НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних наук

_____ Голуб Б.Л.

"_____" _____ 20 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Програмне забезпечення для інформаційної системи аналізу моніторингу якості повітря»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

К.т.н., доцент /Вайганг Г.О./

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

К.ф.-м.н., доцент /Британ А. В./

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав: _____ /Лагода Л.О./

підпис ПІБ

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"16" грудня 2024 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студенту Лагоді Леоніду Олексійовичу

Спеціальність 121 «Інженерія програмного забезпечення»

Тема роботи: Програмне забезпечення системи моніторингу екологічних показників

Затверджена наказом ректора НУБіП України № "2248С" від 16.12.2024

Термін подання завершеної роботи на кафедру 2025 . 05 . 25
рік, місяць, число

Вихідні дані до роботи: опис програмного забезпечення

Перелік питань що розглядаються:

1. Системний аналіз предметної області.
2. Проектування інформаційного та програмного забезпечення.
3. Розробка інформаційного та програмного забезпечення.
4. Рекомендації щодо впровадження та експлуатації системи.
5. Висновки

Керівник бакалаврської кваліфікаційної роботи _____ / Британ А.В./
підпис ініціали та прізвище

Завдання прийняв до виконання _____ / Лагода Л.О. /
підпис ініціали та прізвище

Дата отримання завдання 2025 . 02 . 20
рік, місяць, число

ЗМІСТ

ВСТУП

Моніторинг довкілля є надзвичайно актуальним науково-прикладний напрямом, який є також досить складним, багаторівневим, багатокомпонентним і потребує значних бюджетних витрат [1, 2]. Формування системи моніторингу довкілля в Україні перебуває у стані активного формування, розвитку, вдосконалення та уніфікації як методів отримання і переробки інформації про стан довкілля, так і методів навчання фахівців у цій галузі. Одночасно з цим потребують вдосконалення законодавча і навчально-методична база цієї системи [6, 7].

Як відомо, навколишнє природне середовище піддається природним впливам, що призводять до змін у ньому, зокрема, метеорологічних, сейсмічних, гравіметричних параметрів довкілля. Ці зміни потрібно фіксувати та досліджувати, цим займаються спеціальні служби. Для дослідження антропогенного впливу організовуються спеціальні спостереження за змінами стану біосфери під впливом людської діяльності.

Моніторинг довкілля є багатофункціональною інформаційною системою і включає наступні напрями його проведення:

1) спостереження за станом довкілля та факторами, які впливають на його або його окремі елементи;

2) оцінку фактичного стану всіх елементів довкілля та порівняння отриманих даних із нормативами;

3) прогноз стану довкілля і оцінку прогнозованого стану довкілля;

4) розробка науково-обґрунтованих рекомендацій для допомоги у прийнятті управлінських рішень державними або місцевими органами влади.

Рішення повинні бути направлені на запобігання негативним змінам стану окремих елементів довкілля та дотримання вимог екологічної безпеки.

Таким чином, моніторинг довкілля – це система спостережень, оцінки та прогнозу стану довкілля, яка забезпечує науково-інформаційну підтримку

прийняття управлінських рішень державними або місцевими органами влади.

Для управління станом навколишнього природного середовища необхідно сформувати повноцінну систему моніторингу, яка дозволить виявити критичні ситуації та фактори, які впливають на досліджувані природні системи та наявні індикатори, які є чутливими до цього впливу.

Тож з урахуванням цього мета бакалаврської кваліфікаційної роботи є розробка програмного забезпечення системи моніторингу екологічних показників. Основою при дослідженні рішень з моніторингу екологічних показників були підходи до самого моніторингу та архітектуру систем. Були визначені вимоги щодо функціоналу і на основі цього було розроблено прототип системи, який фіксував дані та зберігав їх. Тому основним завданням бакалаврської кваліфікаційної роботи є розробка програмного забезпечення системи моніторингу екологічних показників

Для досягнення поставленої теми були поставлені та вирішені наступні задачі :

1. Проаналізувати предметну область та найти інформацію щодо застосування програмних рішень у програмному продукті

2. Виявити та проаналізувати вже готові програми по цій темі

3. Сформулювати функціональні вимоги

4. Побудова UML діаграм

5. Обрати інструменти для розробки системи

6. Розробити інтерфейс

7. Розробити базу даних

8. Розробити програмний продукт

9. Провести тестування

Виходячи з цього основними завданнями при розробці є: створення програмного забезпечення, який надає змогу швидко та зручно запросити екологічні показники, надати можливість зберігати дані, та зручний для користувача інтерфейс.

Апробація програмного додатку відбувалася шляхом участі в науково-практичній конференції «Теоретичні та прикладні аспекти розробки комп’ютерних систем», що відбулася на базі факультету інформаційних технологій НУБіП, де було представлено доповідь на тему «ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ ЕКОЛОГІЧНИХ ПОКАЗНИКІВ ».

До складу роботи входить 65 сторінок, список з 21 джерел даних, 24 рисунки, та додатки А, Б, В.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ СФЕРИ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ ТА АНАЛІЗУ ЯКОСТІ ПОВІТРЯ

1.1 Системний аналіз предметної області

Моніторинг екологічних показників — система спостережень, оцінки і контролю за станом навколишнього середовища з метою розроблення заходів стосовно його охорони, раціонального використання природних ресурсів, попередження можливості виникнення критичних ситуацій, шкідливих або небезпечних для здоров'я людей, існування живих організмів, їх угрупувань, природних об'єктів. Основне завдання — спостереження за антропогенними змінами у природі, оцінка її фактичного стану і прогноз на майбутнє. Управління якістю природного середовища виходить за межі системи моніторингу. Поділяють, відповідно до просторово-часових параметрів процесів, які контролюються, на три класи систем локальний, регіональний, глобальний або відповідно до мети контролю біосферний. Локальна система— спостереження за антропогенним впливом на навколишнє середовище в особливо небезпечних зонах і місцях. Регіональна система — спостереження за процесами і явищами в межах певного регіону, де ці процеси та явища можуть відрізнятися і за природним характером, і за антропогенним впливом від базового фону, характерного для всієї біосфери. Глобальна система. — спостереження за загально планетними процесами і явищами в біосфері Землі, включаючи всі їх екологічні компоненти, попередження виникнення екстремальних ситуацій. Санітарна система — спостереження за станом якості природного середовища, ступенем забруднення середовища існування (наявності шумів, алергенів, пилу, патогенних мікроорганізмів тощо) та їх впливом на людину, тваринний і рослинний світ. Біосферна система —

система спостережень за природними процесами і явищами, що дозволяє визначати глобальні зміни фонових показників температури, радіоактивності, вмісту в атмосфері вуглекислого газу, озону тощо.

Основні екологічні показники що підлягають моніторингу [1]:

Температура повітря

Вологість

Атмосферний тиск

Забруднюючі речовини, такі як CO2, CO, NO2, SO2, O3, та тверді частинки PM2.5 та PM10

Рівень шуму

Ультрафіолетове випромінювання

Радіаційний фон

Якість води

Що таке CO_2, CO, NO_2, SO_2, O_3, PM2.5 та PM10?

CO_2 - Діокси́д вуглецю́, вуглекислий газ, тривка хімічна сполука, поширена в природних газах, що містять його в кількості від декількох відсотків до практично чистого вуглекислого газу.

CO - Монооксид вуглецю, також відомий як монооксид карбону, чадний газ безбарвний, дуже отруйний газ без запаху. Утворюється внаслідок неповного згоряння пального в автомобільних двигунах та опалюваних приладах, які працюють на вугіллі або на інших видах природного палива.

NO_2 - діокси́д азо́ту неорганічна сполука складу NO2. За звичайних умов є газом червоно-бурого кольору, з характерним гострим запахом або жовтуватою рідиною.

SO_2 - Діокси́д сі́рки, сульфу́роксид, сірчистий ангідрид, сірчистий газ неорганічна бінарна сполука складу SO2. За звичайних умов це безбарвний газ з різким задушливим запахом.

O_3 – Озо́н алотропна модифікація кисню (O_3). Газ блакитного кольору, має характерний запах "свіжості".

PM2.5 та PM10 - це мікроскопічні тверді частинки. Фактично - це все в повітрі, що не є газом, і складається з величезного різноманіття хімічних сполук та матеріалів, деякі з яких можуть бути токсичними. Через невеликий розмір багатьох частинок, що утворюють PM, деякі з цих токсинів через дихання можуть потрапляти в кров і транспортуватися по всьому тілу, осідаючи в серці, мозку та інших органах.

Особливістю предметної області можна назвати її залежність від георграфічного положення.

Моніторинг здійснюється як в реальному часі так і шляхом збору даних за періодом. Джерелами даних являються метеостанції, сенсори та стаціонарні або мобільні вимірювальні станції.

Тому сміливо можна сказати що предметна область моніторингу екологічних показників є складною та технологічно насиченою. Для успішного впровадження система потребує поєднання знань як з галузей екології, так і з інформаційних систем, технологій сенсорів та баз даних.

1.2 Аналіз вимог до програмної системи

Почнемо з пояснення того чим являється функціональні та нефункціональні вимоги

У життєвому циклі розробки програмного забезпечення, вимога утворюють основу, на якій будується вся система. Вони визначають очікування та специфікації, якими керуються розробники у створенні продукту, що відповідає потребам користувачів і бізнес-цілям. Ці вимоги зазвичай поділяються на дві категорії: функціональний та нефункціональний.

Функціональні вимоги описати конкретні дії, поведінку та функції, які має виконувати система. Вони визначають, що буде робити система, від взаємодії користувача до відповідей системи. Нефункціональні вимогиоднак зосередьтеся на тому, наскільки добре працює система. Вони стосуються таких аспектів, як продуктивність, безпека, масштабованість і надійність,

гарантуючи, що система не тільки функціонує належним чином, але й забезпечує користувацький досвід високої якості.

Розуміння обох типів вимог має важливе значення для успішного виконання проекту. Чіткі, чітко визначені функціональні вимоги гарантують, що система відповідає своїм основним цілям, тоді як нефункціональні вимоги забезпечують її ефективну та надійну роботу в реальних умовах. Збалансування обох гарантує систему, яка є не тільки функціональною, але також надійною та масштабованою, що забезпечує більше задоволення як для користувачів, так і для зацікавлених сторін.

Функціональні вимоги це детальні специфікації, які визначають дії, поведінку та функціональні можливості, які повинна виконувати система, щоб відповідати призначеній меті. У контексті програмного забезпечення та систем вони описують, що має робити система, включаючи завдання, які система повинна виконувати, як вона взаємодіятиме з користувачами та як вона реагуватиме на різні вхідні дані чи події. Ці вимоги є основоположними для того, щоб програмне забезпечення відповідало очікуванням зацікавлених сторін і відповідало запланованим бізнес-потребам.

Функціональні вимоги відіграють вирішальну роль у визначенні поведінки системи. Вони керують процесом розробки, чітко описуючи, що має робити система, гарантуючи, що розробники побудують систему, яка відповідає основному бізнесу та цілям користувачів. Ці вимоги впливають на те, якою буде система використовуваний кінцевим користувачем, впливаючи на загальне взаємодія з користувачем і досвід. Якщо функціональні вимоги чітко визначені та зрозумілі, вони допомагають зменшити неоднозначність, мінімізувати помилки під час розробки та забезпечити відповідність кінцевого продукту потребам і очікуванням користувачів. Отже, вони необхідні для створення успішної, функціональної системи, яка ефективно служить своїй меті.

Нефункціональні вимоги посилаються на атрибути якості системи, які визначають, як вона працює, а не те, що вона робить. На відміну від

функціональних вимог, які визначають дії та завдання, які повинна виконувати система, нефункціональні вимоги зосереджуються на загальних характеристиках системи та її поведінці за різних умов. Вони стосуються таких аспектів, як продуктивність, зручність використання, надійність і масштабованість, гарантуючи, що система відповідає стандартам якості та забезпечує задовільну взаємодію з користувачем.

Нефункціональні вимоги є істотними для забезпечення загальна продуктивність, зручність використання та стійкість системи. Хоча функціональні вимоги гарантують, що система може виконувати свої завдання, нефункціональні вимоги визначають, наскільки добре ці завдання виконуються, впливаючи на задоволеність користувачів та ефективність системи. Встановлюючи стандарти для атрибутів якості, нефункціональні вимоги гарантують, що система є надійною, безпечною та масштабованою, забезпечуючи позитивний досвід користувача та дозволяючи системі адаптуватися до мінливих вимог. Ігнорування цих вимог може призвести до проблем із продуктивністю, поганого залучення користувачів і вразливостей, що робить їх важливою частиною успішної розробки програмного забезпечення.

Повертаємось до того, які вимоги мають бути в системі.

Почнемо з функціональних вимог:

Функціональні вимоги до системи моніторингу екологічних показників такі – Збір даних, збереження інформації, візуалізація, звітність.

Збір даних – це автоматизований прийом показників з зовнішніх джерел, в нашому випадку це API.

Реалізація цієї вимоги така – підключення до API, збір даних. Дані які будуть збиратися це CO, NO2, SO2, O3, PM2.5 та PM10

Збереження інформації – це зберігання даних у базі даних для подальшого аналізу та звітності

Реалізація вимоги така – застосування БД, таке як MongoDB, запис повинен мати наступне

-Ідентифікатор

-Назву міста, регіону, країни

-Час

-Показники

Візуалізація – це надання інтуїтивно зрозумілого інтерфейсу для перегляду даних

Реалізація вимоги така – розробити простий інтерфейс з пошуком, вивід даних та створення звіту

Звітність – це створення звітів для потреб тих чи інших користувачів

Реалізація вимоги така – звіт має генеруватись з фільтрів які надає користувач, тобто країна та час від і час до. У самому звіті має бути назва країни, час коли були зняті показники та дані самих показників

До нефункціональних вимог відносимо наступне – продуктивність, безпеку, надійність та зручність у використанні

Опишемо пункти:

Продуктивність – це швидкість обробки запитів користувача

Безпека – це захист даних що є у базі даних

Надійність – це збереження працездатності у разі помилок, наприклад невірного введення назви

Зручність у використанні або **Юзабіліті** – це зручність та простота інтерфейсу та швидкий доступ до наданих функцій

1.3 Моделювання предметної області

Для моделювання предметної області будемо використовувати діаграми прецедентів, послідовності та активності.

Опишемо що таке взагалі UML та навіщо потрібні вище перераховані діаграми для моделювання предметної області

Unified Modeling Language — уніфікована мова моделювання. Розшифруємо: *modeling* передбачає створення моделі, що описує

об'єкт. *Unified* — підходить для широкого класу проектованих програмних систем, різних областей додатків, типів організацій, рівнів компетентності, розмірів проектів. UML описує об'єкт в єдиному заданому синтаксисі, тому де б ви не намалювали діаграму, її правила будуть зрозумілими для всіх, хто знайомий з цією графічною мовою — навіть в іншій країні.

Одне із завдань UML — служити засобом комунікації всередині команди та при спілкуванні з замовником. Давайте розгляднемо можливі варіанти використання дігарам.

- Проектування. UML-діаграми стануть у пригоді при моделюванні архітектури великих проектів, в якій можна зібрати як великі, так і дрібніші деталі і намалювати каркас програми. По ньому пізніше буде будуватись код.
- Реверс-інжиніринг — створення UML-моделі з існуючого коду додатку, зворотна побудова. Может застосовуватися, наприклад, на проектах підтримки, де є написаний код, але документація неповна або відсутня.
- З моделей можна витягувати текстову інформацію і генерувати відносто читабельні тексти — документувати. Текст і графіка будуть доповнювати один одного.

Як будь-яка інша мова, UML має власні правила оформлення моделей і синтаксис. За допомогою графічної нотації UML можна візуалізувати систему, об'єднати всі компоненти в єдину структуру, уточнювати і покращувати модель у процесі роботи. На загальному рівні графічна нотація UML містить 4 основні типи елементів:

- фігури;
- лінії;
- значки;
- написи.

UML-нотація де-факто є галузевим стандартом у сфері розробки програмного забезпечення, IT-інфраструктури і бізнес-систем.

Діаграма класів — Class diagram

Діаграма прецедентів використовує 2 основних елементи:

1) Actor (учасник) — множина логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Учасником може бути людина, роль людини в системі чи інша система, підсистема або клас, які представляють щось поза сутністю.

2) Use case (прецедент) — опис окремого аспекту поведінки системи з точки зору користувача. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

Діаграма активностей — Activity diagram

Діаграма активностей описує динамічні аспекти поведінки системи у вигляді блок-схеми, яка відображає бізнес-процеси, логіку процедур і потоки робіт — переходи від однієї діяльності до іншої. Таким чином тут зображено алгоритм дій системи або взаємодії кількох систем.

Діаграма послідовності — Sequence Diagram

Використовується для уточнення діаграм прецедентів — описує поведінкові аспекти системи. Діаграма послідовності відображає взаємодію об'єктів в динаміці, в часі. При цьому інформація набуває вигляду повідомлень, а взаємодія об'єктів передбачає обмін цими повідомленнями в рамках сценарію.

Нижче наведно діаграми послідовності, активності та прецедентів нашої предметної області моніторингу екологічних показників

Діаграма прецедентів

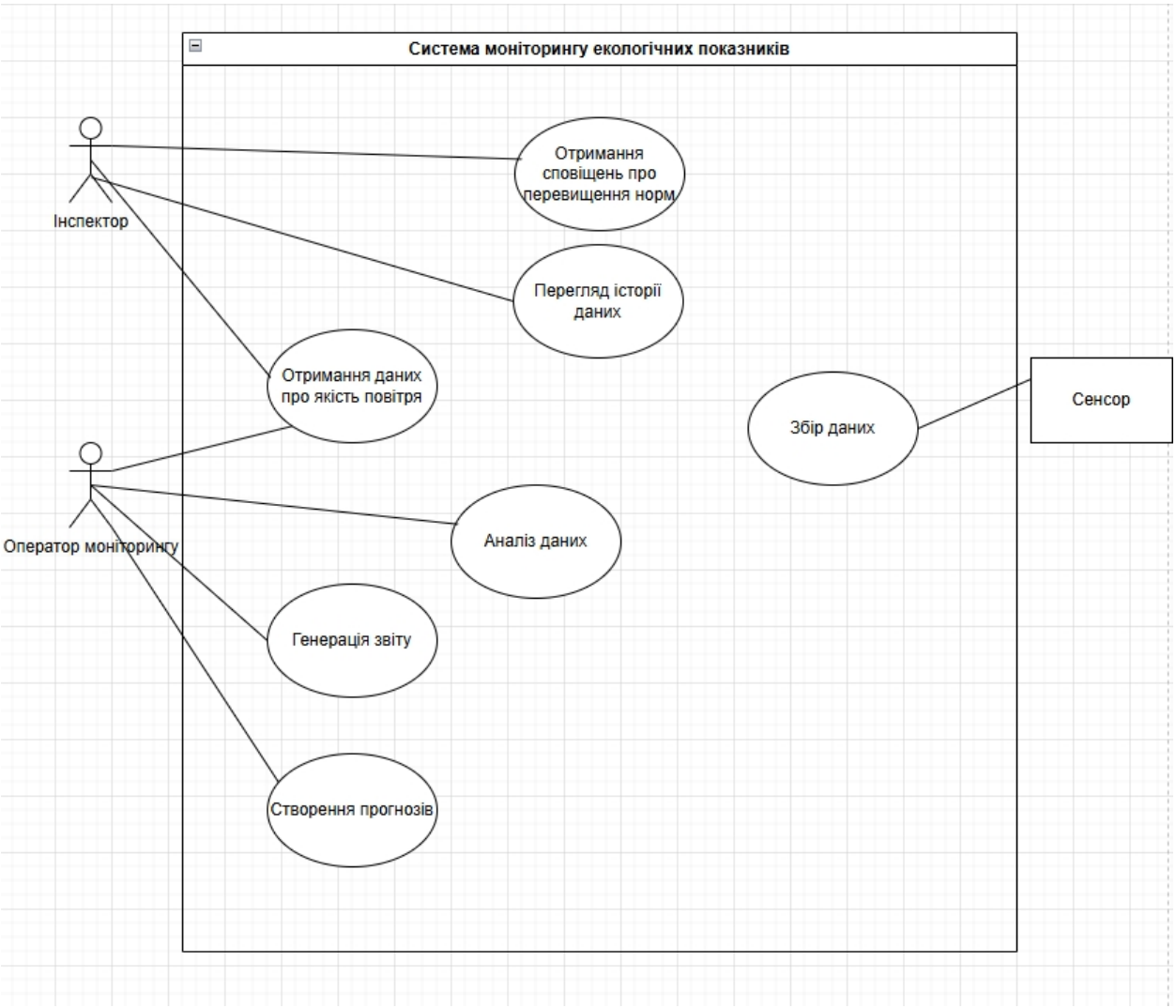


Рис.1.1 Діаграма прецедентів

На цій діаграмі прецедентів системи моніторингу екологічних показників бачимо трьох акторів, а саме:

-Інспектор

-Оператор моніторингу

-Сенсор

Також бачимо самі прецеденти

Інспектор має асоціацію з прецедентами – «Отримання сповіщень про перевищення норм», «Перегляд історії даних», «Отримання даних про якість повітря»

Оператор моніторингу має асоціації – «Отримання даних про якість повітря», «Аналіз даних», «Генерація звіту», «Створення прогнозів»

Сенсор має всього одну асоціацію, а саме – «Збір даних»

Діаграма послідовності

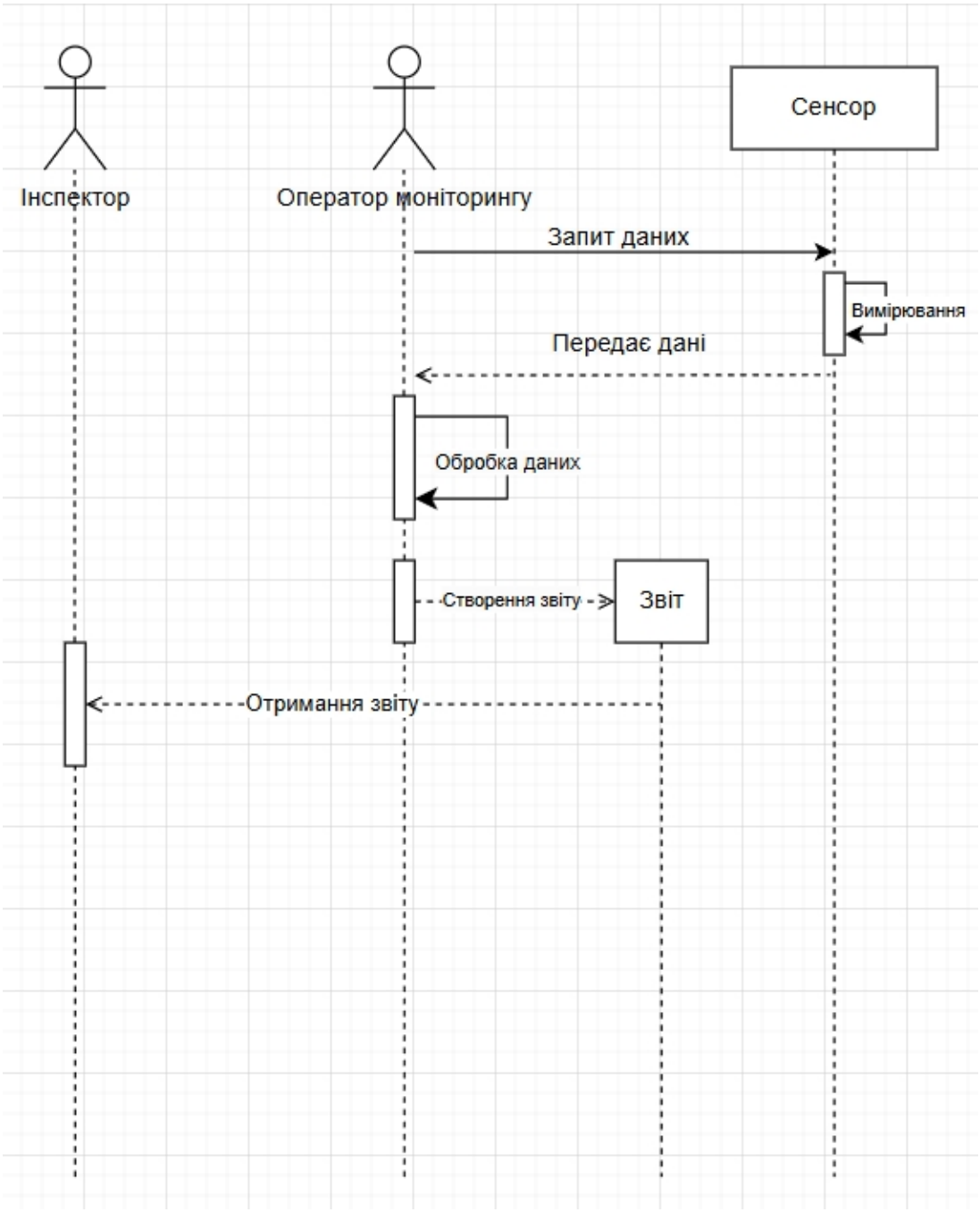


Рис.1.2 Діаграма послідовності

На цій діаграмі послідовності системи моніторингу екологічних показників бачимо послідовність дій створення звіту.

