ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНІКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Магістр»

«Система збору та моніторингу даних про якість повітря за допомогою сенсорів»

(тема кваліфікаційної роботи українською мовою)

«Air quality monitoring and data collection system using sensors»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Мазур Олександр Олександрович
(прізвище, ім'я, по-батькові здобувача)

Керівник д.т.н., професор, Казакова Н.Ф. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., Начальник ЦІТ ОНЕУ, Домаскін О.М.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ___ від ___ _____ 2024 р.

Завідувачка кафедри

_____ _____
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № ___ від _____ 2024 р.

Оцінка _____ / _____ / _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____
(підпис) (прізвище,ім'я)

Одеса 2024

АНОТАЦІЯ

Кваліфікаційна робота на тему «Система збору та моніторингу даних про якість повітря за допомогою сенсорів» спрямована на розробку сучасної інформаційної системи, яка дозволяє оперативно отримувати, обробляти та візуалізувати дані про рівень забруднення повітря.

Основною метою є створення зручного та функціонального інструменту для інформування населення про якість повітря в реальному часі.

У роботі досліджено методи розрахунку індексів якості повітря (US, UK, EU), особливості забруднювачів та технології для інтерактивної візуалізації. Система побудована на основі сучасного технологічного стека, включаючи PHP (Laravel), Vue.js, OpenStreetMap та Telegram API.

Результатом є функціональна система, яка охоплює понад 400 міст України, має інтерактивну мапу та Telegram чат-бот. Робота підкреслює перспективи розвитку та інтеграції системи для більш широкого використання.

ANNOTATION

The thesis titled "Air Quality Data Collection and Monitoring System Using Sensors" focuses on developing a modern information system for real-time air quality monitoring.

The primary goal is to create a user-friendly and functional tool for providing real-time air quality data to the public. The study explores air quality index calculation methods (US, UK, EU), pollutant characteristics, and technologies for interactive data visualization.

The system is built using a modern technology stack, including PHP (Laravel), Vue.js, OpenStreetMap, and Telegram API.

The result is a functional system covering over 400 cities in Ukraine, featuring an interactive map and a Telegram chatbot. The work highlights the system's development prospects and integration potential for broader application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, ТЕРМІНІВ, УМОВНИХ ПОЗНАЧЕНЬ

Скорочення

CAMS – Copernicus Atmosphere Monitoring Service – служба моніторингу атмосфери Copernicus.

ВООЗ – всесвітня організація охорони здоров'я.

ЛОС – летючі органічні сполуки.

AQI – Air Quality Index – індекс якості повітря.

IoT – Internet of Things – інтернет речей.

DEFRA – Department for Environment, Food & Rural Affairs, Британський департамент охорони довкілля, їжі та сільського господарства.

JVM – Java Virtual Machine – віртуальна Java-машина – середовище, в якому виконується скомпільований байт-код.

БД – база даних.

СУБД – системи управління баз даних.

ERD – Entity-Relationship Diagram.

TLS – Transport Layer Security – криптографічний протокол, який забезпечує захист даних під час передачі між клієнтом (наприклад, браузером) і сервером. Є покращеною версією SSL.

SSL – Secure socket layer – криптографічний протокол, який забезпечує захист даних під час передачі між клієнтом (наприклад, браузером) і сервером. SSL більше не використовується в сучасних системах і був замінений його покращеною версією – TLS. Проте термін "SSL" часто використовується як загальна назва для обох протоколів.

SSH – Secure Shell – це криптографічний мережевий протокол, який забезпечує захищений доступ до віддалених систем і серверів.

HTTP – HyperText Transfer Protocol – це протокол передачі даних, який використовується для обміну інформацією між веб-браузером (клієнтом) і

сервером.

HTTPS – HTTP Secure – це розширена версія HTTP, яка забезпечує безпечну передачу даних через використання шифрування за допомогою протоколів SSL або його новішої версії TLS.

OpenSSH – Open Secure Shell – набір інструментів для безпечного доступу до віддалених систем через мережу, який реалізує протокол SSH (Secure Shell).

MVC – Model-View-Controller – архітектурний шаблон проєктування, який розділяє програму на три логічні компоненти: Модель (Model), Подання (View) та Контролер (Controller). Кожна з цих компонент має свої чіткі завдання, що забезпечує організованість, зручність у розробці та підтримці коду.

UI – User Interface – інтерфейс користувача

API – Application Programming Interface – інтерфейс програмування додатків, який визначає набір правил, протоколів та інструментів для взаємодії між різними програмними компонентами.

SPA – Single Page Application – односторінковий веб-додаток, який динамічно завантажує контент і оновлює лише необхідні частини сторінки без перезавантаження всієї сторінки.

ORM – Object-Relational Mapping – це технологія, яка дозволяє взаємодіяти з реляційними БД через об’єктно-орієнтовані структури у програмному коді.

VPS – Virtual Private Server – віртуальний приватний сервер, який працює як окремий фізичний сервер, але розміщується на одній фізичній машині разом з іншими VPS.

ОС – операційна система.

Терміни

Агрегація – це процес об'єднання або зведення даних з кількох джерел або елементів в єдину узагальнену інформацію.

Кросплатформеність – це здатність програмного забезпечення, апаратного

забезпечення або середовища виконання працювати на різних платформах без змін у його вихідному коді чи архітектурі.

Файрвол (Firewall) – це програмний або апаратний засіб захисту мережі, який контролює вхідний і вихідний трафік на основі заданих правил безпеки.

Бізнес-логіка — це "серце" будь-якої системи, яке забезпечує реалізацію ключових функцій, необхідних для досягнення бізнес-цілей.

Вебхук (Webhook) – це механізм, який дозволяє автоматично передавати дані або повідомлення від однієї системи до іншої в реальному часі. Вебхук працює як "зворотний виклик" (callback): коли в системі відбувається певна подія, вона надсилає HTTP-запит (зазвичай POST-запит) на заздалегідь задану URL-адресу.

Фреймворк – це потужний інструмент для розробки програмного забезпечення, який значно спрощує і прискорює створення проєктів. Завдяки готовим рішенням, структурованості та безпеці, фреймворки є вибором номер один для багатьох розробників у світі.

ВСТУП

Проблема забруднення навколишнього середовища, зокрема якості повітря, є однією з найактуальніших у сучасному світі. Забруднення повітря значною мірою впливає на здоров'я людей, екосистеми та клімат. За даними Всесвітньої організації охорони здоров'я (ВООЗ), станом на квітень 2022 року, понад 90% населення світу живе в умовах забрудненого повітря, що є основною причиною понад 7 мільйонів смертей щорічно [1]. Ці цифри підкреслюють важливість пошуку ефективних методів контролю та зниження рівня забруднення для збереження здоров'я населення та навколишнього середовища. В умовах глобалізації і урбанізації проблема стає ще більш гострою, адже забруднення повітря не знає меж і може вражати великі міста, а також віддалені регіони.

Системи моніторингу якості повітря відіграють ключову роль у вирішенні цієї проблеми, оскільки вони забезпечують можливість швидко отримувати дані про рівень забруднення, що дозволяє своєчасно вживати необхідних заходів для поліпшення екологічної ситуації. В останні роки значно поширилися технології сенсорів, які стали доступнішими за ціною і легшими для придбання. Ці технології дозволяють вимірювати концентрацію забруднюючих речовин, таких як діоксид азоту (NO_2), частки $PM_{2.5}$ і PM_{10}, озон (O_3) та інші, а також створювати системи для автоматичного збору та аналізу цих даних в реальному часі.

Забруднення повітря є не лише екологічною проблемою, але й економічною та соціальною, оскільки погіршення якості повітря безпосередньо впливає на продуктивність праці, рівень захворюваності і тривалість життя населення. Оскільки ситуація з забрудненням в Україні, особливо в умовах російської військової агресії, що призводить в тому числі до забруднення великих територій під час бойових дій, стає все більш складною, виникає потреба в створенні інноваційних систем моніторингу, які можуть надавати точну інформацію в реальному

часі.

Метою цієї роботи є розробка системи збору та моніторингу даних про якість повітря за допомогою сенсорів, яка дозволить отримувати дані про рівень забруднення в Україні, агрегувати їх з різних джерел та представляти у вигляді доступної мапи з показниками забруднювачів і відповідними індексами якості повітря. Завданням роботи є створення програмного забезпечення для збору, обробки і відображення цих даних, а також інтеграція з існуючими API для отримання актуальної інформації.

Об'єктом дослідження є система збору і моніторингу даних про якість повітря, яка інтегрує інформацію з різних джерел, зокрема відкритих API та сенсорних систем. Предметом дослідження є методи обробки і агрегації даних, а також технології, які використовуються для візуалізації цих даних і створення інтерактивних мап, що надають користувачам доступ до актуальної інформації про стан повітря в реальному часі.

Для досягнення мети роботи застосовуються сучасні методи збору даних через API, таких як OpenWeather чи Weather API, а також алгоритми агрегації даних з різних джерел. Для візуалізації отриманих даних на інтерактивній мапі, слід розробити веб-застосунок. Окрім цього, передбачається реалізація чат-бота, який дозволить користувачам отримувати сповіщення про якість повітря в найближчому до їхнього місцезнаходження місті на основі їхніх географічних координат.

Розробка цієї системи може стати важливим кроком у боротьбі із забрудненням повітря в Україні, оскільки надає користувачам простий і доступний інструмент для моніторингу рівня забруднення повітря та прийняття рішень для покращення якості довкілля. Проєкт має важливе наукове та практичне значення, оскільки дозволяє поєднати сучасні технології з екологічними проблемами і допомогти в покращенні здоров'я населення та екологічної ситуації в країні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Розгляд актуальності задачі створення системи збору та моніторингу даних про якість повітря за допомогою сенсорів

Забруднення повітря є однією з найсерйозніших глобальних екологічних проблем, що має безпосередній вплив на здоров'я людей, довкілля та економіку. За даними ВООЗ, понад 90% населення світу дихає забрудненим повітрям, що є причиною понад 7 мільйонів смертей щорічно через захворювання, пов'язані з забрудненням повітря, такі як хвороби дихальних шляхів, серцево-судинні захворювання та рак легень [1]. Проблема забруднення повітря набула особливої актуальності на фоні інтенсивної урбанізації, зростання автомобільного руху, промислового виробництва та використання неекологічних джерел енергії.

Системи моніторингу якості повітря є важливим інструментом для своєчасного реагування на зміни рівнів забруднення та запобігання негативним наслідкам для здоров’я людини і навколишнього середовища. Завдяки технологічним досягненням, зокрема в галузі сенсорних технологій, стало можливим створення більш точних і доступних систем для моніторингу якості повітря в реальному часі. Ці системи дозволяють виявляти забруднення на різних рівнях – від локальних точок до регіональних і глобальних масштабів.

На світовому рівні важливість моніторингу якості повітря підтверджується численними міжнародними ініціативами. Так, в рамках Європейського Союзу вже багато років існує система моніторингу якості повітря, що включає сотні станцій, розташованих по всьому континенту. Програма CAMS надає дані про стан атмосферного повітря в Європі, що дозволяє прогнозувати забруднення і вживати заходів щодо поліпшення ситуації. Схожі програми існують у США (наприклад, AirNow) та Китаї, де проблему забруднення повітря також вирішують на

державному рівні через системи моніторингу.

У більшості країн, де якість повітря є серйозною проблемою, створення таких систем моніторингу має на меті не лише контроль поточної ситуації, але й прогнозування рівнів забруднення та формування стратегії по їх зниженню. Проте значною проблемою для деяких країн є обмеження у фінансуванні, оскільки масове встановлення стаціонарних станцій для моніторингу якості повітря потребує значних витрат.

В Україні проблема забруднення повітря також є досить актуальною. Згідно з даними ВООЗ, велика частка українців живуть в умовах, де рівень забруднення повітря перевищує гранично допустимі норми. Зокрема, найбільше забруднене повітря в великих промислових містах, таких як Київ, Харків, Львів, Дніпро, Кривий Ріг та інші. Основними забруднювачами є промисловість, автотранспорт, енергетичний сектор і побутові викиди, бойові дії.

На сьогоднішній день в Україні існує кілька станцій моніторингу якості повітря, що належать до державних та приватних установ. Вони займаються збором даних про рівень забруднення, проте їх кількість обмежена і не охоплює всі регіони країни. Більшість станцій розташовані у великих містах, в той час як сільські регіони залишаються без належного моніторингу. Крім того, важливим аспектом є недостатня актуальність і доступність даних для широкого загалу.

У зв'язку з цим, потреба в створенні більш ефективних та доступних систем моніторингу якості повітря за допомогою сенсорів стає надзвичайно актуальною. Сенсори, що використовуються для моніторингу якості повітря, стають дедалі доступнішими за ціною, що дозволяє значно розширити їх застосування не тільки в межах великих міст, але й в віддалених районах. Це дозволяє оперативно отримувати точну інформацію про рівень забруднення на різних територіях та своєчасно реагувати на небезпеку для здоров'я.

Впровадження систем збору та моніторингу даних про якість повітря за

допомогою сенсорів дозволить забезпечити більш точну та оперативну інформацію про забруднення в режимі реального часу. Такі системи можуть стати важливим інструментом у вирішенні екологічних проблем, оскільки вони не лише дозволяють відстежувати поточний стан, а й надають дані для аналізу органам влади, що дозволить їм своєчасно реагувати на ситуацію з погіршенням якості повітря.

До переваг використання сенсорів для моніторингу якості повітря можна віднести:

1) доступність і низька вартість сенсорів робить їх доступнішими для широкого використання, що дозволяє збільшити кількість точок збору даних;

2) дані можуть бути отримані оперативно і точно, в реальному часі, що дозволяє своєчасно реагувати на підвищення рівня забруднення;

3) можливість масштабування системи в будь-якій частині країни, навіть у віддалених районах.

Однак є й певні труднощі:

1) багато сенсорів можуть давати неточні дані при високих рівнях забруднення або в умовах низьких температур;

2) для забезпечення точності результатів необхідно агрегувати дані з різних джерел і коригувати їх для отримання найбільш достовірної інформації.

1.2 Опис основних забруднювачів повітря

Забруднення повітря є результатом викидів різних хімічних сполук і часток в атмосферу, які можуть мати серйозні наслідки для здоров'я людини та довкілля. Основні забруднювачі повітря включають тверді мікрочастинки ($PM_{2.5}$, PM_{10}), діоксид азоту (NO_2), діоксид сірки (SO_2), окис вуглецю (CO) та озон (O_3). Звичайно, існують також і інші забруднювачі, такі як ЛОС, але більшість провідних індексів

якості повітря, такі як Європейський чи Британський індекс якості повітря, не враховують їх в своїх розрахунках, тому ми не будемо їх розглядати. Кожен з цих забруднювачів має свої джерела та впливає на здоров'я людини та навколишнє середовище по-різному.

Розмір часток безпосередньо пов'язаний з їхнім потенціалом викликати проблеми зі здоров'ям. Частки менші за 10 мікрометрів можуть потрапляти в організм через дихальні шляхи, при цьому найменші частки (менше ніж 2.5 мікрометра) становлять найбільшу небезпеку, оскільки можуть проникати глибоко в легені, а деякі навіть потрапляти в кровотік.

Вплив таких часток може впливати як на легені, так і на серце. Численні наукові дослідження пов'язують забруднення частками з різними проблемами зі здоров'ям, такими як:

- передчасна смерть у людей із серцево-судинними або легеневими захворюваннями;
- інфаркт міокарда;
- порушення серцевого ритму;
- загострення астми;
- зниження функції легенів;
- посилення респіраторних симптомів, таких як подразнення дихальних шляхів, кашель або утруднене дихання.

Окрім короткострокових ефектів, які можна спостерігати після щоденного впливу забруднення частками (на що вказує індекс якості повітря), довготривалий вплив (що триває місяцями та роками) може призвести до розвитку серцево-судинних та респіраторних захворювань, таких як атеросклероз та астма, впливів на нервову систему (наприклад, когнітивні порушення), раку легень та передчасної смерті [2].

Джерелами мікрочастинок $PM_{2.5}$ і PM_{10} є промислові викиди,

автотранспорт, сільське господарство, спалювання палива в побуті, а також природні фактори, такі як пилові бурі та лісові пожежі.

Окис вуглецю (CO) – це безбарвний газ без запаху, який утворюється під час неповного згоряння палива. Джерела CO включають автотранспорт, побутові прилади, а також промислові викиди. CO є особливо небезпечним, оскільки він знижує здатність крові переносити кисень, що може призвести до отруєнь, головного болю, запаморочення, а у великих кількостях – навіть до летальних наслідків.

Надзвичайно високі рівні CO навряд чи спостерігаються на відкритому повітрі. Проте, коли рівень CO підвищується на вулиці, це може бути особливо небезпечно для людей із серцевими захворюваннями. У таких осіб вже є знижена здатність доставляти кисень до серця в ситуаціях, коли воно потребує більшої кількості кисню, ніж зазвичай. Вони особливо вразливі до впливу CO під час фізичних навантажень або стресових ситуацій. У таких випадках навіть короткочасний вплив підвищеного рівня CO може призвести до зниження кисню, що постачається до серця, і викликати біль у грудях, відомий як стенокардія [2].

Діоксид азоту (NO_2) є одним із основних забруднювачів, що утворюється в результаті згоряння палива в двигунах автомобілів, на теплових електростанціях, в промисловості та при спалюванні біомаси. Він може викликати або погіршувати захворювання дихальних шляхів, такі як астма і хронічний бронхіт, а також сприяє утворенню озону на рівні землі, що ще більше посилює забруднення, а також NO_2 має значний вплив на клімат, оскільки він сприяє утворенню кислотних дощів, які можуть пошкоджувати екосистеми та будівлі.

Вплив діоксиду азоту за короткий період часу (1-5 годин) може погіршити стан людей, що хворіють на респіраторні захворювання, зокрема астму, а також може призвести до проявів респіраторних симптомів, таких як кашель, свистяче дихання, утруднене дихання, госпіталізації та звернень до відділень невідкладної

допомоги [2].

Діоксид сірки (SO_2) – це газ, що утворюється під час спалювання вугілля та нафти, а також в результаті промислових процесів, зокрема при виробництві металів. SO_2 є основним забруднювачем, що сприяє утворенню кислотних дощів, які негативно впливають на водні ресурси, ґрунти і рослинність.

Короткостроковий вплив діоксиду сірки (1-6 годин) може спричиняти респіраторні проблеми, зокрема утруднене дихання та посилення симптомів астми. Ці ефекти є особливо небезпечними для людей з астмою, особливо під час глибокого дихання, наприклад, під час фізичних навантажень чи гри. Крім того, короткочасне перебування в умовах підвищеного рівня SO_2 було також пов'язане з ростом кількості звернень до відділень екстреної допомоги та госпіталізацій через респіраторні захворювання, особливо серед осіб груп ризику, таких як діти, літні люди та пацієнти з астмою [2].

Озон (O_3) – це газ, який є корисним на великій висоті в атмосфері, де він створює захисний шар від ультрафіолетового випромінювання. Проте на рівні землі озон є шкідливим забруднювачем, оскільки він утворюється в результаті хімічних реакцій між викидами діоксиду азоту (NO_2) та летючими органічними сполуками під дією сонячного світла. Озон на рівні землі може викликати подразнення дихальних шляхів, посилювати симптоми астми та інших респіраторних захворювань.

1.3 Дослідження впливу якості повітря на повсякденне життя та стан здоров'я людей

Розглянуті у попередньому підрозділі основні забруднювачі повітря вказують на те, що якість повітря, яким ми дихаємо може мати як короткострокові, так і довгострокові наслідки для здоров'я.

Особливо вразливими до забруднення повітря є певні групи людей, серед

яких діти, люди похилого віку, вагітні жінки та люди з хронічними захворюваннями. У дітей організм ще не повністю розвинений, і вони більш чутливі до впливу забруднення. Вони мають більш високі рівні дихальної активності, що збільшує ризик впливу токсичних часток на легені та інші органи [16].

Літні люди часто мають ослаблений імунітет, хронічні хвороби, що робить їх більш уразливими до різноманітних ефектів забруднення повітря, таких як респіраторні проблеми і серцево-судинні захворювання. Крім того, вагітні жінки також перебувають під підвищеним ризиком, оскільки погіршення якості повітря може впливати на розвиток плоду, зокрема, підвищуючи ризик передчасних пологів, низької ваги при народженні та порушень розвитку дитини.

Короткострокові наслідки від забруднення повітря можуть проявлятися у вигляді подразнення очей, носа, горла, а також у вигляді кашлю, утрудненого дихання та посилення симптомів астми або інших респіраторних хвороб. В умовах високого забруднення повітря навіть здорові люди можуть відчувати дискомфорт і погіршення загального самопочуття.

Довгостроковий вплив забруднення повітря може призвести до розвитку хронічних захворювань, таких як бронхіти, емфізема, хронічна обструктивна хвороба легень (ХОЗЛ), а також підвищити ризик розвитку раку легень. Крім того, довготривале забруднення повітря значно підвищує ймовірність розвитку серцево-судинних захворювань, інсультів та інших важких станів.

