НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____ комп'ютерних наук_____
(назва кафедри)

_____ _/_____Голуб Б.Л.

/

(підпис) (ПІБ)

"_02_" _червня_ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Програмне забезпечення системи моніторингу якості води»

Спеціальність 121 – «Інженерія програмного забезпечення»
ОПП «Інженерія програмного забезпечення»

Гарант освітньої програми

_____ _____ _____Вайганг Г.О.____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____д.т.н., професор_____ _____ _____Голуб Б.Л._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Васік Олександр Павлович_____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ комп'ютерних наук_____
(назва кафедри)

_____ /_____Голуб Б.Л.

/

(підпис) (ПІБ)

"_16_" __грудня__ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студента

_____Васік Олександр Павлович_____

(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення системи моніторингу якості води»

затверджена наказом ректора НУБіП України від "_16_" __грудня__2024р. №_2248.с__

Термін подання завершеної роботи на кафедру ____25.05.2025___

(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Рекомендації ВОЗ щодо якості питної води, 4-е видання.

Перелік питань, які потрібно розробити:

Вступ і постановка проблеми, цілі та сфера застосування, огляд літератури, методологія, архітектура та дизайн системи, впровадження, оцінка та результати_____

Дата видачі завдання "_16_"_____грудня_____2024 р.

Керівник бакалаврської кваліфікаційної роботи _____ __Голуб Б.Л.__

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Васік О.П.

(підпис) (прізвище та ініціали студента)

АНОТАЦІЯ

У бакалаврській кваліфікаційній роботі на тему «Система моніторингу якості води» описано розробку програмного забезпечення для збору, зберігання, аналізу та візуалізації даних про якість води. Основною метою є створення Windows Forms застосунку з функціями перегляду поточних та історичних даних, фільтрації, відображення інформації у таблицях і графіках, а також формування звітів у форматах Excel та PDF.

Клієнтська частина програми взаємодіє з базою даних PostgreSQL, що забезпечує оперативний доступ до актуальної інформації. Інтерфейс програми розроблений для зручності користувачів і дозволяє здійснювати моніторинг основних показників якості води за місцем розташування та часом. Система спрямована на покращення контролю за станом водних ресурсів, виявлення відхилень та аналіз змін параметрів у динаміці.

ABSTRACT

The bachelor's degree thesis on ‘Water Quality Monitoring System’ describes the development of software for collecting, storing, analysing and visualising water quality data. The main goal is to create a Windows Forms application with functions for viewing current and historical data, filtering, displaying information in tables and graphs, and generating reports in Excel and PDF formats.

The client part of the application interacts with the PostgreSQL database, which provides quick access to up-to-date information. The application interface is designed for user convenience and allows monitoring of key water quality indicators

by location and time. The system is aimed at improving control over the state of water resources, detecting deviations and analysing changes in parameters over time.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі питання забезпечення населення якісною питною водою та збереження водних екосистем є однією з найважливіших глобальних проблем. Зростання антропогенного навантаження на водні об'єкти внаслідок промислової діяльності, сільського господарства та комунального господарства призводить до їхнього забруднення різноманітними речовинами, що негативно впливає на здоров'я людей та стан довкілля. Традиційні методи контролю якості води, що включають ручний відбір проб та лабораторні дослідження, вимагають багато часу та не завжди забезпечують оперативне отримання повної картини стану водних ресурсів у реальному часі.

Саме тому розробка та впровадження автоматизованих систем моніторингу якості води, які здатні здійснювати безперервний збір даних,

їхню обробку та своєчасне інформування про критичні зміни параметрів, є надзвичайно актуальним завданням. Такі системи дозволяють підвищити ефективність контролю, оперативно реагувати на забруднення, прогнозувати можливі ризики та приймати обґрунтовані управлінські рішення для забезпечення екологічної безпеки та сталого використання водних ресурсів. Впровадження інформаційних технологій у сферу моніторингу якості води відповідає сучасним тенденціям цифровізації та є важливим кроком на шляху до покращення екологічної ситуації в країні.

Мета розробки. Метою даного дипломного проекту є розробка десктопного застосунку для автоматизованого моніторингу якості води. Запропоноване рішення дозволить здійснювати збір даних з локальних джерел (наприклад, датчиків), їхнє зберігання у базі даних, аналіз динаміки ключових параметрів та візуалізацію отриманих результатів у зручному для користувача інтерфейсі. Основне завдання полягає у демонстрації ефективного використання сучасних інформаційних технологій для створення функціональної системи, яка забезпечить оперативний контроль за станом водних ресурсів та сприятиме прийняттю обґрунтованих рішень у сфері їхнього управління.

Методи та технології.

У процесі розробки даної системи моніторингу якості води було використано наступні методи та технології:

Мова програмування C# — для розробки клієнтської частини Windows Forms застосунку та реалізації основної логіки обробки даних.

.NET Framework — як платформа для розробки застосунку.

Npgsql — бібліотека для забезпечення підключення та взаємодії з базою даних PostgreSQL.

PostgreSQL — як реляційна база даних для надійного зберігання даних про якість води, місця розташування та параметри.

Microsoft Chart Controls for .NET — для візуалізації даних у вигляді графіків.

EPPlus — бібліотека для створення звітів у форматі Excel.

iText 7 — бібліотека для створення звітів у форматі PDF.

Апробація.

Опубліковані тези “Програмне забезпечення системи моніторингу якості води” на міжнародній науково-практичній конференції студентів, аспірантів “Актуальні питання розвитку науки та техніки в умовах глобалізації” (місто Боярка. 14.05.2025)

Структура записки. Дипломна записка складається з сторінок, включає вступ, чотири основних розділів, висновки, список використаних джерел та додатки.

У першому розділі подано огляд сучасних методів та систем моніторингу якості води, розглянуто існуючі технології та їхні особливості, а також обґрунтовано актуальність розробки власної системи.

У другому розділі описано процес проектування інформаційного та програмного забезпечення, створенні необхідні діаграми.

У третьому розділі описана розробка програми, створені необхідні діаграми, блок-схеми алгоритмів, описано вибір інструментів розробки.

У четвертому розділі описані рекомендації щодо впровадження та експлуатації системи, показано тестування системи, описані вимоги до апаратного та програмного забезпечення, та вказано склад інсталяційного пакету.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Сучасний світ дуже сильно впливає на підходи до моніторингу та управління критично важливими ресурсами, серед яких особливе місце займає якість води [1]. В умовах зростаючого шкідливого навантаження на водні екосистеми, спричиненого промисловою діяльністю, сільським господарством та урбанізацією, забезпечення населення якісною водою стає першочерговим завданням [2]. Ефективний моніторинг якості води є ключовим елементом для своєчасного виявлення забруднень, оцінки стану водних об'єктів та прийняття обґрунтованих управлінських рішень [3].

На противагу застарілим методам контролю, що передбачають ручний відбір проб та лабораторний аналіз, сучасні технології пропонують комплексні екосистеми взаємопов'язаних пристроїв та програмних рішень для автоматизованого моніторингу якості води [4]. Серед ключових підсистем виокремлюються модулі збору даних (сенсори різного типу), передачі інформації, централізованої обробки та візуалізації отриманих результатів [5].

Автоматизований комплекс моніторингу якості води посідає пріоритетне місце серед інструментів екологічного управління через свій безпосередній вплив на здоров'я населення та стан довкілля. Інтеграція сучасних технологій в сенсорах, засобах передачі даних та програмного забезпечення для обробки та візуалізації інформації відкриває нові можливості для підвищення ефективності моніторингу та прийняття заходів щодо охорони водних ресурсів. Як приклад практичної імплементації можна навести автоматичне сповіщення про перевищення гранично допустимих концентрацій забруднюючих речовин або побудову прогнозних моделей на основі історичних даних.

Особливої значущості набуває фактор оптимізації процесу збору та аналізу даних — один із визначальних стимулів для імплементації подібних технологічних рішень [6]. Автоматизація процесів моніторингу мінімізує вплив людського фактору, забезпечує високу частоту вимірювань та надає можливість для оперативного реагування на зміни стану водних об'єктів.

Проектування та реалізація автоматизованого комплексу моніторингу якості води вимагає урахування комплексу технічних, функціональних та ергономічних аспектів. Зокрема, критично важливими є [5]:

- багатовекторність механізмів отримання даних;
- гнучка конфігурація параметрів;
- надійна передача даних;
- інтелектуальна обробка та аналіз;
- зручний інтерфейс користувача.

Параметри вимірювання води.

Система моніторингу якості води призначена для збору та аналізу широкого спектра параметрів, що характеризують стан водних об'єктів. Конкретний набір параметрів, що контролюються, може варіюватися залежно від типу водного об'єкта (річка, озеро, свердловина, водопровід), цілей моніторингу та вимог нормативних документів. Основні параметри представлені нижче.

Показник кислотності (pH)

Визначає концентрацію іонів водню у воді та є критично важливим для життєдіяльності водних організмів та протікання хімічних реакцій. Діапазон pH зазвичай становить від 0 до 14, де 7 є нейтральним значенням.

Температура води

Впливає на розчинність газів (зокрема кисню), швидкість біологічних та хімічних процесів у воді.

Електропровідність (EC)

Характеризує здатність води проводити електричний струм і є показником загальної кількості розчинених іонів (солей, кислот, лугів).

Рівень розчиненого кисню (DO)

Концентрація молекулярного кисню у воді, необхідного для дихання аеробних організмів. Низький рівень DO може свідчити про забруднення органічними речовинами.

Каламутність (Turbidity)

Міра прозорості води, зумовлена наявністю завислих частинок (глини, мулу, органічних речовин, мікроорганізмів). Висока каламутність може погіршувати проникнення світла та ускладнювати фільтрацію.

Окисно-відновний потенціал (ОВП/ORP)

Характеризує здатність водного середовища віддавати або приймати електрони, що є важливим для оцінки його окисних або відновних властивостей та прогнозування хімічних реакцій.

1.2 Моделювання предметної області

У процесі розробки інформаційної системи важливо не лише розуміти її функціональність, а й мати чітке уявлення про структуру, поведінку та взаємодію між компонентами. Для цього застосовується уніфікована мова моделювання UML (Unified Modeling Language) — стандартний засіб візуального представлення систем. UML забезпечує узгоджене моделювання структури програмного забезпечення, що сприяє кращому аналізу, проектуванню та комунікації між учасниками розробки [10].

Мова UML охоплює низку діаграм, які дозволяють описати різні аспекти системи. Серед них виділяють діаграми варіантів використання (прецедентів), діаграми послідовності, діаграми класів, діаграми активності та інші. У межах даного проекту було побудовано діаграму прецедентів для представлення функціональності системи з погляду користувачів та діаграму послідовності, що ілюструє хронологію взаємодії компонентів системи. Застосування діаграм прецедентів та послідовності є корисним для опису функціональності системи та взаємодії її компонентів (сенсорів, мікроконтролера, користувача) [11].

1.2.1 Діаграма прецедентів. Діаграма прецедентів відображає функціональність системи з точки зору різних категорій користувачів та їхні можливі взаємодії з системою. Вона дозволяє визначити основні сценарії використання системи моніторингу якості води, такі як перегляд поточних даних, отримання сповіщень про перевищення норм та експорт даних. Представлена діаграма прецедентів (рис.1) відображає основну функціональність системи моніторингу якості води з точки зору її користувача.

Рис.1 Діаграма прецедентів

В таблиці 1 показано відношення прецедентів до виконуючих їх акторів

Прецедент	**Актор**
Отримання сповіщень про перевищення норм	Оператор
Контроль вимірювання даних	Оператор
Експорт звітів (PDF, Excel)	Оператор
Вимірювання параметрів води	Станція

Таблиця 1 – Прецеденти і актори

- Опис прецедентів.

Отримання сповіщень про перевищення норм

Система інформує оператора про випадки, коли значення параметрів якості води виходять за встановлені гранично допустимі концентрації.

Контроль вимірювання даних

Оператор має можливість контролювати процес вимірювання даних, можливо, переглядати поточні значення, статус датчиків тощо.

Експорт звітів (PDF, Excel)

Оператор може експортувати сформовані звіти у форматах PDF або Excel для подальшого аналізу чи документування.

Вимірювання параметрів води

Станція здійснює безпосереднє зчитування значень різних параметрів якості води.

Ця діаграма прецедентів дозволяє швидко зрозуміти, які функції доступні оператору та як автоматизована станція бере участь у процесі моніторингу якості води. Вона чітко показує основні способи взаємодії користувача та автоматизованої системи з метою контролю стану водних ресурсів.

1.2.2 Діаграма послідовності. Діаграма послідовності (рис.2), показує, як різні частини системи взаємодіють між собою в часі, коли виконується певна дія. Завдяки цій діаграмі ми можемо крок за кроком побачити, як система реагує на запити користувача та як різні її складові обмінюються інформацією для досягнення потрібного результату.

