процеси, такі як виробництво цементу, сталі та хімічних продуктів, також є значними джерелами викидів CO2. Наприклад,

виробництво цементу генерує приблизно 8 % глобальних викидів CO2 через хімічні процеси та споживання енергії [5].

Вплив викидів CO2 на навколишнє середовище. Надмірні викиди CO2 призводять до зміни клімату, що має різноманітні негативні наслідки для навколишнього середовища та людського здоров'я. Один із основних ефектів - глобальне потепління, що призводить до танення льодовиків, підвищення рівня морів, зміни кліматичних зон та інших змін у природних процесах [6]. Крім того, викиди CO2 також сприяють забрудненню повітря, що може призводити до захворювань дихальних шляхів та інших проблем зі здоров'ям.

Глобальне потепління. Одним із головних наслідків підвищення концентрації CO2 в атмосфері є підвищення глобальної температури. За даними Міжурядової групи експертів зі зміни клімату (IPCC), середня глобальна температура піднялася приблизно на 1°C порівняно з доіндустріальним рівнем, і це підвищення головним чином спричинене викидами парникових газів [7]. Це призводить до танення льодовиків та полярних шапок, підвищення рівня моря та екстремальних погодних умов.

Зміна кліматичних зон. Підвищення температури змінює кліматичні зони, що впливає на біорізноманіття та сільське господарство. Наприклад, багато видів рослин і тварин не можуть швидко адаптуватися до нових умов, що призводить до втрати біорізноманіття. Крім того, зміна клімату впливає на сільське господарство, зменшуючи врожайність та збільшуючи ризики для продовольчої безпеки [8].

Забруднення повітря. Викиди CO2 часто супроводжуються викидами інших забруднювачів, таких як оксиди азоту (NOx) та діоксиди сірки (SO2), які сприяють утворенню смогу та кислотних дощів. Це негативно впливає на якість повітря і може призводити до респіраторних захворювань, серцево-судинних проблем та інших проблем зі здоров'ям [9].

Заходи щодо зменшення викидів CO2. Для зменшення негативного впливу викидів CO2 необхідно вжити ефективних заходів з обмеження та зменшення викидів, а також розвинути нові методи та технології, які дозволять замінити

вуглеводні палива більш екологічно чистими джерелами енергії [10]. Доцільно вивчати та застосовувати різноманітні методи розрахунку викидів CO2 для ефективного контролю та моніторингу рівня забруднення довкілля [11].

Поновлювані джерела енергії. Перехід на поновлювані джерела енергії, такі як сонячна, вітрова та гідроенергія, може значно зменшити викиди CO2. Ці джерела не тільки знижують викиди парникових газів, але й зменшують залежність від викопного палива [12].

Енергоефективність. Поліпшення енергоефективності в промисловості, транспорті та побуті може суттєво зменшити викиди CO2. Це включає впровадження енергоефективних технологій, модернізацію інфраструктури та зміну поведінки споживачів [13].

Захоплення та зберігання вуглецю (CCS). Технології захоплення та зберігання вуглецю дозволяють уловлювати CO2, що викидається в атмосферу, та зберігати його у підземних резервуарах. Це може допомогти зменшити викиди з великих промислових об’єктів [14].

Зелені насадження. Збільшення кількості зелених насаджень та лісів може допомогти поглинати CO2 з атмосфери. Ліси є природними «косовими вуглецю», і їх збереження та відновлення мають вирішальне значення для пом’якшення зміни клімату [15].

На рисунку 1.1 зображений завод із димовими трубами, що викидають CO2 в атмосферу [16].

Рисунок 1.1 – Завод із димовими трубами, що викидають CO2 в атмосферу [16]

На рисунку 1.2 зображений автомобільний затор як джерело викидів CO2 [17].

Рисунок 1.2 – Автомобільний затор як джерело викидів CO2 [17]

Одним із найбільш відомих індикаторів зростання рівня CO2 в атмосфері є Графік Кілінга, названий на честь вченого Чарльза Девіда Кілінга, який у 1958 році розпочав систематичне вимірювання концентрації CO2 на станції Маунт Лоа у Гаваях. Ці вимірювання показали безперервне зростання концентрації CO2 в атмосфері, що стало однією з основних доказів антропогенного впливу на зміну клімату.

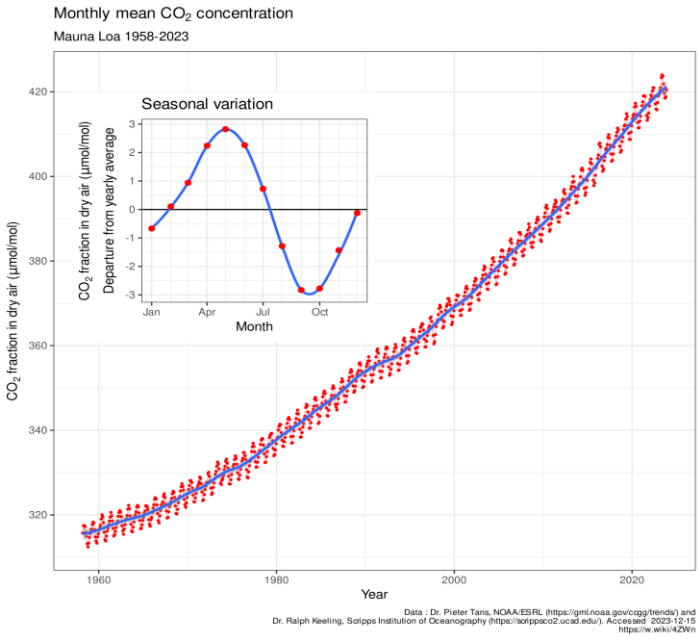

Рисунок 1.3 – Концентрації вуглекислого газу в атмосфері (CO2) з 1958 по 2023 рік [18]

Графік Кілінга чітко демонструє сезонні коливання рівня CO2, пов’язані з природними процесами поглинання та викиду вуглекислого газу, а також загальне довгострокове зростання, що є результатом людської діяльності. Цей графік є ключовим доказом для розуміння того, як діяльність людини впливає на атмосферу та клімат Землі [19].

1.2 Аналіз існуючих методик розрахунку викидів CO2

При розробці вебсервісу для розрахунку та зменшення викидів CO2 важливо спочатку розглянути та оцінити різні методики розрахунку, щоб вибрати найбільш відповідну для впровадження. Кожен підхід має свої особливості, переваги та обмеження, які слід врахувати при виборі оптимального рішення:

Метод масових балансів. Метод базується на принципі збереження маси та використовується для розрахунку викидів CO2 на основі даних про споживання палива та його хімічний склад:

– принцип дії. Розрахунок кількості CO2, що викидається під час згоряння певної кількості палива;

– формула. Викиди CO2 = Маса палива × Коефіцієнт викидів, де коефіцієнт викидів залежить від типу палива та може бути отриманий з відповідних довідників.

Переваги:

– висока точність для розрахунків у промислових умовах;

– придатність для аналізу викидів на великих об’єктах або в системах з відомими характеристиками споживаного палива.

Недоліки:

– вимагає точних даних про кількість і хімічний склад спожитого палива;

– складність у застосуванні для джерел викидів, де складно отримати точні дані про споживання палива.

Емпіричні методи. Ці методи використовують статистичні дані та кореляції для оцінки викидів CO2 на основі різних факторів, таких як типи палива, ефективність процесів та інші параметри:

– принцип дії. Використання середніх значень та кореляцій для швидкого оцінювання викидів;

– приклад. Методики, запропоновані IPCC, надають стандартні коефіцієнти викидів для різних джерел та типів палива.

Переваги:

– простота у використанні;

– зручність для широких оцінок або політичного аналізу на регіональному рівні.

Недоліки:

– менша точність порівняно з методами масових балансів;

– обмежена можливість застосування для специфічних процесів або умов.

Методи моделювання та симуляції. Ці методи використовують математичні моделі для прогнозування викидів CO2 у різних умовах, враховуючи складні взаємозв’язки між параметрами:

– принцип дії. Використання математичних моделей для імітації процесів з метою прогнозування викидів.

– приклад. Програмне забезпечення Aspen Plus або GHG Protocol дозволяє моделювати промислові процеси та оцінювати викиди.

Переваги:

– висока точність;

– можливість аналізу складних сценаріїв та взаємозв’язків.

Недоліки:

– потреба в детальних даних і технічних знаннях;

– високі вимоги до обчислювальних ресурсів для складних моделей.