Забруднення повітря не лише погіршує фізичний стан людини, але й значно знижує якість її життя. Люди, які постійно перебувають у зонах з високим рівнем забруднення повітря, можуть відчувати постійну втому, проблеми з концентрацією, зниження фізичної активності та обмеження у повсякденній діяльності. Оскільки люди з проблемами дихальних шляхів або серця мають труднощі з виконанням навіть легких фізичних вправ, це суттєво обмежує їхній соціальний і професійний розвиток [17].

Вплив забруднення також спостерігається на психічному здоров'ї, оскільки тривале перебування в умовах забрудненого повітря може викликати стрес, тривожність та депресію. Це може бути пов'язано з постійними проблемами зі здоров'ям, труднощами в повсякденному житті та відсутністю доступу до чистого повітря.

1.4 Аналіз методів моніторингу якості повітря

Моніторинг якості повітря є важливим інструментом для оцінки рівня забруднення повітря та подальшої оцінки впливу забруднювачів на здоров'я людини та навколишнє середовище. Існують різні методи моніторингу, які використовують різні технології для збору даних про забруднення. Ці методи можна поділити на традиційні, що використовують великі стаціонарні станції, та сучасні, що використовують сенсорні технології та мобільні пристрої.

Традиційні методи моніторингу якості повітря зазвичай включають використання стаціонарних вимірювальних станцій, оснащених складними пристроями для визначення концентрацій забруднюючих речовин у повітрі. Для роботи таких станцій зазвичай потрібні висококваліфіковані спеціалісти. Такі станції вимірюють рівні різних забруднювачів, таких як $PM_{2.5}$, PM_{10}, NO_2, CO, SO_2 та інші, і передають ці дані в центральні системи для подальшого аналізу.

До переваг використання великих професійних станцій вимірювання якості повітря можна віднести:

1) високу точність даних, оскільки використовуються професійні вимірювальні прилади, сертифіковані відповідно до міжнародних стандартів;

2) можливість здійснювати глибокий аналіз даних за допомогою спеціалізованих лабораторій;

3) такі станції є більш надійними, що забезпечує довготривалу експлуатацію та стабільність вимірів.

Проте, вони мають й свої недоліки:

1) висока вартість установки та обслуговування вимірювальних станцій;

2) обмежена кількість станцій, що покривають лише великі міста та промислові зони, що обмежує точність та повноту даних для сільських та віддалених регіонів;

3) затримка в отриманні даних через необхідність їх обробки в лабораторіях;

4) працювати з ними зазвичай можуть тільки висококваліфіковані спеціалісти, що значно здорожчує їх використання.

З розвитком сенсорних технологій з'явилися нові методи моніторингу, які використовують компактні, доступні та мобільні сенсори для вимірювання якості повітря. Сенсори також можуть вимірювати концентрацію основних забруднювачів повітря, таких як $PM_{2.5}$, PM_{10}, CO, NO_2, O_3 та інших. Вони можуть бути встановлені на вулицях, у транспортних засобах та мобільних пристроях.

Перевагами використання сенсорних технологій для вимірювання якості повітря є:

1) низька вартість та компактність, що дозволяє масово впроваджувати ці технології в різних районах;

2) можливість збору даних в реальному часі, що дозволяє отримувати оперативну інформацію про рівень забруднення;

3) широке застосування у мобільних додатках і особистих моніторах якості повітря.

Проте, вони, звичайно, мають й свої недоліки:

1) низька точність у порівнянні з традиційними методами, особливо при використанні дешевих побутових сенсорів;

2) вплив зовнішніх факторів (температура, вологість) на точність вимірювань;

3) необхідність агрегації даних з численних сенсорів для отримання більш точних результатів.

Окрім реального моніторингу, існують методи прогнозування рівня забруднення повітря. Вони включають в себе математичні моделі, які враховують різні фактори, такі як метеорологічні умови (температура, вітер, вологість), інтенсивність руху транспорту та інші параметри, що можуть вплинути на рівень забруднення. Такі моделі можуть прогнозувати рівень забруднення на декілька годин або навіть діб наперед, що допомагає органам влади приймати своєчасні рішення щодо заходів, необхідних для покращення ситуації.

Кожен з методів моніторингу має свої переваги та обмеження. Традиційні методи є більш точними і надійними, але їх застосування обмежене через високу вартість і потребу в інфраструктурі. Сенсорні технології дозволяють збирати дані в реальному часі та на великій кількості точок, але їх точність може бути нижчою. Прогнозування якості повітря є важливим доповненням до реального моніторингу, оскільки воно дозволяє організувати превентивні заходи, однак його точність залежить від багатьох факторів.

Моніторинг якості повітря є важливим елементом для контролю екологічної ситуації, оцінки впливу забруднення на здоров'я людей та ухвалення ефективних управлінських рішень. Вибір методу моніторингу залежить від цілей, доступних ресурсів та вимог до точності даних. Важливо також комбінувати різні методи для досягнення найбільш повної та точної картини забруднення повітря.

1.5 Дослідження систем сповіщень та візуалізації якості повітря

Моніторинг якості повітря є важливою складовою частиною забезпечення екологічної безпеки та здоров'я населення. Однак отримання даних про рівень забруднення само по собі не є достатнім – необхідно надавати ці дані користувачам у зручному та зрозумілому вигляді, а також забезпечувати оперативне інформування про можливі загрози. У зв'язку з цим важливим елементом систем моніторингу є системи сповіщень та візуалізації, які дозволяють не тільки показувати стан повітря в реальному часі, але й попереджати користувачів про підвищення рівня забруднення.

Системи сповіщень дозволяють швидко інформувати користувачів про поточний рівень забруднення та потенційні загрози для здоров'я, що є важливим інструментом для забезпечення оперативної реакції з боку органів влади та населення.

Сповіщення можуть бути різного типу, залежно від каналів комунікації та формату:

1) багато мобільних застосунків використовують push-сповіщення для інформування користувачів про зміни в якості повітря. Ці сповіщення можуть бути налаштовані для певних інтервалів часу, а також для повідомлення про перевищення конкретних рівнів забруднення, що становлять загрозу для здоров'я. Проте, очевидно, для цього методу необхідно додатково створити мобільний застосунок, що вимагає значно більше зусиль, часу та грошей;

2) для організацій, що потребують регулярного моніторингу якості повітря, підійдуть й сповіщення на електронну пошту. Можна раз в певний проміжок часу використовувати email-розсилки, що надають зведену інформацію про поточний стан забруднення в різних регіонах;

3) чат-боти для месенджерів, такі як Telegram або WhatsApp також є популярним інструментом для сповіщення про забруднення повітря. Вони дозволяють користувачам швидко отримувати актуальну інформацію за допомогою простих запитів або автоматичних сповіщень. А оскільки використання месенджерів сьогодні є дуже масовим явищем, це може бути найбільш дешевим та всеохоплюючим методом сповіщення.

Сповіщення зазвичай використовуються у випадках:

1) підвищення рівня забруднення понад допустимі норми;

2) зміни рівня забруднення в реальному часі (для прийняття подальших рішень, наприклад, обмеження активностей на відкритому повітрі);

3) прогнозування погіршення якості повітря на основі даних з моделей прогнозування.

Ці сповіщення можуть бути адаптовані для різних груп користувачів, таких як діти, люди похилого віку, пацієнти з хронічними захворюваннями, що дозволить їм вчасно вжити заходів для захисту здоров'я.

Візуалізація даних є важливою частиною системи моніторингу, оскільки вона дозволяє користувачам швидко сприймати інформацію та оцінювати рівень забруднення у конкретному регіоні. Найбільш поширеними методами візуалізації є інтерактивні мапи та графіки, що дозволяють детально відображати інформацію про забруднення повітря.

Системи сповіщень та візуалізації якості повітря є незамінними інструментами для інформування громадян про рівень забруднення та допомагають оперативно реагувати на екологічні загрози. Вони дозволяють підвищити рівень обізнаності населення, сприяють формуванню здорових звичок і зменшенню ризиків для здоров'я. Однак для досягнення максимальної ефективності таких систем необхідно постійно вдосконалювати технології збору даних, забезпечувати точність прогнозів та забезпечувати доступність інформації для широкої аудиторії.

1.6 Розгляд економічних аспектів впровадження систем збору та моніторингу даних про якість повітря

Впровадження систем збору та моніторингу даних про якість повітря має важливі економічні наслідки, як для держави, так і для бізнесу та громадян. Витрати на впровадження та обслуговування таких систем, безумовно, є важливими, але вони повинні розглядатися також в контексті довгострокових переваг, які з'являються внаслідок поліпшення стану навколишнього середовища та зниження ризиків для здоров'я населення. Вартість таких систем з часом окупається завдяки зменшенню витрат на лікування захворювань, пов'язаних із забрудненням повітря, зниженню рівня передчасної смертності, а також підвищенню загальної якості життя [18].

Одним з основних економічних аспектів впровадження систем моніторингу якості повітря є витрати на установку та обслуговування обладнання. До таких витрат відносяться:

1) закупівля та інсталяція стаціонарних станцій моніторингу, що включає вартість обладнання, установку датчиків та необхідну інфраструктуру для підключення до централізованих систем збору даних;

2) для більш широкого покриття можна використовувати мобільні сенсори та персональні пристрої для моніторингу, що може значно здешевити процес збору даних;

3) інфраструктура для збору, обробки та аналізу даних, що включає в себе створення централізованих баз даних, серверних потужностей для обробки та аналізу даних, а також програмне забезпечення для відображення інформації;

4) регулярне технічне обслуговування сенсорів і станцій необхідне для забезпечення точності вимірювань та безперервної роботи системи.

Хоча початкові витрати на створення та підтримку систем моніторингу

можуть бути високими, економічні вигоди від їх впровадження значно перевищують витрати. Зокрема, зниження рівня забруднення повітря має безпосередній вплив на економіку, оскільки зменшується кількість хвороб, пов'язаних із забрудненням, що, в свою чергу, знижує витрати на медичне обслуговування та лікарняні виплати.

Основними вигодами від покращення якості повітря є:

1) зниження медичних витрат, оскільки забруднення повітря є причиною численних захворювань, таких як астма, хронічний бронхіт, інфаркти, інсульти та рак легень. Зниження рівня забруднення веде до зменшення витрат на лікування цих захворювань;

2) зниження рівня забруднення сприяє зменшенню числа передчасних смертей, що також має економічний ефект, оскільки люди, які помирають від хвороб, спричинених забрудненням, більше не можуть працювати, що позначається на економічній активності;

3) забруднене повітря негативно впливає на загальне самопочуття людей, що веде до зниження працездатності. Покращення якості повітря може збільшити продуктивність праці, знижуючи кількість днів, пропущених через хвороби;

4) чисте повітря підвищує привабливість міст і регіонів для туристів і інвесторів. Вища якість життя приваблює фахівців з інших країн, що може позитивно вплинути на економічний розвиток регіону.

Впровадження систем моніторингу якості повітря є не лише необхідним заходом для покращення екологічної ситуації, але й економічно вигідним для держави та бізнесу [18]. Початкові витрати на створення та підтримку таких систем окупаються завдяки зменшенню витрат на медичне обслуговування, підвищенню продуктивності праці та поліпшенню якості життя. Крім того, ці системи сприяють розвитку нових технологій, інвестицій у зелені ініціативи та покращенню

інфраструктури міст, що позитивно позначається на економічному розвитку в довгостроковій перспективі. Звичайно, сама лише система моніторингу та візуалізації якості повітря не призведе до змін, проте вона може стати першим кроком на шляху до таких змін, оскільки дає змогу чітко зрозуміти проблему та її масштаби в кожному конкретному місті.

1.7 Аналіз доступних аналогічних систем моніторингу якості повітря

На разі в Україні функціонує відносно невелика кількість повноцінних систем моніторингу якості повітря – громадська ініціатива EcoCity, некомерційний проєкт приватної комерційної організації ЛУН – ЛУН.Місто AIR, та проєкт громадської організації SaveDnipro – SaveEcoBot.

Кожен з цих проєктів має як свої переваги, так і недоліки. Розглянемо окремо кожен з них.

1.7.1 Аналіз громадської ініціативи EcoCity

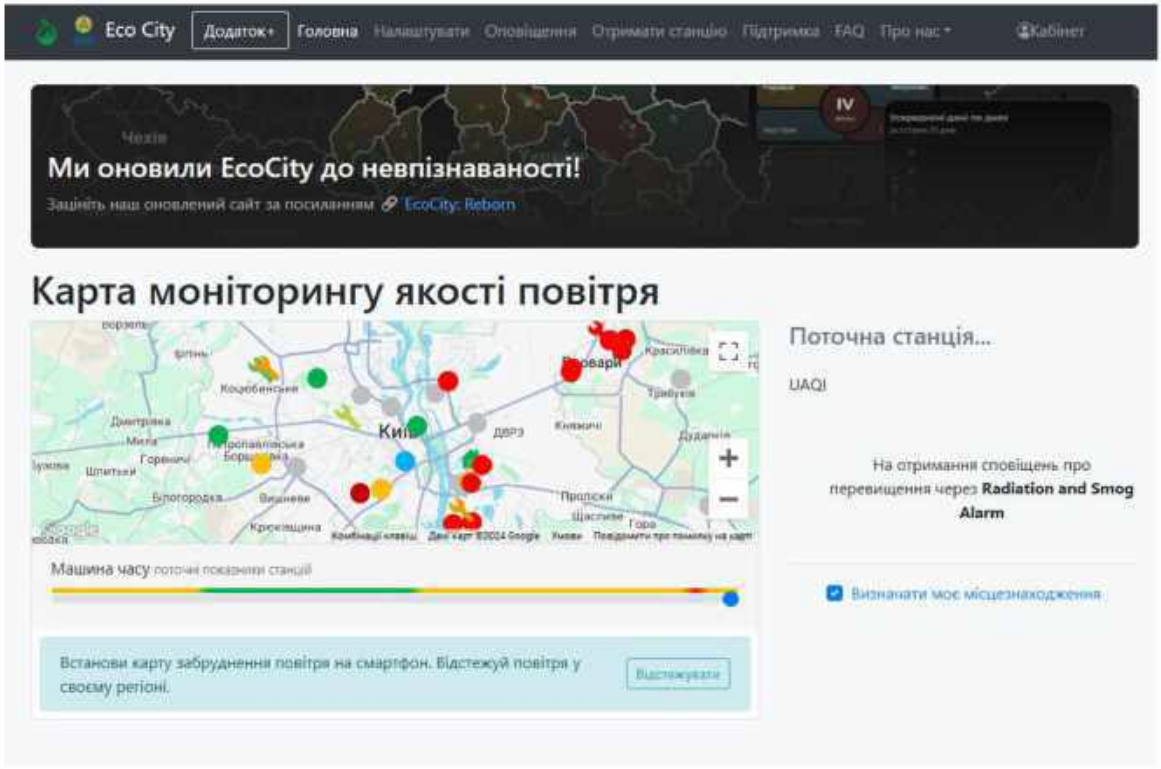

Рисунок 1.1 – Веб-сайт проєкту EcoCity

EcoCity – це соціальний екологічний проєкт (зображений на рис. 1.1), що функціонує як найбільша мережа громадського моніторингу якості повітря в Україні. Реалізований неприбутковою громадською організацією «Фрі Ардуіно», проєкт спрямований на підвищення обізнаності громадян щодо стану повітря та сприяння прийняттю рішень для зменшення забруднення [3].

Основні функції сайту EcoCity наступні:

1) інтерактивна мапа, що відображає поточні показники якості повітря в різних регіонах України. Користувачі можуть відстежувати стан повітря у своєму регіоні та отримувати сповіщення про перевищення допустимих норм;

2) користувачі можуть зареєструвати особистий кабінет для доступу до архівних даних з їхніх станцій моніторингу. Це дозволяє аналізувати динаміку змін якості повітря та отримувати детальну інформацію про забруднення;

3) процес придбання та налаштування станції моніторингу детально описаний на сайті, що дозволяє користувачам самостійно встановлювати пристрої для збору даних про якість повітря;

4) розділи з часто задаваними питаннями та інформацією про підтримку, Підтримка та FAQ, допомагають користувачам вирішувати технічні питання та отримувати необхідну допомогу.

Проєкт Eco-City сприяє розвитку громадського моніторингу якості повітря в Україні, надаючи громадянам інструменти для активного впливу на екологічну ситуацію в країні.

Кількість станцій, а відповідно і кількість покритих міст обмежена. Під'єднані станції, сенсори та інтегровані провайдери даних, які використовуються в проєкті надають показники далеко не по всім, необхідним для підрахунку індексу якості повітря за алгоритмами провідних країн, забруднювачам.

1.7.2 Аналіз проєкту ЛУН – ЛУН.Місто AIR

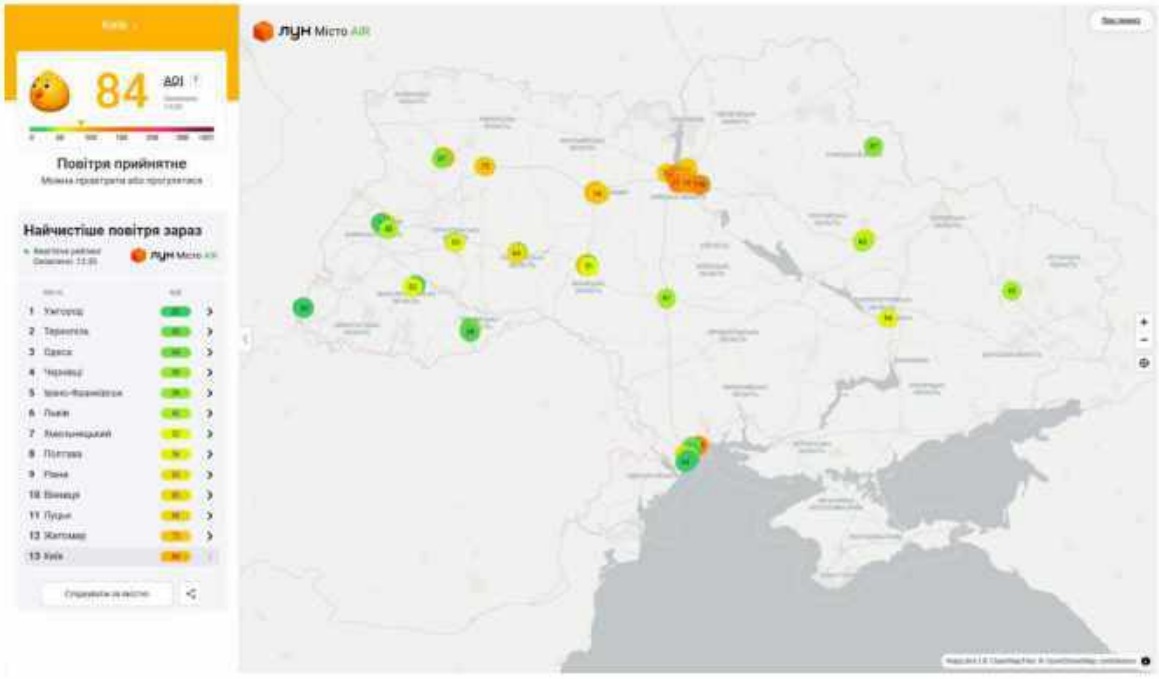

Рисунок 1.2 – Веб-сайт проєкту ЛУН.Місто AIR

Сайт ЛУН Місто Air (зображений на рис. 1.2) пропонує інтерактивну мапу, яка відображає реальний стан якості повітря в українських містах. Користувачі можуть переглядати актуальні показники забруднення повітря по показникам дрібнодисперсних мікрочасток PM_1, $PM_{2.5}$, PM_{10}, а також температуру та вологість. Дані оновлюються в реальному часі, що дозволяє оперативно оцінювати ситуацію та приймати рішення щодо перебування на вулиці [4].

Основними функціями сайту є:

1) інтерактивна мапа відображає рівень забруднення повітря в різних районах міста, використовуючи кольорові індикатори для зручності сприйняття;
2) детальна інформація, яка з'являється при натисканні на конкретну станцію моніторингу користувачі можуть отримати детальні дані про рівень забруднення та погодні умови;

3) сайт надає доступ до історичних даних про якість повітря по обраній станції за останні 48 годин та за останні 30 днів. Це дає можливість відстежувати зміни в забрудненні на конкретних станціях і аналізувати тренди за певний період.

Проєкт ЛУН.Місто Air є некомерційним і спрямований на підвищення обізнаності громадян щодо стану навколишнього середовища та здоров'я. Він реалізований у співпраці з командою Факультету радіофізики, електроніки і комп'ютерних систем КНУ імені Тараса Шевченка та підтримується міськими радами та адміністраціями.

Як і в попередньому випадку, кількість станцій та сенсорів обмежена, що в свою чергу призводить до дуже малої кількості покритих міст – всього тринадцять міст. Також сенсори зчитують тільки дані по мікрочастинкам, тому їх даних недостатньо для обрахунку повноцінних індексів якості повітря.

