Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Машталер Дмитрій Сергійович*
(ПІБ)

академічної групи *121-20-2*
(шифр)

спеціальності *121 «Інженерія програмного забезпечення»*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення клієнтської частини веб застосунку екологічного моніторингу*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-20-2* (група) *Машталера Дмитрія Сергійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення клієнтської частини веб застосунку екологічного моніторингу*

затверджена наказом ректора НТУ «ДП» від *23.05.2024 р.* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів практик та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***10.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***14.06.2024 р.***

Завдання видав _____ *проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Машталер Д.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.04.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *17.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 89 с., 27 рис., 1 табл., 3 дод., 25 джерел.

Сучасні екологічні виклики потребують своєчасного та точного моніторингу для запобігання катастрофам і покращення якості життя. Веб-застосунки можуть автоматизувати та оптимізувати ці процеси, підвищуючи їх ефективність. Розробка клієнтської частини веб-застосунку екологічного моніторингу забезпечить користувачів зручним доступом до актуальних даних і сприятиме оперативному прийняттю рішень. Об’єкт розробки: програмне забезпечення клієнтської частини веб застосунку екологічного моніторингу.

Мета кваліфікаційної роботи полягає у розробці програмного забезпечення для клієнтської частини веб-застосунку екологічного моніторингу. Основною метою є підвищення ефективності та ергономічності процесів моніторингу екологічних показників шляхом створення ефективного та зручного веб-інтерфейсу для користувачів.

У вступі роботи розглянуто сучасний стан проблеми екологічного моніторингу, визначена його актуальність та наведено обґрунтування необхідності розробки програмного забезпечення для поліпшення цих процесів.

У першому розділі проведено аналіз предметної галузі екологічного моніторингу, сформульовано вимоги до розробки, визначено актуальні завдання та призначення програми.

У другому розділі виконано дослідження наявних рішень, обгрунтовано вибір технологій та засобів для розробки веб-застосунку, описано структуру програмного забезпечення та алгоритми його функціонування.

В економічному розділі проведено оцінку трудомісткості розробки, розраховано вартість та необхідний час для створення програмного забезпечення.

Практичне завдання передбачає аналіз існуючих рішень у галузі вузалізації даних екологічного моніторингу, розробку дизайну та прототипів інтерфейсу, а також створення веб-застосунку згідно з розробленим дизайном.

Ключові слова: ЕКОЛОГІЧНИЙ МОНІТОРИНГ, ДОСВІД КОРИСТУВАЧА, ВЕБ-ІНТЕРФЕЙС, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, HTML, CSS, JS, БАЗА ДАНИХ.

ABSTRACT

Explanatory note: 89 p., 27 figures, 1 table, 3 additions, 25 sources.

Current environmental issues will require timely and accurate monitoring to avoid disasters and reduce the cost of living. Web stops can be automated and optimized to improve their efficiency. Development of the client part of the web-based environmental monitoring system will provide the customer with manual access to up-to-date data and quick decision-making. Object of development: development of software for the client part of an environmental monitoring web application.

The purpose of the qualification work is to develop software for the client part of an environmental monitoring web application. The main goal is to improve the efficiency and ergonomics of environmental performance monitoring processes by creating an efficient and user-friendly web interface for users.

In the introduction of the paper, the current state of the problem of environmental monitoring is considered, its relevance is determined, and the justification for the need to develop software to improve these processes is given.

In the first chapter, an analysis of the subject area of environmental monitoring is carried out, requirements for development are formulated, current tasks and purpose of the program are determined.

In the second chapter, the research of available solutions is carried out, the choice of technologies and tools for developing a web application is justified, the software structure and algorithms for its functioning are described.

The economic section evaluates the complexity of development, calculates the cost and time required to create software.

The practical task involves analyzing existing solutions in the field of environmental monitoring data knots, developing interface design and prototypes, as well as creating a web application according to the developed design.

Keywords: ENVIRONMENTAL MONITORING, USER EXPERIENCE, WEB INTERFACE, SOFTWARE, HTML, CSS, JS, DATABASE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

СУБД – система управління базами даних

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

JS – Java Script

UX – User Experience

ПК – персональний комп’ютер

API - Application Programming Interface

ВСТУП

В нас час дуже актульно звертати увагу на екологічні проблеми, так як суспільство не дуже сильно квапиться про стан наволишнього середовища. Свідомість людей втрачається через рутину і це погано, тому суспільству необхідно більше впроваджувати різноманітні механізми моніторингу навколишнього середовища.

Одним з ключових складових таких механізмів є веб-додатки [5] для екологічного моніторингу, які надають можливість широкому колу зацікавлених сторін, а саме науковим дослідникам, громадянам, органам Влади, у школах та промисловості. Для них важливо отримувати доступ до актуальної інформації і взаємодіяти з даними в режимі реального часу [1].

Метою даної бакалаврської роботи є дослідження та розробка клієнтської частини веб-застосунку для екологічного моніторингу. Робота спрямована на вивчення та застосування сучасних методів, принципів та інструментів для створення ефективної та зручної для користувача клієнтської частини веб-додатку, що забезпечує доступ до актуальної інформації про стан навколишнього середовища в режимі реального часу.

Шляхом аналізу наукової літератури, використання сучасних методів розробки та проведення практичних досліджень, ця дипломна робота ставить за мету розкрити основні принципи, методи та інструменти, що лежать в основі успішної розробки клієнтської частини веб-додатків для моніторингу довкілля [10].

Об'єктом розробки є клієнтська частина веб-застосунку для екологічного моніторингу. Ця частина включає в себе інтерфейс користувача, візуалізацію даних, інтерактивні компоненти та механізми взаємодії з серверною частиною додатку. Основними функціями клієнтської частини є надання зручного доступу до даних екологічного моніторингу, відображення цих даних у зрозумілому та наочному вигляді, а також забезпечення можливості взаємодії користувачів з даними та їх аналізу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

В нас час дуже важливо спостерігати за екологією, бо екологічний стан нашої планети напряму впливає на якість нашого життя і майбутність нащадків. Для спостереження за екологією придумали наукову дисципліну – Екологічний моніторинг. Р. Мун [2] ввів поняття таке як «моніторинг довкілля» [10], але для того щоб зрозуміти це поняття потрібно визначитись з поняттям монітронгу, а саме - система постійного спостереження за явищами і процесами, що проходять в навколишньому середовищі, суспільстві, результати якого слугують для обґрунтування управлінських рішень по забезпеченню безпеки людей.

Тепер перейдемо до поняття Екологочного моніторингу - Екологочний моніторинг [7] це система повторних спостережень одного і більше елементів навколишнього природного середовища (ОПВ) в просторі і в часі. Коли ми знаємо що означають ці визначення можемо перейти к об’єктам моніторингу.

Об’єкти моніторингу це складні системи які закономірно об’єднані в єдине ціле. Спостереження за ними допомагає зрозуміти динаміку цих систем і їх вплив на навколишнє середовище. Суб'єкти моніторингу є носіями моніторингових функцій і включають організації, структури або окремих людей, які здійснюють ці функції. Вони відповідають за збір, аналіз та інтерпретацію даних, необхідних для моніторингу об'єктів. В результаті спостережень за об'єктами моніторингу в наше розпорядження надходить комплекс моніторингових показників - сукупність показників, які повинні забезпечити цілісне уявлення про стан системи і його динаміці (якісних і кількісних змінах).

Екологічний моніторинг включає в себе два ключові аспекти моніторингової діяльності, які тісно пов'язані між собою та забезпечують комплексний підхід до спостереження за станом навколишнього середовища.

Першим аспектом є організація моніторингу та проведення спостережень. Цей етап передбачає ретельне планування та підготовку до здійснення спостережень за об'єктами моніторингу. Фахівці розробляють методи та плани спостереження, визначають оптимальні локації для збору даних, встановлюють необхідні системи спостереження та проводять вимірювання. Ця діяльність є фундаментальною для забезпечення якісних та репрезентативних даних про стан довкілля.

Другим важливим аспектом є збір і обробка інформації з отриманням результатів моніторингу та рекомендацій по їх реалізації, що можна охарактеризувати як інформаційно-аналітичне забезпечення. На цьому етапі здійснюється збір даних, отриманих під час спостережень, їх аналіз та обробка. Метою цієї діяльності є формування цілісної картини екологічного стану та розробка обґрунтованих рекомендацій. На основі отриманих результатів фахівці розробляють пропозиції щодо управлінських рішень та можливих заходів, спрямованих на покращення екологічного стану або ефективне управління об'єктами моніторингу.

Ці два основних види моніторингової діяльності взаємопов'язані і спрямовані на забезпечення систематичного спостереження, аналізу та управління станом об'єктів та процесів в навколишньому середовищі.

Є різні методи контролю, або моніторингу стану наволишнього середовища. Найпоширенішими видами моніторингу є хімічний, геофізічний, біологічний, екобіохімічний, коплексний.

Наприклад:

1) Геофізичний моніторинг [2] - систематичне спостереження за влипвом фізичних процесів на природне сиридовище (повені, землетруси, цунамі).

2) Біологічний моніторинг [2] - використовують організми для оцінювання змін в середовищі на основі їх поведінки.

3) Екобіохімічний моніторинг [2] – це оцінка двох складових наволищного середовища, біологічного та хімічного.

4) Комплексний екологічний моніторинг [2] - передбачає постійну оцінку екологічних умов середовища та біологічних об'єктів, а також стану екосистем для розробки коригувальних дій при виявленні відхилень від норми.

5) Хімічний моніторинг [2] - спостереження за хімічним складом різних складових навколишнього середовища (атмосфера, води, грунт, рослини, тварини) з метою визначення рівня забруднення шкідливими речовинами. Основне завдання - виявлення високотоксичних інгредієнтів і контроль їх поширення.

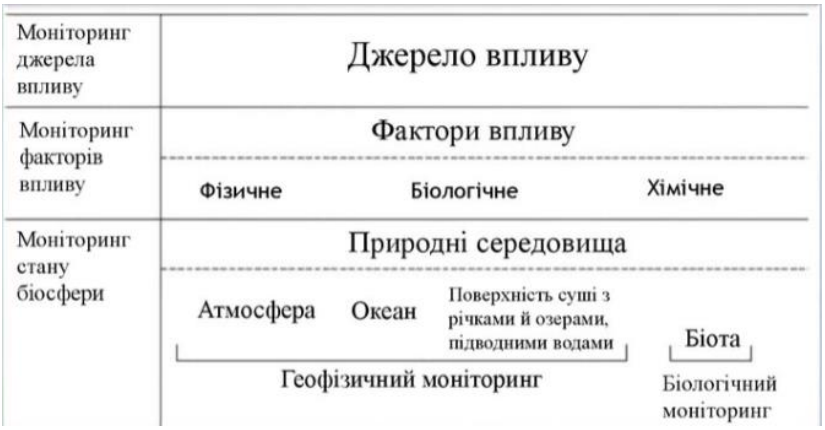

Рис 1.1. Класифікація екологічного моніторингу

На рис 1.1. було показано класифікацію екологічного моніторингу за видами джерел впливу та факторами які підлягають моніторингу.

В екологічному моніторингу навколишнього природного середовища виділяються три рівні залежно від масштабів об’єктів спостереження.

Локальний моніторинг є першим рівнем екологічного спостереження, де об’єктами спостереження є конкретні місцевості або окремі зони з невеликими розмірами, зазвичай до десятків квадратних кілометрів. Цей рівень моніторингу здійснюється на території окремих підприємств, міст або

визначених ділянок ландшафтів. Метою локального моніторингу є систематичне збирання даних про стан довкілля в обмеженому просторі для оцінки впливу конкретних діяльностей на природне середовище. Тут використовуються точні методи спостереження і збору даних, що дозволяють виявляти зміни на місцевому рівні і вчасно реагувати на потенційні проблеми.

Регіональний моніторинг охоплює більші території, такі як адміністративно-територіальні одиниці або окремі економічні і природні регіони. В рамках цього рівня моніторингу вивчаються тенденції розподілу забруднюючих речовин і їх взаємодія в біосфері на великій території. Регіональний моніторинг дозволяє виявляти проблеми, що виникають внаслідок діяльності кількох підприємств чи населених пунктів у межах одного регіону. Важливим аспектом є визначення джерел забруднення і впливу на природне середовище, а також управління цими процесами для збереження стабільності регіонального екологічного балансу.

Глобальний моніторинг [6] є найвищим рівнем екологічного спостереження і відображає міжнародний підхід до контролю за станом навколишнього середовища. Метою глобального моніторингу є виявлення та аналіз глобальних екологічних тенденцій, таких як зміни клімату, втрата біорізноманіття, забруднення океанів та інші глобальні екологічні проблеми. Цей рівень моніторингу включає міжнародне співробітництво через участь в різних міжнародних конвенціях і деклараціях, спрямованих на збереження навколишнього середовища для майбутніх поколінь.

Ці рівні взаємодіють і узгоджуються для забезпечення комплексного моніторингу навколишнього середовища на різних масштабах, що дозволяє ефективно контролювати і зберігати екологічну стійкість.

Рис. 1.2. Рівні екологічного моніторингу

На рис 1.2. було показано рівні екологічного моніторингу з прикладами [8]. З усієї цієї інформації можна сказати що, екологічний моніторинг є систематичним процесом спостереження, аналізу та управління станом навколишнього середовища. Його ціль - забезпечення якості життя та збереження природних ресурсів для майбутніх поколінь. Він включає організацію спостережень, збір та обробку інформації, а також рекомендації для управлінських рішень. Класифікація моніторингу залежить від видів джерел впливу та масштабів об'єктів спостереження. Локальний, регіональний та глобальний рівні моніторингу взаємодіють для забезпечення комплексного контролю та збереження екологічної стійкості.

В роботі було проаналізовано існуючі веб додатки для екологічного моніторингу та порівняв їх. Вони орієнтовані на моніторинг навколишнього середовища. Ничже буде розглянено ці аналоги більш детально, щоб виявити їх особливості та потенційні переваги.

1) Волонтерський проект SaveEcoBot від групи Save Dnipro був створений 4 липня 2018 року та Міністром екології та природних ресурсів Остапом Семераком у прес-центрі Кабінету Міністрів України. Це чат бот з екологічними даними у якого є сайт з веб інтерфейсом у вигляді інтерактивної мапи. SaveEcoBot дає можливість перевіряти інформацію про обсяги викидів та скидів промислових підприємств. Також там можна перевірити, як підприємства-забруднювачі сплачують екоподатки та штрафи.

Бот надасть інформацію і про екологічні перевірки на промислових гігантах. Також він визначає та аналізує концентрацію шкідливих речовин навколо таких підприємств. Бот надсилає автоматичні повідомлення про погіршення якості повітря та надає аналітику за обраний період.

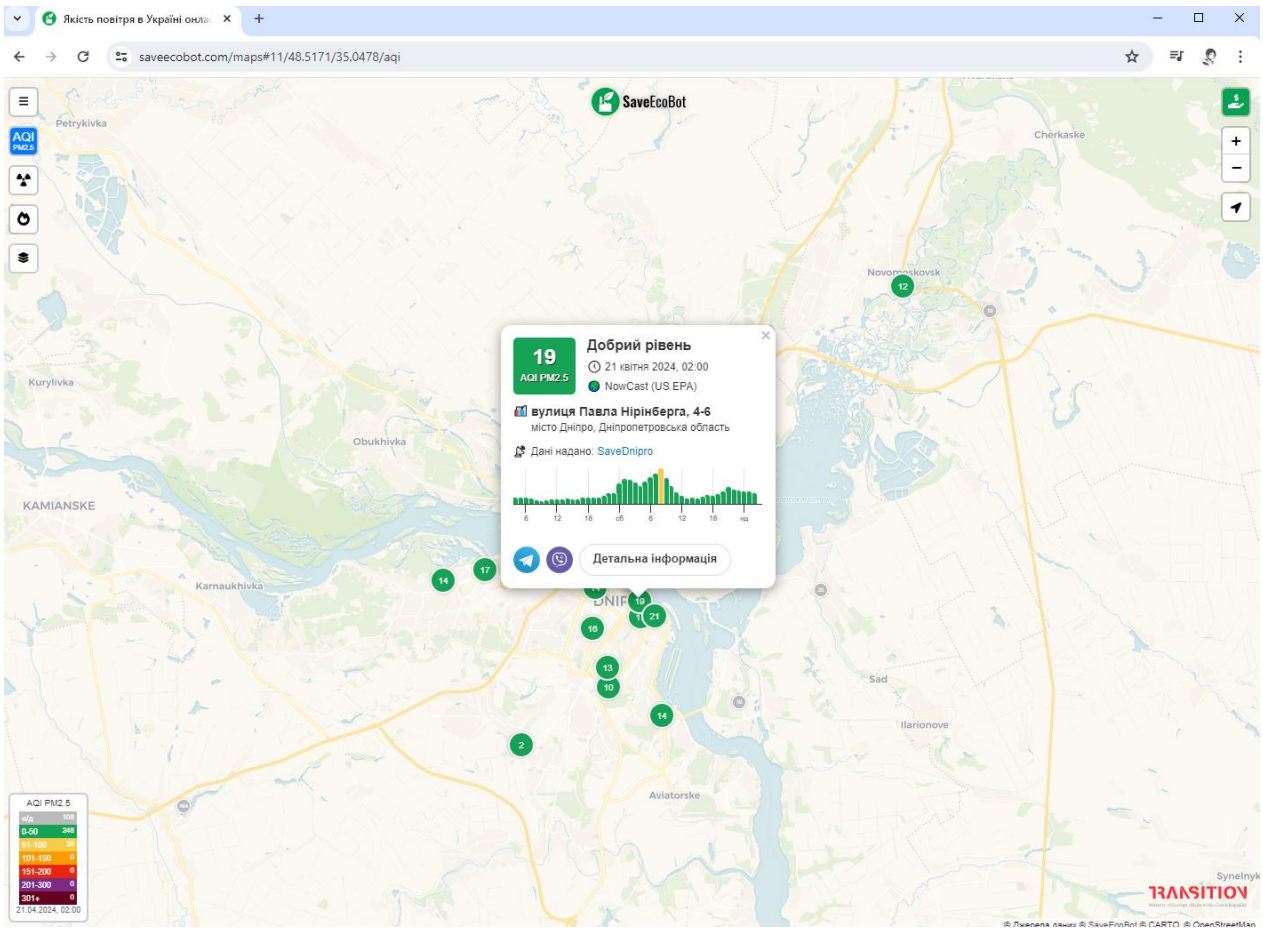

Рис. 1.3. Інтерактивна мапа для екомоніторингу

Як ми бачимо (див. рис. 1.3.) SaveEcoBot може надавати інформацію користувачу у вигляді інтерактивної мапи. На цій мапі можна визначити індекс якості повітря в області, також визначити радіаційний фон, є ще карта пожеж та напрямку вітру, і карта тимчасово окупованих територій. Перехід між мапами зроблений у вигляді кнопок інтерфейсу у лівому верхньому куті. Якщо користувач натисне на кнопку з трьома рисками то користувач побаче плаваюче вікно в якому користувач може дізнится дуже багато корисної інформації, а саме платформи, населені, пункти, аналітика та арі. Як бачимо (див. рис. 1.3.) є інформація про вулицю Павла Нірнберга, 4-6, цю інформацю подано у вигляді діаграми в якій різним кольором позначається рівень забруднення повітря. Користувач може дізнатися про рівень подивившись на таблицю яка знаходиться у лівому нижньому куті.