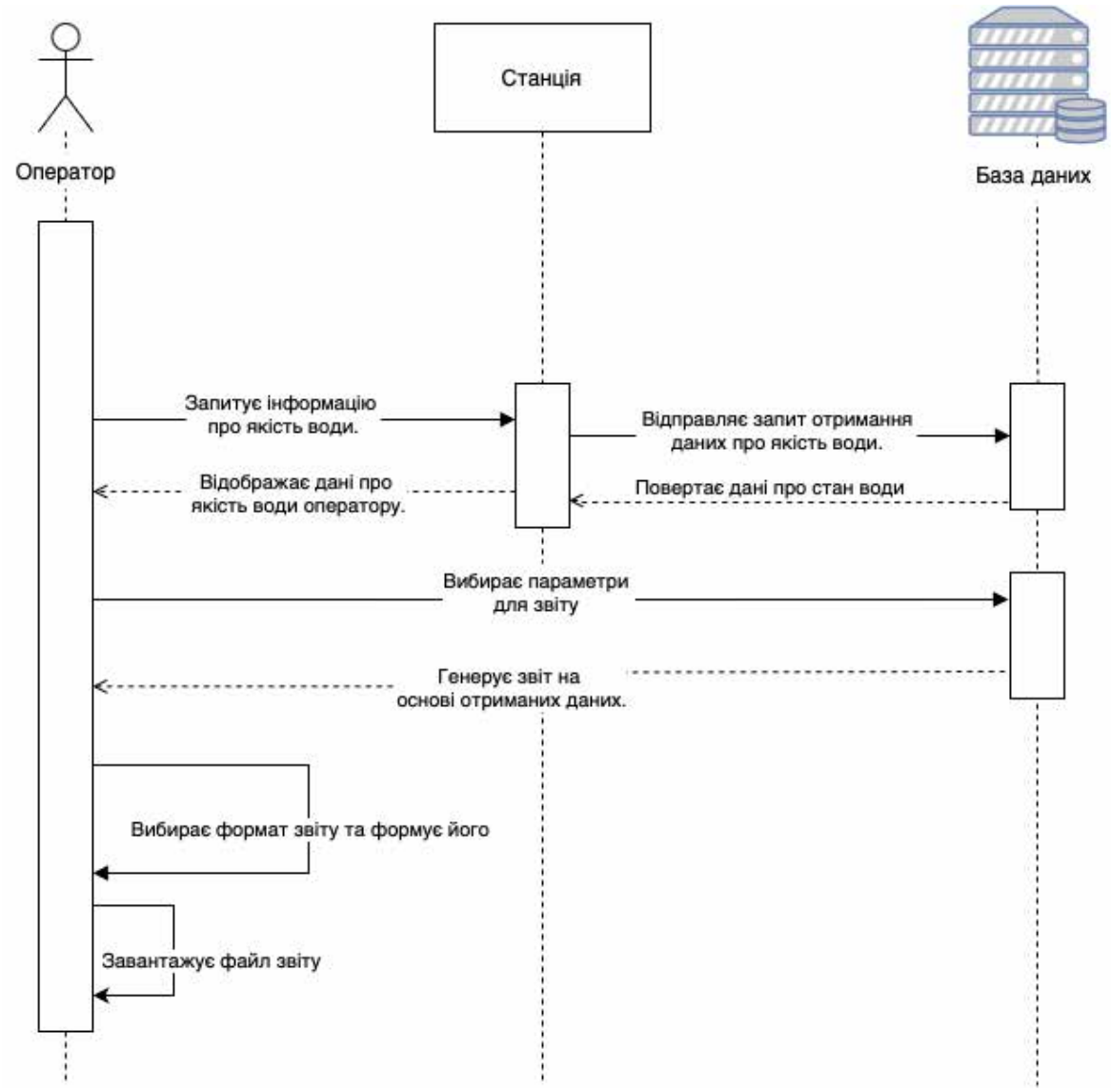

Рис.2 Діаграма послідовності

На рисунку 2 представлена діаграма послідовності, яка ілюструє хронологічну взаємодію між актором "Оператор", "Станцією" (автоматизованою системою збору даних) та "Базою даних" при виконанні основних сценаріїв використання системи моніторингу якості води.

Запит інформації про якість води.

Оператор надсилає запит до Станції для отримання інформації про поточний стан якості води. Станція, отримавши запит, відправляє запит до

Бази даних для отримання актуальних даних про якість води. База даних повертає Станції необхідні дані про стан води. Станція відображає отримані дані про якість води для оператора.

Генерація звіту.

Оператор обирає параметри для звіту. Ця дія ініціює процес у Станції (яка відповідає за формування звіту на основі отриманих даних). Станція формує запит до Бази даних для збору необхідних даних, відповідно до обраних оператором параметрів. База даних повертає Станції дані для звіту. Станція генерує звіт на основі отриманих даних та надає його оператору.

Експорт звіту.

Оператор обирає формат експорту звіту. Ця дія відбувається у Станції. Станція формує файл звіту у вибраному форматі. Станція надає оператору файл для завантаження. Оператор завантажує або зберігає файл звіту.

Ця діаграма послідовності наочно відображає обмін повідомленнями та послідовність дій між оператором, автоматизованою станцією збору даних та сховищем даних (базою даних) під час ключових сценаріїв використання системи моніторингу якості води. Ми бачимо, як оператор ініціює запити, станція обробляє їх, звертається до бази даних за інформацією та надає результат оператору у вигляді відображених даних або експортованого звіту.

1.3 Аналіз вимог до програмної системи

Загальні положення та функціональні вимоги.

Автоматизована система моніторингу якості води вимагає точного проектування та визначення функціональних можливостей, на які орієнтуватиметься програмна реалізація. Сформульовані вимоги дозволяють визначити межі проекту, структуру системи, порядок взаємодії її компонентів та майбутні сценарії використання.

Основні функціональні можливості системи.

- **Збір даних про якість води**
  - Регулярне автоматизоване зчитування даних з підключених датчиків якості води (pH, температура, електропровідність, рівень розчиненого кисню, каламутність тощо).
  - Можливість ручного введення даних про якість води (наприклад, результати лабораторних аналізів).
  - Підтримка різних типів датчиків та протоколів передачі даних.
- **Зберігання даних**
  - Структуроване зберігання отриманих даних у базі даних із зазначенням часу та місця вимірювання.
  - Ведення історії змін параметрів якості води.
  - Можливість експорту даних у різні формати (CSV, Excel тощо).
- **Візуалізація даних**
  - Відображення поточних значень параметрів якості води у зручному графічному або табличному вигляді.
  - Побудова графіків динаміки зміни параметрів за певний період часу.
  - Візуалізація даних на карті з відображенням точок моніторингу.
- **Аналіз та обробка даних**
  - Базові статистичні розрахунки (середнє значення, дисперсія тощо).
  - Формування звітів про стан якості води за визначений період.
- **Аутентифікація та авторизація користувачів**
  - Забезпечення безпечного доступу до системи через логін та пароль.
  - Розмежування прав доступу для різних категорій користувачів (наприклад, адміністратор, оператор, спостерігач).
  - Ведення журналу дій користувачів.
- **Інтерфейс користувача**

- Інтуїтивно зрозумілий графічний інтерфейс для перегляду даних, налаштування системи та формування звітів.
- Підтримка україномовної мов інтерфейсу.

Нефункціональні та системні вимоги.

Нефункціональні вимоги визначають якісні характеристики системи, що не залежать безпосередньо від функціональності, але мають важливе значення для її надійності, масштабованості, зручності використання та безпеки. Нижче наведено основні вимоги до автоматизованої системи моніторингу якості води.

- **Надійність та стабільність**
  - Система повинна функціонувати стабільно протягом тривалого часу, забезпечуючи безперервний збір та зберігання даних.
  - У разі тимчасової втрати зв'язку з окремими датчиками або точками моніторингу, система повинна продовжувати стабільну роботу з іншими компонентами та автоматично відновлювати збір даних після відновлення зв'язку.
- **Масштабованість**
  - Архітектура системи повинна забезпечувати можливість підключення нових датчиків та розширення мережі точок моніторингу без значних змін у програмному коді.
  - Передбачається можливість збільшення обсягу бази даних для зберігання зростаючих масивів історичних даних без суттєвого зниження продуктивності.
  - У майбутньому планується підтримка географічно розподілених точок моніторингу.
- **Продуктивність**

- Затримка між отриманням даних від датчиків та їх відображенням в інтерфейсі користувача не повинна перевищувати 3-5 секунд.
- Система повинна забезпечувати одночасну роботу з даними від значної кількості точок моніторингу без втрати швидкодії.
- Швидкість обробки запитів до бази даних повинна бути достатньою для швидкого формування звітів та візуалізації даних.

- **Безпека та захист даних**
- Обов'язкове використання механізмів аутентифікації для доступу до системи.
- Доступ до бази даних повинен бути обмежений на рівні користувацьких прав для запобігання несанкціонованому доступу.
- Планується реалізація аудиту дій користувачів для відстеження потенційних загроз безпеці.

- **Зручність та ергономіка**
- Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним для використання користувачами з різним рівнем технічної підготовки.
- Візуалізація даних повинна бути наочною та інформативною.
- Передбачається адаптивний дизайн інтерфейсу для коректного відображення на різних типах пристроїв (стаціонарні комп'ютери, планшети).
- Підтримка україномовної мови інтерфейсу.

- **Технологічна незалежність та відкритість**
- При розробці системи перевага надаватиметься використанню відкритих стандартів та безкоштовних бібліотек (наприклад PostgreSQL, NpgSQL, iText7 і тд).

- **Логування та аудит**
  - Система повинна вести детальні журнали подій, включаючи збір даних, системні помилки та дії користувачів.
  - Журнали повинні зберігатися в базі даних та бути доступними для адміністратора для аналізу та діагностики.

1.4 Огляд інформаційних джерел та існуючих рішень

У сфері моніторингу якості води вже існує ряд інформаційних джерел та готових рішень, які використовуються для контролю стану водних об'єктів. Огляд деяких програмних рішень, які можна розглядати як конкурентів, представлено нижче:

- **Aquarius (by Aquatic Informatics)** [7]. Одна з провідних платформ для управління водними даними. Забезпечує збір даних у реальному часі, потужні інструменти аналітики, автоматичну валідацію даних та інтеграцію з різними датчиками та системами. Її перевагами є широкий набір функцій та орієнтація на потреби великих підприємств та державних установ. Недоліком може бути висока вартість та складність впровадження для невеликих проектів.

Рис. 3 – Інтерфейс Aquaris

- **WISKI (by Kisters)** [8]. Комплексне програмне забезпечення для екологічного моніторингу, яке охоплює весь цикл управління даними – від збору до аналізу та звітності. Підтримує як хмарне, так і локальне розгортання, а також інтеграцію з телеметричними системами та лабораторними інформаційними системами (LIMS). До переваг можна віднести гнучкість та можливості налаштування. Серед недоліків –висока вартість та необхідність спеціалізованих знань для ефективного використання.

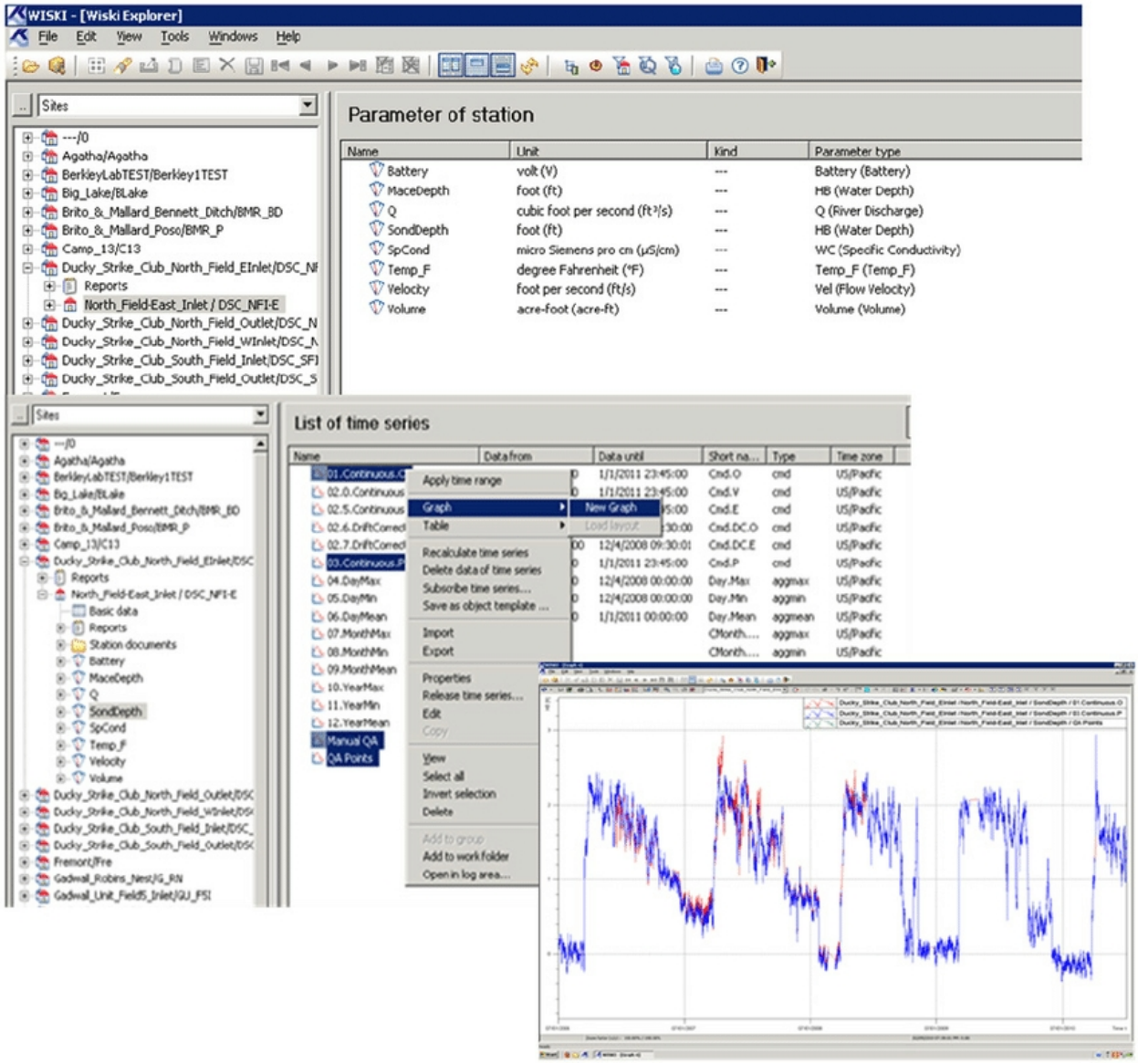

Рис. 4 – Інтерфейс WISKI

- **Envirosuite** [9]. Інтегрована платформа для моніторингу різних аспектів екологічної ситуації, включаючи якість води. Пропонує можливості моніторингу в реальному часі, візуалізацію даних на картах, інструменти для виявлення аномалій та прогнозування. Перевагою є зручний інтерфейс та комплексний підхід до екологічного моніторингу. Недоліком може бути орієнтація на широкий спектр екологічних

параметрів, що може бути надлишковим для вузькоспеціалізованих проектів з моніторингу якості води.