1.7.3 Аналіз проєкту громадської організації SaveDnipro – SaveEcoBot

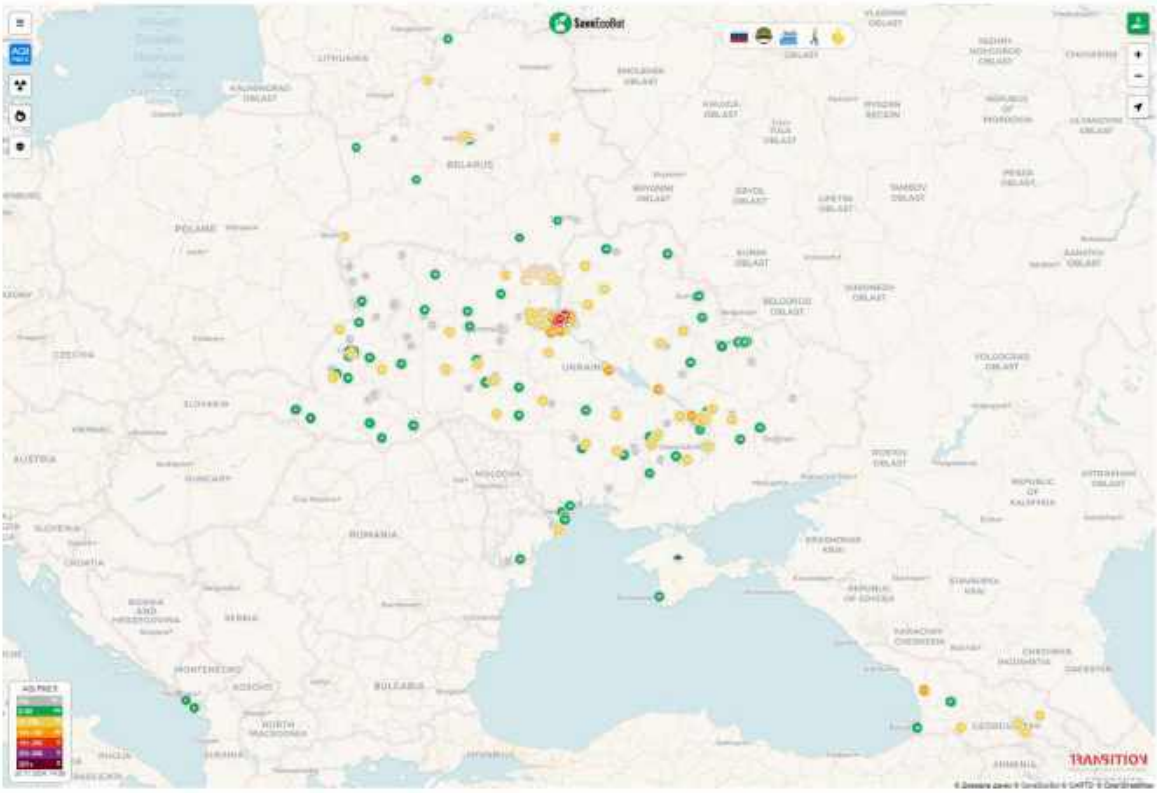

Рисунок 1.3 – Веб-сайт проєкту SaveEcoBot

SaveEcoBot – це перша в Україні екологічна система (зображена на рис. 1.3), яка об'єднує дані про стан довкілля, забруднення, джерела забруднення та інструменти захисту навколишнього середовища. Проєкт надає користувачам доступ до інтерактивних мап якості повітря, радіаційного фону та осередків пожеж, а також інформації про екологічні податки, інспекційні перевірки та екологічні злочини. Користувачі можуть отримувати автоматичні повідомлення про погіршення якості повітря та подавати скарги на забруднення через чат-боти в месенджерах Telegram та Viber [5].

Слід зазначити, що хоча проєкт номінально і пропонує доступ до їх API, насправді ж, жодного доступу отримати не можливо – наразі вони не відповідають на листи.

Проєкт SaveEcoBot покриває найбільшу кількість міст з усіх розглянутих проєктів. Проте, якщо розглядати виключно тему даної дипломної роботи, то даних по різним типам забруднювачів все одно недостатньо для повноцінного обчислювання індексів якості повітря.

1.8 Висновки до розділу 1

У цьому розділі розглянуто актуальність та необхідність створення системи збору та моніторингу даних про якість повітря, а також досліджено основні забруднювачі повітря та їх вплив на здоров'я людини. Погіршення якості повітря має серйозні наслідки для здоров’я населення, спричиняючи різноманітні респіраторні та серцево-судинні захворювання, а також підвищуючи рівень передчасної смертності. Особливо вразливими до забруднення є діти, літні люди та особи з хронічними захворюваннями.

Також проаналізовано методи моніторингу якості повітря, серед яких традиційні стаціонарні станції, мобільні сенсори та новітні технології для збору даних у реальному часі. Кожен з методів має свої переваги і недоліки, але разом вони можуть забезпечити комплексний підхід до моніторингу забруднення повітря.

Системи сповіщень та візуалізації, які включають інтерактивні мапи, чатботи та мобільні застосунки, є важливими інструментами для оперативного інформування населення про стан повітря, що дозволяє швидко реагувати на зміну рівнів забруднення та приймати необхідні заходи для збереження здоров'я.

З економічної точки зору, впровадження систем моніторингу якості повітря не лише сприяє зниженню ризиків для здоров'я населення, але й приносить значні економічні вигоди завдяки зменшенню витрат на медичне обслуговування, підвищенню продуктивності праці та розвитку інфраструктури.

Розглянуті існуючі системи, які функціонують в Україні, дали змогу зрозуміти, який додатковий функціонал можна розробити для покращення системи збору даних та моніторингу якості повітря.

Таким чином, розробка та впровадження ефективної системи моніторингу якості повітря є важливим кроком до покращення екологічної ситуації, підвищення якості життя та збереження здоров'я громадян.

2 ТЕОРЕТИЧНЕ ПІДҐРУНТЯ, ВИБІР ТА ОБҐРУНТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Теоретичні основи моніторингу якості повітря

2.1.1 Принципи роботи систем моніторингу якості повітря.

Системи моніторингу якості повітря використовуються для вимірювання та аналізу рівня забруднення повітря в реальному часі. Такі системи допомагають виявляти перевищення допустимих норм забруднюючих речовин, що можуть мати серйозний вплив на здоров'я людини та навколишнє середовище.

Принципи роботи таких систем базуються на зборі, обробці та аналізі даних, отриманих з різних джерел, зокрема датчиків, сенсорів та інших вимірювальних пристроїв.

Основним завданням системи моніторингу є вимірювання концентрацій різних забруднювачів повітря, таких як:

1) мікрочатинки $PM_{2.5}$ та PM_{10} – дрібні частки пилу, які можуть проникати глибоко в легені і викликати респіраторні захворювання;

2) діоксид азоту (NO_2) – газ, який утворюється внаслідок згоряння палива та є основним забруднювачем в міських районах;

3) оксид вуглецю (CO) – безбарвний газ, що може бути смертельно небезпечним при високих концентраціях;

4) сірчистий газ (SO_2) – результат спалювання вугілля та нафти, який сприяє утворенню кислотних дощів;

5) озон (O_3) – основний компонент фотохімічного забруднення, який є шкідливим для дихальних шляхів.

Для вимірювання концентрації цих забруднювачів використовуються різні типи датчиків, що можуть бути стаціонарними або мобільними. Різні пристрої можуть використовувати різні технології, наприклад, оптичні сенсори, хімічні

сенсори, електрохімічні датчики тощо. Приклад таких сенсорів зображений нижче на рис. 2.1.

Рисунок 2.1 – Приклад сенсору для зчитування вмісту мікрочастинок від громадської організації SaveDnipro

Зібрані дані передаються в центральну систему для подальшої обробки. Система збору даних може включати в себе:

1) стаціонарні станції моніторингу, що встановлюються в різних точках міста або області для безперервного вимірювання концентрацій забруднювачів;

2) мобільні сенсори, які можуть бути встановлені на транспортних засобах або портативних пристроях, що дозволяють отримувати дані з різних місць, зокрема в реальному часі;

3) інтеграція сенсорів з IoT дозволяє створювати «розумні» мережі, в яких пристрої обмінюються даними автоматично, знижуючи необхідність у людському втручанні.

Зібрані дані передаються на сервери через різні канали комунікації, включаючи Wi-Fi, мобільні мережі або бездротові сенсорні мережі.

Отримані дані проходять через систему обробки, яка дозволяє:

1) фільтрувати та очищати дані, видаляючи можливі спотворення або помилки, що можуть виникнути через технічні неполадки або погодні умови;

2) агрегувати та обробляти дані для отримання зведених показників, таких як середнє значення за годину, день або тиждень, а також для аналізу трендів і аномалій;

3) розраховувати індекси якості повітря (AQI) на основі виміряних концентрацій забруднювачів. Індекс AQI дає змогу визначити рівень забруднення та його потенційний вплив на здоров'я населення.

Системи моніторингу можуть використовувати різні алгоритми для аналізу даних, включаючи статистичні методи, методи машинного навчання для прогнозування змін якості повітря та моделювання майбутніх рівнів забруднення.

Ключовим аспектом є візуалізація даних, що дозволяє користувачам, включаючи місцеві органи влади, екологічні організації та громадян, швидко отримувати важливу інформацію про якість повітря. Це може бути реалізовано через:

1) картографічні рішення, які показують рівень забруднення в різних точках міста або регіону в реальному часі;

2) графіки та діаграми: на яких зображено динаміку змін рівня забруднення за певний період часу;

3) системи сповіщень: які надсилають повідомлення в разі перевищення встановлених норм забруднення або за прогнозом погіршення якості повітря.

Це дозволяє швидко реагувати на зміну ситуації та вживати відповідних заходів.

Системи моніторингу якості повітря базуються на зборі, обробці та аналізі даних з різних джерел, що дозволяє своєчасно визначати рівень забруднення та реагувати на загрози для здоров'я населення. Їх ефективність залежить від точності та надійності вимірювальних приладів, якості обробки даних і можливості оперативно інформувати громадськість про зміни в стані повітря.

2.1.2 Огляд основних методів обчислення індексу якості повітря.

Індекс якості повітря (AQI) є важливим показником для оцінки рівня забруднення повітря та його впливу на здоров'я людини. Існують різні методи розрахунку цього індексу, що залежать від використовуваних забруднювачів та методів їх вимірювання. Оскільки рівень забруднення повітря має безпосередній вплив на здоров'я людей, важливо розуміти, як саме формується цей індекс і які забруднювачі враховуються при його обчисленні.

Індекс якості повітря є числовим показником, який відображає рівень забруднення повітря, зокрема концентрацію різних забруднювачів, оглянутих в попередньому підрозділі.

Різні країни можуть використовувати свої методи для розрахунку AQI, але загальна мета однакова – оцінити рівень забруднення повітря і його вплив на здоров'я. Наприклад, в США, Європі та Китаї використовуються різні підходи до обчислення AQI, але загальні принципи залишаються схожими.

Обчислення AQI грунтується на використанні формул, які перераховують концентрацію кожного забруднювача в індекс, що відповідає рівню забруднення

повітря.

Розглянемо кілька методів розрахунку індексу якості повітря.

2.1.2.1 Індекс якості повітря, що використовується в Сполучених Штатах Америки

Індекс якості повітря США (AQI) – це інструмент Агентства з охорони навколишнього середовища (EPA) для інформування про стан зовнішнього повітря та його вплив на здоров'я. Індекс поділений на шість категорій, кожна з яких має своє колірне позначення та відповідає певному діапазону значень. Чим вище значення AQI, тим більший рівень забруднення повітря і тим вищий ризик для здоров'я. Наприклад, значення AQI до 50 вказує на якісне повітря, тоді як показник понад 300 свідчить про небезпечний рівень забруднення [2].

Індекс для забруднювача розраховується за наступним алгоритмом:

1) визначити найвищу концентрацію серед усіх забруднювачів у кожній зоні звітування та скоротіть, як вказано в табл. 2.1 нижче;

Таблиця 2.1 – Скорочення показників забруднювачів

Показник	Правило скорочення
O_3 (ppm)	Обрізати до тисячних (3 цифри після коми)
$PM_{2.5}$ (µg/m^3)	Обрізати до десятих (1 цифра після коми)
PM_{10} (µg/m^3)	Обрізати до цілого числа
CO (ppm)	Обрізати до десятих (1 цифра після коми)
SO_2 (ppb)	Обрізати до цілого числа
NO_2 (ppb)	Обрізати до цілого числа

2) використовуючи приведену нижче табл. 2.2, знайдіть дві контрольні точки, які містять концентрацію;

Таблиця 2.2 – Контрольні точки забруднювачів для розрахунку індексу якості повітря (AQI)

Контрольні точки показників по забруднювачам							AQI[4]
O_3 (ppm) 8 годин	O_3 (ppm) 1 година[1]	$PM_{2.5}$ (µg/m^3)	PM_{10} (µg/m^3)	CO (ppm)	SO_2 (ppb)	NO_2 (ppb)	
0.000 – 0.054	-	0.0 – 9.0	0 – 54	0.0 – 4.4	0 – 35	0 – 53	0 – 50
0.055 – 0.070	-	9.1 – 35.4	55 – 154	4.5 – 9.4	36 – 75	54 – 100	51 – 100
0.071 – 0.085	0.125 – 0.164	35.5 – 55.4	155 – 254	9.5 – 12.4	76 – 185	101 – 360	101 – 150
0.086 – 0.105	0.165 – 0.204	55.5 – 125.4	255 – 354	12.5 – 15.4	[3] 186 – 304	361 – 649	151 – 200
0.106 – 0.200	0.205 – 0.404	125.5 – 225.4	355 – 424	15.5 – 30.4	[3] 305 – 604	650 – 1249	201 – 300
[2]0.201+	0.405+	225.5+	425+	30.5+	[3] 605+	1250+	301+

1 – станції, як правило, повинні повідомляти показники на основі 8-годинних значень озону. Проте іноді, станції повідомляють показники на основі 1-годинних значень O_3. У цих випадках, на додаток до розрахунку 8-годинного значення індексу, O_3 можна розрахувати 1-годинне значення O_3, а також максимум із двох повідомлених значень.

2 – восьмигодинні показники показники O_3 не використовуються для підрахунку високих показників AQI (≥ 301). Показники AQI, вищі за 301 обраховуються на основі погодинних концентрацій озону.

3 – погодинні значення SO_2 не повинні використовуватись для обрахунку високих значень AQI (≥ 200).

4 – для найвищих значень AQI (≥ 301), значення AQI слід обчислювати за допомогою рівняння 1 і концентраціями, указаними для значення AQI 500.

Значення AQI 500 є такими:

O_3 – 0,604 ppm;

$PM_{2.5}$ – 325,4 μg/m^3;

PM_{10} – 604 μg/m^3;

CO ppm—50,4 ppm;

CO_2 – 1004 ppb;

NO_2 – 2049 ppb.

3) використовуючи наведене нижче рівняння 2.1, обчислити індекс;

$$I_p = \frac{I_{Hi}-I_{Lo}}{BP_{Hi}-BP_{Lo}}\left(C_p - BP_{Lo}\right) + I_{Lo}, \quad (2.1)$$

де I_p – індекс забруднювача, C_p – обрізане значення концентрації забруднювача, BP_{Hi} – вище граничне значення забруднювача, BP_{Lo} – нижнє граничне значення забруднювача, I_{Hi} – вище значення AQI, I_{Lo} – нижнє значення AQI.

4) після обчислення індексу забруднювача – округлити його до найближчого цілого значення.

Після обчислення індексів для всіх забруднювачів потрібно вибрати максимальний серед них. Це і буде фінальне значення індексу.

2.1.2.2 Індекс якості повітря, що використовується в Великобританії

Щоденний індекс якості повітря інформує про рівень забруднення повітря та надає рекомендації щодо дій і поради для збереження здоров'я. Індекс має шкалу від 1 до 10, поділену на чотири категорії: від «низького» (1) до «дуже високого» (10), що дозволяє легко зрозуміти рівень забруднення, подібно до індексу сонячної радіації або пилку [6].

Загальний індекс забруднення повітря для певного місця або регіону визначається за найвищою концентрацією одного з п'яти основних забруднювачів:

1) діоксид азоту (NO_2);
2) діоксид сірки (SO_2);
3) озон (O_3);
4) тверді частки розміром менше 2,5 мкм ($PM_{2.5}$);
5) тверді частки розміром менше 10 мкм (PM_{10}).

Для обчислення значень індексу слід звернутись до наведеної нижче табл. 2.3. На основі даних, наведених в табл. 2.3, вибрати відповідний індекс для забруднювача, а для фінального результату слід обрати максимальний із них.

Таблиця 2.3 – Діапазони значень забруднювачів повітря відповідно до рівнів якості повітря в Великобританії

Індекс	Концентрація забруднювачів в $\mu g/m^3$				
	SO_2	NO_2	$PM_{2.5}$	PM_{10}	O_3
1	0 – 88	0 – 67	0 – 11	0 – 16	0 – 33
2	89 – 177	68 – 134	12 – 23	17 – 33	34 – 66
3	178 – 266	135 – 200	24 – 35	34 – 50	67 – 100
4	267 – 354	201 – 267	36 – 41	52 – 58	101 – 120
5	355 – 443	268 – 334	42 – 47	59 – 66	121 – 140
6	444 – 532	335 – 400	48 – 53	67 – 75	141 – 160
7	533 – 710	401 – 467	54 – 58	76 – 83	161 – 187
8	711 – 887	468 – 534	59 – 64	84 – 91	188 – 213
9	888 – 1064	535 – 600	65 – 70	92 – 100	214 – 240
10	$\geq$ 1065	$\geq$ 601	$\geq$ 71	$\geq$ 101	$\geq$ 241

2.1.2.3 Індекс якості повітря, що використовується в Європейському Союзі

Забруднення повітря є найбільшим екологічним ризиком для здоров'я в Європі, спричиняючи серцево-судинні та респіраторні захворювання, які в найтяжчих випадках можуть призводити до передчасної смерті. Індекс якості повітря, розроблений Європейським агентством з охорони довкілля, допомагає

користувачам краще зрозуміти рівень забруднення повітря у своєму регіоні. Незважаючи на те, що якість повітря в Європі покращилася за останні десятиліття, рівні забруднювачів все ще перевищують стандарти ЄС і найсуворіші рекомендації ВООЗ [7].

Європейський індекс якості повітря розраховується на основі п'яти основних забруднювачів, які регулюються європейським законодавством:

1) озон (O_3);
2) діоксид азоту (NO_2);
3) діоксид сірки (SO_2);
4) мікрочастинки з діаметром менше 2,5 мікрометрів ($PM_{2.5}$);
5) мікрочастинки з діаметром менше 10 мікрометрів (PM_{10}).

Індекс визначається за найвищою концентрацією одного з п'яти основних забруднювачів, граничні значення яких наведені нижче в табл. 2.4.

Таблиця 2.4 – Діапазони значень забруднювачів повітря відповідно до рівнів якості повітря в Європейському союзі

Забруднювач	Індекси					
	1	2	3	4	5	6
O_3	0 – 50	50 – 100	100 – 130	130 – 240	240 – 380	≥ 381
NO_2	0 – 40	40 – 90	90 – 120	120 – 230	230 – 340	≥ 341
SO_2	0 – 100	100 – 200	200 – 350	350 – 500	500 – 750	≥ 751
PM_{10}	0 – 20	20 – 40	40 – 50	50 – 100	100 – 150	≥ 151
$PM_{2.5}$	0 – 10	10 – 20	20 – 25	25 – 50	50 – 75	≥ 76

Індекс якості повітря є важливим інструментом для моніторингу стану навколишнього середовища та оцінки впливу забруднення на здоров'я людей. Вибір методу розрахунку індексу залежить від специфіки країни та її екологічної ситуації, але загальні принципи залишаються однаковими.

2.2 Вибір джерел для обрахунку індексів якості повітря

Одним із ключових елементів системи моніторингу якості повітря є надійні джерела даних про концентрацію основних забруднювачів. Для забезпечення точності та актуальності інформації про стан атмосферного повітря розглянуто багато різних джерел, що надають дані по забруднювачам та після дослідження обрано два джерела.

2.2.1 Open Weather

Open Weather API є популярним сервісом, який надає метеорологічні дані, включаючи інформацію про концентрацію основних забруднювачів повітря: озону (O_3), діоксиду азоту (NO_2), діоксиду сірки (SO_2), твердих частинок PM_{10} та $PM_{2.5}$ [8].

Основними перевагами цього API є:

1) широке покриття географічних зон забезпечує дані для більшості регіонів світу, включаючи Україну;
2) актуальність даних забезпечує оновлення інформації кожні кілька годин;
3) легкість інтеграції – API пропонує зручну документацію та можливість роботи з різними мовами програмування.

2.2.2 Weather API

Weather API є ще одним важливим джерелом, яке спеціалізується на

наданні погодних даних у режимі реального часу, а також інформації про якість повітря [9]. Його основні переваги:

1) висока точність прогнозів – сервіс надає прогнози якості повітря на основі аналізу кількох метеорологічних моделей;

2) детальні дані – API дозволяє отримувати дані про концентрацію забруднювачів на основі географічних координат;

3) гнучкість – підтримує різноманітні формати даних для зручної інтеграції у веб-додатки та мобільні платформи.