2) ЛУН Місто Air [3]

Розробниками цього веб додатку є українська ІТ-компанія ЛУН, яка була заснована у 2008 році. Проект ЛУН Місто, започаткований українською ІТ-компанією ЛУН на початку повномасштабної війни, включав серію інтерактивних карт такі як:

ЛУН статистика

Моніторинг якості повітря ЛУН Місто AIR

Мапа Місто без меж

Мапа інтернету під час знеструмлень спільно з Міністерством цифрової інформації

Проєкт по відбудові з United24 та ЛУН

Карта ветеранського бізнесу спільно з Українським ветеранським фондом та Uklon

Чеклист Місто без меж

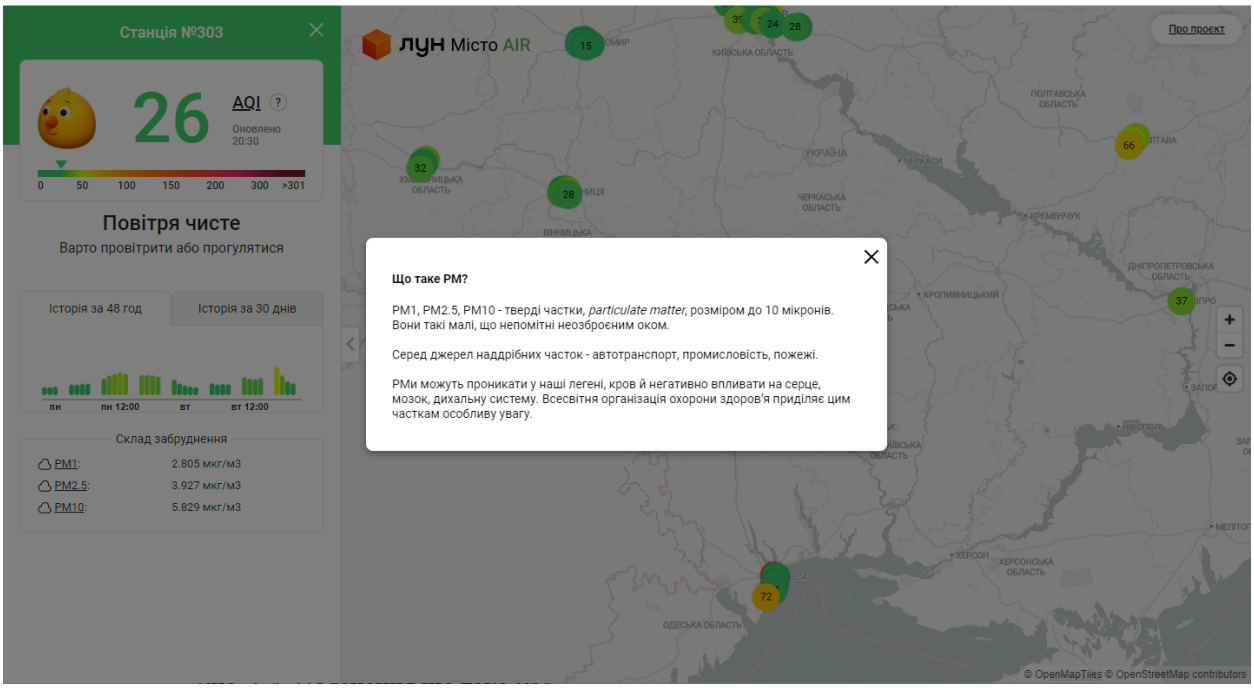

Рис. 1.4. Пояснення що таке PM

Цей додаток надає інформацію тільки про якість повітря, в ньому сподобалося те що є історя за 30 днів, а не тільки за 2 дні як у SaveEcoBot. У додатку Лун Місто Air перевага в більш інтуітивному інтерфейсі та в складі

забруднення. Ще цей додаток зберігають дані, навіть якщо пропадає інтернет. В цьому додатку можна визначити: вологість повітря, температуру та забрудненість атмосфери дрібними часточками PM 1, PM 2,5, PM 10. В ньому навіть є пояснення коли користувач клікає на тверді частки (див. Рис. 1.4.) (PM1, PM2.5, PM10).

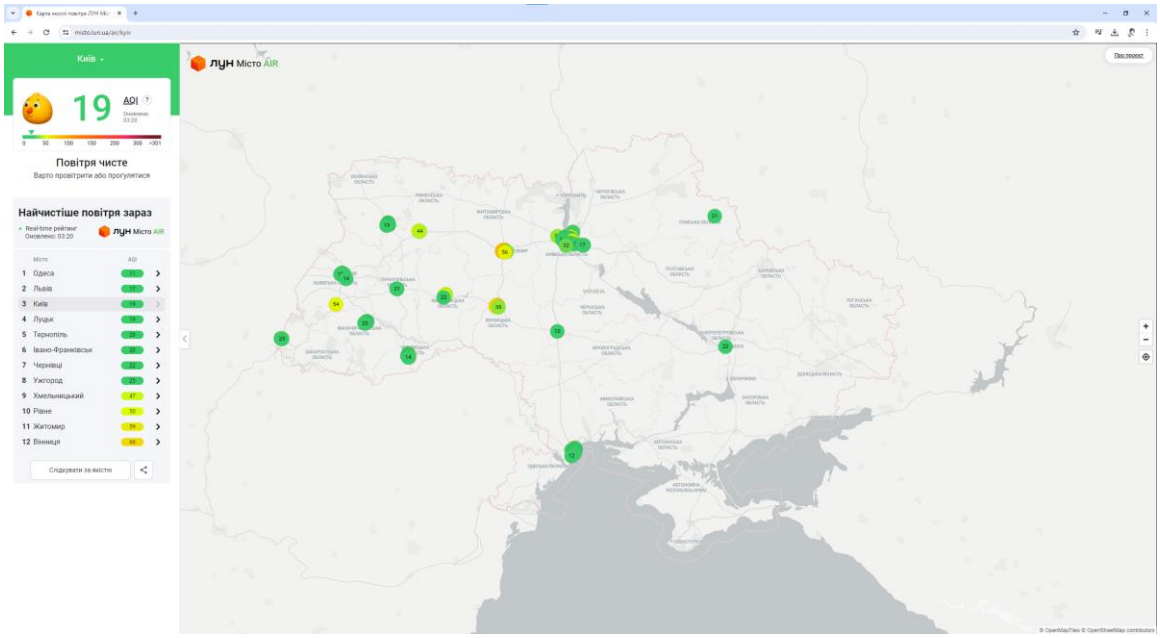

Рис. 1.5. Інтерфейс веб додатку Лун Місто Air

Бачимо (див. рис. 1.5.) що у правому верхньому куті є кнопка про проект яка перенаправляє на інформацію про проект. Ще у правому куті бачимо що є кнопки +- для зміни маштабу та геолокація. Зручно що користувач може одразу дізнатися в яких містах найчистіше повітря на даний момент часу. Ще цікавим рішенням у цьому додатку є те що ти можеш вибрати місто якщо користувач натисне на кнопку з випадаючим меню у лівому верхньому куті та вибере місто яке йому потрібно. Мінусом цього додатку як на мене якщо порівнювати з минулим є те що у нього не багато міст, цей додаток виглядає не дороблением.

3) Open Access — “Відкритий доступ” [4]

Проект “Відкритий доступ” дає користувачам можливість користуватися трьома мапами, а саме Вода, Повітря та ЕкоФінанси (але працює тільки Вода):

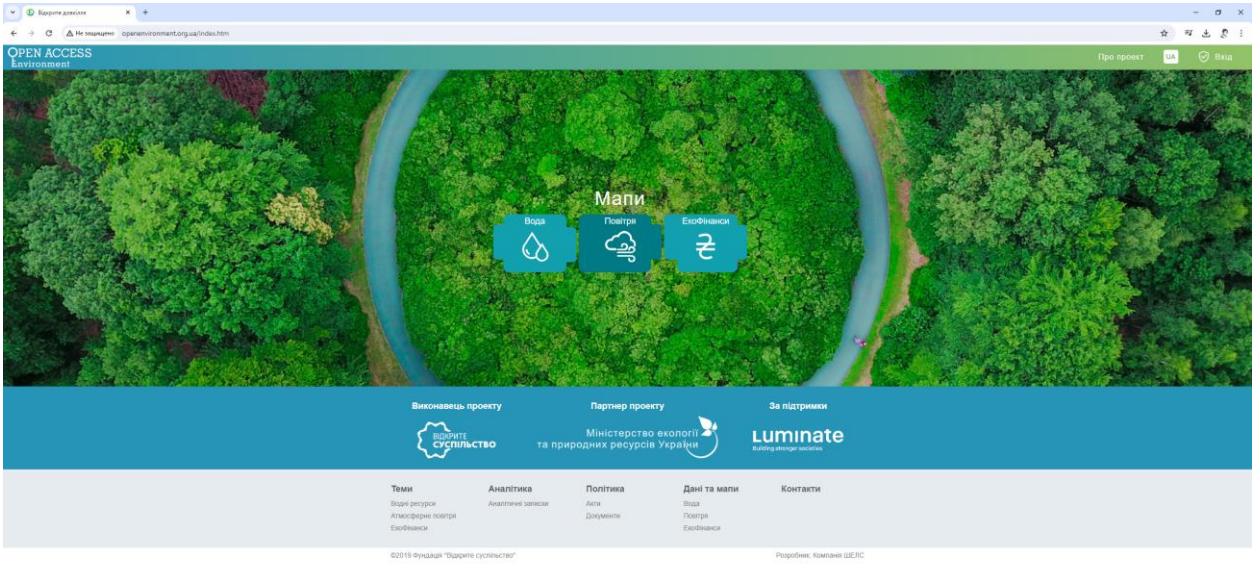

Рис. 1.6. Вибір мапи у веб додатку Open Access — “Відкритий доступ”

На рис (див. рис. 1.6.) показаний вибір з трьох мап: Вода, повітря, Екофінанси, а на шапці сайту бачимо чотири кнопки, а саме вхід, про проект, зміна мови та перехід до головної сторінки якщо нажати на логотип. Користувач одразу може дізнатися про мету цього проекту, що є не поганим рішенням. Якщо вибрати мапу про Воду то користувача перенаправляє до розділу води.

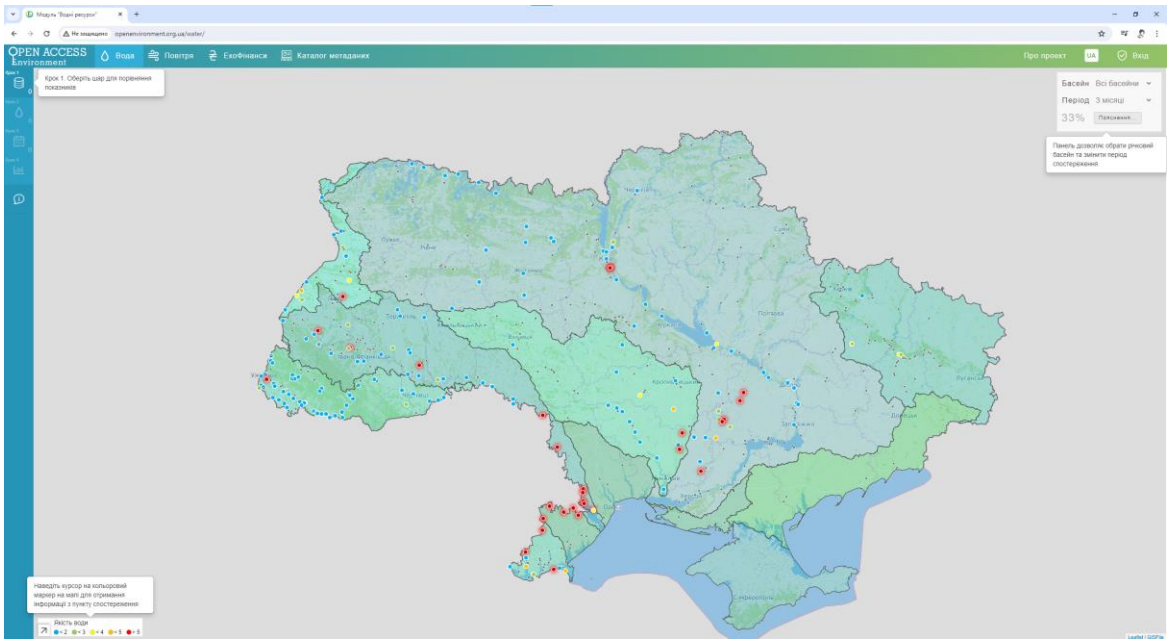

Рис. 1.7. Інтерактивна карта України

Як ми бачимо (див. рис. 1.7.) зразу є багато різних підсказок, а саме що є різні кольорові маркери якості води (синій, зелений, жовтий, помаранчевий, червоний), допомога користувачеві з користуванням панелі навігації по крокам (зліва в горі) та що є панель яка дозволяє обрати річковий басейн та змінити період спостереження.

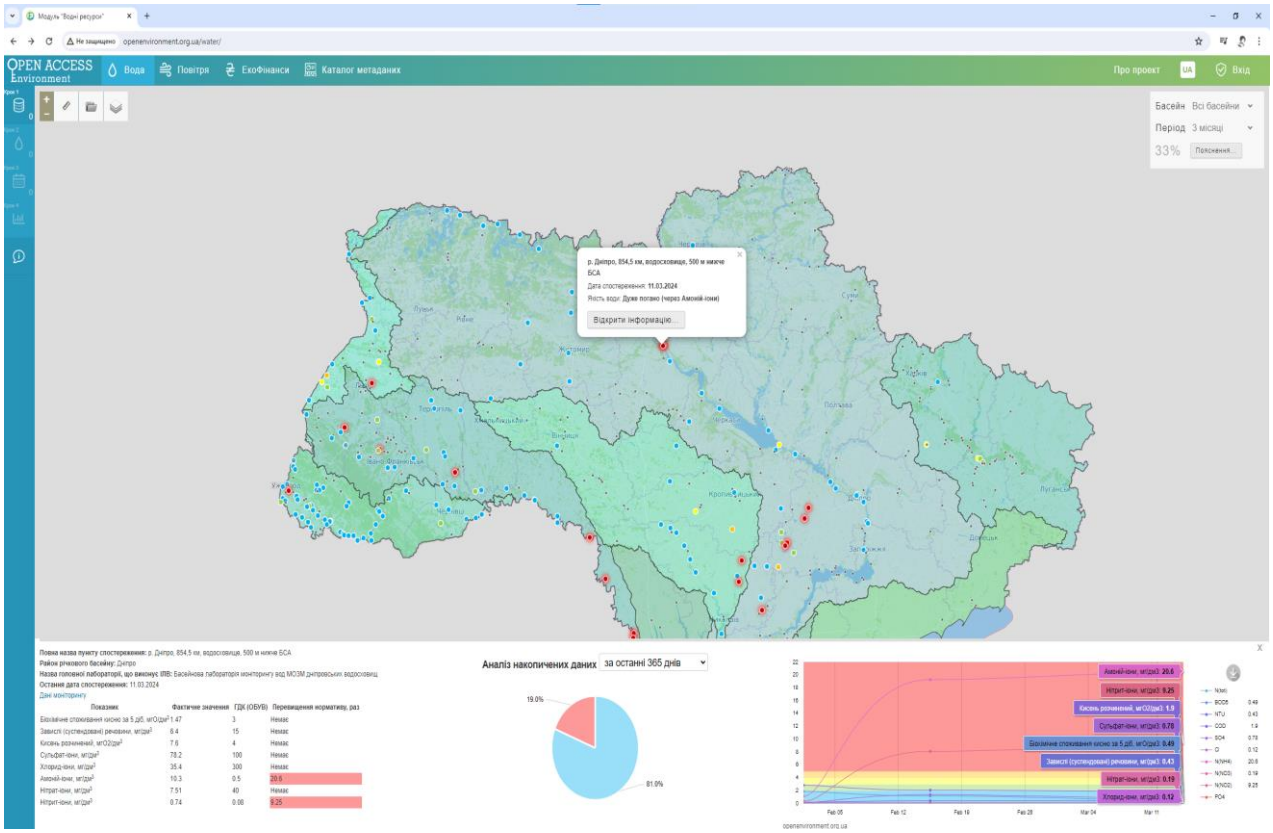

Рис. 1.8. Інтерфейс веб додатку Open Access — “Відкритий доступ”

Якщо нажати на маркер (див. рис 1.8.), а потім нажати на кнопку «відкрити інформацію», після цього виведуться дані про водосховище яке знаходиться у Районі басейну річки Дніпро. Є діаграма аналізу накопичених даних да графік з таблицею. Та додаткова інформація про цей басейн.

Давайте відсортуємо по крокам річки Дніпро, Дністер, Дон, Дунай на наявність Азоту за 2014-2018 рік.

Бачимо що в усіх трьох річках Азот був у допустимій нормі рис (див. рис. 1.9.)