Оператор моніторингу надає запит отримання даних до сенсору, сенсор виконує вимірювання та надсилає дані до оператора, оператор оброблює дані та створює звіт, після чого звіт надсилається до інспектора.

Діаграма активності

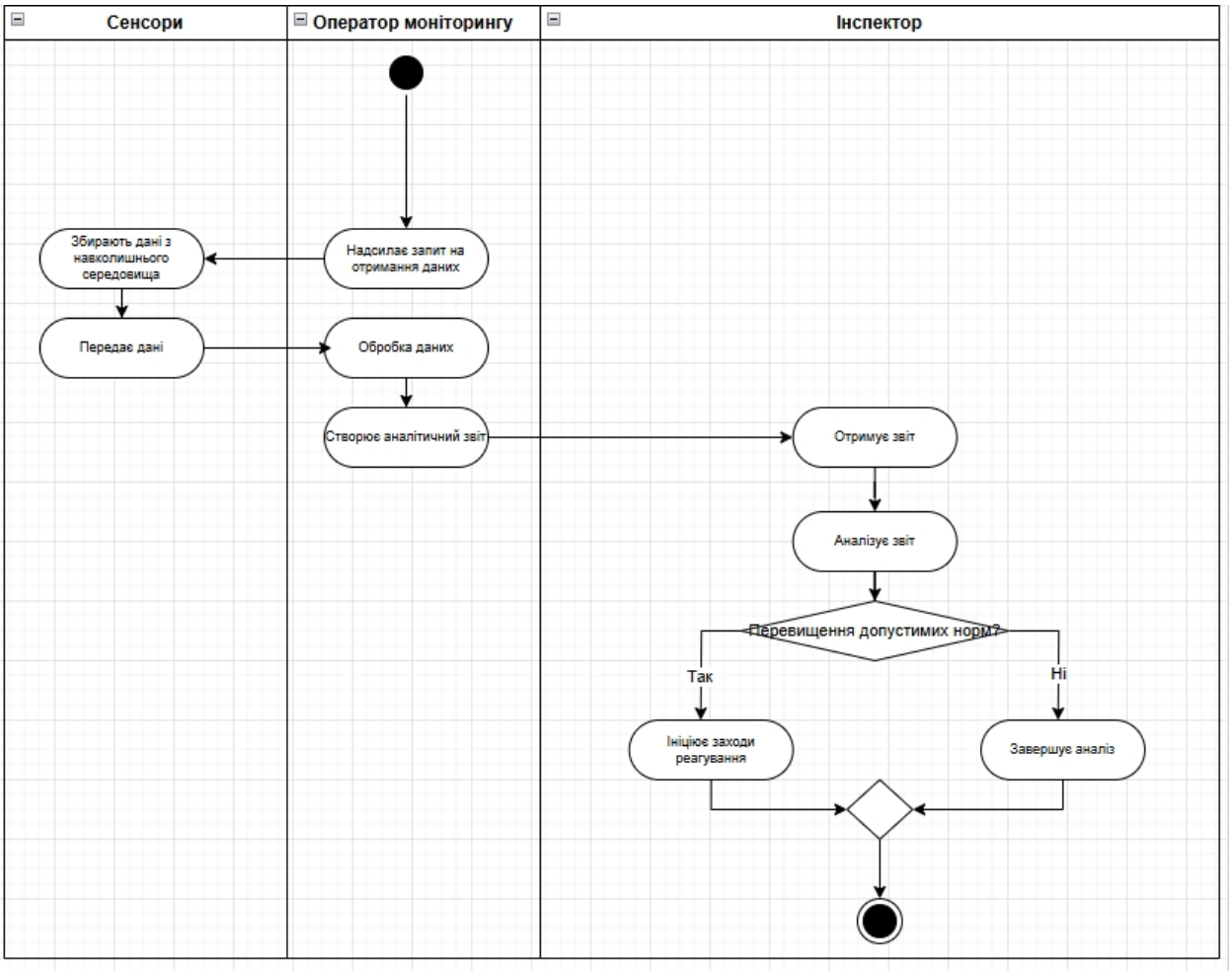


Рис.1.3 Діаграма активності

На цій діаграмі активності бачимо роботу бізнес логіки. Оператор надсилає запит на отримання даних, сенсор збирає дані з середовища і передає їх до оператора, оператор оброблює ці дані та створює звіт, надсилає звіт до інспектора, інспектор отримує готовий звіт та починає його аналізувати, якщо є перевищення допустимих норм то ініціюються заходи реагування щодо перевищення допустимих норм, якщо перевищення допустимих норм немає, то інспектор завершує аналіз і на цьому закінчується діаграма активності.

1.4 Огляд інформаційних джерел та існуючих рішень

З огляду інформаційних джерел [5] розгянемо систему стандартів ISO 14000, та накази і рекомендації Міністерства захисту довкілля України

ISO 14000 - ISO 14000 має схожість із ISO 9000 (сімейством стандартів з менеджменту якості), обидва відносяться до процесу виробництва продукту, а не до самого продукту. Як і ISO 9000, сертифікація здійснюється сторонніми організаціями, а не ISO безпосередньо. Вимоги ISO 14000 є невід'ємною частиною Схеми еко-менеджменту та аудиту ЄС (EMAS). Проте, вимоги EMAS є більш жорсткими щодо підвищення екологічної ефективності діяльності організації, дотримання екологічного законодавства, звітності та залучення співробітників. Сімейство стандартів ISO 14000 насамперед включає стандарт ISO 14001, що являє собою фундаментальний набір правил, що використовуються організаціями по всьому світу, що проектують і впроваджують ефективні системи екологічного менеджменту (далі СЕМ). Іншим стандартом, включеним до цієї серії, є ISO 14004, що дає набір додаткових посібників для досягнення результативності СЕМ. Він також містить спеціальні правила, пов'язані зі специфічними аспектами екологічного менеджменту. Основною метою серії стандартів ISO 14000 та встановлених ними вимогами є просування найбільш ефективних та результативних практик екологічного менеджменту в організаціях, а також надання: корисних, придатних до використання, економічно-вигідних, систематизованих, гнучких та пристосовуваних під діяльність різноманітних організацій інструментів. Стандарти серії ISO 14000 також демонструють найбільш успішні практики, що використовуються для збору, подання та аналізу інформації, що стосується екології.На відміну від попередніх екологічних розпоряджень та правил, що ґрунтуються на примусових підходах, пізніше замінених підходами, заснованими на ринкових механізмах, ISO 14000 був заснований на добровільному підході до екологічного регулювання. Серія стандартів ISO 14000 включає ISO 14001, що містить посібники для створення або

покращення СЕМ. Стандарт ISO 14001 успадкував багато положень від стандарту управління якістю - ISO 9000, який служив як модель для внутрішньої структури ISO 14001 (NationalAcademyPress 1999), обидва з яких можуть впроваджуватися паралельно нарівні один з одним. Як і ISO 9000, ISO 14000 може бути як внутрішнім інструментом менеджменту, і способом продемонструвати екологічну відповідальність своїм клієнтам і потребителям.До розробки серії стандартів ISO 14000, організації добровільно створили свої власні СЕМ, але в них з'явилося багато відмінностей через силу впливу різних компаній на екологію, тому і була розроблена універсальна серія стандартів ISO 14000. СЕМ визначена ISO як «частина системи менеджменту організації, що включає організаційну ресурси розробки, впровадження, виконання та управління політикою у сфері екології».ISO 14001 визначає критерії для СЕМ. Він не встановлює вимоги для екологічної ефективності, але визначає основні правила, яким організація може слідувати для побудови ефективної СЕМ. Він може бути використаний організаціями для підвищення ефективності використання ресурсів, зниження втрат та витрат. Використовуючи ISO 14001, можна продемонструвати захищеність менеджменту організації та її працівників. Так само він може бути використаний для демонстрації зацікавленим сторонам того, що компанія вимірює та покращує екологічний вплив на них. ISO 14001 може бути інтегрований з іншими функціями управління для більш зручного досягнення своїх екологічних та економічних цілей.Як і інші стандарти, ISO 14001 є добровільним. Його головною метою є допомога компаніям у покращенні своїх екологічних показників, дотримуючись при цьому чинного законодавства. Організації відповідальні за встановлення своїх цілей, їх відстеження та досягнення. Стандарт служить допомогою у досягненні цілей і завдань підприємства, і навіть їх моніторингу і измерения.Стандарт може бути застосований до різних рівнів діяльності підприємства, від організаційного, рівня виробництва та надання услуг. Замість того, щоб фокусуватися на конкретних вимірах та цілях екологічної ефективності, стандарт виділяє те, що

організація має робити, щоб досягти цієї мети. Успішність системи залежить від залученості всіх рівнів організації, особливо вищого керівництва, що має бути залучено у розробку, впровадження та управління СЕМ.ISO 14001 має родові ознаки стандартів на системи менеджменту, корисних для організацій, які бажають покращити свою діяльність та управляти ресурсами більш ефективно.

Він підходить для:

• Усіх великих транснаціональних компаній;

• Компаній, як із високими, так і малими ризиками;

• Виробничих компаній та компаній, що надають послуги, включаючи місцеві державні спільноти;

• Усіх галузевих секторів, включаючи публічні та закриті;

• Виробників унікального обладнання та їх постачальників.

Усі стандарти періодично переглядаються ISO актуальність поточним ринковим требованиям.

Список стандартів серії ISO 14000

• ISO 14001 Системи екологічного менеджменту. Вимоги та посібник із застосування

• ISO 14004 Системи екологічного менеджменту. Керівні вказівки за принципами, системами та методами забезпечення функціонування

• ISO 14015 Екологічний менеджмент. Екологічна оцінка майданчиків та організацій

• ISO 14020 Екологічні етикетки та декларації. Основні засади

• ISO 14031 Керування довкіллям. Оцінювання екологічної ефективност

і• ISO 14040 Керування довкіллям. Оцінка життєвого циклу. Принципи та структура.

• ISO 14050 Керування довкіллям. Словник.

• ISO 14062 Екологічний менеджмент. Інтегрування екологічних аспектів у проектування та розробку продукції

• ISO 14063 Екологічний менеджмент. Обмін екологічною інформацією. Рекомендації та приклади

• ISO 14064 Вимірювання, кількісне вимірювання та зменшення викидів парникових газів.

Також можу запропонувати ознайомлення з наказами

Методичні рекомендації щодо врахування кліматичного компонента в документах державного планування та під час здійснення стратегічної екологічної оцінки та оцінки впливу на довкілля Затверджено наказом Міндовкілля від 31.10.2024 № 1382

В документі визначаються підходи до інтеграції кліматичних аспектів у процеси екологічного планування та оцінки впливу на довкілля

Методичні рекомендації щодо здійснення післяпроектного моніторингу

Затверджено наказом Міндовкілля від 15.03.2024 № 291.

Описує порядок проведення моніторингу після реалізації для оцінки впливу на довкілля

Також проаналізувавши існуючи рішення, вирішив провести їх аналіз

Перша система AirVisual – це частина екосистеми компанії IQAir, яке спеціалізується на моніторингу повітря. Сервіс має власну мережу станцій, офіційні дані та показники від користувачів

Функції які надає ця система – моніторинг у реальному часі, глобальне картографування, мобільні застосунки, прогноз якості повітря

З переваг цієї системи можна визначити наступне – якісна аналітика та прогнозування, інтуїтивно зрозумілий інтерфейс та велике охоплення

З недоліків можна визначити наступне – це закрита платформа з обмеженим API, дані не є адаптованими під українські стандарти.

Друга система SaveEcoBot – є українським екологічним чат-ботом та веб, яка інтегрує дані з державних реєстрів.

Функції які надає ця система – публікація даних у реальному часі, відображення забруднення на мапі, надсилає повідомлення про перевищення норм

З переваг системи можна визначити наступне – відкрите програмне забезпечення, локалізація на українське законодавство.

З недоліків системи можна визначити наступне – обмежена кількість сенсорів, неможливість формування звітів та відсутність аналітики

Третя система OpenSensMap – міжнародна відкрита платформа, яка дозволяє публікувати дані з особистих метеостанцій.

Функції які надає ця система – геологічне привязування на інтерактивній мапі, відображення довільних екологічних параметрів, підтримка різних протоколів

З переваг системи можна визначити наступне – відкрите програмне забезпечення, можливість самостійного підключення станції

З недоліків можна визначити наступне – немає підтримки національних норматив, потрібно мати технічні навички, дані можуть бути без перевірки достовірності

1.5 Постановка завдання

Призначення до програмного забезпечення системи моніторингу екологічних показників є забезпечення швидкого та зручного доступу до показників забруднення повітря. Система є корисною як для екологічних служб так і для звичайних містян.

Основне призначення системи – забезпечення отримання показників за пошуком того чи іншого міста країни та подальшого звітування цих показників.

Розроблюване програмне забезпечення є веб застосунком. Дані надходять з відкритої API, а саме WeatherAPI, застосунок розміщено на

сервері. Дані надсилаються та зберігаються у базі даних. Також розроблено звітування за фільтром назви міста та часом від та часом до.

Інтерфейс програми розроблений з розумінням того що користувачу потрібен зручний та простий інтерфейс, де все буде інтуїтивно зрозуміло. Завдяки цьому спрощується навігація на сайті для користувачів з різними рівнями володіння компютера. Саме оформлення є мінімалістичним з акцентом на інформативність.

У застосунку немає розділу ролей і з цього маємо повну відсутність підтримки різних ролей користувачів. Також з цих причин відсутня реєстрація у застосунку.

Звіт буде зберігатися у файлі Excel, в якому буде зберігатись та ж інформація що буде бачити користувач у застосунку.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

2.1 Логічна модель даних у вигляді ER-діаграми

Почнемо з того що взагалі з себе представляє ER-діаграми.

Діаграма зв’язку сутностей, також відома як ERD, це діаграма, яка відображає зв’язок наборів сутностей, що зберігаються в базі даних. Іншими словами, ER-діаграми допомагають пояснити логічну структуру баз даних. ER-діаграми створюються на основі трьох основних понять: сутності, атрибути та зв’язки.

Діаграми ER містять різні символи, які використовують прямокутники для представлення сутностей, овали для визначення атрибутів і ромби для представлення зв’язків.

На перший погляд діаграма ER дуже схожа на блок-схему. Однак діаграма ER містить багато спеціалізованих символів, і їх значення роблять цю модель унікальною. Метою діаграми ER є представлення інфраструктури сутності.

Модель ER розшифровується як Entity Relationship Model — високорівнева схема концептуальної моделі даних. Модель ER допомагає систематично аналізувати вимоги до даних для створення добре спроектованої бази даних. Модель ER представляє сутності реального світу та зв’язки між ними. Створення моделі швидкої допомоги в СУБД вважається найкращою практикою перед впровадженням бази даних.

ER моделювання допомагає вам систематично аналізувати вимоги до даних для створення добре спроектованої бази даних. Отже, вважається найкращою практикою завершити моделювання ER перед впровадженням бази даних.

Ось основні причини використання діаграми ER

- Допомагає визначити терміни, пов’язані з моделюванням зв’язків сутностей
- Надайте попередній перегляд того, як усі ваші таблиці мають з’єднуватися, які поля будуть у кожній таблиці
- Допомагає описувати сутності, атрибути, відносини
- ER-діаграми можна перевести в реляційні таблиці, що дозволяє швидко створювати бази даних
- Діаграми ER можуть використовуватися розробниками баз даних як план для реалізації даних у конкретних програмних програмах
- Розробник бази даних отримує краще розуміння інформації, яка міститься в базі даних за допомогою діаграми ERP
- Діаграма ERD дозволяє спілкуватися з логічною структурою бази даних користувачам

Але для системи моніторингу екологічних показників використовувати реляційну базу даних не потрібно, так як для системи моніторингу екологічних показників більш правильним буде використовувати документно-орієнтовану базу даних, наприклад MongoDB.

З цих причин робити логічну модель даних у вигляді ER-діаграми не є можливим.

Поясню чому саме – як вже було сказано ER-діаграма призначена для реляційної моделі бази даних тому що ER-модель описує зв’язки та атрибути між таблицями і має чітко структуру, такі як 1NF, 2NF, 3NF. Проте документно-орієнтована база даних зберігає самі дані у вигляді JSON-файлів, які можуть мати різну структуру. Також, у реляційній БД звязки виражаються як foreign keys, які показуються в ER-діаграмі. У документно-орієнтованій БД звязки можуть бути як вбудованими так і посиланнями, але жоден з цих звязків не є логічним як має бути у ER-діаграмі.

Вибір документно-орієнтованої БД для роботи системи моніторингу екологічних показників зумовлений наступними міркуваннями.

База даних документів - це тип баз даних NoSQL, які призначені для зберігання і запиту даних у вигляді документів у форматі, подібному до JSON. JavaScript Object Notation (JSON) - це відкритий формат обміну даними, який читається як людиною, так і машиною. Розробники можуть використовувати документи JSON у своєму коді та зберігати їх безпосередньо в базі даних документів. Гнучкий, напівструктурований, ієрархічний характер документів та їхніх баз даних дає змогу їм розвиватися відповідно до потреб додатків.

У чому переваги баз даних документів? Бази даних документів забезпечують гнучкість індексації, продуктивність виконання стандартних запитів та аналітику наборів документів.

Простота розробки

Документи JSON відповідають об'єктам - поширеному типу даних у більшості мов програмування. Під час розробки застосунків розробники можуть гнучко створювати й оновлювати документи безпосередньо з коду. Це означає, що вони витрачають менше часу на попереднє створення моделей даних. Таким чином, розробка додатків відбувається швидше та ефективніше.

Гнучка схема

База даних, орієнтована на документи, дає змогу створювати кілька документів з різними полями в одній колекції. Це може бути зручно при зберіганні неструктурованих даних, таких як електронні листи або публікації в соціальних мережах. Однак у деяких базах даних документів передбачено перевірку схеми, тому можна ввести деякі обмеження для структури.

Продуктивність за будь-якого масштабу

Бази даних документів пропонують вбудовані можливості поширення. Їх можна горизонтально масштабувати на кілька серверів без зниження продуктивності, що також економічно вигідно. Крім того, бази даних документів забезпечують відмовостійкість і доступність завдяки вбудованій реплікації.

Які варіанти використання баз даних документів?

Модель документа добре підходить для каталогів, управління контентом і датчиками, і багато чого іншого. Кожен документ для кожного випадку використання унікальний і розвивається з плином часу.

Управління контентом

База даних документів - чудовий вибір для додатків управління контентом, таких як платформи для блогів і розміщення відео. При використанні бази даних документів кожна сутність, що відстежується додатком, може зберігатися як окремий документ. База даних документів дає змогу розробнику зі зручністю оновлювати додаток у разі зміни вимог. Крім того, якщо необхідно змінити модель даних, то потрібне оновлення тільки зачеплених цією зміною документів. Для внесення змін немає необхідності оновлювати схему і переривати роботу бази даних.

Каталоги

Документні бази даних ефективні для зберігання каталожної інформації. Наприклад, у додатках для інтернет-комерції різні товари зазвичай мають різну кількість атрибутів. Управління тисячами атрибутів у реляційних базах даних неефективне. Крім того, кількість атрибутів впливає на продуктивність читання. При використанні бази даних документів атрибути кожного товару можна описати в одному документі, що спрощує управління і підвищує швидкість читання. Зміна атрибутів одного товару не вплине на інші.