2.3 Вибір мови програмування та технологічного стека

Проектування системи моніторингу якості повітря вимагає вибору оптимального програмного забезпечення та технологічного стека, які забезпечать ефективну роботу як серверної, так і клієнтської частини системи. Вибір мови програмування, фреймворків, бібліотек та інших інструментів впливає на продуктивність, масштабованість, безпеку та зручність користування системою.

2.3.1 Оцінка різних мов програмування для розробки серверної та клієнтської частини.

Вибір мови програмування є одним із ключових аспектів під час розробки серверної та клієнтської частини системи моніторингу якості повітря. Основними критеріями для оцінки мов програмування є їх продуктивність, зручність у використанні, підтримка сторонніх бібліотек і фреймворків, а також можливість інтеграції з API та іншими технологіями.

Серверна частина відповідає за отримання, обробку та зберігання даних, а

також за обробку запитів від клієнтів.

Оскільки вибір мови програмування для клієнтської частини при створенні системи у вигляді веб-сайту, то варіант тільки один – JavaScript та HTML. Тож є сенс розглядати мови програмування тільки для серверної частини.

2.3.1.1 Мова програмування Python

Python – це інтерпретована мова програмування з динамічною типізацією, що підтримує об'єктно-орієнтований підхід. Вона була створена у 1990 році Гвідо ван Россумом і стала однією з найбільш популярних мов для розробників у всьому світі. Python входить до більшості дистрибутивів Linux, що робить її зручною для використання в різних середовищах [10].

Інтерпретатор Python, а також стандартні бібліотеки доступні на всіх основних операційних системах як у скомпільованому вигляді, так і у вигляді вихідного коду. Завдяки цьому Python є кросплатформеною мовою, що забезпечує високу продуктивність додатків, розроблених на її основі.

Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану. Ця гнучкість дозволяє розробникам адаптувати мову до різноманітних задач і стилів програмування.

Дизайн Python орієнтований на об'єктно-орієнтовану модель програмування, яка вирізняється своєю простотою, елегантністю та потужністю. Хоча реалізація об'єктно-орієнтованого підходу в Python може мати свої особливості у порівнянні з іншими мовами, вона залишається зручною та ефективною для використання.

Переваги:

1) широкий вибір бібліотек для обробки даних (Pandas, NumPy);

2) фреймворки Flask і Django забезпечують просту та гнучку розробку серверної частини;

3) легкість інтеграції з API для збору даних.

Недоліки:

1) вища вимогливість до обчислювальних ресурсів порівняно з іншими мовами, такими як Go чи PHP;

2) у великих розподілених системах Python може виявитися менш масштабованим через його продуктивність і обмеження багатопоточності. Розробникам доводиться використовувати додаткові інструменти (наприклад, асинхронну бібліотеку asyncio), щоб компенсувати ці недоліки.

2.3.1.2 Мова програмування PHP

PHP (Hypertext Preprocessor) – це інтерпретована серверна мова програмування, спеціально розроблена для створення динамічних веб-сторінок та веб-додатків. Вперше випущена у 1995 році Рамусом Лердорфом, PHP стала однією з найбільш популярних мов для веб-розробки завдяки простоті використання, гнучкості та великій кількості доступних інструментів [10].

Переваги PHP:

1) код PHP не потребує компіляції, що дозволяє швидко тестувати, розробляти та розгортати додатки;

2) PHP дозволяє реалізовувати як прості скрипти, так і складні серверні додатки;

3) PHP підтримується такими веб-серверами, як Apache, Nginx, IIS, і працює на Windows, Linux, macOS та інших платформах;

4) PHP має багато потужних фреймворків, таких як Laravel, Symfony, CodeIgniter, які спрощують створення та підтримку додатків;

5) мова дозволяє швидко створювати прототипи завдяки легкому налаштуванню, великій кількості готових бібліотек і плагінів.

Недоліки PHP:

1) хоча PHP добре підходить для більшості веб-застосунків, його продуктивність може бути нижчою у порівнянні з такими мовами, як Go або Java, у високонавантажених системах;

2) PHP виконує один запит за раз, що ускладнює роботу з асинхронними задачами чи великою кількістю паралельних з'єднань.

PHP залишається одним із найпопулярніших виборів для веб-розробки завдяки своїй простоті, універсальності та широкій підтримці. Попри деякі обмеження, мова продовжує вдосконалюватись, а сучасні версії пропонують потужні інструменти та оптимізації, які роблять PHP конкурентоспроможним навіть у масштабних проєктах.

2.3.1.3 Мова програмування Java

Java – це об'єктно-орієнтована мова програмування зі строгою типізацією, яка була створена компанією Sun Microsystems і вперше випущена у 1995 році. З 2009 року підтримка та подальший розвиток мови перейшли до компанії Oracle, яка також займається розробкою баз даних та інших програмних рішень [10].

Код, написаний на Java, компілюється у спеціальний формат – байт-код, який виконується за допомогою віртуальної машини Java. Ця особливість дозволяє програмам, написаним на Java, бути кросплатформеними, оскільки JVM забезпечує їх виконання на будь-якій операційній системі, де встановлена відповідна віртуальна машина.

Через використання JVM продуктивність програм на Java раніше поступалася іншим мовам. Однак завдяки сучасним оптимізаціям та вдосконаленням, швидкість виконання Java-програм тепер порівнянна з програмами, написаними

на інших мовах.

Управління пам'яттю в Java здійснюється віртуальною машиною, що дозволяє автоматично виділяти та звільняти ресурси оперативної пам'яті. Хоча це значно спрощує розробку, такі механізми можуть іноді спричиняти збільшене використання пам'яті під час виконання програм.

Переваги Java:

1) завдяки віртуальній машині Java (JVM), програми, написані на Java, можуть виконуватись на будь-якій операційній системі без змін у коді, що робить Java чудовим вибором для масштабованих серверних рішень;

2) сучасні оптимізації JVM, такі як Just-In-Time (JIT) компіляція, дозволяють програмам на Java досягати продуктивності, близької до нативного коду;

3) Java має потужний інструментарій для роботи з багатопоточними програмами, що є критично важливим для серверних застосунків.

Недоліки:

1) завдяки автоматичному управлінню пам'яттю (Garbage Collector), програми на Java можуть споживати більше ресурсів, що може бути недоліком у середовищах з обмеженими ресурсами;

2) у порівнянні з деякими іншими мовами, такими як Go або Node.js, Java-програми потребують більше часу на старт через ініціалізацію JVM;

3) хоча JVM забезпечує кросплатформеність, її використання може ускладнювати інтеграцію з деякими специфічними апаратними рішеннями або системами.

Розглянувши різні мови програмування вирішено використовувати мову програмування PHP, оскільки високих навантажень не очікується, можна орендувати не дорогий сервер для виконання PHP-коду, а також дуже просто розгортати застосунок, адже не потрібна перекомпіляція, а отже і перезапуск JVM.

2.4 Аналіз використання баз даних та технологій агрегації даних

2.4.1 Огляд баз даних для зберігання даних про якість повітря.

Зберігання даних є одним із ключових аспектів у реалізації систем моніторингу якості повітря. Дані, що надходять від сенсорів, API або інших джерел, повинні бути організовані таким чином, щоб забезпечити їх швидкий доступ, обробку та аналіз. Зокрема, це включає часові ряди, географічні координати, концентрації забруднювачів та результати обчислення індексів якості повітря. Вибір бази даних значно впливає на продуктивність системи, її масштабованість і надійність.

Реляційні бази даних вже багато років залишаються стандартом для зберігання структурованих даних. Їхня основна перевага полягає у використанні таблиць із чітко визначеними зв'язками між даними, що дозволяє зручно зберігати та маніпулювати великими обсягами інформації.

NoSQL бази даних, на відміну від реляційних, дозволяють працювати з неструктурованими та напівструктурованими даними. Вони добре підходять для динамічних систем із великими обсягами даних, які швидко змінюються.

У цьому підрозділі буде розглянуто три основні бази даних, які можуть використовуватися для таких завдань, дві реляційні і одна нереляційна – MySQL, PostgreSQL і MongoDB.

2.4.1.1 Огляд реляційної БД MySQL

MySQL – одна з найпоширеніших реляційних баз даних, яка широко використовується у веб-розробці. Її простота, продуктивність і висока доступність зробили її вибором номер один для багатьох розробників і компаній. MySQL

пропонує всі необхідні інструменти для створення серверних застосунків, зокрема можливості роботи з транзакціями, реплікацією даних і масштабуванням [10].

Для систем моніторингу якості повітря MySQL підходить завдяки простій інтеграції з веб-фреймворками, можливості працювати з часовими мітками та великими обсягами структурованих даних.

Однак MySQL має певні обмеження. Наприклад, для великих розподілених систем її масштабованість може бути недостатньою без додаткових інструментів, таких як кешування або розподіл навантаження.

2.4.1.2 Огляд реляційної БД PostgreSQL

PostgreSQL є потужною реляційною базою даних, яка пропонує значно ширші можливості у порівнянні з MySQL. Вона підтримує складні типи даних, транзакції високого рівня та роботу з напівструктурованими даними у форматі JSON. Ці особливості роблять PostgreSQL універсальним вибором для систем моніторингу, де потрібна висока точність і складні запити.

Одна з найсильніших сторін PostgreSQL – це її підтримка геопросторових даних через розширення PostGIS. Для систем моніторингу якості повітря, де важливо враховувати місцезнаходження сенсорів і створювати інтерактивні мапи, PostgreSQL із PostGIS забезпечує ефективну обробку просторових запитів.

Крім того, PostgreSQL добре працює з великими обсягами даних і дозволяє виконувати аналітичні запити, що особливо корисно для аналізу історичних даних або трендів у зміні якості повітря. Наприклад, система може зберігати дані про якість повітря за останні кілька років і на їх основі прогнозувати можливі зміни в майбутньому.

Попри свої переваги, PostgreSQL вимагає більше ресурсів і складнішого налаштування, ніж MySQL. Це може бути недоліком для невеликих проєктів або

команд, які тільки починають працювати з базами даних.

2.4.1.3 Огляд нереляційної БД MongoDB

MongoDB – це документно-орієнтована NoSQL база даних, яка зберігає дані у форматі BSON (розширений JSON). Це дозволяє легко інтегрувати її з сучасними веб-додатками, що працюють із JSON-форматом. MongoDB ідеально підходить для проєктів, де дані надходять із різних джерел у різних форматах.

Однією з ключових переваг MongoDB є її гнучкість. У випадках, коли структура даних змінюється (наприклад, додаються нові параметри забруднення), MongoDB дозволяє швидко адаптувати базу без необхідності змінювати схему.

Однак MongoDB також має свої обмеження. Наприклад, вона менш ефективна для складних транзакцій, де потрібна висока цілісність даних. Це може стати проблемою для систем, де критично важливо уникати дублювання або втрати інформації.

Після розгляду трьох варіантів СУБД вирішено використовувати MySQL, оскільки в проєкті не очікується надвеликих навантажень, сама база легковісна, що здешевлює витрати на оренду серверу, а також MySQL підтримує geospatial типи, які дозволяють на рівні бази даних шукати відстань між геокоординатами.

Нереляційна MongoDB дещо складніша для інтеграції і оскільки структури даних матимуть структурно однаковий формат, то і необхідності в використанні такої СУБД немає.

2.4.2 Вибір алгоритмів агрегації даних для покращення точності.

Для об'єднання даних з різних джерел і видачі результатів як єдину зведену інформацію, є кілька підходів до агрегації даних, в залежності від їхньої структури та типу.

2.4.2.1 Агрегація через середнє/зважене

Якщо дані з двох джерел мають схожу мету, але можуть мати різні рівні точності (наприклад, одне джерело має вищу точність, інше – нижчу), можна використовувати зважене середнє для об'єднання значень.

Алгоритм працює наступним чином – для кожного параметра якості повітря, що надходить з двох джерел, обчислюєте середнє значення, де кожному джерелу надається вага в залежності від його точності. Детальне обчислення наведене нижче в рівнянні 2.2.

$$\text{Об}'\text{єднане значення} = \frac{W_1 \times X_1 + W_2 \times X_2}{W_1 + \quad _2}, \tag{2.2}$$

де X_1, X_2 – це показники з двох джерел, W_1, W_2 – ваги цих джерел.

Це дозволяє уникнути впливу помилкових чи ненадійних даних, якщо одне джерело більш точне. Проте, потрібно мати додаткові дані про точність кожного джерела для правильної ваги.

Нажаль, «залізних» даних про надійність джерел немає, тож цей метод, нажаль, не підходить.

2.4.2.2 Алгоритм інтерполяції

Якщо дані з двох джерел мають різні тимчасові мітки або локації (наприклад, дані з різних часів чи різних місць), можна застосувати інтерполяцію, щоб заповнити пропуски або привести дані до спільної тимчасової шкали.

Формула обрахунку лінійної інтерполяції між двома значеннями наведена нижче в рівнянні 2.3.

$$Interpolated\ Value = X_1 + \frac{(X_2 - X_1)}{(Y_2 - Y_1)} \times (Y - Y_1) \quad (2.3)$$

де X_1, X_2 – значення показників для двох джерел,

Y_1, Y_2 – це міста чи часові мітки (в залежності від інтерпольованих даних)

Y – це точка, для якої потрібно провести інтерполяцію

Оскільки нам не потрібно об'єднувати дані для міст, для яких відсутні показники – ми не будемо використовувати цей алгоритм.

2.4.2.3 Метод максимального вибору (Max/Min Selection)

Метод максимального вибору (Max Selection) – це підхід до агрегації даних, що базується на виборі найбільшого значення серед доступних показників для певного параметра. Цей метод часто застосовується в системах моніторингу, коли дані надходять із кількох джерел, і необхідно визначити загальний рівень впливу або критичний стан на основі наявної інформації.

Цей метод досить простий в реалізації, проте забезпечує найбільшу обережність і запобігає ризику недооцінки критичного стану. У контексті систем моніторингу якості повітря метод Max Selection дозволяє відобразити найбільш несприятливий сценарій, що особливо важливо для попередження населення про можливі небезпеки для здоров'я.

У деяких випадках поряд із максимальним значенням можуть використовуватися й інші метрики, наприклад, мінімальні значення або середнє між максимальним і середнім. Це дозволяє знайти баланс між обережністю та точністю.

Вибрано цей метод, оскільки він наразі найкраще підійде для обробки даних з обраних джерел інформації.

2.5 Технології для візуалізації даних та побудови інтерактивних мап

Ефективна візуалізація даних є ключовим елементом сучасних систем моніторингу, оскільки вона дозволяє користувачам швидко та інтуїтивно зрозуміти складну інформацію. У системах моніторингу якості повітря це особливо важливо, адже від наочності та доступності інформації залежить, наскільки оперативно населення чи органи управління можуть реагувати на зміни екологічного стану.

Серед основних завдань візуалізації – представлення даних про якість повітря у зручному форматі, створення інтерактивних мап із точками забруднення, графіків динаміки змін показників і систем сповіщень.

Інтерактивні мапи є одним із найзручніших способів представлення даних про якість повітря. Вони дозволяють користувачам легко знаходити інформацію про рівень забруднення в конкретних регіонах, відображати тренди та виявляти проблемні зони. Для створення таких мап розглянемо два популярні інструменти: Google Maps та OpenStreetMap, які можна інтегрувати через бібліотеку Leaflet.

Google Maps пропонує потужний API для роботи з мапами, який легко інтегрується у веб- та мобільні застосунки. Цей сервіс вирізняється високою деталізацією картографічних даних, які включають супутникові знімки, маршрути, інформацію про дорожній трафік і багато іншого.

Google Maps API дозволяє додавати власні шари, наприклад, точки із забрудненням повітря, і кастомізувати вигляд мапи відповідно до потреб користувача. Завдяки вбудованим інструментам, таким як геокодування та зворотне геокодування, можна легко отримати координати за адресою або визначити найближче місто до заданої точки.

Основними недоліками Google Maps є залежність від платної моделі використання після перевищення безкоштовного ліміту запитів і обмеження в

адаптації для проєктів із низьким бюджетом.

Натомість OpenStreetMap у поєднанні з бібліотекою Leaflet є повністю безкоштовним рішенням із відкритим кодом. OpenStreetMap надає доступ до географічних даних, створених спільнотою, а Leaflet дозволяє інтегрувати ці дані у вебдодатки, забезпечуючи можливість створення кастомних мап із додатковими шарами. Цей інструмент вирізняється легкістю, гнучкістю та підтримкою адаптивних інтерфейсів, що робить його ідеальним для мобільних застосунків. Однак OpenStreetMap поступається Google Maps у деталізації даних, а для додаткових функцій, таких як геокодування, потребує інтеграції сторонніх рішень.

Обидва інструменти мають свої сильні сторони. Google Maps підійде для проєктів, які потребують високої точності даних і додаткових сервісів, таких як маршрутизація або дані про трафік. Водночас OpenStreetMap через використання Leaflet ідеально підходить для проєктів із обмеженим бюджетом, де важливими є гнучкість і можливість налаштування.

Через обмеженість бюджету та відсутність необхідності в значній кастомізації вигляду мапи обрано OpenStreetMap.

2.6 Висновки до розділу 2

У другому розділі проведено аналіз ключових технологій, методів та інструментів, необхідних для створення системи моніторингу якості повітря.

Розглянуто принципи роботи систем моніторингу, які базуються на зборі, обробці та аналізі даних із сенсорів та інших джерел. Особливу увагу приділено опису основних забруднювачів повітря та методів їх моніторингу, що є основою для оцінки впливу забруднення на здоров'я людей та навколишнє середовище.

Також проведено детальний огляд існуючих підходів до обчислення індексів якості повітря, зокрема використання індексів США, Великобританії та

Європейського Союзу. Акцентовано на важливості вибору алгоритмів для агрегації даних із різних джерел. Найбільш доцільним у рамках цього проєкту визнано використання методу максимального вибору (Max Selection), який забезпечує найбільшу обережність у визначенні рівня забруднення.

При виборі джерел даних про якість повітря обґрунтовано використання трьох основних API: OpenWeather, Weather API та Google Environmental API. Кожне з них забезпечує актуальні дані про забруднювачі, що є необхідними для розрахунку індексів якості повітря.

У межах аналізу мов програмування та технологічного стека для серверної частини системи розглянуто Python, PHP та Java. Зважаючи на простоту розгортання, доступність ресурсів і можливість роботи з API, для реалізації серверної частини обрано мову програмування PHP.

Розглянуто різні бази даних для зберігання інформації про якість повітря. На основі аналізу обрано MySQL як основну СУБД завдяки її ефективності, простоті інтеграції та підтримці геопросторових даних. PostgreSQL та MongoDB також були розглянуті, проте вони не відповідали потребам проєкту через підвищену складність налаштування та непотрібність напівструктурованих даних.

Окрему увагу приділено технологіям для візуалізації даних. Проведено порівняння між Google Maps API та OpenStreetMap із використанням Leaflet. З огляду на бюджетні обмеження та гнучкість налаштувань, обрано OpenStreetMap з інтеграцією через JavaScript бібліотеку Leaflet як основний інструмент для створення інтерактивної мапи.

Таким чином, у цьому розділі не лише закладено теоретичні основи системи моніторингу якості повітря, а й обґрунтовано вибір оптимальних інструментів, технологій і методів для її реалізації. Це забезпечить ефективність, надійність і доступність майбутньої системи.

3 МОДЕЛЮВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Загальні вимоги до інформаційної системи

Інформаційна система моніторингу якості повітря має відповідати ряду вимог, спрямованих на забезпечення її функціональності, надійності та зручності для користувачів.

З точки зори функціональних вимог система повинна надавати наступні послуги:

1) візуально показувати індекси для відповідного міста на мапі;

2) дати можливість змінити тип індексу (в кожен момент часу може бути відображений тільки один тип індексу із Британського, ЄС та США);

3) надати кольоровий та вербальний опис значень індексів для кожного типу індексу для розуміння користувачем відображених даних;

4) надати список міст, які підтримуються інформаційною системою;

5) надати посилання на чат-бот в Telegram, який вміє по даним геолокації користувача знаходити найближче до нього місто та періодично інформувати користувача про зміну якості повітря;

6) надати опис системи для розуміння користувачем принципів роботи системи, її цілі та функціональність, а також надати посилання на інструменти, використані при розробці для уникнення ліцензійних спорів.

7) зблизити мапу до поточного місцезнаходження користувача, а також розширити мапу назад до кордонів України.

8) по наведенню на індекс на мапі надати користувачу детальну інформацію по всім забруднювачам, на основі яких був обрахований індекс.

9) надати детальну інформацію по конкретному місті на окремій сторінці веб-порталу.

Для реалізації функціональних вимог система повинна вміти наступне:

1) для обрахунку індексів конвертувати показники забруднювачів в різні одиниці вимірювання;

2) система повинна вміти обчислювати різні типи індексів якості повітря (US, UK, EU типи індексів);

3) для коректного відображення даних з різних джерел – агрегувати такі дані обраним методом агрегації даних;

4) безпомилково періодично опитувати декількох постачальників даних для зберігання та обрахунку показників забруднювачів повітря;

5) вміти працювати з браузерною геолокацією для отримання найближчого до користувача міста;

6) система має знати геокордони (geoboundaries) України для підсвічювання їх на мапі та корректного масштабування;

7) для інтеграції даних з кількох джерел система повинна вміти об'єднувати ці дані з використанням обраного методу агрегації (Max Selection);

8) для підтримки роботи Telegram чат-боту система повинна вміти працювати з Telegram bot API;

9) для періодичного та автоматичного опитування даних з обраних джерел, та інформування користувача чат-боту Telegram система повинна вміти за розкладом, за допомогою системи Cron, автоматично запускати відповідні скрипти.