Рис. 5 - Інтерфейс Envirosuite

- **Open Waters** [10]. Безкоштовна веб-система з відкритим кодом, розроблена спеціально для управління даними про якість води. Підтримує управління точками моніторингу, пробами, результатами аналізів та забезпечує автоматизовану синхронізацію з EPA WQX (стандарт обміну даними про якість води в США). Перевагами є безкоштовність та можливість налаштування під власні потреби. Недоліками можуть бути обмежена функціональність порівняно з комерційними рішеннями та необхідність самостійної підтримки та розробки додаткових функцій.

1.5 Постановка задачі

Метою цього проекту є розробка прототипу автоматизованої системи моніторингу якості води, яка забезпечить безперервний збір даних з датчиків, їх централізоване зберігання, візуалізацію та базовий аналіз. Система повинна бути гнучкою у підключенні різних типів датчиків, зручною у використанні та економічно ефективною.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- **Розробити графічний інтерфейс користувача (GUI) на основі Windows Forms** для взаємодії користувача з системою, який дозволить переглядати поточні та історичні дані, налаштовувати точки моніторингу та експортувати дані.
- **Забезпечити збирання даних** з датчиків якості води у режимі реального часу.
- **Реалізувати механізм зберігання даних** у реляційній базі даних (наприклад, PostgreSQL) із зазначенням часу та місця вимірювання.
- **Розробити функціональність візуалізації даних** у вигляді графіків, таблиць для зручного аналізу та інтерпретації інформації.
- **Реалізувати базові функції аналізу даних**, такі як розрахунок середніх значень та побудова простих звітів за визначений період.
- **Створити систему аутентифікації та авторизації користувачів** з різними рівнями доступу до функціональності системи (наприклад, адміністратор, оператор, спостерігач).

- **Забезпечити масштабованість системи** для можливості підключення нових датчиків та розширення мережі точок моніторингу в майбутньому.

2 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних, часто у вигляді ER-діаграми, визначає концептуальну структуру бази даних. ER-діаграма показує ключові сутності системи та їхні зв'язки.

Сутності: Об'єкти, що моделюються (наприклад, "Результати", "Параметр").

Зв'язки: Асоціації між сутностями (наприклад, "Результати-Параметр").

Атрибути: Властивості сутностей (наприклад, ”значення”, ”ширина”, ”довгота”).

ER-діаграма створює високо-рівневу модель даних для проектування бази даних. Це допомагає спланувати зберігання даних та перевірити структуру. Такі діаграми корисні для візуалізації структури та зв'язків, забезпечуючи чітке розуміння архітектури даних для розробників.

Представлена ER-діаграма (рис. 6) візуалізує структуру бази даних PostgreSQL для системи моніторингу якості води.

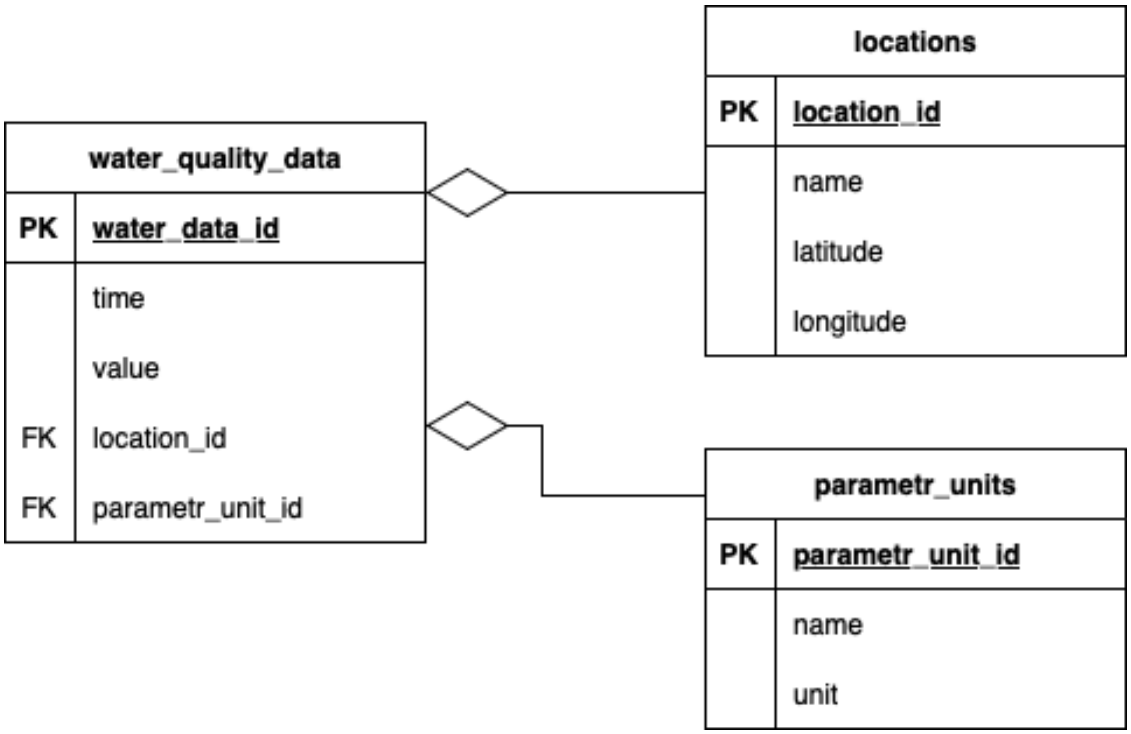

Рис.6 – ER-діаграма

Розглянемо кожну сутність та її атрибути:

Сутність locations

location_id (serial, PK): Унікальний ідентифікатор місця розташування (автоматично зростаючий первинний ключ).

name (character varying(100)): Назва місця розташування.

latitude (double precision): Географічна широта.

longitude (double precision): Географічна довгота.

Сутність parameter_units

parameter_unit_id (serial, PK): Унікальний ідентифікатор параметра та одиниці вимірювання (автоматично зростаючий первинний ключ).

name (character varying(100)): Назва параметра.

unit (character varying(20)): Одиниця вимірювання.

Сутність water_quality_data

water_data_id (serial, PK): Унікальний ідентифікатор запису вимірювання (автоматично зростаючий первинний ключ).

time (timestamp without time zone): Час проведення вимірювання.

value (double precision): Значення виміряного параметра.

location_id (integer, FK): Зовнішній ключ, що посилається на public.locations.

parameter_unit_id (integer, FK): Зовнішній ключ, що посилається на parameter_units.

Зв'язки між сутностями.

Location має багато записів у WaterQualityData (один-до-багатьох).

ParameterUnit має багато записів у WaterQualityData (один-до-багатьох).

Доведення відповідності моделі вимогам реляційності та третій нормальній формі (3NF).

Перша нормальна форма (1NF): Кожна таблиця має первинний ключ, усі записи є унікальними, а кожен атрибут містить атомарні (неподільні) значення.

Друга нормальна форма (2NF): Всі неключові атрибути повністю функціонально залежать від первинного ключа в кожній таблиці.

Третя нормальна форма (3NF): Жоден неключовий атрибут не є транзитивно залежним від первинного ключа. Зовнішні ключі використовуються лише для встановлення зв'язків між таблицями.

2.2 Обґрунтування вибору реляційної моделі (PostgreSQL):

Вибір реляційної бази даних PostgreSQL для даного проекту є обґрунтованим завдяки її сильним сторонам у сумісності з Windows Forms (WinForms), що є важливим для проекту, який передбачає використання

графічного інтерфейсу на базі WinForms. Завдяки підтримці ADO.NET та Entity Framework Core, інтеграція PostgreSQL у WinForms-додатки є простою та ефективною.

Для адміністрування та управління базою даних PostgreSQL доступний інструмент pgAdmin, який надає зручний графічний інтерфейс для виконання різноманітних завдань, таких як створення та модифікація таблиць, виконання SQL-запитів, управління користувачами та ролями, а також моніторинг продуктивності бази даних. pgAdmin підтримує кросплатформенність, що дозволяє використовувати його на різних операційних системах, включаючи Windows, що є додатковою перевагою для розробників WinForms-додатків .

Вибір PostgreSQL для даного проекту обґрунтований не лише його технічними можливостями у сфері обробки структурованих даних, але й зручністю інтеграції з WinForms та наявністю потужних інструментів для адміністрування, що забезпечує ефективну та зручну розробку та підтримку системи моніторингу якості води.

2.3 Створення таблиць бази даних

Для створення таблиць у базах даних використовуються SQL-запити. SQL-запит являє собою інструкцію, яка визначає структуру таблиці, включаючи назву, стовпці, їхні типи даних та обмеження. При створенні таблиці необхідно заздалегідь продумати її структуру, щоб забезпечити цілісність та ефективність зберігання даних.

У PostgreSQL створення таблиці здійснюється за допомогою команди CREATE TABLE. Ця команда дозволяє визначити назву таблиці, її стовпці, типи даних для кожного стовпця, а також різноманітні обмеження, такі як NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY тощо. Ці обмеження

допомагають забезпечити цілісність даних та уникнути помилок при їх введенні [17].

Приклад SQL-запиту для створення таблиці water_quality_data:

```
CREATE TABLE IF NOT EXISTS public.water_quality_data
(
    water_data_id integer NOT NULL DEFAULT nextval('water_quality_data_water_data_id_seq'::regclass),
    "time" timestamp without time zone NOT NULL,
    value double precision NOT NULL,
    location_id integer NOT NULL,
    parameter_unit_id integer NOT NULL,
    CONSTRAINT water_quality_data_pkey PRIMARY KEY (water_data_id),
    CONSTRAINT water_quality_data_location_id_fkey FOREIGN KEY (location_id)
        REFERENCES public.locations (location_id) MATCH SIMPLE,
    CONSTRAINT water_quality_data_parameter_unit_id_fkey FOREIGN KEY (parameter_unit_id)
        REFERENCES public.parameter_units (parameter_unit_id) MATCH SIMPLE

)
```

Рис.7 - Приклад SQL-запиту для створення таблиці water_quality_data

Інші SQL-запити розміщенні в ДОДАТОК А.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Діаграма класів

Діаграма класів (Додаток Б)— це UML-діаграма, яка описує класи з їхніми атрибутами, методами та зв’язки між ними.

Клас у ній є шаблоном для створення об’єктів; кожен клас містить набір атрибутів (дані) і методів (функції, операції). Діаграма показує, як класи взаємопов’язані: зазвичай за допомогою асоціацій (зв’язків), узагальнення (спадкування), а також агрегації/композиції, що виражають відношення «частина-ціле».

Наприклад, у системі моніторингу якості води, клас ViewChartsForm використовує DatabaseHelper для отримання даних з бази даних, а потім відображає їх у вигляді графіків.

У UML-діаграмі класів зазвичай виділяють такі елементи:

Клас (Class): шаблон об’єкта з назвою, списком атрибутів та методів.

Асоціація (Association): зв’язок між двома класами, що показує їх взаємозалежність (позначається суцільною лінією, може містити роль і кратність).

Наслідування (Generalization): ієрархічний зв’язок “підклас–надклас”, що означає, що один клас успадковує властивості іншого.

Агрегація/Композиція: спеціальні зв’язки «частина-ціле», де композиція означає сильну залежність об’єктів (частини не існують поза цілим).

Класова діаграма забезпечує високорівневий огляд архітектури системи, полегшує документування структури програмного забезпечення і є фундаментальним інструментом об’єктно-орієнтованого проектування.

Вона дозволяє розробникам спільно обговорювати дизайн, визначати взаємозв’язки між модулями та перевіряти повноту моделі. В додатку Б показана діаграма класів для системи моніторингу якості води.

Детальний опис призначення класів, їх методів та атрибутів.

- **ViewChartsForm**

  Призначення: Відображення графіків даних.

  Атрибути:

  dbHelper: Приватний атрибут, що є екземпляром класу DatabaseHelper. Використовується для взаємодії з базою даних.

  Методи:

  ViewChartsForm(): Конструктор класу. Ініціалізує об'єкт ViewChartsForm.

  LoadChartLocations():Task<void>: Завантажує дані про місцезнаходження для відображення на графіках (асинхронний метод).

  LoadChartParameters(): Task<void>: Завантажує параметри для відображення на графіках (асинхронний метод).

  generateChartButton_Click(sender: object, e: EventArgs): Обробник події натискання кнопки "Generate Chart". Генерує та відображає графіки.

- **ReportsForm**

  Призначення: Формування звітів.

  Атрибути:

  dbHelper: Приватний атрибут, що є екземпляром класу DatabaseHelper. Використовується для взаємодії з базою даних.

  connectionString: Приватний атрибут, що містить рядок підключення до бази даних.

Методи:

ReportsForm(): Конструктор класу. Ініціалізує об'єкт ReportsForm.

ReportsForm_Load(sender: object, e: EventArgs): Обробник події завантаження форми. Виконує початкове завантаження даних.

LoadLocations(): Task<void>: Завантажує дані про місцезнаходження для включення у звіти (асинхронний метод).

LoadParameters(): Task<void>: Завантажує параметри для включення у звіти (асинхронний метод).

generateReportButton_Click(sender: object, e: EventArgs): Обробник події натискання кнопки "Generate Report". Генерує звіти.

GenerateCombinedReport(dataTable: DataTable, locationName: string, parameterName: string, startDate: DateTime, endDate: DateTime): Task<void>: Генерує звіт на основі переданих даних та параметрів (асинхронний метод).

- **ViewTabularDataForm**

  Призначення: Відображення даних у табличному вигляді.

  Атрибути:

  dbHelper: приватний атрибут, що є екземпляром класу DatabaseHelper: використовується для взаємодії з базою даних.

  Методи: ViewTabularDataForm(): Конструктор класу. Ініціалізує об'єкт ViewTabularDataForm.

