НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
_____ 202_ р.	**_____ 202_ р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Громадська система моніторингу атмосферного повітря»

Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітня програма «Інформаційні управляючі системи і технології»
(назва)

Орієнтація освітньої програми _____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

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

Керівник магістерської кваліфікаційної роботи

_____ _____ Голуб Б.Л.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Юзвик А.О.
(підпис) (ПІБ студента)

КИЇВ-2022

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____Інформаційних технологій_____

ЗАТВЕРДЖУЮ

Завідувач кафедри__Комп'ютерних наук__

_____ _____ _____

доцент, к.т.н Голуб Б.Л.

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

"____"_____20____року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____Юзвик Андрій Олександрович_____

(прізвище, ім'я, по батькові)

Спеціальність ___122 «Комп'ютерні науки»_____

(код і назва)

Освітня програма _____«Інформаційні управляючі системи і технології»_____

(назва)

Орієнтація освітньої програми_____

(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи __Громадська система моніторингу атмосферного повітря_____

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи __виміри екологічних параметрів, які зробили спеціалізовані станції_____

_____

Перелік питань, що підлягають дослідженню:

1. _Системний аналіз_____
2. _Проектування інформаційної частини_____
3. _Проектування системи_____
4. _Впровадження_____

Перелік графічного матеріалу (за потреби) _____

_____

Дата видачі завдання "_____"_____2022_ р.

Керівник магістерської кваліфікаційної роботи _____ Голуб Б.Л.

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____

ЗМІСТ

ВСТУП

Актуальність. Ключову роль в якості життя людини, відіграє атмосферне повітря. Адже саме воно здатне чинити позитивний чи негативний вплив на стан її здоров'я чи навіть моральний стан.

Забруднення повітря важко уникнути незалежно від того, наскільки багатий район, де ви живете. Мікроскопічні забруднювачі повітря можуть проходити крізь захисні механізми нашого організму, проникаючи глибоко в дихальну та кровоносну систему та руйнуючи легені, серце та мозок.

Відсутність видимого пилу не означає, що довкола чисте повітря. У всьому світі, і міста, і села зазнають токсичної дії забруднювачів, рівні яких перевищують середньорічні значення, що рекомендуються в ВООЗ щодо якості повітря.

В міру того як навколишній світ стає все більш забрудненим і перенаселеним, двигуни продовжують викидати в атмосферу забруднюючі речовини і половина всього населення не має доступу до чистих видів палива або технологій (таких як печі та лампи), рівні забруднення повітря, яким ми дихаємо, стають дедалі небезпечнішими — нині 9 з 10 людей дихають забрудненим повітрям, що щорічно призводить до 7 мільйонів випадків смертей.

Об'єкт дослідження. Об'єктом дослідження виступає стан атмосферного повітря, саме він на сьогодні, є головним оскільки при його дослідженні ми маємо змогу знати про рівні забрудненості, в тих чи інших регіонах нашої країни, що дозволить завчасно попереджувати населення про критичні чи задовільні рівні забруднення.

Мета дослідження – автоматизування збору вимірюваних параметрів та забезпечення відображення цих даних у громадській системі моніторингу атмосферного повітря, а також інформування громарськості що до стану повітря та прогнозування, шляхом створення систе

Поставленні завдання. Побудова громадської системи моніторингу атмосферного повітря. Аналіз отриманих екоголічних параметрів, формування звітів а також можливість прогнозування та попередження громадськості що до змін.

Методи дослідження. Основними методами дослідження були: емпіричний, відбувався збір екологічних даних що до стану атмосферного повітря у певних регіонах м.Київ, також відслідковування динаміки зміни цих параметрів та їхня залежність один від одного, і побудова громадської системи моніторингу зі сховищем даних, а також теоретичний, аналіз та опис отриманих показників. Для проведення аналізу даних та підтвердження гіпотез, розрахунку ключових показників ефективності використовується технологія OLAP.

Наукова новизна. Вперше система дає можливість прогнозування та дослідження стану атмосферного повітря що може бути використане провідними екологами чи дослідницькими центрами у яких є необхідність такого дослідження в тих чи інших регіонах а також самими громадянами для завчасного реагування ну будь які зміни.

Зміст записки. Пояснювальна записка складається з 4 основних розділів. У першому розділі розглянуто системний аналіз предметної області, описаний процес виміру, сформоване завдання, проаналізовано існуючі рішення. Другий розділ описує проектування інформаційної частини, відображаються шляхи проектування. У третьому розділі розглядається проектування системи, база даних, сховище даних та Power BI, і також описується алгоритмізація та програмування. Четвертий розділ впровадження, тестування системи, вимоги до апаратного і програмного забезпечення, а також вимоги до вуз

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Забруднення. Забруднення повітря нині вважається найбільшою загрозою здоров'ю через довкілля у світі, щорічно призводить до 7 мільйонів випадків смерті у всьому світі. Забруднення повітря викликає і посилює низку захворювань - від астми до раку, хвороб легень та хвороб серця.

Забруднення повітря стосується викиду шкідливих речовин у повітря — шкідливих речовин, які завдають негативного впливу здоров’ю людини та планеті в цілому. Наразі дев’ять із десяти людей дихають повітрям, у якому кількість забруднюючих речовин перевищує нормативні ліміти ВООЗ, причому найбільше страждають жителі країн із низьким і середнім рівнем доходу[1].

Викиди транспортних засобів, мазут і природний газ для обігріву будинків, побічні продукти виробництва та виробництва електроенергії, зокрема електростанцій, що працюють на вугіллі, і випари від хімічного виробництва є основними джерелами антропогенного забруднення повітря.

Природа викидає в повітря небезпечні речовини, такі як дим від лісових пожеж, які часто викликані людьми; попіл і гази від вивержень вулканів; і гази, такі як метан, які виділяються в результаті розкладання органічних речовин у грунтах.

Трафікове забруднення повітря, суміш газів і частинок, містить більшість елементів антропогенного забруднення повітря: приземний озон, різні форми вуглецю, оксиди азоту, оксиди сірки, летючі органічні сполуки, поліциклічні ароматичні вуглеводні та дрібні тверді частинки.

Озон, атмосферний газ, на рівні землі часто називають смогом. Він утворюється, коли забруднюючі речовини, що викидаються автомобілями, електростанціями, промисловими котлами, нафтопереробними заводами та іншими джерелами, вступають у хімічну реакцію під дією сонячного світла.

Шкідливі гази, які включають вуглекислий газ, монооксид вуглецю, оксиди азоту (NOx) і оксиди сірки (SOx), є компонентами викидів автомобілів і побічними продуктами промислових процесів.

Забруднення повітря складається з хімічних речовин або частинок у повітрі, які можуть завдати шкоди здоров'ю людей, тварин і рослин. Це також завдає шкоди будівлям. Забруднювачі в повітрі мають різні форми. Це можуть бути гази, тверді частинки або краплі рідини.

Джерела забруднення повітря.

Забруднення потрапляє в атмосферу Землі різними шляхами. Більшість забруднювачів повітря створюють люди у вигляді викидів заводів, автомобілів, літаків або аерозольних балонів. Пасивний сигаретний дим також вважається забрудненням повітря. Ці техногенні джерела забруднення називають антропогенними[2].

Деякі типи забруднення повітря, такі як дим від лісових пожеж або попіл від вулканів, є природними. Це так звані природні джерела.

Забруднення повітря найбільш поширене у великих містах, де зосереджені викиди з багатьох різних джерел. Іноді гори або високі будівлі перешкоджають поширенню забруднення повітря. Це забруднення повітря часто виглядає як хмара, яка робить повітря каламутним. Його називають смогом. Слово «смог» походить від поєднання слів «дим» і «туман».

Великі міста бідних країн і країн, що розвиваються, як правило, мають більше забруднення повітря, ніж міста розвинених країн. За даними Всесвітньої організації охорони здоров'я (ВООЗ), одними з найбільш забруднених міст світу є Карачі, Пакистан; Нью-Делі, Індія; Пекін, Китай; Ліма, Перу; і Каїр, Єгипет. Проте багато розвинених країн також мають

проблеми із забрудненням повітря. Лос-Анджелес, штат Каліфорнія, називають Містом Смогу.

Забруднення повітря в приміщенні.

Забруднення повітря зазвичай сприймають як дим великих заводів або вихлопні гази транспортних засобів. Але також існує багато видів забруднення повітря в приміщеннях.

Опалення будинку спалюванням речовин, таких як гас, дрова та вугілля, може забруднити повітря всередині будинку. Попіл і дим ускладнюють дихання, вони можуть прилипати до стін, їжі та одягу.

Природний радон, речовина, що викликає рак, також може накопичуватися в будинках. Радон виділяється поверхнею Землі. Недорогі системи, встановлені професіоналами, можуть знизити рівень радону[3]

Деякі будівельні матеріали, зокрема утеплювачі, також небезпечні для здоров'я людей. Крім того, вентиляція або рух повітря в будинках і кімнатах може призвести до поширення токсичної цвілі. Одна колонія цвілі може існувати у вологому прохолодному місці будинку, наприклад між стінами. Спори цвілі проникають у повітря і поширюються по всьому будинку. Люди можуть захворіти від вдихання спор.

Вплив на людину.

Люди відчувають широкий спектр наслідків для здоров'я від впливу забрудненого повітря. Ефекти можна розбити на короткострокові та довгострокові.

Короткострокові наслідки, які є тимчасовими, включають такі захворювання, як пневмонія або бронхіт. Вони також включають такий дискомфорт, як подразнення носа, горла, очей або шкіри. Забруднене повітря також може викликати головний біль, запаморочення та нудоту. Неприємні запахи, які створюють заводи, сміття чи каналізаційні системи, також вважаються забрудненням повітря. Ці запахи менш серйозні, але все одно неприємні.

Довгостроковий вплив забруднення повітря може тривати роками або все життя. Вони можуть призвести навіть до смерті людини. Довгострокові наслідки забруднення повітря для здоров'я включають хвороби серця, рак легенів і респіраторні захворювання, такі як емфізема. Забруднення повітря також може завдати довгострокової шкоди нервам, мозку, ниркам, печінці та іншим органам людей. Деякі вчені підозрюють, що забруднювачі повітря спричиняють вроджені дефекти. Майже 2,5 мільйона людей щороку помирають у світі від наслідків забруднення повітря на вулиці чи всередині приміщень.

Вплив на навколишнє середовище.

Подібно до людей, тварин і рослин, цілі екосистеми можуть страждати від забруднення повітря. Серпанок, як і смог, є видимим типом забруднення повітря, який затемнює форми та кольори. Туманне забруднення повітря може навіть приглушувати звуки.

Частинки забруднення повітря з часом падають назад на Землю. Забруднення повітря може безпосередньо забруднювати поверхню водойм і грунт. Це може призвести до загибелі культур або зниження їх урожайності. Це може вбити молоді дерева та інші рослини[4].

Частинки діоксиду сірки та оксиду азоту в повітрі можуть створювати кислотні дощі, коли вони змішуються з водою та киснем в атмосфері. Ці забруднювачі повітря надходять здебільшого від вугільних електростанцій та автотранспорту. Коли кислотні дощі випадають на Землю, вони завдають шкоди рослинам, змінюючи склад грунту; погіршує якість води в річках, озерах і струмках; завдає шкоди посівам; і може призвести до руйнування будівель і пам'ятників.

Як і люди, тварини можуть страждати від забруднення повітря. Вроджені дефекти, хвороби та нижчі показники репродуктивної здатності пояснюються забрудненням повітря.

Глобальне потепління.

Глобальне потепління — екологічне явище, спричинене природним та антропогенним забрудненням повітря. Це стосується підвищення температури повітря та океану в усьому світі. Це підвищення температури принаймні частково спричинене збільшенням кількості парникових газів в атмосфері. Парникові гази затримують теплову енергію в атмосфері Землі. (Зазвичай більше тепла Землі виходить у космос.)

Вуглекислий газ є парниковим газом, який найбільше вплинув на глобальне потепління. Вуглекислий газ викидається в атмосферу при спалюванні викопного палива (вугілля, бензину та природного газу). Люди почали покладатися на викопне паливо для живлення автомобілів і літаків, опалення будинків і роботи заводів. Такі дії забруднюють повітря вуглекислим газом.

Інші парникові гази, що викидаються природними та штучними джерелами, також включають метан, закис азоту та фторовані гази. Метан є основним викидом від вугільних заводів і сільськогосподарських процесів. Закис азоту є звичайним викидом промислових заводів, сільського господарства та спалювання викопного палива в автомобілях. Фторовані гази, такі як гідрофторвуглеці, викидаються промисловістю. Фторовані гази часто використовуються замість газів, таких як хлорфторвуглеці (CFC). У багатьох місцях фреони оголошені поза законом, оскільки вони руйнують озоновий шар.

У всьому світі багато країн вжили заходів для зменшення або обмеження викидів парникових газів для боротьби з глобальним потеплінням. Кіотський протокол, вперше прийнятий у Кіото, Японія, у 1997 році, є угодою між 183 країнами про те, що вони працюватимуть над скороченням своїх викидів вуглекислого газу. Сполучені Штати не підписали цей договір.

Регулювання.

На додаток до міжнародного Кіотського протоколу більшість розвинених країн прийняли закони для регулювання викидів і зменшення

забруднення повітря. У Сполучених Штатах точаться дебати щодо системи обмеження викидів, яка називається обмеженням і торгівлею. Ця система буде обмежувати або встановлювати обмеження на кількість забруднення, дозволену компанії. Компанії, які перевищують свій ліміт, повинні будуть платити. Компанії, які забруднювали менше, ніж їх обмеження, могли торгувати або продавати свої залишки квот на забруднення іншим компаніям. Обмеження та торгівля по суті окуплять компанії за обмеження забруднення.

У 2006 році Всесвітня організація охорони здоров'я випустила нові рекомендації щодо якості повітря. Рекомендації ВООЗ жорсткіші, ніж існуючі рекомендації більшості окремих країн. Рекомендації ВООЗ спрямовані на скорочення смертності через забруднення повітря на 15 відсотків на рік[5].

Зменшення.

Кожен може вжити заходів для зменшення забруднення повітря. Для цього мільйони людей щодня вносять прості зміни у своє життя. Їздити громадським транспортом замість того, щоб керувати автомобілем, або їздити на велосипеді замість того, щоб подорожувати транспортними засобами, які викидають вуглекислий газ, — це кілька способів зменшити забруднення повітря. Уникати аерозольних балонів, переробляти обрізки двору замість того, щоб їх спалювати, і не палити сигарети – це інші.

Екологічні параметри. PM-частинки. PM розшифровується як тверді частинки (також звані частинками забруднення): термін для суміші твердих частинок і крапель рідини, що знаходяться в повітрі. Деякі частинки, такі як пил, бруд, кіптява або дим, досить великі або темні, щоб їх можна було побачити неозброєним оком. Інші настільки малі, що їх можна виявити лише за допомогою електронного мікроскопа.

PM10: частинки, які вдихаються, діаметром 10 мікрометрів і менше; і

PM2,5: дрібні частинки, які можна вдихати, діаметр яких зазвичай становить 2,5 мікрометра і менше.

Деякі з них викидаються безпосередньо з джерел, таких як будівельні майданчики, ґрунтові дороги, поля, димові труби або пожежі.