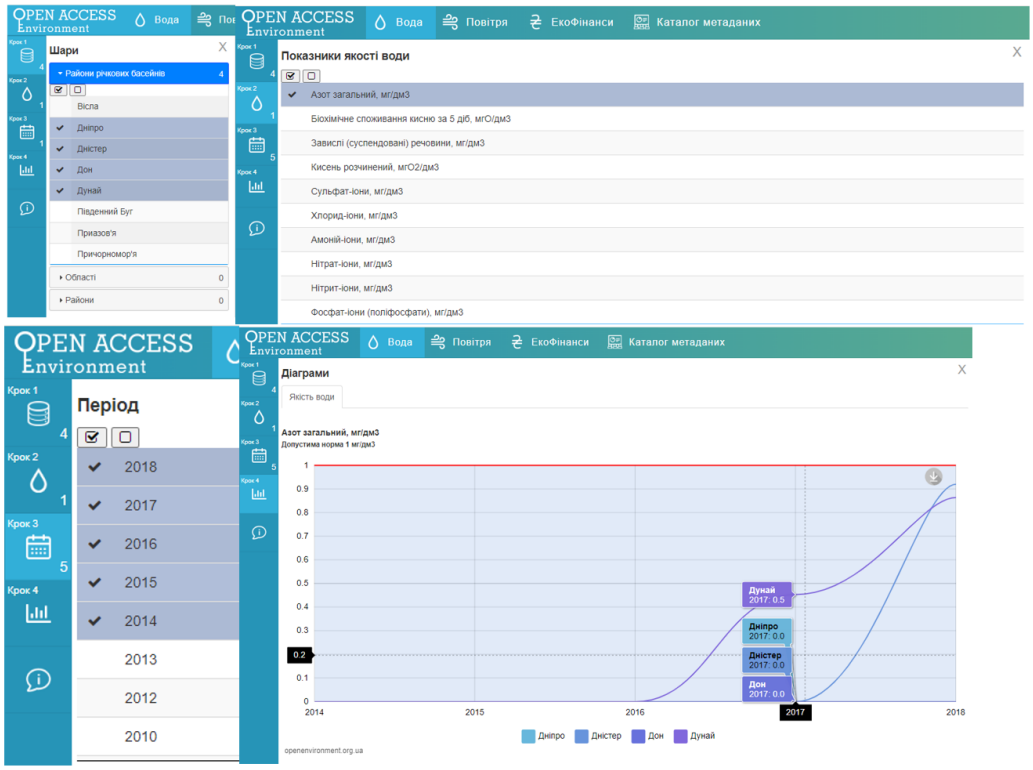

Рис. 1.9. Відстортування річок по якості води

Рис. 1.10. Кнопка входу

Ще у цьому веб-додатку є кнопка входу до системи (у верхньому правому куті (див. рис. 1.6).

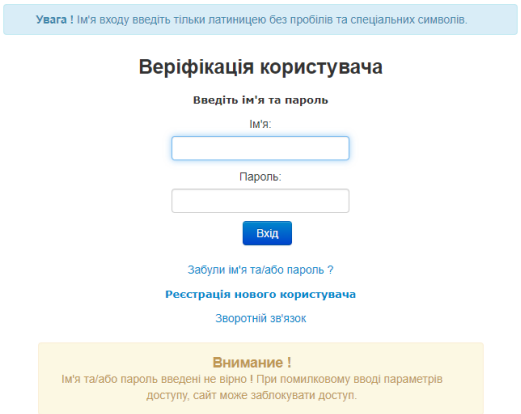

Рис. 1.11. Вікно веріфікації користувача

Діалогове віконце над веріфікацією користувача про попередження (див рис. 1.11) що потрібно вводити тільки латиницею без пробілів та спец. символів. Якщо пароль або ім’я буде введено не вірно буде виведено діалогове вікно про попередження заблокування доступу при помилковому вводі параметрів.

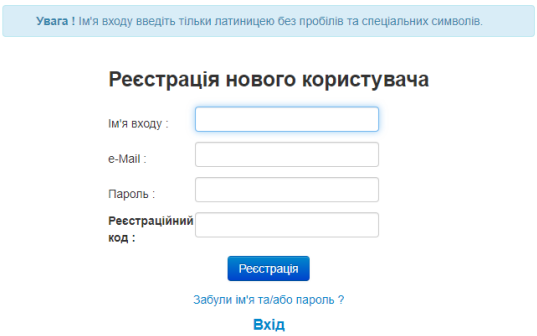

Рис. 1.12. Вікно реєстрації нового користувача

Звісно якщо є вікно веріфікації користувача є і вікно реєстрації нового користувача, це вікно показане на (див. рис. 1.12.). Також присутня валідація данних на сторінці реєстрації або входу

4) Карта моніторингу якості повітря Eco City

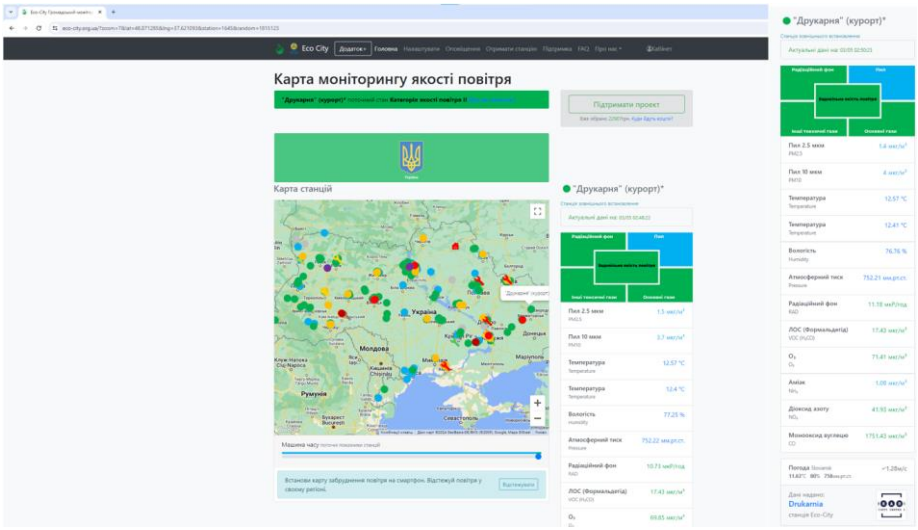

Рис. 1.13. Інтерфейс веб додатку eco-city.org

Так виглядає інтерфейс веб додатку еко-сіті (див. рис. 1.13.). Подобається в якому вигляді зроблений вивід інформації про загальну якість повітря, пил, радіаційний фон, основні та інші токсичні гази. Також в цьому додатку є машина часу за 48 годин. Карту можна розгорнути якщо нажати на кнопку.

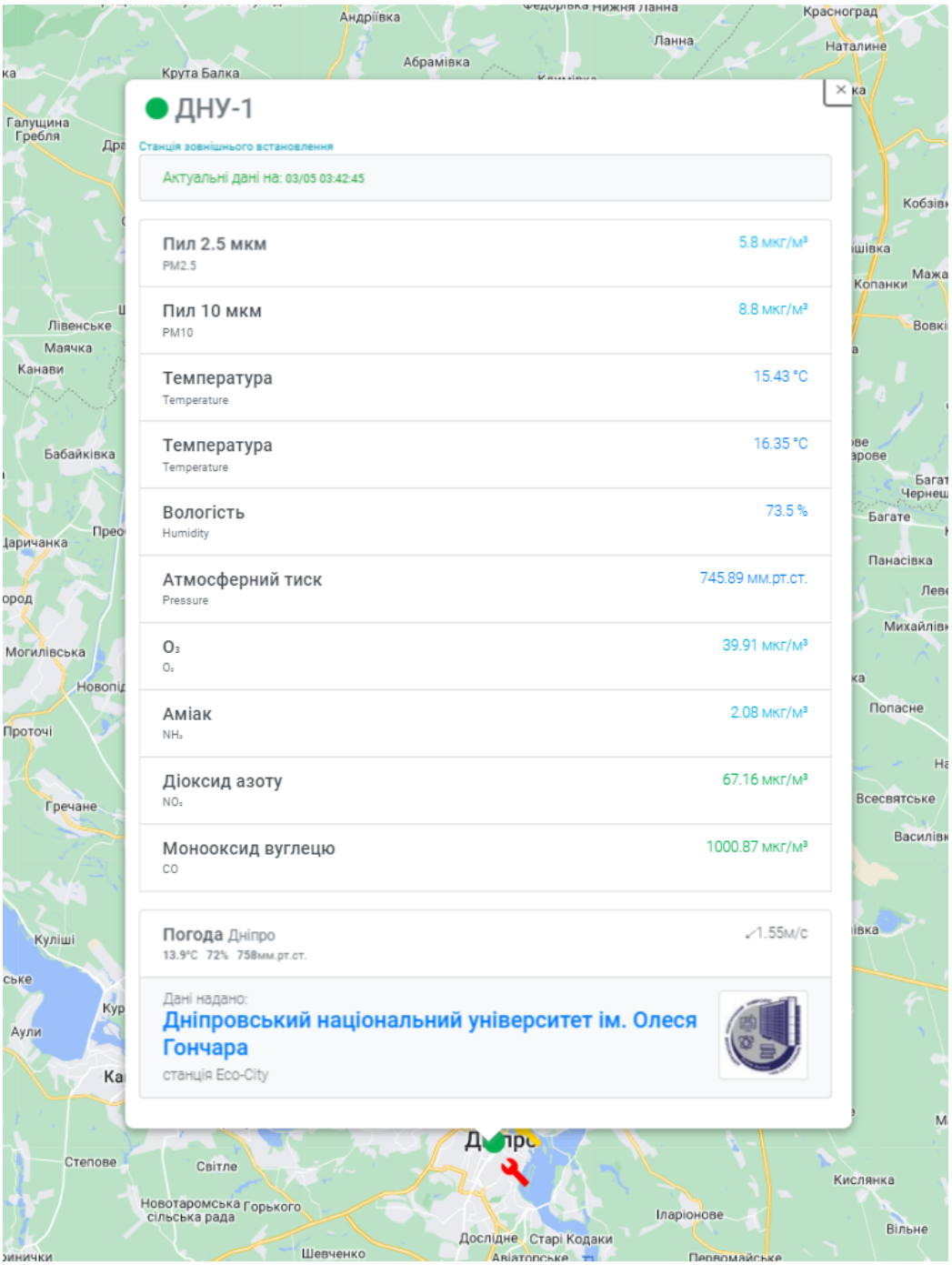

Рис. 1.14. Карта веб додатку eco-city.org

Бачимо що якщо натиснути на маркер буде показане віконце в якому буде показана екоінформація (див. рис. 1.14.)

Після аналізу чотирьох веб-додатків давайте створимо таблицю переваг та обмежень в цих додатках.

Таблиця 1.1

Переваги та обмеження		
Веб-додаток	Перевага	Обмеження
SaveEcoBot	Інтерактивна взаємодія з користувачами у вигляді чатботу, простий і зрозумілий інтерфейс	Складність розробки розумного чатботу
ЛУН Місто Air	Наочна візуалізація даних про якість повітря. Можна визначити забрудненість атмосфери дрібними часточками.	Обмежений лише даними про повітря, та обмежений лише територією міста. Відсутні дані про приміські/сільські райони, точність даних залежить від кількості та розміщення датчиків моніторингу. Дані є про Правобережну Україну і основному і малу частину Лівобержної України

«Продовж. табл. 1.1»

Open Access — “Відкритий доступ”	Користувач може за кроками створити діаграму забруднень. Багато діаграм за різні проміжки часу, це надає більше можливістей дла аналізування та використовування даних для досліджень. Можна увійти у систему.	Працює тільки карта повітря.
Eco City	Зручний доступ до широкого спектру екологічних даних в одному місці. Є машина часу за допомогою якої можна продивлятись зміни на карті за 48 годин. Можна увійти у систему.	Некомпактний інтерфейс.

Якщо узагальнити, додаток Open Access — “Відкритий доступ” виглядає краще ніж інші, бо в нього є цікавий функціонал, за допомогою якого користувач може за кроками створити діаграму забруднень.

Отже, призначенням розробки веб-додатку екологічного моніторингу є надання користувачеві можливості взаємодіяти з системою моніторингу екології через веб-браузер. У додатку користувач зможе вибрати між інтерактивною картою України, за допомогою якої можна дізнатися про різні екологічні дані у містах, також можна буде переглянути інформацію про забруднення, наприклад у діаграмі, та сторінкою, на якій можна буде знайти погоду у містах.

1.2 Призначення розробки та галузь застосування

Програмне забезпечення клієнтської частини веб-застосунку екологічного моніторингу призначене для забезпечення зручного та

ефективного інтерфейсу користувача для систем екологічного моніторингу.

Ключові терміни:

Екологічний моніторинг: система спостережень за станом навколишнього середовища.

Веб-застосунок: програмне забезпечення, що працює у веб-браузері.

Клієнтська частина: компонент програмного забезпечення, що виконується на пристрої користувача.

Необхідність розробки даного програмного забезпечення обумовлена зростаючою потребою в ефективних інструментах для збору, аналізу та візуалізації екологічних даних. Сучасні екологічні виклики вимагають створення доступних та зручних засобів моніторингу стану довкілля, які можуть використовуватися як фахівцями, так і широкою громадськістю.

Галузі застосування веб-застосунку включають:

1. Державні та приватні установи, що здійснюють екологічний моніторинг.
2. Наукові та освітні заклади, що проводять дослідження в галузі екології.
3. Громадські організації, що займаються питаннями охорони навколишнього середовища.
4. Промислові підприємства для контролю впливу на довкілля.
5. Органи місцевого самоврядування для моніторингу екологічної ситуації в регіонах.
6. Громадяни, зацікавлені в отриманні актуальної інформації про стан навколишнього середовища.

Веб-застосунок дозволить підвищити ефективність збору та аналізу екологічних даних, сприятиме прийняттю обґрунтованих рішень щодо охорони довкілля та підвищить рівень екологічної обізнаності суспільства.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему «Розробка програмного забезпечення клієнтської частини веб застосунку екологічного моніторингу» є наказ по Національному технічному університету «Дніпровська політехніка» №469-с від 23.05.2024.

1.4. Постановка завдання

Ціллю даного веб додатку екологічного моніторингу є надання користувачеві взаємодіяти з системою моніторингу екології через веб-браузер. Діаграми та картографічні представлення даних можуть допомогти користувачам краще розуміти стан навколишнього середовища в їхніх регіонах. Вони будуть використані для відображення рівня забруднення повітря, водних джерел, відходів та інших екологічних параметрів.

Для зберігання даних про користувачів буде використанна база даних. Користувач зможе увійти до системи, в якій йому буде надана інформація про різні екологічні параметри. У користувача буде вибір між інтерактивною картою якості повітря України або сторінка в якій можна буде знайти місто та дізнатися про різні екологічні дані в діаграмах. Інтерфейс повинен буде простим щоб користувачу було легко орієнтуватися.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Веб-застосунок екологічного моніторингу повинен забезпечувати наступні функціональні можливості:

Механізм авторизації та реєстрації нових користувачів. Для реалізації цієї функціональності будемо використовувати Firebase Authentication та Firebase Realtime Database для зберігання даних користувачів. Це дозволить створити надійну систему авторизації без необхідності налаштування локального сервера.

Перехід між сторінками буде реалізовано за допомогою натискання кнопок, а саме кнопка інтерактивної карти України про якість повітря або пошуку різних екологічних даних які будуть виведені в діаграмах та таблицях.

Інтерактивні діаграми для відображення екологічних показників, наприклад забруднення повітря. Для цього можна використати бібліотеку Chart.js напрклад.

Збір даних з відкритих екологічних API, наприклад, OpenWeatherMap для даних про якість повітря, та екологічних забруднень.

Надання користувачу можливості ознайомитися з застосунком у вигляді тексту який може бути показаний у вигляді діалогового вікна.

1.5.2. Вимоги до інформаційної безпеки

Поки на етапі прототипу не планується розміщувати на сервері веб-додаток, вимоги до інформаційної безпеки можуть бути менш актуальними. Оскільки додаток ще знаходиться в розробці та тестуванні, обсяг конфіденційної інформації може бути обмеженим або відсутнім, а кількість користувачів, які мають доступ до системи, може бути обмеженою.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для ефективного функціонування веб-застосунку екологічного моніторингу необхідно забезпечити наявність таких технічних засобів:

1. Пристрої введення**:**
    - Клавіатура**:** стандартна клавіатура для ПК або ноутбука.

- Миша або тачпад**:** для зручності навігації.

2. Комп'ютерні пристрої**:**

   Персональний комп'ютер (ПК) або ноутбук**.**

3. Вимоги до характеристик:
   - Процесор: будь-який багатоядерний процесор.
   - Оперативна пам'ять: мінімум 2 ГБ.
   - Дисковий простір: мінімум 500 МБ вільного місця.
   - Операційна система: Windows 7 і новіші, macOS, Linux.

Обов’язково мати встановлений інтернет-браузер.

1.5.4. Вимоги до інформаційної та програмної сумісності

Веб-застосунок повинен бути сумісним з наступними сучасними браузерами: Opera, Google Chrome, Microsoft Edge, Safari, Firefox.

Запуск веб-застосунок буде здійснюватися локально через фреймворк Quasar, який забезпечить належне середовище для розробки та тестування.

Сумісність з відкритими API для отримання екологічних даних та API для авторизації.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Після аналізу функціоналу існуючих веб-додатків для екологічного моніторингу, можна сформулювати план функціонального призначення цього додатка.

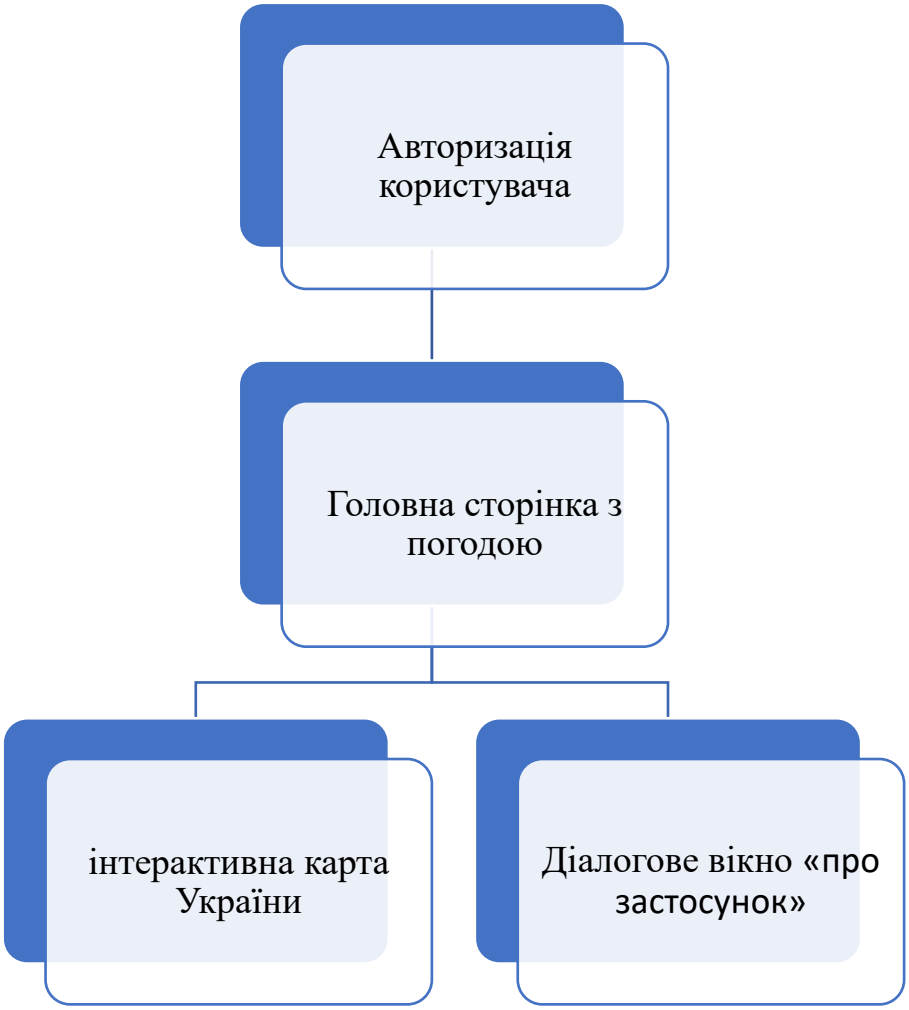

Рис. 2.1. Схема послідовності дій в веб-застосунку для користувача

З рисунку 2.1. видно послідовність дій користувача: спочатку він авторизується, після чого буде перенаправлений на головну сторінку. Тут він може знайти погоду в місті яке його цікавить (за геолокацією або введеним

запитом), а також перейти до сторінки з поясненнями про додаток через діалогове вікно. Користувач також може вийти з системи, натиснувши відповідну кнопку.