  LoadLocations(): Task<void>: Завантажує дані про місцезнаходження для фільтрації табличних даних (асинхронний метод).

  LoadParameters(): Task<void>: Завантажує параметри для фільтрації табличних даних (асинхронний метод).

LoadWaterQualityData(): Task<void>: Завантажує дані про якість води для відображення у таблиці (асинхронний метод).

applyFiltersButton_Click(sender: object, e: EventArgs): Обробник події натискання кнопки "Apply Filters". Застосовує фільтри до табличних даних.

- **DatabaseHelper**

  Призначення: Абстрагування та інкапсуляція взаємодії з базою даних**.**

  Атрибути:

  connectionString: Приватний атрибут, що містить рядок підключення до бази даних.

  Методи:

  DatabaseHelper(connectionString: string): Конструктор класу. Ініціалізує об'єкт DatabaseHelper з рядком підключення.

  ConnectionString: Публічна властивість, що повертає рядок підключення до бази даних.

Зв’язки.

Для реалізованої системи буде переважати асоціативний зв’язок, а саме:

- ViewChartsForm має асоціацію з DatabaseHelper. Це означає, що об'єкти ViewChartsForm використовують об'єкти DatabaseHelper для отримання даних, необхідних для побудови графіків.
- ReportsForm має асоціацію з DatabaseHelper. Об'єкти ReportsForm використовують DatabaseHelper для отримання даних, які потім форматуються у звіти.

- ViewTabularDataForm має асоціацію з DatabaseHelper. Об'єкти ViewTabularDataForm використовують DatabaseHelper для отримання даних, які відображаються у табличному вигляді.

3.2 Діаграма кооперацій

Діаграми кооперацій в UML демонструють поведінковий аспект системи, ілюструючи, як об’єкти (екземпляри класів) взаємодіють між собою для реалізації певного сценарію використання. На цих діаграмах зображаються об’єкти, їхні зв’язки та передані повідомлення, що дозволяє відстежити послідовність дій і визначити ролі кожного об’єкта в реалізації сценарію. Це сприяє уточненню обов’язків класів і перевірці наявності необхідних методів та інтерфейсів для виконання функціональних вимог системи[13].

Як приклад діаграма кооперацій для відображення графіку якості води (рис.8).

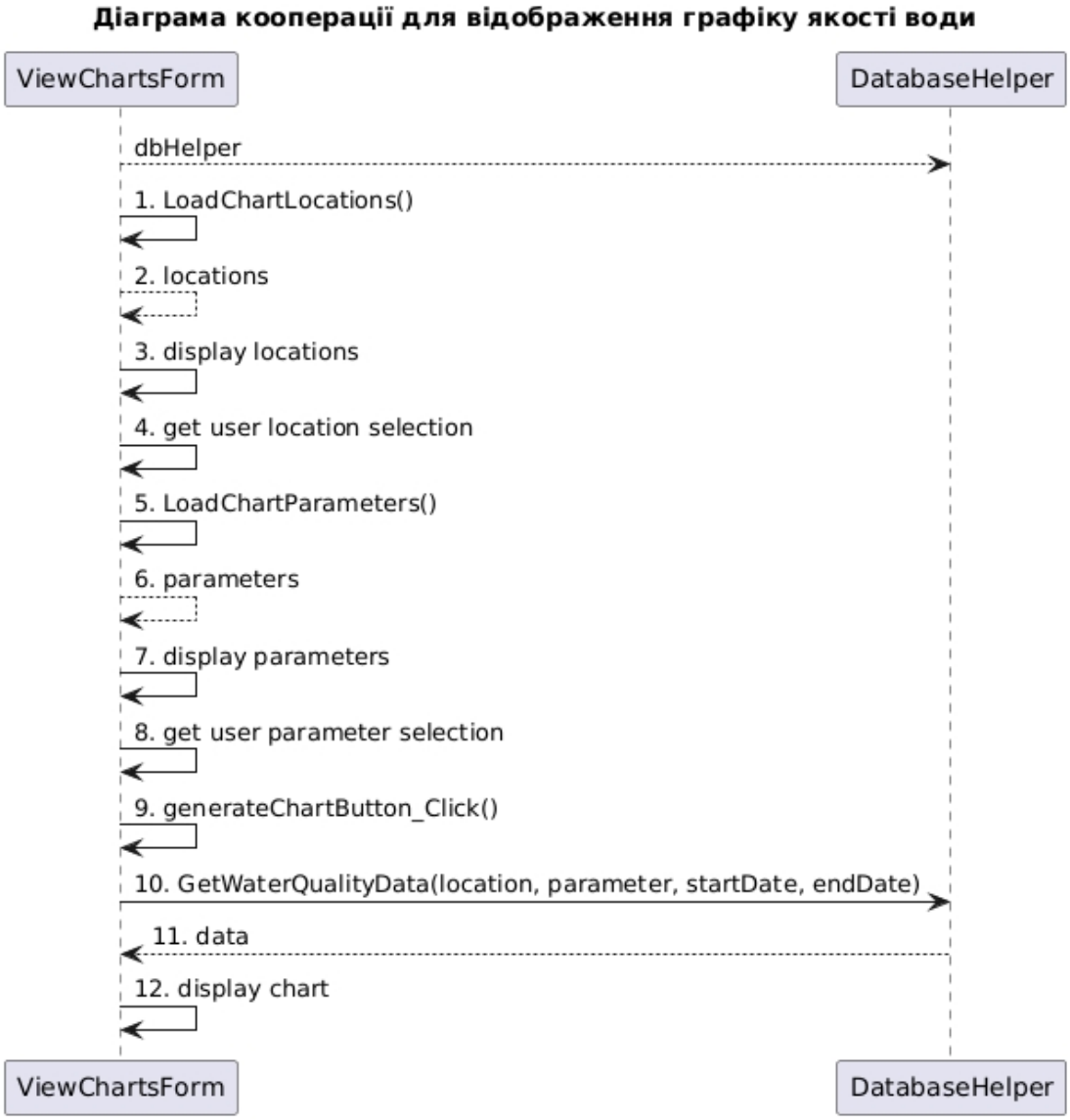

Рис.8 – Діаграма кооперацій

Ця діаграма кооперації відображає взаємодію між об'єктами в системі моніторингу якості води під час сценарію, коли користувач хоче переглянути графік якості води за певний період. Діаграма показує об'єкти, які беруть участь у цьому процесі, їхні зв'язки та послідовність обміну повідомленнями між ними.

Об'єкти.

ViewChartsForm: представляє об'єкт форми відображення графіків. Це інтерфейс, з яким взаємодіє користувач для ініціації та перегляду графіка.

DatabaseHelper: представляє об'єкт, відповідальний за взаємодію з базою даних. Він надає методи для отримання необхідних даних.

Зв'язок.

Між об'єктами ViewChartsForm та DatabaseHelper існує асоціативний зв'язок, позначений як dbHelper. Це вказує на те, що об'єкт ViewChartsForm має посилання на об'єкт DatabaseHelper і використовує його для доступу до даних.

Послідовність повідомлень.

ViewChartsForm надсилає повідомлення LoadChartLocations() самому собі. Це ініціює процес завантаження доступних місць розташування з бази даних (через DatabaseHelper).

ViewChartsForm отримує дані про місця розташування (locations) у відповідь на попереднє повідомлення.

ViewChartsForm відображає список доступних місць розташування для вибору користувачем.

Користувач здійснює вибір місця розташування.

ViewChartsForm надсилає повідомлення LoadChartParameters() самому собі. Це ініціює завантаження доступних параметрів вимірювань (також через DatabaseHelper).

ViewChartsForm отримує дані про параметри (parameters).

ViewChartsForm відображає список доступних параметрів для вибору користувачем.

Користувач здійснює вибір параметра.

ViewChartsForm надсилає повідомлення generateChartButton_Click() самому собі, що імітує натискання кнопки генерації графіка.

ViewChartsForm надсилає повідомлення GetWaterQualityData(location, parameter, startDate, endDate) до об'єкта DatabaseHelper, передаючи вибрані користувачем місце розташування, параметр та період часу.

DatabaseHelper обробляє запит та повертає набір даних (data) з бази даних до ViewChartsForm.

ViewChartsForm отримує дані та відображає графік якості води на основі цих даних.

Таким чином, діаграма кооперації показує, як об'єкт форми відображення графіків співпрацює з об'єктом, що відповідає за доступ до бази даних, для отримання та відображення необхідної інформації користувачеві.

3.3 Діаграма пакетів

Діаграма пакетів (Package Diagram) — це структурна діаграма в UML, яка використовується для організації та впорядкування елементів моделі, таких як класи, інтерфейси, компоненти та інші пакети. Вона дозволяє представити високорівневу структуру системи, показуючи, як різні частини взаємодіють між собою через залежності [14].

Основні елементи діаграми пакетів.

Пакет: група логічно пов’язаних елементів UML, яка служить для організації моделі в зручні для управління блоки. Пакети можуть містити інші пакети, утворюючи ієрархічну структуру [14].

Залежності: відображають відносини між пакетами, показуючи, як зміни в одному пакеті можуть впливати на інші. Існують два основних типи залежностей:

Імпорт пакета: означає, що елементи одного пакета доступні в іншому без необхідності повного включення їхньої реалізації [14].

Доступ до пакета: вказує, що один пакет використовує елементи іншого, але не включає їх у свій простір імен [14].

Діаграму пакетів використовують для:

Спрощення складних діаграм класів шляхом групування класів у пакети [14].

Відображення архітектури багаторівневих систем, де кожен рівень представлений окремим пакетом [14].

Організації великомасштабних проектів, забезпечуючи чітке уявлення про структуру та взаємозв'язки між компонентами системи [14].

На представленій діаграмі пакетів (рис.9) ми можемо побачити які пакети та класи присутні в системі.

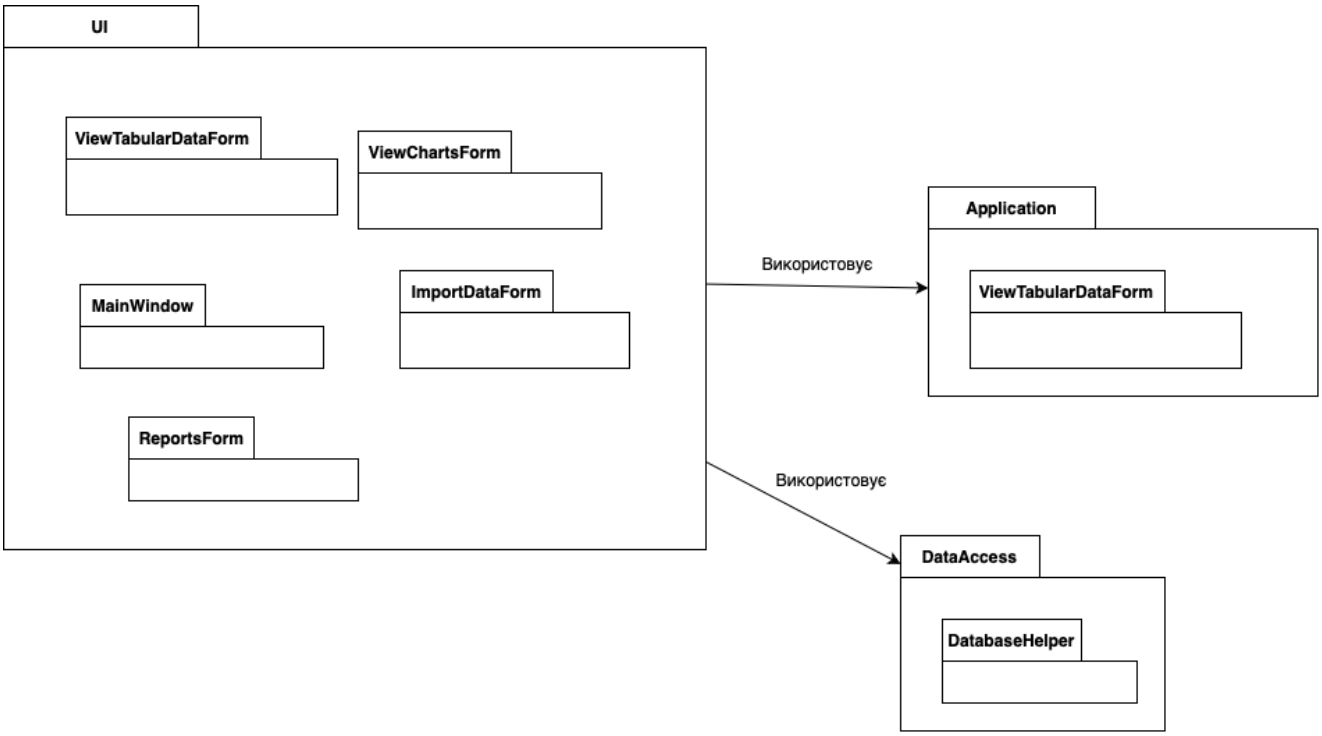

Рис. 9 – Діаграма пакетів

Ця діаграма пакетів відображає високо-рівневу логічну структуру системи моніторингу якості води, організовану в три основні пакети:

UI (Користувацький інтерфейс): Цей пакет відповідає за всі аспекти взаємодії користувача з системою. Він містить класи, що забезпечують графічне представлення даних та функціональність для користувача. До них відносяться:

MainWindow: головне вікно програми, що забезпечує загальну структуру інтерфейсу.

ViewChartsForm: форма для відображення даних у вигляді графіків.

ViewTabularDataForm: форма для відображення даних у табличному форматі.

ReportsForm: форма для створення та перегляду звітів.

ImportDataForm: форма для імпорту даних у систему.

Application (Додаток): цей пакет містить клас Program, який є точкою входу в систему та відповідає за її запуск та загальне управління. Він координує роботу різних частин системи.

DataAccess (Доступ до даних): цей пакет інкапсулює всю логіку, пов'язану з доступом до даних. Він містить клас DatabaseHelper, який надає методи для взаємодії з базою даних, абстрагуючи деталі конкретної СУБД.