Більшість частинок утворюється в атмосфері в результаті складних реакцій хімічних речовин, таких як діоксид сірки та оксиди азоту, які є забруднювачами, що викидаються електростанціями, промисловими підприємствами та автомобілями.

Тверді частинки містять мікроскопічні тверді частинки або краплі рідини, які настільки малі, що їх можна вдихнути та спричинити серйозні проблеми зі здоров'ям. Деякі частинки діаметром менше 10 мікрометрів можуть проникнути глибоко в легені, а деякі навіть можуть потрапити в кров. З них частинки діаметром менше 2,5 мікрометрів, також відомі як дрібні частки або PM2,5, становлять найбільший ризик для здоров'я.

Тверді частинки в повітрі (PM) не є окремим забруднювачем, а є сумішшю багатьох хімічних речовин. Це складна суміш твердих речовин і аерозолів, що складається з дрібних крапель рідини, сухих твердих фрагментів і твердих ядер з рідким покриттям. Частинки дуже різноманітні за розміром, формою та хімічним складом і можуть містити неорганічні іони, металеві сполуки, елементарний вуглець, органічні сполуки та сполуки із земної кори. Частинки визначаються за їхнім діаметром для цілей регулювання якості повітря. Ті, що мають діаметр 10 мікрон або менше (PM10), вдихаються в легені та можуть спричинити негативний вплив на здоров'я. Дрібні тверді частинки визначаються як частинки діаметром 2,5 мікрона або менше (PM2,5). Таким чином, PM2,5 містить частину PM10.

Яка різниця між PM10 і PM2,5?

PM10 і PM2,5 часто походять з різних джерел викидів, а також мають різний хімічний склад. Викиди від згоряння бензину, нафти, дизельного палива чи деревини утворюють значну частину забруднювача PM2,5, що міститься у зовнішньому повітрі, а також значну частку PM10. PM10 також включає пил з будівельних майданчиків, звалищ і сільського господарства,

лісових пожеж і спалювання чагарників/відходів, промислових джерел, пил, що переноситься вітром з відкритих земель, пилок і фрагменти бактерій.

ТЧ можуть викидатися безпосередньо з джерел (первинні частинки) або утворюватися в атмосфері в результаті хімічних реакцій газів (вторинних частинок), таких як діоксид сірки (SO_2), оксиди азоту (NOX) і певні органічні сполуки. Ці органічні сполуки можуть виділятися як природними джерелами, такими як дерева та рослинність, так і штучними (антропогенними) джерелами, такими як промислові процеси та вихлопи автомобілів[6].

Які шкідливі наслідки можуть спричинити тверді частки?

Ряд несприятливих наслідків для здоров'я пов'язаний із впливом як PM2,5, так і PM10. Для PM2.5 короткочасне опромінення (тривалістю до 24 годин) було пов'язане з передчасною смертю, збільшенням госпіталізацій із захворюваннями серця або легенів, гострим і хронічним бронхітом, нападами астми, відвідуванням відділення невідкладної допомоги, респіраторними симптомами та обмеженнями дні діяльності. Повідомлялося про ці несприятливі наслідки для здоров'я в основному у немовлят, дітей і людей похилого віку з уже існуючими захворюваннями серця або легенів. Крім того, з усіх поширених забруднювачів повітря PM2,5 пов'язаний з найбільшою часткою несприятливих наслідків для здоров'я, пов'язаних із забрудненням повітря, як у Сполучених Штатах, так і в усьому світі згідно з проектом Всесвітньої організації охорони здоров'я «Глобальний тягар хвороб».

Короткочасний вплив PM10 був пов'язаний головним чином із загостренням респіраторних захворювань, включаючи астму та хронічне обструктивне захворювання легень (ХОЗЛ), що призводило до госпіталізації та відвідування відділень невідкладної допомоги.

Довготривалий (від місяців до років) вплив PM2,5 пов'язують із передчасною смертю, особливо у людей із хронічними захворюваннями

серця чи легенів, а також уповільненням розвитку функції легенів у дітей. Наслідки тривалого впливу PM10 менш очевидні, хоча кілька досліджень показують зв'язок між довгостроковим впливом PM10 і респіраторною смертністю. Міжнародне агентство з дослідження раку (IARC) у 2015 році опублікувало огляд, в якому зроблено висновок, що тверді частки в забрудненому повітрі на вулиці спричиняють рак легенів.

Дослідження вказують на те, що люди похилого віку з хронічними захворюваннями серця або легенів, діти та астматики є групами, які найімовірніше зазнають негативних наслідків для здоров'я під час впливу PM10 та PM2,5. Крім того, діти та немовлята чутливі до шкоди від вдихання забруднюючих речовин, таких як PM, оскільки вони вдихають більше повітря на фунт ваги тіла, ніж дорослі - вони дихають швидше, проводять більше часу на вулиці та мають менші розміри тіла. Крім того, незріла імунна система дітей може бути більш сприйнятливою до ПМ, ніж здорові дорослі.

Дослідження в рамках дослідження дитячого здоров'я, ініційованого CARB, показало, що діти, які живуть у громадах з високим рівнем PM2,5, мали повільніший ріст легенів і менші легені у віці 18 років порівняно з дітьми, які жили в громадах з низьким рівнем PM2,5.

Алгоритм збирання даних. Пристрій для виміру екологічних параметрів, який комунікує із запрограмованою платою на ARDUINO, яка являється виступає стацією у громадській системі. Станція може включати в себе декілька датчиків, що збирають дані з певним інтервалом. Виміри відправляються на сервер, який в свою чергу записує ці дані до бази даних на Azure. Іншим шляхом отримання даних є дані з відкритого сервісу SaveEcoBot, звідки беруться параметри по більшості станцій які є в громадській системі. З сервера відбуваються запити до API SaveEcoBot, що в свою чергу повертає акнуальні(оснанні) виміри які виконуються що 2 хвилини. Після чого ці виміри зберігаються у БД яка розташована на Azure.

1.2 Постановка завдання

Розробка громадської системи моніторингу атмосферного повітря, яка дасть змогу аналізувати екологічні параметри, відображати рівні показників, а також давати такі можливості:

- можливість доступу до системи;
- перегляд усіх станцій;
- можливість візуального відображення зміни параметрів за періодами;
- відображати, мінімальні, максимальні, та середні показники за періодами.
- нормалізувати вхідні дані;
- зрівнювати виміри двох станцій, за параметрами;
- відображати усі екологічні станції на мапі.

Дати можливість перегляду звітів за певними періодами, такими як:

- одна година;
- вісім годин;
- один день;
- один тиждень;
- один місяць;

Станції повинні проводити основні виміри наступних екологічних параметрів:

- вологість повітря;
- температура;
- параметр PM2.5 – повітряний забруднювач, до складу якого входять тверді мікрочастинки розміром приблизно від 10 нанометрів до 2,5 мікрометрів;
- параметр PM10 – повітряний забруднювач, до складу якого входять тверді мікрочастинки розміром від 10 мікрометрів;

- Air Quality Index – індекс якості повітря, який розраховується на основі зазначених вище параметрів.

1.3 Аналіз наявних рішень

Один з популярних закордонних сервісів по моніторингу якості повітря це aqicn.org (рис. 1). Відображає як приватні станції так і публічні. Дає змогу переглядати усі наявні параметри станції в одній таблиці, що є зручно для користувача. Основна ціль проекту – надати публічний доступ до усіх підключених станцій по всій планеті, навіть в країнах Азії.

Плюси проекту:

- доступність;
- зручність в користуванні;
- відображення всіх параметрів в одній таблиці;
- можливість придбання власної станції.

Мінуси:

- відсутність відкритих даних для розробників;
- плата за підключення власної станції;
- відсутність онлайн оновлення даних.

Рис. 1 Веб-ресурс aqicn.org

Друге рішення – це відкритий веб-ресурс IQAir (рис. 2). Багато хто з нас не мають доступу до своєї тимчасової інформації про якість повітря, а часткове забруднення повітря залишається незазначеним.

IQAir працює над тим, щоб це змінити. Сьогодні керують найбільшою у світі безкоштовною платформою для отримання інформації про якість повітря в режимі реального часу та приваблюють постійно зростаюче число громадян, організацій і організацій світу.

Плюси:

- інтерактивний інтерфейс;
- топ країн з найбільшим атмосферним забрудненням;
- відображення небезпечних на безпечних рівнів.

Мінуси:

- відсутність можливості додавання власної станції;
- відсутність відкритих даних для розробників.

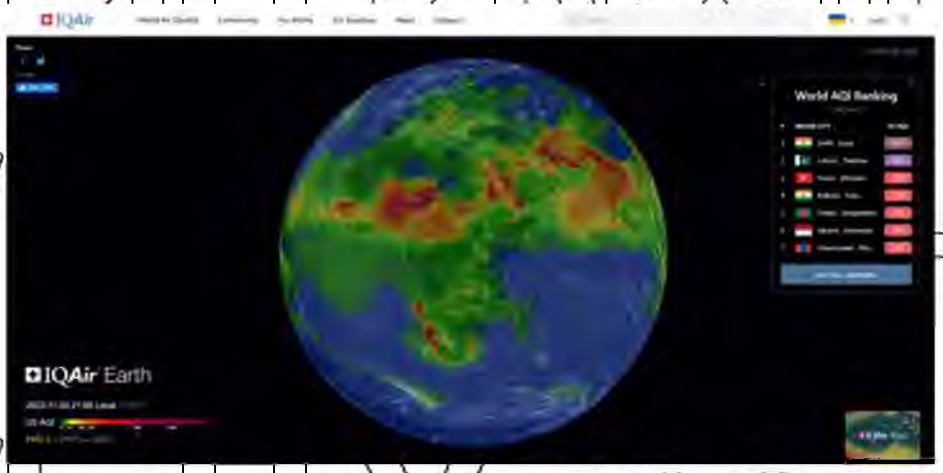

Рис. 2 Веб-ресурс IQAir

Ще одне існуюче рішення – веб-ресурс waqi.info (рис. 3). Проєкт з моніторингу атмосферного повітря, який відображає інформацію для жителів Сполучених Штатів Америки, наявні такі параметри як: PM2.5, PM10, OZONE.

Плюси:

- відображення основних параметрів;
- можливість перегляду інформації по всім штатам країни;
- кольорові індикатори рівнів екологічних параметрів.

Мінуси:

- відсутність інтерактивної карти;
- відсутність додавання власних станцій;
- відсутність відкритих даних для розробників.

Рис. 3 Веб-ресурс airnow.gov

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ЧАСТИНИ

2.1 Методи та засоби проектування

База даних — це організований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. База даних зазвичай контролюється системою керування базами даних (СУБД). Дані та СУБД разом із додатками, які з ними пов'язані, називають системою баз даних, яку часто скорочують до просто бази даних[7].

Дані в найпоширеніших типах баз даних, що працюють сьогодні, зазвичай моделюються в рядках і стовпцях у серії таблиць, щоб зробити обробку та запити даних ефективними. Потім можна легко отримати доступ до даних, керувати ними, змінювати, оновлювати, контролювати та організовувати. Більшість баз даних використовують структуровану мову запитів (SQL) для запису та запиту даних.

SQL — це мова програмування, яка використовується майже всіма реляційними базами даних для запитів, обробки та визначення даних, а також для забезпечення контролю доступу. SQL вперше було розроблено в IBM у 1970-х роках за участю Oracle як основного учасника, що призвело до впровадження стандарту SQL ANSI, SQL стимулював багато розширень від таких компаній, як IBM, Oracle і Microsoft. Хоча SQL все ще широко використовується сьогодні, починають з'являтися нові мови програмування[8].

Реалізація мови структурованих запитів (SQL) включає серверну машину, яка обробляє запити до бази даних і повертає результати. Процес SQL проходить через кілька компонентів програмного забезпечення, включаючи наступні.

Синтаксичний аналізатор починається з токенізації або заміни деяких слів у операторі SQL спеціальними символами. Потім він перевіряє оператор на наступне:

Коректність.

Синтаксичний аналізатор перевіряє, чи оператор SQL відповідає семантиці SQL або правилам, які забезпечують правильність оператора запиту. Наприклад, аналізатор перевіряє, чи команда SQL закінчується крапкою з комою. Якщо крапка з комою відсутня, аналізатор повертає помилку.

Авторизація.

Синтаксичний аналізатор також підтверджує, що користувач, який виконує запит, має необхідні повноваження для маніпулювання відповідними даними. Наприклад, лише адміністратори можуть мати право видаляти дані.

Реляційний механізм, або процесор запитів, створює план для отримання, запису або оновлення відповідних даних найбільш ефективним способом. Наприклад, він перевіряє наявність подібних запитів, повторно використовує попередні методи обробки даних або створює новий. Він записує план у представленні оператора SQL середнього рівня, що називається байт-кодом. Реляційні бази даних використовують байт-код для ефективного пошуку та модифікації бази даних.

Механізм зберігання, або механізм бази даних, — це програмний компонент, який обробляє байт-код і виконує запланований оператор SQL. Він читає та зберігає дані у файлах бази даних на фізичному дисковому сховищі. Після завершення система зберігання повертає результат запитуючій програмі.

Команди мови структурованих запитів (SQL) — це конкретні ключові слова або оператори SQL, які розробники використовують для маніпулювання даними, що зберігаються в реляційній базі даних. Ви можете класифікувати команди SQL наступним чином[9].

Мова визначення даних (DDL) відноситься до команд SQL, які проектують структуру бази даних. Інженери баз даних використовують DDL для створення та модифікації об'єктів бази даних відповідно до бізнес-вимог. Наприклад, розробник бази даних використовує команду CREATE для створення об'єктів бази даних, таких як таблиці, подання та індекси.

Мова запитів до даних (DQL) складається з інструкцій для отримання даних, що зберігаються в реляційних базах даних. Програмні програми використовують команду SELECT для фільтрації та повернення конкретних результатів із таблиці SQL.

Інструкції мови обробки даних (DML) записують нову інформацію або змінюють існуючі записи в реляційній базі даних. Наприклад, програма використовує команду INSERT для збереження нового запису в базі даних.

Адміністратори бази даних використовують мову керування даними (DCL), щоб керувати або дозволяти доступ до бази даних для інших користувачів. Наприклад, вони можуть використовувати команду GRANT, щоб дозволити певним програмам маніпулювати однією або кількома таблицями.

Реляційна система використовує мову керування транзакціями (TCL) для автоматичного внесення змін до бази даних. Наприклад, база даних використовує команду ROLLBACK для скасування помилкової транзакції.

Стандарти SQL — це набір формально визначених інструкцій щодо мови структурованих запитів (SQL). Американський національний інститут стандартів (ANSI) і Міжнародна організація стандартизації (ISO) прийняли стандарти SQL у 1986 році. Постачальники програмного забезпечення використовують стандарти ANSI SQL для створення програмного забезпечення баз даних SQL для розробників.

SQL-ін'єкція – це кібератака, яка передбачає обман бази даних за допомогою запитів SQL. Хакери використовують SQL-ін'єкції, щоб отримати, змінити або пошкодити дані в базі даних SQL. Наприклад, вони

можуть заповнити SQL-запит замість імені особи у формі подання, щоб здійснити атаку SQL-ін'єкції.

