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 агрохіманалізу

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

___к.т.н., доцент_____ _____ ____Вайганг Г.О.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи :_____/ __ Боровик В.І. /

Консультант бакалаврської кваліфікаційної роботи

___к.т.н, доцент _____ _____ _ Пархоменко І.І.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ ___Віюк В.Ю._____
(підпис) (ПІБ студента)

КИЇВ – 2025

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.

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

_____Віюку Владиславу Юрійовичу_____
(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: Програмне забезпечення системи моніторингу агрохіманалізу

Затверджена наказом ректора НУБіП України від "16" Грудня 2024 р. №2249 "С"

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

_____

Перелік питань, які потрібно розробити:

опис предметної області, аналіз вимог, моделювання системи, огляд рішень, постановка завдання, проєктування даних і архітектури, розробка бази даних, вибір інструментів, програмування модулів, тестування, впровадження, вимоги до ресурсів

Перелік графічних документів (за потреби)

_____

Дата видачі завдання "_____"_____20___ р.
Керівник бакалаврської кваліфікаційної роботи

_____ _____ ____Боровик В.І.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ ___Віюк В.Ю_____
(підпис) (ПІБ студента)

Зміст

Вступ

Зараз сучасні сільські господарства активно впроваджують технології, серед цих технологій особливе місце займає інформаційні системи моніторингу стану полів. В теперешніх умовах зростання вимог до ефективного агробиробництва, виникає потреба у розробці програмного забезпечення, для оперативного контролю за агрохімічними параметрами, аналізування даних з датчиків, та створення рекомендацій для агрохімічних рішень.

Метою дипломного проєкту є розробка системи моніторингу агрохіманалізу, яке буде забезпечувати збір даних з датчиків, їхнє збереження та обробку. Основні завдання які були вирішені в цьому преєкті, це:

- Створення зручного користувацького інтерфейсу;
- Зв’язок з хмарною базою даних Firebase;
- Реалізацію форматування планів добрив на основі датчиків;
- Генерування звітів;

Дана тема дипломного проєкту є актуальною, оскільки вона дозволяє автоматизувати важливі процеси, зменшити вплив людського фактору та підвищити продуктивність в аграрній галузі.

1 Системний аналіз предметної області

1.1 Опис предметної області

На сьогоднішній день сільське господарство потребує точного аналізу показників ґрунту та контролю до управління ресурсами. Агрохіманаліз - це лабораторне дослідження ґрунту для визначення вмісту параметрів що впливають на його врожайність та якість.

Після агрохіманалізу агрономи приймають рішення щодо меліорації, удобрення та інших агрохімічних заходів.

Зазвичай це паперовий або фрагментарно автоматизований процес, в якому є недоліки:

- Час витрачений для аналізу інформації
- Помилки при роботі з даними
- Неповний контроль над динамікою змін

Для вирішення цих проблем виникає потреба у інформаційній системі, яка допоможе вирішити:

- Збирання та зберігання даних, та їх аналізування
- Відслідковувати стан ґрунту
- Формувати звіти на основі аналізу та даних

Програмне забезпечення системи моніторингу агрохіманалізу орієнтується на використанням його господарствами, фахівцями, та іншими зацікавленими особами.

В даній предметній області основними об’єктами є:

- Поля
- Датчики
- Плани добрива
- Звіти
- Користувачи системи

Тому дана предметна область спрямована на збір інформації, її зберігання та аналіз, щоби покращити управління земельними ділянками для господарств.

1.2 Аналіз вимог до програмної системи

1.2.1 Загальні положення

Метою створення програмного забезпечення слугує збір, збереження даних після агрохімічного аналізу ґрунту.
Дана система має забезпечувати зручний та зрозумілий інтерфейс для користувача, підтримка кількох користувачів, а також підтримувати можливість роботи з великим обсягом даних.

1.2.2 Функціональні вимоги

Управління полями

- Можливість створювати поля та подальше їхнє редагування інформації(Нумерація поля, тип ґрунту на полі, датчики котрі знаходяться на полі та його площа) також можливість видалення поля;

Робота з датчиками на полі

- Створення датчика, його редагування та видалення;
- Прив’язка датчиків до конкретного поля;

Планування добрив

- Автоматичне формування плану за заданими нормативами;

Звітність

- Генерація звітів по конкретному полі;

Система користувачив

- Авторизація користувача;
- Відмінність прав доступу (агроном, адміністратор);

1.2.2 Нефункціональні вимоги

- Надійність та доступність;
- Дані зберігаються динамічно, в хмарі;
- Продуктивність;
- Швидкий відгук інтерфейсу системи;
- Підтримка декількох одночасно активних користувачів;
- Зручність системи;
- Інтуїтивно зрозумілий інтерфейс для користувачів без технічної освіти;

1.2.3 Обмеження

- Система створюється для роботи в середовищі Windows;
- Дані зберігаються у базі даних Firebase;
- Обов’язкове підключення до інтернету;

1.3 Моделювання предметної області

Для структури предметної області системи моніторингу агрохіманалізу використовується об’єктно-орієнтований підхід, завдяки якому можна чітко описати основні сутності, характеристики та зв’язки у системі. Для моделювання використовується мова UML (Unified Modeling Language - Уніфікована Мова Моделювання).