При переході з головної сторінки на сторінку з інтерактивною картою користувач побачить маркери на карті, за допомогою яких він може дізнатись про рівень забруднення та погоду у місті. Ця інформація буде виводитись у діалоговому вікні. Діаграма забруднення відображатиметься також у вигляді діалогового вікна після натискання відповідної кнопки.

2.2 Опис застосованих математичних методів

У цій роботі не використовується складних математичних методів. Веб-застосунок використовує основні арифметичні операції для обробки даних і відображення результатів.

2.3 Опис використаної архітектури та шаблонів проектування

Цей проект використовує клієнт-серверну архітектуру, де клієнтська частина взаємодіє з серверною через API. Користувацький інтерфейс реалізовано за допомогою JavaScript-фреймворку Quasar, який базується на Vue.js і забезпечує ефективну розробку завдяки великій кількості готових компонентів та утиліт. Серверна частина реалізована за допомогою Firebase, який виступає як сервіс для аутентифікації користувачів, зберігання даних та обміну повідомленнями в реальному часі.

Якщо казати про шаблони проектування то їх є декілька, а саме:

1) Шаблон проектування "Модальний вікно" (Modal Window Pattern): Використовується для відображення діаграми якості повітря у модальному вікні (див. рис. 2.2.). Це робиться за допомогою div з класом "modal", який відкривається та закривається, коли користувач натискає відповідні кнопки.

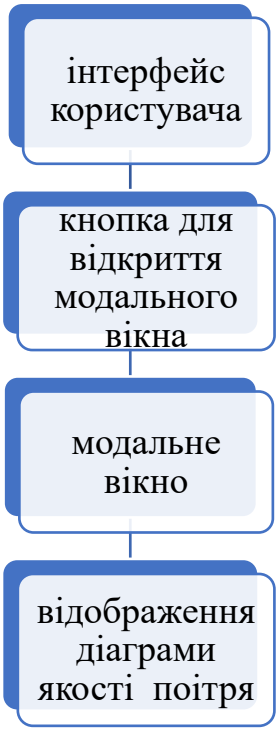

Рис. 2.2. Шаблон проектування "Модальний вікно"

2) Шаблон проектування "Компонент" (Component Pattern):

Код розділений на компоненти, такі як LeafletMap та WeatherApp, що полегшує підтримку та повторне використання коду. Кожен компонент відповідає за свій власний функціонал і може бути легко інтегрований у більші системи.

3) Шаблон проектування "Синглтон" (Singleton Pattern):

Шаблон "Синглтон" можна побачити у використанні карти Leaflet. Ініціалізація карти відбувається лише один раз, і ця карта використовується протягом усього життєвого циклу компонента.

4) Шаблон проектування "Фасад" (Facade Pattern):

Методи getWeatherInfo та getAirQualityInfo приховують складність взаємодії з API і забезпечують простий інтерфейс для отримання погодних даних та даних про якість повітря.

5) Шаблон проектування "Спостерігач" (Observer Pattern):

Використовується для оновлення даних на карті. Коли користувач натискає на маркер міста, методи getWeatherInfo та getAirQualityInfo викликаються для отримання нових даних, і карта оновлюється відповідно.

6) Component-based architecture (Архітектура на основі компонентів):

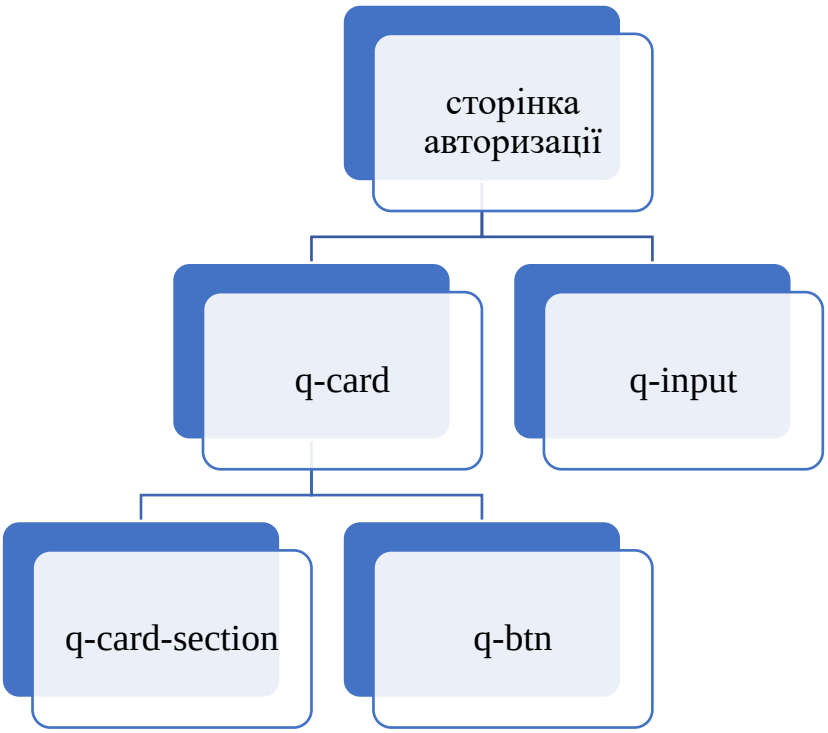

Рис. 2.3. Шаблон архітектури на основі компонентів

Сторінка login-page розділена на компоненти (див. рис. 2.3) q-card, q-card-section, q-form, q-input, та q-btn [13]. Кожен компонент відповідає за конкретний фрагмент інтерфейсу, що сприяє модульності і простоті управління.

2.4 Опис використаних технологій та мов програмування:

У розробці веб-додатку для відображення погоди та якості повітря було використано комплекс сучасних технологій, центральним з яких став Quasar Framework, що базується на Vue.js 3.

Перший етап взаємодії користувача з додатком - це авторизація. Для реалізації системи аутентифікації було використано Firebase Authentication. Форми входу та реєстрації створені за допомогою компонентів Quasar, таких як q-input та q-btn. Для обробки помилок авторизації було реалізовано систему перевірок з використанням try-catch блоків у асинхронних функціях. Повідомлення про помилки відображаються за допомогою компонента Notify з Quasar Framework, що дозволяє інформувати користувача про неправильний ввід даних або інші проблеми при авторизації.

Після успішної авторизації користувач потрапляє на головну сторінку додатку. Тут реалізовано кілька ключових функцій:

Пошук міста: використовується q-input компонент Quasar та асинхронні запити до OpenWeatherMap API за допомогою Axios.

Геолокація: реалізована за допомогою HTML5 Geolocation API. При натисканні на відповідну кнопку, додаток запитує дозвіл на отримання місцезнаходження користувача.

Кнопка "Про застосунок": відкриває модальне вікно з інформацією, реалізоване за допомогою q-dialog компонента Quasar.

Кнопка переходу до інтерактивної карти: реалізована з використанням Vue Router для навігації між компонентами додатку.

Для відображення погодних даних на головній сторінці використовуються асинхронні запити до OpenWeatherMap API. Отримані дані обробляються за допомогою JavaScript [23] і відображаються у відповідних блоках сторінки. Прогноз погоди на 4 дні реалізовано за допомогою окремого запиту до API прогнозу погоди, результати якого фільтруються та відображаються у вигляді окремих блоків.

Інтерактивна карта, доступна через окремий маршрут, розроблена з використанням бібліотеки Leaflet.js. На карті розміщено маркери основних міст України. При натисканні на маркер відбувається наступне:

Виконується асинхронний запит до OpenWeatherMap API для отримання актуальних даних про погоду та якість повітря для вибраного міста.

Отримані дані відображаються у спливаючому вікні (popup) Leaflet, яке містить інформацію про температуру, вологість, тиск, швидкість вітру та якість повітря.

Додатково, користувач може натиснути кнопку "Показати графік", що викликає появу модального вікна з круговою діаграмою якості повітря, створеною за допомогою бібліотеки Chart.js.

Для створення адаптивного дизайну та стилізації компонентів використовувались можливості Quasar Framework у поєднанні з кастомними стилями CSS. Це забезпечило коректне відображення додатку на різних пристроях.

У процесі розробки було проведено тестування всіх ключових функцій додатку:

Перевірка коректності роботи авторизації та обробки помилок при вході/реєстрації.

Тестування точності геолокації та коректності відображення погодних даних для визначеного місцезнаходження.

Перевірка працездатності пошуку міст та відображення відповідних погодних даних.

Тестування інтерактивної карти, включаючи коректність розміщення маркерів та відображення інформації при натисканні на них.

Перевірка коректності відображення графіків якості повітря та прогнозу погоди на 4 дні.

Таким чином, комбінація цих технологій дозволила створити повнофункціональний веб-додаток, який надає користувачам зручний та інтерактивний доступ до актуальної інформації про погоду та якість повітря в містах України.

2.5 Опис структури програми та алгоритмів її функціонування

1) Процес реєстрації та входу користувача.

Рис. 2.4. Схема процесів реєстрації та входу користувача

Ця схема показує процеси реєстрації та входу користувача (див. рис. 2.4.), включаючи взаємодію з Firebase Auth, маршрутизацію за допомогою Vue Router, а також збереження даних у LocalStorage. Вона демонструє, як інтерфейс користувача взаємодіє з різними компонентами системи для забезпечення функціональності аутентифікації.

Реєстрація користувача:

Користувач заповнює форму реєстрації та надсилає запит на реєстрацію.

Запит на реєстрацію надсилається до Firebase Auth.

Firebase Auth відповідає, вказуючи на успішність або невдачу реєстрації.

Якщо реєстрація успішна, користувача перенаправляють на /app.

Якщо реєстрація не вдалася, на інтерфейсі користувача відображається повідомлення про помилку.

Вхід користувача:

Користувач заповнює форму входу та надсилає запит на вхід.

Запит на вхід надсилається до Firebase Auth.

Firebase Auth відповідає, вказуючи на успішність або невдачу входу.

Якщо вхід успішний, користувача перенаправляють на /app і дані користувача зберігаються у LocalStorage.

Якщо вхід не вдався, на інтерфейсі користувача відображається повідомлення про помилку.

Таким чином, цей застосунок використовує Firebase Auth для управління аутентифікацією, Vue Router для маршрутизації між сторінками, та LocalStorage для збереження даних користувача. Взаємодія цих компонентів забезпечує плавний процес реєстрації та входу для користувача, забезпечуючи надійну аутентифікацію та збереження стану сеансу.

2) Структура сторінки для показу погоди та інтерактиної карти

Програма має три основні компоненти: інтерфейс користувача (UI), API, і внутрішні компоненти (див. рис. 2.4.). Інтерфейс користувача включає головну сторінку з полем для введення назви міста і кнопкою "Пошук", сторінку з картою, відображеною за допомогою Leaflet.js, попапи з інформацією про погоду та якість повітря, і модальні вікна для діаграм якості повітря. Для отримання даних використовується OpenWeatherMap API, який надає інформацію про погоду та якість повітря. Внутрішні компоненти включають компонент для пошуку міста, компонент для відображення карти з маркерами, компонент для попапів з інформацією про погоду і компонент для модальних вікон з діаграмами якості повітря.

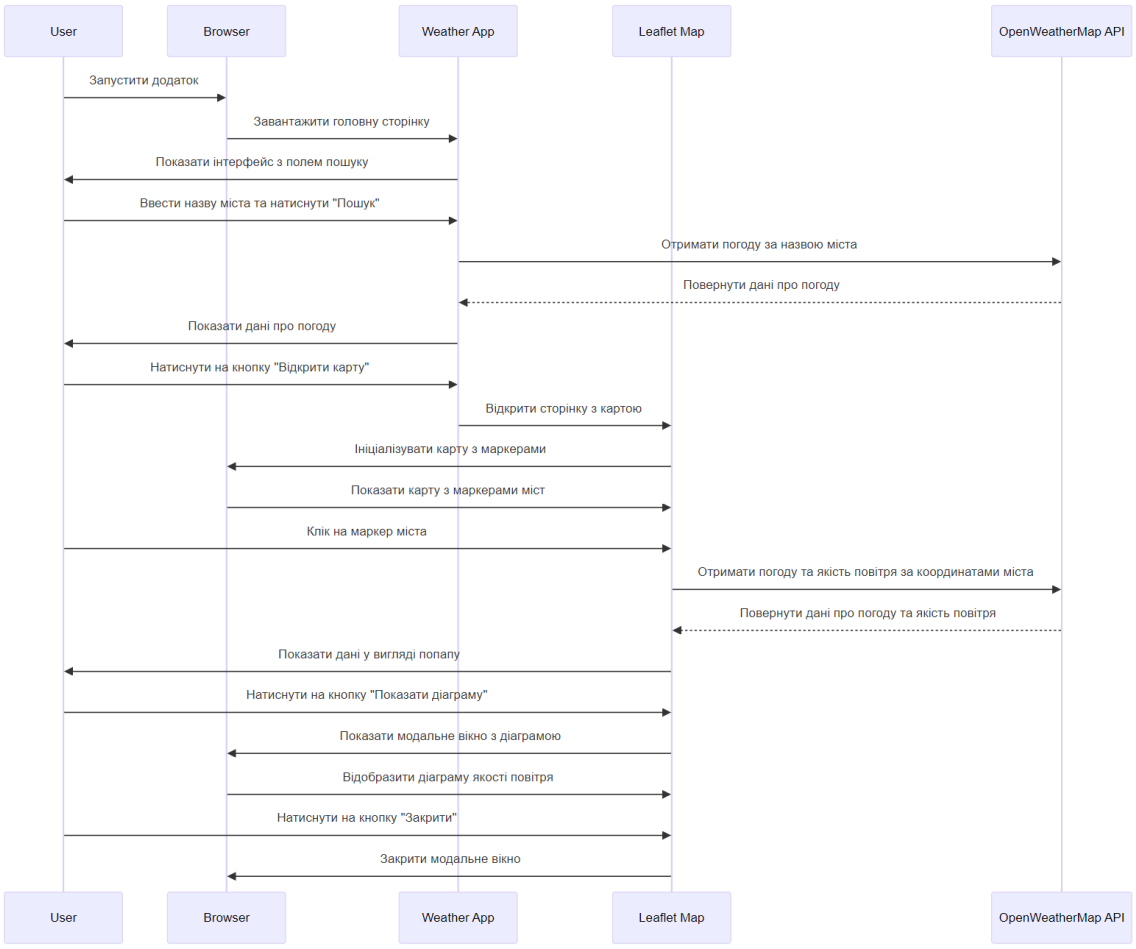

Рис. 2.4. Шаблон архітектури на основі компонентів

Коли користувач запускає додаток у браузері, завантажується головна сторінка з полем для введення назви міста і кнопкою "Пошук". Після введення назви міста і натискання кнопки "Пошук", додаток надсилає запит до OpenWeatherMap API для отримання даних про погоду за введеною назвою міста. Після отримання даних про погоду, додаток відображає їх користувачу. Коли користувач натискає на кнопку "Відкрити карту", завантажується сторінка з картою, ініціалізована за допомогою Leaflet.js, і карта відображається у браузері з маркерами міст.

Якщо користувач клікає на маркер міста на карті, додаток надсилає запит до OpenWeatherMap API для отримання даних про погоду та якість повітря за координатами обраного міста. Після отримання даних, додаток відображає їх у вигляді попапу на карті. Коли користувач натискає на кнопку "Показати діаграму", додаток відкриває модальне вікно з діаграмою якості повітря, яка

відображає відповідну інформацію. Якщо користувач натискає на кнопку "Закрити" у модальному вікні, додаток закриває вікно і повертається до відображення карти.

Для підвищення продуктивності програми передбачено кешування даних про погоду та якість повітря, що дозволяє уникнути зайвих запитів до API. Також забезпечено адаптивний дизайн для зручного використання додатка на різних пристроях та валідацію введення користувача для запобігання некоректним запитам.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні та вихідні дані відіграють ключову роль у функціонуванні програмного забезпечення. Їх ефективне управління та обробка є основою для створення надійного та зручного у використанні додатку. Розглянемо структуру вхідних та вихідних даних.

Почнемо з вхідних даних:

Пошуковий запит користувача: Користувач може ввести назву міста в поле пошуку, щоб отримати інформацію про погоду та якість повітря для цього місця.

Координати місцезнаходження користувача: Застосунок може отримати координати місцезнаходження користувача через браузерний API Geolocation, щоб відобразити погоду та якість повітря за цими координатами.

Дані з зовнішніх API: Програма отримує дані про погоду та якість повітря з зовнішніх API, таких як OpenWeatherMap API. Ці дані є вхідними даними для програми.

Введені дані користувача для реєстрації та авторизації: Користувач вводить свої облікові дані (ім'я, прізвище, електронну адресу та пароль) для реєстрації або авторизації в системі.

Вихідні дані:

Відображення поточної погоди: Програма відображає поточну інформацію про погоду для обраного міста або місцезнаходження користувача, включаючи температуру, опис погоди, вологість, тиск, швидкість та напрямок вітру тощо.

Прогноз погоди на кілька днів: Додаток показує прогноз погоди на найближчі кілька днів для обраного міста або місцезнаходження користувача.

Інформація про якість повітря: Програма відображає дані про якість повітря для обраного міста або місцезнаходження користувача, включаючи індекс якості повітря (AQI), рівні різних забруднюючих речовин (PM2.5, PM10, озон, діоксид азоту, діоксид сірки, монооксид вуглецю).

Діаграма якості повітря: Користувач може переглянути діаграму, що візуалізує рівні різних забруднюючих речовин для обраного міста.

Повідомлення про помилки та успішні дії: Програма відображає повідомлення про помилки (наприклад, неможливість отримати дані з API) або про успішні дії (наприклад, успішну реєстрацію або авторизацію).

Організація вхідних та вихідних даних здійснюється за допомогою відповідних структур даних (об'єктів, масивів) та методів обробки даних у коді. Наприклад, дані про погоду та якість повітря зберігаються в об'єктах, які потім використовуються для відображення інформації у вигляді тексту або візуалізації на діаграмі [22].