Керування датчиками

Інтернет речей (IoT) сприяє тому, що організації стали регулярно збирати дані з інтелектуальних пристроїв, таких як датчики і лічильники. Дані датчиків зазвичай надходять у вигляді безперервного потоку змінних значень. Через проблеми із затримкою деякі об'єкти даних можуть бути неповними або дубльованими чи зовсім відсутніми. Крім того, необхідно зібрати великий обсяг даних, перш ніж фільтрувати або підсумовувати їх для аналітики.У цьому випадку зручніше використовувати сховища документів. Ви можете оперативно зберігати дані датчиків у тому вигляді, в якому вони є, не

займаючись їхнім очищенням або приведенням у відповідність до заздалегідь заданих схем. Ви також можете масштабувати їх у міру необхідності і видаляти документи цілком після завершення аналізу.

Як працюють бази даних документів

У базах даних документів дані зберігаються у вигляді пар «ключ-значення» у форматі JSON. Читання і запис документів у форматі JSON у бази даних можна здійснювати програмно.Структура документів JSONJSON представляє дані трьома способами:

Ключове значення «ключ-значення» записано у фігурних дужках. Ключ - це рядок, а значення може бути будь-яким типом даних, наприклад цілим, десятковим або логічним. Наприклад, просте ключове значення: {"year": 2013}.

Масив - це впорядкований набір значень, визначених у лівих ([) і правих (]) дужках. Елементи масиву розділені комами. Наприклад, {"fruit": ["apple", "mango"]}.

Об'єкти - це набір пар «ключ-значення». По суті, документи JSON дають розробникам можливість вбудовувати об'єкти і створювати вкладені пари. Наприклад, {"address": {"country": "USA", "state": "Texas"}}

Операції з базою даних документів

Можна створювати, читати, оновлювати та видаляти цілі документи, що зберігаються в базі даних. Бази даних документів надають мову запитів або API, що дають змогу розробникам виконувати такі операції:

Створення

У базі даних можна створювати документи. Кожен документ має унікальний ідентифікатор, який слугує ключем.

Читання

Для читання даних документа можна використовувати API або мову запитів. Можна виконувати запити, використовуючи значення полів або

ключі, а також додавати індекси в базу даних для підвищення продуктивності читання.

Оновлення

Можна гнучко оновлювати наявні документи. Можна переписати весь документ або оновити окремі значення.

У системі моніторингу екологічних показників є різні типи даних, наприклад вологість, температура, тверді речовини або гази. Також з датчиків приходять метадані по часу, локації та даті. Не забуваємо що той же самий MongoDB, гарно оптимізований для великих вставок та читанню документів. Якщо ж робилася реляційна база даних, то потрібно було створювати велику кількість таблиць з ключами для сутностей.

Нижче показано вид сутності так званої колекції.

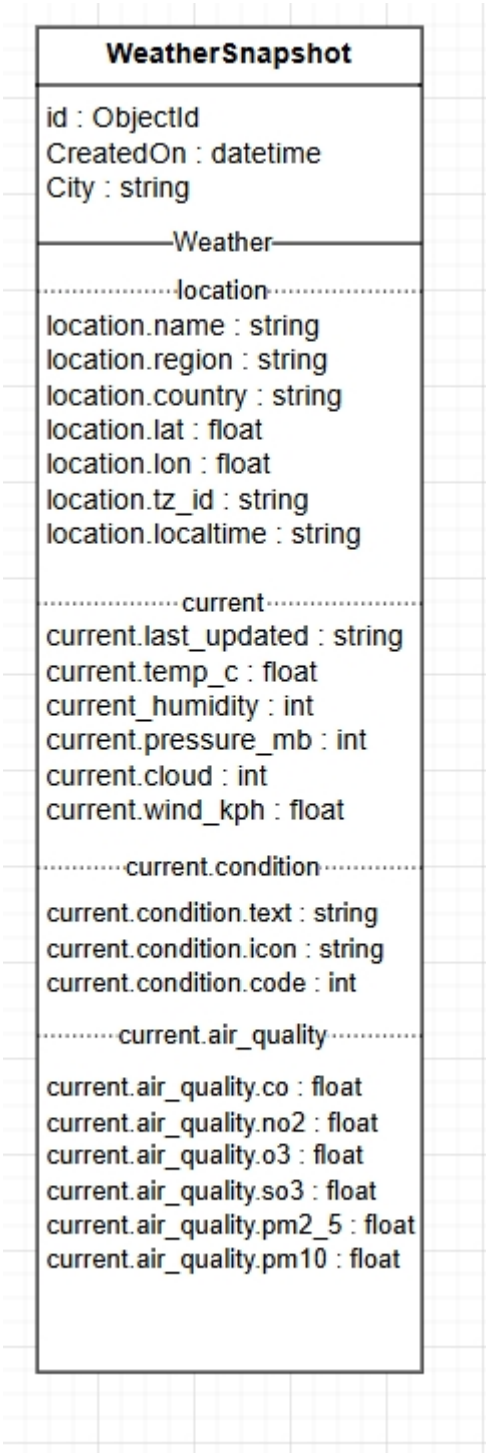


Рис 2.1 Сутність колекції WeatherSnapshot

2.2 Діаграма класів та кооперацій

Знову почнемо с того що таке діаграма класів та діаграма кооперацій та з чого вони складаються.

Діаграма класів

На діаграмах класів буде показано різноманітні класи, які утворюють систему і їх взаємозв'язки. Діаграми класів називають «статичними діаграмами», оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв'язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи «є частиною» інших класів, — але не показано методи, які при цьому викликаються.

Клас

Клас визначає атрибути і методи набору об'єктів. Всі об'єкти цього класу (екземпляри цього класу) мають спільну поведінку і однаковий набір атрибутів (кожен з об'єктів має свій власний набір значень). Іноді замість назви «клас» використовують назву «тип», але, слід зауважити, що ці назви описують різні речі: тип є загальнішим визначенням.

У UML класи позначаються прямокутниками з назвою класу, у цих прямокутниках у вигляді двох «відсіків» може бути показано атрибути і операції класу.

Атрибути

У UML атрибути показуються щонайменше назвою, також може бути показано їх тип, початкове значення і інші властивості. Крім того, атрибути може бути показано з областю видимості атрибута:

- + відповідає *публічним (public)* атрибутам
- # відповідає *захищеним (protected)* атрибутам
- - відповідає *приватним (private)* атрибутам

Операції

Операції (методи) також показуються принаймні назвою, крім того, може бути показано їх параметри і типи значень, які буде повернуто. Операції, як і атрибути, може бути показано з областю видимості:

- + відповідає *публічним (public)* операціям
- # відповідає *захищеним (protected)* операціям
- - відповідає *приватним (private)* операціям

Діаграма кооперацій

Поняття кооперації є одним з фундаментальних понять у мові UML. Воно служить для позначення безлічі взаємодіючих з певною метою об'єктів в загальному контексті модельованої системи.

Мета самої кооперації полягає в тому, щоб специфікувати особливості реалізації окремих найбільш значущих операцій в системі. Кооперація визначає структуру поведінки системи в термінах взаємодії учасників цієї кооперації.

Діаграма кооперації насамперед відображає структуру взаємодії та містить такі елементи:

- Екземпляри акторів і класів, що беруть участь в реалізації варіанту використання;
- Асоціацію між екземплярами акторів і класів;
- Повідомлення, що передаються між екземплярами акторів і класів.

Кооперація може бути представлена на двох рівнях:

- рівні специфікації - показує ролі класифікаторів та ролі асоціацій у розглянутому взаємодії;
- рівні прикладів - вказує екземпляри і зв'язки, що утворюють окремі ролі в кооперації.

Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але й усі структурні відносини між об'єктами, які беруть участь у цій взаємодії.

На відміну від діаграми послідовності, на діаграмі кооперації зображаються тільки відносини між об'єктами, що грають певні ролі у взаємодії. З іншого боку, на цій діаграмі не вказується час у вигляді окремого виміру. Тому послідовність взаємодій і паралельних потоків може бути визначена за допомогою порядкових номерів.

Отже, якщо необхідно явно специфікувати взаємозв'язок між об'єктами в реальному часі, краще це робити на діаграмі послідовності.

За допомогою діаграми кооперації можна описати повний контекст взаємодій як своєрідний часовий "зріз" сукупності об'єктів, взаємодіючих між собою для виконання певного завдання або бізнес-цілі програмної системи.

Нижче наведено діаграму кооперації для нашої системи.

Клієнт прийшовши в магазин обирає запчастину. Потім він замовляє її у парцівника, той в свою чергу оформляє замовлення. Після того як працівник отримав запчастину зі складу, клієнт зобов’язаний підписати замовлення та розрахуватися на касі. Далі працівник видає запчастину клієнту.

Рис.2.2 Діаграма кооперацій «Створення звіту»

На цій діаграмі бачимо принцип взаємодій для створення звіту. Сенсор надає дані для оператору моніторингу, після чого оператор моніторингу створює звіт

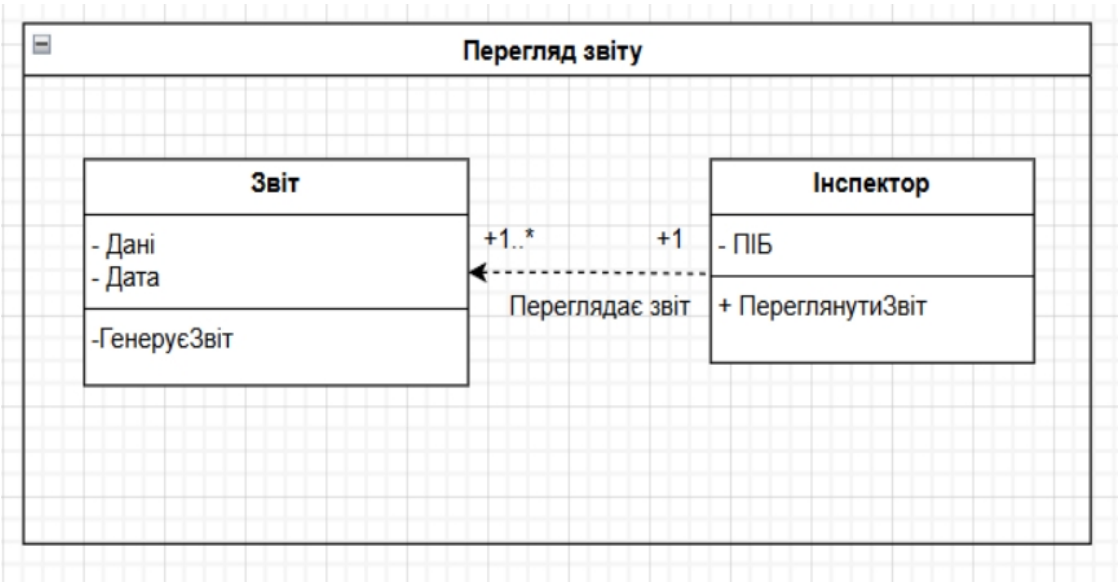


Рис.2.3 Діаграма кооперацій «Перегляд звіту»

На цій діаграмі бачимо принцип перегляду звіту. Інспектор переглядає звіт.

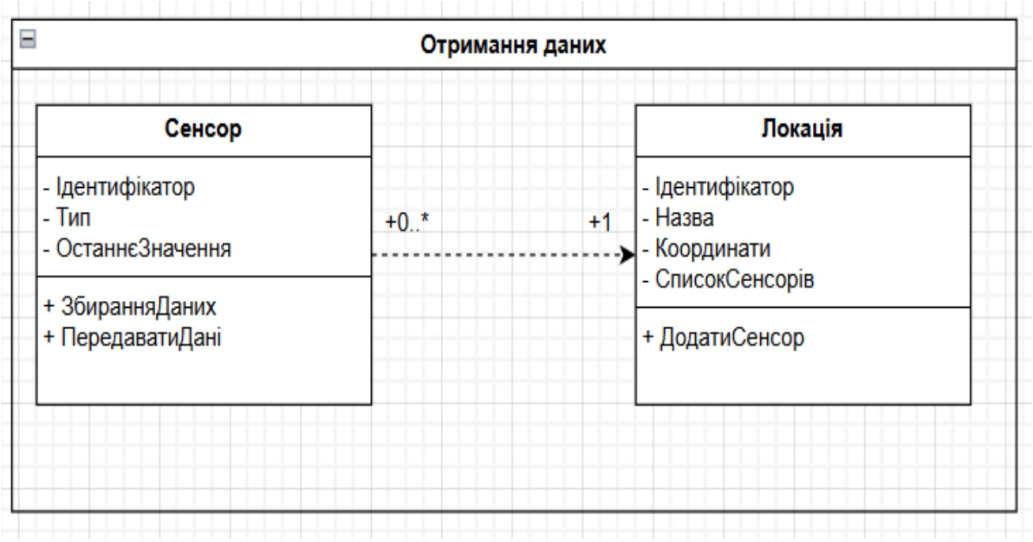


Рис.2.4 Діаграма кооперацій «Отримання даних»

На цій діаграмі бачимо принцип отримання даних. Сенсор отримує дані з локації

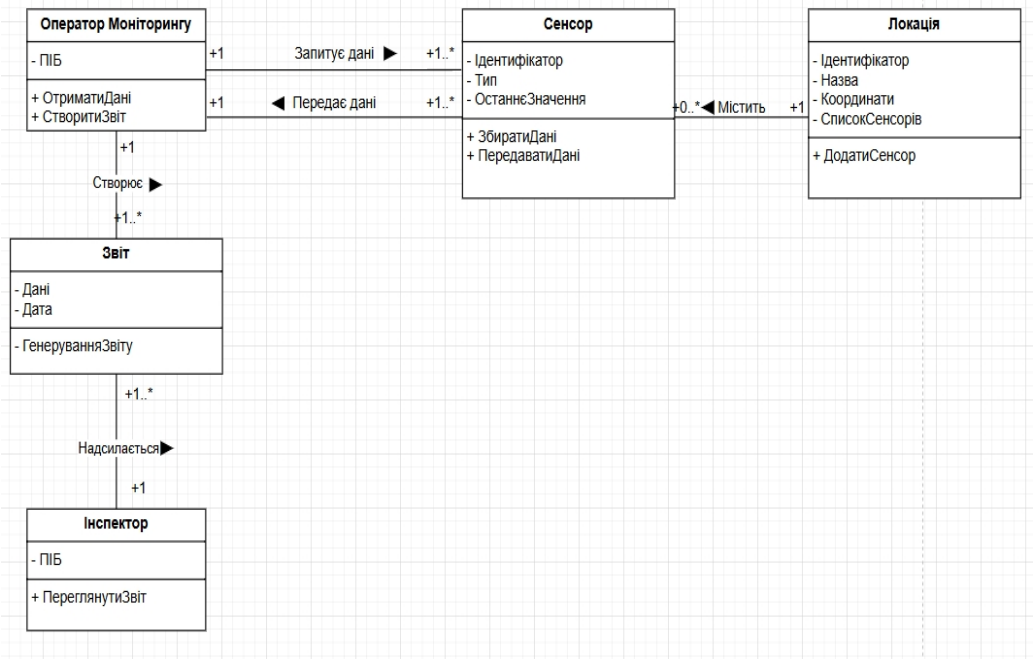


Рис.2.5 Діаграма класів

На цій діаграмі класів бачимо взаємодії між класами. Оператор моніторингу запитує дані у сенсору, який містить дані локації, сенсор передає

дані до оператора моніторингу, після чого оператор моніторингу створює звіт, який потім надсилається до інспектора.

2.3 Діаграма пакетів

Пакети в уніфікованій мові моделювання використовуються для групування елементів і надання просторів імен для згрупованих елементів. Пакет може містити інші пакети, що забезпечує ієрархічну організацію пакетів.

Майже всі елементи UML можна згрупувати в пакети. Таким чином, класи, об'єкти, варіанти використання, компоненти, вузли, екземпляри вузлів тощо можуть бути організовані в пакети, що робить керованою організацію безлічі елементів, що містяться в реальній моделі UML.

Великі системи створюють особливі проблеми. Малювання моделі класу для великої системи занадто велике для розуміння. Занадто багато зв'язків між класами, щоб зрозуміти. Корисним методом розв'язання цієї проблеми є пакет UML. Пакети в довідці Unified Modeling Language.

1. Щоб згрупувати елементи
2. Щоб надати простір імен для згрупованих елементів
3. Пакет може містити інші пакети, що забезпечує ієрархічну організацію пакетів.
4. Елементи UML можна групувати в пакети.

Таким чином, пакетна діаграма, структурна діаграма, показує розташування та організацію елементів моделі в середньому та великому проєкті. Діаграми пакетів можуть відображати як структуру, так і залежності між підсистемами або модулями, показуючи різні уявлення про систему, наприклад, як про багаторівневий застосунок - моделі багаторівневого застосунку.

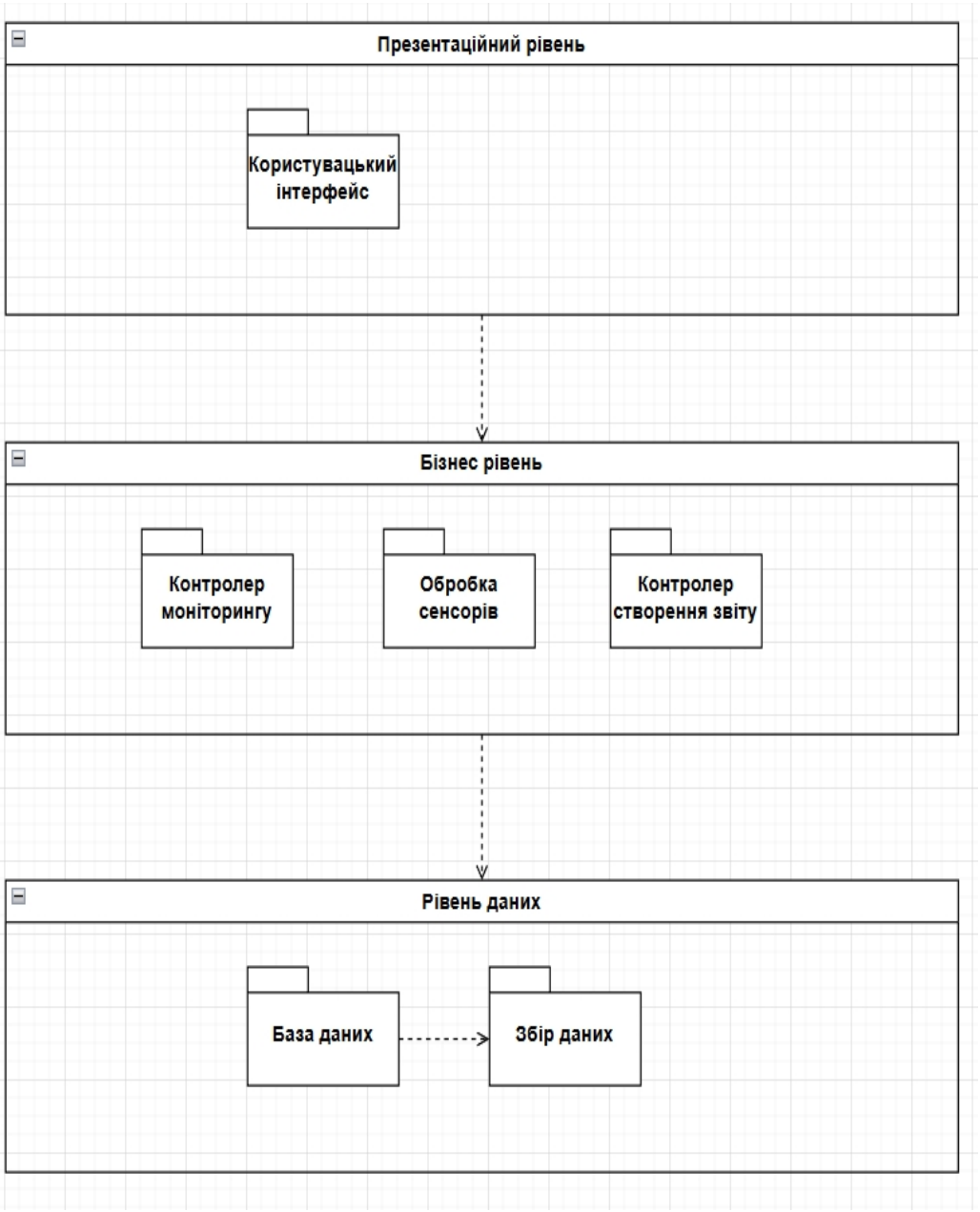


Рис.2.6 Діаграма пакетів

На цій діаграмі бачимо три рівні та пакети в них. У презентаційному рівні маємо пакет користувацького інтерфейсу. У бізнес рівні бачимо три пакети – контролер моніторингу, обробку сенсорів, контролер створення звіту. І останній рівень – це рівень даних в якому знаходяться два пакети бази даних та збору даних. Також бачимо залежність до кожного рівня та залежність між пакетами бази даних та збору даних.

2.4 Діаграма компонентів

Коротенько про те що таке діаграма компонентів

На діаграмах компонентів буде показано компоненти програмного, а також елементи, з яких вони складаються, такі як файли з початковими кодами, програмні бібліотеки або таблиці реляційних баз даних.

Компоненти можуть мати інтерфейси (тобто абстрактні класи з операціями), які надають змогу створювати асоціації між компонентами.