1.3.1 Основні сутності предметної області

- Поле
  - ІД - Унікальний ідентифікатор;
  - Номер поля - Номер або назва поля;
  - Тип ґрунту - Класифікація типу ґрунту;
  - Площа - Площа поля, гектари;
  - Датчики - Датчики котрі прив’язані до поля;

- Датчики
  - ІД - Унікальний ідентифікатор;
  - Тип датчику - Класифікація типу датчика;
  - Поле - Поле на якому знаходиться датчик;

- План добрив
  - ІД - Унікальний ідентифікатор;
  - Поле - Поле для якого створений план;
  - Датчики - Датчики котрі прив’язані до поля;
  - Тип ґрунту - Класифікація типу ґрунту;
  - План добрива - Рекомендації щодо добавки добрив;

- Звітність
  - ІД - Унікальний ідентифікатор;
  - Номер поля - Номер або назва поля;
  - Датчики - Датчики котрі прив’язані до поля;
  - Тип ґрунту - Класифікація типу ґрунту;
  - Площа - Площа поля, гектари;
  - План добрива - Плани для конкретного поля;
  - Результати - Результати датчиків;

- Користувачі
  - Логін та пароль - дані авторизації;
  - ПІБ - ПІБ користувача;
  - Роль - Роль користувача;
  - Пошта - Електрона пошта користувача;
  - Телефон - Номер телефону користувача;

1.3.2 Діаграма

На даних діаграмах класів зображено зв’язки та сутності:

- Користувач - Звіт:
  - Тип - асоціація ;
  - Зв’язок - один до багатьох;

- Агрохімічний датчик - Звіт:
  - Тип - асоціація ;
  - Зв’язок - один до багатьох;

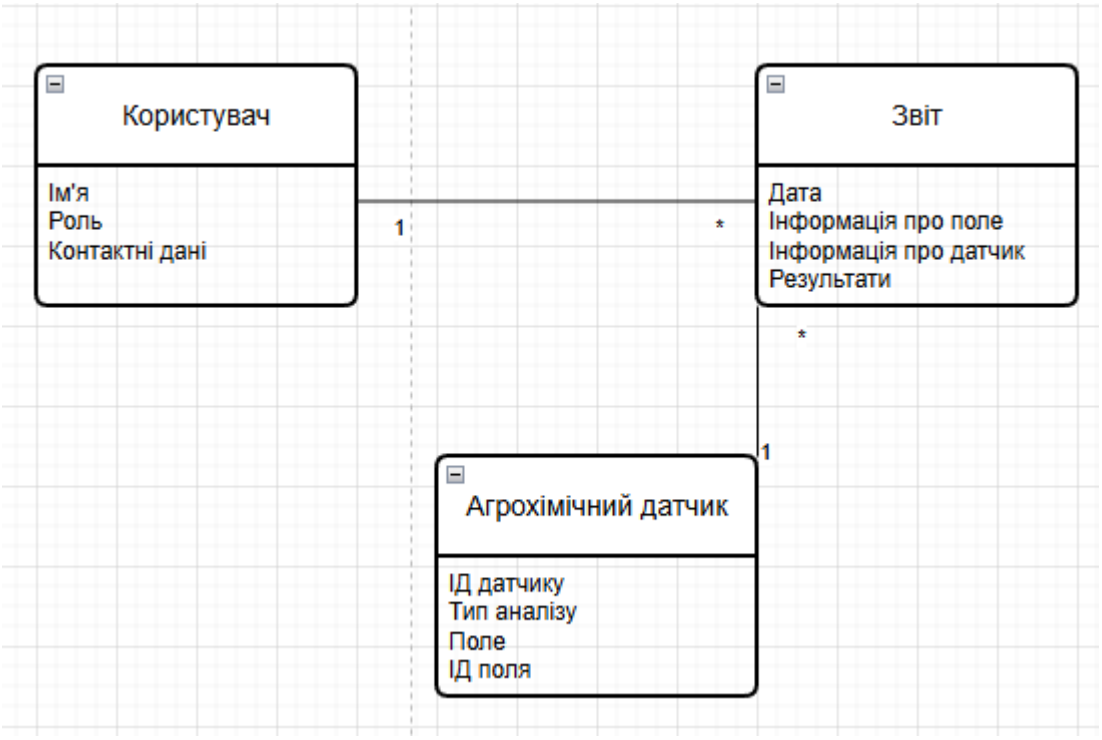

Рис 1.1 Діаграма зв’язків між користувачем, звітами та агрохімічними датчиками.

- Агрохімічний датчик - Звіт:
  - Тип - асоціація;
  - Зв’язок - один до багатьох;

- План добрив - Агрохімічний датчик:
  - Тип - залежність;
  - Зв’язок - один до одного;