Для захисту від несанкціонованого доступу до сервера та захисту даних користувачів система повинна надавати дані в зашифрованому вигляді за допомогою протоколу HTTPS (використовувати TLS-сертифікати), а для унеможливлювання доступу до бази даних – заборонити їй отримувати запити від зовнішньої мережі. За допомогою файрволу заборонити запити до всіх портів окрім 80 (HTTP протокол), 443 (HTTPS протокол) та 22 (SSH протокол). Доступ до

серверу повинен бути доступним виключно через SSH-протокол з використанням OpenSSH.

Для здешевлення системи, виконанню всіх поставлених вимог та заради простоти розгортки застосунку використано VPS-сервер на базі ОС Ubuntu 24.04.

Сформульовані вимоги до інформаційної системи відображають потреби кінцевих користувачів та завдання, які вона має вирішувати. Ці вимоги забезпечують створення функціональної, надійної та зручної системи моніторингу якості повітря, здатної відповідати сучасним викликам та стандартам.

3.2 Проектування бази даних інформаційної системи

База даних має зберігати відносно невелику кількість різних сутностей. Для повноцінного функціонування достатньо зберігати дані міст, для яких збираються виміри показники забруднення, власне самі показники забруднення та пораховані по ним індекси, які будуть виведені на мапі та анонімізовані (без персональних даних) локації користувачів для того, щоб знати кому і для якого міста надсилати сповіщення в чат-бот.

Також слід зазначити, що в різних провайдерах даних міста можуть мати різні назви і для того, щоб зіставити дані з відповідним містом слід також мати таблицю БД зі списком так званих псевдонімів.

ERD бази даних зображено на рис. 3.1.

Рисунок 3.1 – EDR бази даних проєкту

Структура бази даних відповідає основним вимогам для зберігання даних, необхідних для функціонування системи моніторингу якості повітря. Вона містить сутності для збереження інформації про міста, їхні альтернативні назви (псевдоніми), дані про забруднення, обчислені індекси якості повітря та локації

користувачів.

Важливим аспектом є таблиця БД псевдонімів міст, яка дозволяє зіставляти дані від різних провайдерів, у яких назви міст можуть відрізнятися. Це забезпечує коректну інтеграцію даних з різних джерел.

Додатково, база даних підтримує збереження анонімізованих локацій користувачів для надсилання сповіщень через чат-бот. Завдяки цьому забезпечується функціональність геолокаційного моніторингу якості повітря, не порушуючи конфіденційності користувачів.

ERD чітко демонструє логічні зв'язки між сутностями, що сприяє організованому зберіганню даних і полегшує обробку запитів у системі. Такий дизайн бази даних є оптимальним для забезпечення функціональності системи та ефективного управління даними.

3.3 Вибір програмних бібліотек та фреймворків для розробки системи

3.3.1 JavaScript фреймворк Vue.js

Vue.js – це прогресивний JavaScript-фреймворк для створення користувацьких інтерфейсів (UI) та SPA веб-додатків. Він фокусується на спрощенні розробки інтерфейсів через реактивне програмування та компонентний підхід [11].

Розроблений Еван Ю (Evan You) у 2014 році, станом на сьогодні цей фреймворк є одним із найпопулярніших інструментів для веб-розробників завдяки своїй простоті, легкості та гнучкості.

Основні особливості Vue.js:

1) Vue.js використовує реактивний механізм, що дозволяє автоматично оновлювати інтерфейс при зміні даних;

2) інтерфейс розділяється на компоненти – незалежні та повторно використовувані блоки, які мають власну логіку, розмітку та стилі;

3) використовує HTML-подібний синтаксис для створення інтерфейсів, дозволяючи прив'язувати дані до елементів через директиви (наприклад, v-if, v-for, v-bind);

4) можна використовувати Vue як частину проєкту (для окремих сторінок або компонентів) або створювати повноцінні односторінкові застосунки (SPA);

5) двостороннє прив'язування даних – за допомогою директиви v-model Vue забезпечує синхронізацію між моделлю даних і UI.

Vue.js – це ідеальний вибір для тих, хто шукає простий, гнучкий і продуктивний фреймворк для створення сучасних веб-застосунків.

3.3.2 PHP фреймворк Laravel

Laravel – це популярний фреймворк для веб-розробки на мові програмування PHP, створений Тейлором Отвеллом (Taylor Otwell) у 2011 році. Завдяки своїй простоті, потужності та великій екосистемі інструментів, Laravel став одним із найбільш використовуваних PHP-фреймворків [12].

Основні можливості Laravel:

1) MVC-архітектура забезпечує структурованість і логічний поділ застосунку;

2) простий і зрозумілий механізм маршрутизації для обробки HTTP-запитів;

3) інструменти для створення, оновлення та керуванню структурою бази даних через код;

4) Eloquent ORM є потужним інструментом для роботи з базою даних, який дозволяє взаємодіяти з даними як з об'єктами;

5) надає інструмент планувальника завдань (Scheduler) для автоматизації виконання завдань без стороннього програмного забезпечення, наприклад, cron;

6) вбудований захист від поширених веб-загроз, таких як SQL-ін'єкції, CSRF, XSS;

7) має чудову документацію та інтуїтивний синтаксис.

Laravel – це потужний інструмент для веб-розробки, який поєднує простоту, функціональність і надійність. Завдяки багатій екосистемі та активній спільноті, Laravel дозволяє розробникам створювати масштабовані та якісні веб-додатки, забезпечуючи продуктивність і комфорт у роботі.

3.3.3 Бібліотека для інтеграції OpenStreetMap – Leaflet.js

Leaflet.js – це легка, проста у використанні, та потужна JavaScript-бібліотека для створення інтерактивних карт. Вона дозволяє легко інтегрувати карти на веб-сторінки та додавати до них різноманітні функціональні можливості [13].

Особливості Leaflet.js:

1) малий розмір бібліотеки (~39 KB), що забезпечує швидке завантаження;

2) має вбудовану підтримку роботи з сенсорними екранами;

3) підтримує кастомні шари, іконки, маркери, спливаючі вікна, лінії, багатокутники тощо;

4) робота з відкритими даними: Leaflet інтегрується з OpenStreetMap;

5) багата екосистема плагінів для додавання функцій, таких як геокодування, маршрутизація, теплові карти тощо.

Leaflet.js – це чудовий вибір для створення швидких, інтерактивних і зручних карт у веб-застосунках.

3.3.4 Бібліотека з готовими UI-рішеннями PrimeVue

PrimeVue – це повнофункціональна бібліотека компонентів для Vue.js, яка забезпечує розробників сучасними, кастомізованими, та інтерактивними UI-компонентами для створення веб-додатків. PrimeVue пропонує понад 90 готових компонентів, таких як таблиці, форми, меню, діаграми, модальні вікна тощо.

Основні переваги PrimeVue:

1) широкий вибір компонентів – включає як базові елементи (кнопки, поля вводу), так і складні (таблиці, графіки, календарі, редактори);
2) компоненти PrimeVue легко налаштовуються та інтегруються у проєкти на Vue.js;
3) компоненти адаптуються до різних розмірів екранів і підтримують мобільні пристрої;
4) пропонує кілька тем, які можна легко налаштовувати під стиль додатка;
5) забезпечує модульність – компоненти можна додавати окремо, завантажуючи лише ті, які потрібні для проєкту.

PrimeVue – це потужний інструмент для створення сучасних, естетичних і функціональних інтерфейсів на основі Vue.js, що забезпечує гнучкість і зручність у розробці.

3.3.5 Бібліотека для роботи з HTTP-запитами Guzzle

Guzzle – це популярна PHP-бібліотека для роботи з HTTP-запитами. Вона дозволяє легко виконувати HTTP-запити, працювати з API, а також керувати потоками запитів (асинхронні або синхронні запити). Guzzle підтримує протоколи HTTP/1.1, HTTP/2 та інші [15].

Основні можливості Guzzle:

1. Простий інтерфейс для виконання HTTP-запитів
2. Асинхронна робота – використання промісів (Promises) для асинхронного виконання запитів.
3. Автоматичне кодування та декодування JSON.
4. Підтримка стрімінгу великих файлів або потокових відповідей.

Guzzle – це потужний і гнучкий інструмент для роботи з HTTP-запитами, що робить його незамінним вибором для PHP-розробників, які взаємодіють із веб-сервісами та API.

3.3.6 Бібліотека для роботи з Telegram bot API irazasyed/telegram-bot-sdk

Telegram Bot SDK – це PHP-бібліотека, створена для спрощення розробки Telegram-ботів. Вона надає інтуїтивний API для взаємодії з Telegram Bot API, дозволяючи розробникам швидко створювати, налаштовувати та керувати ботами.

Переваги:

1) дозволяє швидко інтегруватись з Telegram Bot API;
2) можливість налаштувати бот під специфічні потреби;
3) підтримка всіх основних можливостей Telegram API;
4) добре задокументована бібліотека з прикладами використання.

Telegram Bot SDK – це потужний інструмент для PHP-розробників, який значно спрощує створення та управління ботами для Telegram.

3.4 Проектування програмної архітектури інформаційної системи

Проектування програмної архітектури інформаційної системи є одним із ключових етапів розробки, оскільки від її якості залежить стабільність, масштабованість та ефективність роботи системи. Архітектура визначає, як компоненти системи взаємодіють між собою, як здійснюється обробка даних, а також забезпечує основу для подальшого розвитку і підтримки.

Для розробки системи обрано шаблон MVC (схематично зображений на рис. 3.2), оскільки він дає змогу легко і зрозуміло розділити застосунок на три різних логічні частини, що в подальшому дасть змогу легше розуміти та вносити зміни в кодову базу.

Логіка додатку, яка відповідає за маршрутизацію, роботу з БД, авторизацію тощо, здебільшого вже створена обраним фреймворком – Laravel.

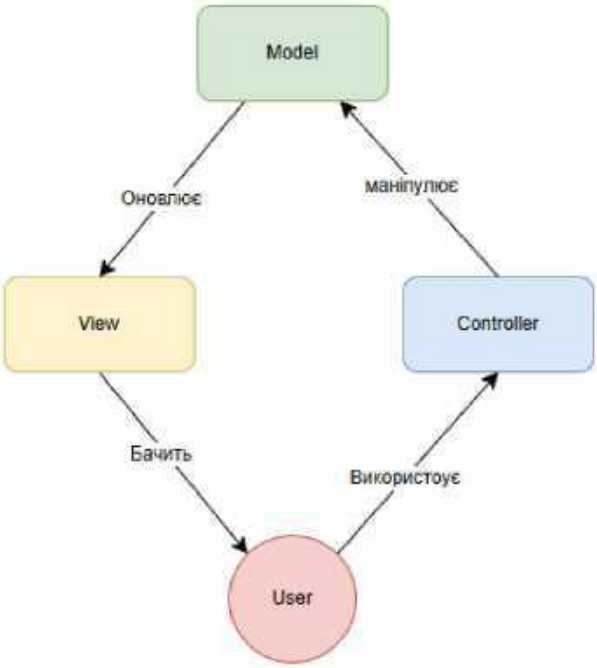

Рисунок 3.2 – Діаграма роботи MVC патерну

Окрім цього, виділено окремий модуль системи, який відповідає за бізнес-логіку. В ньому знаходиться код для інтеграції провайдерів даних, обробки цих даних і т.п.

Загалом для забезпечення та підтримки роботи системи нам необхідно спроєктувати та розробити наступні модулі бізнес логіки та логіки додатку:

1) високорівневий модуль роботи з базою даних;
2) модуль інтеграції провайдерів даних;
3) модуль обчислення індексів;
4) модуль агрегації даних з різних джерел;
5) модуль конвертації одиниць вимірювання;
6) модуль роботи з Telegram чат-ботом.

Спроєктуємо кожен модуль бізнес-логіки в наступних підрозділах.

3.4.1 Високорівневий модуль роботи з базою даних

Рисунок 3.3 – Діаграма класів високорівневого модуля роботи з БД

Даний модуль (зображений на рис. 3.3) включає в себе класи для роботи з базою даних. Імплементація запитів буде в методах цих класів, тож для роботи з базою даних в інших модулях додатку будуть використовуватись зрозумілі іменовані виклики заміть складнішого SQL-коду.

Загалом кожен метод в цих класах досить простий у виконанні та не вартий детального опису, проте тут слід виділити метод getNearestCity. Він за допомогою вбудованих функцій MySQL ST_Distance обчислює відстань між двома точками (локацією користувача та містом в БД), сортує по обчисленій відстані та бере перший запис, який і являється найближчим до користувача містом.

3.4.2 Модуль інтеграції провайдерів даних

Для того, щоб повністю інтегрувати дані, отримані з OpenWeather API потрібно зробити HTTP запит з координатами міста, використовуючи бібліотеку Guzzle і зберегти отримані результати, позначивши попередні результати вимірювань, збережених в БД як застарілі. На діаграмі, зображеній на рис. 3.4, це виглядає наступним чином:

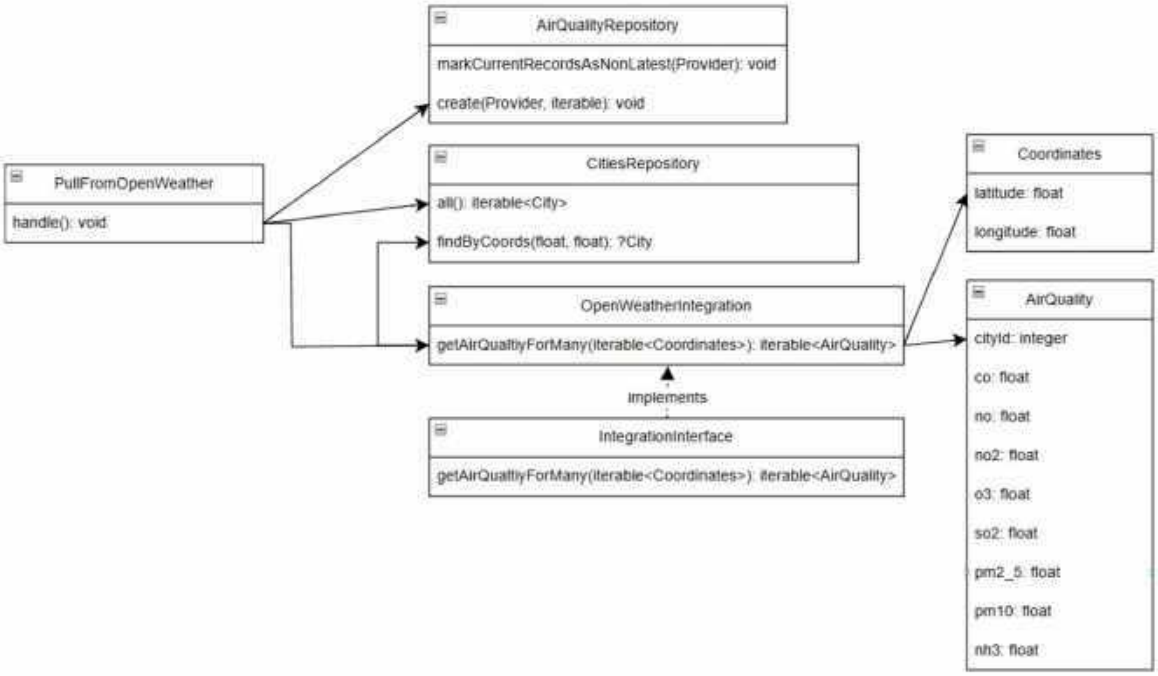

Рисунок 3.4 – Діаграма класів модуля інтерації провайдерів даних

Важливим компонентом тут є PullFromOpenWeather клас-команда. За допомогою планувальника задач Laravel він запускатиметься щогодини, і для всіх міст, що підтримуються системою, посилатиме асинхронно по 25 (це число можна буде змінити за бажанням) запитів на API OpenWeather, після чого, отримані результати буде зберігати в таблиці БД air_quality_records.

За таким самим принципом працюватиме інтеграція з WeatherAPI, єдина різниця буде в імплементації WeatherApiIntegration класу.

Така структура модуля дозволить нам з легкістю розширювати систему новими провайдерами даних, адже для інтеграції нового провайдера потрібну буде лише реалізувати інтерфейс, та створити команду для планувальника задач.

3.4.3 Модуль конвертації одиниць вимірювання

Це досить простий механізм, але дуже важливий, оскільки для обчислення різних індексів можуть використовуватись різні одиниці вимірювання вмісту забруднювачів в повітрі. Наприклад, Американський індекс якості повітря деякі забруднювачі вимірює в ppm та ppb (к-сть частинок речовини на мільйон частинок іншої речовини та к-сть частинок речовини на мільярд частинок іншої речовини відповідно), а Європейський індекс розраховуються в μg/m³ (кількість мікрограмів (мільйонних часток грама) речовини, що міститься в одному кубічному метрі повітря або іншого середовища).

API зазвичай надають дані в Європейських одиницях вимірювання, тож для обчислення Американського типу індекса їх слід привести в відповідні одиниці вимірювання.

Конвертація μg/m³ у ppm залежить від молекулярної маси речовини та умов (температури та тиску), оскільки μg/m³ – це масова концентрація, а ppm – це об'ємна частка.

Тож для конвертації нам спочатку потрібно дізнатись молекулярну масу та молярний об'єм забруднювачів. Він приведений нижче в табл. 3.1.

Таблиця 3.1 – Молекулярна маса та об'єм забруднювачів

Речовина	Молекулярна маса	Молярний об'єм
CO	28.01	22.4
NO_2	46.0055	22.45
SO_2	64.066	21.89
O_3	48.0	21.6

Використовуючи дані, наведені вишче можна конвертувати μg/m³ в ppm за формулою:

$$ppm = \frac{P \times V_m}{M \times 1000}$$

Де P – концентрація забруднювача в µg/m³,

V_m – молярний об'єм

M – молярна маса.

Для конвертації в ppb формула та ж сама, тільки додатково результат треба помножити на 1000.

3.4.4 Модуль обчислення індексів якості повітря

Система підтримує три типи індексів якості повітря: Американський, Британський та Європейський. Діаграма модуля обрахунку цих індексів зображена на рис. 3.5.

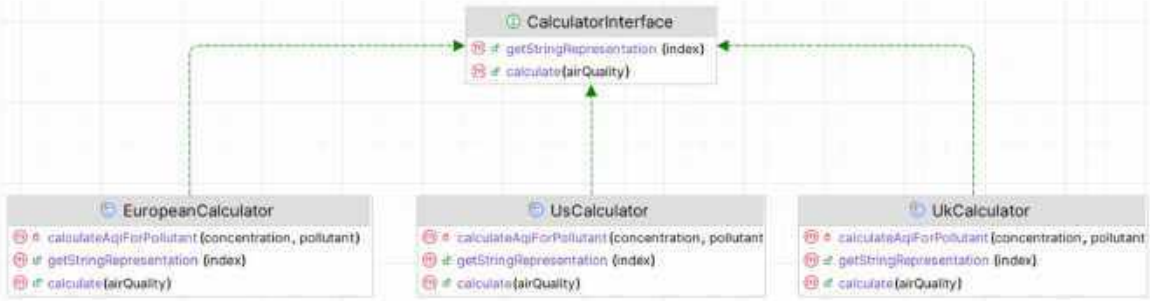

Рисунок 3.5 – Діаграма класів модуля обчислення індексів якості повітря

Є один інтерфейс, що описує два методи – строкова репрезентація (людиночитаєма) індексу, та числова репрезентація. Кожен клас імплементує інтерфейс з урахуванням правил обрахунку індексу, описаних в другому розділі.

3.4.5 Модуль агрегації даних з різних джерел

Модуль агрегації даних з різних джерел є ключовим компонентом інформаційної системи, оскільки він відповідає за інтеграцію, обробку та уніфікацію даних, отриманих від кількох зовнішніх провайдерів. Його мета – забезпечити точність, узгодженість і актуальність інформації, яка використовується для

обчислення індексів якості повітря та відображення цих даних у системі. Схема класів даного модуля зображена на рис. 3.6.