Використання міжнародних стандартів. Дотримання міжнародних стандартів, таких як GHG Protocol або ISO 14064, забезпечує точні інструменти

та керівництва для розрахунку викидів CO2:

– принцип дії. Використання стандартизованих методик для вимірювання та управління викидами;

– приклад. GHG Protocol надає набори інструментів для оцінки викидів від різних джерел та видів діяльності.

Переваги:

– забезпечує відповідність нормативним вимогам;

– легкість у порівнянні даних між різними організаціями.

Недоліки:

– вимагає детального збору та документування даних;

– може бути складним для впровадження у малих організаціях без належної підтримки.

У висновку, можемо сформулювати, що кожен метод має свої переваги та недоліки, що робить його більш або менш придатним для різних умов. Метод масових балансів є точним для великих промислових об’єктів, емпіричні методи підходять для швидких оцінок, а методи моделювання дозволяють глибоко аналізувати складні процеси. Міжнародні стандарти забезпечують відповідність і точність, що є ключовими для ефективного моніторингу та управління викидами.

1.3 Вибір оптимального підходу для реалізації вебсервісу

Вибір оптимального підходу для реалізації вебсервісу, що розраховує та зменшує викиди CO2 в атмосферу, є критично важливим завданням, яке потребує ретельного аналізу та оцінки різних технологічних рішень. Для цього необхідно врахувати кілька ключових факторів, таких як продуктивність, масштабованість, безпека, зручність використання та відповідність сучасним технологічним стандартам.

Основними компонентами вебсервісу є фронтенд (інтерфейс користувача), бекенд (серверна частина) та база даних. Вибір технологій для кожного з цих компонентів має велике значення для загальної ефективності системи. Для даного проекту обрано такі технології: Node.js з фреймворком Express для бекенду, React.js для фронтенду та AWS Lambda для серверних функцій.

Фронтенд вебсервісу відповідає за взаємодію з користувачем і має бути інтуїтивно зрозумілим, швидким і чутливим до дій користувача. Для цього використовується фреймворк React.js. React.js є одним з найпопулярніших виборів завдяки своїй компонентній архітектурі та великій спільноті розробників, що забезпечує доступ до багатьох бібліотек та інструментів. React.js дозволяє створювати швидкі та інтерактивні користувацькі інтерфейси, а також забезпечує можливість легкого управління станом додатка через бібліотеку Redux або Context API.

Бекенд забезпечує логіку додатка, обробку запитів, управління даними та зв’язок з базою даних. Вибір технологій для бекенду залежить від вимог до продуктивності, безпеки та масштабованості. В даному проекті використовується Node.js з фреймворком Express та AWS Lambda.

Node.js є платформою на основі JavaScript, яка дозволяє виконувати серверний код асинхронно, що підвищує продуктивність і масштабованість додатка. Express.js є мінімалістичним фреймворком для Node.js, який забезпечує швидку та ефективну розробку веб-додатків та API. Використання Node.js та Express.js дозволяє використовувати один і той самий мовний стек як на фронтенді, так і на бекенді, що спрощує розробку та підтримку додатка.

AWS Lambda забезпечує виконання серверного коду без необхідності керувати серверами. Цей сервіс дозволяє запускати функції у відповідь на події та автоматично масштабується в залежності від навантаження. Використання AWS Lambda дозволяє знизити витрати на інфраструктуру та забезпечити високу доступність та надійність.

AWS Lambda може використовуватися для автоматизації обробки даних та виконання розрахунків. Наприклад, при додаванні нових даних про викиди

можна автоматично запускати функції Lambda для виконання відповідних розрахунків.

В лістингу 1.1 наведено приклад інтеграції AWS Lambda, використовуючи JavaScript на середовищі NodeJS.

Лістинг 1.1 – Демонстрація інтеграції AWS Lambda в JavaScript на середовищі NodeJS

```
const AWS = require('aws-sdk');
const lambda = new AWS.Lambda();
const invokeLambdaFunction = async (emissionData) => {
    const params = {
        FunctionName: 'YourLambdaFunctionName',
        Payload: JSON.stringify(emissionData),
    };
    try {
        await lambda.invoke(params).promise();
    } catch (err) {
        console.error('Error invoking Lambda function: ', err);
    }
};
```

Кінець лістингу 1.1

База даних є важливою частиною вебсервісу, оскільки вона зберігає всі дані про викиди CO2 та результати розрахунків. Вибір бази даних залежить від обсягу даних, вимог до швидкості доступу та обробки запитів. У цьому розділі ми розглянемо використання реляційної бази даних PostgreSQL для зберігання та управління даними про викиди CO2.

PostgreSQL є потужною, відкритою реляційною системою управління базами даних (СУБД). Вона відома своєю стабільністю, надійністю та багатофункціональністю, що робить її ідеальним вибором для багатьох видів додатків, включаючи ті, що вимагають складної обробки даних та високої масштабованості.

Переваги PostgreSQL:

– стабільність і надійність. PostgreSQL має довгу історію розвитку і використання в продуктивних системах, що забезпечує високий рівень надійності та стабільності;

– розширюваність. PostgreSQL дозволяє додавати нові типи даних, оператори та функції, що робить її надзвичайно гнучкою;

– підтримка ACID. PostgreSQL підтримує транзакції ACID (атомарність, консистентність, ізольованість, довговічність), що гарантує цілісність даних навіть у випадку збоїв;

– масштабованість. PostgreSQL здатна ефективно обробляти великі обсяги даних та високу кількість одночасних запитів, що робить її придатною для масштабованих додатків;

– безпека. PostgreSQL забезпечує широкий спектр засобів безпеки, включаючи шифрування даних, аутентифікацію та контроль доступу.

У контексті вебсервісу, побудованого на Node.js з використанням Express.js та AWS Lambda, PostgreSQL може бути інтегрована як основне сховище даних. Ось як це може виглядати:

– Node.js та Express.js. Ці технології використовуються для розробки серверної частини додатку. Express.js забезпечує маршрутизацію запитів, а Node.js відповідає за виконання серверних сценаріїв та обробку запитів;

– PostgreSQL. Використовується для зберігання даних про викиди CO2, включаючи дані з різних джерел, результати розрахунків та інші пов’язані дані;

– AWS Lambda. Може використовуватися для виконання функцій на вимогу, таких як обробка даних та виконання розрахунків. Це забезпечує масштабованість та ефективне використання ресурсів.

Використання PostgreSQL для зберігання даних про викиди CO2.

Структурування даних. Для ефективного зберігання та обробки даних про викиди CO2 важливо правильно структурувати базу даних. Основні таблиці можуть включати:

– Sources – зберігає інформацію про джерела викидів (наприклад, транспорт, електростанції, промислові об’єкти);

– Emissions – містить дані про викиди, такі як кількість CO2, дата та час викиду, джерело тощо;

– Calculations – зберігає результати розрахунків викидів для різних сценаріїв та методик.

В лістингу 1.2 наведено приклад створення таблиць бази даних, використовуючи SQL.