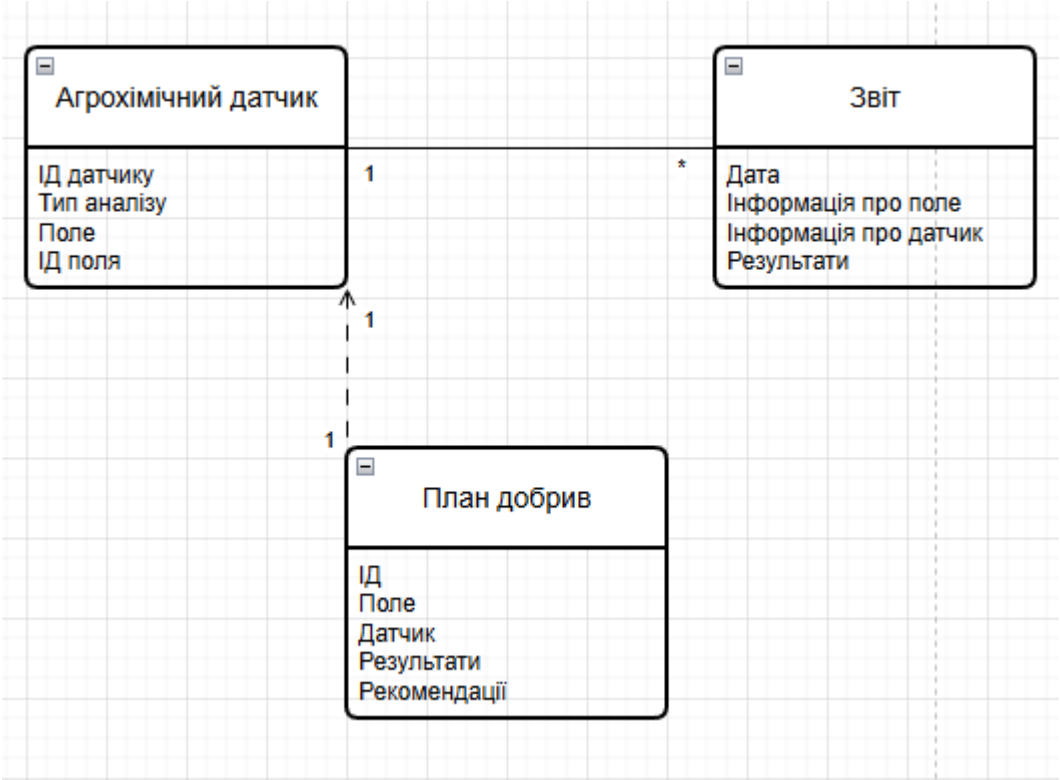

Рис 1.2 Діаграма зв’язків між планом добрив, звітами та агрохімічними датчиками

- Мапа полів - Агрохімічний датчик:

- Тип - асоціація;
- Зв’язок - один до багатьох;

- План добрив - Мапа полів:
  - Тип - залежність;
  - Зв’язок - один до одного;

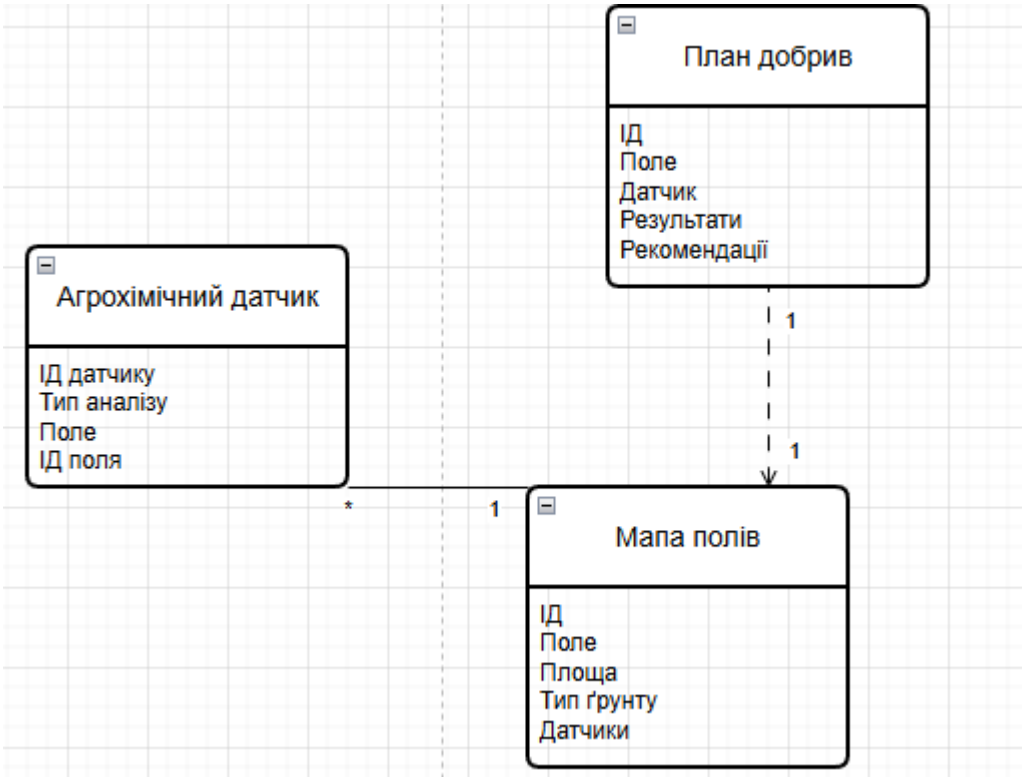

Рис 1. 3 Діаграма зв’язків між мапою полів, планом добрив та агрохімічними датчиками.

Завдяки моделювання предметної області можна структурувати знання про систему моніторингу агрохіманалізу, також можна визначити основні об’єкти, котрі в подальшому будуть реалізовані у системі. Дана модель буде слугувати базою для проектування бази даних у системи моніторингу агрохіманалізу.