Належна організація вхідних та вихідних даних є важливою для забезпечення зрозумілості, легкості підтримки та можливості масштабування програми в майбутньому.

2.7 Опис розробленого програмного продукту

2.7.1.Використані технічні засоби

Якщо почати з системних вимог, то для цього веб-застосунку вистачить мінімальних ресурсів ПК або ноутбуку. Веб-браузер для цього застосунку:

Google Chrome, Mozilla Firefox, Microsoft Edge або інший сучасний браузер. Обов'язково мати підключення до мережі Internet та мишу або клавіатуру, якщо у вас ноутбук то тачпад.

2.7.2 Використані програмні засоби

Для розробки веб-застосунку використовувались різноманітні програмні засоби. В якості текстового редактора було використано Visual Studio Code.

Для створення цього веб-застосунку була використана мова програмування – JavaScript [9], мова розмітки - HTML, мова стилів – CSS. За домогою цих компонентів було створено сторінку авторизації користувача, де користувач може увійти у систему або авторизуватися, сторінку з погодою та картою.

Для роботи з картами використовувалася бібліотека Leaflet.

Для реалізації функції аутентифікації та роботи з базою даних було використано хмарну платформу Firebase від Google. Firebase забезпечує зручні інструменти для аутентифікації користувачів, зберігання даних в реальному часі та хостингу веб-застосунків.

Під час розробки також використовувалися інші допоміжні бібліотеки та інструменти, такі як Chart.js для відображення діаграм [15], а також інструменти для роботи з API для отримання даних про погоду та якість повітря.

2.7.3 Виклик та завантаження програми

Щоб завантажити програму потрібно спочатку встановити Quasar CLI, для цього у command prompt потрібно написати команду npm install -g @quasar/cli. Після цього потрібно створити перший проект, а саме за допомого команди quasar create quasar-project(наприклад).

Для виклику веб-застосунку потрібно запустити command prompt, а потім перейти у деректорію з веб-застосунком за допомогою команди cd та переходимо до створенної нами папки з проектом quasar-project, після цього пишемо команду quasar dev, яка ініціалізує запуск локального серверу на порту 9000.

2.7.4 Опис інтерфейсу користувача

Після запуску локального серверу відкривається вікно для авторизації (див. рис. 2.1). За замовчуванням відображається форма входу, але якщо користувачеві потрібно зареєструватися, він може натиснути на посилання "Зареєструватися" і перейти до сторінки реєстрації.

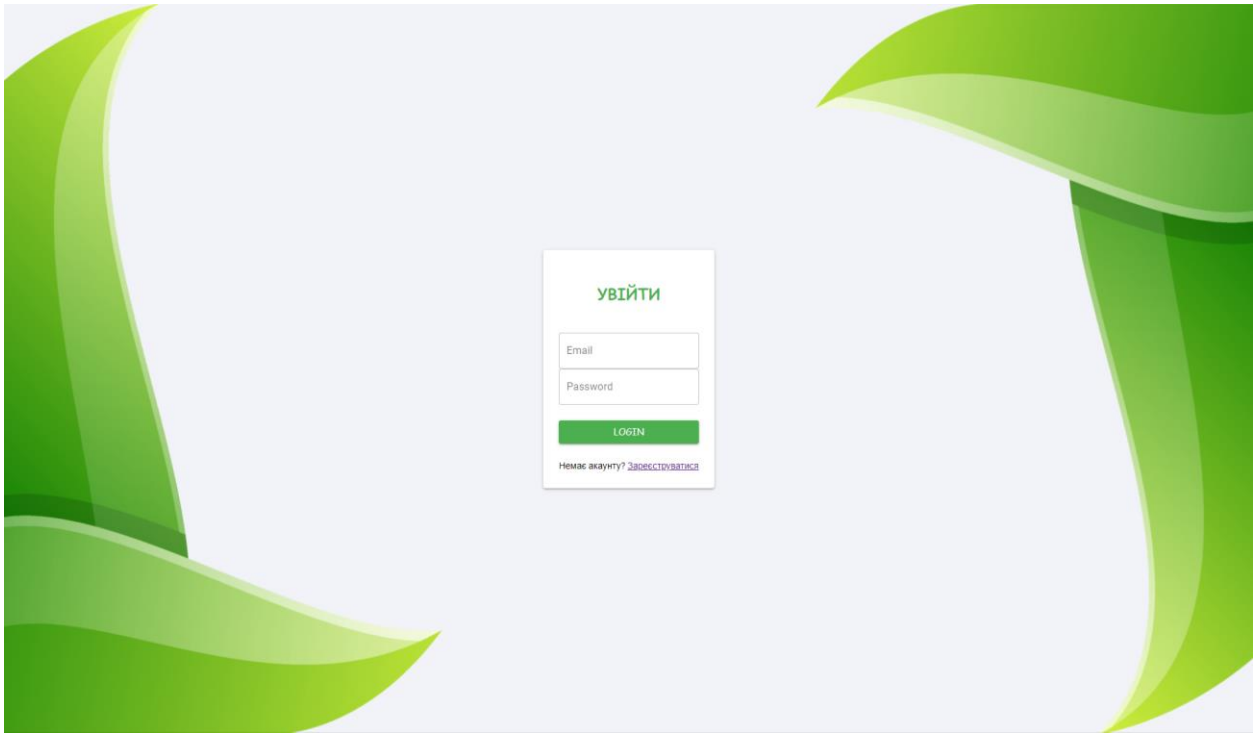

Рис. 2.5. Сторінка авторизації користувача

Якщо користувач введе будь що замість пошти, то буде застосована валідація даних. Зокрема, з'явиться червоне діалогове вікно в нижній частині екрану з повідомленням про помилку. (див. рис. 2.5).

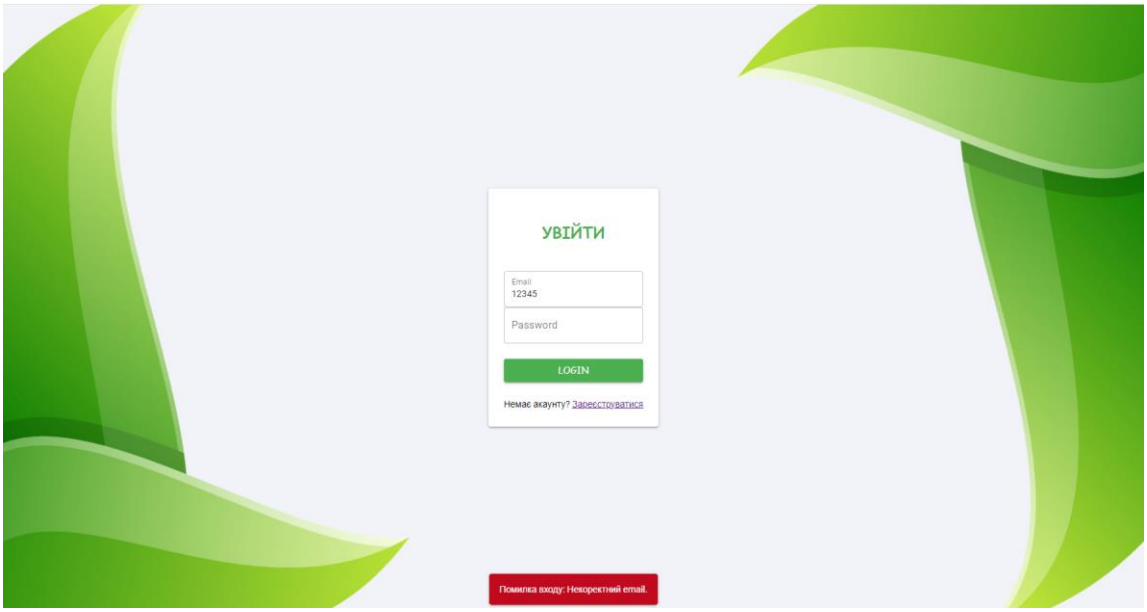

Рис. 2.2. Помилка входу, якщо користувач вводить некоректну пошту

У випадку коли користувач вводить електронну пошту, яка не була зареєстрована або недійсний пароль, то буде також повідомленням про помилку. (див. рис. 2.6).

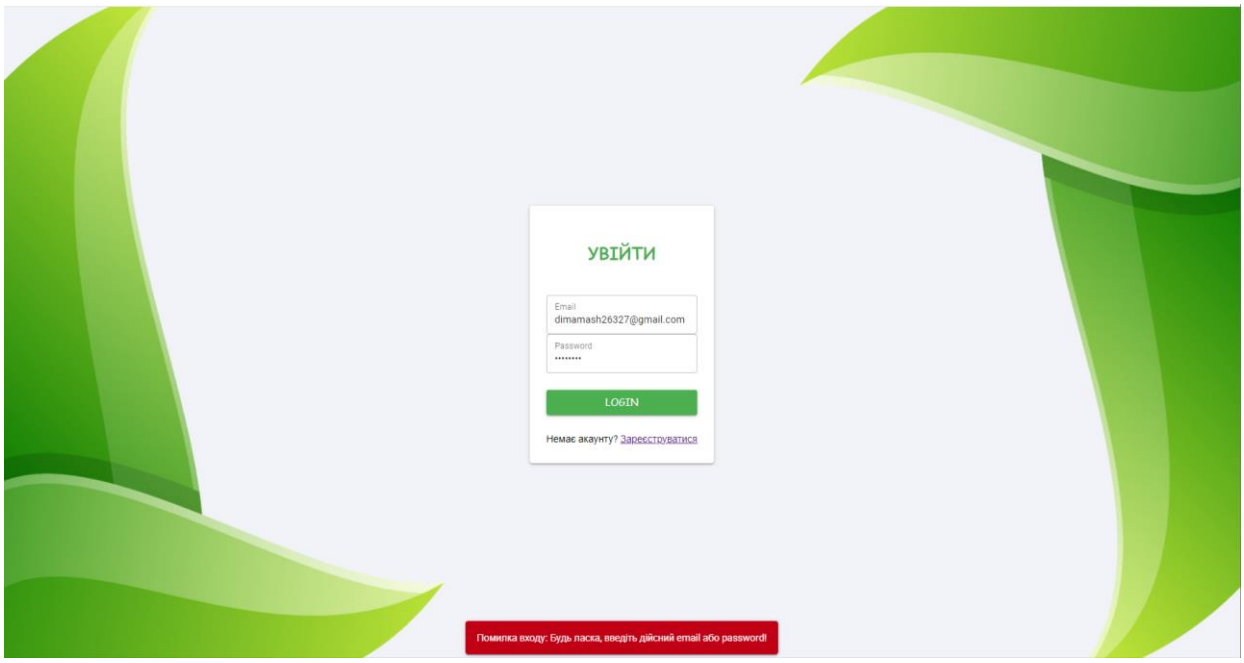

Рис. 2.6. Помилка входу, якщо користувач вводить недійсну пошту або пароль

Після того, як користувач натиснув на кнопку «Зареєструватися», його буде перенаправлено на сторінку реєстрації (див. рис. 2.7), де він зможе ввести пароль та електронну пошту для реєстрації. Якщо користувач хоче повернутися на сторінку входу, він може натиснути на кнопку «Увійти».

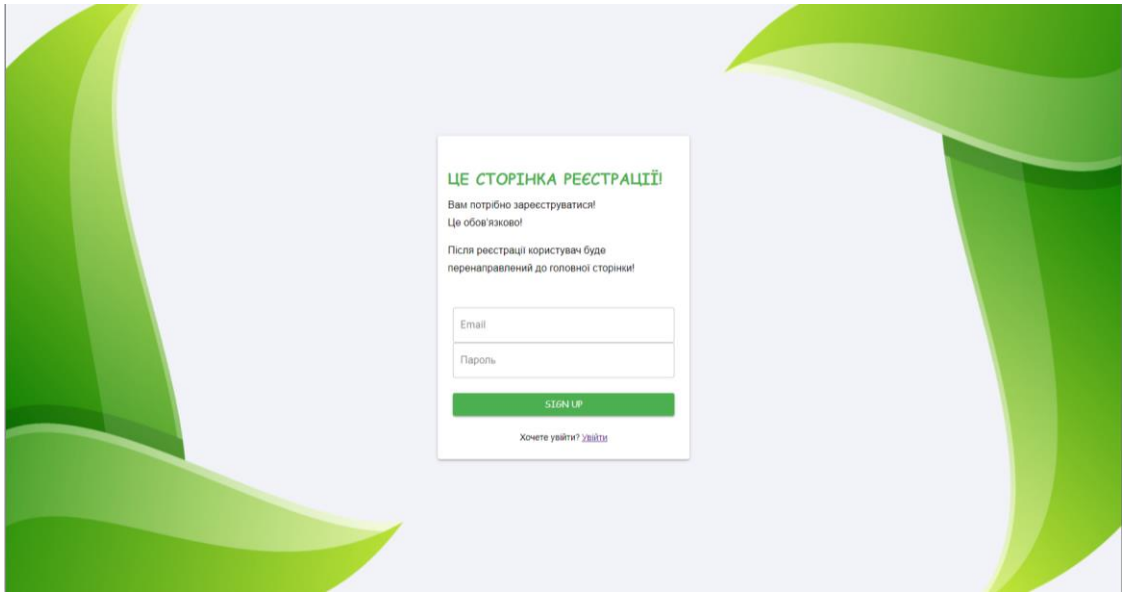

Рис. 2.7. Сторінка реєстрації

На сторінці з реєстрацією також присутня валідація даних, а саме перевірка наявності паролю та пошти (див. рис. 2.8). Після того, як користувач увів свої дані пошти та паролю, він може натиснути на кнопку «SIGN UP».

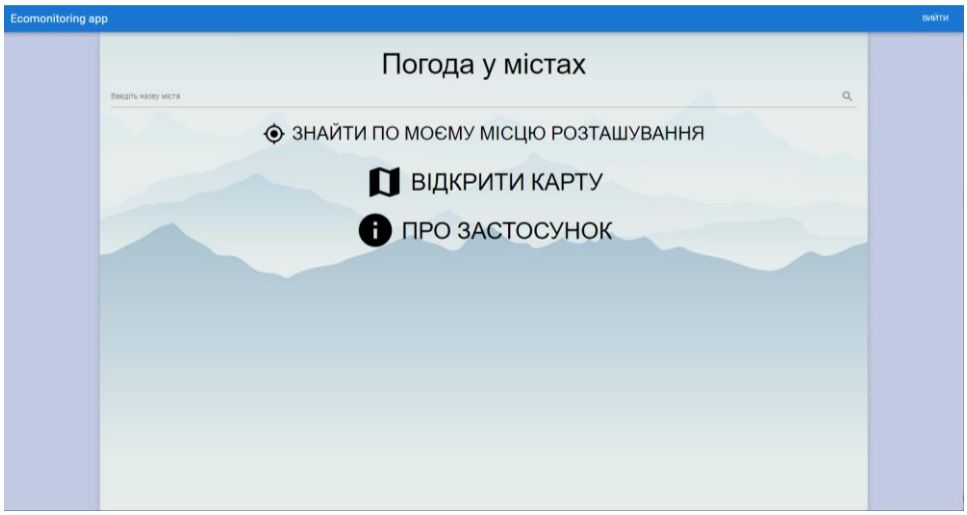

Рис. 2.8. Сторінка «Ecomonitoring app»

Успішна авторизація користувача призведе до його автоматичного перенаправлення на головну сторінку з назвою «Ecomonitoring app» (див. рис. 2.9). На цій сторінці можна побачити, що у верхньому правому куті розташована кнопка «Вийти», яка дозволяє користувачу вийти з додатку. На цій сторінці можна побачити декілька основний функцій:

1. Знайти по моєму місцю розташування - ця функція дозволяє знайти інформацію про погоду за геолокацією.
2. Відкрити карту - ця функція дозволяє переглядати карту з забрудненнями вітру у різних містах України.
3. Про застосунок - цей розділ надає основну інформацію про додаток.

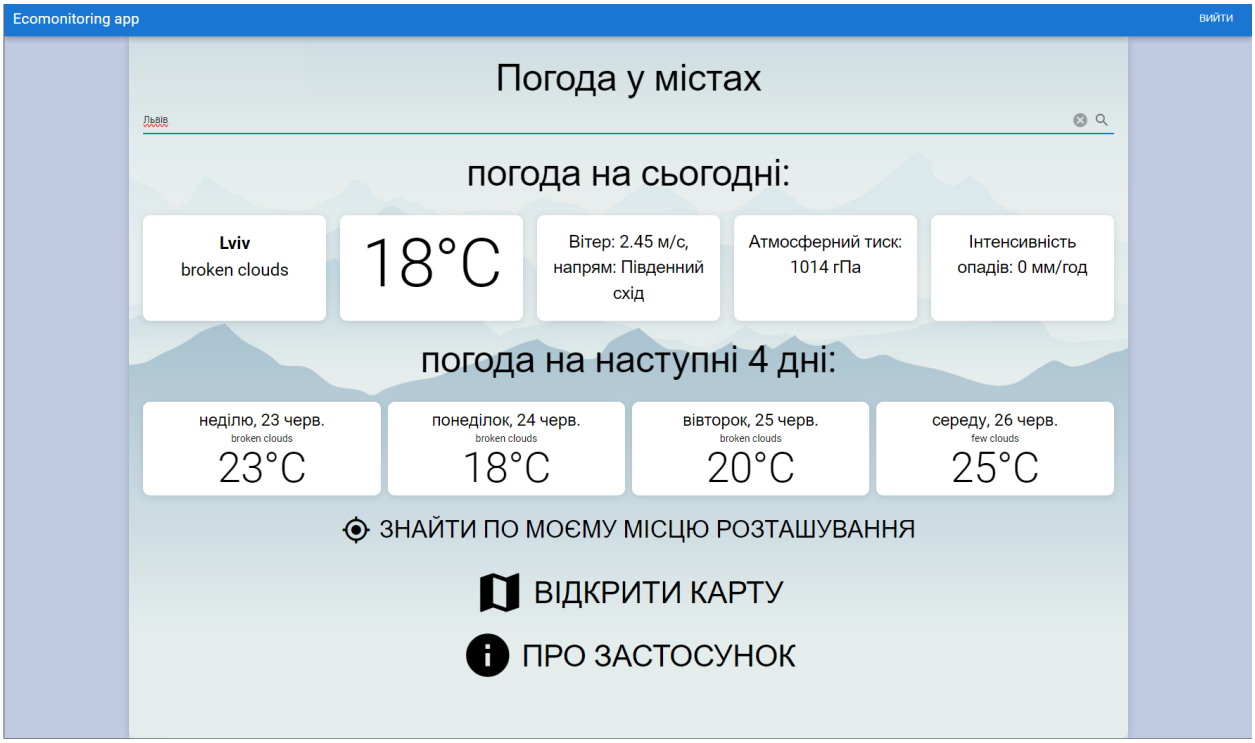

Рис. 2.9. Сторінка «Ecomonitoring app»

В полі над цими кнопками можна ввести назву міста, щоб отримати детальну інформацію про погодні умови в обраному місці, для прикладу давайте напишемо Львів (див. рис. 2.9). Як ми бачимо після того як користувач натисну на кнопку з зображенням лупи, то йому виводиться погода у місті

Львів, а саме погода на сьогодні, та погода на наступні 4 дні. У блоці з погодою на сьогодні, додатково відображаються детальні параметри: напрямок та швидкість вітру, атмосферний тиск, та інтенсивність опадів, а якщо подивитися на блок з погодою на 4 дні, то в ньому відображається прогноз погоди для міста Львів на кожен із наступних чотирьох днів. Для кожного дня зазначено дату, очікувану температуру та погодні умови.