MySQL — це система керування реляційними базами даних з відкритим кодом, яку пропонує Oracle. Розробники можуть завантажувати та використовувати MySQL без сплати ліцензійної плати. Вони можуть інсталювати MySQL на різних операційних системах або хмарних серверах. MySQL — популярна система баз даних для веб-додатків.

Мова структурованих запитів (SQL) є стандартною мовою для створення бази даних і роботи з нею. MySQL — це програма реляційної бази даних, яка використовує запити SQL. У той час як команди SQL визначені міжнародними стандартами, програмне забезпечення MySQL зазнає постійних оновлень і вдосконалень.

NoSQL відноситься до нереляційних баз даних, які не використовують таблиці для зберігання даних. Розробники зберігають інформацію в різних типах баз даних NoSQL, включаючи графіки, документи та ключ-значення. Бази даних NoSQL популярні для сучасних програм, оскільки вони горизонтально масштабовані. Горизонтальне масштабування означає збільшення обчислювальної потужності шляхом додавання більшої кількості комп'ютерів, на яких працює програмне забезпечення NoSQL.

Структурована мова запитів (SQL) забезпечує єдину мову обробки даних, але реалізація NoSQL залежить від різних технологій. Розробники використовують SQL для транзакційних і аналітичних додатків, тоді як NoSQL підходить для адаптивних додатків, які активно використовуються.

SQL Server — це офіційна назва системи керування реляційною базою даних Microsoft, яка обробляє дані за допомогою SQL. MS SQL Server має кілька версій, кожна з яких розроблена для певних робочих навантажень і вимог.

Діаграма варіантів використання UML (рис. 4) є основною формою вимог до системи/програмного забезпечення для нової недостатньо

розробленої програми. Варіанти використання вказують на очікувану поведінку (що), а не на точний спосіб її здійснення (як). Випадки використання після визначення можуть позначатися як текстовим, так і візуальним представленням (тобто діаграмою варіантів використання). Ключова концепція моделювання варіантів використання полягає в тому, що воно допомагає нам проектувати систему з точки зору кінцевого користувача. Це ефективна техніка для передачі інформації про поведінку системи в термінах користувача шляхом визначення всієї видимої ззовні поведінки системи.

Діаграма варіантів використання зазвичай проста. Вона не показує деталі випадків використання:

Вона лише підсумовує деякі зв'язки між варіантами використання, акторами та системами.

Вона не показує порядок, у якому кроки виконуються для досягнення цілей кожного випадку використання.

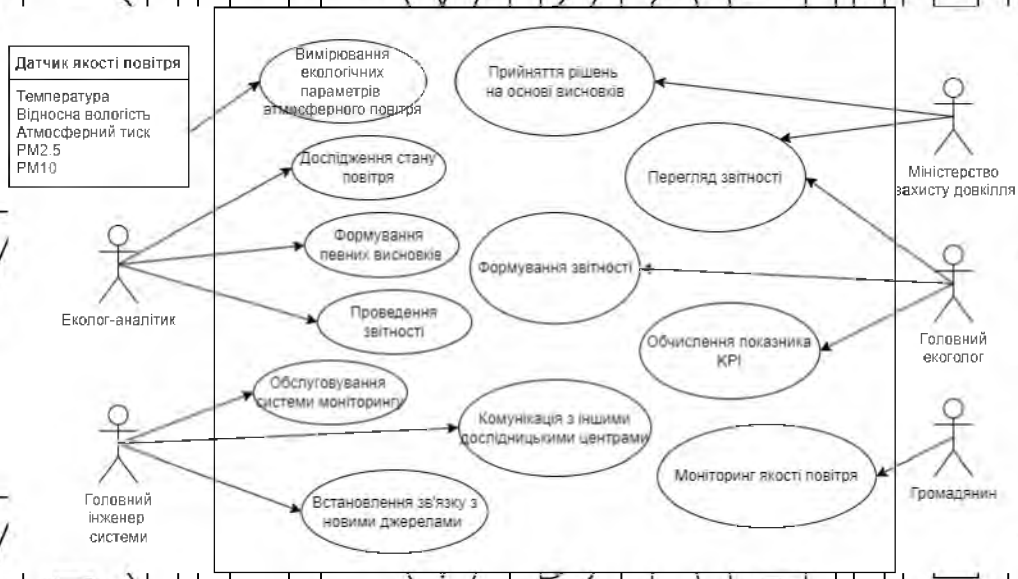

Рис. 4 – Діаграма прецидентів

На діаграмі можна побачити 5 основних акторів системи та їх відношення до прецидентів:

- Міністерство захисту довкілля (приймає рішення на основі висновків та переглядає звітність);

- Головний еколог (перегляд звітності, формування звітності, обчислення показника KPI);

- Громадянин (моніторинг якості повітря);

- Головний інженер системи (обслуговування системи моніторингу, комунікація з іншими дослідницькими центрами, встановлення зв'язку з новими джерелами);

- Еколог-аналітик (дослідження стану повітря, формування певних висновків, проведення звітності)

Для передачі даних з датчика до сервера використовується технологія MQTT.

MQTT (MQ Telemetry Transport) — це легкий відкритий протокол (рис. 5) обміну повідомленнями, який надає клієнтам мережі з обмеженими ресурсами простий спосіб розподілу телеметричної інформації в середовищах з низькою пропускною здатністю. Протокол, який використовує шаблон зв'язку публікації/підписки, використовується для зв'язку між машинами (M2M).

Створений як протокол з низькими накладними витратами, щоб врахувати обмеження пропускної здатності та ЦП, MQTT був розроблений для роботи у вбудованому середовищі, де він міг забезпечити надійний ефективний шлях для зв'язку. Підходить для підключення пристроїв з невеликим кодом, MQTT є гарним вибором для бездротових мереж, які мають різні рівні затримки через періодичні обмеження пропускної здатності або ненадійні з'єднання. Протокол має застосування в різних галузях промисловості, від автомобільної до енергетики та телекомунікацій.

Хоча MQTT починався як власний протокол, який використовувався для зв'язку з системами диспетчерського керування та збору даних (SCADA) у нафтовій і газовій промисловості, він став популярним на арені інтелектуальних пристроїв і сьогодні є провідним протоколом з відкритим кодом для підключення IoT і промислові пристрої IoT (IIoT)[10].

Хоча TT у MQTT означає телеметричний транспорт, MQ стосується продукту під назвою IBM MQ. Незважаючи на те, що MQTT іноді позначається як Message Queuing Telemetry Transport, у зв'язку MQTT немає черги повідомлень.

Націлена на максимізацію доступної пропускної здатності, комунікаційна модель MQTT публікація/підписка (pub/sub) є альтернативою традиційній архітектурі клієнт-сервер, яка спілкується безпосередньо з кінцевою точкою. Навпаки, у моделі pub/sub клієнт, який надсилає повідомлення (видавець), відокремлюється від клієнта або клієнтів, які отримують повідомлення (або підписників).

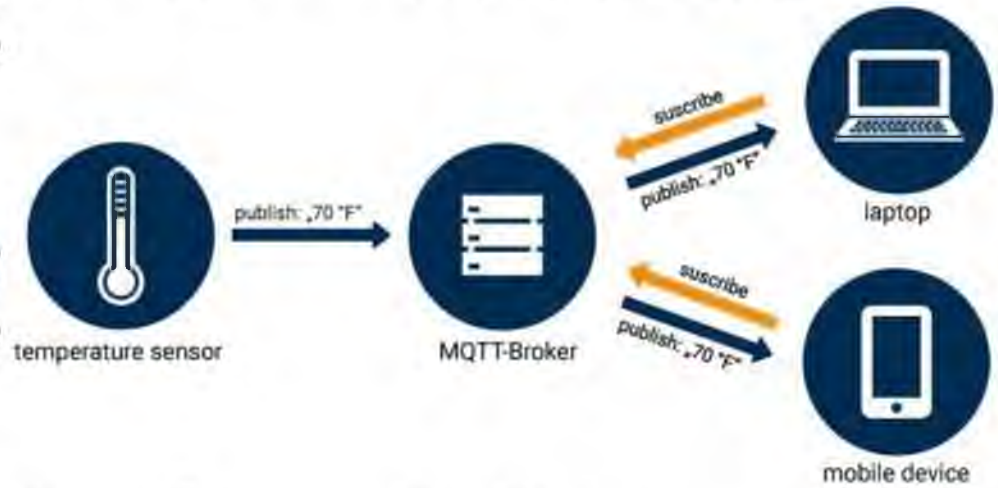

Рис. 5 Схема MQTT

Сховище даних. Сховище даних — це центральне сховище інформації, яку можна аналізувати для прийняття більш обґрунтованих рішень. Дані надходять у сховище даних із транзакційних систем, реляційних баз даних та інших джерел, як правило, регулярно. Бізнес-аналітики, інженери обробки даних, спеціалісти з обробки даних і особи, які приймають рішення, отримують доступ до даних за допомогою інструментів бізнес-аналітики (BI), клієнтів SQL та інших аналітичних програм.

Потреба в зберіганні даних виникла, коли підприємства почали покладатися на комп'ютерні системи для створення, зберігання та

отримання важливих бізнес-документів. Концепція сховищ даних була представлена в 1988 році дослідниками IBM Баррі Девліном і Полом Мерфі.

Сховища даних призначені для аналізу історичних даних. Порівняння консолідованих даних із кількох різнорідних джерел може дати розуміння ефективності компанії. Сховище даних розроблено таким чином, щоб дозволити користувачам виконувати запити й аналізувати історичні дані, отримані з транзакційних джерел.

Дані, додані до сховища, не змінюються та не можуть бути змінені. Сховище — це джерело, яке використовується для запуску аналітики минулих подій з акцентом на зміни з часом. Дані, що зберігаються, повинні зберігатися безпечно, надійно.

Ведення сховища даних.

Існують певні кроки, які виконуються для підтримки сховища даних. Одним із кроків є вилучення даних, яке передбачає збір великої кількості даних із кількох джерел. Після того, як набір даних скомпільовано, він проходить очищення даних, процес аналізу на наявність помилок і виправлення або виключення будь-яких знайдених.

Потім очищені дані перетворюються з формату бази даних у формат сховища. Після зберігання в сховищі дані проходять сортування, консолідацію та узагальнення, щоб ними було легше користуватися. З часом до сховища додається більше даних, оскільки різні джерела даних оновлюються.

Ключовою книгою зі сховищ даних є практичний посібник W. H. Inmon Building the Data Warehouse, який вперше був опублікований у 1990 році та кілька разів перевидавався.

Сьогодні компанії можуть інвестувати в послуги програмного забезпечення для хмарних сховищ даних від таких компаній, як Microsoft, Google, Amazon і Oracle тощо[11].

Data mining.

Компанії зберігають дані переважно для інтелектуального аналізу даних. Це передбачає пошук шаблонів інформації, які допоможуть їм покращити їхні бізнес-процеси.

Процес вилучення інформації для виявлення закономірностей, тенденцій і корисних даних, які дозволили б компанії прийняти кероване даними рішення з величезних наборів даних, називається інтелектуальним аналізом даних.

Іншими словами, ми можемо сказати, що інтелектуальний аналіз даних — це процес дослідження прихованих шаблонів інформації з різних точок зору для класифікації в корисні дані, які збираються та збираються в певних областях, таких як сховища даних, ефективний аналіз, алгоритм інтелектуального аналізу даних, допомога в прийнятті рішень створення та інші вимоги до даних, щоб зрештою скоротити витрати та отримати дохід.

Інтелектуальний аналіз даних — це автоматичний пошук великих сховищ інформації для виявлення тенденцій і закономірностей, які виходять за рамки простих процедур аналізу. Інтелектуальний аналіз даних використовує складні математичні алгоритми для сегментів даних і оцінює ймовірність майбутніх подій. Інтелектуальний аналіз даних також називається виявленням даних (KDD).

Інтелектуальний аналіз даних — це процес, який використовується організаціями для вилучення конкретних даних із величезних баз даних для вирішення бізнес-завдань. Це в першу чергу перетворює необроблені дані на корисну інформацію[12].

Інтелектуальний аналіз даних подібний до Data Science, який виконує особа в конкретній ситуації, на певному наборі даних із певною метою. Цей процес включає різні види послуг, такі як видобуток тексту, видобуток веб-сайтів, видобуток аудіо та відео, видобуток графічних даних і видобуток соціальних мереж. Це робиться за допомогою простого або дуже специфічного програмного забезпечення. Завдяки аутсорсингу видобутку даних всю роботу можна виконувати швидше з низькими експлуатаційними

витратами. Спеціалізовані фірми також можуть використовувати нові технології для збору даних, які неможливо знайти вручну. Є маса інформації, доступної на різних платформах, але дуже мало знань доступно. Найбільше завдання полягає в тому, щоб проаналізувати дані, щоб отримати важливу інформацію, яка може бути використана для вирішення проблеми або розвитку компанії. Є багато потужних інструментів і методів, доступних для аналізу даних і отримання кращого розуміння з них.

Хороша система сховища даних полегшує різним відділам компанії доступ до даних один одного. Наприклад, команда маркетингу може оцінити дані групи продажів, щоб прийняти рішення про те, як налаштувати свої кампанії продажів.

Дані та аналітика стали незамінними для компаній, щоб залишатися конкурентоспроможними. Бізнес-користувачі покладаються на звіти, інформаційні панелі та інструменти аналітики, щоб отримувати статистичні дані зі своїх даних, відстежувати ефективність бізнесу та підтримувати прийняття рішень. Сховища даних забезпечують ці звіти, інформаційні панелі та інструменти аналітики, ефективно зберігаючи дані, щоб мінімізувати введення та виведення даних і швидко надавати результати запитів сотням і тисячам користувачів одночасно.

Архітектура сховища даних складається з рівнів. Верхній рівень — це front-end клієнт, який представляє результати за допомогою інструментів звітування, аналізу та аналізу даних. Середній рівень складається з механізму аналітики, який використовується для доступу та аналізу даних. Нижній рівень архітектури — це сервер бази даних, де дані завантажуються та зберігаються. Дані зберігаються двома різними способами: 1) дані, до яких часто звертаються, зберігаються в дуже швидкому сховищі (як-от диски SSD) і 2) дані, до яких звертаються рідко, зберігаються в дешевому сховищі об'єктів, як-от Amazon S3. Сховище даних автоматично забезпечить переміщення даних, до яких часто звертаються, у «швидке» сховище, щоб оптимізувати швидкість запитів.

2.2 OLAP

Загальні поняття з напрямку OLAP-технології. Оперативні дані збираються з БД різних с/г об'єктів, очищаються, трансформуються і «складаються» в сховище даних.

Онлайн-аналітична обробка (OLAP) — це технологія, яка організовує великі бізнес-бази даних і підтримує комплексний аналіз. Його можна використовувати для виконання складних аналітичних запитів без негативного впливу на транзакційні системи[13].

Бази даних, які компанія використовує для зберігання всіх своїх транзакцій і записів, називаються базами даних онлайн-обробки транзакцій (OLTP). Ці бази даних зазвичай містять записи, які вводяться по одному. Часто вони містять велику кількість інформації, яка є цінною для організації. Однак бази даних, які використовуються для OLTP, не призначені для аналізу. Тому отримання відповідей із цих баз даних є дорогим з точки зору часу та зусиль. Системи OLAP були розроблені, щоб допомогти отримати цю бізнес-аналітичну інформацію з даних високопродуктивним способом. Це пояснюється тим, що бази даних OLAP оптимізовані для інтенсивного читання та низького навантаження на запис.