1.4 Огляд інформаційних джерел та існуючих рішень

Було проведено аналіз наукових джерел, та документів та існуючих програмних продуктів, які повністю або частково реалізують подібний функціонал, для обґрунтування доцільності створення даного програмного продукту.

1.4.1Аналіз наукових джерел

У фаховій та науковій літературі розглядають питання автоматизації аграрного виробництва, створення систем точного аналізу складу ґрунтів:

- Системи точного землеробства - використовують дрони, GPS та датчики для оптимізації процесів (Гудзь С.П. 2020)
- Методи агрохімічного аналізу ґрунту - були описані в роботах (О.О. Кучми). Там зазначено про необхідність регулярно моніторити вміст NPK, pH для прийняття рішень удобрення.
- Інформаційні системи в аграрній галузі - потрібні для зменшення людського фактору.(ІТ в агробізнесі, Ігнатенко Ю. 2021)

1.4.2 Існуючи аналогі

FieldView

Одна з відоміших комерційних платформ для землеробства. Вона надає таку можливість: Моніторинг полів, моніторинг складу в ґрунті, моніторинг погодних умов та надає рекомендації щодо обробки полів. ЇЇ основний недолік у високій вартості.

Soft.Farm

Це Український продукт який має елементи агромоніторингу. Його функціонал включає агроскаутинг, облік полів, зв’язок з датчиками, проте немає повноцінного агрохімічного модуля.

Можна прийти до висновку що більшість із існуючих продуктів не відповідають потребам малих та середніх господарств України, або є дорогими та складними у використанні.

Цим можна обґрунтувати створення програмного забезпечення, адаптованого під Українські господарства.

1.5 Постановка завдання

Мета роботи

Метою даної дипломної роботи є розробка програмного забезпечення, яке буде здійснювати моніторинг стану ґрунту господарств. Завданням продукту буде збирання та збереження даних про агрохімічний стан ґрунту, та його подальший аналіз.

Дослідження завдання

- Аналіз предметної області. Пошук процесів та об’єктів агрохімічного моніторингу;
- Аналіз функціональних та не функціональних вимог до програмного забезпечення;
- Моделювання предметної області. Використання UML-діаграм для предметної області;
- Проектування системи. Визначення способів взаємодії між компонентами в системі;
- Створення бази даних;
- Створення інтерфейсу. Інтерфейс з підтримкою базових операції.
- Розробка функціоналу. Для створення звітів та агрохімічних показників;
- Тестування. Тестування програмного продукту на наявність не відповідності до вимог;

Вхідні дані

- Дані про поля: Ід, номер, площа, тип ґрунту, датчики;
- Результати аналізу: вміст NPK, pH тощо;
- Дані з датчиків: вологість, температура ґрунту тощо;
- Інформація про користувачів системи;

Очікувані результати

- Система має виконувати:
- Облік полів та їхні характеристики;
- Зберігання показників;
- Зручний інтерфейс;
- Формування звітів;
- Зв’язок з базою даних;

2 Проектування інформаційного та програмного забезпечення

2.1 Логічна модель даних у вигляді ER-діаграми

Користувачи

- ІД користувача (первинний ключ);
- ПІБ;
- Роль користувача;
- Контактні дані;

Звіт

- ІД звіту (первинний ключ);
- ІД користувача (зв’язок з користувачем);
- ІД поля (зв’язок з полем);
- ІД датчику (зв’язок з датчиком);
- ІД плану (зв’язок з планом);
- Дата;
- Результати;
- Інформація про поле;
- Інформація про датчик;

План добрив

- ІД плану (первинний ключ);
- ІД поля (зв’язок з полем);
- ІД датчику (зв’язок з датчиком);
- Результати;
- Рекомендації;

Агрохімічний датчик

- ІД датчику (первинний ключ);
- ІД поля (зв’язок з полем);
- Поле;
- Тип аналізу;

Мапа полів

- ІД поля (первинний ключ);

- ІД датчику (зв’язок з датчиком);
- Номер поля;
- Тип ґрунту;
- Тип аналізу;
- Площа