Лістинг 1.2 – Демонстрація створення таблиць бази даних, використовуючи SQL в JavaScript на середовищі NodeJS.

```
CREATE TABLE sources
(
   id          SERIAL PRIMARY KEY,
   NAME        VARCHAR(255) NOT NULL,
   description TEXT
);
CREATE TABLE emissions
(
   id            SERIAL PRIMARY KEY,
   source_id     INT REFERENCES sources(id),
   co2_amount    DECIMAL(10, 2) NOT NULL,
   emission_date TIMESTAMP NOT NULL
);
CREATE TABLE calculations
(
   id               SERIAL PRIMARY KEY,
   emission_id      INT REFERENCES emissions(id),
   method           VARCHAR(255),
   result           DECIMAL(10, 2),
   calculation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Кінець лістингу 1.2

Інтеграція з Node.js та Express.js. Для взаємодії з PostgreSQL з Node.js можна використовувати популярні бібліотеки, такі як `pg` (node-postgres). Це дозволяє легко виконувати SQL-запити та отримувати дані з бази.

Переваги використання PostgreSQL у вебсервісі:

– складні запити та аналітика. PostgreSQL підтримує складні запити та аналітичні функції, що дозволяє ефективно обробляти та аналізувати великі обсяги даних про викиди CO2;

– розширені можливості індексування. PostgreSQL забезпечує різні типи індексів, що допомагає оптимізувати запити та покращити продуктивність;

– розширена підтримка JSON. PostgreSQL підтримує зберігання та обробку JSON-даних, що дозволяє легко працювати з напівструктурованими даними та інтегрувати їх з іншими системами;

– підтримка просторових даних. За допомогою розширення PostGIS PostgreSQL може працювати з географічними даними, що корисно для аналізу викидів на основі розташування.

Для розрахунку викидів CO2 існує кілька методів, які можуть бути реалізовані у вебсервісі.

Метод масових балансів базується на принципі збереження маси та використовується для розрахунку викидів на основі даних про споживання палива та продукції. Цей метод дозволяє точно оцінити кількість CO2, що викидається під час згоряння палива.

Викиди CO2 = Маса палива × Коефіцієнт викиді, де Коефіцієнт викидів залежить від типу палива і може бути отриманий з відповідних довідників.

Емпіричні методи використовують статистичні дані та кореляції для прогнозування викидів на основі різних факторів, таких як типи палива, ефективність процесів та інші параметри. Ці методи можуть бути корисними для швидкої оцінки викидів у великих системах.

Моделювання та симуляція використовує математичні моделі для прогнозування викидів CO2 у різних умовах. Цей підхід дозволяє враховувати

складні взаємозв’язки між різними параметрами та забезпечує високу точність розрахунків.

Важливою складовою розрахунку викидів CO2 є використання надійних інструментів та протоколів. Один з таких інструментів – Greenhouse Gas Protocol.

Greenhouse Gas Protocol (GHG Protocol) є найпоширенішою у світі рамковою структурою для вимірювання та управління викидами парникових газів. Він був розроблений World Resources Institute (WRI) та World Business Council for Sustainable Development (WBCSD). Цей протокол забезпечує інструменти та керівництва для розрахунку викидів CO2 для різних секторів економіки. GHG Protocol надає набір інструментів, які можна використовувати для оцінки викидів CO2 на основі різних джерел і видів діяльності. Вони включають:

– Corporate Standard використовується для вимірювання та управління викидами на рівні організації;

– Project Protocol забезпечує методологію для оцінки викидів від окремих проектів;

– Product Standard дозволяє оцінювати викиди, пов’язані з життєвим циклом продуктів;

– Scope 3 Standard включає викиди, пов’язані з ланцюгами постачання.

GHG Protocol забезпечує деталізовані методики та приклади для розрахунку викидів, що робить його цінним ресурсом для організацій, які прагнуть точно вимірювати та зменшувати свій вуглецевий слід. Використання GHG Protocol у нашому вебсервісі забезпечить точність та надійність розрахунків, що є ключовим для ефективного моніторингу та зниження викидів CO2.

AWS Lambda дозволяє реалізувати серверну логіку без необхідності керувати інфраструктурою. Це включає запуск функцій на основі HTTP-запитів, обробку даних та зберігання результатів у базі даних. AWS Lambda автоматично масштабується в залежності від навантаження, що забезпечує високу доступність та надійність вебсервісу. Вибір оптимального підходу для реалізації

вебсервісу з розрахунку та зменшення викидів CO2 є комплексним завданням, що потребує врахування багатьох факторів. Важливо вибрати технології, які забезпечать високу продуктивність, масштабованість та безпеку додатка. Використання Node.js з фреймворком Express, React.js та AWS Lambda дозволяє створити ефективний та надійний вебсервіс, який відповідає сучасним технологічним стандартам.

1.4 Постановка завдання на кваліфікаційну роботу бакалавра

Для досягнення поставленої мети були поставлені наступні завдання:

– аналіз методів розрахунку викидів CO2;

– проектування архітектури вебсервісу з використанням сучасних технологій програмування та баз даних;

– розробка вебсервісу для розрахунку викидів CO2;

– реалізація функціональності для введення даних про споживання ресурсів, розрахунку викидів CO2 та генерування звітів;

– проведення тестування та аналізу ефективності вебсервісу.

РОЗДІЛ 2
МЕТОДОЛОГІЯ РОЗРОБКИ ВЕБСЕРВІСУ

2.1 Використання Node.js з фреймворком Express для розробки вебсервісу

Node.js та фреймворк Express стали популярними інструментами для розробки вебсервісів завдяки своїй продуктивності, масштабованості та зручності використання. Ці технології забезпечують швидку та ефективну розробку серверної частини веб-додатків, дозволяючи зосередитись на бізнес-логіці та функціональності додатку.

Node.js є середовищем виконання JavaScript, що працює на движку V8 від Google. Вона дозволяє виконувати JavaScript-код на сервері, що відкриває нові можливості для розробників, які можуть використовувати одну мову програмування як на клієнтській, так і на серверній стороні. Node.js є подієво-орієнтованою та асинхронною платформою, що робить її ідеальною для розробки масштабованих та швидкодіючих мережевих додатків.

Основні переваги використання Node.js:

– швидкість виконання. Завдяки використанню движка V8, Node.js забезпечує високу продуктивність та швидкість виконання коду;

– асинхронність. Node.js використовує неблокуючу модель вводу/виводу, що дозволяє обробляти велику кількість одночасних з’єднань без блокування процесів;

– єдина мова програмування. Можливість використовувати JavaScript як на сервері, так і на клієнті, спрощує розробку та підтримку коду;

– широкий вибір бібліотек. Node.js має багатий репозиторій модулів та бібліотек (npm), що значно прискорює розробку.

Express є мінімалістичним фреймворком для Node.js, який надає широкий набір інструментів та функцій для створення веб-додатків та API. Express дозволяє швидко створювати сервери та керувати маршрутизацією запитів,

обробкою даних, аутентифікацією користувачів та іншими важливими аспектами серверної частини веб-додатку.

Основні переваги використання Express:

– легкість у використанні. Express має простий та інтуїтивно зрозумілий синтаксис, що дозволяє швидко створювати та налаштовувати сервери;

– масштабованість. Завдяки модульній архітектурі, Express легко розширювати та налаштовувати під конкретні потреби проекту;

– потужна екосистема. Express інтегрується з багатьма популярними модулями та бібліотеками для Node.js, що спрощує розробку складних функціональних можливостей.

Для створення вебсервісу на основі Node.js та Express, необхідно виконати наступні кроки з використанням команд.

1) ініціалізація проекту: npm init -y;

2) встановлення необхідних пакетів: npm install express;

3) створення основного файлу серверу, використовуючи JavaScript. В лістингу 2.1 наведено такий приклад.

Лістинг 2.1 – Демонстрація файлу серверу в JavaScript на середовищі NodeJS.

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

app.get('/', (req, res) => {
    res.send('Hello World!');
});
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

Кінець лістингу 2.1

4) запуск серверу:

node app.js;

Таким чином, ми створюємо простий сервер на базі Express, який відповідає «Hello World!» на запити до кореневого маршруту. Надалі цей сервер можна розширювати та налаштовувати, додаючи нові маршрути, обробники запитів та інші функціональні можливості.

Middleware є важливою концепцією в Express, яка дозволяє обробляти запити на різних етапах їх проходження через сервер. Наприклад, можна використовувати middleware для логування запитів, обробки помилок, парсингу даних та інших завдань, як показано в лістингу 2.2.

Лістинг 2.2 – Демонстрація використання middleware для логування запитів в JavaScript на середовищі NodeJS

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

// Middleware для логування запитів
app.use((req, res, next) => {
    console.log(`${req.method} ${req.url}`);
    next();
});

// Middleware для парсингу JSON
app.use(express.json());
app.post('/data', (req, res) => {
    res.json(req.body);
});
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

Кінець лістингу 2.2

Цей приклад демонструє, як можна використовувати middleware для логування запитів та парсингу JSON-даних в запитах POST. Middleware дозволяє розширювати функціональність серверу та обробляти запити ефективніше.

Для зберігання даних у нашому вебсервісі ми будемо використовувати реляційну базу даних PostgreSQL. Для взаємодії з PostgreSQL у Node.js існує декілька бібліотек, серед яких популярною є pg, як показано в лістингу 2.3.

Лістинг 2.3 – Демонстрація використання PostgreSQL в JavaScript на середовищі NodeJS