Сховище представляє дані в більш зрозумілій для аналізу структурі. Користувач (аналітик) отримує інтуїтивно зрозумілу модель даних, у вигляді багатовимірних кубів[14]. Куб є структурою даних, яка забезпечує можливість швидкого аналізу даних за рамками обмежень реляційних баз даних. Куби здатні відображати і підсумовувати великі обсяги даних, також надаючи користувачам доступ до будь-яких точках даних з можливістю пошуку. Таким чином, дані можуть бути зведені, фрагментовані і оброблені в міру необхідності для вирішення найбільш широкого спектра питань, що відносяться до сфери використання системи. Аналітик може отримувати зведені (наприклад, по роках) або, навпаки, детальні (по тижнях) відомості та здійснювати інші маніпуляції в процесі аналізу. Інструментом, який

забезпечує необхідні для аналізу маніпуляції над даними, є OLAP (Online Analytical Processing, оперативний аналіз даних).

Семантична модель даних — це концептуальна модель, яка описує значення елементів даних, які вона містить. Організації часто мають власні терміни для речей, іноді з синонімами або навіть різними значеннями для одного терміна. Наприклад, база даних інвентаризації може відстежувати одиницю обладнання за допомогою ідентифікатора активу та серійного номера, але база даних продажів може посилатися на серійний номер як ідентифікатор активу. Немає простого способу зв'язати ці значення без моделі, яка описує зв'язок.

Семантичне моделювання (рис. 6) забезпечує рівень абстракції над схемою бази даних, тому користувачам не потрібно знати основні структури даних. Це спрощує кінцевим користувачам запит даних без виконання агрегатів і об'єднань через базову схему. Крім того, зазвичай стовпці перейменовуються на більш зручні для користувача імена, щоб контекст і значення даних були більш очевидними.

Семантичне моделювання переважно використовується для сценаріїв з інтенсивним читанням, таких як аналітика та бізнес-аналітика (OLAP), на відміну від обробки транзакційних даних (OLTP), що потребує більшого запису. Здебільшого це пов'язано з характером типового семантичного шару:

Поведінки агрегації налаштовуються так, щоб інструменти звітування відображали їх належним чином.

Визначено бізнес-логіку та розрахунки.

Включаються розрахунки, орієнтовані на час.

Дані часто інтегруються з кількох джерел.

З цих причин традиційно семантичний рівень розміщується над сховищем даних.

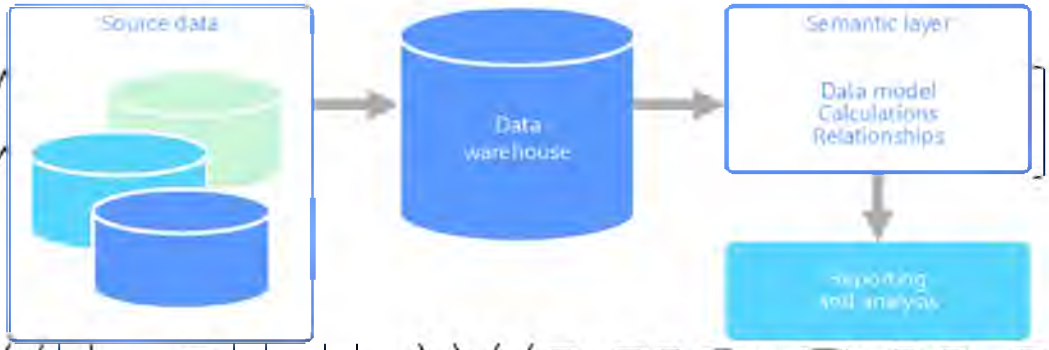

Рис. 6 Семантичний рівень

За допомогою технології OLAP користувач має можливість сформувати звіти, та зробити висновки по роботі відповідного с/г об'єкту, які в подальшому будуть представлені керівнику підприємства для допомоги в прийнятті управлінських рішень.

Вимір – це безліч об'єктів одного або декількох типів, організованих у вигляді ієрархічної структури і забезпечують інформаційний контекст числового показника (факту). Для збереження необхідних даних були розроблені такі таблиці вимірів:

Таблиці вимірів є батьківськими щодо таблиці фактів, тож первинні ключі таблиць вимірів є зовнішніми ключами таблиці фактів. Первинний ключ таблиці фактів є складеним і складається з усіх зовнішніх ключів. Тип схеми СД – сніжинка, оскільки наявна деталізована таблиця вимірів.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Архітектура системи

Технології розробки. Так як громадська система є веб орієнтованою системою, були обрані наступні технології для її розробки:

- Front-end частина: React, Redux, Recharts
- Back-end частина: Node.js, express.

React — це бібліотека, яка допомагає розробникам створювати користувацькі інтерфейси (UI) як дерево невеликих фрагментів, які називаються компонентами. Компонент — це суміш HTML і JavaScript, яка фіксує всю логіку, необхідну для відображення невеликої частини більшого інтерфейсу користувача. Кожен із цих компонентів можна вбудовувати в послідовно складні частини програми[15].

Будучи частиною мови JavaScript, використання React дає багато переваг. Продукти, створені за допомогою React, легко масштабувати, єдина мова, яка використовується на стороні сервера/клієнта/мобільного пристрою, забезпечує виняткову продуктивність, існують шаблони робочого процесу для зручної командної роботи, код інтерфейсу користувача читається та підтримується тощо. Провідні світові компанії використовували React та інші технології JS у деяких продуктах, такі як (Instagram, Reddit і Facebook – найяскравіші приклади).

Однією з основних причин використання React.js для веб-розробки є остаточно оптимізований інтерфейс розробки бібліотеки та мова кодування. Таким чином, спрощений API React підсилений швидкою продуктивністю для досягнення безпроблемного та швидкого робочого процесу розробки. Компоненти та концепції React дуже прості для розуміння, тому тут не потрібно багато часу на навчання.

На відміну від інших популярних фреймворків, таких як Vue та Angular, тут немає навали додаткових HTML-атрибутів (створених, коли JavaScript «впихають» у HTML — стандартна практика для традиційних фреймворків і бібліотечних рішень JS). У довгостроковій перспективі, помістивши JSX у JavaScript (буквально навпаки), React надає набагато чистіший, читабельніший і всеохоплюючий код.

Redux — це легкий інструмент керування станом, який допомагає компонентам у нашому React App спілкуватися один з одним[16]. Проста концепція цього полягає в тому, що кожен стан компонента зберігається в сховищі, яке буде глобальним. Щоб кожен компонент міг отримати доступ до будь-якого стану з цього сховища.

Redux став популярним завдяки своїй простоті та чудовій документації. Ще одна чудова річ у цьому полягає в тому, що розмір Redux становить лише 2 Кб. Тому нам не потрібно турбуватися про розмір нашої програми. Архітектура Redux зображена на рисунку 7.

Redux можна використовувати з будь-якою бібліотекою Javascript React, Angular тощо.

Переваги Redux:

- Стани доступу глобально: можна легко керувати станом програми, оскільки стан зроблено глобальним.
- Простий зв’язок: допомагає компонентам легко спілкуватися один з одним, не надсилаючи жодних реквізитів від одного компонента до іншого.
- Ремонтопридатність: Redux допомагає нам організувати нашу кодову базу. Щоб хтось із хорошими знаннями Redux міг легко зрозуміти структуру програми.
- Тестування та налагодження: Redux допомагає нам легко тестувати та налагоджувати код.

Додаток Redux складається з глобального сховища, яке зберігатиме стан усієї програми. Кожен із компонентів може отримати доступ до центрального сховища для доступу до стану.

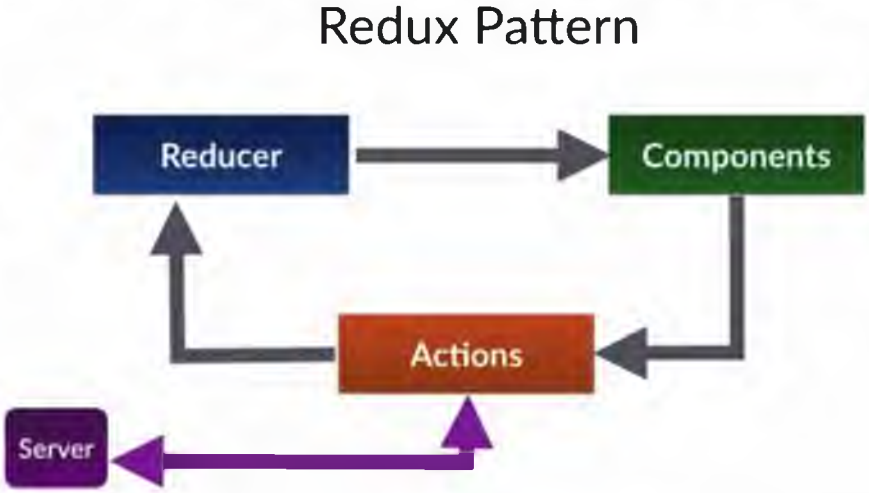

Рис. 7 Архітектура Redux

Store.

Store зберігає стан усієї програми. Для однієї програми може бути лише один store. Як розробник, ми можемо отримати доступ, оновити, зареєструвати або скасувати реєстрацію стану в store.

Actions.

Actions – це події. Це місце, де ми робимо всі наші запити до будь-яких викликів API або взаємодії з користувачем. Дії надсилаються за допомогою функції store.dispatch().

Reducer.

Reducer — це чисті функції, які беруть поточний стан програми, виконують дію та повертають новий стан. Ці стани зберігаються як об'єкти, і вони визначають, як змінюється стан програми у відповідь на дію, надіслану до сховища.

Recharts – бібліотека для побудови графіків.

На рисунку 8 зображене сховище даних Redux громадської системи моніторингу атмосферного повітря. На ньому можна побачити 3 основні об'єкти в яких зберігається така інформація:

auth – інформація що до авторизації користувача, його логін та стан авторизації;

stations – список станцій наявних у системі які приходять з back-end частини проекту;

compare – дані які необхідні для порівняння 2 станцій, а саме їх виміри, назви, спільні вимірювані параметри, а також id обраного екологічного параметра для порівняння.

Рис. 8 Redux дані

Node.js складається з двигуна Google V8 JavaScript, рівня абстракції платформи libUV і основної бібліотеки, написаної на JavaScript. Крім того, Node.js базується на відкритому веб-стеку (HTML, CSS і JS) і працює через стандартний порт 80[17].

Node.js надає розробникам комплексний інструмент для роботи в парадигмі введення-виведення. Райан Дал, творець Node.js, «надихався такими програмами, як Gmail», і, створюючи Node.js, мав на меті створювати веб-сайти в реальному часі з можливістю push.

Node.js — це кросплатформне середовище виконання з відкритим кодом для розробки серверних і мережевих програм. Програми Node.js

написані на JavaScript і можуть запускатися в середовищі виконання Node.js в OS X, Microsoft Windows і Linux.

Node.js також надає багату бібліотеку різноманітних модулів JavaScript, що значною мірою спрощує розробку веб-додатків за допомогою Node.js.

Нижче наведено деякі важливі функції, які роблять Node.js першим вибором архітекторів програмного забезпечення.

Асинхронний і керований подіями – усі API бібліотеки Node.js є асинхронними, тобто неблокуючими. По суті, це означає, що сервер на основі Node.js ніколи не чекає, поки API поверне дані. Сервер переходить до наступного API після його виклику, а механізм сповіщення про події Node.js допомагає серверу отримати відповідь від попереднього виклику API[18].

Дуже швидкий. Бібліотека Node.js створена на основі JavaScript Engine V8 від Google Chrome і дуже швидко виконує код.

Однопотоковий, але високомасштабований – Node.js використовує однопотокову модель із зацикленням подій. Механізм подій допомагає серверу відповідати неблокуючим способом і робить сервер високомасштабованим на відміну від традиційних серверів, які створюють обмежені потоки для обробки запитів. Node.js використовує одну потокову програму, і та сама програма може обслуговувати набагато більшу кількість запитів, ніж традиційні сервери, такі як HTTP-сервер Apache.

Відсутність буферизації − програми Node.js ніколи не буферизують дані. Ці програми просто виводять дані порціями.

Ліцензія – Node.js випущено згідно з ліцензією MIT.

Архітектура системи зображена на рисунку 9.

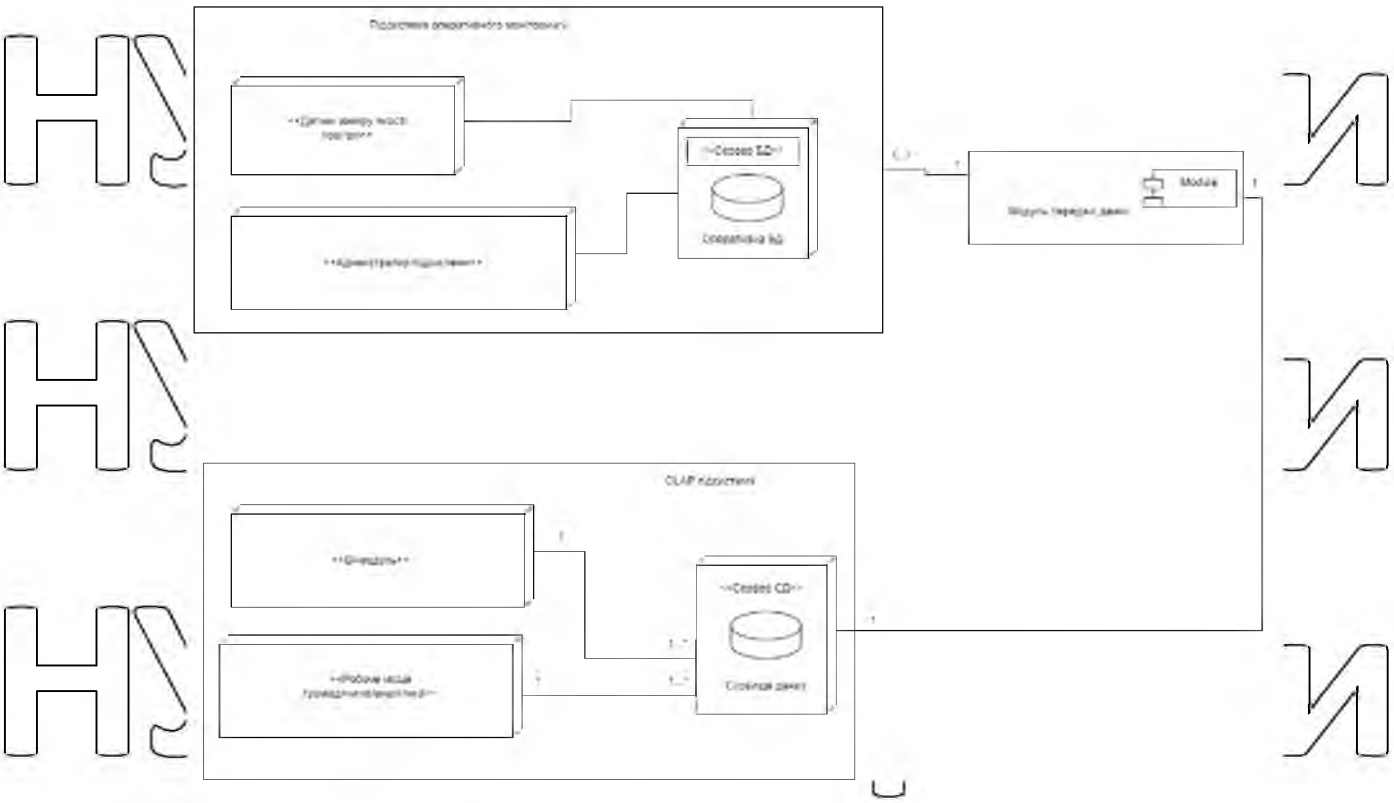

Рис. 9 Архітектура громадської системи моніторингу атмосферного повітря

На схемі відображені основні вузли які наявні в системі

3.2 Вузли системи

Підсистема оперативного моніторингу включає в себе такі вузли:

- Датчик виміру якості повітря. Станція моніторингу якості повітря SaveEcoSensor 3.0 дає можливість вимірювати вміст пилу фракцій 2.5 та 10 мікронів у повітрі (PM2.5 і PM10). Інтегрований сенсор температури-вологості-тиску дозволяє автоматично коригувати отриману інформацію у залежності від погодних умов, а модуль підігріву мінімізує вплив під час туману, опадів та при від'ємних температурах. Вбудований перетворювач живлення та апаратний сторожовий таймер (Watchdog) значно підвищують надійність пристрою.