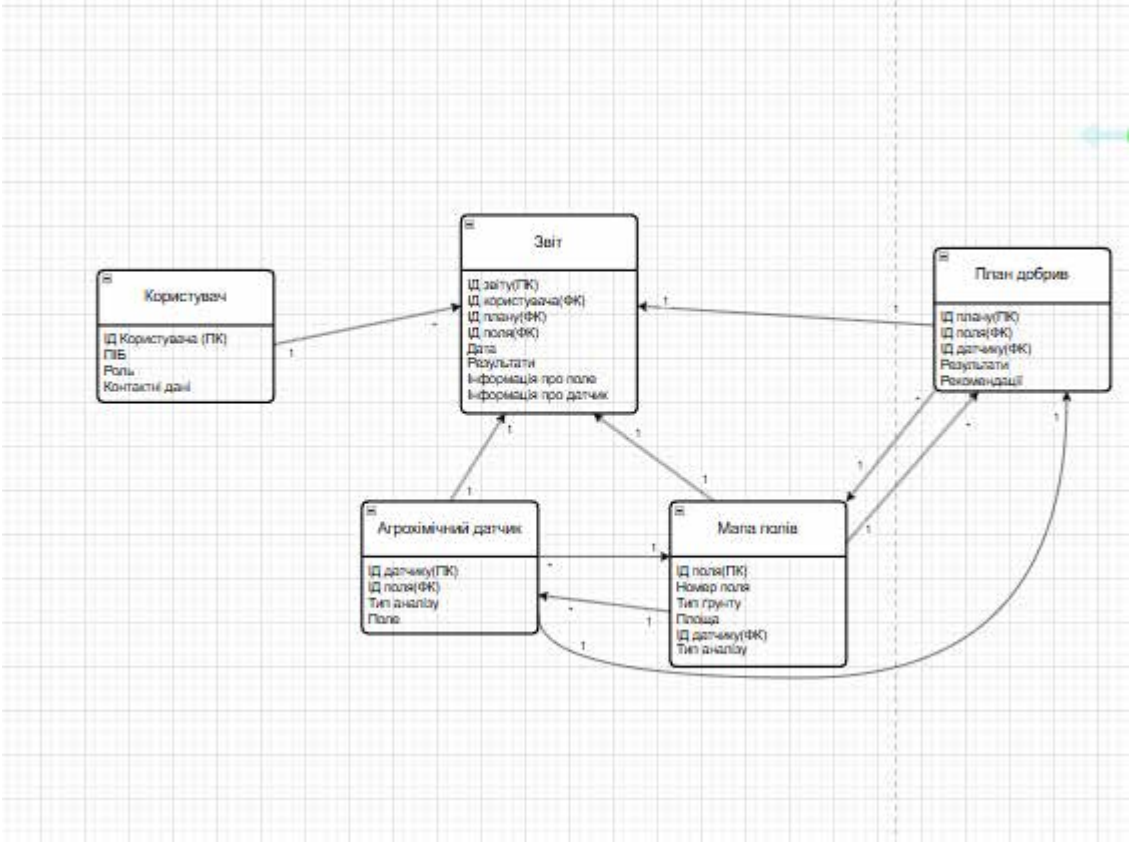

Рис. 2.1 Логічна модель даних програмного продукту

2.2 Діаграма класів та кооперацій

2.2.1 Діаграма класів

Користувачі

- ІД користувача - тип даних int;
- ПІБ - тип даних string;
- Роль користувача - тип даних string;
- Пошта - тип даних string;
- Номер телефону - тип даних int;

Звіт

- ІД звіту - тип даних int;
- ІД користувача - тип даних int;
- ІД поля - тип даних int;
- ІД датчику - тип даних int;
- ІД плану - тип даних int;
- Дата - тип даних date;
- Результати - тип даних float;

План добрив

- ІД плану - тип даних int;
- ІД поля - тип даних int;
- ІД датчику - тип даних int;
- Результати - тип даних float;
- Рекомендації - тип даних string;

Агрохімічний датчик

- ІД датчику - тип даних int;
- ІД поля - тип даних int;
- Поле - тип даних string;
- Тип аналізу - тип даних string;

Мапа полів

- ІД поля - тип даних int;
- ІД датчику - тип даних int;
- Номер поля - тип даних int;
- Тип ґрунту - тип даних string;
- Тип аналізу - тип даних string;
- Площа - тип даних float;

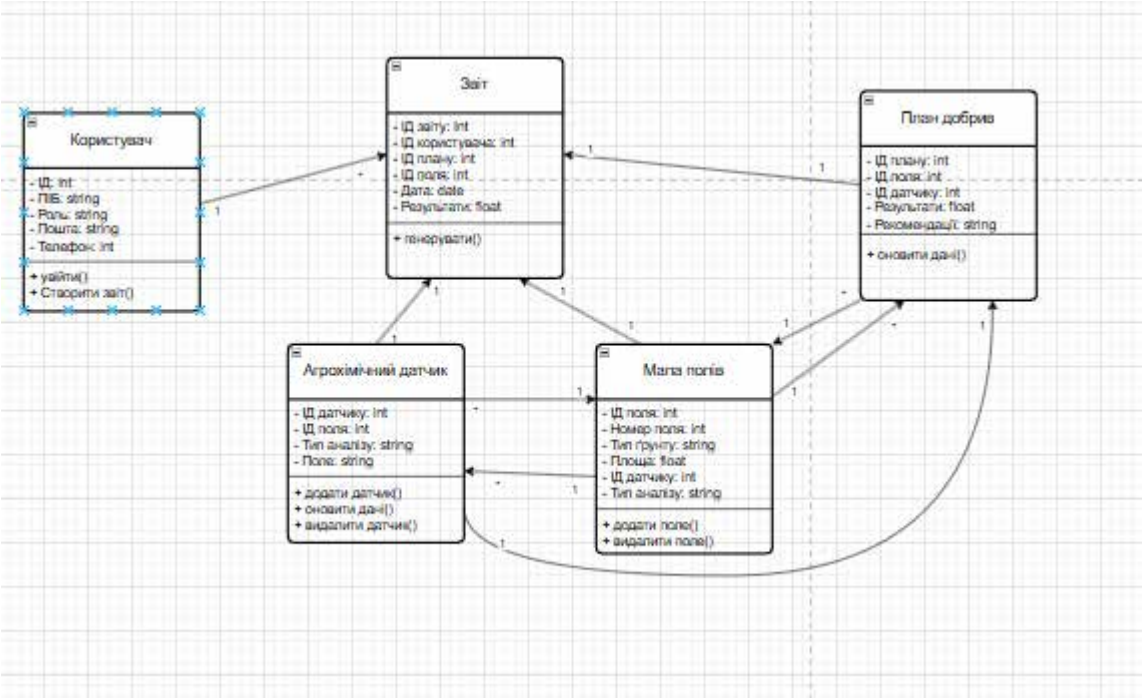

Рис. 2.2 Діаграма класів

2.2.2 Діаграми кооперацій

Діаграма роботи зі звітами.