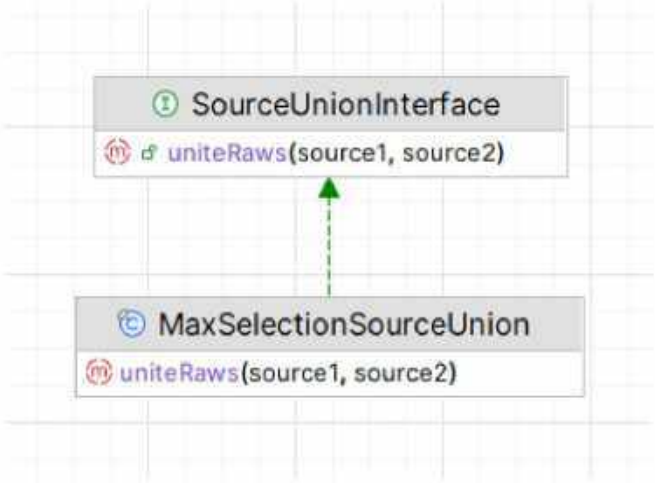

Рисунок 3.6 – Діаграма класів модуля агрегації даних

3.4.6 Модуль роботи з Telegram чат-ботом

Модуль роботи з Telegram чат-ботом є важливою частиною інформаційної системи, яка забезпечує інтерактивність і дозволяє користувачам отримувати дані про якість повітря в режимі реального часу. Основне завдання модуля — обробляти запити від користувачів, надавати відповідну інформацію та забезпечувати зручну взаємодію через Telegram.

Принцип роботи з Telegram ботами зображений на рис.3.7.

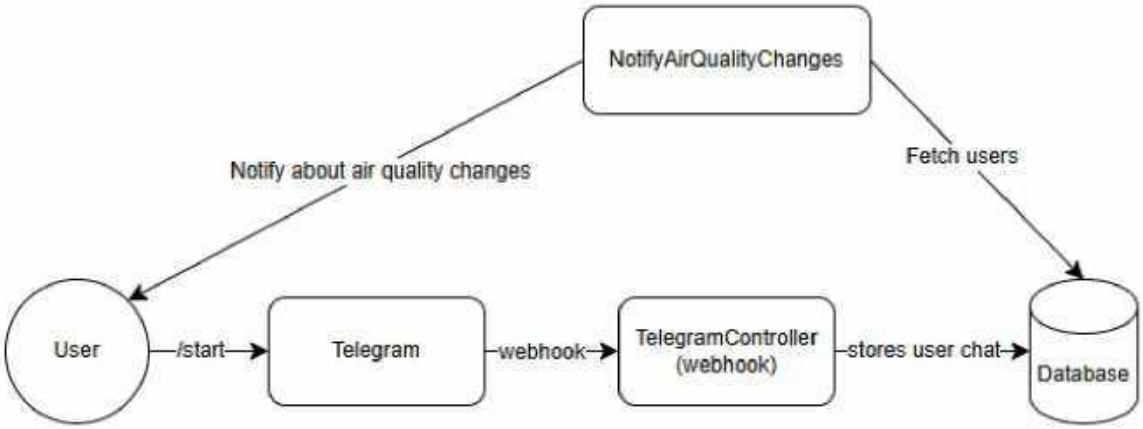

Рисунок 3.7 – Принцип роботи з Telegram Bot API

Тож для реалізації нам потрібно буде дві частини – контролер, який буде обробляти вебхуки від Telegram, а друга частина буде писати в чат, який був збережений при обробці вебхука, дані про зміну якості повітря відповідно до переданої геолокації користувача. Також, потрібно реалізувати дві команди, які будуть реєструвати та видаляти конфігурацію вебхука.

3.5 Розробка застосунку

Розробка застосунку є ключовим етапом реалізації інформаційної системи, що поєднує у собі всі компоненти, описані в попередніх розділах, для створення цілісного продукту. Основними завданнями цього етапу є: реалізація функціональних модулів, інтеграція їх між собою, забезпечення коректної роботи інтерфейсу користувача, а також тестування і оптимізація системи.

Розробка серверної частини застосунку розпочалася з встановлення всіх необхідних бібліотек. Команди встановлення наведені в лістингу 3.1.

Лістинг 3.1 – Bash-скрипт для встановлення бібліотек та фреймворків

```
composer create-project --prefer-dist laravel/laravel
composer require guzzlehttp/guzzle
```

```
composer require irazasyed/telegram-bot-sdk
npm install primevue
npm install leaflet
npm install vue-leaflet
```

Після встановлення необхідних фреймворків та бібліотек слід приступити до розробки визначених модулів.

Відповідно до моделювання інформаційної системи розроблено високорівневий модуль роботи з базою даних представлений трьома класами, які називаються репозиторії, відповідно до патерну програмування Repository.

Окрім цього інтегровано провайдерів даних в систему, їх дані об'єднуються за допомогою обраного методу агрегації даних, імплементовано конвертер одиниць вимірювання, створено модуль обчислення індексів якості повітря, а також реалізовано роботу з Telegram чат-ботом.

В лістингу 3.2 наведено деякі фрагменти коду створених модулів.

Лістинг 3.2 – Фрагменти коду серверної частини застосунку

```
final class CitiesRepository
{
…
    public function findCity(string $city, float $latitude,
float $longitude): ?City
    {
        $cityObj = $this->findByCoords($latitude, $longitude,
false);
        if ($cityObj === null) {
            Log::critical('Unable to find corresponding city
for coordinates', [
                'latitude' => $latitude,
                'longitude' => $longitude,
            ]);

            $cityObj = $this->findByCity($city);
            if ($cityObj === null) {
                Log::critical('Unable to find corresponding
city for name', ['name' => $city]);
```

```
                    return $this->getNearestCity($latitude,
$longitude);
                }
            }

            return $cityObj;
        }

        private function getNearestCity(float $latitude, float
$longitude): City
        {
            return City::selectRaw(
                'id, name, ST_Distance(location,
ST_GeomFromText(?)) as distance',
                ["POINT($longitude $latitude)"]
            )
                ->orderBy('distance')
                ->limit(1)
                ->firstOrFail();
        }
    …
    }
    final class ConcentrationConverter
    {
        private const float MOLECULAR_WEIGHT_CO = 28.01;
        private const float MOLECULAR_WEIGHT_NO2 = 46.0055;
        private const float MOLECULAR_WEIGHT_SO2 = 64.066;
        private const float MOLECULAR_WEIGHT_O3 = 48.0;

        private const float MOLAR_VOLUME_CO = 22.4;
        private const float MOLAR_VOLUME_NO2 = 22.45;
        private const float MOLAR_VOLUME_SO2 = 21.89;
        private const float MOLAR_VOLUME_O3 = 21.6;

        public function microgramsPerCubicMeterToPpm(Pollutant
$pollutant, float $concentration): float
        {
            return match ($pollutant) {
                Pollutant::CO => ($concentration *
self::MOLAR_VOLUME_CO) / self::MOLECULAR_WEIGHT_CO * 1000,
                Pollutant::SO2 => ($concentration *
self::MOLAR_VOLUME_SO2) / self::MOLECULAR_WEIGHT_SO2 * 1000,
```

```
                Pollutant::O3 => ($concentration *
self::MOLAR_VOLUME_O3) / self::MOLECULAR_WEIGHT_O3 * 1000,
                Pollutant::NO2 => ($concentration *
self::MOLAR_VOLUME_NO2) / self::MOLECULAR_WEIGHT_NO2 * 1000,
                default => throw new
InvalidArgumentException('Pollutant not supported: ' . $pollutant-
>value),
            };
        }

        public function microgramsPerCubitMeterToPpb(Pollutant
$pollutant, float $concentration): float
        {
            return $this->microgramsPerCubicMeterToPpm($pollutant,
$concentration) * 1000;
        }
    }

    final class WeatherApiIntegration
    {
        private const string ACTION_CURRENT = '/current.json';

        public function __construct(
            private readonly Client $httpClient,
            private readonly PromiseUtils $promiseUtils,
            private readonly ResponseUtils $responseUtils,
            private readonly CitiesRepository $citiesRepository
        ) {
        }

        public function getAirQualityForMany(Collection
$coordinates): Generator
        {
            $promises = [];
            $url = config('weather_api.base_url') .
self::ACTION_CURRENT;
            $apiKey = config('weather_api.api_key');
            foreach ($coordinates as $coordinate) {
                $promises[] = $this->httpClient->getAsync($url, [
                    'query' => [
                        'q' => $coordinate->latitude . ',' .
$coordinate->longitude,
```

```
                    'aqi' => 'yes',
                    'key' => $apiKey,
                ],
            ]);
        }

        $fulfilledPromises = Utils::settle($promises)->wait();

        return $this->collectFulfilled($fulfilledPromises);
    }
}
```

Більше прикладів коду системи наведено в розділі ДОДАТКИ.

Після розробки серверної частини розроблено клієнтську частину, зареєстровано чат-бота Telegram.

В рамках клієнтської частини розроблено інтерактивну мапу з мітками, які показують відповідне значення обраного індексу. Приклад сторінки наведений на рис. 3.8.

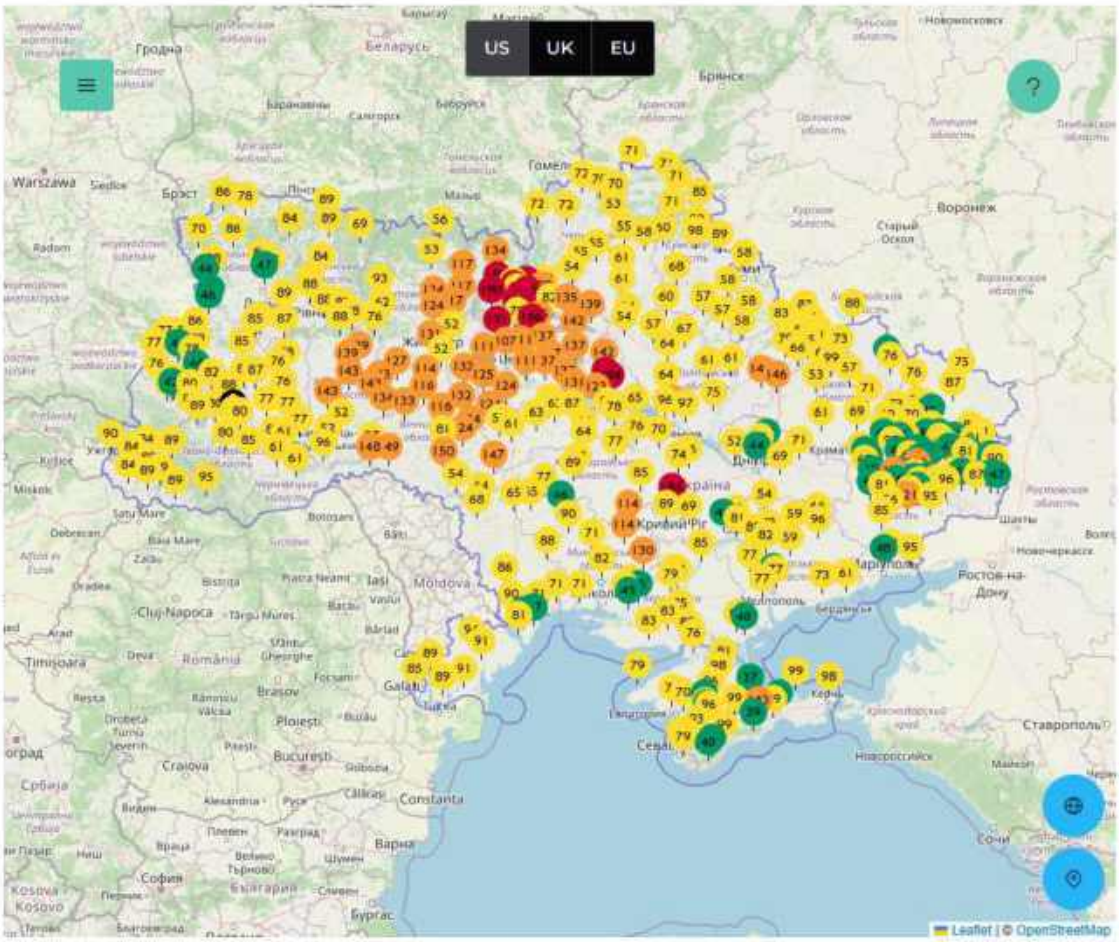

Рисунок 3.8 – Інтерактивна мапа проєкту з обраним US-індексом

Наступним розроблено сторінку «про проєкт» (About), в якому наведено короткий опис проєкту, доступні індекси та посилання на їх детальний опис, QR-код з посиланням на Telegram чат-бот, список підтримуваних міст України а також посилання на інструменти, що використовуються в проєкті. Вигляд сторінки наведений на рис. 3.9.

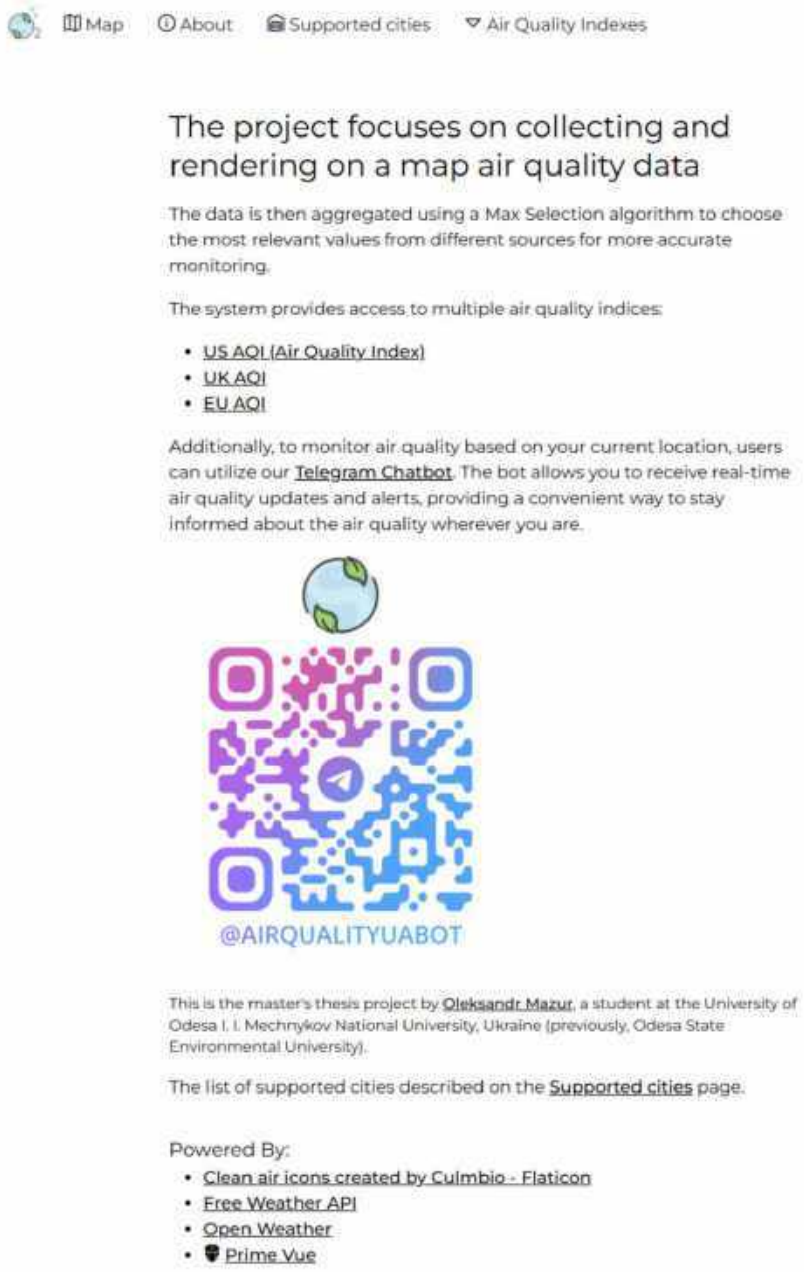

Рисунок 3.9 – Вигляд сторінки About

Після цього розроблено сторінку зі списком підтримуваних міст, результат показаний на рис. 3.10:

Рисунок 3.10 – Сторінка зі списком міст України, для яких доступний моніторинг якості проєкту

Кожен елемент в списку міст є посиланням на сторінку з детальною інформацією по обраному місту. На цій сторінці вказано дату й час останнього

оновлення даних, поточні значення індексів якості повітря з детальними показниками забруднювачів. Приклад такої сторінки наведений на рис. 3.11.

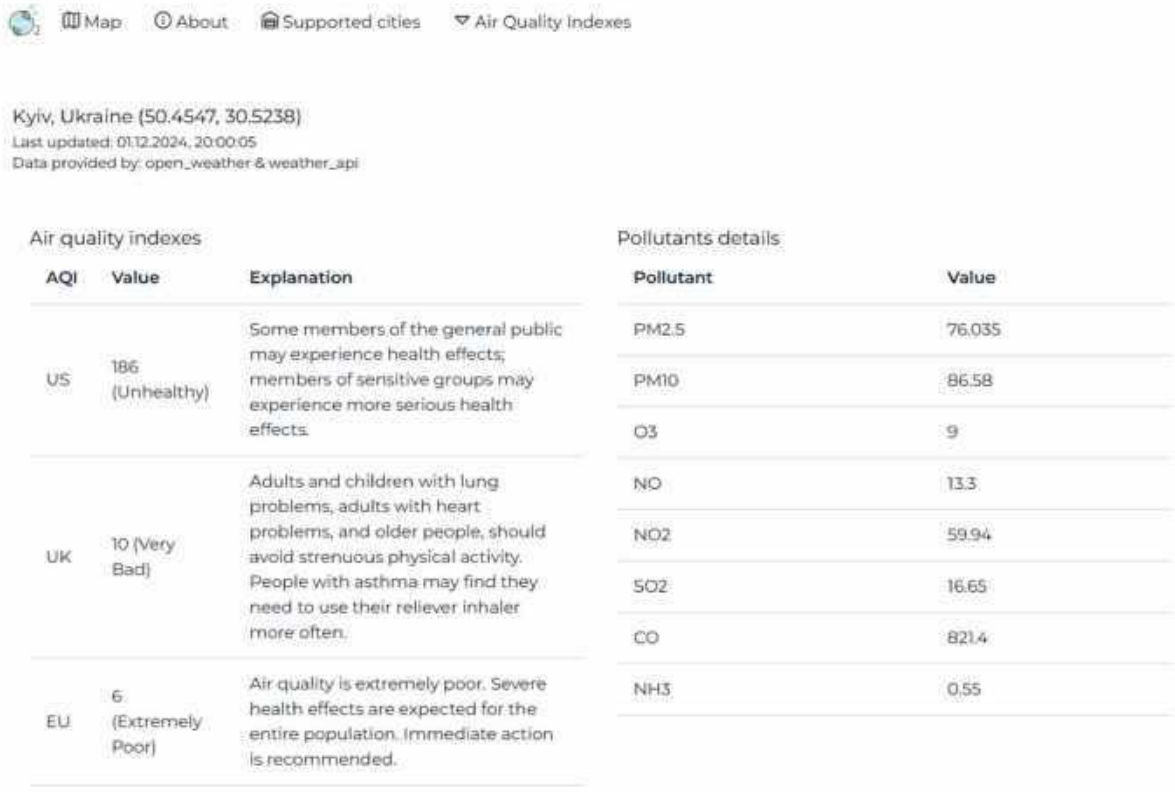

Рисунок 3.11 – Сторінка з детальною інформацією по обраному місту

Після цього було створено Telegram чат-бот, посилання на який вказано на сторінці About (рис. 3.9).

Чат бот запрошує місцезнаходження користувача та після його надання спочатку одразу відповідає з поточними даними даними по найближчому до користувача місту, а потім щогодини пише поточний стан повітря.

Приклад роботи чат-боту наведено нижче на рисунку 3.12.