Залежності між пакетами.

Пакет UI залежить від пакетів Application та DataAccess.

Це означає, що класи користувацького інтерфейсу використовують функціональність класів з пакетів Application (для управління життєвим циклом програми) та DataAccess (для отримання та відображення даних).

Ця діаграма пакетів забезпечує чітке уявлення про високо-рівневу архітектуру системи, показуючи основні функціональні модулі та їхні залежності.

3.4 Діаграма компонентів

Діаграма компонентів (Component Diagram) в UML — це структурна діаграма, яка відображає фізичну структуру програмної системи, зокрема її компоненти, інтерфейси та залежності між ними. Вона використовується для моделювання високорівневої архітектури системи та демонструє, як різні частини системи взаємодіють одна з одною через інтерфейси [15].

Компонент в UML представляє собою модульну частину системи, яка інкапсулює стан і поведінку низки класифікаторів. Його поведінка визначається через надані та необхідні інтерфейси, він є самодостатнім і може бути замінений іншим компонентом за умови, що їхні інтерфейси збігаються [16].

Основні елементи діаграми компонентів

Компоненти: Зображуються як прямокутники з міткою «component» або зі спеціальним символом у верхньому правому куті.

Інтерфейси: Показують, які сервіси надає або вимагає компонент.

Зв'язки: Відображають взаємодію між компонентами через їхні інтерфейси.

Застосування діаграм компонентів

Моделювання фізичної структури системи: Допомагає уявити, які компоненти входять до складу системи та як вони взаємодіють між собою [15].

Планування архітектури: Сприяє розробці модульної та масштабованої архітектури, де кожен компонент виконує чітко визначену функцію [15].

Аналіз залежностей: Дозволяє виявити та проаналізувати залежності між різними частинами системи, що важливо для її підтримки та розвитку [15].

Представлена діаграма компонентів (рис. 10) показує усі компоненти які є в системі моніторингу якості води.

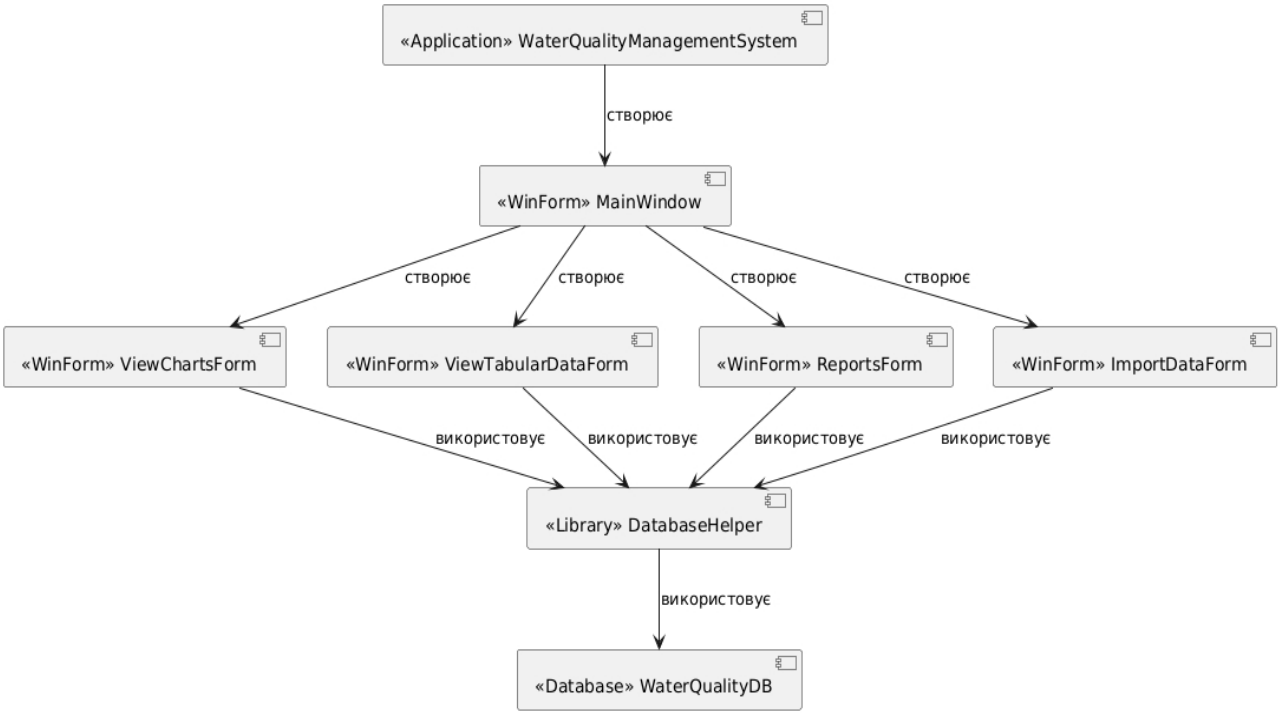

Рис. 10 – Діаграма компонентів

Компоненти:

WaterQualityManagementSystem (WQMS): Головний виконуваний модуль програми, що представляє всю систему.

MainWindow: Головне вікно програми, що забезпечує основний інтерфейс користувача.

ViewChartsForm: Компонент, що відповідає за відображення графіків якості води.

ViewTabularDataForm: Компонент, що відповідає за відображення табличних даних про якість води.

ReportsForm: Компонент, що відповідає за формування звітів про якість води.

ImportDataForm: Компонент, що відповідає за імпорт даних у систему.

DatabaseHelperLib: Бібліотека, що інкапсулює функціональність для взаємодії з базою даних.

WaterQualityDB: База даних, що зберігає дані про якість води.

Зв'язки:

WQMS --> MainWindow: використовує: Система використовує головне вікно для відображення інтерфейсу.

MainWindow --> ViewChartsForm: створює: Головне вікно створює екземпляр форми для відображення графіків.

MainWindow --> ViewTabularDataForm: створює: Головне вікно створює екземпляр форми для відображення табличних даних.

MainWindow --> ReportsForm: створює: Головне вікно створює екземпляр форми для створення звітів.

MainWindow --> ImportDataForm: створює: Головне вікно створює екземпляр форми для імпорту даних.

ViewChartsForm --> DatabaseHelperLib: використовує: Форма для відображення графіків використовує бібліотеку DatabaseHelperLib для отримання даних з бази даних.

ViewTabularDataForm --> DatabaseHelperLib: використовує: Форма для відображення табличних даних використовує бібліотеку DatabaseHelperLib для отримання даних з бази даних.

ReportsForm --> DatabaseHelperLib: використовує: Форма для створення звітів використовує бібліотеку DatabaseHelperLib для отримання даних з бази даних.

ImportDataForm --> DatabaseHelperLib: використовує: Форма імпорту даних використовує бібліотеку DatabaseHelperLib для збереження даних у бази даних.

DatabaseHelperLib --> WaterQualityDB: використовує: Бібліотека DatabaseHelperLib використовує базу даних WaterQualityDB для зберігання та отримання даних.

Ця діаграма показує, як програмні компоненти системи організовані та як вони взаємодіють один з одним для забезпечення функціональності системи моніторинга якості води.

3.5 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментарію для створення прикладного програмного забезпечення є дуже важливим етапом у процесі розробки.

Основними критеріями вибору були наявність широкої спільноти, великої кількості документації та інструментів для швидкого вирішення типових завдань. Можливість обробляти значні обсяги даних та забезпечувати швидкий відгук системи. Можливість інтеграції з іншими системами та технологіями. Відкритість програмного забезпечення.

Після проведення аналізу було обрано інструменти які забезпечать зручний і надійний процес розробки:

Мова програмування C# та платформа .NET:

C# – це сучасна, об'єктно-орієнтована мова програмування, розроблена Microsoft. Вона забезпечує високу продуктивність, безпеку типів та широкий спектр можливостей для розробки додатків різного типу.

.NET (зокрема .NET Framework або .NET Core) – це платформа для розробки, що надає велику бібліотеку класів (FCL), середовище виконання (CLR) та інструменти для створення настільних, веб- та хмарних додатків. Використання .NET дозволяє створювати надійні та масштабовані рішення з високою швидкістю розробки.

PostgreSQL (як система управління базами даних):

PostgreSQL – це відкрита об'єктно-реляційна система управління базами даних (СУБД). Вона підходить для зберігання великих обсягів даних моніторингу якості води, забезпечуючи цілісність даних та високу швидкість запитів.

Npgsql (для взаємодії з базою даних):

Npgsql – це .NET-провайдер даних для PostgreSQL, що дозволяє C#-додаткам взаємодіяти з базою даних PostgreSQL. Він забезпечує високу продуктивність та повну підтримку всіх функцій PostgreSQL.

OfficeOpenXml (EPPlus) (для експорту в Excel):

EPPlus – це бібліотека для .NET, яка дозволяє легко створювати, читати та маніпулювати файлами Excel (формат XLSX) без встановлення Microsoft Office. Вона була обрана для реалізації функціоналу експорту звітів у формат Excel, що є зручним для подальшого аналізу даних.

iText7 (для експорту в PDF):

iText7 – це потужна бібліотека для .NET (та Java), призначена для створення, маніпулювання та рендерингу PDF-документів. Її використання дозволяє генерувати професійні PDF-звіти, що є важливим для документування та поширення результатів моніторингу.

Windows Forms (для розробки користувацького інтерфейсу):

Windows Forms – це фреймворк для створення графічних інтерфейсів користувача (GUI) для настільних додатків на платформі .NET. Він забезпечує швидку розробку візуальних компонентів та подій, що дозволило створити інтуїтивно зрозумілий інтерфейс для системи моніторингу.

System.Windows.Forms.DataVisualization.Charting (для побудови графіків):

Цей вбудований компонент .NET Framework надає широкі можливості для візуалізації даних у вигляді різноманітних графіків та діаграм. Він був обраний для побудови графіків якості води, що дозволяє користувачам швидко аналізувати тенденції та зміни.

3.6 Алгоритмізація та програмування програмних модулів

Алгоритмізація та програмування програмних модулів є ключовим етапом у розробці програмного забезпечення, особливо в контексті модульного підходу, який забезпечує ефективність, масштабованість та зручність супроводу систем.

Модульне програмування передбачає розбиття програми на незалежні компоненти — модулі, кожен з яких виконує окрему функцію або обробляє певний аспект даних. Це дозволяє зосередитися на конкретних частинах програми, спрощує тестування та налагодження[18].

В системі моніторингу якості води модулями виступають класи WinForms. Для основних функцій системи були розроблені блок схеми. Блок схему для отримання та відображення даних про якість води (рис.11) можна побачити на рисунку 11.

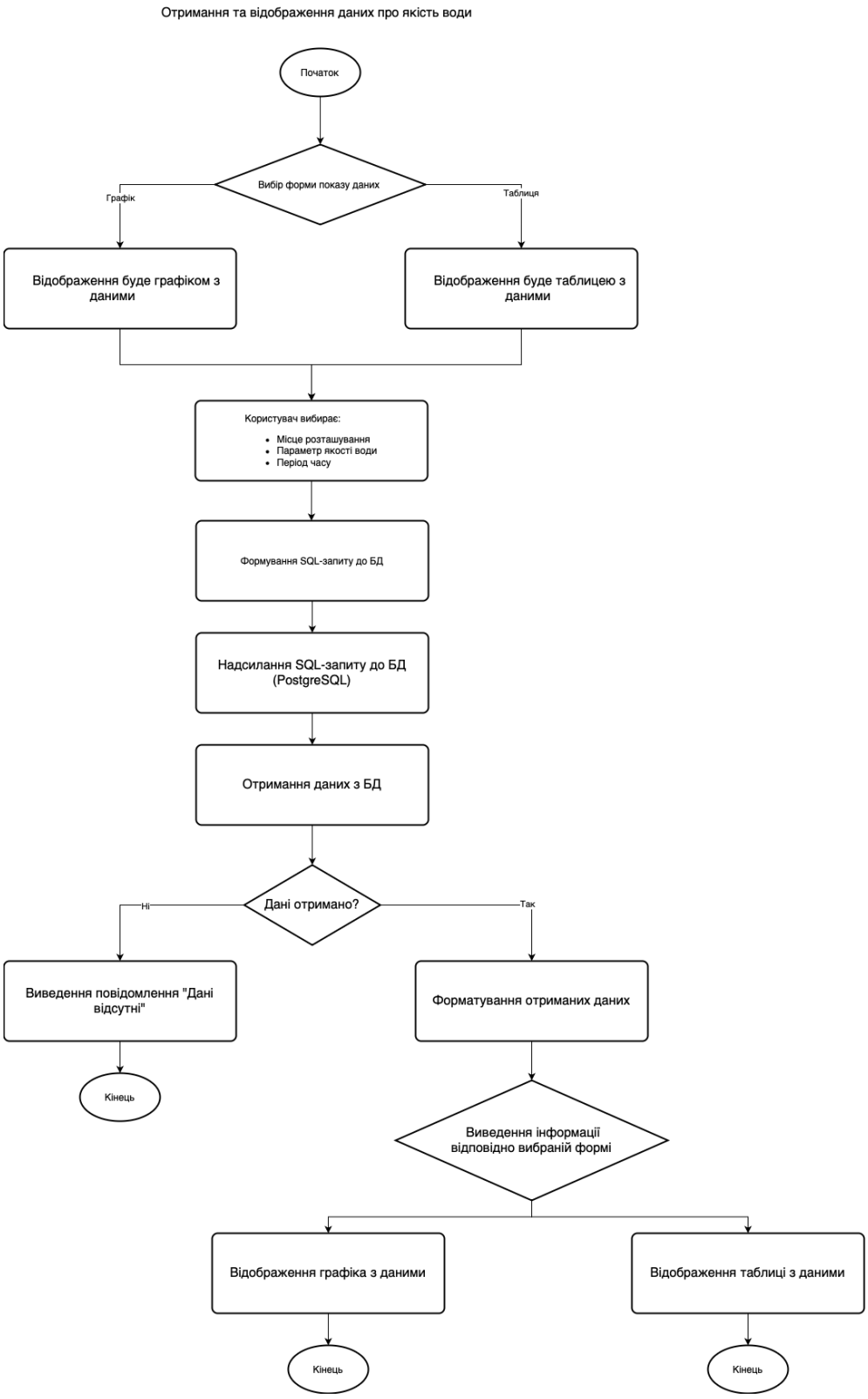

Рис.11 – Блок-схема алгоритму отримання та відображення даних про якість води

Ця блок-схема описує процес отримання та відображення даних про якість води в системі. Вона починається з вибору користувачем способу відображення даних (графік або таблиця) і завершується виведенням інформації.