Переходимо до нашої діаграми компонентів (Рис.2.7)

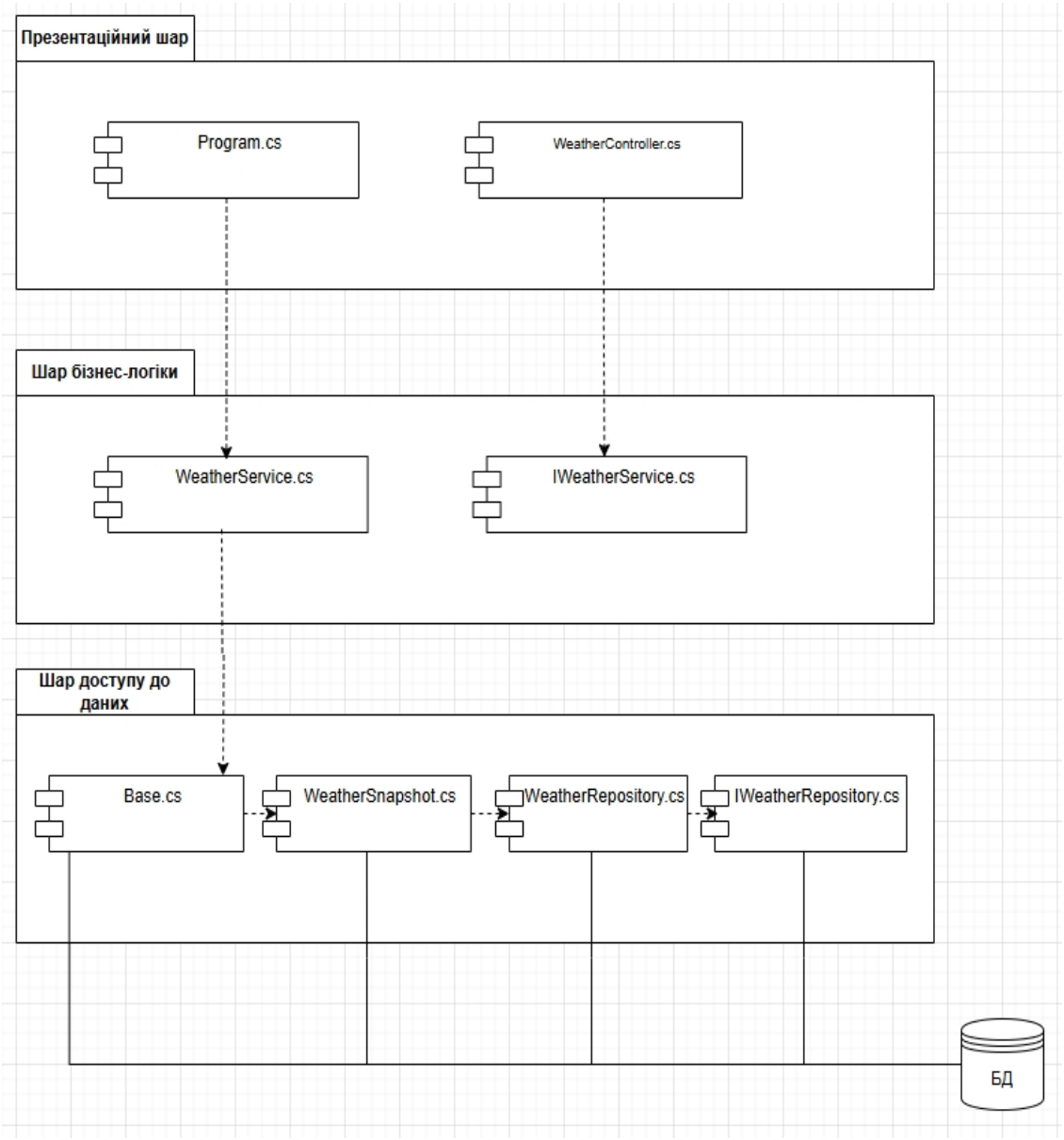


Рис.2.7 Діаграма компонентів

На цій діаграмі компонентів бачимо три шари та компоненти в них. У презентаційному шарі маємо два компоненти Program.cs та WeatherController.cs. У шарі бізнес-логіки маємо також два компонента WeatherService.cs та IWeatherService.cs. У останньому шарі, шар доступу до даних маємо чотири компонента – це Base.cs, WeatherSnapshot.cs, WeatherRepository.cs, IWeatherRepository.cs та їх асоціацію до БД. Бачимо залежності віл Program.cs до WeatherService.cs які залежать до усіх компонентів у шарі доступу до даних. Також залежність між WeatherController.cs та IWeatherService.cs. І на останок повна Base.cs до інших компонентів у шарі доступу до даних.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

Визначимо те що має робити інформаційна база системи моніторингу екологічних показників.

Функціонал системи моніторингу екологічних показників включає наступне:

1. Зберігання інформації показників
2. Створювати звіти

Так як, система буде отримувати дані у реальному часі в JSON форматі, було використано документно-орієнтована база даних, а саме MongoDB

MongoDB представляє найрозповсюдженішу на даний момент документо-орієнтовану систему управління базами даних. За різними оцінками входить до десятки найбільш використовуваних баз даних у світі.

Документи замість рядків

Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою інформацію. Документ можна уявити як сховище ключів і значень.

Ключ являє собою просту мітку, з якою асоційовано певний шматок даних.

Однак за всіх відмінностей є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційних СУБД зустрічається таке поняття як первинний ключ. Це поняття описує якийсь стовпець, який має унікальні значення. У MongoDB для кожного документа є унікальний ідентифікатор, який називається _id. І якщо явно не вказати його значення, то MongoDB автоматично згенерує для нього значення.

Кожному ключу зіставляється певне значення. Але тут також треба враховувати одну особливість: якщо в реляційних базах є чітко окреслена структура, де є поля, і якщо якесь поле не має значення, йому (залежно від налаштувань конкретної бд) можна присвоїти значення NULL. У MongoDB все інакше. Якщо якомусь ключу не зіставлено значення, то цей ключ просто опускається в документі і не вживається.

Колекції

Якщо в традиційному світі SQL є таблиці, то у світі MongoDB є колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об'єкти, то в колекції можуть містити найрізноманітніші об'єкти, що мають різну структуру і різний набір властивостей.

Реплікація

Уся система MongoDB може являти собою не тільки одну базу даних, яка розташовується на одному фізичному сервері. Функціональність MongoDB дає змогу розташувати кілька баз даних на кількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Система зберігання даних у MongoDB являє собою набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Усі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. І якщо основний вузол з якихось причин виходить з ладу, то один із вторинних вузлів стає головним.

Формат даних у MongoDB

Одним із популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних у MongoDB у цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (БіСон) або скорочення від binary JSON.

BSON дає змогу працювати з даними швидше: швидше виконується пошук і обробка. Хоча треба зазначити, що BSON на відміну від зберігання даних у форматі JSON має невеликий недолік: загалом дані в JSON-форматі займають менше місця, ніж у форматі BSON, з іншого боку, цей недолік з лишком окупається швидкістю.

Кросплатформеність

MongoDB написана на C++, тому її легко портувати на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і самому скомпілювати MongoDB, але рекомендується використовувати бібліотеки з офсайту.

Простота у використанні

Відсутність жорсткої схеми бази даних і у зв'язку з цим потреби при найменшій зміні концепції зберігання даних перестворювати цю схему значно полегшують роботу з базами даних MongoDB і подальшим їх масштабуванням. Крім того, економиться час розробників. Їм більше не треба думати про перестворення бази даних і витрачати час на побудову складних запитів.

Але, навіть з огляду на всі недоліки традиційних баз даних і переваги MongoDB, важливо розуміти, що завдання бувають різні, і методи їх вирішення бувають різні. У якійсь ситуації MongoDB дійсно поліпшить продуктивність вашого застосунку, наприклад, якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішаний підхід: зберігати один тип даних у MongoDB, а інший тип даних - у традиційних БД.

Для теми системи моніторингу екологічних показників MongoDB використовується з таких причин:

1. Показники можуть відрізнятися між собою, наприклад як температура, вологість, гази і тверді речовини. MongoDB дозволяє зберігати ці записи без структури
2. Так як показники будуть надаватися у великому обсязі. MongoDB дозволяє робити горизонтальне масштабування через таку річ як шардінг, завдяки цьому зберігаються та оброблюються дані. Шардінг – це процес збереження даних на декількох машинах
3. Також MongoDB дуже добре оптимізований під високу швидкість запису, якщо система моніторингу буде надавати показники щосекунди
4. Також із плюсів можна назвати налаштоване інтегрування з різними мовами програмування

Виходячи з усього цього можна сміливо казати що MongoDB являється самим вірним із усіх варіантів для системи моніторингу екологічних показників

3.2 Розробка інформаційної бази

Тут буде представлена готова колекція MongoDB, вже з справжніми даними у ній

Як вже казалось вище колекція це як таблиця у реляційній базі даних, але більш гнучка. І для колекції непотрібна схема, і завдяки цьому документ може вміщати велике різноманіття даних

Для системи моніторингу екологічних показників будуть братися лише декілька даних, а саме:

- Назва міста
- Час
- Показники (CO, NO_2, SO_2, O_3, PM2.5, PM10)

Нижче наведена повна колекція з даними в ній (Рис. 3.1)

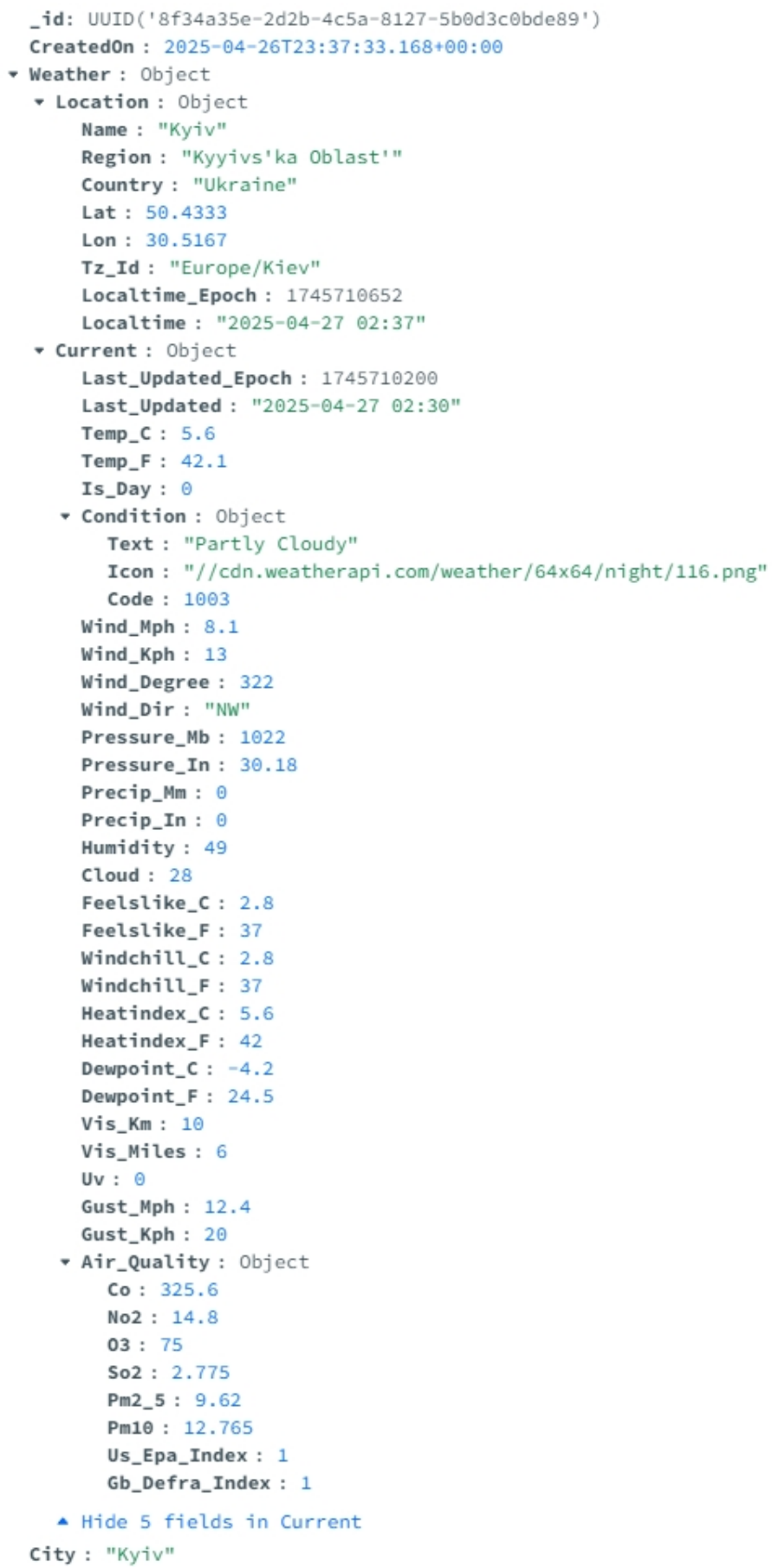


Рис.3.1 Колекція WeatherSnapshot

В цій колекції є велике перелік даних, але ті, які будуть братися вже наведені вище.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для системи моніторингу екологічних показників був вибраний наступний інструментарій

Для бекенду – C#, робота з кодом здійснюватиметься у Visual Studio 2022 [15].

Для фронтенду – це TypeScript [16] з бібліотекою React [17], робота з кодом здійснюватиметься у Visual Studio Code 2022 [18, 19]

Дані про показники будуть отримуватися з відкритої API – WeatherAPI

Докладніше про WeatherAPI можна буде дізнатися з посилання у використаних джерелах [21].

Мовою розробки було обрано C#. На сьогоднішній момент мова програмування C# - одна з найпотужніших, таких, що швидко розвиваються, і затребуваних мов в IT-галузі. Наразі на ній пишуть найрізноманітніші додатки: від невеликих десктопних програм до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

C# є мовою з Сі-подібним синтаксисом і близька в цьому відношенні до C++ і Java. Тому, якщо ви знайомі з однією з цих мов, то опанувати C# буде легше.

C# є об'єктно-орієнтованою і в цьому плані багато перейняла у Java та C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дає змогу розв'язати завдання з побудови великих, але водночас гнучких, масштабованих і розширюваних застосунків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональностей.

Роль платформи .NET

Коли говорять C#, нерідко мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, .NET MAUI). І, навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття пов'язані, ототожнювати їх невірно. Мова C# була створена спеціально для роботи з фреймворком .NET, однак саме поняття .NET дещо ширше.

Підтримка кількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд із C# це також VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi.NET. Під час компіляції код будь-якою з цих мов компілюється в збірку загальною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому за певних умов окремі модулі одного додатка можуть бути написані окремими мовами.

Кросплатформеність. .NET є переносною платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент - .NET 9 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти додатки мовою C# для найрізноманітніших платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І який би застосунок планувався до створення на C# - текстовий редактор, чат або складний веб-сайт - так чи інакше використовується бібліотека класів .NET.

Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти під час побудови тих чи інших додатків. Наприклад, для роботи з базами даних у цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних застосунків із багатим насиченим інтерфейсом - технологія WPF і WinUI, для

створення простіших графічних застосунків - Windows Forms. Для розробки кросплатформних мобільних і десктопних додатків - Xamarin/MAUI. Для створення веб-сайтів і веб-додатків - ASP.NET тощо.

До цього варто додати Blazor, який активно розвивається і набирає популярність, - фреймворк, що працює поверх .NET, який дає змогу створювати веб-додатки як на стороні сервера, так і на стороні клієнта. А в майбутньому буде підтримувати створення мобільних додатків і, можливо, десктоп-додатків.

Продуктивність. Згідно з низкою тестів веб-додатки на .NET у низці категорій сильно випереджають веб-додатки, побудовані за допомогою інших технологій. Додатки на .NET в принципі відрізняються високою продуктивністю.

Також ще слід зазначити таку особливість мови C# і фреймворка .NET, як автоматичне збирання сміття. А це означає, що нам у більшості випадків не доведеться, на відміну від C++, піклуватися про звільнення пам'яті. Вищезгадане загальномовне середовище CLR саме викличе збирач сміття і очистить пам'ять.

.NET Framework і .NET 9

Варто зазначити, що .NET довгий час розвивався переважно як платформа для Windows під назвою .NET Framework. У 2019 вийшла остання версія цієї платформи - .NET Framework 4.8. Вона більше не розвивається

З 2014 Microsoft почав розвивати альтернативну платформу - .NET Core, яка вже призначалася для різних платформ і повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Потім Microsoft послідовно випустив низку версій цієї платформи: .NET Core 1, .NET Core 2, .NET Core 3, .NET 5. І поточною версією є платформа .NET 9. Тому слід розрізняти .NET Framework, який призначений переважно для Windows, і кросплатформний .NET 9. У цьому посібнику йтиметься про C# 13 у зв'язці з .NET 9, оскільки це актуальна платформа.

Керований і некерований код

Нерідко додаток, створений на C#, називають керованим кодом (managed code). Що це означає? А це означає, що цей застосунок створено на основі платформи .NET і тому ним керує загальномовне середовище CLR, яке завантажує застосунок і за потреби очищає пам'ять. Але є також додатки, наприклад, створені на мові C++, які компілюються не в загальну мову CIL, як C#, VB.NET або F#, а у звичайний машинний код. У цьому випадку .NET не керує додатком.

Водночас платформа .NET надає можливості для взаємодії з некерованим кодом.

TypeScript (TS) – мова програмування, що є підмножиною JavaScript (JS), та будується на його основі. TypeScript використовує синтаксис JavaScript (з додаванням синтаксису для підтримки типів), та надає необов’язкову статичну типізацію.

Спочатку ви пишете код на TypeScript, після чого компілюєте його у JavaScript, за допомогою компілятора TypeScript. Після того, як отримано звичайний код JavaScript, можна розгорнути його в будь-якому середовищі, де виконується JavaScript.

Файли TypeScript мають розширення .ts, замість .js, як у файлів JavaScript.

Використання типів підвищує надійність, допомагаючи уникнути багатьох помилок під час виконання. Типи дають змогу виявляти деякі види помилок ще на етапі компіляції.

Отже, TypeScript можна вважати надбудовою над JavaScript. TypeScript додає необов’язкову статичну типізацію до JavaScript і допомагає уникнути потенційних помилок, що можуть виникнути під час виконання.

Ббібліотека React .

React JS — це відкритий JavaScript-фреймворк, а точніше, бібліотекою JavaScript, яка використовується для розробки інтерфейсів користувача. Він був створений компанією Facebook і швидко набув популярності серед розробників з усього світу. Реакт дозволяє ефективно створювати застосунки

з високою продуктивністю і масштабованістю. Одним з ключових концепцій у React JS є компоненти. Вони представляють собою незалежні блоки коду, які відповідають за рендеринг певної частини користувацького інтерфейсу.

React використовується для розробки користувацького інтерфейсу веб-додатків. Основною метою React Frontend є створення інтерактивних, динамічних та відзивчивих інтерфейсів для користувачів. React Front end дозволяє створювати багатофункціональні та інтерактивні застосунки зі швидким рендерингом і перехідом між сторінками. Також він має такі переваги:

- використання віртуального DOM та ефективного алгоритму оновлення дозволяє робити мінімальні зміни в реальному DOM, що покращує продуктивність застосунків;
- JS React базується на компонентній архітектурі, що дозволяє розбивати інтерфейс на незалежні компоненти. Це спрощує розробку, тестування та підтримку коду, оскільки компоненти можуть бути повторно використані та легко змінюються;
- Реакт пропагує односторонній потік даних, що сприяє простоті та передбачуваності управління станом додатків, полегшує відлагодження та тестування застосунків;
- спільнота розробників, що використовує Реакт велика та активна. Є безліч ресурсів, бібліотек та інструментів для розробки. Також існує багато сторонніх бібліотек, які підтримують React і допомагають розширити його функціональність;

React добре підходить для розробки проектів будь-якого масштабу. Він надає можливості для легкого розширення та перевикористання компонентів, інтеграції з іншими бібліотеками та фреймворками. Також підтримує серверний рендеринг, що дозволяє поліпшити швидкість завантаження сторінок та оптимізувати пошукову оптимізацію.

Visual Studio

Microsoft Visual Studio - це **інтегроване середовище розробки (IDE)** створений Microsoft для програмування програмного забезпечення на різних платформах. Завдяки його сумісності з численними мовами програмування, такими як **C#, Python, JavaScript та .NET** , став одним з найбільш використовуваних інструментів розробниками з усього світу. Ця IDE пропонує передові інструменти для налагодження, написання коду та управління проектами, роблячи роботу розробників більш ефективною та організованою.

Visual Studio - це платформа розробки, що дозволяє писати, тестувати та розповсюджувати програмне забезпечення для операційних систем Windows, macOS та Linux. Заснований у 1997 році, він з часом розвивався, впроваджуючи передові функції, такі як **IntelliSense, інтеграція з GitHub, Live Share та підтримка AI для завершення коду** .

Відрізняється від **Visual Studio Code** , який є легким та багатоплатформним редактором коду, тоді як Visual Studio - це повнофункціональне IDE з розширеними можливостями для циклу розробки програмного забезпечення.

Visual Studio використовується для розробки широкого спектра додатків, включаючи:

- **Додатки для робочого столу Windows** : завдяки підтримці для **.NET, C++ та інші технології** , ідеально підходить для створення бізнес-програмного забезпечення та настільних застосунків.
- **Веб-сайти та веб-додатки** : пропонує інтеграцію з **ASP.NET, HTML, CSS та JavaScript** , дозволяючи розробку динамічних сайтів та веб-базованих додатків.
- **Мобільний додаток для Android та iOS** : завдяки **Xamarin** , можна розробляти кросплатформені додатки з єдиною базою коду.

- **Розробка хмарних технологій** : інтеграція з **Microsoft Azure** дозволяє створювати, тестувати та розгортати додатки у хмарі.
- **Розробка відеоігор** : підтримує інструменти для **розробка гри з Unity** , роблячи його популярним вибором серед розробників ігор.
- **Машинне навчання та штучний інтелект** : завдяки інтеграції з бібліотеками **штучний інтелект та автоматичне навчання** , є корисним для проектів, що базуються на великих даних та AI.

Visual Studio включає низку передових інструментів для розробки програмного забезпечення, включаючи:

- **Розумний редактор коду** з **IntelliSense** , який автоматично пропонує завершення коду.
- **Розширена налагодження помилок** для виявлення помилок та оптимізації коду.
- **Підтримка контролю за вихідним кодом** з вбудованою інтеграцією з **Git та GitHub** .
- **Live Share** , що дозволяє реальний співробітництво між кількома розробниками.
- **Підтримка багатьох платформ** , дозволяючи розробку на Windows, macOS та Linux.
- **Налаштування за допомогою розширень** , щоб налаштувати IDE відповідно до своїх потреб.