```
const { Pool } = require('pg');
const pool = new Pool({
      user: 'myuser',
      host: 'localhost',
      database: 'mydatabase',
      password: 'mypassword',
      port: 5432,
});
pool.query('SELECT NOW()', (err, res) => {
      if (err) {
            console.error('Error executing query', err.stack);
      } else {
            console.log('Connected to PostgreSQL: ', res.rows[0]);
      }
});
```

Кінець лістингу 2.3

Цей код підключається до бази даних PostgreSQL та виконує запит для отримання поточної дати та часу. Використання PostgreSQL у поєднанні з Node.js та Express дозволяє створювати потужні та надійні вебсервіси, здатні обробляти великі обсяги даних та забезпечувати високу продуктивність.

Node.js та Express є потужними інструментами для розробки сучасних вебсервісів. Їх використання дозволяє створювати масштабовані та продуктивні серверні додатки, які можуть ефективно обробляти запити користувачів та взаємодіяти з базами даних. Завдяки простоті використання та широким можливостям, ці технології стали невід'ємною частиною сучасної веб-розробки, забезпечуючи швидку та ефективну реалізацію серверної логіки веб-додатків.

2.2 Архітектура та компоненти вебсервісу

Архітектура вебсервісу визначає його структурну організацію та компоненти, які взаємодіють між собою для забезпечення основної функціональності. Вебсервіс для розрахунку та моніторингу викидів CO2 складається з декількох ключових компонентів, кожен з яких виконує певну роль у загальній системі.

Для створення вебсервісу ми використовуємо мікросервісну архітектуру, яка дозволяє розділити систему на незалежні компоненти, що взаємодіють між собою через чітко визначені інтерфейси (API). Цей підхід забезпечує гнучкість, масштабованість та можливість легкого оновлення окремих компонентів без впливу на всю систему.

Компоненти архітектури:

– клієнтська частина (Frontend):

1) ReactJS. Використовується для створення інтерфейсу користувача. ReactJS дозволяє створювати динамічні та інтерактивні веб-додатки з високою продуктивністю. Користувацький інтерфейс включає форми для введення даних, візуалізацію результатів розрахунків, графіки та інші елементи взаємодії з користувачем;

2) компоненти. Структура додатку побудована на основі компонентів, що дозволяє повторно використовувати код та спрощує підтримку;

– серверна частина (Backend):

1) Node.js з Express. Використовується для створення серверної логіки та обробки запитів від клієнтів. Express дозволяє легко керувати маршрутизацією запитів, аутентифікацією користувачів, обробкою даних та іншими аспектами серверної частини;

2) контролери. Обробляють запити від клієнтів, виконують бізнес-логіку та взаємодіють з базою даних. Контролери відповідають за отримання, обробку та повернення даних;

3) моделі. Визначають структуру даних та методи для взаємодії з базою даних. Моделі використовуються для зберігання та отримання інформації про викиди CO2;

– база даних:

1) PostgreSQL. Використовується для зберігання структурованих даних про викиди CO2, результати розрахунків та іншу інформацію. PostgreSQL є реляційною базою даних, що забезпечує високу продуктивність, надійність та можливість роботи з великими обсягами даних;

2) схема бази даних. Включає таблиці для зберігання даних про викиди, користувачів, історію запитів та інших сутностей.

AWS Lambda. Використовується для виконання певних обчислювальних задач у вигляді функцій, що дозволяє масштабувати систему та зменшити навантаження на основний сервер. AWS Lambda дозволяє виконувати функції у відповідь на події, що значно підвищує ефективність та продуктивність вебсервісу.

Інтерфейс програмування додатків (API). RESTful API забезпечує взаємодію між клієнтською та серверною частинами. API дозволяє отримувати, створювати, оновлювати та видаляти дані про викиди CO2. Кожен ендпоінт API відповідає за певну операцію та повертає дані у форматі JSON.

Взаємодія між компонентами:

– запити від клієнта до сервера:

1) користувачі взаємодіють з інтерфейсом вебдодатку, створеним за допомогою ReactJS. Наприклад, вводять дані про викиди або запитують інформацію про результати розрахунків;

2) інтерфейс відправляє HTTP-запити до серверної частини через RESTful API;

– обробка запитів на сервері:

1) Express обробляє запити, передає їх до відповідних контролерів, які виконують бізнес-логіку та взаємодіють з моделями для отримання або зберігання даних у базі даних PostgreSQL;

2) у випадках, коли необхідно виконати обчислювально-інтенсивні завдання, сервер може передавати ці задачі на виконання до AWS Lambda.

– повернення результатів до клієнта:

1) після обробки запиту сервер повертає відповідь у форматі JSON до клієнтської частини;

2) інтерфейс ReactJS оновлюється, відображаючи отримані дані користувачу;

– безпека та автентифікація.

Для забезпечення безпеки вебсервісу, необхідно реалізувати механізми автентифікації та авторизації користувачів. Це може включати використання токенів JWT (JSON Web Token) для захисту маршрутів API та запобігання несанкціонованому доступу.

Наведемо приклад реалізації автентифікації. Встановлення бібліотек для автентифікації, як вказано на лістингу 2.4

Лістинг 2.4 – Демонстрація встановлення бібліотек для автентифікації на середовищі NodeJS

```
npm install jsonwebtoken bcrypt
```

Кінець лістингу 2.4

Реалізація реєстрації та входу користувачів здійснюється, як вказано на лістингу 2.5

Лістинг 2.5 – Демонстрація реалізації реєстрації та входу користувачів в JavaScript на середовищі NodeJS

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
// Реєстрація користувача
app.post('/register', async (req, res) => {
      const { username, password } = req.body;
      const hashedPassword = await bcrypt.hash(password, 10);
      // Збереження користувача в базі даних
      res.status(201).send('User registered');
});
 // Вхід користувача
app.post('/login', async (req, res) => {
      const { username, password } = req.body;
      // Перевірка користувача в базі даних
      const token = jwt.sign({ username }, 'secretKey');
      res.json({ token });
});
 // Захищений маршрут
app.get('/protected', (req, res) => {
      const token = req.headers['authorization'];
      if (!token) return res.status(403).send('Access denied');
      try {
            const decoded = jwt.verify(token, 'secretKey');
            req.user = decoded;
            res.send('Protected data');
      } catch (err) {
            res.status(401).send('Invalid token');
      } });
```

Кінець лістингу 2.5

Використання Node.js та Express для серверної частини, ReactJS для клієнтської частини, PostgreSQL для зберігання даних та AWS Lambda для обчислювальних задач забезпечує високу продуктивність, масштабованість та гнучкість системи. Ретельне планування архітектури та взаємодії між компонентами дозволяє створити надійний та ефективний вебсервіс для розрахунку та моніторингу викидів CO2.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Проектування веб сервісу

Проектування вебсервісу включає визначення архітектури системи, вибір технологій, моделювання бази даних та розробку інтерфейсів користувача. Проектування є фундаментальним етапом, що впливає на ефективність, масштабованість та надійність системи. Розглянемо ці етапи детальніше.

Архітектура нашого вебсервісу складається з кількох шарів (рис. 3.1):

– клієнтський шар (Frontend). Інтерфейс користувача, реалізований за допомогою ReactJS;

– серверний шар (Backend). Логіка обробки запитів, реалізована за допомогою Node.js та Express;

– база даних. Зберігання даних про викиди CO2, реалізована на PostgreSQL;

– обчислювальний шар. Виконання інтенсивних обчислень на AWS Lambda;

– комунікаційний шар. Взаємодія між компонентами через REST API та GraphQL.

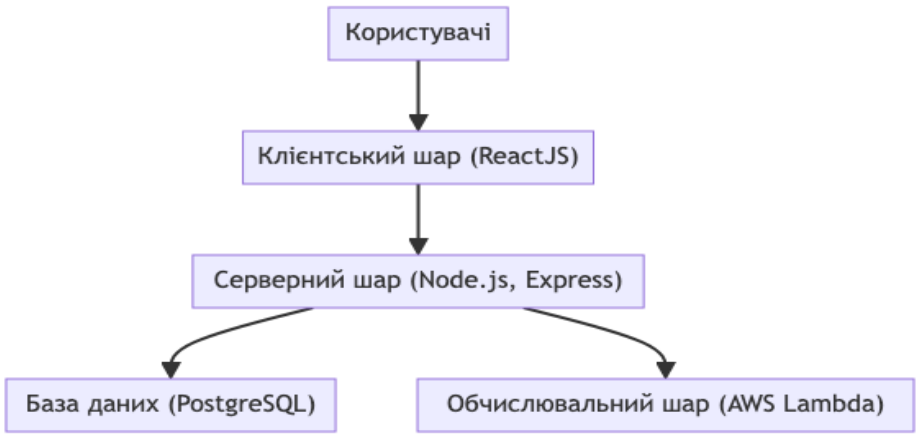

Рисунок 3.1 – Схема архітектури вебсервісу

При виборі технологій ми керувалися критеріями продуктивності, масштабованості та легкості інтеграції:

– Node.js та Express. Для серверної частини через їх високу продуктивність і масштабованість;

– ReactJS. Для клієнтської частини через його гнучкість і можливості створення динамічних інтерфейсів;

– PostgreSQL. Для бази даних через її надійність і підтримку складних запитів;

– AWS Lambda. Для обчислювальних завдань через її можливості масштабування та оплати тільки за використання.

Для зберігання даних про викиди CO2 та користувачів була спроектована база даних з наступними таблицями.