Звіт створюється на основі вибраного поля на карті полів. Користувач на формі подає запит на створення звіту в систему. Система генерує звіт та добавляє його в каталог звітів. Далі користувач вибирає потрібний звіт.

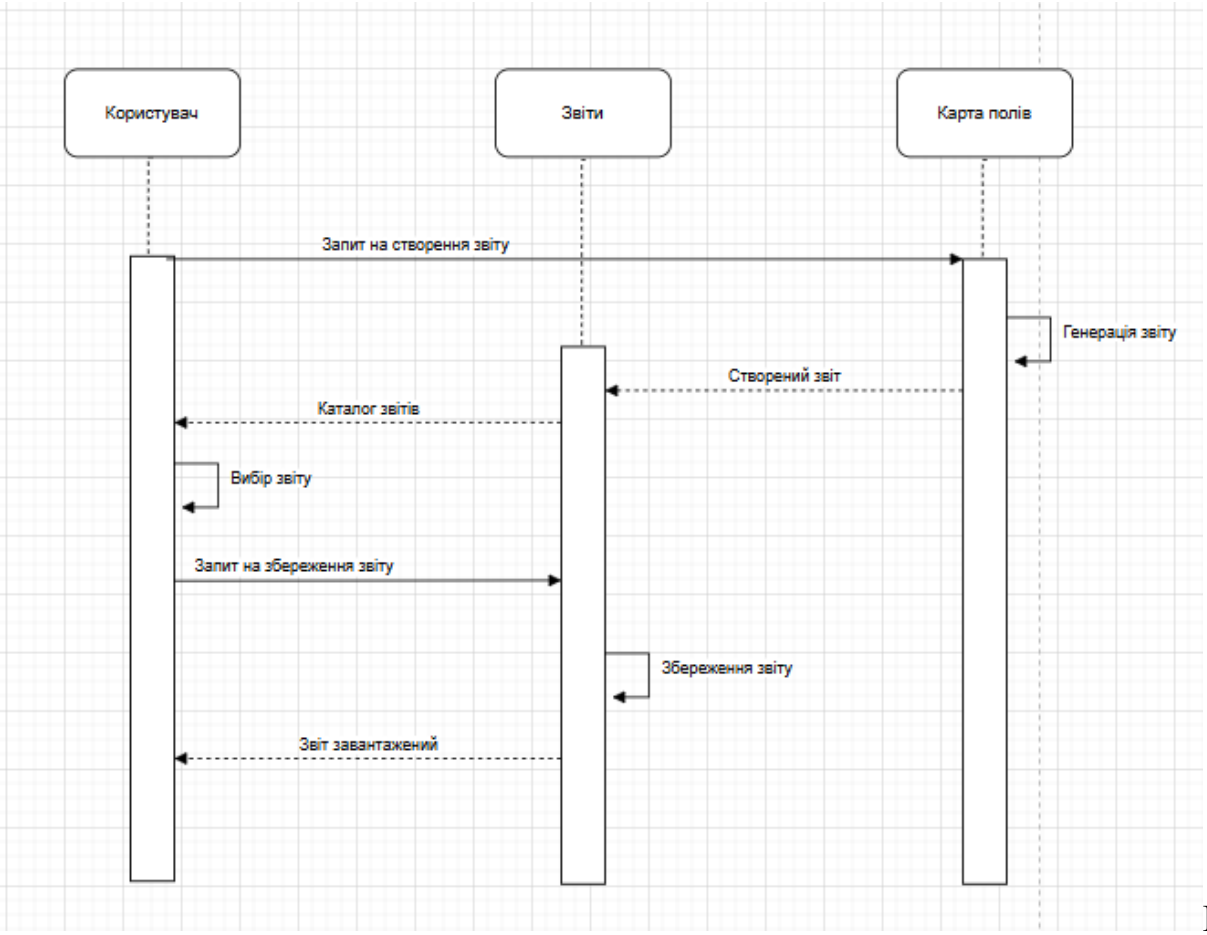

Рис 2.3 Діаграма кооперацій звітності

Діаграма роботи з датчиками

Для роботи з датчиками потрібно мати поле. Після створення поля, користувач переходить на форму датчиків. Користувач подає запит на створення датчику в систему, датчик створюється у вибраному полі. Також можна видалити датчик вибравши його із каталогу.