Visual Studio Code.

Visual Studio Code (VS Code) – це безкоштовний, відкритий і кросплатформний редактор коду, розроблений компанією Microsoft. Вперше він був випущений у квітні 2015 року і з тих пір набув широкої популярності серед розробників.

На відміну від свого повноцінного IDE-аналога, Visual Studio, VS Code розроблений як легкий і добре настроюваний, що робить його ідеальним для розробників, які хочуть швидко та ефективно кодувати без зайвого роздуття.

Завдяки підтримці декількох мов програмування, інтегрованому Git-контролю, можливостям налагодження та великій бібліотеці розширень, VS Code став широко розповсюдженим інструментом для розробників програмного забезпечення по всьому світу.

C# був обраний завдяки його надійності, надійним систему робить те що C# поєднується с .NET, для системи моніторингу екологічних показників це є важливим так скільки обробка буде відбуватися з працювати з великою кількістю даних. Також C# надає зручну інтеграцію с різноманіттям баз даних, в тому числі і з базою даних MongoDB.

TypeScript та React були обрані завдяки тому що TypeScript надає змогу створювати додатку з чіткою структурою коду, зо знижує ймовірність помилок. React в свою чергу забезпечує гнучку розробку інтерфейсу, та підтримує компонентну архітектуру, що полегшує подальшу модифікацію системи. Також бібліотека має купу вже готових компонентів для побудови таблиць та усіляких форм.

3.4 Алгоритмізація та програмування програмних модулів

Для алгоритмізації винесу два алгоритми. Це алгоритм пошуку та алгоритм створення звіту.

Спочатку поясню та покажу алгоритм пошуку.

Пошук показників є ключовою функцією у системі моніторингу екологічних показників. Процес починається з того, що користувач вводить назву міста у текстове поле на сторінці додатку

Після натискання кнопки вводу, система перевіряє чи не є поле пустим, і якщо поле є заповненим надсилається запит до серверної частини програми

Сервер, отримавши назву, формує запит до API. Після отримання відповіді, сервер перевіряє наявність помилок, наприклад неправильну назву міста.

Якщо дані успішно отримано, вони передаються назад. Користувач бачить результат у вигляди вмісту речовин у повітрі. Якщо місто не знайдено, система проінформує про це користувача.

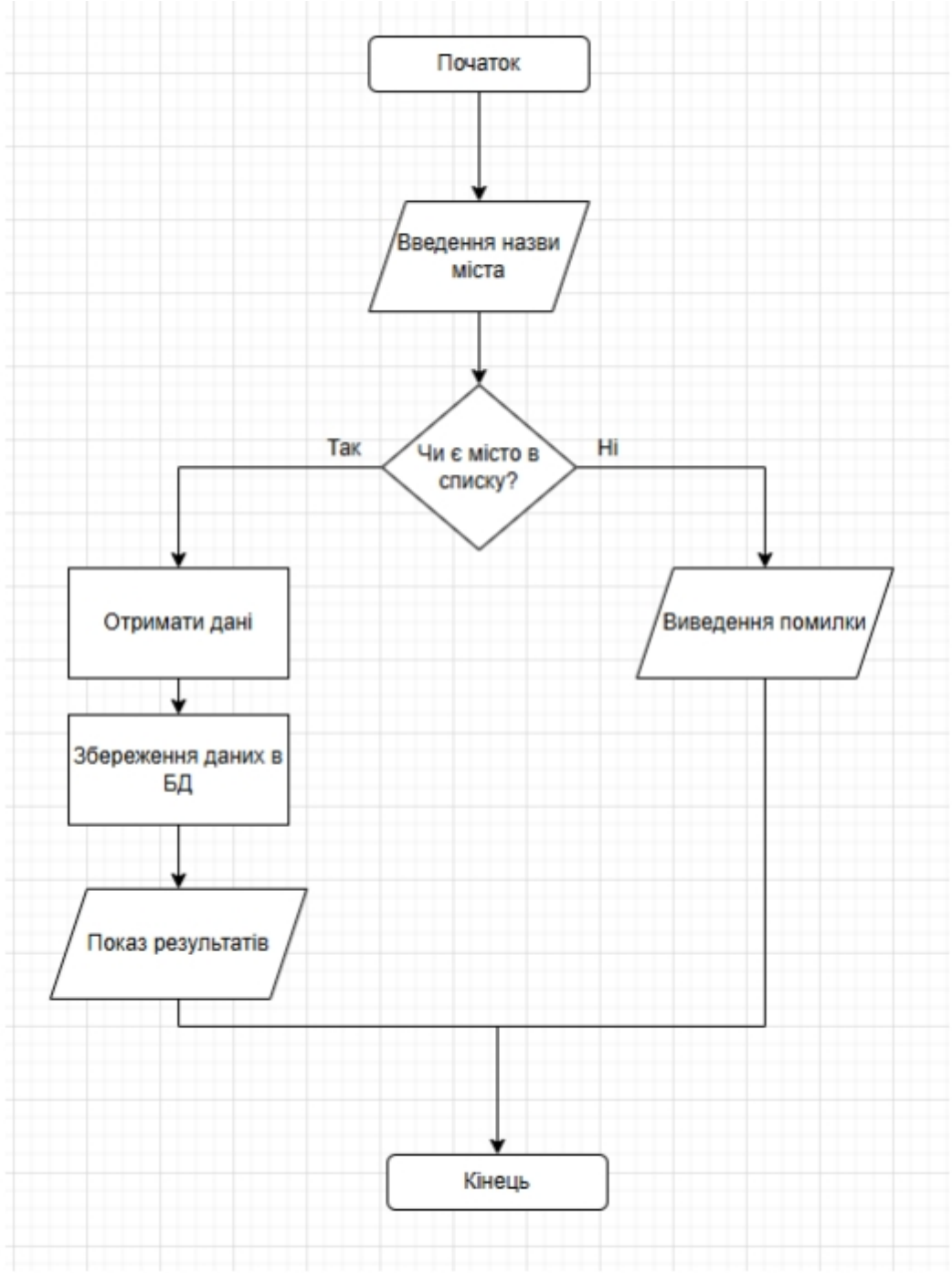


Рис.3.2 Алгоритм пошуку

Створення звіту надає змогу користувачу зафіксувати дані на момент запиту у зручному форматі. Сам процес створення звіту починається з вибору міста, за яким треба сформувати звіт. Також можна вказати часовий період, за яким система буде формувати звітність.

Після вибору міста та часу система надсилає запит до серверу. Сервер звертається до БД.

Після отримання даних, система обробляє їх і створює звіт

Створений звіт стає доступним для завантаження користувачу. Якщо ж користувач не вказав місто та час, буде надаватися звіт з усіма даними які є. Також якщо виникає помилка, наприклад неправильно вказане місто, система проінформує користувача про це.

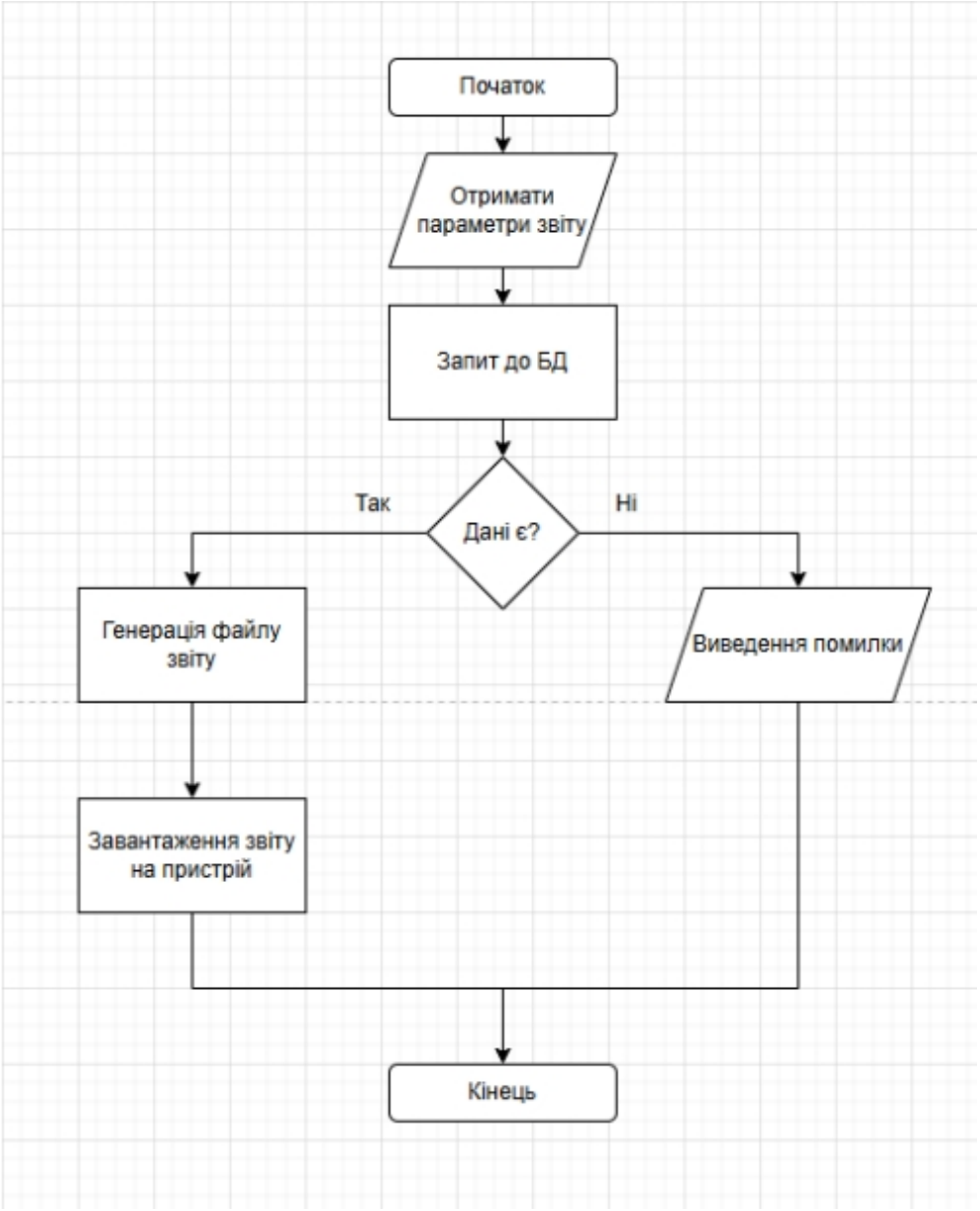


Рис.3.3 Алгоритм створення звіту

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Функціональне тестування — це тип тестування програмного забезпечення, під час якого система перевіряється на відповідність вимогам і специфікаціям. Функціональне тестування гарантує, що вимоги або специфікації належним чином задовольняються продуктом. Цей тип тестування, у першу чергу, стосується результату обробки. Він зосереджений на моделюванні фактичного використання системи, але не розробляє жодних припущень про структуру системи. За базу визначення цього типу тестування береться перевірка, що кожна функція програми працює відповідно до вимог і специфікацій. Цей вид не стосується вихідного коду програми. Кожна функціональність програми чи сайту перевіряється шляхом надання відповідних тестових вхідних даних, очікування вихідного результату та порівняння фактичного результату з очікуваним результатом.

Функціональне тестування — це загальний термін для багатьох типів тестів, призначених для перевірки всієї функціональності програмного забезпечення. Функціональне тестування підтверджує, що програмне забезпечення працює належним чином і не містить помилок. Щоб підтвердити це, тестувальник моделює реальний сценарій кінцевого користувача та порівнює результати тесту з очікуваними результатами, що зазначені у документі вимог. Функціональне тестування може виконуватися вручну або автоматизовано.

Здебільшого, функціональне тестування зосереджено на зовнішній поведінці програмного забезпечення, тому немає потреби у знаннях внутрішнього вихідного коду, і здебільшого воно вважається тестуванням чорного ящику. Цей тип тестування виконується перед нефункціональним

тестуванням. Існує кілька типів функціонального тестування, наприклад: димове тестування санітарне тестування регресійне тестування тощо.

Перейдемо до тестування функціоналу системи.

1. Пошук по місту
2. Створення звіту

Тестування пошуку (рис.4.1)

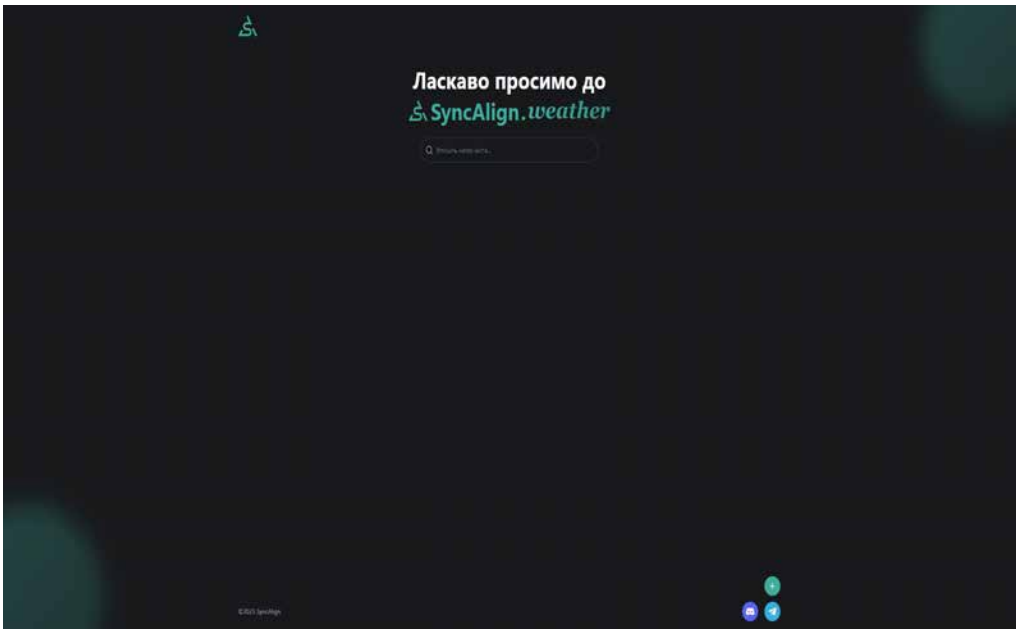


Рис.4.1 Головна сторінка

При переході на головну сторінку все працює, як те і очікується.

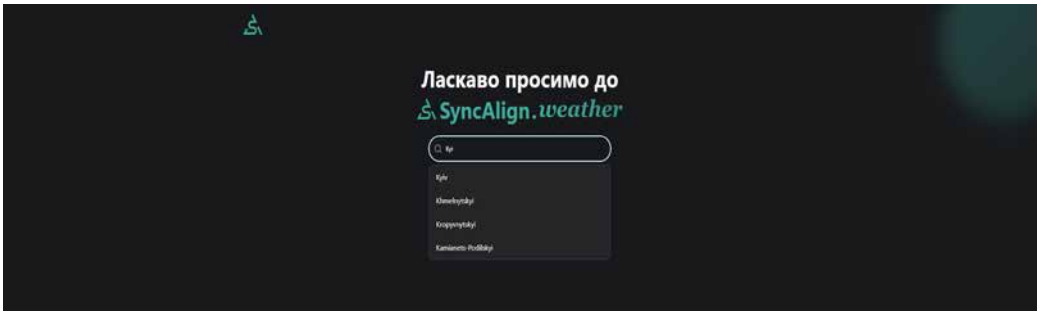


Рис.4.2 Текстове поле пошуку

При введені перших букв бачимо, що міста в яких є ці букву висвічуються для автозаповнення. Також все працює як треба

Рис.4.3 Сторінка показників

Коли ввели назву міста по котрому хочемо отримати інформацію показників, з’являється сторінка з показниками. Тобто також все працює у штатному режимі.

Результат спроби ввести неіснуюче місто представлено на рис. 4.4.

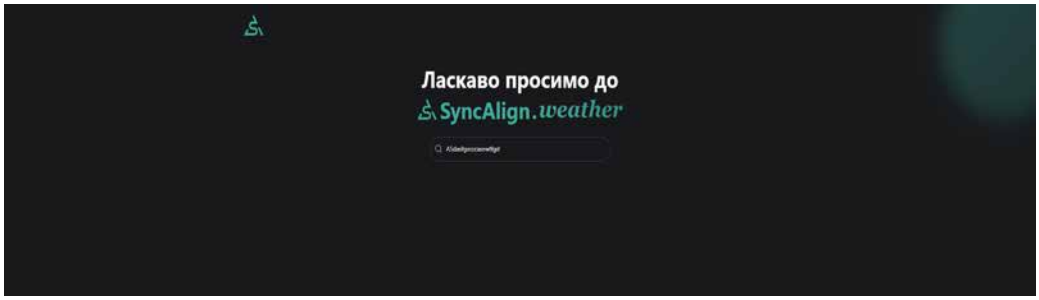


Рис.4.4 Неправильний ввід назви міста

Рис.4.5 Сторінка помилки

Як видно з рисунку при вводі неправильної назви міста, система відправляє на сторінку помилки. Це також є правильною робота системи

Результати функціонального тестування створення звіту наведені нижче.

На головній сторінці присутня кнопка на якій зображено «+», при наведені на який ховером з’являється кнопка «Створити звіт»

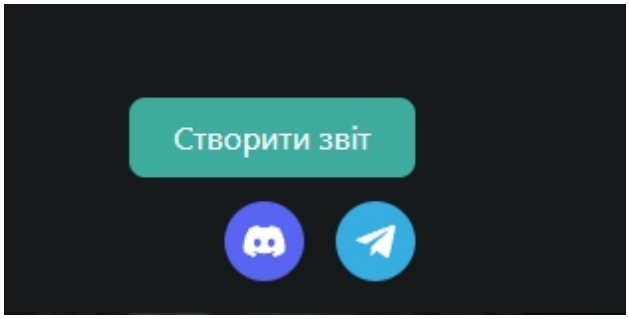


Рис.4.5 Кнопка «Створити звіт»

При натисканні кнопки відбувається перехід на нову сторінку по створенню звіту.

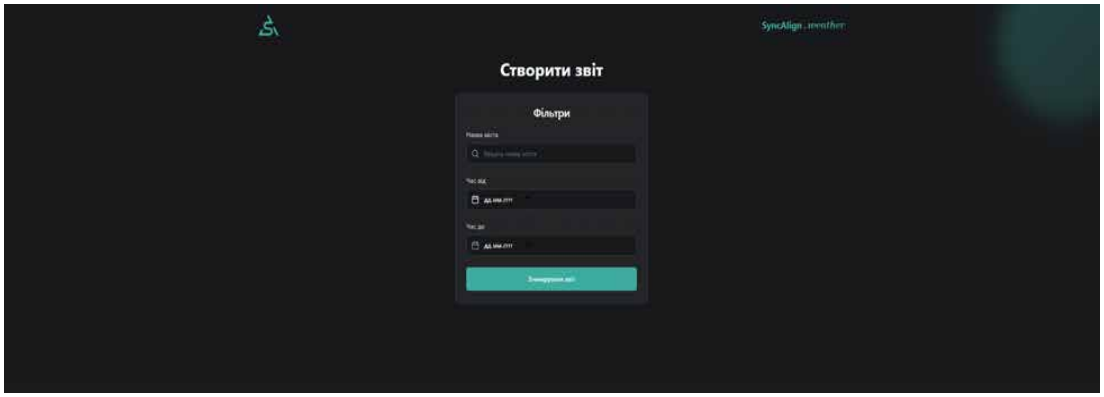


Рис.4.6 Сторінка створення звіту

Перехід повністю робочий, тому можливо казати що все працює належним чином

Далі при введені інформації по місту та часу має сгенеруватись та завантажитись звіт у форматі Excel.

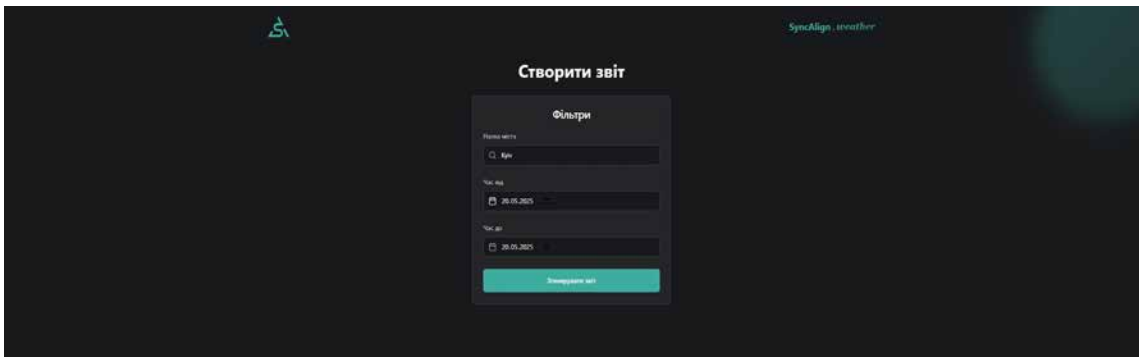


Рис.4.7 Дані які ввели

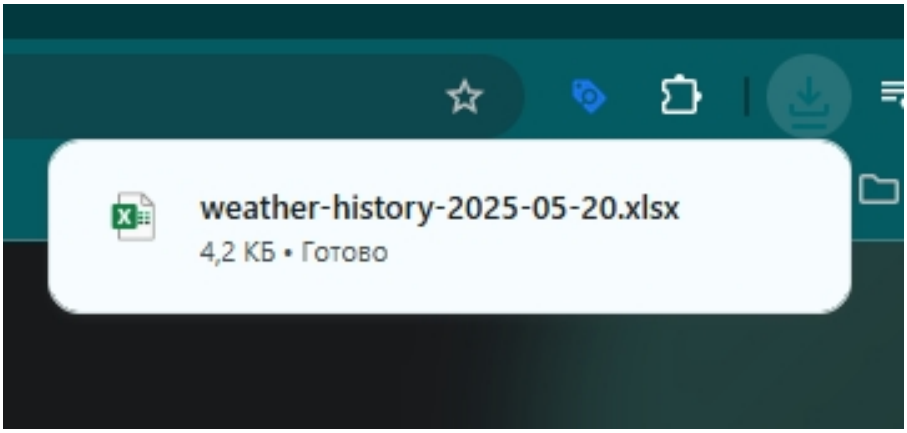
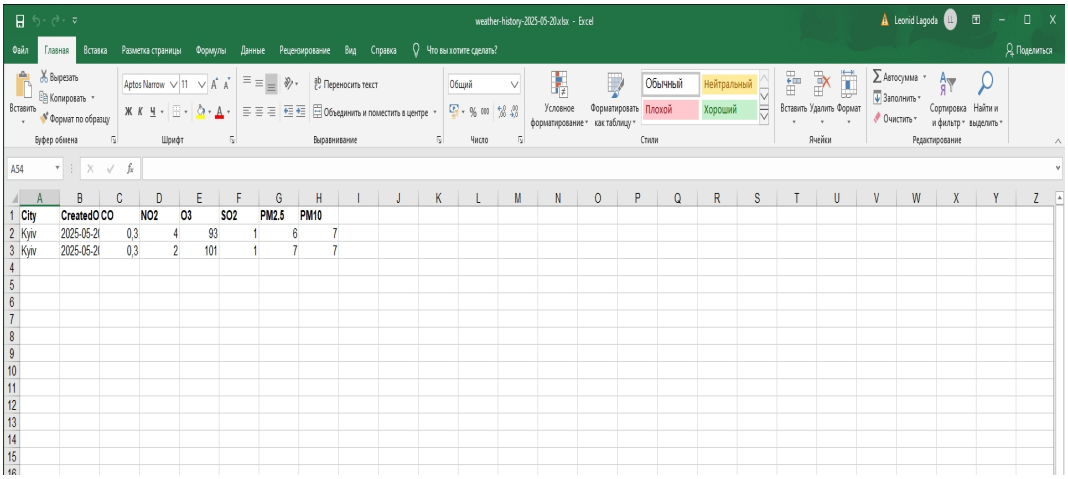


Рис.4.8 Файл який завантажився



	A	B	C	D	E	F	G	H
1	City	CreatedO	CO	NO2	O3	SO2	PM2.5	PM10
2	Kyiv	2025-05-2(	0,3	4	93	1	6	7
3	Kyiv	2025-05-2(	0,3	2	101	1	7	7

Рис.4.9 Дані які знаходяться у файлі

Як видно функція створення звіту працює повністю справно.