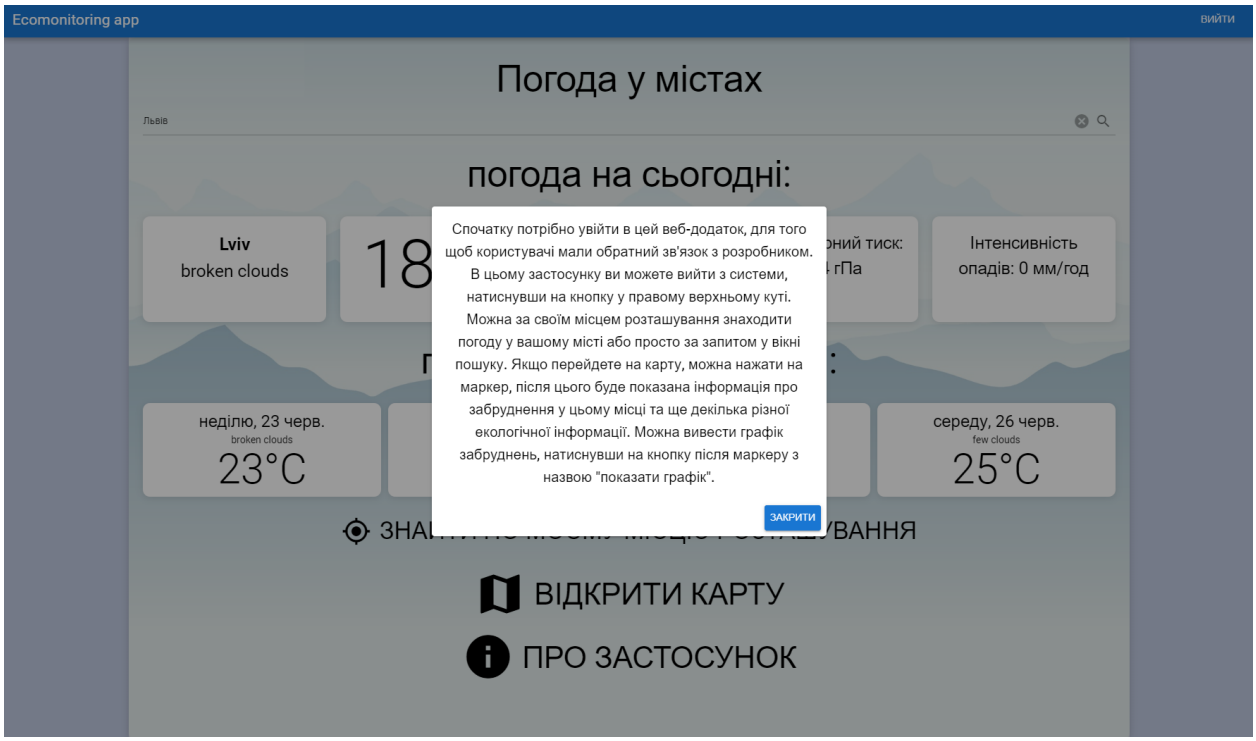

Рис. 2.10. Сторінка «Ecomonitoring app»

Якщо користувач натисне на кнопку з назвою «про застосунок», то він зможе ознайомитись з інформацію про додаток, якщо коротко то це його призначення та функціонал (див. рис. 2.10). Коли користувачу потрібно буде вийти з цього модального вікна, то він може натиснути на кнопку «закрити». Закриття вікні відбувається за допомогою функцій closeInfoModal(), а відкриття за допомогою showInfoModal().

Якщо говорити про карту, то щоб її відкрити, користувачу потрібно натиснути на кнопку «відкрити карту».

Після натискання на кнопку «відкрити карту» відкривається сторінка з картою (див. рис. 2.11). Для відображення карти використовується бібліотека Leaflet, на цю карту додаються маркери за допомогою функції addCityMarkers(), усього таких маркерів у додатку 24 [14].

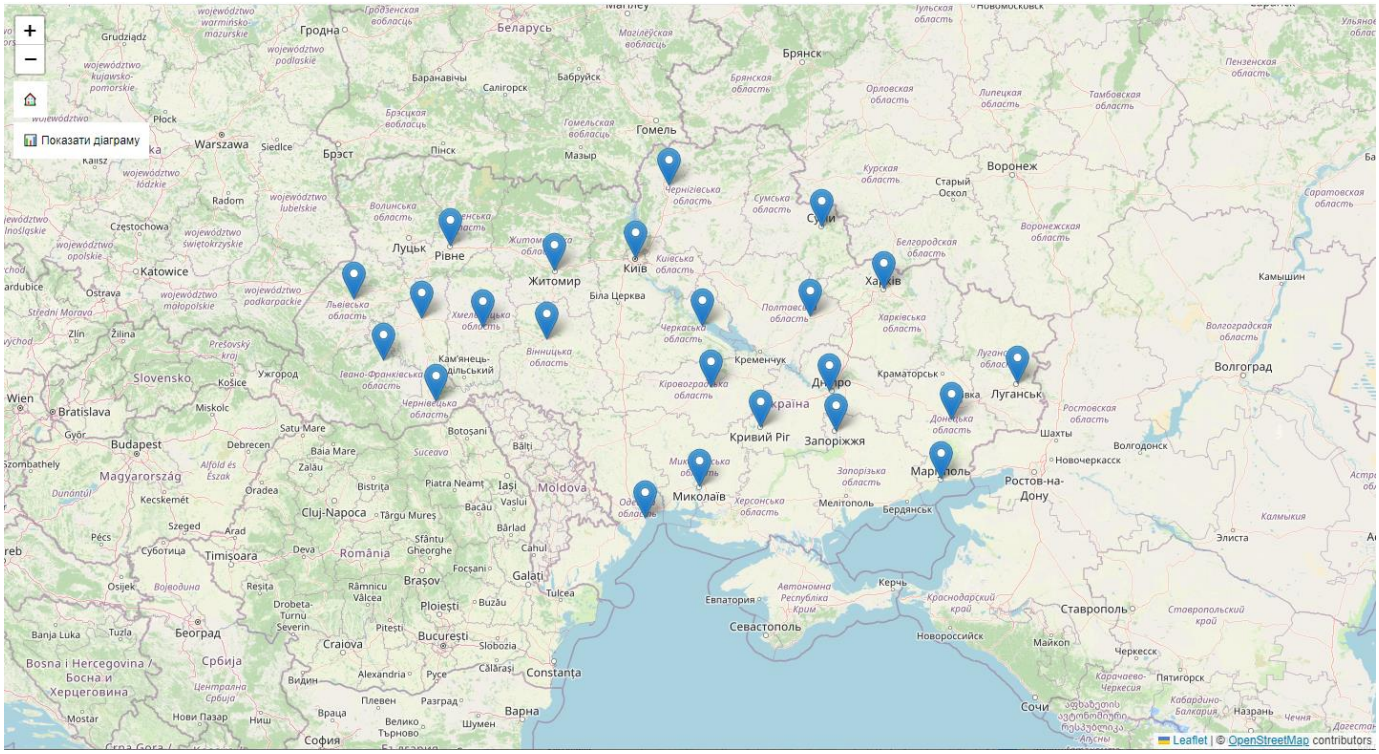

Рис. 2.11. Карта України з маркерами

Коли користувач натискає на маркер, викликається функція onMarkerClick(cityName), яка запитує дані про погоду та забруднення для обраного міста, наприклад Одеси, в відобразяться різні дані, такі як погода (у першому стовпчику) та забруднення (у другому стовпчику) (див. рис. 2.12).

Для отримання даних про погоду (fetchWeatherData(cityName)) та забруднення (fetchPollutionData(cityName)) була використана бібліотека Fetch API.

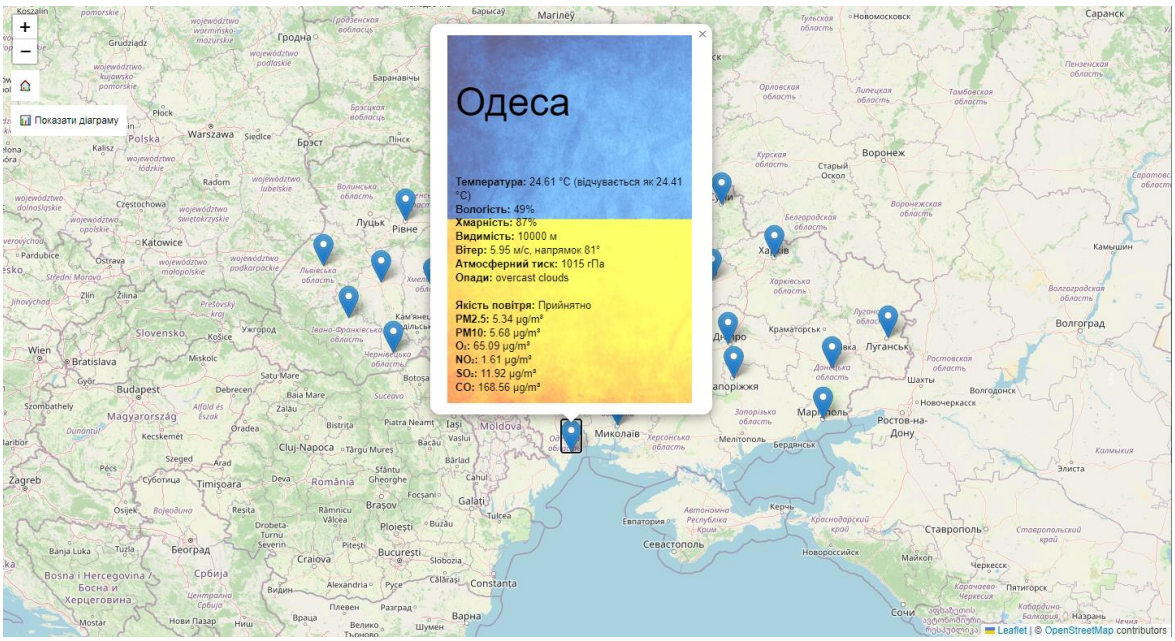

Рис. 2.12. Моніторинг якості повітря та погоди в місті Одеса

Після того, як користувач натисне на кнопку «Показати діаграму», відображається діаграма, яка показує рівні забруднення повітря за різними категоріями (PM2.5, PM10, O_3, NO_2, SO_2, CO) (див. рис. 2.13). Для побудови діаграми використовується бібліотека Chart.js [16].

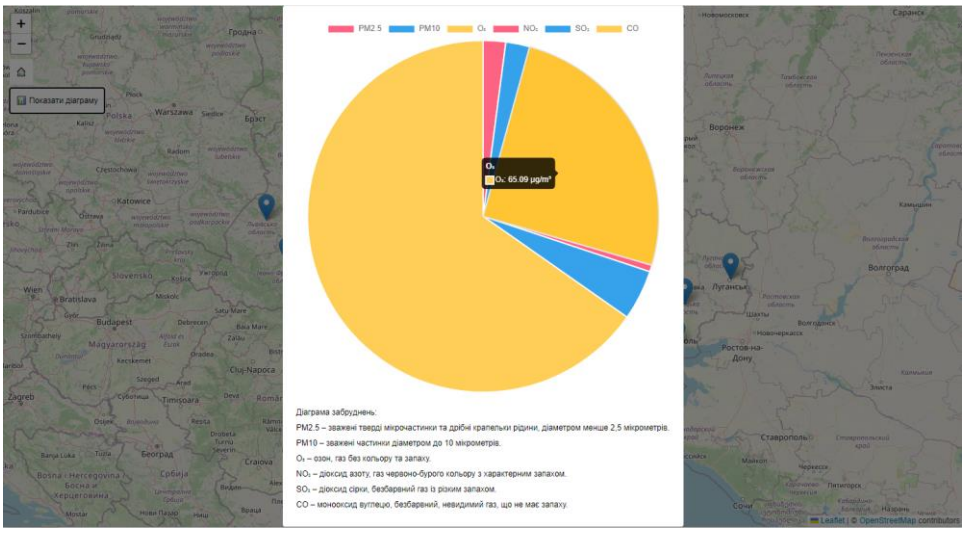

Рис. 2.13. Діаграма забруднень

Дані для діаграми отримуються через API запити до сервісів, які надають інформацію про якість повітря, за допомогою функції fetchPollutionChartData(cityName). Ці дані обробляються та візуалізуються у вигляді діаграми за допомогою функції renderPollutionChart(data).

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми – 1000.
2. Коефіцієнт корекції програми в ході її розробки – 0,25.
3. Коефіцієнт складності програми – 1,2.
4. Годинна заробітна плата програміста – 150 грн/год.

Примітка:

Заробітна плата програміста кваліфікації Junior Software Engineer, який працює з мовою програмування JavaScript, становить в середньому десь $500-$1200 на місяць [11]. Заробітна плата програміста за годину виходить 150 грн/год. Це якщо програміст буде працювати 22 дня на місяць, враховуючи курс Національоного банку України, а саме 1 долар = 40 грн.

Визначення норм праці для створення програмного забезпечення є складним завданням через творчий характер роботи програміста. Тому трудомісткість розробки програмного забезпечення може бути оцінена лише приблизно, використовуючи різні моделі з різним рівнем точності.

5. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3.

6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1.

7. Вартість машино-години ЕОМ – 60 грн/год.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ люд-годин,} \quad (3.1)$$

де:

- де t_o-витрати праці на підготовку й опис поставленої задачі (приймається 45 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q \cdot C(1+p) \quad (3.2)$$

де:

- q - передбачуване число операторів (850);

- с - коефіцієнт складності програми (1,15);

- р - коефіцієнт корекції програми в ході її розробки (0,2).

Звідси умовне число операторів в програмі:

$Q = 1000 \cdot 1,2 \cdot (1 + 0,5) = 1500$

Витрати праці на вивчення опису задачі t_u визначаються з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин} \quad (3.3)$$

де:

- B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,25);

- k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,2).

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,25). З урахуванням коефіцієнта кваліфікації k = 1,1, отримуємо витрати праці на вивчення опису завдання:

$t_u = (1500 \cdot 1,3) / (80 \cdot 1,1) = 22,27$ людино-годин

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k}, \text{ людино-годин} \quad (3.4)$$

Підставивши відповідні значення в формулу, отримаємо:

t_a= 1500/ (22,5 · 1,1) = 60,61 людино-годин

Витрати праці на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин} \quad (3.5)$$

Підставивши відповідні значення в формулу, отримаємо:

t_n= 1500 / (22,5 · 1) = 66,67 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- За умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k} \quad (3.6)$$

$t_{отл}$= 1500/ (5 · 1) = 300 людино-годин

- За умови комплексного налагодження завдання:

$$t^к_{отл}=1,5\cdot t_{отл} \quad (3.7)$$

$t^к_{отл}$= 1,5 · 300 = 450 людино-годин

Витрати праці на підготовку документації визначаються за формулою:

$$t_д=t_{др}+t_{до} \quad (3.8)$$

де:

- де $t_{др}$-трудомісткість підготовки матеріалів і рукопису.

- $t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \quad (3.9)$$

$$t_{до}=0,75 \cdot t_{др} \quad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$= 1500/ (18 · 1,1) = 75,76 людино-годин.

$t_{до}$ = 0,75 · 75,76 = 56,82 людино-годин.

$t_д$= 75,76 + 56,82 = 132,58 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 22,27 + 60,61 + 66,67 + 450 + 132,58 = 728.13 людино-години

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $В_{ПЗ}$ включають витрати на заробітну плату виконавця програми $В_{ЗП}$і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$В_{ПЗ}=В_{ЗП}+В_{Мч}, \text{ грн}, \quad (3.11)$$

де:

- $В_{ПЗ}$ – заробітна плата виконавців, яка визначається за формулою:

$$В_{ЗП}=t\cdot С_{ПР}, \text{ грн}, \quad (3.12)$$

де:

- t - загальна трудомісткість, людино-годин

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/год.

З урахуванням того, що середня годинна зарплата програміста становить 200 грн/год, отримуємо:

$$В_{ЗП}= 782{,}13 \cdot 150 = 117319{,}5 \text{грн}.$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$В_{Мч} = t_{отл} \cdot СМ, \text{ грн}, \quad (3.13)$$

де:

- $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год

- $C_{мч}$ - вартість машино-години ЕОМ, грн/год (55 грн/год).

Підставивши в формулу (3.13) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$В_{Мч} = 450 \cdot 60 = 27000 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$В_{ПЗ} = 117319,5 + 27000 = 144319 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{міс.} \quad (3.14)$$

де:

- Bk- число виконавців (дорівнює 1);

- Fp - місячний фонд робочого часу (при 40-годинному робочому тижні (Fp = 168 годин).

Звідси очікуваний період створення ПЗ:

T= 782,13 / 1·168 ≈ 4,65 міс.

Висновки: трудомісткість розробленого веб-застосунку становить 782,13 людино-години. Обчислена вартість роботи по створенню програми дорівнює 144319,5 гривень. Визначений затрачений час на створення програмного забезпечення становить 4,65 місяців.

ВИСНОВКИ

У межах представленої кваліфікаційної роботи було успішно розроблено веб-застосунок для екологічного моніторингу клієнтської частини. Основна мета цього застосунку - надання користувачам доступу до різноманітних екологічних даних, зокрема інформації про забруднення повітря та погодні умови.

Для розробки веб-застосунку було використано сучасні технології веб-розробки, включаючи HTML5 для структурування контенту [24], CSS3 для стилізації [17], та JavaScript (ES6+) для створення інтерактивності [21]. Це забезпечило створення сучасного, responsive дизайну, який адаптується до різних пристроїв. Застосунок розроблено з використанням Quasar Framework, який базується на Vue.js 3 - прогресивному фреймворку для створення користувацьких інтерфейсів [18]. Це дозволило створити компонентну архітектуру. Для створення інтерфейсу користувача було використано Quasar Framework [13], який надав широкий спектр готових компонентів та інструментів для швидкої розробки.