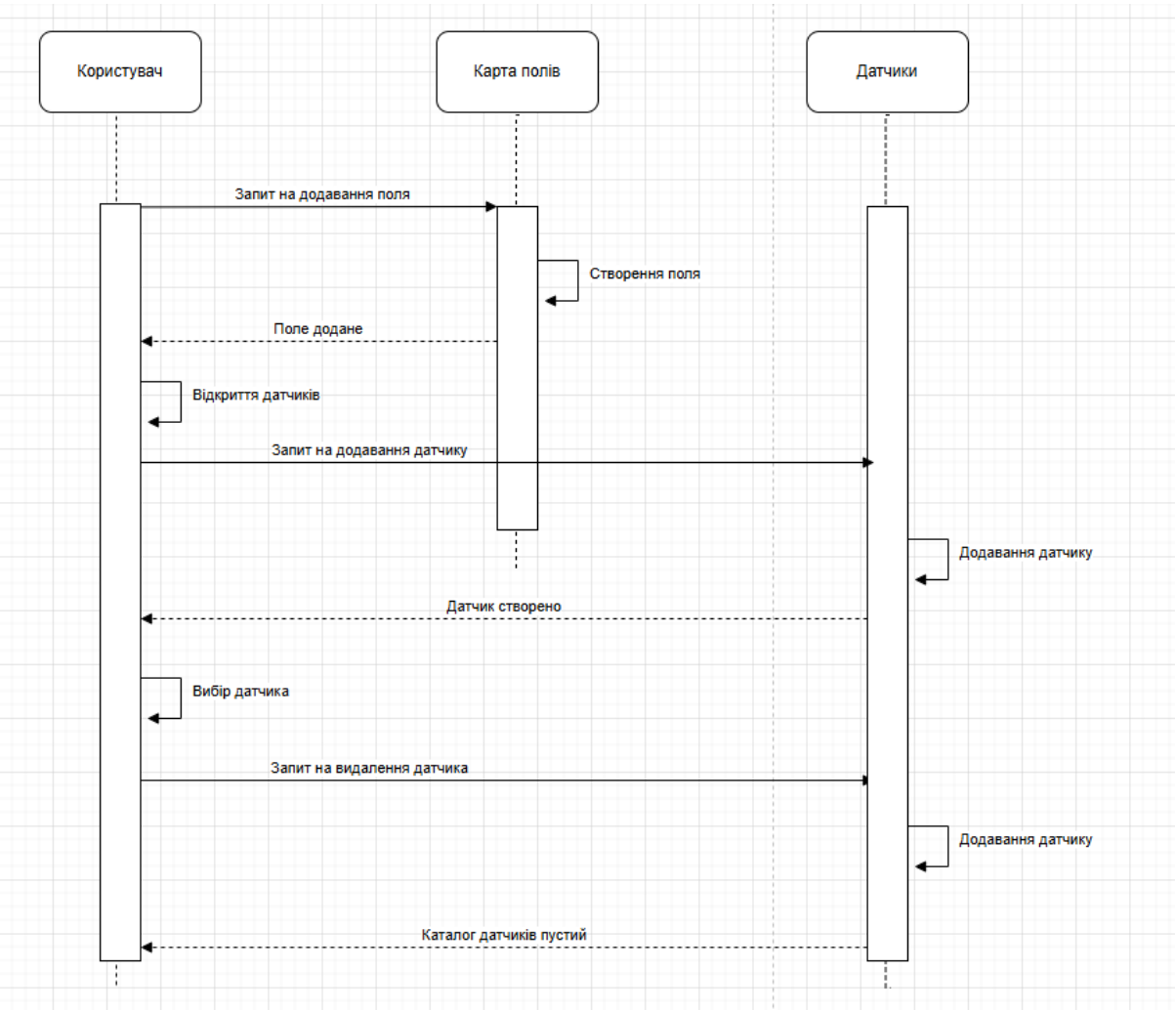

Рис. 2.4 Діаграма кооперації роботи з датчиками

Діаграма створення плану добрив

План добрива створюється на основі датчиків та полів. Після створення поля та датчику на ньому, користувач переходить на форму плани добрива, та подає запит на створення в систему. Система вираховує план на основі результатів датчика.