У випадку не внесення даних або внесення невірних даних спостерігається такий результат роботи програми.

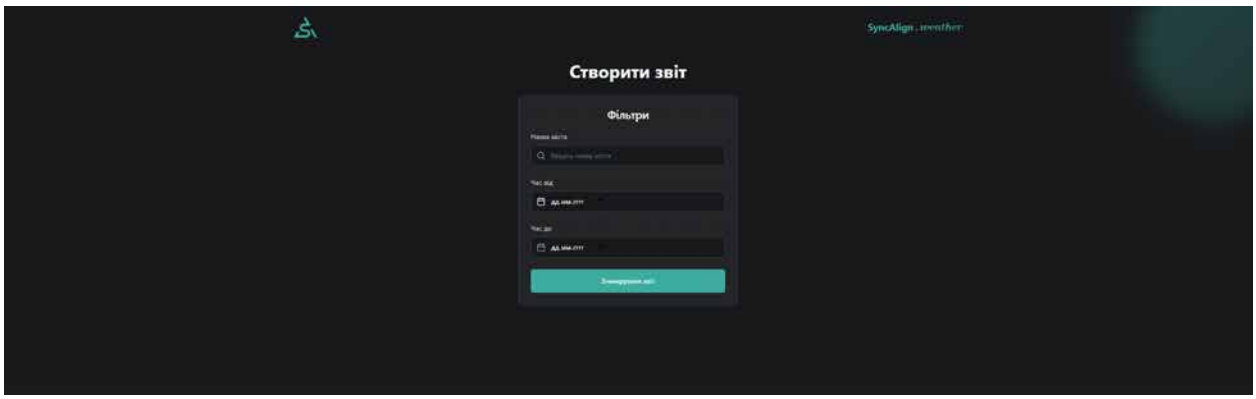
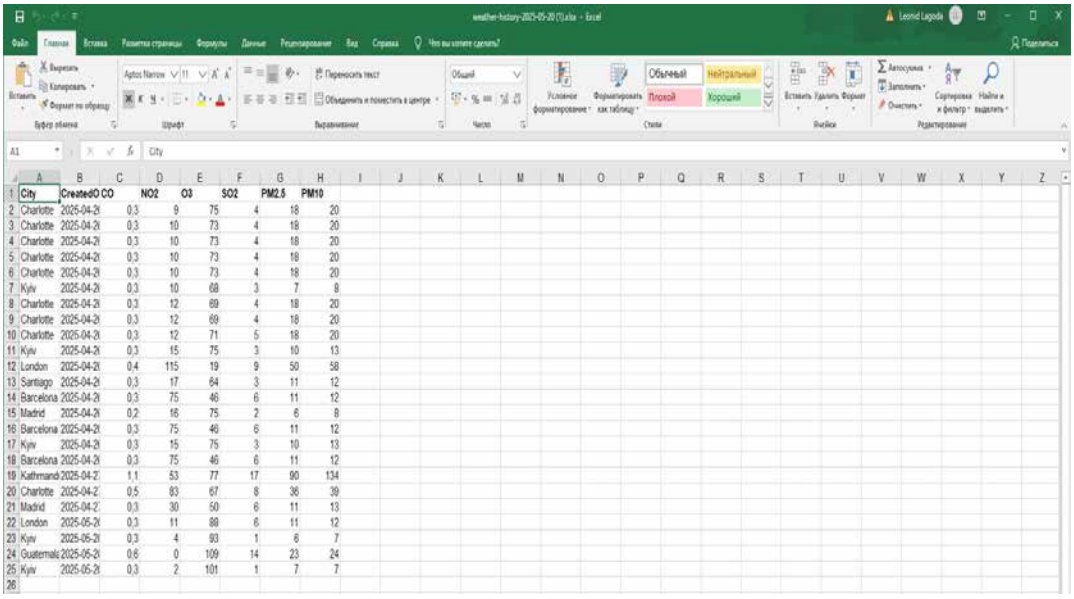


Рис.4.10 Сторінка створення звіту



	A	B	C	D	E	F	G	H
1	City	CreatedO	CO	NO2	O3	SO2	PM2.5	PM10
2	Charlotte	2025-04-2	0,3	9	75	4	18	20
3	Charlotte	2025-04-2	0,3	10	73	4	18	20
4	Charlotte	2025-04-2	0,3	10	73	4	18	20
5	Charlotte	2025-04-2	0,3	10	73	4	18	20
6	Charlotte	2025-04-2	0,3	10	73	4	18	20
7	Kyiv	2025-04-2	0,3	10	68	3	7	8
8	Charlotte	2025-04-2	0,3	12	69	4	18	20
9	Charlotte	2025-04-2	0,3	12	69	4	18	20
10	Charlotte	2025-04-2	0,3	12	71	5	18	20
11	Kyiv	2025-04-2	0,3	15	75	3	10	13
12	London	2025-04-2	0,4	115	19	9	50	58
13	Santiago	2025-04-2	0,3	17	64	3	11	12
14	Barcelona	2025-04-2	0,3	75	46	6	11	12
15	Madrid	2025-04-2	0,2	18	75	2	6	8
16	Barcelona	2025-04-2	0,3	75	46	6	11	12
17	Kyiv	2025-04-2	0,3	15	75	3	10	13
18	Barcelona	2025-04-2	0,3	75	46	6	11	12
19	Kathmand	2025-04-2	1,1	53	77	17	90	134
20	Charlotte	2025-04-2	0,5	83	67	8	36	39
21	Madrid	2025-04-2	0,3	30	50	6	11	13
22	London	2025-05-2	0,3	11	88	6	11	12
23	Kyiv	2025-05-2	0,3	4	93	1	6	7
24	Guatemal	2025-05-2	0,6	0	109	14	23	24
25	Kyiv	2025-05-2	0,3	2	101	1	7	7
26								

Рис.4.11Файл завантажений коли не було внесено дані

Як описувалось раніше, якщо не внести дані по фільтрам то буде завантажений файл з усіма даними які є. Тобто також все працює як потрібно

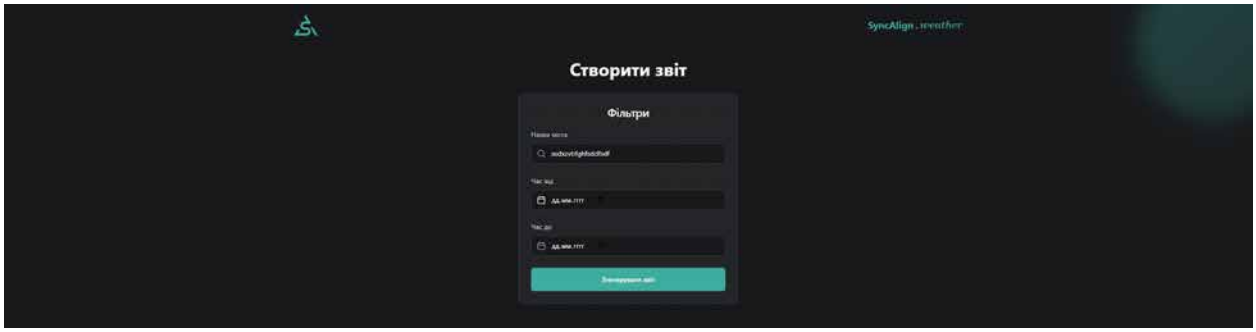


Рис.4.12 Невірно введені дані

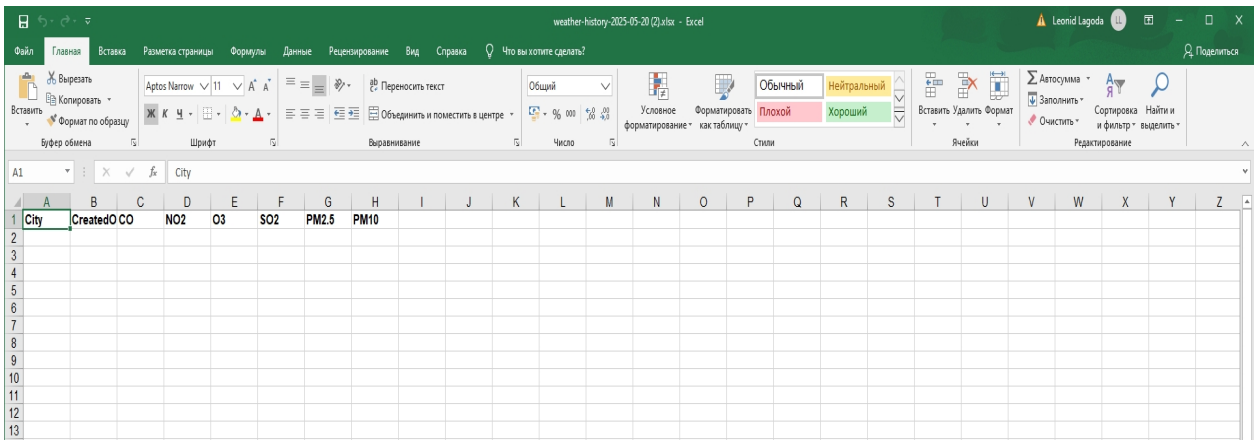


Рис.4.13 Файл при якому дані були невірними

Також як описувалось раніше, при невірному введені даних буде завантажено повністю пустий файл. Знову можемо казати що система функціонує повністю справно.

В результаті функціонального тестування було виявлено що система моніторингу екологічних показників, працює рівно так як і було заявлено і відповідає всім функціям які були закладені.

4.2 Вимоги до апаратного та програмного забезпечення

У цьому пункті будуть наведені вимоги до апаратного та програмного забезпечення до системи моніторингу екологічних показників.

Апаратні вимоги – це вимоги до апаратного забезпечення (процесор, відеокарта тощо).

Програмні вимоги – це вимоги до программного забезпечення.

Апаратні вимоги

Мінімальні:

- процесор: 2 ядра, 1.8 GHz;
- оперативна пам'ять: 4 GB RAM;
- жорсткий диск: 10 GB вільного місця (HDD);
- відеокарта: з підтримкою DirectX 11, 1 GB відеопам'яті;
- мережа: 100 Mbps Ethernet.

Рекомендовані:

● процесор: 4 ядра, 2.4+ GHz;

● оперативна пам'ять: 8+ GB RAM;

● жорсткий диск: 20+ GB вільного місця (SSD);

● відеокарта: з підтримкою DirectX 12, 2+ GB відеопам'яті;

● мережа: Gigabit Ethernet.

Програмні вимоги

Мінімальні:

● ОС: Windows 10 версія 21H2 або новіша;

● DirectX 11. Рекомендовані:

● ОС: Windows 11 22H2 або новіша;

● DirectX 12;

● оновлені драйвери відеокарти для оптимальної роботи WPF. Встановлений

Браузер (Наприклад, **Chrome**).

ВИСНОВКИ

При виконанні бакалаврської кваліфікаційної роботи було розроблено програмне забезпечення системи моніторингу екологічних показників.

Для розробки програмного забезпечення були обрані наступні інструменти

1. База даних на MongoDB, так як він являється самим поширеним серед документо-орієнтованих баз даних

Для розробки програмного продукту було використано середовище Visual Studio 2022 та Visual Studio Code 2022, тому як також являются самим розпоширеним середовищем для роботи з кодом. Мова програмування яка була обрана C#, завдяки його надійності. Для розробки фронт частини програмного продукту було обрано інструментарій React та TypeScript, обрані були завдяки їх простоті та багатому функціоналу.

В якості інформаційної системи було обрано документо-орієнтовану базу даних, що надало можливість зберігати велику кількість даних різного типу.

Програмне забезпечення було розроблено з реалізацією всіх поставленими завданнями, що дає змогу оперативно дізнатися поточні показники забруднення повітря та зміг сформувати звіт на основі цих показників.

Надано рекомендації щодо впровадження та експлуатації системи. Проведено функціональне тестування системи моніторингу екологічних показників та показана вже сама готова система. При тестуванні не було виявлено жодної проблеми і все працювало у штатному режимі. І в останньому пункті були наведені вимоги до апаратного та програмного забезпечення користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Екологічний моніторинг (https://nubip.edu.ua/sites/default/files/u243/24.pdf)

2. Джигирей В.С. Екологія та охорона навколишнього природного середовища. — К., 2006; Словник-довідник з екології / К.М. Ситник, О.В. Брайон, А.В. Городецький та ін. — К., 2004; Стольберг Ф.В. Экология города. — К., 2000.
(https://www.pharmencyclopedia.com.ua/article/1500/monitoring-ekologichnij)

3. Що таке функціональні та нефункціональні вимоги (https://visuresolutions.com/uk/посібник-з-відстеження-управління-вимогами/функціональні-та-нефункціональні-вимоги/#elementor-toc__heading-anchor-1)

4.Що таке UML-діаграми (https://evergreens.com.ua/ua/articles/uml-diagrams.html)

5.Інформація щодо ISO 14000 (https://ru.wikipedia.org/wiki/ISO_14000)

6. Методичні вказівки щодо врахування кліматичного компонента в документах державного планування та під час здійснення стратегічної екологічної оцінки та оцінки впливу на довкілля (https://zakon.rada.gov.ua/rada/show/v1382926-24#n12)

7. Про затвердження Методичних рекомендацій щодо здійснення післяпроектного моніторингу
(https://zakon.rada.gov.ua/rada/show/v0291926-24#Text)

8. Що таке ER-діаграма (https://www.guru99.com/uk/er-diagram-tutorial-dbms.html)

9.Що собою являє документно-орієнтована база даних (https://aws.amazon.com/ru/nosql/document/)

10.Різновид UML-діаграм(https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html)

11.Діаграма кооперацій (https://studfile.net/preview/5010027/page:4/)

12.Перекладач (https://www.deepl.com/ru/translator)

13.Діаграма пакетів (https://blog.visual-paradigm.com/ru/what-is-a-package-what-is-a-package-diagram-in-uml/)

14.Введеня в MongoDB (https://metanit.com/nosql/mongodb/1.1.php)

15.Введення в C# та .NET (https://metanit.com/sharp/tutorial/1.1.php)

16.Введення в TypeScript (https://www.it-notes.wiki/javascript/typescript/what-is-typescript/)

17.Що таке бібліотека React (https://cases.media/en/article/sho-take-react-js-yak-pochati-vivchati-reakt-navichki-dlya-react-developer?srsltid=AfmBOorBqed2Ted9O4C9TWU-i2j9NGBRC1EQ8Om5gCLy6w9xyx4WOVvH)

18.Що таке Visual Studio (https://macrosoft.store/uk/blog/post/29-%D0%B4%D0%BB%D1%8F-%D1%87%D0%BE%D0%B3%D0%BE-%D0%BF%D0%BE%D1%82%D1%80%D1%96%D0%B1%D0%BD%D0%B0-microsoft-visual-studio)

19.Visual Studio Code (https://top-keis.com.ua/shho-take-visual-studio-code/)

20.Що таке функціональні вимоги (https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/)

21.https://openweathermap.org/api

ДОДАТКИ

ДОДАТОК А

База даних

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Configuration;
namespace MongoDB
{
  public static class MongoDbExtensions
  {
    public static IServiceCollection AddMongoDb(this IServiceCollection services,
IConfiguration config)
    {
      services.Configure<MongoDBSettings>(
        config.GetSection(MongoDBSettings.SectionName));

      services.AddSingleton<MongoDBStore>();

      return services;
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace MongoDB
{
  public class MongoDBSettings
  {
    public const string SectionName = "MongoDB";
    public string ConnectionString { get; set; }
    public string DatabaseName { get; set; }
  }
}
using MongoDB.Driver;
using Microsoft.Extensions.Options;
using MongoDB.DAL;
namespace MongoDB
{
  public class MongoDBStore
  {
    private readonly IMongoDatabase _database;
    public IMongoCollection<WeatherSnapshot> WeatherSnapshot { get; }
    public MongoDBStore(IOptions<MongoDBSettings> settings)
    {
      var client = new MongoClient(settings.Value.ConnectionString);
      _database = client.GetDatabase(settings.Value.DatabaseName);
  WeatherSnapshot =
_database.GetCollection<WeatherSnapshot>(nameof(WeatherSnapshot));
    }
```

```
    }
}
```

ДОДАТОК Б

Код Програми (Бек)

```
using Microsoft.AspNetCore.Mvc;
using MongoDB.DAL;
using SyncAlignWeather.Identity.Dto;
using SyncAlignWeather.Identity.Services.Interfaces;

namespace SyncAlignWeather.Identity.Controllers
{
    [ApiController]
    [Route("api/weather")]
    public class WeatherController : ControllerBase
    {
        private readonly IWeatherService _weatherService;

        public WeatherController(IWeatherService weatherService)
        {
            _weatherService = weatherService;
        }

        [HttpPost("{city}")]
        public async Task<ActionResult<GetWeatherDataDto>> GetWeather(string city)
        {
            try
            {
                var weather = await _weatherService.GetCurrentWeatherAsync(city);
                return Ok(weather);
            }
            catch (HttpRequestException ex)
            {
                return StatusCode(400, "EROOOR");
            }
        }

        [HttpGet("history")]
        public async Task<ActionResult<List<GetWeatherFilteredDataDto>>> GetFilteredWeather(
            [FromQuery] string? city,
            [FromQuery] DateTime? from,
            [FromQuery] DateTime? to)
        {
            try
            {
                var results = await _weatherService.GetFilteredAsync(city, from, to);
                return Ok(results);
            }
            catch (Exception ex)
            {
                return StatusCode(400, "EROOOR");
            }
        }
```

```
        [HttpGet("history/export")]
        public async Task<IActionResult> ExportFilteredWeatherToExcel(
            [FromQuery] string? city,
            [FromQuery] DateTime? from,
            [FromQuery] DateTime? to)
        {
            try
            {
                var excelBytes = await _weatherService.GetFilteredExcelAsync(city, from, to);

                return File(
                    excelBytes,
                    "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
                    $"weather-history-{DateTime.UtcNow:yyyyMMddHHmmss}.xlsx"
                );
            }
            catch (Exception ex)
            {
                return StatusCode(400, "EROOOR");
            }
        }

    }
}


using MongoDB.DAL;
using SyncAlignWeather.Identity.Dto;

namespace SyncAlignWeather.Identity.Converters.ToDal
{
    public static class Converter
    {
        public static Weather ToDal(this WeatherDataDto dto)
        {
            return new Weather
            {
                Location = new MongoDB.DAL.Location
                {
                    Name = dto.Location.Name,
                    Region = dto.Location.Region,
                    Country = dto.Location.Country,
                    Lat = dto.Location.Lat,
                    Lon = dto.Location.Lon,
                    Tz_Id = dto.Location.Tz_Id,
                    Localtime_Epoch = dto.Location.Localtime_Epoch,
                    Localtime = dto.Location.Localtime
                },
                Current = new MongoDB.DAL.CurrentWeather
                {
                    Last_Updated_Epoch = dto.Current.Last_Updated_Epoch,
                    Last_Updated = dto.Current.Last_Updated,
                    Temp_C = dto.Current.Temp_C,
                    Temp_F = dto.Current.Temp_F,
                    Is_Day = dto.Current.Is_Day,
                    Condition = new MongoDB.DAL.Condition
                    {
```