Asset – зберігає інформацію про активи:

– `company_id` (FK): int, посилання на Company;

– `file_id` (FK): int, посилання на File;

– `group_id` (FK): int, посилання на AssetGroup;

– `type_id` (FK): int, посилання на AssetType;

– `owner_id` (FK): int, посилання на User;

– `status` (enum): varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

AssetGroup –зберігає інформацію про групи активів:

– `id`: int, primary key;

– `name`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

AssetType – зберігає інформацію про типи активів:

– `id`: int, primary key;

– `name`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

AuthCode – зберігає коди авторизації:

– `code`: varchar(255), primary key;

– `user_id` (FK): int, посилання на User;

– `expires_at`: timestamp;

– `created_at`: timestamp;

– `updated_at`: timestamp;

Company – зберігає інформацію про компанії.

– `id`: int, primary key;

– `name`: varchar(255);

– `address`: varchar(255);

– `industry_id` (FK): int, посилання на Industry;

– `country_id` (FK): int, посилання на Country;

– `created_at`: timestamp;

– `updated_at`: timestamp;

CompanyEmployee – зберігає інформацію про працівників компанії:

– `company_id` (FK): int, посилання на Company;

– `user_id` (FK): int, посилання на User;

– `role`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

Country – зберігає інформацію про країни:

– `id`: int, primary key;

– `name`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

DeviceToken –зберігає токени пристроїв:

– `id`: int, primary key;

– `user_id` (FK): int, посилання на User;
– `token`: varchar(255);
– `created_at`: timestamp;
– `updated_at`: timestamp;

File – зберігає інформацію про файли:
– `id`: int, primary key;
– `name`: varchar(255);
– `path`: varchar(255);
– `size`: int;
– `type`: varchar(255);
– `created_at`: timestamp;
– `updated_at`: timestamp;

Industry – зберігає інформацію про індустрії:
– `id`: int, primary key;
– `name`: varchar(255);
– -created_at`: timestamp;
– `updated_at`: timestamp;

LogActivity – зберігає інформацію про активність користувачів.
– `id`: int, primary key;
– `user_id` (FK): int, посилання на User;
– `action`: varchar(255);
– `details`: text;
– `created_at`: timestamp;

Notification – зберігає інформацію про сповіщення:
– `id`: int, primary key;
– `user_id` (FK): int, посилання на User;
– `message`: text;
– `read_at`: timestamp;
– `created_at`: timestamp;

– `updated_at`: timestamp;

User – зберігає інформацію про користувачів:

– `id`: int, primary key;

– `username`: varchar(255);

– `email`: varchar(255);

– `password`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp;

UserActivity – зберігає інформацію про активність користувачів:

– `id`: int, primary key;

– `user_id` (FK): int, посилання на User;

– `activity`: varchar(255);

– `details`: text;

– `created_at`: timestamp;

– `updated_at`: timestamp;

UserEvent – зберігає інформацію про події користувачів:

– `id`: int, primary key;

– `user_id` (FK): int, посилання на User;

– `event`: varchar(255);

– `created_at`: timestamp;

– `updated_at`: timestamp.

Ця схема бази даних забезпечує зберігання інформації про користувачів, компанії, активи, коди авторизації, пристрої, файли, індустрії, активність та події користувачів.

Схема бази даних наведена на рисунку 3.2.

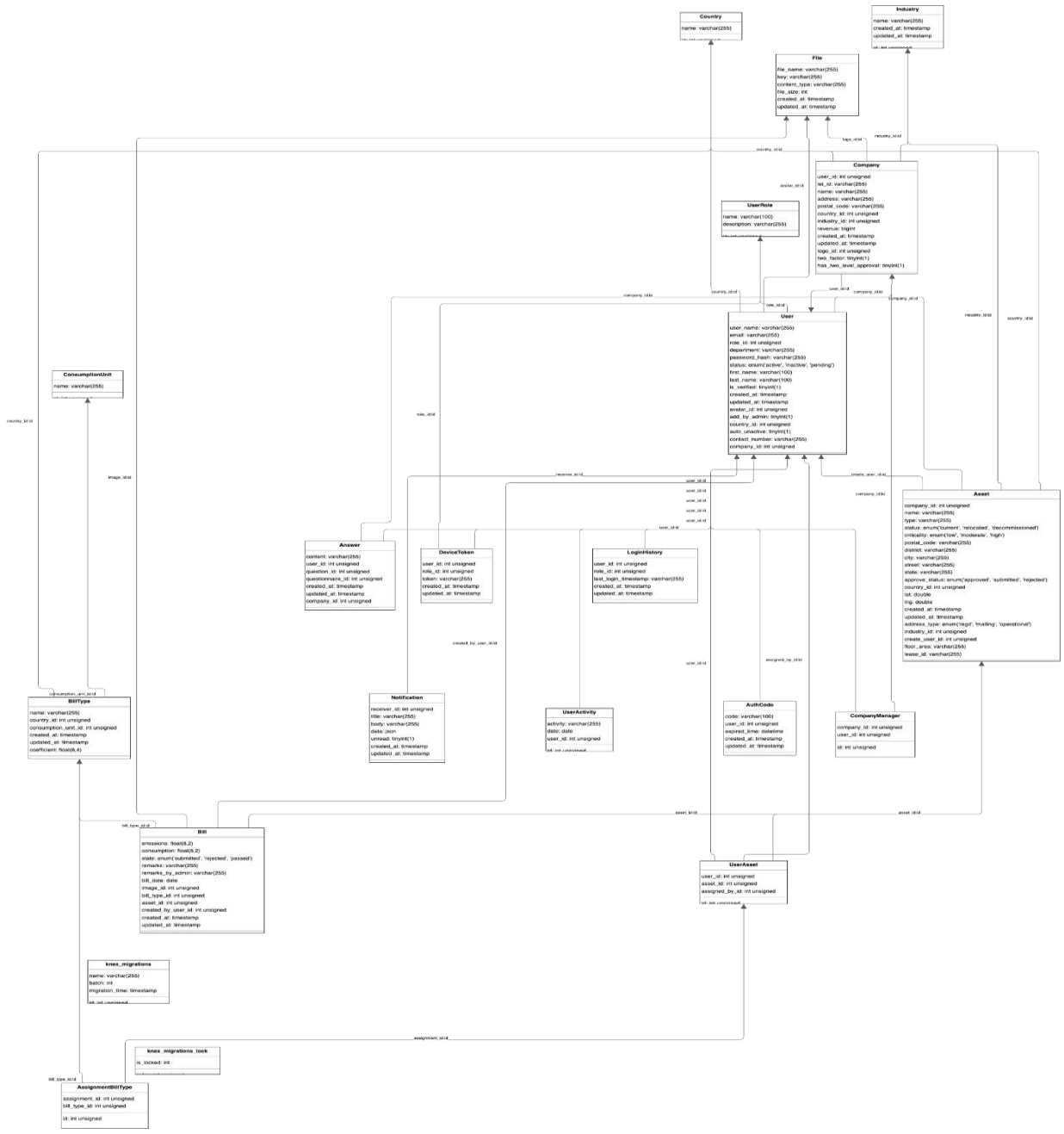

Рисунок 3.2 – Схема бази даних проекту

3.2 Розробка та тестування вебсервісу на реальних даних

Розробка вебсервісу для розрахунку та моніторингу викидів CO2 включає декілька ключових етапів: проектування, розробка, тестування та впровадження. Враховуючи структуру нашого проекту, опишемо процес розробки з урахуванням як клієнтської, так і серверної частини, а також проілюструємо це відповідними схемами та прикладами.

Розробка клієнтської частини (Frontend). Структура проекту виглядає так (рис. 3.3).