Показники, які вимірює SaveEcoSensor;

1. Пил фракцій PM2.5 та PM10, мкг/м^3;
2. Температура, °C;
3. Відносна вологість, %;

4. Атмосферний тиск, гПа;

5. НЕСА – показники по камері підігріву: температура, °С та відносна вологість, %.

- Адміністратор підсистеми – людина яка є обізнаною в сфері програмування, в будь який момент зможе за необхідності додати новий функціонал, підправити старий або відповісти на технічні запитання користувачів системи. Може додавати видаляти станції, також вмикати та вимикати їх, і працювати зі звітами, налаштовувати/корегувати.

- Сервер БД – SQL сервер який знаходиться у хмарному сервісі Azure. Як СУБД обрана MS SQL Server.

Система керування базами даних (СУБД) — це програмний пакет, призначений для зберігання, отримання, запитів і керування даними. Інтерфейс користувача (UI) дозволяє авторизованим особам створювати, читати, оновлювати та видаляти дані[19].

Системи керування базами даних важливі, оскільки вони надають програмістам, адміністраторам баз даних і кінцевим користувачам централізований перегляд даних, а кінцевим користувачам не потрібно розуміти, де фізично розташовані дані. API (інтерфейси прикладних програм) обробляють запити та відповіді на певні типи даних через Інтернет.

Реляційні та нереляційні компоненти СУБД, які доставляються через Інтернет, у маркетингових матеріалах можуть називатися DBaaS (база даних як послуга). За даними дослідницької фірми Gartner, системи керування базами даних, призначені для підтримки розподілених даних у хмарі, наразі складають половину всього ринку СУБД.

Для вибору СУБД були сформовані вимоги до функціональних можливостей та швидкодії. Основні вимоги до СУБД:

- підтримка клієнт-серверної архітектури;
- дозвіл на перенесення існуючих інструментів обробки даних в хмарну платформу без серйозних змін;

- можливість розташувати у Microsoft Azure Cloud. Це забезпечить безперебійний доступ до БД та позбавить необхідності її оновлення;
- можливість аналізу даних;
- можливість управління транзакціями;
- реалізований пошук по фразах, тексту, словами.

Сама оперативна база даних зображена на рисунку 10.

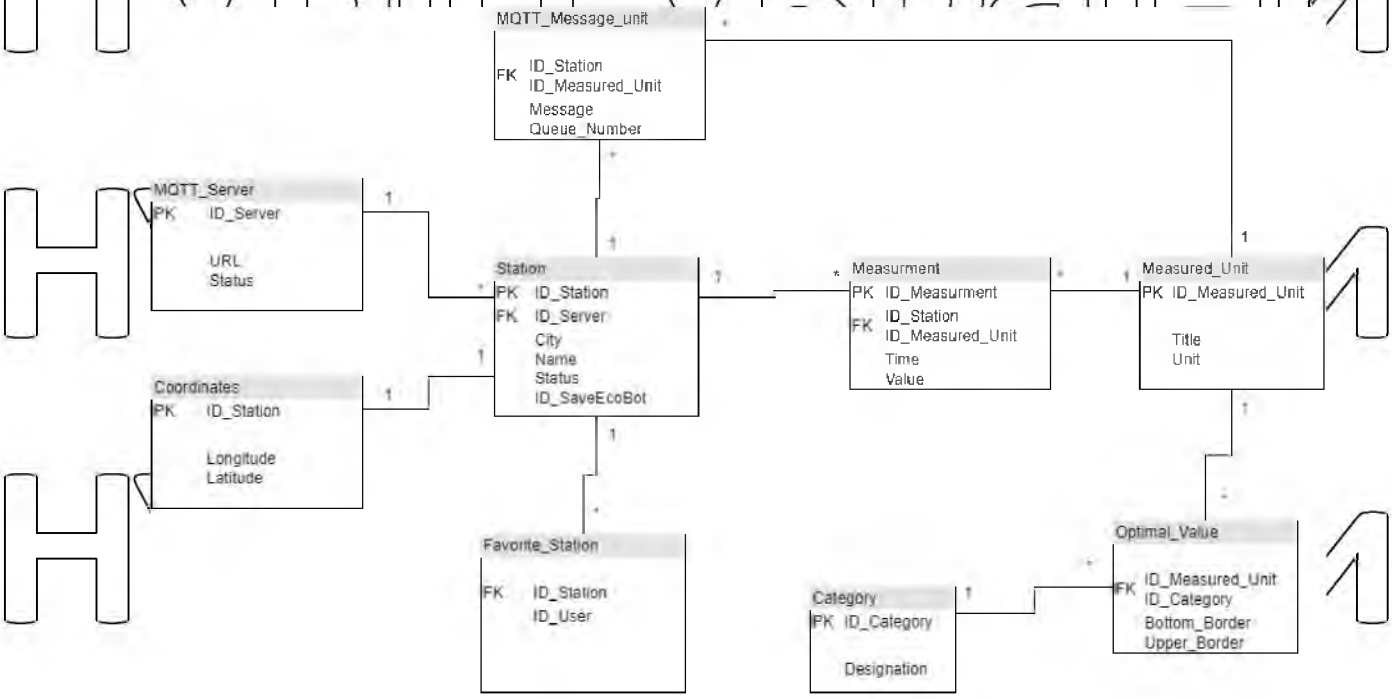

Рис. 10 Оперативна база даних громадської системи

MQTT_Message_Unit – таблиця в якій знаходяться повідомлення вимірів які надходять з MQTT серверу. Message – назва повідомлення. Queue_Number – порядковий номер за яким приходить повідомлення до сервера.

Station – таблиця в якій знаходяться основні данні по всіх станціях, які наявні в системі. City – країна, Name – назва/адреса станції, Status – статус запису даних.

MQTT_Server – підключені MQTT сервери, URL – адреса серверу, Status – статус запису даних.

Coordinates – координати станцій, які наявні в громадській системі. Необхідні для визначення точного місця дислокації станції, та відображення його на карті.

Favorite_Station – таблиця з обраними станціями.

Measurment – таблиця, до якої записуються виміри зроблені станціями. Основні поля time та value.

Measured_Unit – дані про екологічні параметри які вимірює станція, такі як: вологість, температура, PM10, PM2.5 і т.п. Title – назва параметру, Unit – одиниця виміру.

Optimal_Value – оптимальні значення для деяких параметрів. В таблиці наявні верхні та нижні межі для кожного екологічного показника.

Category – таблиця з назвами категорій до, яких відносяться оптимальні значення.

Збереження даних атмосферного повітря для подальшого їх аналізу було забезпечено СД, структура якого представлена на рис. 11.

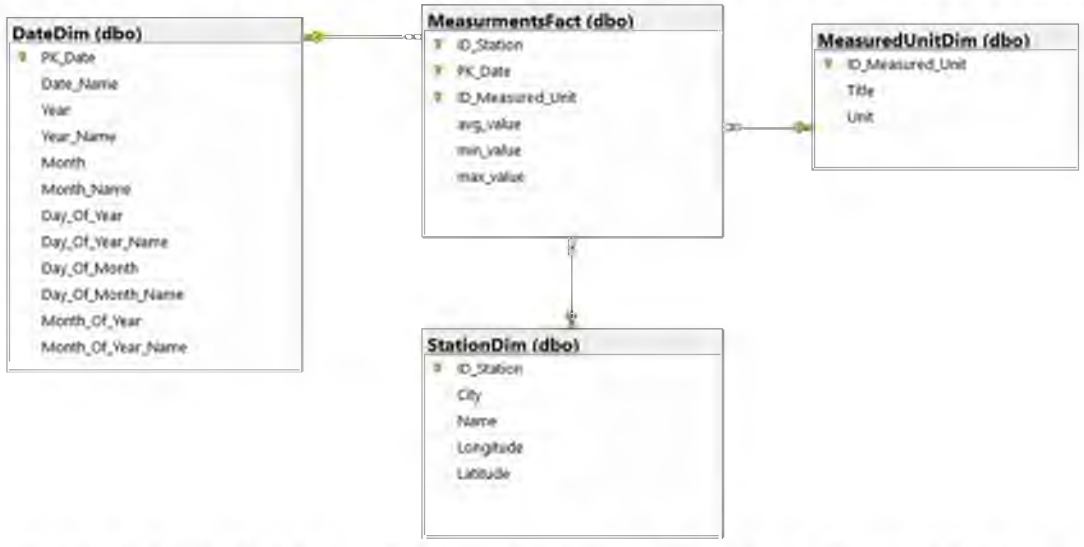

Рис. 11 - Структура сховища даних

MeasuredUnitDim – містить дані про область, де знаходиться господарство, а саме назва регіону та кліматичної зони;

- StationDim – містить інформацію про господарство;
- DateDim – часовий вимір (Рік – місяць – день).

Факт – це величина (зазвичай числова), яка є предметом аналізу. Таблиця фактів представленого сховища:

MeasurmentDim, містить інформацію у розрізі часу, станції і виміру інформацію про:

- Середнє значення виміру;
- Максимальне значення виміру;
- Мінімальне значення виміру;

Power BI. Power BI — це загальна назва для асортименту хмарних програм і служб, які допомагають організаціям збирати, керувати та аналізувати дані з різноманітних джерел за допомогою зручного для користувача інтерфейсу[20].

Такі інструменти бізнес-аналітики, як Power BI, можна використовувати для багатьох цілей.

В першу чергу Power BI об’єднує дані та обробляє їх, перетворюючи їх на зрозумілу інформацію, часто використовуючи візуально привабливі та прості в обробці діаграми та графіки. Це дозволяє користувачам створювати та ділитися чіткими та корисними знімками того, що відбувається в їхньому бізнесі.

Power BI підключається до низки джерел даних, від базових електронних таблиць Excel до баз даних, а також до хмарних і локальних програм.

Power BI — це щось на кшталт загального терміну, і воно може стосуватися або настільної програми Windows під назвою Power BI Desktop, онлайн-служби SaaS (програмне забезпечення як послуга) під назвою Power BI Service, або мобільних програм Power BI, доступних на телефонах і планшетах Windows, як а також для пристроїв iOS та Android.

Power BI побудовано на основі Microsoft Excel, і тому крива навчання від Excel до Power BI не така крута, будь-хто, хто вміє користуватися Excel, може використовувати Power BI, але останній набагато потужніший, ніж його аналог електронної таблиці.

Що робить Power BI?

Microsoft Power BI використовується для створення звітів і аналізу інформації на основі даних компанії. Power BI може підключатися до широкого діапазону наборів даних і «приводить у порядок» інформацію, яку він подає, щоб її можна було краще засвоїти та зрозуміти. Звітами та візуальними матеріалами, створеними на основі цих даних, можна поділитися з іншими користувачами.

Power BI допомагає користувачам побачити не лише те, що сталося в минулому та теперішньому, але й те, що може статися в майбутньому. Power BI оснащений можливостями машинного навчання, тобто він може виявляти закономірності в даних і використовувати їх, щоб робити обґрунтовані прогнози та запускати сценарії «що, якщо». Ці оцінки дозволяють користувачам створювати прогнози та готуватися до задоволення майбутнього попиту та інших ключових показників.

Чому люди використовують Power BI?

Для компаній, яким потрібна більша потужність звітності та аналітичні можливості, ніж ті, що пропонує Excel, Power BI — це наступний рівень бізнес-аналітики[21]. За допомогою Power BI компанії можуть збирати, ретельно аналізувати та візуалізувати дані з усієї компанії, надаючи їм краще розуміння своїх операцій і ефективності та дозволяючи їм приймати більш обґрунтовані рішення на основі реальних даних.

У громадській системі моніторингу атмосферного повітря інструмент Power BI використовується для визначення основних показників ефективності за відповідними екологічними вимірами та містами. Показник ефективності Air Quality Index для м.Київ відображено на рисунку 12 та показник ефекривності PM10 на рисунку 13.