```
                    Text = dto.Current.Condition.Text,
                    Icon = dto.Current.Condition.Icon,
                    Code = dto.Current.Condition.Code
                },
                Wind_Mph = dto.Current.Wind_Mph,
                Wind_Kph = dto.Current.Wind_Kph,
                Wind_Degree = dto.Current.Wind_Degree,
                Wind_Dir = dto.Current.Wind_Dir,
                Pressure_Mb = dto.Current.Pressure_Mb,
                Pressure_In = dto.Current.Pressure_In,
                Precip_Mm = dto.Current.Precip_Mm,
                Precip_In = dto.Current.Precip_In,
                Humidity = dto.Current.Humidity,
                Cloud = dto.Current.Cloud,
                Feelslike_C = dto.Current.Feelslike_C,
                Feelslike_F = dto.Current.Feelslike_F,
                Windchill_C = dto.Current.Windchill_C,
                Windchill_F = dto.Current.Windchill_F,
                Heatindex_C = dto.Current.Heatindex_C,
                Heatindex_F = dto.Current.Heatindex_F,
                Dewpoint_C = dto.Current.Dewpoint_C,
                Dewpoint_F = dto.Current.Dewpoint_F,
                Vis_Km = dto.Current.Vis_Km,
                Vis_Miles = dto.Current.Vis_Miles,
                Uv = dto.Current.Uv,
                Gust_Mph = dto.Current.Gust_Mph,
                Gust_Kph = dto.Current.Gust_Kph,
                Air_Quality = new MongoDB.DAL.AirQuality
                {
                    Co = dto.Current.Air_Quality.Co,
                    No2 = dto.Current.Air_Quality.No2,
                    O3 = dto.Current.Air_Quality.O3,
                    So2 = dto.Current.Air_Quality.So2,
                    Pm2_5 = dto.Current.Air_Quality.Pm2_5,
                    Pm10 = dto.Current.Air_Quality.Pm10,
                    Us_Epa_Index = dto.Current.Air_Quality.UsEpaIndex,
                    Gb_Defra_Index = dto.Current.Air_Quality.GbDefraIndex
                }
            }
        };
    }
  }
}
using MongoDB.DAL;
using SyncAlignWeather.Identity.Domain;
using SyncAlignWeather.Identity.Dto;

namespace SyncAlignWeather.Identity.Converters.ToDomain
{
  public static class Converter
  {
    public static WeatherSnapshotDomain ToDomain(this WeatherDataDto dto)
    {
      return new WeatherSnapshotDomain(
        new Domain.Location(
          dto.Location.Name,
```

```
            dto.Location.Region,
            dto.Location.Country,
            dto.Location.Lat,
            dto.Location.Lon,
            dto.Location.Tz_Id,
            dto.Location.Localtime_Epoch,
            dto.Location.Localtime
        ),
        new Domain.CurrentWeather(
            dto.Current.Last_Updated_Epoch,
            dto.Current.Last_Updated,
            dto.Current.Temp_C,
            dto.Current.Temp_F,
            dto.Current.Is_Day,
            new Domain.Condition(
                dto.Current.Condition.Text,
                dto.Current.Condition.Icon,
                dto.Current.Condition.Code
            ),
            dto.Current.Wind_Mph,
            dto.Current.Wind_Kph,
            dto.Current.Wind_Degree,
            dto.Current.Wind_Dir,
            dto.Current.Pressure_Mb,
            dto.Current.Pressure_In,
            dto.Current.Precip_Mm,
            dto.Current.Precip_In,
            dto.Current.Humidity,
            dto.Current.Cloud,
            dto.Current.Feelslike_C,
            dto.Current.Feelslike_F,
            dto.Current.Windchill_C,
            dto.Current.Windchill_F,
            dto.Current.Heatindex_C,
            dto.Current.Heatindex_F,
            dto.Current.Dewpoint_C,
            dto.Current.Dewpoint_F,
            dto.Current.Vis_Km,
            dto.Current.Vis_Miles,
            dto.Current.Uv,
            dto.Current.Gust_Mph,
            dto.Current.Gust_Kph,
            new Domain.AirQuality(
                dto.Current.Air_Quality.Co,
                dto.Current.Air_Quality.No2,
                dto.Current.Air_Quality.O3,
                dto.Current.Air_Quality.So2,
                dto.Current.Air_Quality.Pm2_5,
                dto.Current.Air_Quality.Pm10,
                dto.Current.Air_Quality.UsEpaIndex,
                dto.Current.Air_Quality.GbDefraIndex
            )
        ),
        null,
        null
    );
```

```
}

public static WeatherSnapshotDomain ToDomain(this WeatherSnapshot snapshot)
{
   return new WeatherSnapshotDomain(
      new Domain.Location(
         snapshot.Weather.Location.Name,
         snapshot.Weather.Location.Region,
         snapshot.Weather.Location.Country,
         snapshot.Weather.Location.Lat,
         snapshot.Weather.Location.Lon,
         snapshot.Weather.Location.Tz_Id,
         snapshot.Weather.Location.Localtime_Epoch,
         snapshot.Weather.Location.Localtime
      ),
      new Domain.CurrentWeather(
         snapshot.Weather.Current.Last_Updated_Epoch,
         snapshot.Weather.Current.Last_Updated,
         snapshot.Weather.Current.Temp_C,
         snapshot.Weather.Current.Temp_F,
         snapshot.Weather.Current.Is_Day,
         new Domain.Condition(
            snapshot.Weather.Current.Condition.Text,
            snapshot.Weather.Current.Condition.Icon,
            snapshot.Weather.Current.Condition.Code
         ),
         snapshot.Weather.Current.Wind_Mph,
         snapshot.Weather.Current.Wind_Kph,
         snapshot.Weather.Current.Wind_Degree,
         snapshot.Weather.Current.Wind_Dir,
         snapshot.Weather.Current.Pressure_Mb,
         snapshot.Weather.Current.Pressure_In,
         snapshot.Weather.Current.Precip_Mm,
         snapshot.Weather.Current.Precip_In,
         snapshot.Weather.Current.Humidity,
         snapshot.Weather.Current.Cloud,
         snapshot.Weather.Current.Feelslike_C,
         snapshot.Weather.Current.Feelslike_F,
         snapshot.Weather.Current.Windchill_C,
         snapshot.Weather.Current.Windchill_F,
         snapshot.Weather.Current.Heatindex_C,
         snapshot.Weather.Current.Heatindex_F,
         snapshot.Weather.Current.Dewpoint_C,
         snapshot.Weather.Current.Dewpoint_F,
         snapshot.Weather.Current.Vis_Km,
         snapshot.Weather.Current.Vis_Miles,
         snapshot.Weather.Current.Uv,
         snapshot.Weather.Current.Gust_Mph,
         snapshot.Weather.Current.Gust_Kph,
         new Domain.AirQuality(
            snapshot.Weather.Current.Air_Quality.Co,
            snapshot.Weather.Current.Air_Quality.No2,
            snapshot.Weather.Current.Air_Quality.O3,
            snapshot.Weather.Current.Air_Quality.So2,
            snapshot.Weather.Current.Air_Quality.Pm2_5,
            snapshot.Weather.Current.Air_Quality.Pm10,
```

```
                        snapshot.Weather.Current.Air_Quality.Us_Epa_Index,
                        snapshot.Weather.Current.Air_Quality.Gb_Defra_Index
                    )
                ),
                snapshot.City,
                snapshot.CreatedOn
            );
        }
    }
}
using SyncAlignWeather.Identity.Domain;
using SyncAlignWeather.Identity.Dto;

namespace SyncAlignWeather.Identity.Converters.ToDto
{
    public static class Converter
    {
        public static GetWeatherDataDto ToDto(this WeatherSnapshotDomain
weatherSnapshotDomain)
        {
            return new GetWeatherDataDto
            {
                Location = new GetWeatherDataLocation
                {
                    Name = weatherSnapshotDomain.Location.Name,
                    Region = weatherSnapshotDomain.Location.Region,
                    Country = weatherSnapshotDomain.Location.Country,
                    Lat = weatherSnapshotDomain.Location.Lat,
                    Lon = weatherSnapshotDomain.Location.Lon,
                    Tz_Id = weatherSnapshotDomain.Location.Tz_Id,
                    Localtime_Epoch = weatherSnapshotDomain.Location.Localtime_Epoch,
                    Localtime = weatherSnapshotDomain.Location.Localtime
                },
                Current = new GetCurrentWeather
                {
                    Last_Updated_Epoch = weatherSnapshotDomain.Current.Last_Updated_Epoch,
                    Last_Updated = weatherSnapshotDomain.Current.Last_Updated,
                    Temp_C = weatherSnapshotDomain.Current.Temp_C,
                    Temp_F = weatherSnapshotDomain.Current.Temp_F,
                    Is_Day = weatherSnapshotDomain.Current.Is_Day,
                    Wind_Mph = weatherSnapshotDomain.Current.Wind_Mph,
                    Wind_Kph = weatherSnapshotDomain.Current.Wind_Kph,
                    Wind_Degree = weatherSnapshotDomain.Current.Wind_Degree,
                    Wind_Dir = weatherSnapshotDomain.Current.Wind_Dir,
                    Pressure_Mb = weatherSnapshotDomain.Current.Pressure_Mb,
                    Pressure_In = weatherSnapshotDomain.Current.Pressure_In,
                    Precip_Mm = weatherSnapshotDomain.Current.Precip_Mm,
                    Precip_In = weatherSnapshotDomain.Current.Precip_In,
                    Humidity = weatherSnapshotDomain.Current.Humidity,
                    Cloud = weatherSnapshotDomain.Current.Cloud,
                    Feelslike_C = weatherSnapshotDomain.Current.Feelslike_C,
                    Feelslike_F = weatherSnapshotDomain.Current.Feelslike_F,
                    Windchill_C = weatherSnapshotDomain.Current.Windchill_C,
                    Windchill_F = weatherSnapshotDomain.Current.Windchill_F,
                    Heatindex_C = weatherSnapshotDomain.Current.Heatindex_C,
                    Heatindex_F = weatherSnapshotDomain.Current.Heatindex_F,
```

```
                    Dewpoint_C = weatherSnapshotDomain.Current.Dewpoint_C,
                    Dewpoint_F = weatherSnapshotDomain.Current.Dewpoint_F,
                    Vis_Km = weatherSnapshotDomain.Current.Vis_Km,
                    Vis_Miles = weatherSnapshotDomain.Current.Vis_Miles,
                    Uv = weatherSnapshotDomain.Current.Uv,
                    Gust_Mph = weatherSnapshotDomain.Current.Gust_Mph,
                    Gust_Kph = weatherSnapshotDomain.Current.Gust_Kph,
                    Air_Quality = new GetWeatherDataAirQuality
                    {
                        Co = weatherSnapshotDomain.Current.Air_Quality.Co,
                        No2 = weatherSnapshotDomain.Current.Air_Quality.No2,
                        O3 = weatherSnapshotDomain.Current.Air_Quality.O3,
                        So2 = weatherSnapshotDomain.Current.Air_Quality.So2,
                        Pm2_5 = weatherSnapshotDomain.Current.Air_Quality.Pm2_5,
                        Pm10 = weatherSnapshotDomain.Current.Air_Quality.Pm10,
                        UsEpaIndex = weatherSnapshotDomain.Current.Air_Quality.UsEpaIndex,
                        GbDefraIndex = weatherSnapshotDomain.Current.Air_Quality.GbDefraIndex
                    }
                }
            };
        }

        public static GetWeatherFilteredDataDto ToFilteredDto(this WeatherSnapshotDomain
weatherSnapshotDomain)
        {
            return new GetWeatherFilteredDataDto
            {
                City = weatherSnapshotDomain.City,
                CreatedOn = weatherSnapshotDomain.From,
                Location = new GetWeatherFilteredDataLocation
                {
                    Name = weatherSnapshotDomain.Location.Name,
                    Region = weatherSnapshotDomain.Location.Region,
                    Country = weatherSnapshotDomain.Location.Country,
                    Lat = weatherSnapshotDomain.Location.Lat,
                    Lon = weatherSnapshotDomain.Location.Lon,
                    Tz_Id = weatherSnapshotDomain.Location.Tz_Id,
                    Localtime_Epoch = weatherSnapshotDomain.Location.Localtime_Epoch,
                    Localtime = weatherSnapshotDomain.Location.Localtime
                },
                Current = new GetCurrentFilteredWeather
                {
                    Last_Updated_Epoch = weatherSnapshotDomain.Current.Last_Updated_Epoch,
                    Last_Updated = weatherSnapshotDomain.Current.Last_Updated,
                    Temp_C = weatherSnapshotDomain.Current.Temp_C,
                    Temp_F = weatherSnapshotDomain.Current.Temp_F,
                    Is_Day = weatherSnapshotDomain.Current.Is_Day,
                    Condition = new GetWeatherFilteredDataCondition
                    {
                        Text = weatherSnapshotDomain.Current.Condition.Text,
                        Icon = weatherSnapshotDomain.Current.Condition.Icon,
                        Code = weatherSnapshotDomain.Current.Condition.Code
                    },
                    Wind_Mph = weatherSnapshotDomain.Current.Wind_Mph,
                    Wind_Kph = weatherSnapshotDomain.Current.Wind_Kph,
                    Wind_Degree = weatherSnapshotDomain.Current.Wind_Degree,
```

```
                Wind_Dir = weatherSnapshotDomain.Current.Wind_Dir,
                Pressure_Mb = weatherSnapshotDomain.Current.Pressure_Mb,
                Pressure_In = weatherSnapshotDomain.Current.Pressure_In,
                Precip_Mm = weatherSnapshotDomain.Current.Precip_Mm,
                Precip_In = weatherSnapshotDomain.Current.Precip_In,
                Humidity = weatherSnapshotDomain.Current.Humidity,
                Cloud = weatherSnapshotDomain.Current.Cloud,
                Feelslike_C = weatherSnapshotDomain.Current.Feelslike_C,
                Feelslike_F = weatherSnapshotDomain.Current.Feelslike_F,
                Windchill_C = weatherSnapshotDomain.Current.Windchill_C,
                Windchill_F = weatherSnapshotDomain.Current.Windchill_F,
                Heatindex_C = weatherSnapshotDomain.Current.Heatindex_C,
                Heatindex_F = weatherSnapshotDomain.Current.Heatindex_F,
                Dewpoint_C = weatherSnapshotDomain.Current.Dewpoint_C,
                Dewpoint_F = weatherSnapshotDomain.Current.Dewpoint_F,
                Vis_Km = weatherSnapshotDomain.Current.Vis_Km,
                Vis_Miles = weatherSnapshotDomain.Current.Vis_Miles,
                Uv = weatherSnapshotDomain.Current.Uv,
                Gust_Mph = weatherSnapshotDomain.Current.Gust_Mph,
                Gust_Kph = weatherSnapshotDomain.Current.Gust_Kph,
                Air_Quality = new GetWeatherFilteredDataAirQuality
                {
                    Co = weatherSnapshotDomain.Current.Air_Quality.Co,
                    No2 = weatherSnapshotDomain.Current.Air_Quality.No2,
                    O3 = weatherSnapshotDomain.Current.Air_Quality.O3,
                    So2 = weatherSnapshotDomain.Current.Air_Quality.So2,
                    Pm2_5 = weatherSnapshotDomain.Current.Air_Quality.Pm2_5,
                    Pm10 = weatherSnapshotDomain.Current.Air_Quality.Pm10,
                    UsEpaIndex = weatherSnapshotDomain.Current.Air_Quality.UsEpaIndex,
                    GbDefraIndex = weatherSnapshotDomain.Current.Air_Quality.GbDefraIndex
                }
            }
        };
    }
  }
}

using System;

namespace SyncAlignWeather.Identity.Domain
{
  public class WeatherSnapshotDomain
  {
    public WeatherSnapshotDomain(
      Location location,
      CurrentWeather current,
      string? city,
      DateTime? from)
    {
      Location = location;
      Current = current;
      City = city;
      From = from;
    }

    public Location Location { get; set; }
    public CurrentWeather Current { get; set; }
```

```
    public string? City { get; set; }
    public DateTime? From { get; set; }
}

public class Location
{
    public Location(
        string name,
        string region,
        string country,
        double lat,
        double lon,
        string tzId,
        long localtimeEpoch,
        string localtime)
    {
        Name = name;
        Region = region;
        Country = country;
        Lat = lat;
        Lon = lon;
        Tz_Id = tzId;
        Localtime_Epoch = localtimeEpoch;
        Localtime = localtime;
    }

    public string Name { get; set; }
    public string Region { get; set; }
    public string Country { get; set; }
    public double Lat { get; set; }
    public double Lon { get; set; }
    public string Tz_Id { get; set; }
    public long Localtime_Epoch { get; set; }
    public string Localtime { get; set; }
}

public class CurrentWeather
{
    public CurrentWeather(
        long lastUpdatedEpoch,
        string lastUpdated,
        double tempC,
        double tempF,
        int isDay,
        Condition condition,
        double windMph,
        double windKph,
        int windDegree,
        string windDir,
        double pressureMb,
        double pressureIn,
        double precipMm,
        double precipIn,
        int humidity,
        int cloud,
        double feelslikeC,
```

```
    double feelslikeF,
    double windchillC,
    double windchillF,
    double heatindexC,
    double heatindexF,
    double dewpointC,
    double dewpointF,
    double visKm,
    double visMiles,
    double uv,
    double gustMph,
    double gustKph,
    AirQuality airQuality)
{
    Last_Updated_Epoch = lastUpdatedEpoch;
    Last_Updated = lastUpdated;
    Temp_C = tempC;
    Temp_F = tempF;
    Is_Day = isDay;
    Condition = condition;
    Wind_Mph = windMph;
    Wind_Kph = windKph;
    Wind_Degree = windDegree;
    Wind_Dir = windDir;
    Pressure_Mb = pressureMb;
    Pressure_In = pressureIn;
    Precip_Mm = precipMm;
    Precip_In = precipIn;
    Humidity = humidity;
    Cloud = cloud;
    Feelslike_C = feelslikeC;
    Feelslike_F = feelslikeF;
    Windchill_C = windchillC;
    Windchill_F = windchillF;
    Heatindex_C = heatindexC;
    Heatindex_F = heatindexF;
    Dewpoint_C = dewpointC;
    Dewpoint_F = dewpointF;
    Vis_Km = visKm;
    Vis_Miles = visMiles;
    Uv = uv;
    Gust_Mph = gustMph;
    Gust_Kph = gustKph;
    Air_Quality = airQuality;
}

public long Last_Updated_Epoch { get; set; }
public string Last_Updated { get; set; }
public double Temp_C { get; set; }
public double Temp_F { get; set; }
public int Is_Day { get; set; }
public Condition Condition { get; set; }
public double Wind_Mph { get; set; }
public double Wind_Kph { get; set; }
public int Wind_Degree { get; set; }
public string Wind_Dir { get; set; }
```

```
    public double Pressure_Mb { get; set; }
    public double Pressure_In { get; set; }
    public double Precip_Mm { get; set; }
    public double Precip_In { get; set; }
    public int Humidity { get; set; }
    public int Cloud { get; set; }
    public double Feelslike_C { get; set; }
    public double Feelslike_F { get; set; }
    public double Windchill_C { get; set; }
    public double Windchill_F { get; set; }
    public double Heatindex_C { get; set; }
    public double Heatindex_F { get; set; }
    public double Dewpoint_C { get; set; }
    public double Dewpoint_F { get; set; }
    public double Vis_Km { get; set; }
    public double Vis_Miles { get; set; }
    public double Uv { get; set; }
    public double Gust_Mph { get; set; }
    public double Gust_Kph { get; set; }
    public AirQuality Air_Quality { get; set; }
}

public class Condition
{
    public Condition(string text, string icon, int code)
    {
        Text = text;
        Icon = icon;
        Code = code;
    }

    public string Text { get; set; }
    public string Icon { get; set; }
    public int Code { get; set; }
}

public class AirQuality
{
    public AirQuality(
        double co,
        double no2,
        double o3,
        double so2,
        double pm2_5,
        double pm10,
        int usEpaIndex,
        int gbDefraIndex)
    {
        Co = co;
        No2 = no2;
        O3 = o3;
        So2 = so2;
        Pm2_5 = pm2_5;
        Pm10 = pm10;
        UsEpaIndex = usEpaIndex;
        GbDefraIndex = gbDefraIndex;
```

```
            Recalculate();
        }

        public double Co { get; set; }
        public double No2 { get; set; }
        public double O3 { get; set; }
        public double So2 { get; set; }
        public double Pm2_5 { get; set; }
        public double Pm10 { get; set; }
        public int UsEpaIndex { get; set; }
        public int GbDefraIndex { get; set; }

        private void Recalculate()
        {
            Co = Math.Round(Co / 1000.0, 1);
            No2 = Math.Round(No2, 0);
            O3 = Math.Round(O3, 0);
            So2 = Math.Round(So2, 0);
            Pm2_5 = Math.Round(Pm2_5, 0);
            Pm10 = Math.Round(Pm10, 0);
        }
    }
}

namespace SyncAlignWeather.Identity.Dto
{
    public class GetWeatherDataDto
    {
        public GetWeatherDataLocation Location { get; set; }
        public GetCurrentWeather Current { get; set; }
    }

    public class GetWeatherDataLocation
    {
        public string Name { get; set; }
        public string Region { get; set; }
        public string Country { get; set; }
        public double Lat { get; set; }
        public double Lon { get; set; }
        public string Tz_Id { get; set; }
        public long Localtime_Epoch { get; set; }
        public string Localtime { get; set; }
    }

    public class GetCurrentWeather
    {
        public long Last_Updated_Epoch { get; set; }
        public string Last_Updated { get; set; }
        public double Temp_C { get; set; }
        public double Temp_F { get; set; }
        public int Is_Day { get; set; }
        public double Wind_Mph { get; set; }
        public double Wind_Kph { get; set; }
        public int Wind_Degree { get; set; }
        public string Wind_Dir { get; set; }
        public double Pressure_Mb { get; set; }
```