- **Етапи процесу.**
  - **Початок:** процес починається.
  - **Вибір форми показу даних:** користувач обирає, в якому вигляді він хоче переглянути дані:
  - **Графік:** якщо обрано графік, відображення буде у вигляді графіка з даними.
  - **Таблиця:** якщо обрано таблицю, відображення буде у вигляді таблиці з даними.
  - **Користувач вибирає:** після вибору форми відображення, користувач уточнює критерії для отримання даних: місце розташування, параметр якості води, період часу.
  - **Формування SQL-запиту до БД:** на основі вибору користувача формується відповідний SQL-запит до бази даних.
  - **Надсилання SQL-запиту до БД (PostgreSQL):** сформований SQL-запит надсилається до бази даних PostgreSQL.
  - **Отримання даних з БД:** система намагається отримати дані з бази даних.
  - **Дані отримано?:** перевірка, чи були отримані дані з бази даних.

    Ні (Дані відсутні): якщо дані не отримано, виводиться повідомлення "Дані відсутні", і процес завершується.

    Так (Дані отримано): якщо дані отримано, процес переходить до наступного етапу.

- **Форматування отриманих даних:** отримані дані форматуються відповідно до обраної форми відображення (графік або таблиця).
- **Виведення інформації відповідно вибраній формі:** інформація виводиться користувачеві:
- **Відображення графіка з даними:** якщо на початку було обрано графік.
- **Відображення таблиці з даними:** якщо на початку було обрано таблицю.
- **Кінець:** процес завершується.

Ця блок-схема наочно демонструє потік операцій від ініціації користувачем запиту до відображення результатів, включаючи взаємодію з базою даних та обробку можливих сценаріїв відсутності даних.

Цьому алгоритму відповідає частина коду форми ViewTabularDataFrom.cs (рис.12).

```
private async void applyFiltersButton_Click(object sender, EventArgs e)
{
    string locationFilter = locationFilterComboBox.SelectedItem?.ToString();
    string parameterFilter = parameterFilterComboBox.SelectedItem?.ToString();
    DateTime startDate = startDateFilterPicker.Value.Date;
    DateTime endDate = endDateFilterPicker.Value.Date.AddDays(1).AddSeconds(-1);

    string sql = "SELECT wqd.time AS Час, " +
                 "wqd.value AS Значення, " +
                 "l.name AS Місцезнаходження, " +
                 "pu.name AS Параметр, " +
                 "pu.unit AS Одиниця " +
                 "FROM water_quality_data wqd " +
                 "JOIN locations l ON wqd.location_id = l.location_id " +
                 "JOIN parameter_units pu ON wqd.parameter_unit_id = pu.parameter_unit_id " +
                 "WHERE 1=1 ";

    if (locationFilter != "Всі" && !string.IsNullOrEmpty(locationFilter))
    {
        sql += "AND l.name = @location_name ";
    }

    if (parameterFilter != "Всі" && !string.IsNullOrEmpty(parameterFilter))
    {
        sql += "AND pu.name = @parameter_name ";
    }

    sql += "AND wqd.time >= @start_date AND wqd.time <= @end_date " +
           "ORDER BY wqd.time DESC;";

    try
    {
        using (var connection = new NpgsqlConnection(_dbHelper.ConnectionString))
        {
            await connection.OpenAsync();
            using (var command = new NpgsqlCommand(sql, connection))
            using (var adapter = new NpgsqlDataAdapter(command))
            {
                if (locationFilter != "Всі" && !string.IsNullOrEmpty(locationFilter))
                {
                    command.Parameters.AddWithValue("@location_name", locationFilter);
                }
                if (parameterFilter != "Всі" && !string.IsNullOrEmpty(parameterFilter))
                {
                    command.Parameters.AddWithValue("@parameter_name", parameterFilter);
                }
                command.Parameters.AddWithValue("@start_date", startDate);
                command.Parameters.AddWithValue("@end_date", endDate);

                var dataTable = new DataTable();
                adapter.Fill(dataTable);
                dataDataGridView.DataSource = dataTable;
            }
            Console.WriteLine("Дані відфільтровано.");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Помилка фільтрації даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        Console.WriteLine(ex);
    }
}
```

Рис. 12 – частина коду форми ViewTabularDataFrom.cs

Також була створена блок-схема алгоритму генерації комбінованого звіту (Excel та PDF) (рис.13).

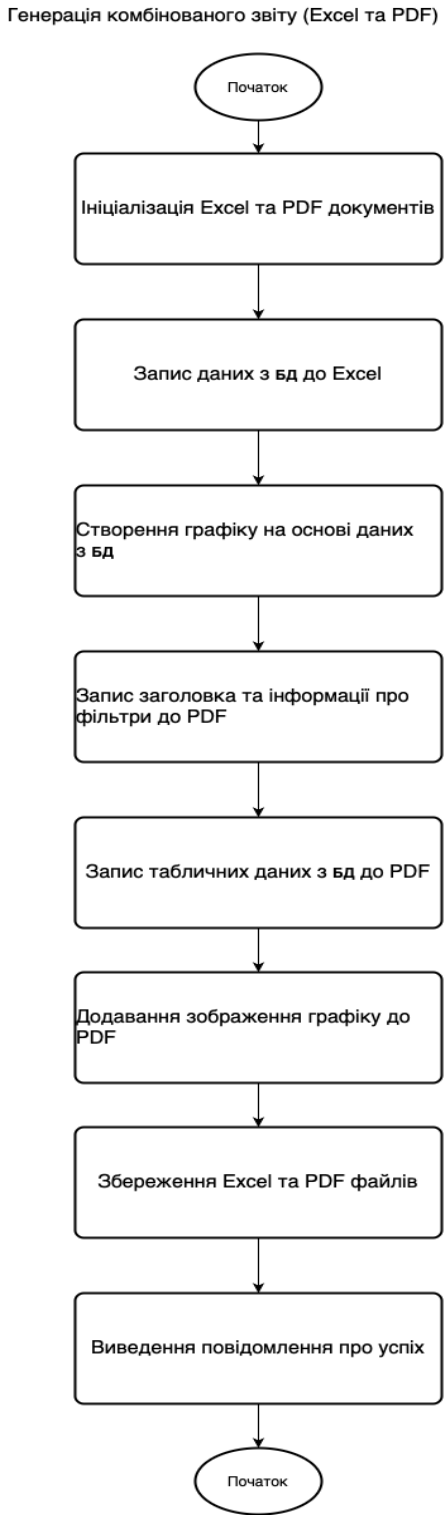

Рис.13 - Блок-схема алгоритму генерації комбінованого звіту (Excel та PDF)

Ця блок-схема описує процес генерації комбінованого звіту у форматах Excel та PDF. Вона деталізує кроки, які виконуються для створення звіту на основі даних про якість води.

- **Етапи процесу.**
  - **Початок:** процес генерації звіту починається.
  - **Ініціалізація Excel та PDF документів:** на цьому етапі створюються нові екземпляри документів Excel (за допомогою ExcelPackage) та PDF (за допомогою PdfWriter та PdfDocument), які будуть використовуватися для формування звіту.
  - **Запис даних з БД до Excel:** дані, отримані з бази даних, записуються в робочий аркуш Excel. Це включає заголовки стовпців та всі відповідні записи.
  - **Створення графіку на основі даних з БД:** на основі тих самих даних з бази даних створюється графік (за допомогою компонента Chart), який візуалізує тенденції або розподіл показників якості води.
  - **Запис заголовка та інформації про фільтри до PDF:** у PDF-документ додається заголовок звіту та інформація про застосовані фільтри (місце розташування, параметр, період часу), що забезпечує контекст для даних.
  - **Запис табличних даних з БД до PDF:** табличні дані, отримані з бази даних, форматуються та записуються в PDF-документ.
  - **Додавання зображення графіку до PDF:** зображення раніше створеного графіку вставляється в PDF-документ, доповнюючи текстову інформацію візуальним представленням даних.
  - **Збереження Excel та PDF файлів:** сформовані документи Excel та PDF зберігаються у файловій системі.

- **Виведення повідомлення про успіх:** користувачеві виводиться повідомлення про успішне створення та збереження звітів.
- **Кінець:** процес генерації комбінованого звіту завершується.

Ця блок-схема детально ілюструє послідовність дій, необхідних для створення комплексного звіту, що включає як табличні дані, так і графічне представлення, у двох різних форматах.

Цьому алгоритму відповідає частина коду форми ReportsFrom.cs (рис.14).

```
using (var memoryStream = new MemoryStream())
{
    chart.SaveImage(memoryStream, ImageFormat.Png);
    byte[] chartImage = memoryStream.ToArray();

    using (var ms = new MemoryStream())
    using (var writer = new PdfWriter(ms))
    using (var pdfDocument = new PdfDocument(writer))
    using (var document = new Document(pdfDocument))
    {
        PdfFont fontBold = PdfFontFactory.CreateFont(iText.IO.Font.Constants.StandardFonts.HELVETICA_BOLD);
        Paragraph title = new Paragraph("Звіт про якість
води").SetFont(fontBold).SetFontSize(16).SetTextAlignment(TextAlignment.CENTER);
        document.Add(title);

        document.Add(new Paragraph($"Місцезнаходження: {locationName}"));
        document.Add(new Paragraph($"Параметр: {parameterName}"));
        document.Add(new Paragraph($"Період: з {startDate:d} по {endDate:d}"));
        document.Add(new Paragraph(" "));

        iText.Layout.Element.Table pdfTable = new iText.Layout.Element.Table(dataTable.Columns.Count);
        pdfTable.SetWidth(UnitValue.CreatePercentValue(100));
        foreach (DataColumn column in dataTable.Columns)
        {
            pdfTable.AddHeaderCell(new Paragraph(column.ColumnName).SetFont(fontBold));
        }
        foreach (DataRow row in dataTable.Rows)
        {
            foreach (object cell in row.ItemArray)
            {
                string cellValue = cell.ToString();
                if (dataTable.Columns[dataTable.Columns.IndexOf("time")].ColumnName.ToLower() == "time" &&
DateTime.TryParse(cellValue, out DateTime dateTimeValue))
                {
                    cellValue = dateTimeValue.ToString("yyyy-MM-dd HH:mm:ss"); // Формат для PDF
                }
                pdfTable.AddCell(new Paragraph(cellValue));
            }
        }
        document.Add(pdfTable);
        document.Add(new Paragraph(" "));

        ImageData imageData = ImageDataFactory.Create(chartImage);
        iText.Layout.Element.Image image = new iText.Layout.Element.Image(imageData).ScaleToFit(700, 500); // Збільшена діаграма
        document.Add(image);

        document.Close();
        byte[] pdfData = ms.ToArray();

        File.WriteAllBytes("WaterQualityReport.xlsx", excelData);
        File.WriteAllBytes("WaterQualityReport.pdf", pdfData);

        MessageBox.Show("Звіт успішно створено та збережено як WaterQualityReport.xlsx та WaterQualityReport.pdf", "Успіх",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
```

Рис. 14 - частина коду форми ReportsFrom.cs

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Головне вікно системи моніторингу якості води показана на рисунку 15.

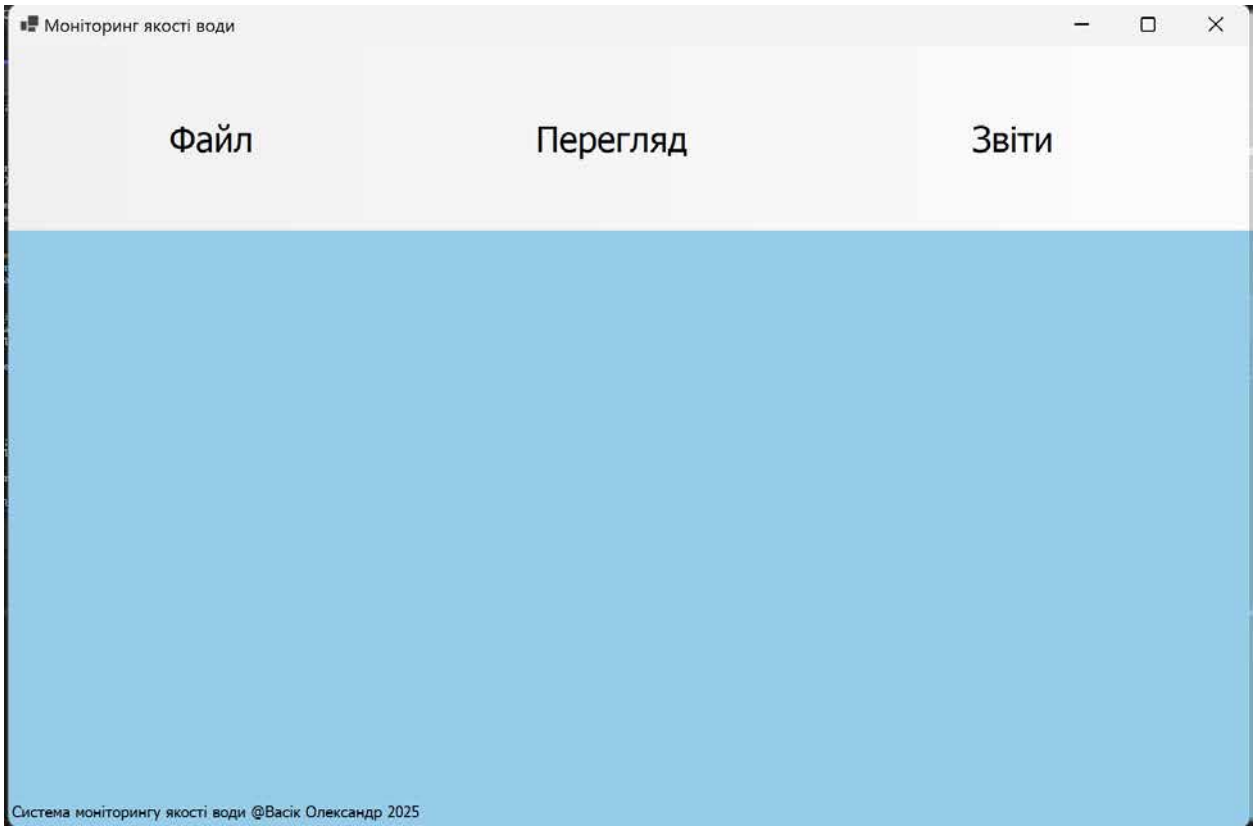

Рис. 15 – Головне вікно

Зверху цього вікна бачимо меню де є пункти: “Файл, Перегляд, Звіти”. Це меню вибору звідки можна перейти на усі вікна програми. Далі розберемо усі вікна програми та її функціонал. На рисунку 16 показано вікно форми звідки можна додати до бази даних вимірювання за такої необхідності.