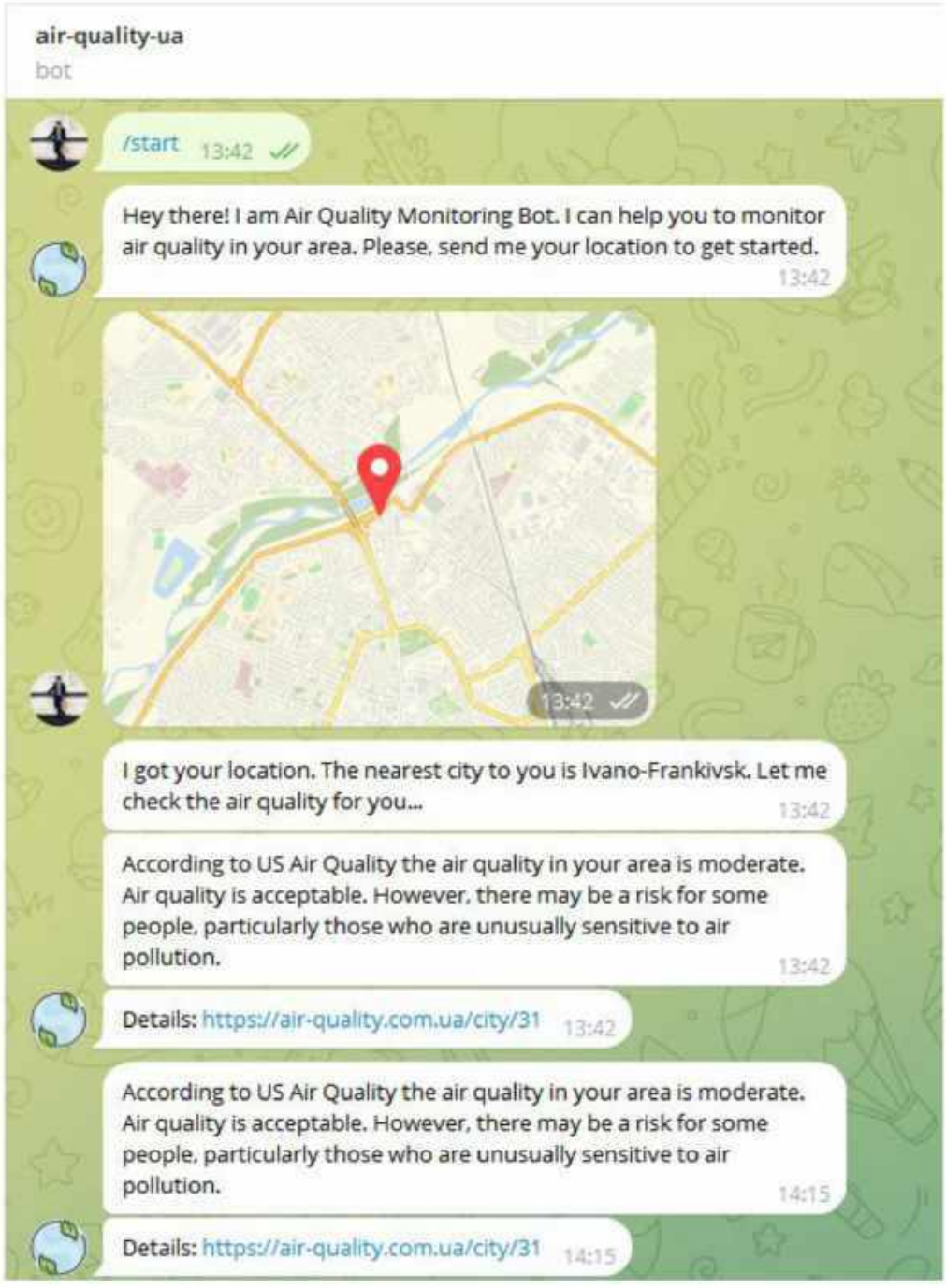

Рисунок 3.12 – Приклад роботи розробленого Telegram чат-боту

3.6 Висновки до розділу 3

Розробка інформаційної системи моніторингу якості повітря потребує ретельного проектування та моделювання її компонентів. У третьому розділі визначено ключові вимоги до системи, спроектовано базу даних, а також описано програмну архітектуру та основні модулі, які забезпечують її функціональність, та реалізовано програмні модулі, необхідні для повноцінного функціонування системи.

Загальні вимоги до системи дозволяють створити інструмент, який відповідає потребам кінцевих користувачів і сучасним стандартам у сфері моніторингу якості повітря. База даних спроектована таким чином, щоб забезпечити зберігання й обробку даних про забруднювачі повітря, індекси якості, а також геолокаційної інформації користувачів, що дає змогу надавати індивідуальні сповіщення через Telegram чат-бот.

Архітектура системи побудована з використанням підходу MVC, що дозволяє чітко розділити відповідальність між різними компонентами. Це забезпечує легкість у підтримці, модифікації та розширенні функціональності системи. У рамках архітектури виділено такі важливі модулі, як:

- модулі інтеграції агрегації даних, що дозволяє об'єднувати інформацію з кількох джерел і забезпечувати її точність за допомогою алгоритму Max Selection;
- модуль роботи з Telegram чат-ботом, який забезпечує інтерактивну взаємодію з користувачами, включаючи обробку запитів і надсилання сповіщень;
- модуль обчислення індексів якості повітря, який відповідає за реалізацію алгоритмів розрахунку трьох різних типів індексів (US, UK, EU);
- модуль конвертації одиниць вимірювання, що гарантує точність даних, отриманих із різних джерел.

Проектування бази даних та логіки додатка орієнтовано на забезпечення зручності інтеграції нових провайдерів даних, масштабованості та захисту даних користувачів. Використання перевірених бібліотек і фреймворків, таких як Laravel, Vue.js, Leaflet.js та Telegram SDK, дозволяє зменшити витрати часу на розробку та забезпечити високу якість системи.

Розробка застосунку включала створення серверної та клієнтської частин, інтеграцію API провайдерів даних, модуль агрегації та конвертації одиниць вимірювання, а також обчислення індексів якості повітря. Система підтримує три індекси якості повітря: US, UK та EU. Особливу увагу було приділено інтеграції з Telegram ботом, який забезпечує інформування користувачів у реальному часі.

Для візуалізації даних було реалізовано інтерактивну мапу з мітками, які відображають індекси забруднення, а також створено зручний інтерфейс для перегляду даних про міста та їхні детальні показники. Додатково розроблено сторінку "Про проєкт", що містить загальний опис системи, підтримувані індекси, список міст та посилання на Telegram бот.

Завдяки спроектованій архітектурі та реалізованим модулям, система демонструє високу функціональність, масштабованість та зручність у використанні. Вона відповідає сучасним вимогам до моніторингу якості повітря та надає користувачам необхідну інформацію для оцінки стану екології.

Цей етап розробки став завершальним у створенні прототипу системи, який може бути основою для подальшого розширення, оптимізації та впровадження у виробниче середовище.

4 ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ ТА НАПРЯМКИ ПОКРАЩЕННЯ

4.1 Відповідність реалізованих функцій початковим вимогам

В процесі розробки системи моніторингу якості повітря були поставлені вимоги, що визначали основні функціональні та технічні можливості системи. У цьому підрозділі розглянемо, наскільки реалізовані функції відповідають цим вимогам.

Система реалізує інтерактивну мапу, яка відображає індекси якості повітря для 400 міст України. Користувачі можуть переглядати рівень забруднення у вибраному місті, а також змінювати тип індексу (US, UK, EU) через інтерфейс. Реалізована мапа відповідає вимогам щодо зручності використання, інтерактивності та відображення даних у реальному часі.

На окремих сторінках для кожного міста відображаються детальні дані про якість повітря, включаючи значення забруднювачів та відповідні індекси. Інформація оновлюється відповідно до даних, отриманих із зовнішніх API.

Система надає користувачам Telegram-чатбот, який:

1) отримує геолокацію користувача;
2) визначає найближче місто за координатами;
3) надсилає сповіщення про якість повітря в цьому місті щогодини.

Чатбот працює стабільно, забезпечує інтерактивність та відповідає вимогам щодо спрощеного доступу до інформації.

Система дозволяє користувачам використовувати функцію геолокації браузера для автоматичного визначення найближчого до них міста. Після отримання координат мапа автоматично збільшує масштаб, щоб сфокусуватися на місцезнаходженні користувача.

Реалізовано механізм агрегації даних із двох обраних джерел –

OpenWeather API та Weather API. Дані інтегруються методом Max Selection, що забезпечує відображення найбільш критичних показників для кожного міста.

Система відповідає вимогам безпеки:

1) усі дані передаються через захищений протокол HTTPS;
2) доступ до сервера обмежений через SSH із використанням OpenSSH;
3) доступ до бази даних із зовнішньої мережі заборонений.

Система надає користувачам сторінку «Про проєкт», яка містить опис основних функцій, список підтримуваних міст, детальні пояснення індексів якості повітря, а також посилання на Telegram чат-бот.

Розроблена система повністю відповідає початковим функціональним вимогам. Вона забезпечує візуалізацію даних на мапі, доступ до детальної інформації, інтеграцію з Telegram чат-ботом, підтримку геолокації та безпеку даних. Це робить її ефективним інструментом для моніторингу якості повітря.

4.2 Аналіз ефективності системи в порівнянні з системою SaveEcoBot

Метод оцінки ефективності через порівняння базується на аналізі ключових характеристик досліджуваної системи щодо аналогічних рішень. Його суть полягає у визначенні сильних і слабких сторін системи через зіставлення з конкурентами або існуючими стандартами.

Для проведення такого аналізу спочатку визначаються параметри, які найбільше впливають на ефективність роботи системи. Це можуть бути функціональність, зручність використання, швидкість виконання операцій тощо.

На основі отриманих результатів формуються висновки щодо відповідності системи заявленим вимогам і очікуванням. Якщо виявлено аспекти, де система поступається конкурентам, це може стати основою для подальшого вдосконалення. Такий підхід дозволяє об’єктивно оцінити досягнуті результати та

визначити можливості розвитку.

Нижче, в табл. 4.1 приведені основні відмінності між системи.

Таблиця 4.1 – Відмінності систем по обраним показникам

Показники	Розроблена система	SaveEcoBot
Функціонал	Основний акцент на відображенні якості повітря для різних міст України в реальному часі.	Більш широкий функціонал, орієнтований не лише на моніторинг якості повітря, а й на екологічну обізнаність. Крім якості повітря, охоплює інші екологічні аспекти, такі як пункти прийому вторсировини, небезпечних відходів тощо.
Інтерфейс	Мінімалістичний та інтуїтивно зрозумілий інтерфейс.	Багатофункціональний інтерфейс, що може виглядати перевантаженим для нових користувачів.
Індекси якості повітря	Показує міжнародні визнані індекси якості повітря для міст України.	Показує індекси, що основані здебільшого тільки на показникам мікрочастинок ($PM_{2.5}$, PM_{10})
Географічне покриття	Надає дані по 400 найбільших містах України	Надає дані по значно меншій кількості міст, проте присутні найбільш густонаселені міста. Найбільші міста мають більше точок даних в самому місті.
Додатковий функціонал	Мінімум додаткових функцій. Сфокусований лише на якості повітря.	Окрім моніторингу повітря, надає екологічну інформацію, таку як: пункти утилізації відходів, місця збору батарейок, ламп, моніторингу радіаційного фону.
Джерела даних	Лише два великих постачальники даних з невідомою точністю.	Багато мілких постачальників, включаючи дані від міськрад та власних сканерів.

Порівнюючи розроблену систему та SaveEcoBot, можна відзначити, що кожна з них має свої унікальні особливості та підходи до моніторингу якості повітря. Розроблена система зосереджена на відображенні актуальних даних про якість повітря для міст України, забезпечуючи простий і зрозумілий інтерфейс. Вона використовує міжнародно визнані індекси якості повітря, що дозволяє користувачам оцінювати стан повітря на основі універсальних стандартів. Географічне покриття охоплює 400 найбільших міст України, що є її значною перевагою, однак система має обмежений набір джерел даних.

Натомість SaveEcoBot пропонує ширший функціонал, орієнтований не лише на моніторинг якості повітря, а й на підвищення екологічної обізнаності. Інтерфейс системи більш насичений функціями, що може бути складним для новачків, але надає додаткову екологічну інформацію, таку як пункти утилізації відходів чи моніторинг радіаційного фону. SaveEcoBot співпрацює з численними джерелами даних, включаючи міські ради та власні датчики, що забезпечує детальнішу картину, особливо для густонаселених міст.

Таким чином, розроблена система є більш вузькоспеціалізованою, сфокусованою на моніторингу якості повітря, тоді як SaveEcoBot пропонує більш універсальне екологічне рішення з розширеним набором функцій.

4.3 Напрямки вдосконалення системи

Покращення системи моніторингу якості повітря сприяє її подальшому розвитку та адаптації до змінних потреб користувачів і технологічного прогресу. Впровадження нових функцій, інтеграція додаткових джерел даних, розширення географічного покриття та підвищення точності розрахунків дозволяють зробити систему більш ефективною, гнучкою та конкурентоспроможною.

У цьому розділі буде розглянуто можливі варіанти вдосконалення, які

включають додавання нових функцій, інтеграцію з іншими системами, розширення географії тощо.

4.3.1 Аналіз можливостей розвитку функціоналу

Додавання нових функцій до системи може значно підвищити її функціональність та привабливість для користувачів. Одним із напрямків є інтеграція додаткових джерел даних. Наприклад, можна додати підтримку додаткових API, які пропонують інформацію про якість повітря з інших регіонів або джерел, таких як дані від міськрад чи локальних сенсорів, а також партнерство з платформами на зразок ЛУН.Місто. Це дозволить забезпечити ще точніші та більш деталізовані дані для моніторингу.

Ще одним перспективним напрямком є розширення комунікаційних можливостей шляхом впровадження нових чат-ботів. Окрім Telegram, система може бути інтегрована з платформами Facebook Messenger, WhatsApp чи Viber, що зробить її доступною для ширшого кола користувачів і дозволить отримувати дані про якість повітря безпосередньо через зручні для них канали.

Історичний аналіз є важливим інструментом для користувачів, які хочуть відстежувати динаміку змін якості повітря. Додавання можливості переглядати дані за останні 48 годин або 30 днів створить додаткову цінність для системи. Це допоможе зрозуміти, як змінюються показники, і, за потреби, коригувати свої плани чи заходи.

Покращення інтерфейсу також відіграє важливу роль у розвитку системи. Додавання зручної форми для швидкого пошуку міста безпосередньо на мапі зробить використання системи інтуїтивно зрозумілим навіть для нових користувачів. Крім того, створення браузерного віджету, який показуватиме якість повітря залежно від місцезнаходження користувача або обраного в налаштуваннях міста,

дозволить інтегрувати дані системи у зовнішні веб-сайти та застосунки.

Ще одним кроком є імплементація додаткових типів індексів якості повітря, таких як Індійський чи Китайський. Це зробить систему більш універсальною для користувачів, які цікавляться глобальними стандартами та хочуть порівнювати різні методики оцінки якості повітря.

Реалізація цих функцій дозволить не лише покращити якість наданих послуг, але й зробить систему більш гнучкою та пристосованою до зростаючих потреб користувачів.

4.3.2 Можливості інтеграції з іншими системами

Одним із напрямків розвитку інформаційної системи моніторингу якості повітря є створення відкритого програмного інтерфейсу (API), який дозволить їй виступати агрегатором і надавачем даних для сторонніх систем. Такий підхід дозволить значно розширити межі використання системи та зробить її важливим джерелом екологічної інформації для різних організацій, платформ і розробників.

API може надавати доступ до даних про якість повітря в реальному часі для конкретних географічних координат або міст. Ключовою функцією буде агрегування даних з декількох джерел, які вже інтегровані у систему, і надання уніфікованої та стандартизованої інформації. Це дозволить стороннім системам отримувати зведені дані без необхідності інтегруватися з кожним провайдером окремо.

Зокрема, API може підтримувати запити для отримання різних типів індексів якості повітря (наприклад, US AQI, UK AQI, EU AQI) з можливістю вибору параметрів, таких як тип індексу, місто або часовий інтервал.

Для забезпечення інтеграції з іншими системами, API підтримуватиме формати передачі даних, які є стандартом у сучасній веб-розробці, зокрема JSON та

XML. Це дозволить легко інтегрувати дані в мобільні застосунки, веб-портали чи аналітичні платформи. API також може підтримувати функціонал підписки на оновлення даних через вебхуки. У цьому випадку зовнішня система отримуватиме повідомлення про зміни в даних, наприклад, про перевищення допустимих норм якості повітря в конкретному місті.

Завдяки цьому інформація про якість повітря стане доступною ширшій аудиторії, що сприятиме підвищенню екологічної обізнаності та поліпшенню умов життя.

4.3.3 Перспективи розширення географії системи

Розширення географії інформаційної системи моніторингу якості повітря сприяє її розвитку, що дозволить охопити ширше коло користувачів та підвищити її цінність як інструмента екологічного моніторингу. Збільшення кількості міст у системі є природним кроком на шляху до створення більш повної та детальної картини якості повітря в Україні. Це можливо завдяки інтеграції додаткових джерел даних, таких як локальні екологічні ініціативи, дані міськрад чи сторонні API.

Для забезпечення доступності даних у регіонах, де немає точних вимірювань, можна використовувати методи інтерполяції. Наприклад, за допомогою просторової інтерполяції можливо розрахувати значення забруднювачів для міст чи місцевостей, що розташовані між кількома точками з вимірами. Такі підходи дозволять покрити більшу територію навіть за обмеженої кількості сенсорів або даних від провайдерів, зберігаючи при цьому достатню точність.

Крім того, є можливість обчислення "загальних" показників якості повітря для крупніших адміністративних одиниць, таких як райони чи області. Для цього можна використовувати усереднення даних або підходи на основі вагового коефіцієнта залежно від чисельності населення чи площі населених пунктів.

Наприклад, рівень забруднення для області може обчислюватися на основі даних з усіх міст, що входять до її складу, із коригуванням на частку кожного міста в загальній чисельності населення.

Такий функціонал дозволить системі надавати інформацію для регіональних органів влади та екологічних організацій, сприяючи прийняттю рішень на рівні адміністративних одиниць. Інформація про середні показники для районів чи областей також буде корисною для створення звітів, проведення наукових досліджень або інформаційних кампаній, спрямованих на підвищення екологічної обізнаності населення.

Розширення географії системи таким чином забезпечить її більшу адаптивність до потреб користувачів, підвищить її функціональність та зробить вагомим інструментом для моніторингу екологічного стану на національному рівні.

4.4 Висновки до розділу 4

Розділ 4 підсумовує оцінку ефективності розробленої системи та можливі напрямки її вдосконалення. У підрозділі 4.1 було доведено, що система повністю відповідає початковим вимогам. Реалізовані функції забезпечують повний спектр запланованих можливостей, таких як інтерактивна мапа, відображення детальної інформації про якість повітря, інтеграція з Telegram чат-ботом і використання геолокації. Крім того, система відповідає вимогам безпеки та стабільності.

Порівняння розробленої системи з аналогічною платформою SaveEcoBot у підрозділі 4.2 показало, що обидві системи мають свої унікальні переваги. Розроблена система є більш спеціалізованою, зосередженою на моніторингу якості повітря, тоді як SaveEcoBot надає ширший спектр екологічних даних. Аналіз допоміг визначити сильні та слабкі сторони системи, що дозволяє планувати подальші вдосконалення.

У підрозділі 4.3 були розглянуті перспективи розвитку функціоналу. Зокрема, інтеграція додаткових джерел даних, впровадження нових чат-ботів, додавання історичних даних, покращення інтерфейсу та підтримка нових індексів можуть значно підвищити цінність системи для користувачів. Окремо було акцентовано на створенні API, який дозволить системі виступати агрегатором і постачальником даних для інших платформ.

Розширення географії системи, розглянуте в підрозділі 4.4, відкриває нові можливості для охоплення більшої аудиторії. Використання методів інтерполяції та обчислення показників для адміністративних одиниць, таких як райони та області, дозволить забезпечити доступність даних навіть у регіонах із низькою щільністю сенсорів. Такі зміни підвищать значущість системи на регіональному та національному рівнях.

Таким чином, проведений аналіз підтвердив ефективність розробленої системи, водночас окресливши перспективи для її подальшого вдосконалення. Це створює можливість для більш широкого використання системи та підвищення її впливу на екологічну обізнаність населення.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено сучасну інформаційну систему моніторингу якості повітря, яка відповідає актуальним вимогам до екологічних технологій. Дана система базується на передових програмних рішеннях, забезпечує точність і надійність збору даних, а також зручність використання для кінцевих користувачів. У роботі висвітлено всі ключові аспекти створення системи, починаючи від аналізу предметної області та моделювання, і завершуючи розробкою функціональних модулів та інтеграцією з існуючими технологіями.

Актуальність роботи обумовлена необхідністю підвищення обізнаності населення щодо якості повітря, особливо в умовах урбанізації, збільшення кількості транспортних засобів та інших факторів, що впливають на рівень забруднення. Були досліджені основні забруднювачі повітря, їх вплив на здоров'я людей і навколишнє середовище, а також методи моніторингу та розрахунку індексів якості повітря, що стало основою для створення системи.

Проектування системи базувалося на чітко визначених вимогах, серед яких – інтеграція з джерелами даних, обробка отриманої інформації, обчислення індексів якості повітря та їх наочне відображення для користувачів. Було розроблено інтуїтивний інтерфейс, що дозволяє користувачам швидко знаходити необхідну інформацію, а також Telegram чат-бот, який забезпечує оперативний доступ до даних через популярний месенджер. Важливою особливістю є використання інтерактивної мапи, що надає можливість переглядати показники якості повітря для понад 400 міст України.

Система базується на сучасній архітектурі MVC, що забезпечує чіткий поділ відповідальності між її компонентами. Для роботи з базою даних було реалізовано високорівневий модуль, який спрощує доступ до даних і забезпечує їх

структуроване зберігання. Інтеграція з Open Weather API та Weather API дозволила забезпечити високу актуальність і точність даних, що обробляються за допомогою методу Max Selection. Для підтримки різних типів індексів було створено модуль конвертації одиниць вимірювання, що враховує специфіку кожного типу даних.

Ефективність системи була проаналізована шляхом порівняння її функціоналу з аналогічними рішеннями, такими як SaveEcoBot. Результати аналізу показали, що розроблена система має чітке позиціонування, зосереджуючись на якості повітря та забезпечуючи користувачів простим і зрозумілим інструментом для отримання інформації. При цьому були визначені можливості для подальшого розвитку, зокрема інтеграція нових джерел даних, підтримка додаткових індексів, розширення географії та створення нових функціональних компонентів, таких як веб-віджети або чат-боти для інших платформ.

Розроблена система є перспективною платформою для впровадження в екологічну сферу України. Вона здатна сприяти підвищенню екологічної свідомості населення, підтримувати зусилля державних і місцевих органів у моніторингу стану повітря, а також сприяти покращенню якості життя. Запропоновані можливості для інтеграції з іншими системами відкривають перспективи для її розвитку в регіональних і міжнародних масштабах.