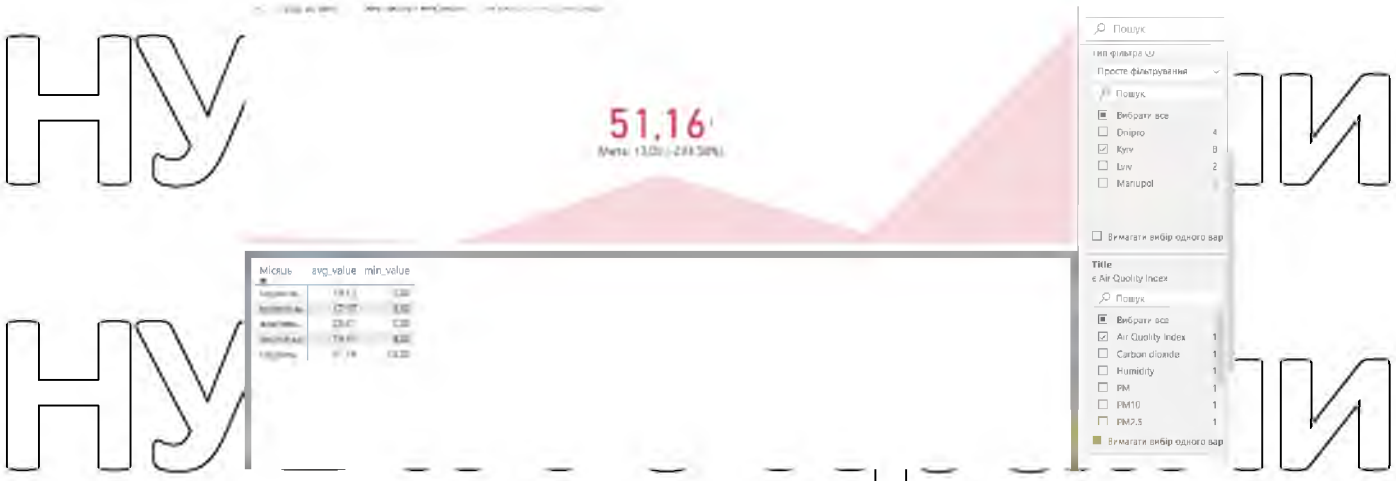

Рис. 12 Показник ефективності AQI

Рис. 13 Показник ефективності PM10

3.3 Алгоритмізація та програмування

Запис даних до оперативної бази даних відбувається що 2 хвилини, по кожній з активних станцій. Під час цього процесу робиться відповідний запит до SaveEcoBot API, проводиться вибірка даних(рис. 14) за станціями, які наявні в громадській системі.

```
async function getMeasurementsBot() {
  var stationsID = []
  var stationsInfo = []
  var stationsIDtoEcoID = []
  var connection = new Connection(config.ecoSensors)
  connection.connect()
  connection.on('connect', function (err) {
    let sqlRequest = new Request(
      "select ID_Station, ID_SaveEcoBot from Station where Status = 'enabled' AND ID_SaveEcoBot IS NOT NULL",
      function (err, rowCount) {
        connection.close()
        if (err) {
          console.log(err)
        } else {
          //console.log(stationsID);

          request(
            'https://api.saveecobot.com/output.json',
            { json: true },
            function (error, response, body) {
              console.error('error:', error) // Print the error if one occurred
              console.log('statusCode:', response && response.statusCode) // Print the response status code if a response was received

              body.forEach((station) => {
                if (stationsID.includes(station.id)) {
                  stationsInfo.push(station)
                }
              })

              //console.log(stationsID);
              console.log(stationsIDtoEcoID)

              stationsInfo.forEach((station) => {
                let tempStationID = stationsIDtoEcoID.find(
                  (element) => element.ID_SaveEcoBot == station.id
                )

                station.pollutants.forEach((measurment) => {
                  let unitID = saveEcoBotMeasurmentUnitId[measurment.pol]
                  writeOneMeasurment(
                    tempStationID.ID_Station,
                    measurment.value,
                    unitID
                  )
                })
              })
            }
          )
        }
      }
    )
```

Рис. 14 Отримання даних із SaveEcoBot API

Для отримання вимірів екологічних станцій, серверу необхідно отримати дані:

- проміжок часу;
- ID станції;
- екологічний параметр, за яким здійснюються виміри;

Код який проводить вибірку вимірювань станцій з front-end частини зображено на рисунку 15. Та з back-end частини на рисунку 16.

```
getStationMeasurements: (DateFrom, DateTo, ID_Station, ID_Measured_Unit) =>
  instance
    .post(
      "api/station/measurements",
      {
        DateFrom: DateFrom,
        DateTo: DateTo,
        ID_Station: ID_Station,
        ID_Measured_Unit: ID_Measured_Unit,
      },
      {
        headers: {
          "x-auth-token": localStorage.getItem("token"),
        },
      }
    )
    .then((response) => response.data),
```

Рис. 15 Вибірка вимірів з front-end частини

```
router.post("/measurements", auth, async (req, res) => {
  const { DateFrom, DateTo, ID_Station, ID_Measured_Unit } = req.body
  var connection = new Connection(configDB.user(req.user))
  connection.connect()
  connection.on("connect", function (err) {
    var all = []
    request = new Request(
      `select * from Measurment
                    where Time ≥ '${DateFrom}' and Time < '${DateTo}' and ID_Station   '${ID_Station}' and ID_Measured_Unit = ${ID_Measured_Unit} order by Time;`,
      function (err,          ,       ) {
        connection.close()
        if (err) {
          console.log(err)
          res.status(500).send("Server error")
        } else {
          res.json(all)
        }
      }
    )
    request.on("row", columns => {
      var row = {}
      columns.forEach(column => {
        row[column.metadata.colName] = column.value
      })
      all.push(row)
    })
    connection.execSql(request)
  })
})
```

Рис. 16 Вибірка вимірів з back-end частини

На серверній частині відбувається запит до сховища даних, за певний період часу який обрав користувач. Після отримання даних на сервері, вони відразу передаються до інтерфейсу користувача.

Вже на клієнтській частині ці виміри форматуються в необхідний формат для бібліотеки Recharts за допомогою функції formatChartObject(рис. 17).

```
export const formatChartObject = (array, nameStation = false) => {
  const name = nameStation && nameStation
  const done = [
    array.map(m => {
      return {
        value: Math.ceil(m.Value),
        date: moment(m.Time).format("YYYY/MM/DD HH:mm"),
        name
      }
    })
  ]
  return done[0]
}
```

Рис. 17 Форматування отриманих даних

Для отримання всіх станцій наявних в громадській системі, клієнт формує рядок запиту в залежності від обраних користувачем полів(наприклад тільки обрані станції, чи відсортовані станції за містом). Після формування рядка запиту, він відправляється на сервер, звідки вже в свою чергу буде виконуватись запит то оперативної бази даних, після отримання відповіді, усі наявні станції будуть відображені на головній сторінці користувача.

Фрагмент коду, який проводить вибірку станцій наявних в системі з клієнтської сторони зображено на рисунку 18, а також зі сторони сервера обробки даних на рисунку 24.

```
getAllStations: (string) =>
  instance
    .get(`api/station/${string}`, {
      headers: {
        "x-auth-token": localStorage.getItem("token"),
      },
    })
    .then((response) => response.data),
getStationsUnit: (id) =>
  instance
    .post(
      "api/station/units/",
      { ID_Station: id },
      {
        headers: {
          "x-auth-token": localStorage.getItem("token"),
        },
      }
    )
    .then((response) => response.data),
```

Рис. 18 Отримання станції на клієнтській стороні

```
var url = req.query
var connection = new Connection(configDB.user(req.user))

connection.connect()
connection.on("connect",
  let requestStr = `select ID_Station, Name, Status, ID_Server, ID_SaveEcoBot, Longitude, Latitude, (select Favorite_Station.ID_Station from Favorite_Station where ID_User = ${req.user.id}
  let favStr = ""
  let orderStr = ""

    requestStr += ` where CHARINDEX('${url.searchString}', CONCAT(ID_Station, Name)) != 0`


  if (url.onlyFav) {
    if (url.onlyFav == "true") {
      favStr =
        (url.searchString ? " and" : " where") +
        ` exists (select * from Favorite_Station where ID_User = ${req.user.id} and Favorite_Station.ID_Station = Station_Coordinates.ID_Station)`


    orderStr = " order by ID_Station"
    orderStr = " order by ID_Station DESC"


  request = new Request(requestStr + favStr + orderStr, function (err,
    console.log(err)
    res.status(500).send("Server error")
    res.json(all)
  })
  request.on("row", columns => {
    columns.forEach(column => {
      row[column.metadata.colName] = column.value
    })
    all.push(row)
  })
```

Рис. 19 Отримання станцій з БД. Серверна частина.

Після отримання всіх станцій та їх екологічних показників, користувач має можливість переглянути відповідні виміри по кожній станції окремо.

Всі дані які надходять з серверу, зберігаються в сховищі Redux. Кожний об’єкт або масив інформації який був отриманий за запитом має власну комірку у сховищі. Всі дані, які знаходяться там відображаються на сайті. Це дозволяє при змінні інформації в сховищі змінювати її на сайті без перезавантаження сторінки, що є надзвичайно важливим для такої системи, як моніторинг екологічних параметрів.

Саме сховище представляє собою state(стан), в якому зберігаються всі отримані дані а також редюсери які проводять обробку даних для їх правильного відображення в системі. Конфігурація сховища відображена на рисунку 20.

```
import { configureStore } from "@reduxjs/toolkit"
import authReducer from "../features/authSlice"
import stationsReducer from "../features/stationsSlice"
import compareReducer from "../features/compareSlice"

export default configureStore({
  reducer: {
    auth: authReducer,
    stations: stationsReducer,
    compare: compareReducer
  }
})
```

Рис. 20 Конфігурація Redux

Як можна побачити з конфігурації вона складається з 3 основних слоїв:

- Auth – відлов дає за авторизацію користувача в системі;
- Stations – об'єкт відповідає за всі станції які приходять з сервера та їхні виміри;
- Compare – об'єкт з додатковими даними по станціям які порівнюються.

Порівняння 2 станцій відбувається за їх спільними показниками, які наявні в обох. Вибірка таких показників відбувається за допомогою функції parseCommonUnits(рис. 21).

```
export const parseCommonUnits = (one, two) => {
  let array_second = []
  let array = []
  if (one.length > two.length) {
    array = one
    array_second = two
  } else {
    array = two
    array_second = one
  }
  let commonArray = []

  if (array) {
    for (let x of array_second) {
      for (let y of array) {
        x.ID_Measured_Unit === y.ID_Measured_Unit && commonArray.push(x)
      }
    }
  }
  return commonArray
}
```

Рис. 21 Функція пошуку спільних параметрів

Відображення станцій на мапі відбувається за допомогою їх координат які також приходять, в повній інформації з сервера. Компонент мапи зображений на рисунку 22.

```
return (
  <div className={s.mapWrapper}>
    <MapContainer center={[50.44034956835362, 30.5429873688861 38]} zoom={6} scrollWheelZoom={true} className={s.map}>
      {array.map(station => {
        const dot = [station?.Latitude, station?.Longitude]
        const path = `/station/${station.ID_Station}`

        return (
          <Marker position={dot} icon={Icon}>
            <Popup>
              <Link to={path}>{station.Name}</Link>
            </Popup>
          </Marker>
        )
      })}
      <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
    </MapContainer>
    <div className={s.menuWrapper}>
      <fieldset>
        <input
          type="checkbox"
          id="inputCheckSaveEco"
          checked={onlySave}
          onChange={() => setOnlySave(!onlySave)}
          disabled={onlyOwn}
        />
        <label htmlFor="inputCheckSaveEco">Show only SaveEcoBot stations</label>
      </fieldset>
      <fieldset>
        <input type="checkbox" id="inputCheckOwn" checked={onlyOwn} onChange={() => setOnlyOwn(!onlyOwn)} disabled={onlySave} />
        <label htmlFor="inputCheckOwn">Show only own stations</label>
      </fieldset>
    </div>
  </div>
```

Рис. 22 Компонент мапи

4 ВПРОВАДЖЕННЯ

4.1 Тестування системи

Після першого входу на сайт буде відображено сторінка входу користувача, рис. 23. Користувач для того щоб перейти до сторінки станції повинен мати свій логін та пароль, який видається адміністратором сайту. Наразі ця система є закритою до публічного доступу, оскільки необхідна більша кількість даних та стабільність роботи додатку.

Рис.23 Сторінка входу

Після входу користувача на сайт, він може бачити основні сторінки якимйому доступні(рис. 24). Наразі це 4:

- Сторінка всіх станцій;
- Сторінка однієї(обраної) станції;
- Сторінка порівняння 2 станцій;
- Сторінка мапи.

Відразу після входу, відображаються всі станції(рис. 25), по яким користувач може здійснити перехід, для більш детального ознайомлення з їхніми вимірами.

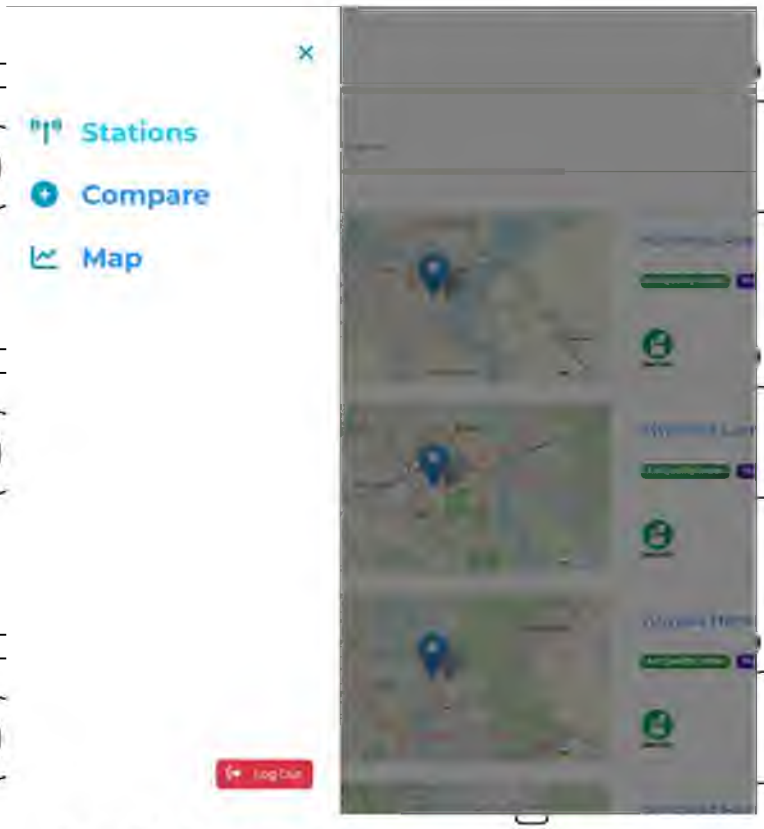

Рис. 24 Сторінки доступні користувачеві

Рис.25 Сторінка усіх станцій

На головній сторінці є можливість пошуку станції за її назвою(рис. 26), а також сортування по обраним(рис. 27), та власним(рис. 28).