Для реалізації системи авторизації та аутентифікації користувачів було інтегровано Firebase Authentication, що забезпечило надійний та безпечний механізм управління користувачами [20]. Розробка проводилась у середовищі Visual Studio Code, що сприяло підвищенню продуктивності завдяки широкому набору інструментів та розширень. Застосунок побудовано за принципами Single Page Application (SPA), що забезпечує швидку та плавну навігацію між різними сторінками без перезавантаження [25].

Було створено 5 основних сторінок. Застосунок інтегрується з двома API: OpenWeatherMap API для отримання даних про погоду та забруднення повітря, і Firebase Authentication API для авторизації користувачів [12].

Загальна трудомісткість розробки склала 782,13 людино-годин, що еквівалентно 4,65 місяців роботи. Вартість розробки оцінюється в 144319,5 гривень.

Розроблений веб-застосунок успішно виконує поставлені завдання, надаючи користувачам зручний інтерфейс для доступу до екологічних даних. Використання сучасних технологій та фреймворків забезпечило створення продукту, який відповідає сучасним вимогам веб-розробки. Інтеграція Firebase Authentication гарантує безпечну авторизацію користувачів.

Проект має потенціал для подальшого розвитку, включаючи додавання нових функцій, розширення набору даних для моніторингу та оптимізацію продуктивності. В цілому, розроблений веб-застосунок є ефективним інструментом для екологічного моніторингу, який може бути корисним для широкого кола користувачів, зацікавлених у екологічній ситуації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Особливості web-додаткців. URL: https://sites.znu.edu.ua/webprog/lect/1191.ukr.html (дата звернення: 17.04.24)

2. Хаутов А. П. Екологічний Моніторинг. URL: https://stud.com.ua/135902/ekologiya/ekologichniy_monitoring (дата звернення: 18.04.24)

3. Дія. Відкриті дані. Екологія. URL: https://diia.data.gov.ua/value/ecology (дата звернення: 22.04.24)

4. ТОП-8 Екододатків. URL: https://ecopolitic.com.ua/ua/news/top-8-ekododatkiv-yaki-dopomozhut-ukraincyam-pereviriti-chi-bezpechne-dovkillya-poruch-iz-nimi/ (дата звернення: 20.04.24)

5. Вебзастосунок. URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA (дата звернення: 28.04.24)

6. Основні завдання екологічного моніторингу. URL: https://disted.edu.vn.ua/courses/learn/5374 (дата звернення: 2.05.24)

7. Екологічний моніторинг довкілля. URL: https://ecologyknu.wixsite.com/ecologymanual/10-ekologichnij-monitoring-dovkilly (дата звернення: 4.05.24)

8. Рівні екологічного моніторингу. URL: https://ecomonitoring.info/2019/07/15/%D0%BF%D1%80%D0%BE-%D1%80%D1%96%D0%B2%D0%BD%D1%96-%D0%B5%D0%BA%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%BE%D0%B3%D0%BE-%D0%BC%D0%BE%D0%BD%D1%96%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D1%83/ (дата звернення: 5.05.24)

9. JS Tutorial. URL: https://www.w3schools.com/js/ (дата звернення: 6.05.24)

10. Моніторинг довкілля. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D1%96%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3_%D0%B4%D0%BE%D0%B2%D0%BA%D1%96%D0%BB%D0%BB%D1%8F (дата звернення: 1.05.24)

11. Скільки заробляють IT-фахівці в Україні у 2023-2024 році. URL:

https://lemon.school/blog/skolko-zarabatyvayut-it-speczyalysty-v-ukrayne (дата звернення: 8.05.24)

12. Firebase Documentation. URL: https://firebase.google.com/docs/guides?hl=ru (дата звернення: 15.05.24)

13. Quasar Documentation. URL: https://quasar.dev/docs/ (дата звернення: 12.05.24)

14. Leaflet Tutorials. Url: https://leafletjs.com/examples.html (дата звернення: 25.05.24)

15. Let's get started with Chart.js! URL:https://www.chartjs.org/docs/latest/getting-started/ (дата звернення: 27.05.24)

16. Chart.js Documentation. Url: https://davidaparicio.github.io/chartjs/ (дата звернення: 27.05.24)

17. CSS Tutorial. URL: https://www.javatpoint.com/css-tutorial (дата звернення: 5.05.24)

18. Hands-On Full Stack Web Development with Vue.js 2.0" by Roman Kuba 2024.-366 p. (дата звернення: 21.05.24)

19. Vue.js in Action" by Erik Hanchett and Benjamin Listwon 2024.-304 p. (дата звернення: 23.05.24)

20. Firebase Essentials" by Ashok Kumar S 2018.-535 p. (дата звернення: 24.05.24)

21. Data Visualization with JavaScript" by Stephen A. Thomas 2015.-365 p. (дата звернення: 20.05.24)

22. Interactive Data Visualization for the Web" by Scott Murray. 2017.-474 p. (дата звернення: 30.05.24)

23 . Eloquent JavaScript" by Marijn Haverbeke 2014.-512p. (дата звернення: 27.05.24)

24. HTML and CSS: Design and Build Websites" by Jon Duckett 2011.-472 p. (дата звернення: 29.05.24)

25. Clean Code: A Handbook of Agile Software Craftsmanship 2008.-496 p. (дата звернення: 8.06.24)

Додаток А

ЛІСТИНГ ПРОГРАМИ

IndexPage.vue:

```
<template>
  <!-- Головний контейнер сторінки -->
  <q-page class="weather-app-background flex column justify-center items-center">
    <div class="weather-app-container col q-py-md q-px-lg">
      <!-- Заголовок застосунку -->
      <div class="pogoda_info">Погода у містах</div>

      <!-- Поле пошуку міста -->
      <q-input
        v-model="search"
        @keyup.enter="getWeatherBySearch"
        placeholder="Введіть назву міста"
        dense
        clearable
      >
        <template v-slot:append>
          <!-- Кнопка пошуку -->
          <q-btn
            @click="getWeatherBySearch"
            round
            dense
            flat
            icon="search"
          />
        </template>
      </q-input>

      <!-- Блок відображення погодних даних -->
      <div v-if="weatherData" class="col text-center q-my-md">
        <div class="pogoda_info">погода на сьогодні:</div>
        <div class="weather-info col text-center q-my-md">
          <!-- Блоки з детальною інформацією про погоду -->
          <div class="weather-info-block">
            <div class="text-h4 cityName enlarged-text">{{ weatherData.name }}</div>
            <div class="text-b6 enlarged-text">{{ weatherData.weather.description }}</div>
          </div>
          <div class="weather-info-block">
            <div class="text-h1">{{ Math.round(weatherData.main.temp) }}°C</div>
```

```
            </div>
            <div class="weather-info-block">
              <div class="enlarged-text">Вітер: {{ weatherData.wind.speed }} м/с, напрям: {{ weatherData.wind.direction }}</div>
            </div>
            <div class="weather-info-block">
              <div class="enlarged-text">Атмосферний тиск: {{ weatherData.main.pressure }} гПа</div>
            </div>
            <div v-if="weatherData.precipitation" class="weather-info-block">
              <div class="enlarged-text">Інтенсивність опадів: {{ weatherData.precipitation.intensity }} мм/год</div>
            </div>
          </div>

          <!-- Блок прогнозу погоди на 4 дні -->
          <div class="pogoda_info">погода на наступні 4 дні:</div>
          <div class="forecast-container">
            <div v-for="(day, index) in forecastData" :key="index" class="forecast-day">
              <div class="text-h5">{{ day.date }}</div>
              <div>{{ day.weather.description }}</div>
              <div class="text-h2">{{ Math.round(day.temp) }}°C</div>
            </div>
          </div>
        </div>

        <!-- Кнопка для визначення погоди за геолокацією -->
        <div class="text-center q-my-lg">
          <q-btn @click="getLocation" class="col" flat>
            <q-icon left size="3em" name="my_location" />
            <div class="text-h4">Знайти по моєму місцю розташування</div>
          </q-btn>
        </div>

        <!-- Група кнопок для додаткових функцій -->
        <div class="button-group col flex justify-center">
          <q-btn @click="openLeafletMap" class="col q-mb-sm enlarged-button" flat>
            <q-icon left size="3em" name="map" />
            <div class="enlarged-text">Відкрити карту</div>
          </q-btn>
          <q-btn @click="showInfoDialog" class="col q-mb-sm enlarged-button" flat>
            <q-icon left size="3em" name="info" />
            <div class="enlarged-text">Про застосунок</div>
```

```
        </q-btn>
       </div>
      </div>

      <!-- Діалогове вікно з інформацією про застосунок -->
      <q-dialog v-model="showInfo" persistent content-css="max-width: 300px;">
       <q-card>
        <q-card-section class="text-center">
         <div class="text-h6">
```

Спочатку потрібно увійти в цей веб-додаток, для того щоб користувачі мали обратний зв'язок з розробником. В цьому застосунку ви можете вийти з системи, натиснувши на кнопку у правому верхньому куті. Можна за своїм місцем розташування знаходити погоду у вашому місті або просто за запитом у вікні пошуку. Якщо перейдете на карту, можна нажати на маркер, після цього буде показана інформація про забруднення у цьому місці та ще декілька різної екологічної інформації. Можна вивести графік забруднень, натиснувши на кнопку після маркеру з назвою "показати графік".

```
         </div>
        </q-card-section>
        <q-card-actions align="right">
         <q-btn label="Закрити" color="primary" @click="closeInfoDialog" />
        </q-card-actions>
       </q-card>
      </q-dialog>
     </q-page>
    </template>

    <script>
    import { defineComponent } from 'vue';
    import axios from 'axios';

    export default defineComponent({
     name: 'WeatherApp',
     data() {
      return {
       search: '',           // Змінна для збереження введеного пошукового запиту
       weatherData: null,    // Змінна для збереження даних про погоду
       forecastData: [],     // Змінна для збереження прогнозу погоди
       apiUrl: 'https://api.openweathermap.org/data/2.5',  // Базовий URL API
       apiKey: '7624ed0636cd6fc63420a898f3927f94',        // API ключ для доступу до OpenWeatherMap
       showInfo: false       // Змінна для відображення інформаційного діалогу
      };
     },
     methods: {
```

```
    // Метод для отримання погоди за пошуковим запитом
    async getWeatherBySearch() {
      try {
        // Отримуємо поточну погоду
        const weatherResponse = await axios.get(`${this.apiUrl}/weather?q=${this.search}&appid=${this.apiKey}&units=metric`);
        this.handleWeatherResponse(weatherResponse.data);

        // Отримуємо прогноз погоди
        const forecastResponse = await axios.get(`${this.apiUrl}/forecast?q=${this.search}&appid=${this.apiKey}&units=metric`);
        this.handleForecastResponse(forecastResponse.data);
      } catch (error) {
        console.error('Помилка отримання погодних даних:', error);
      }
    },
    // Метод для отримання геолокації користувача
    async getLocation() {
      navigator.geolocation.getCurrentPosition(position => {
        const { latitude, longitude } = position.coords;
        this.getWeatherByCoords(latitude, longitude);
      });
    },
    // Метод для отримання погоди за координатами
    async getWeatherByCoords(lat, lon) {
      try {
        // Отримуємо поточну погоду
        const weatherResponse = await axios.get(`${this.apiUrl}/weather?lat=${lat}&lon=${lon}&appid=${this.apiKey}&units=metric`);
        this.handleWeatherResponse(weatherResponse.data);

        // Отримуємо прогноз погоди
        const forecastResponse = await axios.get(`${this.apiUrl}/forecast?lat=${lat}&lon=${lon}&appid=${this.apiKey}&units=metric`);
        this.handleForecastResponse(forecastResponse.data);
      } catch (error) {
        console.error('Помилка отримання погодних даних:', error);
      }
    },
    // Обробка відповіді з поточними погодними даними
    handleWeatherResponse(weatherData) {
      const { name, weather, main, wind, rain, snow } = weatherData;
```

```
        const weatherInfo = {
          name,
          weather: {
            description: weather[0].description
          },
          main: {
            temp: Math.round(main.temp),
            pressure: main.pressure
          },
          wind: {
            speed: wind.speed,
            direction: this.getWindDirection(wind.deg)
          },
          precipitation: {
            type: rain ? 'Rain' : snow ? 'Snow' : 'None',
            intensity: rain ? (rain['1h'] || 0) : snow ? (snow['1h'] || 0) : 0
          }
        };
        this.weatherData = weatherInfo;
      },
      // Обробка відповіді з прогнозом погоди
      handleForecastResponse(forecastData) {
        const dailyData = forecastData.list.filter(item => item.dt_txt.includes('12:00:00'));
        this.forecastData = dailyData.slice(1, 5).map(item => ({
          date: new Date(item.dt_txt).toLocaleDateString('uk-UA', { weekday: 'long', day: 'numeric', month: 'short' }),
          temp: item.main.temp,
          weather: {
            description: item.weather[0].description
          }
        }));
      },
      // Метод для визначення напрямку вітру
      getWindDirection(degrees) {
        const directions = ['Північ', 'Північний схід', 'Схід', 'Південний схід', 'Південь', 'Південний захід', 'Захід', 'Північний захід'];
        const index = Math.round(degrees / 45) % 8;
        return directions[index];
      },
      // Метод для відкриття карти Leaflet
      openLeafletMap() {
        this.$router.push('/leaflet-map');
```

```
    },
    // Метод для показу інформаційного діалогу
    showInfoDialog() {
      this.showInfo = true;
    },
    // Метод для закриття інформаційного діалогу
    closeInfoDialog() {
      this.showInfo = false;
    }
  }
});
</script>

<style scoped>
/* Стилі для фону застосунку */
.weather-app-background {
  background: linear-gradient(135deg, hsla(223, 36%, 81%, 0.945), hsla(223, 36%, 81%, 0.945));
  height: 80vh;
  display: flex;
  justify-content: center;
  align-items: center;
  overflow: hidden;
}

/* Стилі для контейнера застосунку */
.weather-app-container {
  background-image: url('src/assets/bg-app.jpg');
  background-size: cover;
  background-position: center;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
  width: 80%;
  max-width: 1500px;
  position: relative;
  z-index: 1;
}

/* Псевдоелемент для створення напівпрозорого шару */
.weather-app-container::before {
  content: '';
  position: absolute;
```

```
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: rgba(255, 255, 255, 0.5);
  z-index: -1;
}

/* Стилі для блоків з інформацією про погоду */
.weather-info {
  margin-top: 20px;
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  grid-gap: 20px;
}

/* Стилі для окремих блоків погодної інформації */
.weather-info-block {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

/* Стилі для назви міста */
.cityName {
  font-size: 24px;
  font-weight: bold;
}

/* Стилі для центрування тексту */
.text-center {
  text-align: center;
}

/* Стилі для групи кнопок */
.button-group {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

```
/* Стилі для збільшених кнопок */
.enlarged-button {
  font-size: 1.75em;
}

/* Стилі для збільшеного тексту */
.enlarged-text {
  font-size: 1.75em;
}

/* Стилі для контейнера прогнозу */
.forecast-container {
  margin-top: 20px;
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 10px;
}

/* Стилі для блоку прогнозу на день */
.forecast-day {
  background-color: #fff;
  padding: 10px;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

/* Стилі для заголовків секцій погоди */
.pogoda_info {
  text-align: center;
  font-size: 3.75em;
}
</style>
```

LeafletMap.vue:

```
<template>
  <!-- Головний контейнер для карти -->
  <div id="map" ref="mapRef">
    <!-- Контейнер для кнопок -->
    <div class="button-container">
      <!-- Кнопка для повернення на головну сторінку -->
```

```
        <button @click="goHome" class="home-button">🏠</button>
        <!-- Кнопка для відображення графіку якості повітря -->
        <button v-if="selectedCity && airQualityData" @click="showChart" class="graph-button">📊 Показати діаграму</button>
      </div>
      <!-- Модальне вікно для відображення графіку -->
      <div v-if="showModal" class="modal" @click.self="closeChart">
        <div class="modal-content">
          <!-- Канвас для графіку -->
          <canvas id="airQualityChart"></canvas>
          <!-- Інформація про забруднювачі -->
          <div class="pollutant-info">
            <p>Діаграма забруднень:</p>
            <p>PM2.5 – зважені тверді мікрочастинки та дрібні крапельки рідини, діаметром менше 2,5 мікрометрів.</p>
            <p>PM10 – зважені частинки діаметром до 10 мікрометрів.</p>
            <p>O₃ – озон, газ без кольору та запаху.</p>
            <p>NO₂ – діоксид азоту, газ червоно-бурого кольору з характерним запахом.</p>
            <p>SO₂ – діоксид сірки, безбарвний газ із різким запахом.</p>
            <p>CO – монооксид вуглецю, безбарвний, невидимий газ, що не має запаху.</p>
          </div>
        </div>
      </div>
    </div>
  </template>

  <script>
  // Імпорт необхідних модулів
  import L from 'leaflet';
  import 'leaflet/dist/leaflet.css';
  import { Chart, PieController, ArcElement, Tooltip, Legend } from 'chart.js';

  // Реєстрація компонентів Chart.js
  Chart.register(PieController, ArcElement, Tooltip, Legend);

  export default {
    name: 'LeafletMap',
    data() {
      return {
        // Ініціалізація даних компонента
        map: null,
```

```
    ukraineBounds: [[52.37, 22.13], [41.18, 40.24]],
    cities: [
      { name: 'Київ', coordinates: [50.4501, 30.5234] },
      { name: 'Харків', coordinates: [49.9935, 36.2304] },
      { name: 'Одеса', coordinates: [46.4775, 30.7326] },
      { name: 'Дніпро', coordinates: [48.4647, 34.9871] },
      { name: 'Донецьк', coordinates: [48.0153, 37.8032] },
      { name: 'Запоріжжя', coordinates: [47.8388, 35.1396] },
      { name: 'Львів', coordinates: [49.8397, 24.0297] },
      { name: 'Кривий Ріг', coordinates: [47.9017, 33.3953] },
      { name: 'Миколаїв', coordinates: [46.9749, 31.9946] },
      { name: 'Маріуполь', coordinates: [47.0953, 37.5407] },
      { name: 'Луганськ', coordinates: [48.5676, 39.3179] },
      { name: 'Вінниця', coordinates: [49.2331, 28.4682] },
      { name: 'Полтава', coordinates: [49.5888, 34.5514] },
      { name: 'Чернігів', coordinates: [51.4982, 31.2893] },
      { name: 'Черкаси', coordinates: [49.4444, 32.0598] },
      { name: 'Житомир', coordinates: [50.2547, 28.6587] },
      { name: 'Суми', coordinates: [50.9076, 34.7988] },
      { name: 'Хмельницький', coordinates: [49.4238, 27.0013] },
      { name: 'Чернівці', coordinates: [48.2924, 25.9352] },
      { name: 'Рівне', coordinates: [50.6199, 26.2516] },
      { name: 'Івано-Франківськ', coordinates: [48.9226, 24.7111] },
      { name: 'Кропивницький', coordinates: [48.5134, 32.2623] },
      { name: 'Тернопіль', coordinates: [49.5538, 25.5948] },
    ],
    selectedCity: null,
    showModal: false,
    airQualityData: null,
    weatherData: null,
    chart: null,
  };
},
mounted() {
  // Ініціалізація карти при монтуванні компонента
  this.initMap();
},
methods: {
  // Метод ініціалізації карти
  initMap() {
    this.map = L.map(this.$refs.mapRef).fitBounds(this.ukraineBounds);
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
```