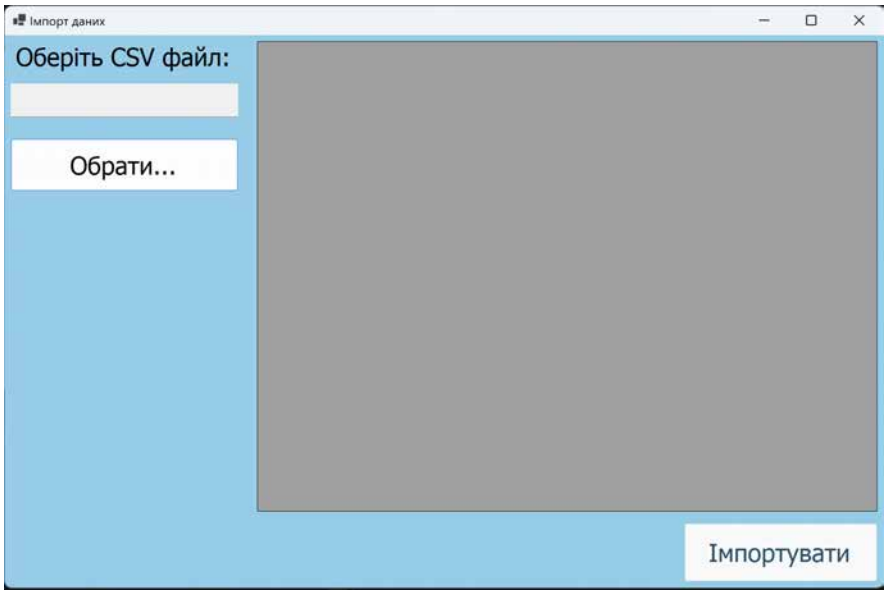

Рис. 16 – Вікно форми імпорту даних

Для того щоб обрати файл треба натиснути кнопку обрати та вибрати необхідний файл. Після вибору файлу інформація з файлу з’явиться справа в форматі таблиці. Це показано на рисунку 17.

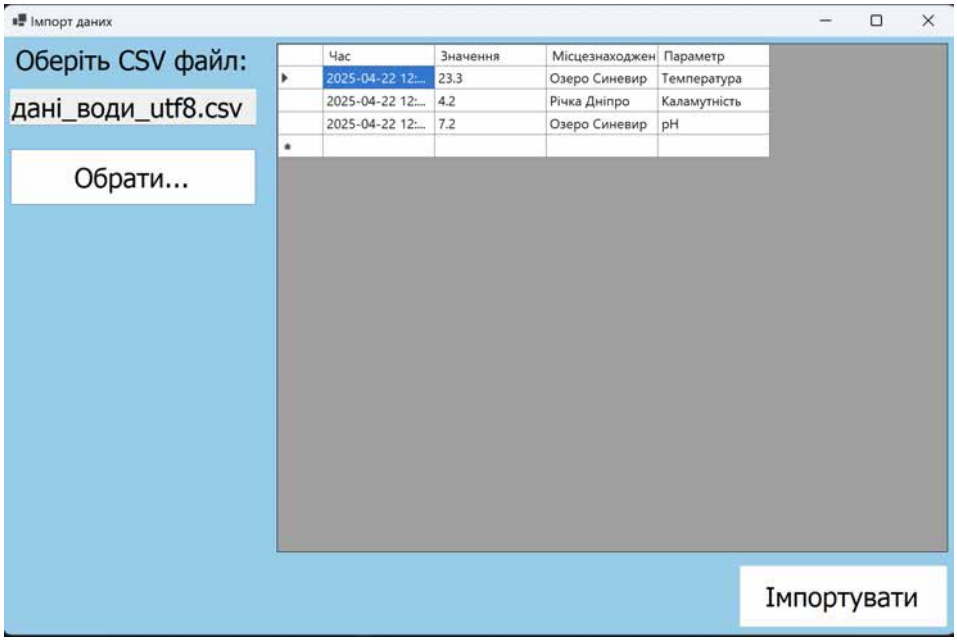

Рис. 17 – Файл для імпорту обрано і інформація з нього з’явилась у табличному вигляді

Далі необхідно натиснути на “Імпортувати”, після цього програма сповістить або про успішне додавання до бази даних або про помилку і виведе причину помилки. Успішна спроба додавання даних показана на рисунку 18.

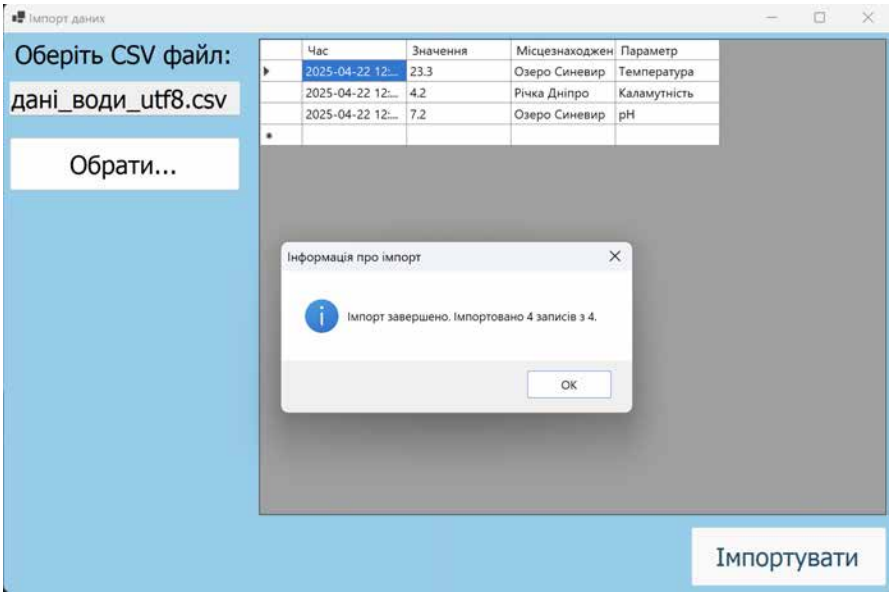

Рис.18 – Успішна спроба імпорту даних до бази даних.

Далі розберемо вікно форми для показу даних у вигляді таблиці. При відкритті цього вікна одразу виводяться вимірювання які зберігаються в базі даних. Це показано на рисунку 19.

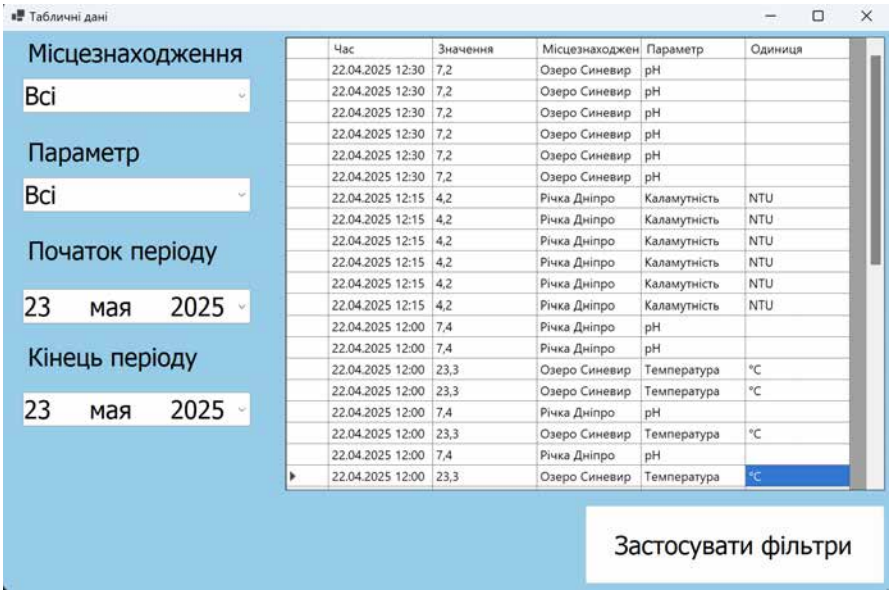

Рис. 19 – Виведення вимірювань в табличному виді

При необхідності фільтрування даних треба вибрати бажані фільтри для пошуку, такі як “Місцезнаходження”, “Параметр”, “Початок періоду”, “Кінець періоду”. Після цього натиснути кнопку “Застосувати фільтри” і таблиця оновиться. Зміна таблиць відповідно фільтрів показана на рисунку 20.

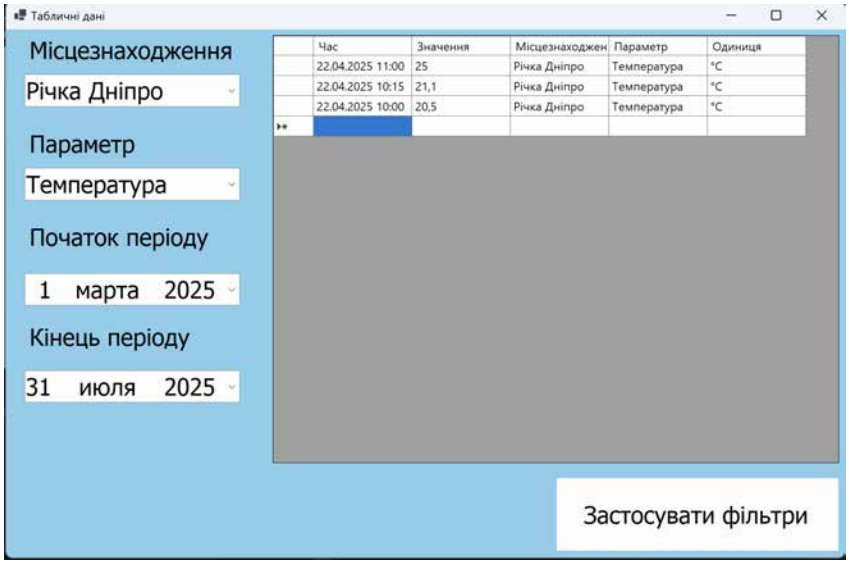

Рис. 20 – Виведення відфільтрованих даних таблиці

Також в системі є вікно показу даних у форматі графіку. Алгоритм дій там такий самий як і з табличним варіантом. Створення графіку по заданим параметрам показано на рисунку 21.

Рис. 21- Створення графіку по заданим параметрам

Останнє вікно системи моніторингу якості води це створення та завантаження комбінованого звіту у форматі Excel та PDF. Для створення комбінованого звіту необхідно задати параметри так само як і у вікнах показу даних у форматі таблиці та показу даних у форматі графіка. Після цього натиснути “Згенерувати звіт”. Одразу як створяться звіти вони будуть збережені на комп’ютер і з’явиться повідомлення про їх успішне створення. Саме вікно зі звітами та повідомлення про успішне створення показано на рисунку 22.