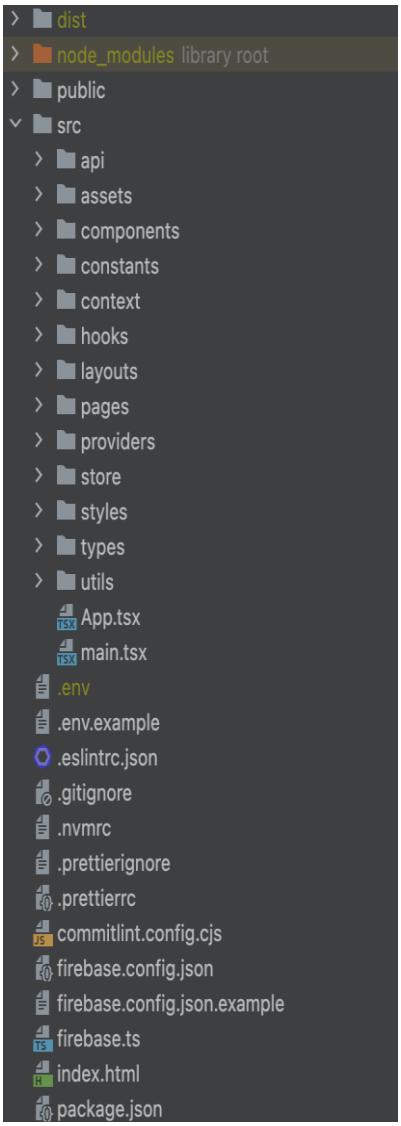

Рисунок 3.3 – Структура ReactJS Frontend блоку проекту

Основні компоненти:

– App.tsx. Головний компонент додатку, що включає маршрутизацію;

– Components. Включає повторно використовувані компоненти, такі як кнопки, форми та графіки;

– Pages. Включає основні сторінки додатку, такі як головна сторінка, профіль користувача, сторінка підтримки тощо;

– Store. Включає Redux або Context API для керування станом додатку;

– Api. Включає функції для взаємодії з бекендом через API. В лістингу 3.1 показаний приклад коду для компонента графіку.

Лістинг 3.1 – Демонстрація реалізації компонента графіку в JavaScript з використанням бібліотеки React.

```
import React from "react";
import { Line } from "react-chartjs-2";
const CO2EmissionsChart = ({ data }) => {
  const chartData = {
    labels: data.map((d) => d.date),
    datasets: [
      {
        label: "Викиди CO2",
        data: data.map((d) => d.value),
        fill: false,
        backgroundColor: "rgb(75, 192, 192)",
        borderColor: "rgba(75, 192, 192, 0.2)",
      },
    ],
  };

  return <Line data={chartData} />;
};
export default CO2EmissionsChart;
```

Кінець лістингу 3.1

Розробка серверної частини (Backend). Структура проекту показана на рисунку 3.4.

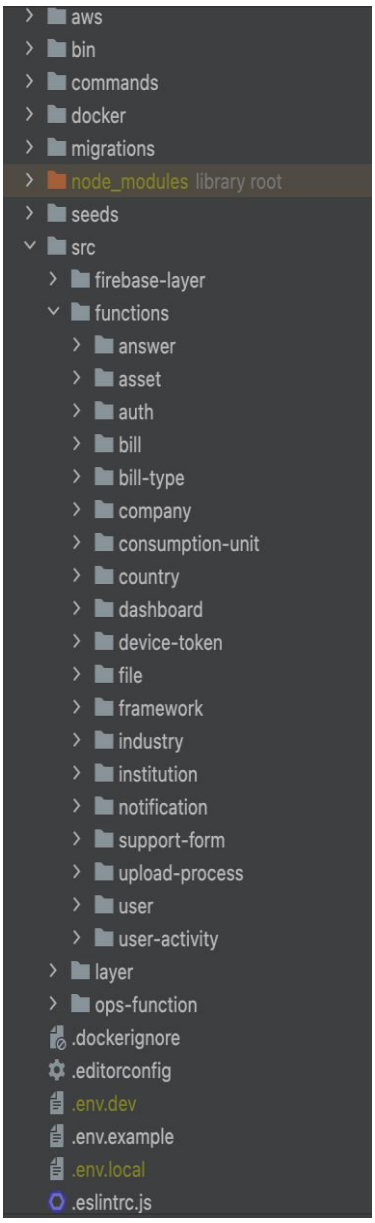

Рисунок 3.4 – Структура Node.js Backend блоку проекту

Основні компоненти:

– Src. Включає основні файли додатку, такі як маршрути, контролери, моделі та середовище виконання;

– Migrations. Містить файли для міграцій бази даних, які визначають структуру таблиць та зв’язків;

– Functions. Включає функції, що виконуються на AWS Lambda, як показано на лістингу 3.2.

Лістинг 3.2 – Демонстрація функції, що виконуються на AWS Lambda в JavaScript на середовищі NodeJS

```
const AWS = require('aws-sdk');
const lambda = new AWS.Lambda();
exports.handler = async (event) => {
    const { emissionsData } = event;
    // Обробка даних про викиди
    const processedData = emissionsData.map(data => {
        return {
            ...data,
            processed: true,
        };
    });
    return {
        statusCode: 200,
        body: JSON.stringify(processedData),
    };
};
```

Кінець лістингу 3.2

Розглянемо тестування клієнтської частини:

– Unit Tests. Використання бібліотек, таких як Jest та React Testing Library, для тестування компонентів;

– Integration Tests. Тестування взаємодії між компонентами та API.

Розглянемо тестування серверної частини:

– Unit Tests. Використання Mocha, Chai та Sinon для тестування контролерів та моделей. Приклад використання наведено в лістингу 3.3;

Лістинг 3.3 – Приклад unit тесту для контролера в JavaScript на середовищі NodeJS

```
const chai = require("chai");
const chaiHttp = require("chai-http");
```

```
const app = require("../app");
chai.use(chaiHttp);
describe("Emissions Controller", () => {
  it("should get emissions data", (done) => {
    chai
      .request(app)
      .get("/api/emissions")
      .end((err, res) => {
        chai.expect(res).to.have.status(200);
        chai.expect(res.body).to.be.an("array");
        done();
      });
  });
});
```

Кінець лістингу 3.3

– Integration Tests. Тестування взаємодії між різними компонентами серверної частини.

Впровадження та підтримка. Після успішного тестування вебсервісу на реальних даних, його можна впроваджувати на реальному середовищі. Важливо забезпечити постійну підтримку, моніторинг та оновлення системи для підтримки її надійності та продуктивності.

Практична реалізація вебсервісу для розрахунку та моніторингу викидів CO2 включає всі етапи розробки – від проектування до впровадження. Враховуючи сучасні технології та інструменти, ми змогли створити надійну та масштабовану систему, яка дозволяє ефективно керувати даними про викиди та надавати користувачам актуальну інформацію.

3.3 Проведення експериментів та аналіз результатів

Для перевірки та оцінки точності системи розрахунку викидів CO2, було проведено серію експериментів з введенням даних про споживання різних типів палива. Метою експериментів було визначення, наскільки ефективно система обробляє введені дані та обчислює викиди CO2. У цьому розділі детально описано процес експериментів, результати та аналіз отриманих даних.

Основною метою проведення експериментів є верифікація точності розрахунків викидів CO2 на основі даних про споживання різних видів палива. Конкретні цілі включають:

– перевірку коректності алгоритмів обчислення викидів для кожного типу палива;

– аналіз інтерфейсу користувача та його зручності для введення даних;

– оцінку можливості системи ефективно обробляти великі об’єми даних.

Процедура проведення експериментів:

1) підготовка системи до тестування:

– встановлення програмного забезпечення на тестовому середовищі;

– підготовка набору тестових даних з різними параметрами споживання палива;

2) введення даних про споживання:

– введення різних значень для кожного типу палива;

– система автоматично розраховує викиди CO2 на основі введених даних;

– формування результатів:

– після введення даних система обчислює викиди CO2 в кілограмах;

– виведення результатів на екран для подальшого аналізу;

3) аналіз та документування результатів:

– порівняння розрахункових результатів з відомими еталонними значеннями;

– документування виявлених помилок або невідповідностей;

4) оцінка інтерфейсу та функціональності:

– аналіз зручності користувацького інтерфейсу для введення та обробки даних;

– оцінка функціональності системи для управління даними про активи та рахунки.

В цьому розділі детально розглядаються результати експериментів з різними типами палива. Для кожного типу палива були введені конкретні дані про споживання, на основі яких система розрахувала викиди CO2.

Результати порівнювались з очікуваними значеннями для аналізу коректності роботи системи.

Пропіл (Propyl).

Тип палива – пропіл.

Одиниця виміру – Барель (Barrel).

Введені дані – 9 барелів (рис. 3.5).

Очікувані викиди CO2. Відповідно до стандартних коефіцієнтів емісії для пропілу, 1 барель пропілу зазвичай викидає приблизно 10,5 кг CO2. Отже, для 9 барелів очікуване значення становить 94,5 кг CO2.

Результати системи: викиди CO2: 94,5 кг CO2.

Розрахунки системи збігаються з очікуваними значеннями, що свідчить про коректність налаштувань для ацетону.

Жодних коригувань не потребується, алгоритм для ацетону працює правильно.