```
      attribution:
        '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
      }).addTo(this.map);

      this.cities.forEach((city) => {
        const marker = L.marker(city.coordinates).addTo(this.map);
        marker.bindPopup(`<h3>${city.name}</h3>`);
        this.getWeatherInfo(city.coordinates, marker, city.name);
        this.getAirQualityInfo(city.coordinates, marker, city.name);
        marker.on('click', () => {
          this.selectedCity = city.name;
          this.getWeatherInfo(city.coordinates, marker, city.name);
          this.getAirQualityInfo(city.coordinates, marker, city.name);
        });
      });
    },
    // Метод отримання інформації про погоду
    async getWeatherInfo(coordinates, marker, cityName) {
      const apiKey = '7624ed0636cd6fc63420a898f3927f94';
      const [lat, lon] = coordinates;
      const url = `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${apiKey}&units=metric`;

      try {
        const response = await fetch(url);
        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }
        const data = await response.json();

        const { wind, main, weather, clouds, visibility } = data;
        const windSpeed = wind.speed;
        const windDirection = wind.deg;
        const pressure = main.pressure;
        const precipitation = weather[0].description;
        const temperature = main.temp;
        const feelsLike = main.feels_like;
        const humidity = main.humidity;
        const cloudiness = clouds.all;
        const visibilityDistance = visibility || 'Не вдалося отримати дані';

        this.weatherData = {
```

```
            temperature,
            feelsLike,
            humidity,
            cloudiness,
            visibilityDistance,
            windSpeed,
            windDirection,
            pressure,
            precipitation,
          };

          this.updatePopupContent(marker, cityName);
        } catch (error) {
          console.error('Error fetching weather data:', error);
          marker.setPopupContent(`<h3>${cityName}</h3><br>Не вдалося отримати дані про погоду для ${cityName}.`);
        }
      },
      // Метод отримання інформації про якість повітря
      async getAirQualityInfo(coordinates, marker, cityName) {
        const apiKey = '7624ed0636cd6fc63420a898f3927f94';
        const [lat, lon] = coordinates;
        const url = `http://api.openweathermap.org/data/2.5/air_pollution?lat=${lat}&lon=${lon}&appid=${apiKey}`;

        try {
          const response = await fetch(url);
          if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
          }
          const data = await response.json();

          if (data.list && data.list.length > 0) {
            const aqi = data.list[0].main.aqi;
            const airQuality = this.getAirQualityDescription(aqi);
            const { pm2_5, pm10, o3, no2, so2, co } = data.list[0].components;

            this.airQualityData = { airQuality, pm2_5, pm10, o3, no2, so2, co };

            this.updatePopupContent(marker, cityName);
          } else {
```

```
            marker.setPopupContent(`<h3>${cityName}</h3><br>Не вдалося отримати дані про якість повітря.`);
          }
        } catch (error) {
          console.error('Error fetching air quality data:', error);
          marker.setPopupContent(`<h3>${cityName}</h3><br>Не вдалося отримати дані про якість повітря.`);
        }
      },
      // Метод оновлення вмісту спливаючого вікна маркера
      updatePopupContent(marker, cityName) {
        if (this.weatherData && this.airQualityData) {
          const {
            temperature,
            feelsLike,
            humidity,
            cloudiness,
            visibilityDistance,
            windSpeed,
            windDirection,
            pressure,
            precipitation,
          } = this.weatherData;

          const {
            airQuality,
            pm2_5,
            pm10,
            o3,
            no2,
            so2,
            co,
          } = this.airQualityData;

          const popupContent = `
            <div style="position: relative; width: 100%; height: 100%; padding: 10px; box-sizing: border-box;">
              <div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; background-image: url('https://sotni.ru/wp-content/uploads/2023/08/flag-ukrainy-17-1.webp'); background-size: cover; opacity: 0.6;"></div>
              <div style="position: relative; z-index: 1; color: black; max-height: 100%; overflow: auto;">
                <h3>${cityName}</h3>
                <br><b>Температура:</b> ${temperature} °C (відчувається як ${feelsLike} °C)<br>
```

```
        <b>Вологість:</b> ${humidity}%<br>
        <b>Хмарність:</b> ${cloudiness}%<br>
        <b>Видимість:</b> ${visibilityDistance} м<br>
        <b>Вітер:</b> ${windSpeed} м/с, напрямок ${windDirection}°<br>
        <b>Атмосферний тиск:</b> ${pressure} гПа<br>
        <b>Опади:</b> ${precipitation}<br>
        <br><b>Якість повітря:</b> ${airQuality}<br>
        <b>PM2.5:</b> ${pm2_5} µg/m³<br>
        <b>PM10:</b> ${pm10} µg/m³<br>
        <b>O₃:</b> ${o3} µg/m³<br>
        <b>NO₂:</b> ${no2} µg/m³<br>
        <b>SO₂:</b> ${so2} µg/m³<br>
        <b>CO:</b> ${co} µg/m³
      </div>
    </div>
   `;

   marker.setPopupContent(popupContent);
  }
 },
 // Метод отримання опису якості повітря
 getAirQualityDescription(aqi) {
  switch (aqi) {
   case 1:
    return 'Добре';
   case 2:
    return 'Прийнятно';
   case 3:
    return 'Помірно';
   case 4:
    return 'Погано';
   case 5:
    return 'Дуже погано';
   default:
    return 'Невідомо';
  }
 },
 // Метод повернення на головну сторінку
 goHome() {
  this.map.fitBounds(this.ukraineBounds);
  this.$router.push('/app');
 },
```

```
// Метод відображення графіку якості повітря
showChart() {
  this.showModal = true;
  this.$nextTick(() => {
    if (this.chart) {
      this.chart.destroy();
    }
    const ctx = document.getElementById('airQualityChart').getContext('2d');
    this.chart = new Chart(ctx, {
      type: 'pie',
      data: {
        labels: ['PM2.5', 'PM10', 'O₃', 'NO₂', 'SO₂', 'CO'],
        datasets: [{
          data: [
            this.airQualityData.pm2_5,
            this.airQualityData.pm10,
            this.airQualityData.o3,
            this.airQualityData.no2,
            this.airQualityData.so2,
            this.airQualityData.co,
          ],
          backgroundColor: [
            '#FF6384',
            '#36A2EB',
            '#FFCE56',
            '#FF6384',
            '#36A2EB',
            '#FFCE56',
          ],
        }],
      },
      options: {
        responsive: true,
        plugins: {
          legend: {
            position: 'top',
          },
          tooltip: {
            callbacks: {
              label: function(tooltipItem) {
                return tooltipItem.label + ': ' + tooltipItem.raw + ' µg/m³';
              },
```

```
            },
          },
        },
      },
    });
  });
 },
 // Метод закриття графіку
 closeChart() {
  this.showModal = false;
 },
},
};
</script>

<style scoped>
/* Стилі для карти */
#map {
 height: 100vh;
 width: 100%;
 position: relative;
}

/* Стилі для контейнера кнопок */
.button-container {
 position: absolute;
 top: 80px;
 left: 10px;
 z-index: 1000;
 display: flex;
 flex-direction: column;
 align-items: flex-start;
}

/* Стилі для кнопок */
.home-button,
.graph-button {
 background: white;
 border: none;
 padding: 10px;
 cursor: pointer;
 margin-bottom: 5px;
```

```
}
/* Стилі для модального вікна */
.modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  background: rgba(0, 0, 0, 0.5);
  z-index: 1001;
}

/* Стилі для вмісту модального вікна */
.modal-content {
  background: white;
  padding: 20px;
  border-radius: 5px;
  width: 80%;
  max-width: 600px;
}

/* Стилі для інформації про забруднювачі */
.pollutant-info {
  margin-top: 20px;
  text-align: left;
}

/* Стилі для параграфів у інформації про забруднювачі */
.pollutant-info p {
  margin: 5px 0;
}
</style>
```

PageForRegistration.vue:

```
<template>
  <q-page class="registration-page">
    <q-card class="registration-card">
      <!-- Заголовок та інструкції -->
      <q-card-section class="q-pt-xl">
        <div class="text-h4 text-uppercase q-my-none text-weight-bold text-primary fredoka">
```

```
            Це сторінка реєстрації!
          </div>
          <div class="text q-my-sm text-subtitle1">
            Вам потрібно зареєструватися!<p>Це обов'язково!</p>
          </div>
          <div class="text q-my-sm text-subtitle1">
            Після реєстрації користувач буде перенаправлений до головної сторінки!
          </div>
        </q-card-section>

        <!-- Форма реєстрації -->
        <q-card-section class="q-pa-md">
          <q-form ref="form" @submit="submit">
            <q-input v-model="user.email" label="Email" name="email" outlined />
            <q-input v-model="user.password" label="Пароль" name="password" type="password" outlined />

            <!-- Кнопка реєстрації -->
            <q-btn class="full-width q-mt-lg fredoka" label="Sign up" type="submit" style="background-color:
#4CAF50; color: #fff;" />
          </q-form>

          <!-- Посилання на сторінку входу -->
          <div class="q-mt-lg text-center">
            Хочете увійти? <router-link to="/login">Увійти</router-link>
          </div>
        </q-card-section>
      </q-card>
    </q-page>
  </template>

  <script setup>
  import { ref, reactive } from 'vue';
  import { useRouter } from 'vue-router';
  import { Notify } from 'quasar';
  import register from 'src/firebase/firebase-register';

  const router = useRouter();

  // Об'єкт для зберігання даних користувача
  const user = reactive({
    email: '',
    password: ''
```

```
});

const form = ref(null);

// Функція для обробки відправки форми
const submit = async () => {
  if (form.value.validate()) {
    try {
      if (!user.password) {
        throw new Error('Будь ласка, введіть пароль.');
      }
      await register(user);
      router.push('/app');
    } catch (error) {
      console.error('Registration error:', error);
      // Відображення повідомлення про помилку
      Notify.create({
        message: `Помилка реєстрації: ${error.message}`,
        color: 'negative',
        position: 'bottom'
      });
    }
  } else {
    // Повідомлення про неправильні дані
    Notify.create({
      message: 'Будь ласка, перевірте правильність введених даних.',
      color: 'negative',
      position: 'bottom'
    });
  }
};
</script>

<style scoped>
/* Контейнер сторінки реєстрації */
.registration-page {
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}
 /* Картка реєстрації */
```

```
.registration-card {
  max-width: 400px;
  width: 100%;
}

.text-h4 {
  font-size: 1.5rem;
}

.text-primary {
  color: #4CAF50 !important;
}

.full-width {
  width: 100%;
}

.q-pt-xl {
  padding-top: 3rem !important;
}

.q-pa-md {
  padding: 1.5rem !important;
}

.q-mt-lg {
  margin-top: 1.5rem !important;
}

.fredoka {
  font-family: 'Fredoka One', cursive;
}
</style>
```

PageForLogin.vue:

```
<template>
  <q-page class="login-page">
    <q-card class="login-card">
      <!-- Заголовок -->
      <q-card-section class="q-pt-xl text-center">
        <div class="text-h4 text-uppercase q-my-none text-weight-bold text-primary fredoka">Увійти</div>
      </q-card-section>
```

```
        <!-- Форма входу -->
        <q-card-section class="q-pa-md">
          <q-form ref="form" @submit="submit">
            <q-input v-model="user.email" label="Email" name="email" outlined />
            <q-input v-model="user.password" label="Password" name="password" type="password" outlined />

            <!-- Кнопка входу -->
            <q-btn class="full-width q-mt-lg fredoka" style="background-color: #4CAF50; color: #ffffff;" label="Login" type="submit" />
          </q-form>

          <!-- Посилання на сторінку реєстрації -->
          <div class="q-mt-lg text-center">
            Немає акаунту? <router-link to="/register">Зареєструватися</router-link>
          </div>
        </q-card-section>
      </q-card>
    </q-page>
  </template>

  <script setup>
  import { ref, reactive } from 'vue';
  import { useRouter } from 'vue-router';
  import { Notify } from 'quasar';
  import login from '../firebase/firebase-login';

  const router = useRouter();

  // Об'єкт для зберігання даних користувача
  const user = reactive({
    email: '',
    password: ''
  });

  const form = ref(null);

  // Функція для обробки відправки форми
  const submit = async () => {
    if (form.value.validate()) {
      try {
        await login(user);
        router.push('/app');
```

```
    } catch (error) {
      console.error('Login error:', error);
      // Відображення повідомлення про помилку
      Notify.create({
        message: `Помилка входу: ${error.message}`,
        color: 'negative',
        position: 'bottom'
      });
    }
  }
};
</script>

<style scoped>
/* Контейнер сторінки входу */
.login-page {
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Картка входу */
.login-card {
  max-width: 400px;
  width: 100%;
}

.text-h4 {
  font-size: 1.5rem;
}

.text-primary {
  color: #4CAF50 !important;
}

.full-width {
  width: 100%;
}

.q-pt-xl {
  padding-top: 3rem !important;
```

```
}

.q-pa-md {
  padding: 1.5rem !important;
}

.q-mt-lg {
  margin-top: 1.5rem !important;
}

.fredoka {
  font-family: 'Fredoka One', cursive;
}
</style>
```

firebase-register.js:

```
import { createUserWithEmailAndPassword, updateProfile } from 'firebase/auth';
import { auth } from './index';

// Функція для реєстрації нового користувача
const register = async (user) => {
  try {
    // Створення нового користувача за допомогою email та пароля
    const userCredential = await createUserWithEmailAndPassword(auth, user.email, user.password);

    // Оновлення профілю користувача з ім'ям та прізвищем
    await updateProfile(userCredential.user, {
      displayName: `${user.first_name} ${user.last_name}`
    });

    return userCredential.user;
  } catch (error) {
    // Обробка різних типів помилок
    let errorMessage = '';
    switch (error.code) {
      case 'auth/email-already-in-use':
        errorMessage = 'Цей email вже використовується іншим користувачем.';
        break;
      case 'auth/invalid-email':
        errorMessage = 'Некоректний формат email.';
        break;
      case 'auth/weak-password':
        errorMessage = 'Пароль занадто слабкий. Мінімальна довжина пароля - 6 символів.';
```

```
            break;
          default:
            errorMessage = `Помилка реєстрації: ${error.message}`;
            break;
        }
        throw new Error(errorMessage);
      }
    };

export default register;
```

firebase-login.js:

```
import { signInWithEmailAndPassword } from 'firebase/auth';
    import { auth } from './index';

    // Функція для входу користувача
    const login = async (user) => {
      try {
        // Спроба входу користувача за допомогою email та пароля
        const userCredential = await signInWithEmailAndPassword(auth, user.email, user.password);
        return userCredential.user;
      } catch (error) {
        // Обробка різних типів помилок
        if (error.code === 'auth/user-not-found') {
          throw new Error('Користувача з таким email не знайдено.');
        } else if (error.code === 'auth/wrong-password') {
          throw new Error('Неправильний пароль.');
        } else if (error.code === 'auth/invalid-email') {
          throw new Error('Некоректний email.');
        } else {
          throw new Error('Будь ласка, введіть дійсний email або password!');
        }
      }
    };
export default login;
```

firebase-signout.js:

```
    import { signOut } from 'firebase/auth';
    import { auth } from './index';

    // Функція для виходу користувача з системи
    const signout = async () => {
      try {
        // Спроба виходу користувача
```

```
    await signOut(auth);
   } catch (error) {
    // Якщо виникла помилка, перекидаємо її далі
    throw new Error(error.message);
   }
  };

export default signout;
```

index.js:

```
  import { initializeApp } from "firebase/app"
  import { getAuth, onAuthStateChanged } from "firebase/auth"
  import { LocalStorage } from 'quasar'

  // Конфігурація Firebase
  const firebaseConfig = {
    apiKey: "AIzaSyBqMw9gjeFrTghm2zxN2q9artYGw2lOmlQ",
    authDomain: "vue-ecom-b5c28.firebaseapp.com",
    projectId: "vue-ecom-b5c28",
    storageBucket: "vue-ecom-b5c28.appspot.com",
    messagingSenderId: "771022702743",
    appId: "1:771022702743:web:2dc266ab537a29adaf01d2",
    measurementId: "G-HERBQDCE4Y"
  };

  // Ініціалізація Firebase
  export const app = initializeApp(firebaseConfig);
  // Отримання об'єкту аутентифікації
  export const auth = getAuth(app);

  // Слідкування за зміною стану аутентифікації
  onAuthStateChanged(auth, (user) => {
   if (user) {
    // Якщо користувач авторизований, зберігаємо його дані в LocalStorage
    LocalStorage.set('user', user)
   } else {
    // Якщо користувач вийшов, видаляємо його дані з LocalStorage
    LocalStorage.remove('user')
   }
});
```

AuthLayout.vue:

```
  <template>
    <q-layout view="lHh Lpr lFf">
```

```
            <q-page-container class="pt-0-important bg-auth">
              <q-page>
                <div class="row" style="height: 100vh">
                  <div class="col-12 flex content-center justify-center">
                    <router-view/>
                  </div>
                </div>
              </q-page>
            </q-page-container>
          </q-layout>
        </template>

        <style scoped>
          /* Стиль для фонового зображення на сторінці аутентифікації */
          .bg-auth {
            background-image: url("src/assets/background.jpg");
            background-size: cover;
            background-repeat: no-repeat;
            background-position: center;
          }
</style>
```

MainLayout.vue:

```
        <template>
          <q-layout view="lHh Lpr lFf">
           <q-header elevated>
            <q-toolbar>
             <q-toolbar-title>
              Ecomonitoring app
             </q-toolbar-title>

             <q-btn flat @click="logout">Вийти</q-btn>
            </q-toolbar>
           </q-header>

           <q-page-container>
            <router-view />
           </q-page-container>
          </q-layout>
         </template>

         <script setup>
         import signout from 'src/firebase/firebase-signout'
```

```
        import { useRouter } from 'vue-router'

        const router = useRouter()

        // Функція для виходу користувача
        const logout = () => {
         signout().then(() => {
          // Після успішного виходу перенаправляємо на сторінку входу
          router.push('/login')
         })
        }
</script>
```

Додаток Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

Додаток В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_ Машталер.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Машталер.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
EcoMonitoringApp.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Mashtaler.D.S_EcoMonitoring App.ppt	Презентація кваліфікаційної роботи