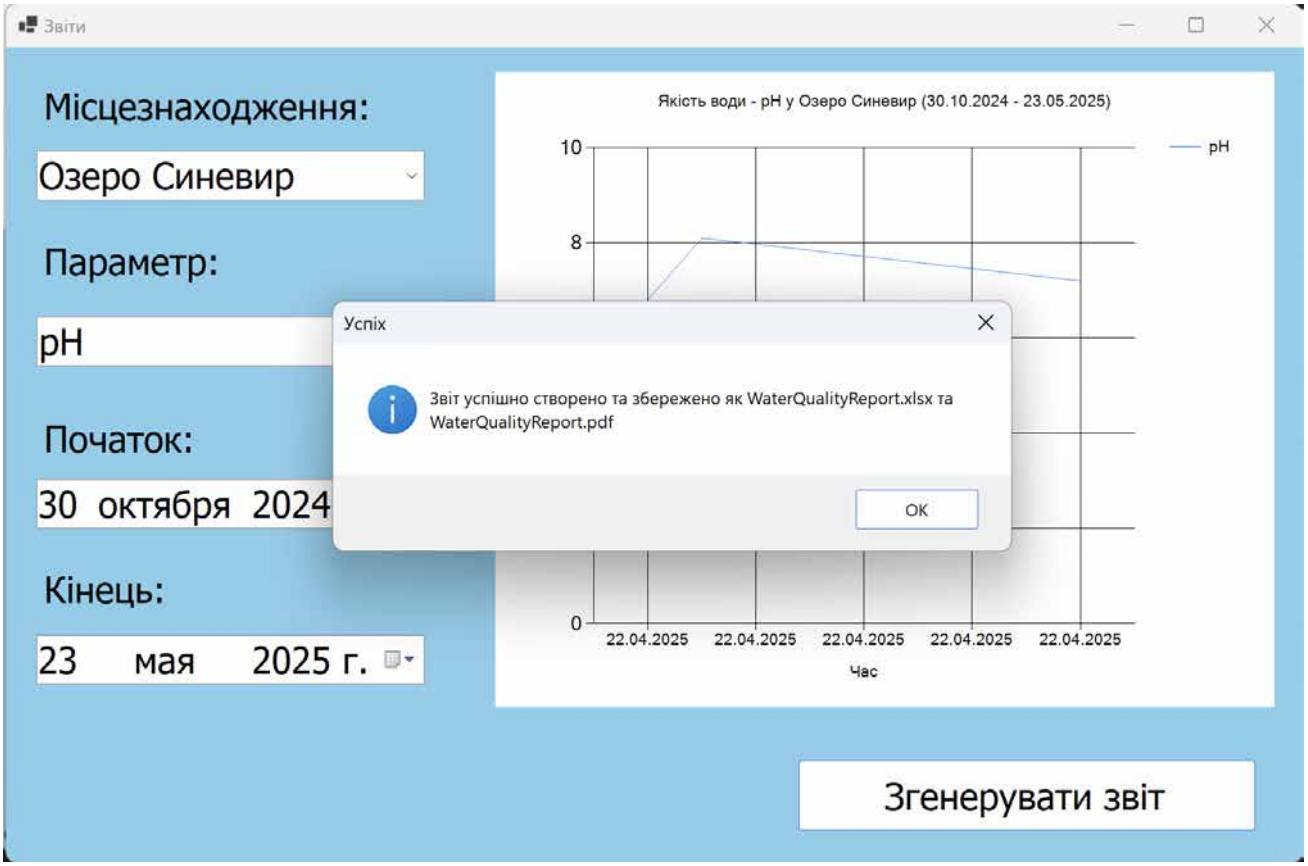

Рис. 22 – Повідомлення про успішне створення звітів у форматах Excel та PDF

Створенні звіти показані на рисунках 23-24.

:

: pH

: 30.10.2024 23.05.2025

time	value	location_name	parameter_name
2025-04-22 10:30:00	2025-08-06 00:00:00		pH
2025-04-22 10:45:00	2025-01-08 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH
2025-04-22 12:30:00	2025-02-07 00:00:00		pH

Рис.23 – Звіт у форматі PDF

	A	B	C	D	E
1	time	value	location_name	parameter_name	
2	2025-04-22 10:30:00	6,8	Озеро Синевир	pH	
3	2025-04-22 10:45:00	8,1	Озеро Синевир	pH	
4	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
5	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
6	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
7	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
8	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
9	2025-04-22 12:30:00	7,2	Озеро Синевир	pH	
10					
11					

Рис.24 – Звіт у форматі Excel

Підсумок.

Проведене тестування системи моніторингу якості води підтвердило її функціональність та відповідність поставленим вимогам. Користувацький

інтерфейс, реалізований у середовищі Windows Forms, забезпечує зрозумілу навігацію, що дозволяє користувачам ефективно взаємодіяти з програмою.

Функціонал імпорту даних з файлів до бази даних, візуалізація даних у табличному вигляді, а також можливість фільтрації за різними параметрами, такими як місцезнаходження, тип параметра та часовий період, працює стабільно та без збоїв. Графічне відображення даних дозволяє наочно аналізувати зміни показників якості води у часі, що є важливим для своєчасного виявлення потенційних проблем.

Модуль генерації звітів у форматах PDF та Excel функціонує коректно, забезпечуючи збереження відфільтрованих даних у зручному для подальшого аналізу вигляді. Це сприяє ефективному документуванню та обміну інформацією між зацікавленими сторонами.

Загалом, система демонструє високу надійність та зручність у використанні, що робить її ефективним інструментом для моніторингу якості води.

4.2 Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної та ефективної роботи системи моніторингу якості води необхідно дотримуватися певних вимог до апаратного та програмного забезпечення. Ці вимоги враховують особливості розробки на платформі .NET з використанням PostgreSQL як бази даних.

4.2.1 Апаратні вимоги. Апаратні вимоги до робочої станції, на якій буде функціонувати прикладне програмне забезпечення, є помірними, оскільки більшість обчислювальних операцій, пов'язаних з базою даних, виконуються сервером PostgreSQL.

Процесор (CPU): двоядерний процесор з тактовою частотою 2.0 ГГц або вище. Рекомендується Intel Core i3 або аналогічний AMD.

Оперативна пам'ять (RAM): мінімум 4 ГБ, рекомендовано 8 ГБ або більше для комфортної роботи, особливо при обробці великих обсягів даних або генерації складних звітів.

Накопичувач: мінімум 100 МБ вільного місця на жорсткому диску (HDD) або твердотільному накопичувачі (SSD) для встановлення програми. Додаткове місце знадобиться для зберігання звітів та імпортованих даних. Рекомендується SSD для швидшого завантаження програми та доступу до файлів.

Дисплей: монітор з роздільною здатністю не менше 1280x720 пікселів.

Мережеве підключення: стабільне мережеве підключення (LAN або Wi-Fi) для доступу до сервера бази даних PostgreSQL, якщо він розташований на віддаленому комп'ютері.

4.2.2 Програмні вимоги. Програмне забезпечення, необхідне для коректної роботи системи, включає операційну систему, середовище виконання та компоненти для взаємодії з базою даних.

Операційна система.

Microsoft Windows 7 Service Pack 1 або новіша версія (включаючи Windows 8, Windows 10, Windows 11). Рекомендується використовувати актуальні версії Windows для забезпечення безпеки та сумісності.

Середовище виконання.

.NET Framework 4.7.2 або новіша версія. Система розроблена на базі .NET Framework, тому його наявність є обов'язковою.

Система управління базами даних (СУБД).

PostgreSQL 9.x або новіша версія. Сервер бази даних PostgreSQL повинен бути встановлений та налаштований (локально або на віддаленому сервері) для зберігання та обробки даних моніторингу.

Драйвер бази даних.

Npgsql (входить до складу програми). Цей .NET-провайдер для PostgreSQL забезпечує зв'язок між прикладним програмним забезпеченням та базою даних.

Додаткові бібліотеки (входять до складу програми).

OfficeOpenXml (EPPlus): Використовується для генерації звітів у форматі Excel.

iText7: Використовується для генерації звітів у форматі PDF.

Дотримання цих вимог дозволить системі моніторингу якості води функціонувати правильно та стабільно, забезпечуючи надійне отримання, обробку та відображення даних.

4.3 Склад інсталяційного пакету

Інсталяційний пакет прикладного програмного забезпечення системи моніторингу якості води буде містити всі необхідні компоненти для розгортання та коректної роботи програми на кінцевій робочій станції. Його склад передбачає мінімальні дії з боку користувача для встановлення.

До складу інсталяційного пакету входитимуть:

виконуваний файл програми (.exe) - основний виконуваний файл системи моніторингу якості води.

бібліотеки .NET Framework - необхідні бібліотеки, які забезпечують роботу програми, якщо вони не присутні в системі користувача або потребують оновлення.

бібліотеки для взаємодії з PostgreSQL (Npgsql) - всі необхідні DLL-файли для коректної роботи з базою даних PostgreSQL.

бібліотеки для генерації звітів (OfficeOpenXml, iText7) - всі необхідні DLL-файли для функціоналу експорту даних у формати Excel та PDF.

ВИСНОВКИ

Дана дипломна робота присвячена розробці та реалізації прикладного програмного забезпечення для **системи моніторингу якості води**. Основною метою проєкту було створення інструментарію, здатного забезпечити ефективний збір, зберігання, аналіз та візуалізацію даних, що є основою для будь-якої системи моніторингу якості води.

В процесі виконання дипломної роботи було досягнуто наступних ключових результатів.

Проведено комплексний аналіз предметної області та вимог: Дослідження специфіки параметрів якості води, існуючих методологій моніторингу та функціональних потреб цільових користувачів дозволило сформувати набір функціональних та нефункціональних вимог до розроблюваної системи.

Розроблено архітектуру програмного забезпечення: Згідно з принципами сучасного програмного проєктування, включаючи модульність та об'єктно-орієнтований підхід, було спроєктовано архітектуру системи. Це було показано у розроблених UML-діаграмах (класів, кооперації, пакетів, компонентів) та блок-схемах алгоритмів.

Здійснено обґрунтований вибір та ефективне застосування технологічного стеку: Для реалізації системи були обрані надійні та відкриті технології. Використання мови програмування **C#** та платформи **.NET** забезпечило високу продуктивність та стабільність додатку. **PostgreSQL** була обрана як система управління базами даних, що гарантує ефективне та безпечне зберігання значних обсягів моніторингових даних. Інтеграція бібліотек, таких як **Npgsql** для взаємодії з базою даних, **OfficeOpenXml**

(EPPlus) та **iText7** для генерації звітів у форматах Excel та PDF відповідно, дозволила реалізувати функціонал системи.

Результатом виконаної дипломної роботи є прикладне програмне забезпечення, яке демонструє такі можливості.

Надійне зберігання та управління даними: забезпечено централізоване та структуроване зберігання даних про якість води, що гарантує їх цілісність та доступність.

Інтуїтивно зрозумілий користувацький інтерфейс: Розроблений на базі Windows Forms інтерфейс надає користувачам зручні засоби для доступу до даних, їх фільтрації, а також ініціації відображення та генерації звітів.

Гнучкі інструменти візуалізації та звітності: Реалізована можливість представлення даних у вигляді таблиць та графіків, а також генерація детальних звітів, спрощує аналіз та звітність інформації про дані вимірювання.

Створена система моніторингу якості води є завершеним програмним продуктом, що володіє потенціалом для підвищення ефективності екологічного моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. United Nations. (2022). *The Sustainable Development Goals Report 2022* — [Електронний ресурс]. — Режим доступу: https://unstats.un.org/sdgs/report/2022/ (дата звернення: 09.05.2025).
2. World Health Organization. (2017). *Guidelines for drinking-water quality: fourth edition incorporating the first addendum* — [Електронний ресурс]. — Режим доступу: https://www.who.int/publications/i/item/9789241549950 (дата звернення: 09.05.2025).
3. Chapman, D. V. (Ed.). (1996). *Water quality assessments: A guide to the use of biota, sediments and water in environmental monitoring*. CRC Press.
4. Cardoso, M. R., Ferreira, E. C., Fonseca, J. P., & Martins, R. C. (2018). Water quality monitoring using IoT: A survey on architectures, technologies, and applications. *Sensors, 18*(3), 689.
5. Ray, P. P. (2016). A survey on Internet of Things architectures. *Journal of King Saud University-Computer and Information Sciences, 28*(3), 287–318.
6. Балакін, В. М. (2010). *Екологічний моніторинг*. К.: Либідь.
7. Aquarius (by Aquatic Informatics) — [Електронний ресурс]. — Режим доступу: https://aquaticinformatics.com/products/aquarius-environmental-water-data-management/ (дата звернення: 09.05.2025).
8. WISKI (by Kisters) — [Електронний ресурс]. — Режим доступу: https://www.kisters.es/app/uploads/2023/04/230411-wiski-overview-en.pdf (дата звернення: 09.05.2025).

9. Envirosuite — [Електронний ресурс]. — Режим доступу: https://envirosuite.com (дата звернення: 09.05.2025).
10. Open Waters — [Електронний ресурс]. — Режим доступу: https://github.com/open-environment/open-waters (дата звернення: 09.05.2025).
11. UML to model pollution patterns of the Smart Cities for Metropolitan Zone of Guadalajara — [Електронний ресурс]. — Режим доступу: https://www.redalyc.org/journal/5122/512269058004/html/ (дата звернення: 09.05.2025).
12. An Integration Method of UML Diagrams and Arduino UNO for Developing Water Pollution Detection System — [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/383451676_An_Integration_Method_of_UML_Diagrams_and_Arduino_UNO_for_Developing_Water_Pollution_Detection_System (дата звернення: 09.05.2025).
13. Unified Modeling Language (UML) Diagrams | GeeksforGeeks. (2025) — [Електронний ресурс]. — Режим доступу: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата звернення: 09.05.2025).
14. Visual Paradigm. (n.d.). *What is Package Diagram?* — [Електронний ресурс]. — Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/ (дата звернення: 09.05.2025).
15. FoxmindEd. (2024). *UML діаграми, їхні основні типи та процес розроблення* — [Електронний ресурс]. — Режим доступу: https://foxminded.ua/uml-diagramy/ (дата звернення: 09.05.2025).

16. Microsoft Support. (n.d.). *Створення схеми компонентів UML* — [Електронний ресурс]. — Режим доступу: https://support.microsoft.com/uk-ua/topic/створення-схеми-компонентів-uml-aa924ecb-e4d2-4172-976e-a78fa157b074 (дата звернення: 09.05.2025).
17. PostgreSQL Documentation. (2025). *CREATE TABLE* — [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/docs/current/sql-createtable.html (дата звернення: 09.05.2025).
18. Modular Programming and Increasing Need for Secure Software Development — [Електронний ресурс]. — Режим доступу: https://lazarusalliance.com/modular-programming-and-increasing-need-for-secure-software-development/ (дата звернення: 22.05.2025).

ДОДАТОК А

Створення таблиць

```
CREATE TABLE IF NOT EXISTS public.parameter_units
(
    parameter_unit_id integer NOT NULL DEFAULT nextval('parameter_units_parameter_unit_id_seq'::regclass),
    name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    unit character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT parameter_units_pkey PRIMARY KEY (parameter_unit_id),
    CONSTRAINT parameter_units_name_key UNIQUE (name)
)
CREATE TABLE IF NOT EXISTS public.locations
(
    location_id integer NOT NULL DEFAULT nextval('locations_location_id_seq'::regclass),
    name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    latitude double precision,
    longitude double precision,
    CONSTRAINT locations_pkey PRIMARY KEY (location_id),
    CONSTRAINT locations_name_key UNIQUE (name)
)
```

ДОДАТОК Б

Діаграма класів