Рис. 26 Пошук станції

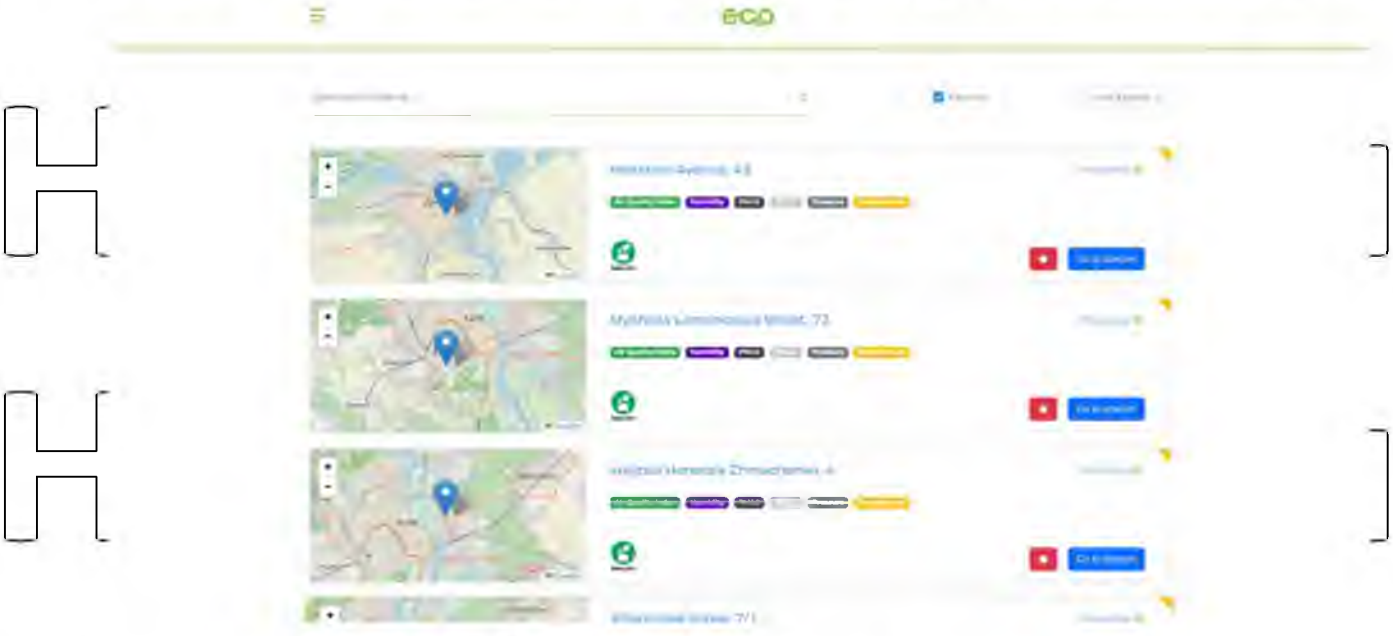

Рис. 27 Сортування за обраними станціями

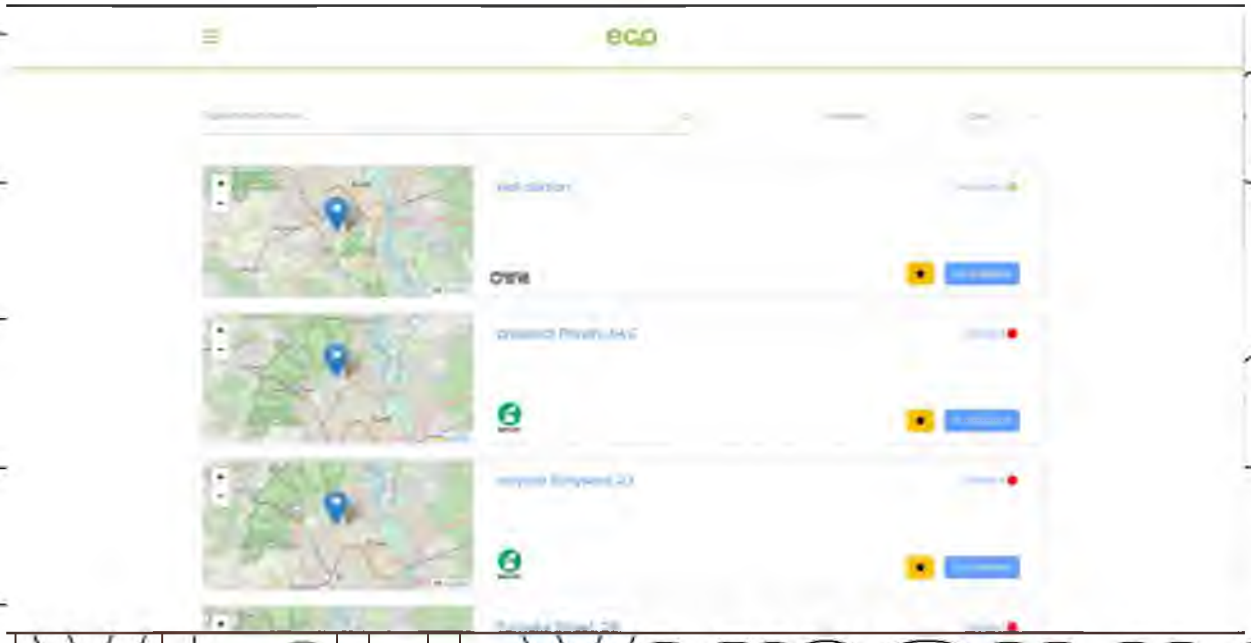

Рис. 28 Сортування за власними станціями

Користувач має можливість переходу на сторінку обраної станції(рис. 29), для більш детального перегляду даних. Після чого він може обрати відповідний параметр(рис. 30) для побудови графіка та період(рис. 31) за який необхідно відобразити виміри. Внизу сторінки є можливість перегляду мінімальних, максимальних та середніх значень за період(рис. 32), разом з кольоровим індикатором що відображає оптимальні рівні для показників.

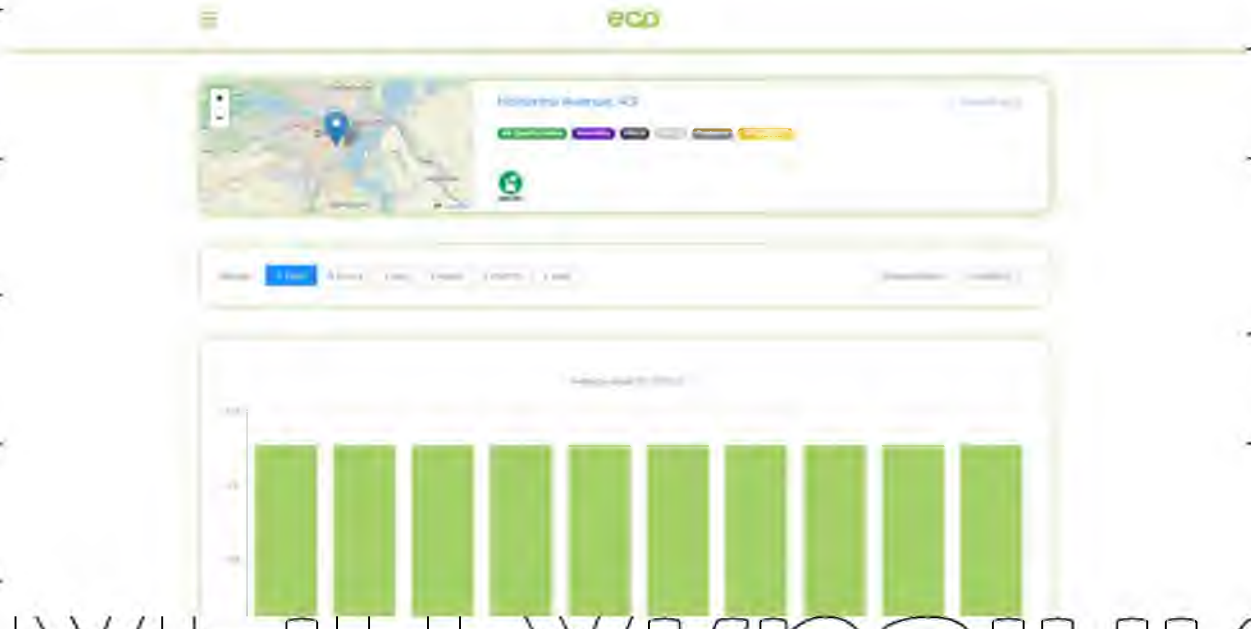

Рис. 29 Сторінка станції

Рис. 30 Вибір вимірюваного параметру

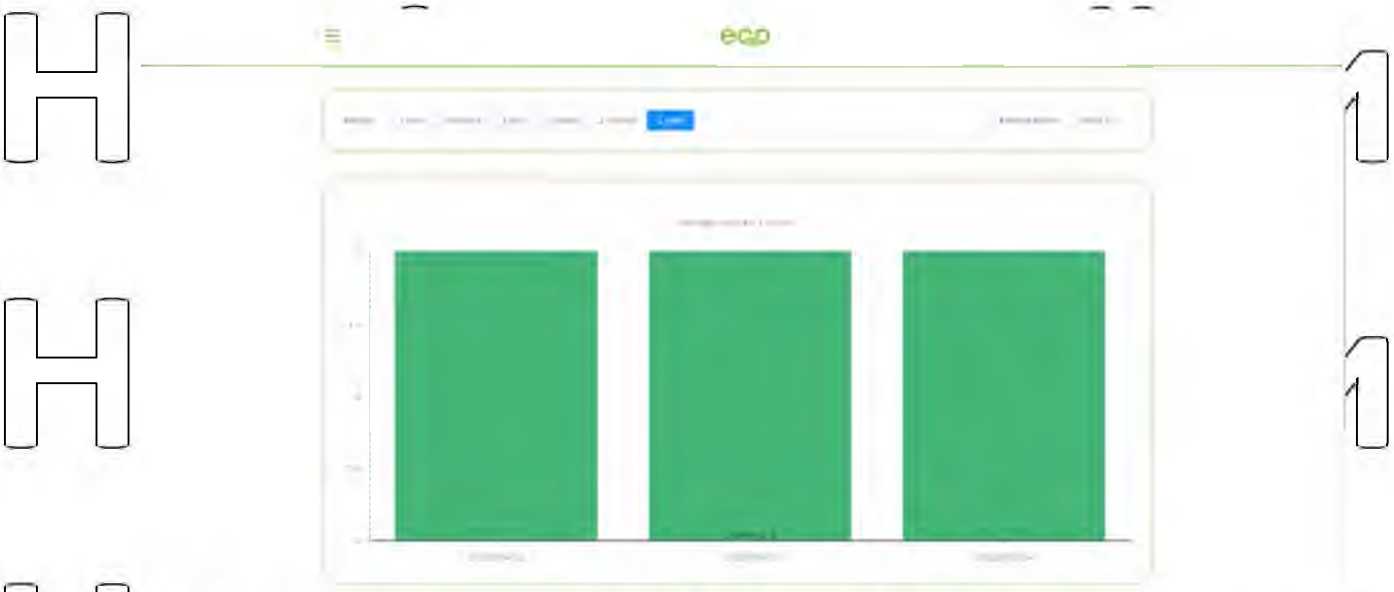

Рис. 31 Вибір періоду за рік

Рис. 32 Середнє, мін./макс. значення за обраний період

Сторінка порівняння станцій(рис. 33). Користувач має можливість обрати станції(рис. 34), їх спільний параметр(рис. 35) та період(рис. 36) за який він хоче переглянути дані по ним. Після чого буде сформований графік з двома лініями, кожна з яких буде відповідати за певну обрану станцію.

Рис. 33 Сторінка порівняння станцій

Рис. 34 Вибір станції

Рис. 35 Вибір спільного параметру

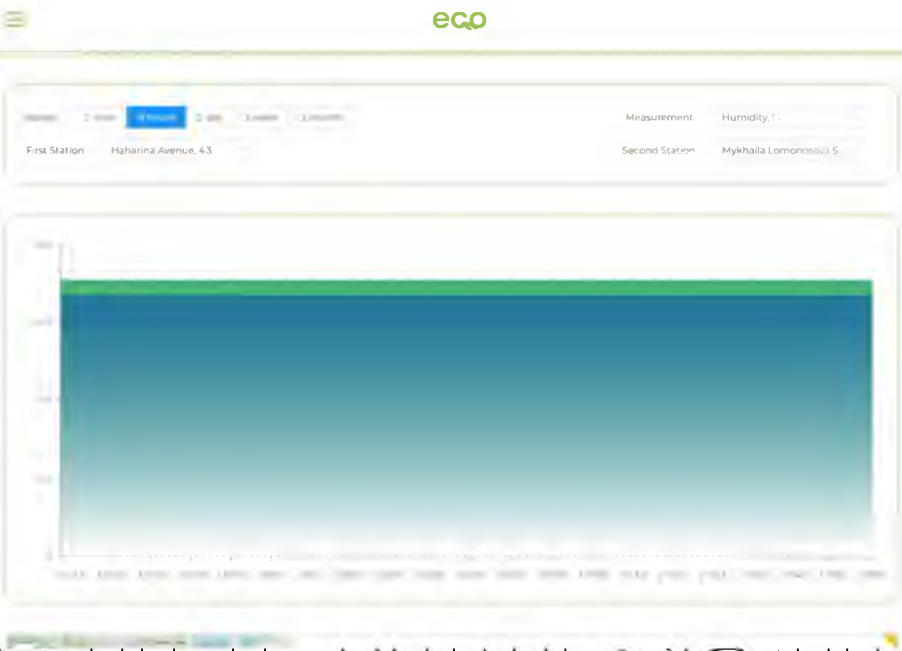

Рис. 36 Вибір періоду вимірювань

Мапа зі всіма станціями наявними в системі зображена на рисунку 37. Також на ній є можливість відображення власних чи SaveEcoBot станцій.

Рис. 37 Мапа станцій

4.2 Вимоги програмного та апаратного забезпечення

Вимоги до апаратного забезпечення.

Апаратні вимоги до комп'ютера, на якому буде розташований сервер бази даних та сховища даних:

- Процесор Pentium з мінімальною тактовою частотою від 2.7 ГГц (мінімум 4 ядра);
- Оперативна пам’ять – не менше 4 ГБ;
- Вільне місце на жорсткому диску – не менше 6 ГБ;
- Мережа від 50 Мбіт/с;

Апаратні вимоги до комп’ютера, на якому буде розташований сервер обробки даних:

- Процесор Pentium з мінімальною тактовою частотою від 2 ГГц (мінімум 2 ядра);
- Оперативна пам’ять – не менше 4 ГБ;
- Вільне місце на жорсткому диску – не менше 1 ГБ;
- Мережа від 10 Мбіт/с;

Вимоги по програмного забезпечення.

З програмного забезпечення знадобиться лише Node.js версії 14+.

Датчик виміру якості повітря, необхідні складові:

1. Пиломір SDS011;
2. Контродер Wemos D1 mini V2 Pro;
3. Сенсор температури, вологості та тиску BME280;
4. Пластиковий корпус;
5. Блок живлення;
6. Вбудована автоматизована камера підігріву повітря;
7. Вбудовані перетворювач живлення (5->3.3V) та апаратний сторожовий таймер (watchdog);
8. Можливість розширення додатковим високоякісним сенсором двоокису вуглецю (CO2) Senseair S8 ;
9. При виробництві використовується прилют, який не містить важких металів та відповідає RoHS[22].

10. Маркування на корпус наноситься шляхом ультрафіолетового друку, що мінімізує використання полімерних матеріалів та дозволяє відмовитись від шкідливої для довкілля ПВХ-стрічки

ВИСНОВКИ

На меті дипломного проекту було автоматизувати збору вимірюваних параметрів та забезпечити відображення цих даних у громадській системі моніторингу атмосферного повітря, а також інформувати громадськість що до стану повітря, шляхом створення системи.

У ході виконання дипломної роботи, було детально досліджено предметну область, проведено системний аналіз та побудовано діаграму прецедентів, побудована архітектура системи, що надало можливість детальніше розглянути можливості як інформаційної системи так і предметної області. Засобами реалізації інформаційної системи було створено оперативну базу даних, побудоване сховище даних, та проведені спеціальні операції для коректного збереження інформації у громадській системі моніторингу. Веб-додаток було створено за на основі технологій таких, як React.js для відображення інтерфейса користувача та Redux для зберігання даних, а також Node.js, за допомогою якого була побудована серверна частина проекту. Система була розміщена на безкоштовному хостингу – Heroku. У результаті роботи було створено громадську систему моніторингу екологічних параметрів. Пояснювальна записка містить 61 сторінку.