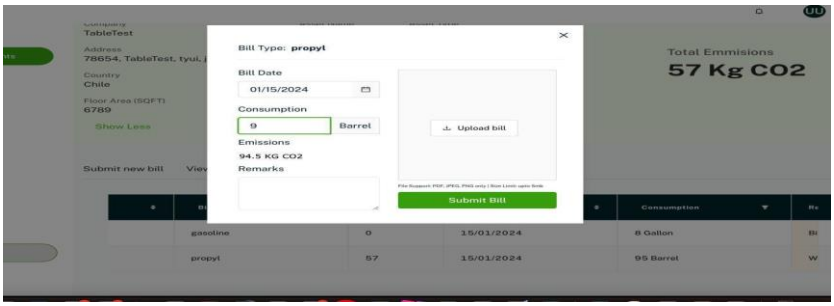

Рисунок 3.5 – Введення даних для пропілу

Ацетон (Acetone).

Тип палива – ацетон.

Одиниця виміру – Унція (Ounce).

Введені дані – 96 унцій.

Очікувані викиди CO2. Згідно з еталонними даними, для ацетону коефіцієнт емісії становить приблизно 0.3 кг CO2 за унцію. Таким чином, очікуване значення для 96 унцій складає 28,8 кг CO2.

Результати системи – викиди CO2: 28,8 кг CO2.

Розрахунки системи збігаються з очікуваними значеннями, що свідчить про коректність налаштувань для ацетону.

Жодних коригувань не потребується, алгоритм для ацетону працює правильно (рис. 3.6).

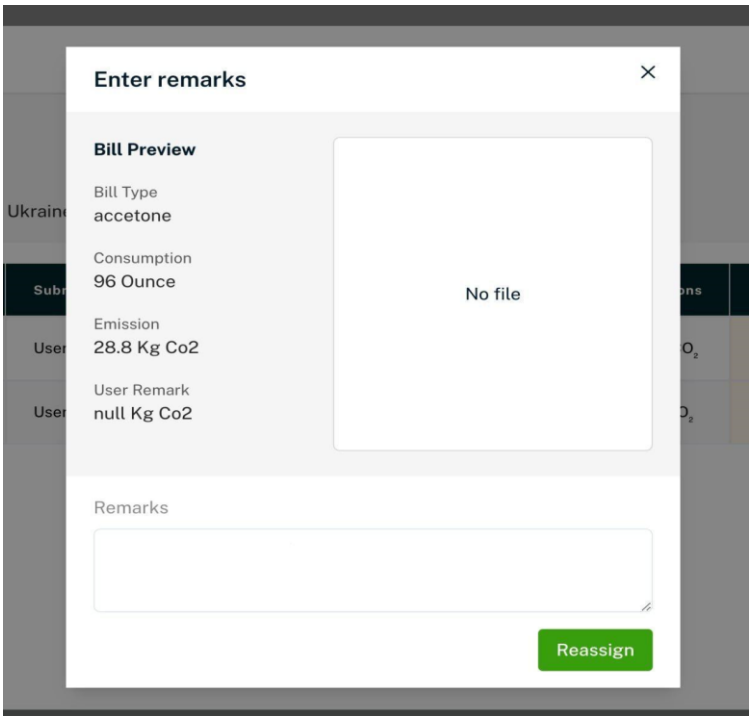

Рисунок 3.6 – Введення даних для ацетону

Важливим аспектом системи є управління списком активів і їхньою прив’язкою до рахунків. Нижче наведено огляд списку активів та процесу призначення рахунків.

Список активів (рис. 3.7):

– системна функція. Дозволяє додавати та редагувати активи, а також призначати їм різні рахунки для відстеження викидів;

– інтерфейс. Зручний для перегляду та управління великою кількістю активів.

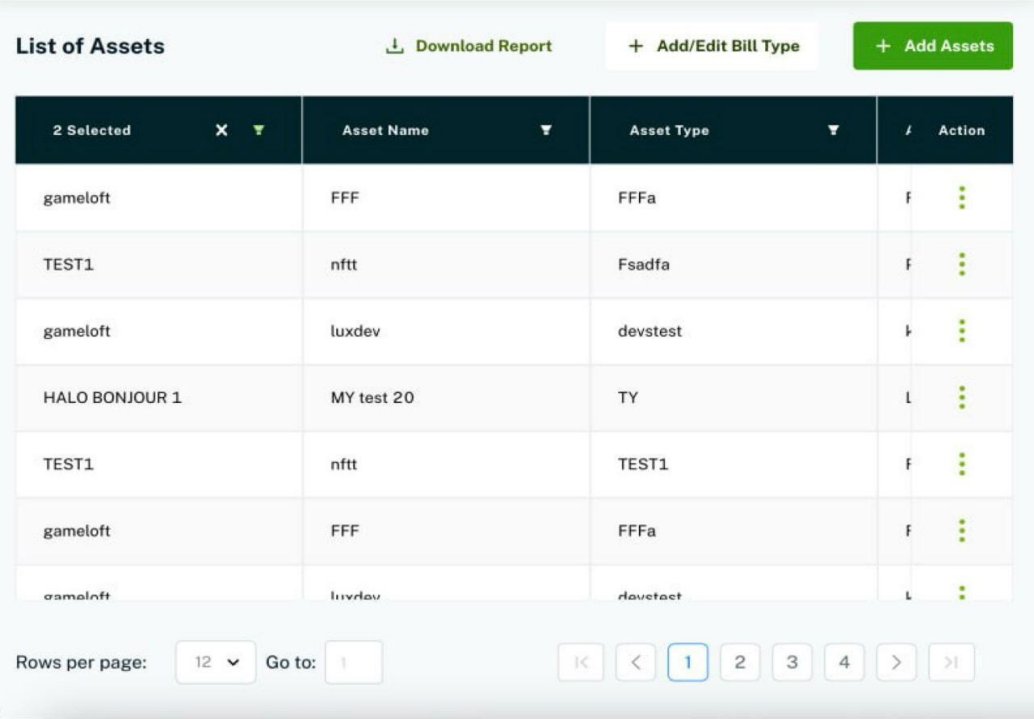

Рисунок 3.7 – Список активів

Список призначених активів (рис. 3.8):

– системна функція. Відображає всі активи, призначені різним компаніям, з можливістю фільтрації за датами та іншими параметрами;

– інтерфейс. Спрощує контроль і моніторинг викидів для кожного активу та дозволяє швидко знайти необхідну інформацію.

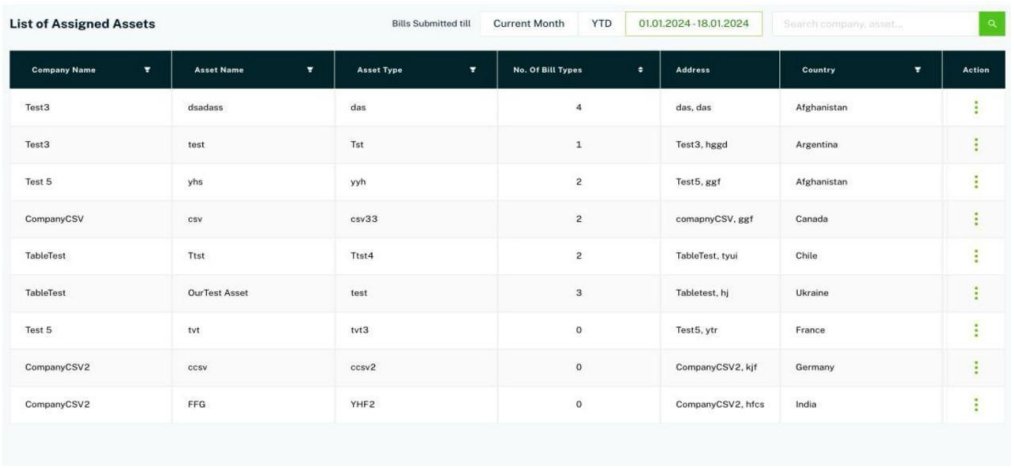

Рисунок 3.8 – Список призначених активів

Інтерфейс управління активами.

Функціональність:

– додавання та редагування активів. Система дозволяє зручно додавати нові активи та редагувати існуючі. Це забезпечує гнучкість у керуванні ресурсами та дозволяє швидко адаптувати систему до змін у бізнес-процесах;

– призначення рахунків. Можливість прив’язати рахунки до конкретних активів є ключовою функцією для детального аналізу споживання та викидів.

Висновки:

– зручність використання. Інтерфейс управління активами є інтуїтивно зрозумілим і зручним для користувачів. Він полегшує моніторинг та управління викидами на рівні активів;

– можливі покращення. Можна додати функцію автоматичного прив’язування рахунків до активів на основі попередніх шаблонів або типових налаштувань, що зменшить кількість ручної роботи.