Таким чином, дипломна робота досягла своєї мети, а розроблена система відповідає сучасним викликам та вимогам у сфері моніторингу якості повітря. Вона є не лише ефективним інструментом для збирання та аналізу даних, але й зручним сервісом для користувачів, що дозволяє сприяти покращенню екологічної ситуації в Україні.

ПЕРЕЛІК ПОСИЛАНЬ

1) WHO – Billions of people still breathe unhealthy air: new WHO data [Електронний ресурс] – Режим доступу: https://www.who.int/news/item/04-04-2022-billions-of-people-still-breathe-unhealthy-air-new-who-data

2) Technical Assistance Document for the Reporting of Daily Air Quality – the Air Quality Index (AQI) [Електронне видання] / EPA-454/B-24-002 May 2024 – 26 с. – Режим доступу: https://document.airnow.gov/technical-assistance-document-for-the-reporting-of-daily-air-quailty.pdf

3) EcoCity Громадський моніторинг стану якості повітря [Електронний ресурс] – Режим доступу: https://eco-city.org.ua

4) Карта якості повітря ЛУН Місто AIR – [Електронний ресурс] – Режим доступу: https://misto.lun.ua/air

5) Єдина в Україні екологічна система – SaveEcoBot – [Електронний ресурс] – Режим доступу: https://saveecobot.com

6) Daily Air Quality Index – Defra, UK – [Електронний ресурс] – Режим доступу: https://uk-air.defra.gov.uk/air-pollution/daqi

7) European Air Quality Index Calculation – CAMS Training – [Електронний ресурс] – Режим доступу: https://ecmwf-projects.github.io/copernicus-training-cams/proc-aq-index.html

8) Current weather and forecast – OpenWeatherMap – [Електронний ресурс] – Режим доступу: https://openweathermap.org

9) Free Weather API – WeatherAPI.com – [Електронний ресурс] – Режим доступу: https://www.weatherapi.com

10) Wikipedia, вільна бібліотека [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org

11) Vue.js – The Progressive JavaScript Framework | Vue.js – [Електронний ресурс] – Режим доступу: https://vuejs.org

12) Laravel – The PHP Framework For Web Artisans – [Електронний ресурс] – Режим доступу: https://laravel.com

13) Leaflet – a JavaScript library for interactive maps – [Електронний ресурс] – Режим доступу: https://leafletjs.com

14) PrimeVue – Vue UI Component Library – [Електронний ресурс] – Режим доступу: https://primevue.org

15) Guzzle, PHP HTTP Client – Guzzle Documentation – [Електронний ресурс] – Режим доступу: https://docs.guzzlephp.org/en/stable

16) S.M. Shafi Environmental Pollution. – Atlantic Publishers & Dist, 2005. – 456 с.

17) B. R. Gurjar; Luisa T. Molina; C. Shekhar P. Air Pollution: Health and Environmental Impacts. – Ojha, 2010. – 556 с.

18) Lynne Lewis, Thomas Tietenberg Environmental Economics and Policy. – Lynne Lewis, Thomas Tietenberg, 2019. – 382 с.

ДОДАТКИ

ДОДАТОК А КОД КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ

Нижче наведені основні коди графічних користувацьких інтерфейсів в системі.

А.1 Код графічного користувацького інтерфейсу сторінки з мапою

```
    <template>
        <MapMenu v-if="!showLegend" @closed="() => menuOpened =
false" @opened="() => menuOpened = true"/>
        <SelectButton v-if="!showLegend"
                      optionLabel="label"
                      optionValue="value"
                      dataKey="value"
                      :allow-empty="false"
                      v-model="aqi_index_type"
                      :options="aqi_index_types"
                      class="absolute"
                      style="z-index: 999999; left:50%;transform:
translate(-50%, -50%);top:40px"
                      @change="redrawMarkers">
            <template #option="slotProps">
                <p v-tooltip="slotProps.option.tooltip">{{
slotProps.option.label }}</p>
            </template>
        </SelectButton>
        <div id="leafletMap" class="h-screen w-full"></div>

        <Button icon="pi pi-question" rounded @click="() =>
showLegend = true" v-if="!showLegend && !menuOpened"
                style="position: absolute; z-index: 9999999;
right: 50px; top: 50px"/>
    </template>

    <script>
    import leaflet from 'leaflet';
    import {useGeolocation} from "@vueuse/core";
    import {Marker} from "../../types/Marker.js";
    import {toRaw} from "vue";
    import ukraine from "../../../geojson/ukraine-geoboundaries-
adm0.json";
    import MapMenu from "../components/MapMenu.vue";
```

```
import SelectButton from "primevue/selectbutton";
import Card from "primevue/card";
import Tooltip from "primevue/tooltip";
import Dialog from "primevue/dialog";
import TabPanel from "primevue/tabpanel";
import TabPanels from "primevue/tabpanels";
import Tab from "primevue/tab";
import TabList from "primevue/tablist";
import Tabs from "primevue/tabs";
import Button from "primevue/button";
import Avatar from "primevue/avatar";
import Column from "primevue/column";
import DataTable from "primevue/datatable";
import {AirQuality} from "../../types/AirQuality.js";

// Kyiv coordinates
const DEFAULT_LATITUDE = 50.450001;
const DEFAULT_LONGITUDE = 30.523333;

const AQI_INDEX_TYPE_US = 'aqi_us';
const AQI_INDEX_TYPE_UK = 'aqi_uk';
const AQI_INDEX_TYPE_EU = 'aqi_eu';

const {isSupported, coords, error} = useGeolocation()

export default {
    name: "AirQualityMap",
    components: {
        Avatar,
        MapMenu,
        Button,
        SelectButton,
        Card,
        Dialog,
        Tabs,
        TabList,
        Tab,
        TabPanels,
        TabPanel,
        DataTable,
        Column
    },
```

```
        directives: {tooltip: Tooltip},
        data() {
            return {
                showLegend: false,
                menuOpened: false,
                aqi_index_type: AQI_INDEX_TYPE_US,
                aqi_index_types: [
                    {label: 'US', value: AQI_INDEX_TYPE_US},
                    {label: 'UK', value: AQI_INDEX_TYPE_UK},
                    {label: 'EU', value: AQI_INDEX_TYPE_EU},
                ],
                rawMarkers: [],
                markersOnMap: [],
                map: null,
                border: null,
                userOnMap: null,
                userMarker: new Marker(DEFAULT_LATITUDE,
DEFAULT_LONGITUDE, Marker.TYPE_USER),
                geolocationCoords: coords,
                geolocationError: error,
            };
        },
        async mounted() {
            this.map = toRaw(leaflet.map('leafletMap',
{zoomControl: false, zoomAnimation: true}));

            const zoomToBordersFn = this.zoomToBorders;
            const zoomToUserFn = this.locateAndZoomIn;

            const zoomToBorderControl = L.Control.extend({
                options: {
                    position: 'bottomright'
                },

                onAdd: function (map) {
                    const btn = L.DomUtil.create('button',
'leaflet-bar leaflet-control leaflet-control-custom');
                    btn.innerHTML = '<i class="pi pi-globe"></i>';
                    btn.className = 'p-button p-button-rounded p-
button-info w-14 h-14';
                    btn.title = 'Zoom to borders';
                    btn.onclick = function () {
```

```
                        zoomToBordersFn();
                    }

                    return btn;
                }
            });
            const currentLocationControl = L.Control.extend({
                options: {
                    position: 'bottomright'
                },

                onAdd: function (map) {
                    const btn = L.DomUtil.create('button',
'leaflet-bar leaflet-control leaflet-control-custom');
                    btn.innerHTML = '<i class="pi pi-map-
marker"></i>';
                    btn.className = 'p-button p-button-rounded p-
button-info w-14 h-14';
                    btn.title = 'Locate me';
                    btn.onclick = function () {
                        zoomToUserFn();
                    }

                    return btn;
                }
            });

            this.map.addControl(new currentLocationControl());
            this.map.addControl(new zoomToBorderControl());

            leaflet

.tileLayer("https://tile.openstreetmap.org/{z}/{x}/{y}.png", {
                    maxZoom: 19,
                    attribution:
                        '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>',
                })
                .addTo(this.map);

            this.border = leaflet
                .geoJSON(ukraine, {
```

```
                    style: {
                        opacity: .8,
                        color: "#595cef",
                        fillOpacity: .03,
                        weight: 1,
                    }
                })
                .addTo(this.map);

            await this.fillMarkers();

            this.zoomToBorders();
        },
        methods: {
            locateAndZoomIn() {

this.map.setView([parseFloat(this.geolocationCoords.latitude),
parseFloat(this.geolocationCoords.longitude)], 10);
            },
            zoomToBorders() {
                this.map.fitBounds(this.border.getBounds());
            },
            placeUserMarker() {
                if (this.userOnMap) {
                    this.map.removeLayer(this.userOnMap);
                }

                this.userOnMap = leaflet
                    .marker([this.geolocationCoords.latitude,
this.geolocationCoords.longitude], {
                        icon: leaflet.icon({
                            iconUrl: this.userMarker.getIcon(),
                            iconSize: [25, 41],
                            shadowSize: [50, 64],
                            iconAnchor: [12, 41],
                            shadowAnchor: [4, 62],
                        }),
                    })
                    .addTo(this.map)
                    .bindPopup('You are here');
            },
            redrawMarkers() {
```

```
                if (!this.aqi_index_type) {
                    return;
                }

                this.markersOnMap.forEach(marker => {
                    this.map.removeLayer(marker);
                });

                this.markersOnMap = [];
                this.drawMarkers(this.rawMarkers.map((marker) =>
new Marker(
                    marker.latitude,
                    marker.longitude,
                    Marker.TYPE_AIR_QUALITY,
                    new AirQuality(
                        marker.provider,
                        marker.pm10,
                        marker.pm2_5,
                        marker.nh3,
                        marker.o3,
                        marker.no,
                        marker.no2,
                        marker.so2,
                        marker.co,
                        marker.created_at,
                    ),
                    this.aqi_index_type,
                    marker[this.aqi_index_type]
                )));
            },
            drawMarkers(markers) {
                markers.forEach(marker => {
                    const leafletMarker = leaflet
                        .marker([marker.latitude,
marker.longitude], {
                            icon: leaflet.icon({
                                iconUrl: marker.getIcon(),
                                iconSize: [25, 41],
                                shadowSize: [50, 64],
                                iconAnchor: [12, 41],
                                shadowAnchor: [4, 62],
                            }),
```

```
                        })
                        .addTo(this.map);
                    this.markersOnMap.push(leafletMarker);
                    leafletMarker
                        .bindTooltip(
                            `<div class="marker-tooltip">
    <p></p><strong>Provider</strong>:
${marker.airQuality.provider}</p>
    <p></p><strong>PM10</strong>: ${marker.airQuality.pm10}</p>
    <p></p><strong>PM2.5</strong>: ${marker.airQuality.pm2_5}</p>
    <p></p><strong>NH3</strong>: ${marker.airQuality.nh3}</p>
    <p></p><strong>O3</strong>: ${marker.airQuality.o3}</p>
    <p></p><strong>NO</strong>: ${marker.airQuality.no}</p>
    <p></p><strong>NO2</strong>: ${marker.airQuality.no2}</p>
    <p></p><strong>SO2</strong>: ${marker.airQuality.so2}</p>
    <p></p><strong>CO</strong>: ${marker.airQuality.co}</p>
    <p></p><strong>Updated at</strong>: ${new
Date(marker.airQuality.updated_at).toLocaleString()}</p>
    </div>`,
                            {
                                permanent: false,
                                interactive: false,
                                direction: 'top',
                                className: 'marker-tooltip',
                            }
                        );
                });
            },
            async fillMarkers() {
                await this.loadMarkers().then(markers => {
                    this.drawMarkers(markers);
                });
            },
            async loadMarkers() {
                try {
                    const response = await axios.get('/current-
air-quality-indexes');
                    this.rawMarkers = response.data;
                    return response.data.map((marker) => new
Marker(
                        marker.latitude,
                        marker.longitude,
```

```
                        Marker.TYPE_AIR_QUALITY,
                        new AirQuality(
                            marker.provider,
                            marker.pm10,
                            marker.pm2_5,
                            marker.nh3,
                            marker.o3,
                            marker.no,
                            marker.no2,
                            marker.so2,
                            marker.co,
                            marker.created_at,
                        ),
                        this.aqi_index_type,
                        marker[this.aqi_index_type]
                    ));
                } catch (error) {
                    throw new Error('Error while fetching
markers');
                }
            }
        },
        watch: {
            geolocationCoords: {
                handler: function () {
                    this.placeUserMarker();
                },
                deep: true
            }
        },
    };
    </script>
```

ДОДАТОК Б КОДИ СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ

Нижче наведені основні коди серверної частини інформаційної системи

Б.1 Код обрахунку значень Американського індексу якості повітря

```
<?php

declare(strict_types=1);

namespace Mazur\Application\AirQuality\AqiCalculator;

use Mazur\Application\AirQuality\ApiIntegrations\Dto\AirQuality;
use Mazur\Application\Converter\ConcentrationConverter;

final class UsCalculator implements CalculatorInterface
{
    private const array SCALE = [
        Pollutant::O3->value => [
            [0, 50, .0, .054],
            [51, 100, .055, .070],
            [101, 150, .125, .164],
            [151, 200, .165, .204],
            [201, 300, .205, .404],
            [301, 500, .405, .604],
        ],
        Pollutant::PM2_5->value => [
            [0, 50, .0, 9.0],
            [51, 100, 9.1, 35.4],
            [101, 150, 35.5, 55.4],
            [151, 200, 55.5, 125.4],
            [201, 300, 125.5, 225.4],
            [301, 500, 225.5, 325.4],
        ],
        Pollutant::PM10->value => [
            [0, 50, 0, 54],
            [51, 100, 55, 154],
            [101, 150, 155, 254],
            [151, 200, 255, 354],
            [201, 300, 355, 424],
            [301, 500, 425, 604],
        ],
        Pollutant::CO->value => [
```

```
            [0, 50, .0, 4.4],
            [51, 100, 4.5, 9.4],
            [101, 150, 9.5, 12.4],
            [151, 200, 12.5, 15.4],
            [201, 300, 15.5, 30.4],
            [301, 500, 30.5, 50.4],
        ],
        Pollutant::SO2->value => [
            [0, 50, 0, 35],
            [51, 100, 36, 75],
            [101, 150, 76, 185],
            [151, 200, null, null],
            [201, 300, null, null],
            [301, 500, null, null],
        ],
        Pollutant::NO2->value => [
            [0, 50, 0, 53],
            [51, 100, 54, 100],
            [101, 150, 101, 360],
            [151, 200, 361, 649],
            [201, 300, 650, 1249],
            [301, 500, 1250, 2049],
        ],
    ];

    public function __construct(private readonly
ConcentrationConverter $concentrationConverter)
    {
    }

    /** @throws PollutantNotFoundException */
    public function calculate(AirQuality $airQuality): int
    {
        return (int)round(
            max(
                ...array_filter([
                    $this->calculateAqiForPollutant(
                        (int)$this->concentrationConverter-
>microgramsPerCubitMeterToPpb(Pollutant::SO2, $airQuality->so2),
                        Pollutant::SO2
                    ),
                    $this->calculateAqiForPollutant(
```

```
                        (int)$this->concentrationConverter-
>microgramsPerCubitMeterToPpb(Pollutant::NO2, $airQuality->no2),
                        Pollutant::NO2
                    ),
                    $this-
>calculateAqiForPollutant(floor($airQuality->pm2_5 * 10) / 10,
Pollutant::PM2_5),
                    $this-
>calculateAqiForPollutant((int)$airQuality->pm10, Pollutant::PM10),
                    $this->calculateAqiForPollutant(
                        floor(
                            $this->concentrationConverter-
>microgramsPerCubicMeterToPpm(Pollutant::O3, $airQuality->o3)
                            * 1000
                        ) / 1000,
                        Pollutant::O3
                    ),
                    $this->calculateAqiForPollutant(
                        floor(
                            $this->concentrationConverter-
>microgramsPerCubicMeterToPpm(Pollutant::CO, $airQuality->co)
                            * 10
                        ) / 10,
                        Pollutant::CO
                    ),
                ])
            )
        );
    }

    private function calculateAqiForPollutant(float $concentration,
Pollutant $pollutant): ?float
    {
        if (!array_key_exists($pollutant->value, self::SCALE)) {
            throw new PollutantNotFoundException('Pollutant not
found: ' . $pollutant->name);
        }

        $scale = self::SCALE[$pollutant->value];
        $Ihi = 500;
        $Ilo = 301;
        $BPlo = null;
```

```
        $BPhi = null;
        foreach ($scale as $row) {
            [$Ihi, $Ilo, $BPlo, $BPhi] = $row;
            if ($BPlo === null || $BPhi === null) {
                // AQI can not be calculated for this pollutant
                return null;
            }

            if ($concentration >= $BPlo && $concentration <= $BPhi)
{
                break;
            }
        }

        return ($Ihi - $Ilo) / ($BPhi - $BPlo) * ($concentration -
$BPlo) + $Ilo;
    }

    public function getStringRepresentation(int $index):
IndexStringRepresentation
    {
        $indexStr = match (true) {
            $index >= 0 && $index <= 50 => 'Good',
            $index >= 51 && $index <= 100 => 'Moderate',
            $index >= 101 && $index <= 150 => 'Unhealthy for
Sensitive Groups',
            $index >= 151 && $index <= 200 => 'Unhealthy',
            $index >= 201 && $index <= 300 => 'Very Unhealthy',
            $index >= 301 => 'Hazardous',
            default => 'Unknown',
        };

        $description = match (true) {
            $index >= 0 && $index <= 50 => 'Air quality is
considered satisfactory, and air pollution poses little or no
risk.',
            $index >= 51 && $index <= 100 => 'Air quality is
acceptable. However, there may be a risk for some people,
particularly those who are unusually sensitive to air pollution.',
            $index >= 101 && $index <= 150 => 'Members of sensitive
groups may experience health effects. The general public is less
likely to be affected.',
```

```
            $index >= 151 && $index <= 200 => 'Some members of the
general public may experience health effects; members of sensitive
groups may experience more serious health effects.',
            $index >= 201 && $index <= 300 => 'Health alert: The
risk of health effects is increased for everyone.',
            $index >= 301 => 'Health warning of emergency
conditions: everyone is more likely to be affected.',
            default => 'Unknown',
        };

        return new IndexStringRepresentation($indexStr,
$description);
    }
}
```

Б.2 Код інтеграції з провайдером даних WeatherAPI

```
    <?php

    namespace Mazur\Application\AirQuality\ApiIntegra-
tions\WeatherApi;

    /** @psalm-import-type _Promise from PromiseUtils */
    final class WeatherApiIntegration
    {
        private const string ACTION_CURRENT = '/current.json';

        public function __construct(
            private readonly Client $httpClient,
            private readonly PromiseUtils $promiseUtils,
            private readonly ResponseUtils $responseUtils,
            private readonly CitiesRepository $citiesRepository
        ) {
        }

        public function getAirQualityForMany(Collection $coordi-
nates): Generator
        {
            $promises = [];
            $url = config('weather_api.base_url') .
self::ACTION_CURRENT;
            $apiKey = config('weather_api.api_key');
            foreach ($coordinates as $coordinate) {
```

```
                $promises[] = $this->httpClient->getAsync($url, [
                    'query' => [
                        'q' => $coordinate->latitude . ',' . $co-
ordinate->longitude,
                        'aqi' => 'yes',
                        'key' => $apiKey,
                    ],
                ]);
            }

            $fulfilledPromises = Utils::settle($promises)->wait();

            return $this->collectFulfilled($fulfilledPromises);
        }

        private function collectFulfilled(iterable $ful-
filledPromises): Generator
        {
            /** @var _Promise $promise */
            foreach ($fulfilledPromises as $promise) {
                if (!$this->promiseUtils->isFulfilled($promise)) {
                    continue;
                }

                /** @var Response $response */
                $response = $promise['value'];
                if (!$this->responseUtils->isOk($response)) {
                    continue;
                }

                $rawResponse = $response->getBody()->getCon-
tents();

                $arr = json_decode($rawResponse, true, 512,
JSON_THROW_ON_ERROR);

                $city = $this->citiesRepository->findCity(
                    $arr['location']['name'],
                    (float)$arr['location']['lat'],
                    (float)$arr['location']['lon']
                );
```

```
                if ($city === null) {
                    continue;
                }

                yield new AirQuality(
                    cityId: $city->id,
                    co: $arr['current']['air_quality']['co'] ??
.0,
                    no: $arr['current']['air_quality']['no'] ??
.0,
                    no2: $arr['current']['air_quality']['no2'] ??
.0,
                    o3: $arr['current']['air_quality']['o3'] ??
.0,
                    so2: $arr['current']['air_quality']['so2'] ??
.0,
                    pm2_5: $arr['current']['air_quality']['pm2_5']
?? .0,
                    pm10: $arr['current']['air_quality']['pm10']
?? .0,
                    nh3: $arr['current']['air_quality']['nh3'] ??
.0,
                );
            }
        }
    }
```