Таким чином, провідний еколог або будь-який громадянин країни має можливість спостерігати за станом атмосферного повітря в режимі реального часу, використовуючи при цьому доступні для нього апаратні та програмні засоби.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вплив навколишнього середовища на здоров'я людини. [Електронний ресурс]. – Режим доступу: http://lubotin-rada.gov.ua/news/id/2180

2. Air Pollution. [Електронний ресурс]. – Режим доступу: https://education.nationalgeographic.org/resource/air-pollution

3. Radon and Health. [Електронний ресурс] – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/radon-and-health

4. Проблематика громадського здоров'я [Електронний ресурс]. – Режим доступу: http://cgz.vn.ua/problematika-gromadskogo-zdorovya/problematika-gromadskogo-zdorovya_455.html

5. ВООЗ: як забруднене повітря впливає на здоров'я населення. [Електронний ресурс]. – Режим доступу: https://eco.rayon.in.ua/topics/487624-vooz-yak-zabrudnene-povitrya-vplivae-na-zdorov-a-naselennya

6. Inhalable Particulate Matter and Health(PM2.5 and PM10). [Електронний ресурс]. – Режим доступу: https://ww2.arb.ca.gov/resources/inhalable-particulate-matter-and-health

7. Основні поняття баз даних. [Електронний ресурс]. – Режим доступу: https://cutt.ly/rN1QV6r

8. Методологія інформаційних систем та баз даних: теоретичний і практичний підходи, с.103 [Посібник]. – Режим доступу: https://cutt.ly/fN1Wa9U

9. Загальна характеристика мови SQL – [Електронний ресурс]. – Режим доступу:

https://elib.ntu.edu.ua/sites/default/files/elib_upload/BD_2016_3/page14.html

10. Протокол MQTT. [Електронний ресурс]. – Режим доступу: https://pupenasan.github.io/TI40/%D0%9B%D0%B5%D0%BA%D1%86/MQTT.html

11. What is a Data Warehouse? [Електронний ресурс]. – Режим доступу: https://www.investopedia.com/terms/d/data-warehousing.asp

12. Data Mining. [Електронний ресурс]. – Режим доступу: https://www.javatpoint.com/data-mining

13. Огляд онлайнової аналітичної обробки(OLAP). [Електронний ресурс]. – Режим доступу: https://cutt.ly/mN1W8AM

14. Концепція сховищ даних. [Електронний ресурс]. – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php?file=/481684/mod_resource/content/2/%d0%9b%d0%b5%d0%ba%d1%86%d1%96%d1%8f%202.pdf

15. React. [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/

16. Redux. [Електронний ресурс]. – Режим доступу: https://redux.js.org/introduction/getting-started

17. Node.js - Introduction. [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

18. Node.js Case by case. [Стаття]. – Режим доступу: https://www.toptal.com/javascript/why-the-hell-would-i-use-node-js

19. Система управління базами даних [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/tehnikakomp/home/samostijne-vivcenna-materialu/sistema-upravlinna-bazami-danih-subd-subd-miereosoft-access

20. Power BI. [Електронний ресурс]. – Режим доступу

https://powerbi.microsoft.com/ru-ru/what-is-power-bi/
21. What is Power BI. [Електронний ресурс]. – Режим доступу: https://www.simplilearn.com/tutorials/power-bi-tutorial/what-is-power-bi
22. SaveDnipro. [Електронний ресурс]. – Режим доступу: https://www.savednipro.org/product/stanciya-monitoringu-yakosti-povitrya/

НУБіП України

НУБіП України

НУБіП України

НУБіП України

SQL Запити до Бази даних

НУБіП України

НУБіП України

НУБіП України

НУБіП України

Запити на створення уявлень

```
CREATE VIEW Measurment_Station_Unit AS
SELECT Value, Measured_Unit.Title AS Unit_Title,
       Unit, Time, Station.Name AS Station_Name,
       Measurment.ID_Station AS ID_Station
FROM Measurment INNER JOIN Station ON Measurment.ID_Station = Station.ID_Station
INNER JOIN Measured_Unit ON Measurment.ID_Measured_Unit = Measured_Unit.ID_Measured_Unit;

CREATE VIEW Station_Coordinates AS
SELECT Station.ID_Station, Name, Status,
       Longitude, Latitude
FROM Station INNER JOIN Coordinates ON Station.ID_Station = Coordinates.ID_Station;

CREATE VIEW Optimal_Value_Category AS
SELECT ID_Measured_Unit, Designation,
       Bottom_Border, Upper_Border
FROM Optimal_Value INNER JOIN Category ON Optimal_Value.ID_Category = Category.ID_Category;
```

Створення збереженої процедури Add_Station

```
CREATE PROCEDURE Add_Station
    @city nvarchar(20),
    @name nvarchar(40),
    @status varchar(20) = 'disabled',
    @id_server integer = NULL,
    @id_SaveEcoBot nvarchar(20) = NULL,
    @longitude float = NULL,
    @latitude float = NULL,
    @configuration_id integer = 1
AS BEGIN
    DECLARE @ID_Station char(4);
    IF @id_server is not NULL
        BEGIN
            SET @ID_Station = (SELECT CAST(Last_MQTT_ID AS CHAR) FROM System_Configuration
            WHERE Configuration_ID = @configuration_id);
            UPDATE System_Configuration
            SET Last_MQTT_ID = Last_MQTT_ID + 1
            WHERE Configuration_ID = @configuration_id;
        END
    ELSE
        BEGIN
            SET @ID_Station = (SELECT CAST(Last_SaveEcoBot_ID AS CHAR) FROM System_Configuration
            WHERE Configuration_ID = @configuration_id);
            UPDATE System_Configuration
            SET Last_SaveEcoBot_ID = Last_SaveEcoBot_ID + 1
            WHERE Configuration_ID = @configuration_id;
        END
    WHILE (LEN(@ID_Station) < IIF(@id_server is not null, 3, 4))
        SET @ID_Station = CONCAT('0', @ID_Station);
    IF @id_server is not NULL
        SET @ID_Station = CONCAT('1', @ID_Station);
    INSERT INTO Station(ID_Station, City, Name, Status, ID_Server, ID_SaveEcoBot)
    VALUES (@ID_Station, @city, @name, @status, @id_server, @id_SaveEcoBot);
    INSERT INTO Coordinates(ID_Station, Longitude, Latitude) VALUES (@ID_Station, @longitude, @latitude);
END
```

Запити на створення тригерів

```
CREATE TRIGGER Station_Delete
ON Station
INSTEAD OF DELETE
AS BEGIN
    DELETE FROM Coordinates
    WHERE ID_Station = (SELECT ID_Station FROM deleted);
    DELETE FROM Measurment
    WHERE ID_Station = (SELECT ID_Station FROM deleted);
    DELETE FROM Station
    WHERE ID_Station = (SELECT ID_Station FROM deleted);
END

delete from station where ID_Station = '0002';

CREATE TRIGGER Measured_Unit_Delete
ON Measured_Unit
INSTEAD OF DELETE
AS BEGIN
    DELETE FROM Optimal_Value
    WHERE ID_Measured_Unit = (SELECT ID_Measured_Unit FROM deleted);
    DELETE FROM Measurment
    WHERE ID_Measured_Unit = (SELECT ID_Measured_Unit FROM deleted);
    DELETE FROM Measured_Unit
    WHERE ID_Measured_Unit = (SELECT ID_Measured_Unit FROM deleted);
END
```

НУБіП України

НУБіП України

НУБіП України

НУБіП України

Код програми

НУБіП України

НУБіП України

НУБіП України

НУБіП України

api.js

```
import axios from 'axios';
import {loginUser} from '../redux/features/authSlice';
const instance = axios.create({
  baseURL: 'http://localhost:4000/',
  headers: {
   'Content-type': 'application/json',
  },
});
export const userAPI = {
  login: (login, password) =>
   instance
     .post('api/auth', {login: login, password: password})
     .then((response) => response.data),
};
export const stationsAPI = {
  getAllStations: (string) =>
    instance
      .get(`api/station/${string}`, {
        headers: {
          'x-auth-token': sessionStorage.getItem('token'),
        },
      })
      .then((response) => response.data),
  getStationsUnit: (id) =>
    instance
      .post(
        'api/station/units',
        {ID_Station: id},
        {
```

```
        headers: {
          'x-auth-token': sessionStorage.getItem('token'),
        },
      }
    )
    .then((response) => response.data),
  getStationFullUnits: (id) =>
    instance
      .post(
        'api/station/unitsFull/',
        {ID_Station: id},
        {
          headers: {
            'x-auth-token': sessionStorage.getItem('token'),
          },
        }
      )
      .then((response) => response.data),
  getStationOptimal: (idUnit) =>
    instance
      .post(
        'api/measurement/optimalValue',
        {ID_Measured_Unit: idUnit},
        {
          headers: {
            'x-auth-token': sessionStorage.getItem('token'),
          },
        }
      )
      .then((response) => response.data),
```

```
getStationMeasurements: (DateFrom, DateTo, ID_Station, ID_Measured_Unit) =>
  instance
    .post(
      'api/station/measurements',
      {
        DateFrom: DateFrom,
        DateTo: DateTo,
        ID_Station: ID_Station,
        ID_Measured_Unit: ID_Measured_Unit,
      },
      {
        headers: {
          'x-auth-token': sessionStorage.getItem('token'),
        },
      }
    )
    .then((response) => response.data),
  updateStatusFavorive: (ID_Station, isFavorite) =>
    instance
      .post(
        'api/station/changeFavorite',
        {ID_Station, isFavorite: `${isFavorite}`},
        {
          headers: {
            'x-auth-token': sessionStorage.getItem('token'),
          },
        }
      )
      .then((response) => response.data),
};
```

stationsSlice.js

```
import {createSlice} from '@reduxjs/toolkit';
import {stationsAPI} from '../../api/api';
import {formatChartObject, parseCommonUnits} from '../../util/util';

const initialState = {
  allStations: [],
  currentStation: {},
  loading: false,
  currentPageIndex: ['1'],
  page: 1,
};

export const stationsSlice = createSlice({
  name: 'stations',
  initialState,
  reducers: {
    setStations: (state, action) => {
      state.allStations = action.payload;
    },
    updateStations: (state, action) => {
      return {
        ...state,
        allStations: [
          ...state.allStations.map((s) => {
            if (s.ID_Station === action.payload.ID_Station) {
              return {
                ...s,
                ...action.payload,
                Favorite: action.payload.Favorite,
```

```
        };
      } else return {...s};
    }),
  ],
};
},
setStationsUnit: (state, action) => {
  state.allStations = state.allStations.map((s) => {
    if (s.ID_Station === action.payload.id) {
      return {
        ...s,
        units: [...action.payload.units],
      };
    } else return {...s};
  });
},
setCurrentPageIndex: (state, action) => {
  return {
    ...state,
    currentPageIndex: action.payload,
  };
},
setCurrentStation: (state, action) => {
  return {
    ...state,
    currentStation: {
      ...state.allStations.filter(
        (s) => s.ID_Station === action.payload
      )[0],
    },
```

```
    };
  },
  setStationFullUnits: (state, action) => {
    return {
      ...state,
      currentStation: {...state.currentStation, fullUnits: action.payload},
    };
  },
  setPage: (state, action) => {
    return {
      ...state,
      page: action.payload,
    };
  },
  setCurrentStationOptimal: (state, action) => {
    const optimal = state.currentStation.optimal
      ? [...state.currentStation.optimal, action.payload]
      : [action.payload];
    return {
      ...state,
      currentStation: {
        ...state.currentStation,
        optimal,
      },
    };
  },
  setLoading: (state, action) => {
    return {
      ...state,
      loading: action.payload,
```

```
    };
  },
  setMeasurements: (state, action) => {
    return {
      ...state,
      currentStation: {...state.currentStation, measurements: action.payload},
    };
  },
  setMeasurementsFormated: (state, action) => {
    return {
      ...state,
      currentStation: {
        ...state.currentStation,
        measurementsFormated: action.payload,
      },
    };
  },
  setSelectedMeasuredId: (state, action) => {
    return {
      ...state,
      currentStation: {
        ...state.currentStation,
        selectedMeasuredId: action.payload,
      },
    };
  },
  setUnitInfo: (state) => {
    return {
      ...state,
      currentStation: {
```

```
            ...state.currentStation,
            selectedUnitInfo: state.currentStation.fullUnits?.filter(
              (u) =>
                u.ID_Measured_Unit === state.currentStation.selectedMeasuredId
            ),
            selectedUnitInfoOptimal: state.currentStation.optimal?.filter(
              (u) =>
                u[0]?.ID_Measured_Unit === state.currentStation.selectedMeasuredId
            ),
          },
        };
      }
    },
  });
export const {
  setStations,
  setStationsUnit,
  setCurrentStation,
  setStationFullUnits,
  setCurrentStationOptimal,
  setLoading,
  setMeasurements,
  setMeasurementsFormated,
  setSelectedMeasuredId,
  setUnitInfo,
  setCurrentPageIndex,
  setPage,
  updateStations,
} = stationsSlice.actions;
//@Thunks
```

```
export const getStations = (string) => async (dispatch, getState) => {
 dispatch(setLoading(true));
 const data = await stationsAPI.getAllStations(string);
 await dispatch(setStations(data));
 const state = getState();
 state.stations.allStations.map(async (s) => {
  let unitData = await stationsAPI.getStationsUnit(s.ID_Station);
  dispatch(setStationsUnit({id: s.ID_Station, units: unitData}));
 });
 dispatch(setLoading(false));
};
export const setCurrentStationThunk =
 (id, DateFrom, DateTo, ID_Measured_Unit) => async (dispatch, getState) => {
  await dispatch(setCurrentStation(id));
  let data = await stationsAPI.getStationFullUnits(id);
  await dispatch(setStationFullUnits(data));
  let state = getState();
  state.stations.currentStation.fullUnits.map(async (u) => {
   let optimal = await stationsAPI.getStationOptimal(u.ID_Measured_Unit);
   await dispatch(setCurrentStationOptimal(optimal));
  });
  let measurements = await stationsAPI.getStationMeasurements(
   DateFrom,
   DateTo,
   state.stations.currentStation.ID_Station,
   state.stations.currentStation.fullUnits[0].ID_Measured_Unit
  );
  await dispatch(setMeasurements(measurements));
  state = getState();
  await dispatch(
```

```
    setMeasurementsFormated(
      formatChartObject(state.stations.currentStation.measurements)
    )
  );
  await dispatch(
    setSelectedMeasuredId(
      state.stations.currentStation.fullUnits[0].ID_Measured_Unit
    )
  );
  await dispatch(setUnitInfo());
};
export const setCurrentStationMeasurements =
  (DateFrom, DateTo, ID_Measured_Unit) => async (dispatch, getState) => {
  let state = getState();
  let measurements = await stationsAPI.getStationMeasurements(
    DateFrom,
    DateTo,
    state.stations.currentStation.ID_Station,
    ID_Measured_Unit
  );
  await dispatch(setMeasurements(measurements));
  state = getState();
  dispatch(
    setMeasurementsFormated(
      formatChartObject(state.stations.currentStation.measurements)
    )
  );
  dispatch(setLoading(false));
};
export const updateFavorite =
```

```
  (ID_Station, isFavorite) => async (dispatch, getState) => {
    dispatch(setLoading(true));
    const data = await stationsAPI.updateStatusFavorive(ID_Station, isFavorite);
    dispatch(updateStations(data));
    dispatch(setLoading(false));
  };
// @SELECTORS
export const selectAllStations = (state) => state.stations.allStations;
export const selectPage = (state) => state.stations.page;
export const selectCurrentPageIndex = (state) =>
  state.stations.currentPageIndex;
export const selectLoading = (state) => state.stations.loading;
export const selectCurrentStation = (state) => state.stations.currentStation;
export const selectSelectedUnitInfo = (state) =>
  state.stations.currentStation.selectedUnitInfo;
export const selectSelectedUnitInfoOptimal = (state) =>
  state.stations.currentStation.selectedUnitInfoOptimal;
export default stationsSlice.reducer;
```