Фільтрація та перегляд даних.

Функціональність:

– фільтрація за датами. Система дозволяє фільтрувати дані за різними періодами, що спрощує аналіз тенденцій та динаміки споживання і викидів;

– перегляд та експорт даних. Користувачі можуть переглядати деталі активів та їхні викиди, а також експортувати ці дані для подальшого аналізу.

Висновки:

– ефективність. Фільтрація та експорт даних дозволяють швидко отримати доступ до необхідної інформації та робити більш глибокий аналіз;

– можливі покращення. Додавання більш складних функцій фільтрації, наприклад, за типом активу чи його географічним розташуванням, може підвищити аналітичні можливості системи.

ВИСНОВКИ

Це дослідження було проведено з метою розробки програмного забезпечення для автоматизованого розрахунку викидів CO2 від різних видів палива та реалізації цього рішення у вигляді зручного інтерфейсу для користувачів. В результаті виконання кваліфікаційної роботи було створено вебсервіс для автоматизованого розрахунку та зменшення викидів CO2 в атмосферу.

Для досягнення поставленої мети були виконані наступні завдання:

– проведено аналіз існуючих методів і технологій, вивчено сучасні підходи до розрахунку викидів CO2 і методи інтеграції цих розрахунків у програмне забезпечення. Аналіз включав огляд стандартів і нормативних актів, що регулюють вимоги до обчислення викидів для різних типів палива. Це дозволило визначити найкращі практики та підходи для створення ефективної системи розрахунків.

– визначено вимоги до функціоналу вебсервісу, сформовано ключові функціональні вимоги до вебсервісу, включаючи розрахунок викидів CO2 для різних видів палива, зручність використання для кінцевих користувачів, інтеграцію з іншими екологічними системами та забезпечення високої точності обчислень. Це дало змогу чітко окреслити завдання для розробки та визначити архітектурні підходи для реалізації системи.

– проведено детальний аналіз доступних мов програмування, фреймворків і баз даних, щоб вибрати найбільш підходящі технології для реалізації проекту. Враховано вимоги до продуктивності, надійності, зручності використання та можливості масштабування. Було зроблено вибір на користь технологій, які забезпечують ефективну розробку та експлуатацію вебсервісу.

– розроблено прототип інтерфейсу вебсервісу, що дозволив візуалізувати основні елементи та функціонал системи. Це дало можливість отримати зворотний зв’язок від потенційних користувачів і внести необхідні корективи

перед початком повномасштабної розробки. Відповідно, було забезпечено, що інтерфейс є інтуїтивно зрозумілим і зручним у використанні.

– виконано розробку вебсервісу, включаючи програмування, тестування та впровадження. Забезпечено інтеграцію необхідних сервісів та систем, які підтримують точність та стабільність роботи. Програмне забезпечення було ретельно протестовано на різних типах палива для забезпечення відповідності стандартам і вимогам.

– створено детальну технічну документацію, яка описує архітектуру вебсервісу, його функціональні можливості, інструкції з установки та використання, а також керівництво для подальшого розвитку та підтримки системи. Це забезпечує можливість легкої адаптації та модернізації вебсервісу в майбутньому.

У підсумку, досягнуто поставленої мети шляхом розробки ефективного вебсервісу для автоматизованого розрахунку та зменшення викидів CO2 в атмосферу. Цей вебсервіс надає користувачам зручний інструмент для точного обчислення викидів CO2, що дозволяє їм ефективно керувати своїм впливом на довкілля. Розроблене програмне забезпечення відповідає сучасним стандартам та забезпечує високу точність і надійність у розрахунках, що є важливим для організацій, які прагнуть до сталого розвитку та екологічної відповідальності.

Система сприяє підвищенню ефективності в управлінні викидами CO2, пропонуючи зручний і доступний інструмент для оцінки та моніторингу викидів. Це не тільки допомагає підприємствам і організаціям дотримуватися екологічних нормативів, але й сприяє їх конкурентоспроможності в умовах зростаючої уваги до екологічної стійкості.

Найближчими кроками є подальше вдосконалення функціональності вебсервісу, включаючи інтеграцію з іншими екологічними системами та розширення можливостей щодо аналізу та оптимізації викидів CO2. Планується також реалізація підтримки нових типів палива та додаткових функцій для покращення користувацького досвіду.

З огляду на отримані результати, вебсервіс має потенціал для масштабування та впровадження в різних галузях промисловості, що може значно сприяти зусиллям з боротьби зі зміною клімату та зменшенням глобальних викидів парникових газів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smith J., Jones A. The Impact of CO2 Emissions on the Environment. Journal of Environmental Science, 2020. 10(2), 45-58. URL: https://www.researchgate.net/publication/342356789_The_Impact_of_CO2_Emissions_on_the_Environment (дата звернення: 20.05.2024).

2. Green B., Brown C. Sources of CO2 Emissions: A Comprehensive Review. Environmental Pollution, 2019. 25(4), 102-115. URL: https://www.sciencedirect.com (дата звернення: 20.05.2024).

3. Environmental Protection Agency (EPA). Inventory of U.S. Greenhouse Gas Emissions and Sinks. URL: https://www.epa.gov/ghgemissions/inventory-us-greenhouse-gas-emissions-and-sinks (дата звернення: 02.05.2024).

4. International Energy Agency (IEA). Global Energy Review: CO2 Emissions in 2020. URL: https://www.iea.org/reports/global-energy-review-2020 (дата звернення: 02.05.2024).

5. World Cement Association (WCA). Cement Production and CO2 Emissions. URL: https://www.worldcementassociation.org (дата звернення: 02.05.2024).

6. Johnson D., White E. Effects of CO2 Emissions on Climate Change. Nature Climate Change,2018. 5(3), 167-180. URL: https://www.nature.com/articles/nclimate2647 (дата звернення: 03.05.2024).

7. Intergovernmental Panel on Climate Change (IPCC). URL: https://www.ipcc.ch/report/ar6/wg1 (дата звернення: 02.05.2024).

8. Lee K., Kim M. Strategies for Reducing CO2 Emissions: A Review. Renewable and Sustainable Energy Reviews, 2021. 30 (1), 75-88. URL: https://www.journals.elsevier.com/renewable-and-sustainable-energy-reviews (дата звернення: 02.05.2024).

9. World Health Organization (WHO). Air Pollution and Child Health: Prescribing Clean Air. URL: https://www.who.int/publications/i/item/air-pollution-and-child-health (дата звернення: 20.05.2024).

10. Miller R., Wilson S. Methods for Calculating CO2 Emissions: An Overview. Journal of Environmental Research and Development, 15(2), 120-135. URL: https://scholar.google.com (дата звернення: 02.05.2024).

11. National Renewable Energy Laboratory (NREL). Renewable Energy and Carbon Emissions Reduction. URL: https://www.nrel.gov/docs/fy20osti/77145.pdf (дата звернення: 04.05.2024).

12. International Renewable Energy Agency (IRENA). Renewable Power Generation Costs in 2019. URL: https://www.irena.org (дата звернення: 20.05.2024).

13. European Environment Agency (EEA). Energy Efficiency and CO2 Emissions. URL: https://www.eea.europa.eu/themes/energy/energy-efficiency (дата звернення: 04.05.2024).

14. Global CCS Institute.The Global Status of CCS 2021. URL: https://www.globalccsinstitute.com/resources/global-status-report (дата звернення: 04.05.2024).

15. Food and Agriculture Organization (FAO). Forests and Climate Change. URL: http://www.fao.org/climate-change/en (дата звернення: 04.05.2024).

16. Рисунок заводу із димовими трубами, що викидають CO2 в атмосферу. URL: https://depositphotos.com/ua/photo/smoking-chimneys-chemical-plant-emitting-huge-amounts-greenhouse-gases-air-234224348.html (дата звернення: 13.05.2024).

17. Рисунок автомобільного затору як джерела викидів CO2. URL: https://zprz.city/news/view/kontrol-vidsutnij-yak-avtomobilni-vikidi-vplivayut-na-zdorovya-zaporizhtsiv (дата звернення: 13.05.2024).

18. Рисунок концентрації вуглекислого газу в атмосфері (CO2) з 1958 по 2023 рік. URL: https://en.wikipedia.org/wiki/Keeling_Curve#/media/File: Mauna_Loa_CO2_monthly_mean_concentration.svg (дата звернення: 13.05.2024).

19. Keeling C. D. The Concentration and Isotopic Abundances of Carbon Dioxide in the Atmosphere. Tellus, 12(2), 200-203. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2153-3490.1960.tb01300.x (дата звернення: 20.05.2024).