```
        public double Pressure_In { get; set; }
        public double Precip_Mm { get; set; }
        public double Precip_In { get; set; }
        public int Humidity { get; set; }
        public int Cloud { get; set; }
        public double Feelslike_C { get; set; }
        public double Feelslike_F { get; set; }
        public double Windchill_C { get; set; }
        public double Windchill_F { get; set; }
        public double Heatindex_C { get; set; }
        public double Heatindex_F { get; set; }
        public double Dewpoint_C { get; set; }
        public double Dewpoint_F { get; set; }
        public double Vis_Km { get; set; }
        public double Vis_Miles { get; set; }
        public double Uv { get; set; }
        public double Gust_Mph { get; set; }
        public double Gust_Kph { get; set; }
        public GetWeatherDataAirQuality Air_Quality { get; set; }
    }

    public class GetWeatherDataAirQuality
    {
        public double Co { get; set; }
        public double No2 { get; set; }
        public double O3 { get; set; }
        public double So2 { get; set; }
        public double Pm2_5 { get; set; }
        public double Pm10 { get; set; }
        public int UsEpaIndex { get; set; }
        public int GbDefraIndex { get; set; }
    }
}

namespace SyncAlignWeather.Identity.Dto
{
    public class GetWeatherFilteredDataDto
    {
        public GetWeatherFilteredDataLocation Location { get; set; }
        public GetCurrentFilteredWeather Current { get; set; }
        public string? City { get; set; }
        public DateTime? CreatedOn { get; set; }
    }

    public class GetWeatherFilteredDataLocation
    {
        public string Name { get; set; }
        public string Region { get; set; }
        public string Country { get; set; }
        public double Lat { get; set; }
        public double Lon { get; set; }
        public string Tz_Id { get; set; }
        public long Localtime_Epoch { get; set; }
        public string Localtime { get; set; }
    }

    public class GetCurrentFilteredWeather
```

```
    {
        public long Last_Updated_Epoch { get; set; }
        public string Last_Updated { get; set; }
        public double Temp_C { get; set; }
        public double Temp_F { get; set; }
        public int Is_Day { get; set; }
        public GetWeatherFilteredDataCondition Condition { get; set; }
        public double Wind_Mph { get; set; }
        public double Wind_Kph { get; set; }
        public int Wind_Degree { get; set; }
        public string Wind_Dir { get; set; }
        public double Pressure_Mb { get; set; }
        public double Pressure_In { get; set; }
        public double Precip_Mm { get; set; }
        public double Precip_In { get; set; }
        public int Humidity { get; set; }
        public int Cloud { get; set; }
        public double Feelslike_C { get; set; }
        public double Feelslike_F { get; set; }
        public double Windchill_C { get; set; }
        public double Windchill_F { get; set; }
        public double Heatindex_C { get; set; }
        public double Heatindex_F { get; set; }
        public double Dewpoint_C { get; set; }
        public double Dewpoint_F { get; set; }
        public double Vis_Km { get; set; }
        public double Vis_Miles { get; set; }
        public double Uv { get; set; }
        public double Gust_Mph { get; set; }
        public double Gust_Kph { get; set; }
        public GetWeatherFilteredDataAirQuality Air_Quality { get; set; }
    }

    public class GetWeatherFilteredDataCondition
    {
        public string Text { get; set; }
        public string Icon { get; set; }
        public int Code { get; set; }
    }

    public class GetWeatherFilteredDataAirQuality
    {
        public double Co { get; set; }
        public double No2 { get; set; }
        public double O3 { get; set; }
        public double So2 { get; set; }
        public double Pm2_5 { get; set; }
        public double Pm10 { get; set; }
        public int UsEpaIndex { get; set; }
        public int GbDefraIndex { get; set; }
    }
}

using System.Text.Json.Serialization;

namespace SyncAlignWeather.Identity.Dto
```

```
{
    public class WeatherDataDto
    {
        [JsonPropertyName("location")]
        public Location Location { get; set; }

        [JsonPropertyName("current")]
        public CurrentWeather Current { get; set; }
    }

    public class Location
    {
        [JsonPropertyName("name")]
        public string Name { get; set; }

        [JsonPropertyName("region")]
        public string Region { get; set; }

        [JsonPropertyName("country")]
        public string Country { get; set; }

        [JsonPropertyName("lat")]
        public double Lat { get; set; }

        [JsonPropertyName("lon")]
        public double Lon { get; set; }

        [JsonPropertyName("tz_id")]
        public string Tz_Id { get; set; }

        [JsonPropertyName("localtime_epoch")]
        public long Localtime_Epoch { get; set; }

        [JsonPropertyName("localtime")]
        public string Localtime { get; set; }
    }

    public class CurrentWeather
    {
        [JsonPropertyName("last_updated_epoch")]
        public long Last_Updated_Epoch { get; set; }

        [JsonPropertyName("last_updated")]
        public string Last_Updated { get; set; }

        [JsonPropertyName("temp_c")]
        public double Temp_C { get; set; }

        [JsonPropertyName("temp_f")]
        public double Temp_F { get; set; }

        [JsonPropertyName("is_day")]
        public int Is_Day { get; set; }

        [JsonPropertyName("condition")]
        public Condition Condition { get; set; }
```

```
[JsonPropertyName("wind_mph")]
public double Wind_Mph { get; set; }

[JsonPropertyName("wind_kph")]
public double Wind_Kph { get; set; }

[JsonPropertyName("wind_degree")]
public int Wind_Degree { get; set; }

[JsonPropertyName("wind_dir")]
public string Wind_Dir { get; set; }

[JsonPropertyName("pressure_mb")]
public double Pressure_Mb { get; set; }

[JsonPropertyName("pressure_in")]
public double Pressure_In { get; set; }

[JsonPropertyName("precip_mm")]
public double Precip_Mm { get; set; }

[JsonPropertyName("precip_in")]
public double Precip_In { get; set; }

[JsonPropertyName("humidity")]
public int Humidity { get; set; }

[JsonPropertyName("cloud")]
public int Cloud { get; set; }

[JsonPropertyName("feelslike_c")]
public double Feelslike_C { get; set; }

[JsonPropertyName("feelslike_f")]
public double Feelslike_F { get; set; }

[JsonPropertyName("windchill_c")]
public double Windchill_C { get; set; }

[JsonPropertyName("windchill_f")]
public double Windchill_F { get; set; }

[JsonPropertyName("heatindex_c")]
public double Heatindex_C { get; set; }

[JsonPropertyName("heatindex_f")]
public double Heatindex_F { get; set; }

[JsonPropertyName("dewpoint_c")]
public double Dewpoint_C { get; set; }

[JsonPropertyName("dewpoint_f")]
public double Dewpoint_F { get; set; }

[JsonPropertyName("vis_km")]
```

```
    public double Vis_Km { get; set; }

    [JsonPropertyName("vis_miles")]
    public double Vis_Miles { get; set; }

    [JsonPropertyName("uv")]
    public double Uv { get; set; }

    [JsonPropertyName("gust_mph")]
    public double Gust_Mph { get; set; }

    [JsonPropertyName("gust_kph")]
    public double Gust_Kph { get; set; }

    [JsonPropertyName("air_quality")]
    public AirQuality Air_Quality { get; set; }
}

public class Condition
{
    [JsonPropertyName("text")]
    public string Text { get; set; }

    [JsonPropertyName("icon")]
    public string Icon { get; set; }

    [JsonPropertyName("code")]
    public int Code { get; set; }
}

public class AirQuality
{
    [JsonPropertyName("co")]
    public double Co { get; set; }

    [JsonPropertyName("no2")]
    public double No2 { get; set; }

    [JsonPropertyName("o3")]
    public double O3 { get; set; }

    [JsonPropertyName("so2")]
    public double So2 { get; set; }

    [JsonPropertyName("pm2_5")]
    public double Pm2_5 { get; set; }

    [JsonPropertyName("pm10")]
    public double Pm10 { get; set; }

    [JsonPropertyName("us-epa-index")]
    public int UsEpaIndex { get; set; }

    [JsonPropertyName("gb-defra-index")]
    public int GbDefraIndex { get; set; }
}
```

```
}
using MongoDB;
using MongoDB.Bson;
using MongoDB.DAL;
using MongoDB.Driver;
using SyncAlignWeather.Identity.Repository.Interfaces;

namespace SyncAlignWeather.Identity.Repository
{
   public class WeatherRepository : IWeatherRepository
   {
      private readonly IMongoCollection<WeatherSnapshot> _collection;

      public WeatherRepository(MongoDBStore store)
      {
         _collection = store.WeatherSnapshot;
      }

      public async Task AddAsync(WeatherSnapshot snapshot)
      {
         await _collection.InsertOneAsync(snapshot);
      }

      public async Task<List<WeatherSnapshot>> GetFilteredAsync(string? city, DateTime? from,
DateTime? to)
      {
         var query = _collection.AsQueryable();

         if (from.HasValue)
            query = query.Where(x => x.CreatedOn >= from.Value.Date);

         if (to.HasValue)
         {
            var endOfDay = to.Value.Date.AddDays(1).AddTicks(-1);
            query = query.Where(x => x.CreatedOn <= endOfDay);
         }

         var results = await Task.Run(() => query.ToList());

         if (!string.IsNullOrEmpty(city))
         {
            results = results
      .Where(x                    =>                    x.Weather.Location.Name.Contains(city,
StringComparison.OrdinalIgnoreCase))
                  .ToList();
         }

         return results;
      }
   }
}
```

Код програми (Фронт)

```
import type React from "react"

import { useState, useEffect, useRef } from "react"
import { Search } from "lucide-react"
import { Input } from "@/components/ui/input"
import { useRouter } from "next/navigation"


const CITIES = [
  "Kyiv",
  "Kharkiv",
  "Odesa",
  "Dnipro",
  "Donetsk",
  "Zaporizhzhia",
  "Lviv",
  "Kryvyi Rih",
  "Mykolaiv",
  "Mariupol",
  "Luhansk",
  "Vinnytsia",
  "Makiivka",
  "Sevastopol",
  "Simferopol",
  "Kherson",
  "Poltava",
  "Chernihiv",
  "Cherkasy",
  "Zhytomyr",
  "Sumy",
  "Khmelnytskyi",
  "Chernivtsi",
  "Rivne",
  "Kropyvnytskyi",
  "Ivano-Frankivsk",
  "Ternopil",
  "Lutsk",
  "Uzhhorod",
  "Kamianets-Podilskyi",
]

export default function CitySearch()
{
   const [query, setQuery] = useState("")
  const [suggestions, setSuggestions] = useState<string[]>([])
  const [isOpen, setIsOpen] = useState(false)
  const inputRef = useRef<HTMLInputElement>(null)
  const suggestionsRef = useRef<HTMLDivElement>(null)
  const router = useRouter()

  useEffect(() => {
```

```
            if (query.length > 0)
            {
              const filteredCities = CITIES.filter((city) => city.toLowerCase().includes(query.toLowerCase())).slice(0, 5) // Limit to 5 suggestions

            setSuggestions(filteredCities)
             setIsOpen(filteredCities.length > 0)
            }
            else
            {
               setSuggestions([])
            setIsOpen(false)
           }
          }, [query])

          useEffect(() => {

          const handleClickOutside = (event: MouseEvent) => {
            if (
             suggestionsRef.current &&
             !suggestionsRef.current.contains(event.target as Node) &&
             inputRef.current &&
             !inputRef.current.contains(event.target as Node)
            ) {
             setIsOpen(false)
            }
           }

           document.addEventListener("mousedown", handleClickOutside)
           return () => {
              document.removeEventListener("mousedown", handleClickOutside)
           }
          }, [])

          const handleSelectCity = (city: string) => {
           setQuery(city)
           setIsOpen(false)

           router.push(`/city/${encodeURIComponent(city)}`)
          }

          const handleSubmit = (e: React.FormEvent) => {
           e.preventDefault()
           if (query.trim()) {
            handleSelectCity(query)
           }
          }

          return (
           < div className = "relative w-full" >
            < form onSubmit ={ handleSubmit}>
             < Search className = "absolute left-3 top-1/2 -translate-y-1/2 text-gray-400 z-10" size ={ 20} />
             < Input
              ref={ inputRef}
        type = "text"
```

```
            value ={ query}
          onChange ={ (e) => setQuery(e.target.value)}
          placeholder = "Впишіть назву міста..."
            className = "w-full bg-transparent border-gray-700 rounded-full pl-10 py-6 focus:border-primary focus:ring-primary"
            onFocus ={ () => query.length > 0 && suggestions.length > 0 && setIsOpen(true)}
          />
        </ form >

        {
        isOpen && suggestions.length > 0 && (
          < div
          ref= { suggestionsRef }
          className = "absolute mt-2 w-full bg-[#242527] border border-gray-700 rounded-lg shadow-lg z-20 overflow-hidden"
          >
          {
          suggestions.map((city, index) => (
            < div
            key ={ index}
          className = "px-4 py-3 hover:bg-gray-700 cursor-pointer text-left transition-colors"
            onClick ={ () => handleSelectCity(city)}
          >
            { city}
          </ div >
          ))}
          </ div >
        )}
        </ div >
      )
    }


    import { useEffect, useState } from "react"
    import axios from "axios"
    import Image from "next/image"
    import Link from "next/link"
    import { Card, CardContent } from "@/components/ui/card"
    import { useRouter } from "next/navigation"

    interface CityPageProps
    {
      params: {
        name: string
    }
    }

    interface AirQuality
    {
      co: string
      no2: string
      o3: string
      so2: string
      pm2_5: string
      pm10: string
    }
```

```
export default function CityPage({ params }: CityPageProps) {
  const cityName = decodeURIComponent(params.name)
  const router = useRouter()

  const [airQualityData, setAirQualityData] = useState < AirQuality | null > (null)
  const [hasError, setHasError] = useState(false)
  const [actualCityName, setActualCityName] = useState<string>("")

  useEffect(() => {
  const fetchAirQuality = async () => {
  try
  {
     const response = await axios.post(`https://localhost:7259/api/weather/${cityName}`)
      const aq = response.data.current.air_Quality

      setAirQualityData({
     co: `${ aq.co}
        мг / м³`,
        no2: `${ aq.no2}
        мкг / м³`,
        o3: `${ aq.o3}
        мкг / м³`,
        so2: `${ aq.so2}
        мкг / м³`,
        pm2_5: `${ aq.pm2_5}
           мкг / м³`,
        pm10: `${ aq.pm10}
        мкг / м³`,
       })
      setActualCityName(response.data.location.name)
     } catch (error) {
    console.error("Error getting data", error)
      setHasError(true)
     }
    }

    fetchAirQuality()
  }, [cityName])


  if (hasError)
{
    return (
      < main className = "text-white flex flex-col items-center justify-center h-screen" >
        < p className = "text-lg text-red-500 mb-4" > Помилка при завантаженні даних </ p >
        < button
          onClick ={ () => router.back()}
    className = "px-4 py-2 bg-primary text-white rounded hover:opacity-80 transition"
   >
    Назад
   </ button >

</ main >
    )
   }
```

```
if (!airQualityData)
{
   return (
     < main className = "text-white flex items-center justify-center h-screen" >
       < p className = "text-lg text-gray-400" > Завантаження...</ p >
     </ main >
   )
  }

return (

  < div className = "relative min-h-screen bg-[#18191b] text-white overflow-hidden pb-20" >

    < div className = "absolute -bottom-32 -left-32 w-96 h-96 rounded-full bg-gradient-to-br from-primary to-primary/60 blur-xl opacity-30" ></ div >

    < div className = "absolute -top-20 -right-20 w-80 h-80 rounded-full bg-gradient-to-br from-primary to-primary/60 blur-xl opacity-30" ></ div >


    < header className = "container mx-auto pt-6 px-4" >

      < div className = "flex justify-between items-center" >

        < Link href = "/" className = "cursor-pointer hover:opacity-80 transition-opacity" >

          < Image src = "/logo.svg" alt = "SyncAlign Logo" width ={ 48}
height ={ 43} />

        </ Link >

        < div className = "flex items-center" >

          < span className = "text-primary text-xl font-bold" > SyncAlign </ span >

          < span className = "text-primary text-xl font-bold mx-1" >.</ span >

          < span className = "text-transparent bg-clip-text bg-gradient-to-r from-primary to-primary/80 text-xl font-serif italic font-bold" >
            weather
          </ span >

        </ div >

      </ div >

    </ header >


    < main className = "container mx-auto flex flex-col items-center px-4 py-12" >

      < h1 className = "text-3xl md:text-4xl font-bold mb-8 text-center" >
        Результат по місту <span className="text-primary">{ actualCityName }</ span >

      </ h1 >
```

```
          < div className = "grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6 w-full max-w-4xl" >
            {
          Object.entries(airQualityData).map(([key, value]) => (
              < Card key ={ key}
          className = "bg-[#242527] border-gray-700 hover:border-primary transition-colors" >
                < CardContent className = "p-6 flex flex-col items-center" >
                  < h3 className = "text-lg font-medium mb-3 text-primary" >{ key.toUpperCase()}</ h3
>
                  < p className = "text-2xl font-bold text-white" >{ value}</ p >
                  < p className = "text-sm text-gray-400 mt-1" >
                    { getPollutantDescription(key)}
                  </ p >
                </  CardContent >
              </ Card >
            ))}
          </ div >
        </ main >

        < footer className = "container mx-auto px-4 py-4 absolute bottom-0 left-0 right-0" >
          < div className = "flex justify-between items-center" >
            < div className = "text-gray-400 text-sm" >©2025 SyncAlign </ div >
          </ div >
        </ footer >
      </ div >
    )
  }

  function getPollutantDescription(pollutant: string): string {
    switch (pollutant) {
      case "co": return "Монооксид вуглецю"
      case "no2":
      return "Діоксид азоту"
      case "o3":
      return "Озон"
      case "so2":
      return "Діоксид сірки"
      case "pm2_5":
      return "Дрібні частинки"
      case "pm10":
      return "Великі частинки"
      default:
      return ""
    }
  }

  import Image from "next/image"
  import Link from "next/link"
  import { Input } from "@/components/ui/input"
  import { Button } from "@/components/ui/button"
  import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
  import { Search, Calendar } from "lucide-react"
  import { useRef } from "react"
  import axios from "axios"

  export default function ReportPage() {
```

```
  const cityRef = useRef<HTMLInputElement>(null)
  const fromRef = useRef<HTMLInputElement>(null)
  const toRef = useRef<HTMLInputElement>(null)

  const handleReportDownload = async () => {
    const city = cityRef.current?.value
    const from = fromRef.current?.value
    const to = toRef.current?.value

    const query = new URLSearchParams()
    if (city) query.append("city", city)
    if (from) query.append("from", from)
    if (to) query.append("to", to)

    const url = `https://localhost:7259/api/weather/history/export?${query.toString()}`

    try
{
    const response = await axios.get(url, { responseType: "blob" })
      const downloadUrl = window.URL.createObjectURL(new Blob([response.data]))
      const link = document.createElement("a")
      link.href = downloadUrl
      link.download = `weather - history -${ new Date().toISOString().slice(0, 10)}.xlsx`
      document.body.appendChild(link)
      link.click()
      link.remove()
    }
  catch (error)
  {
    console.error("Download failed", error)
    }
   }

    return (
      < div className = "relative min-h-screen bg-[#18191b] text-white overflow-hidden" >
        {/* Decorative circles */}
        < div className = "absolute -bottom-32 -left-32 w-96 h-96 rounded-full bg-gradient-to-br from-primary to-primary/60 blur-xl opacity-30" ></ div >
        < div className = "absolute -top-20 -right-20 w-80 h-80 rounded-full bg-gradient-to-br from-primary to-primary/60 blur-xl opacity-30" ></ div >

        {/* Header with logo */}
        < header className = "container mx-auto pt-6 px-4" >
          < div className = "flex justify-between items-center" >
            < Link href = "/" className = "cursor-pointer hover:opacity-80 transition-opacity" >
              < Image src = "/logo.svg" alt = "SyncAlign Logo" width ={ 48}
  height ={ 43}
  className = "text-primary" />
            </ Link >
            < div className = "flex items-center" >
              < span className = "text-primary text-xl font-bold" > SyncAlign </ span >
              < span className = "text-primary text-xl font-bold mx-1" >.</ span >
              < span className = "text-transparent bg-clip-text bg-gradient-to-r from-primary to-primary/80 text-xl font-serif italic font-bold" >
                weather
              </ span >
```

```
            </ div >
          </ div >
        </ header >

        {/* Main content */}
        < main className = "container mx-auto flex flex-col items-center justify-center px-4 py-12" >
          < h1 className = "text-3xl md:text-4xl font-bold mb-8" > Створити звіт </ h1 >

          < Card className = "w-full max-w-md bg-[#242527] border-gray-700" >
            < CardHeader >
              < CardTitle className = "text-white text-center" > Фільтри </ CardTitle >
            </ CardHeader >
            < CardContent className = "space-y-6" >
              {/* City name filter */}
              < div className = "space-y-2" >
                < label htmlFor = "city" className = "text-sm text-gray-300 block" >
      Назва міста
                </ label >
                < div className = "relative" >
                  < Search className = "absolute left-3 top-1/2 -translate-y-1/2 text-gray-400" size ={ 18}
/>
                  < Input
                    id = "city"
                    type = "text"
                    ref={ cityRef}
      placeholder = "Введіть назву міста"
                    className = "w-full bg-[#18191b] border-gray-700 rounded-lg pl-10 py-5
focus:border-primary focus:ring-primary text-white"
                  />
                </ div >
              </ div >

              {/* Time from filter */}
              < div className = "space-y-2" >
                < label htmlFor = "timeFrom" className = "text-sm text-gray-300 block" >
      Час від
                </ label >
                < div className = "relative" >
                  < Calendar className = "absolute left-3 top-1/2 -translate-y-1/2 text-white" size ={ 18}
/>
                  < Input
                    id = "timeFrom"
                    type = "date"
                    ref={ fromRef}
      maxLength ={ 10}
      className = "w-full bg-[#18191b] border-gray-700 rounded-lg pl-10 py-5 focus:border-primary
focus:ring-primary text-white"
      />

      </ div >

      </ div >

              {/* Time to filter */}
              < div className = "space-y-2" >
                < label htmlFor = "timeTo" className = "text-sm text-gray-300 block" >
```

```
        Час до
                </ label >
                < div className = "relative" >
                  < Calendar className = "absolute left-3 top-1/2 -translate-y-1/2 text-gray-400" size ={
18} />
                  < Input
                    id = "timeTo"
                    type = "date"
                    ref={ toRef}
        className = "w-full bg-[#18191b] border-gray-700 rounded-lg pl-10 py-5 focus:border-primary
focus:ring-primary text-white"
        />

        </ div >

        </ div >
                {/* Submit button */}
                < Button onClick ={ handleReportDownload}
        className = "w-full bg-primary hover:bg-primary/80 text-white py-6 mt-4" >
                  Згенерувати звіт
                </ Button >
              </ CardContent >
            </ Card >
          </ main >

          {/* Footer */}
          < footer className = "container mx-auto px-4 py-4 absolute bottom-0 left-0 right-0" >
            < div className = "flex justify-between items-center" >
              < div className = "text-gray-400 text-sm" >©2025 SyncAlign </ div >
              < div className = "flex space-x-4" ></ div >
            </ div >
          </ footer >
        </ div >
      )
    }
```