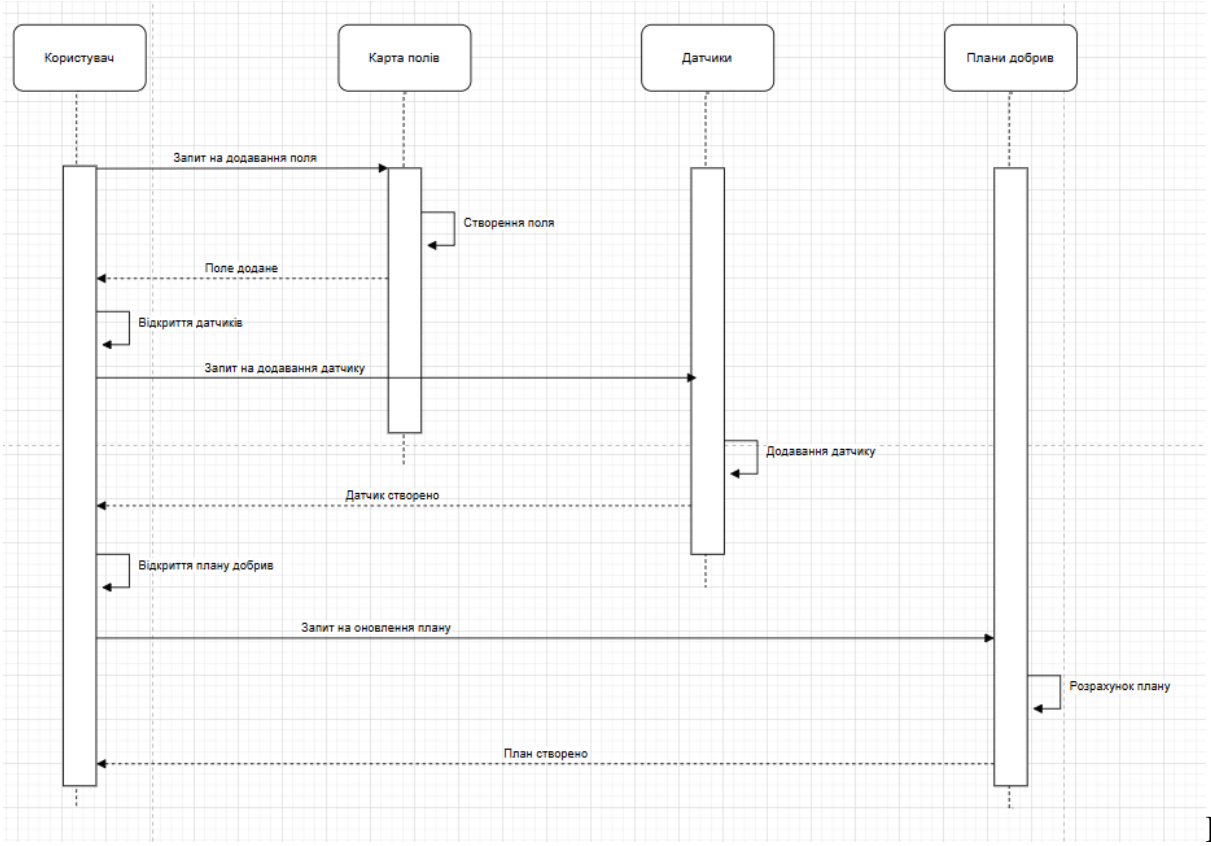

Рис. 2.5 Діаграма кооперацій створення плану добрив

2.3 Діаграма пакетів

Створення діаграми пакетів програмного продукту, допоможе зрозуміти структуру системи з точки зору модулів. На діаграмі групуються класи та інтерфейси в логічні блоки.
Кожен з цих пакетів виконує свою роль в системі.

Додаток:

- Пакет UI - це форми, кнопки та відображення.

Бекенд:

- Пакет BusinessLogic - зв’язує модель з UI, реалізовує бізнес логіку системи.

Зовнішні сервіси:

- Пакет DataSensors integration - даний пакет відповідає за з’єднання датчиків агрохіманалізу з системою та збору даних з датчиків.
- Пакет DataBase - цей пакет потрібний для збереження даних у базі та перегляду даних з бази даних.
- Пакет Reports - дозволяє зберігати звіти у форматі PDF.

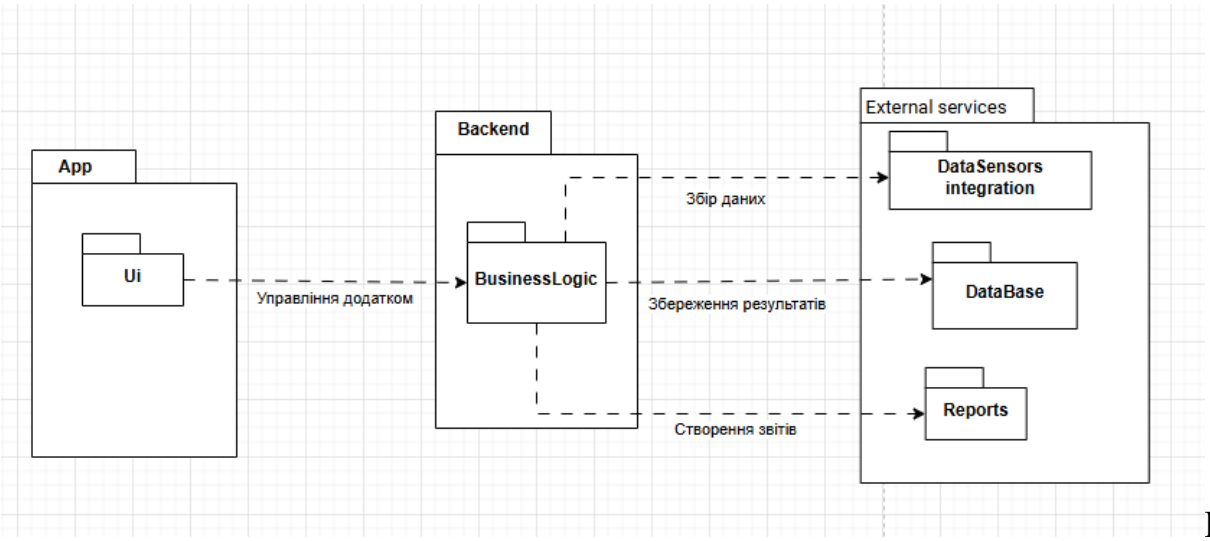

Рис 2.6 Діаграма пакетів

2.4 Діаграма компонентів

Діаграма компонентів показує як компоненти системи взаємодіють між собою, як вони пов’язані та які вони використовують інтерфейси.

Клієнтський додаток - інтерфейс користувача, взаємодіє через бекенд.

- Форма входу;
- Панель користувача;
- Каталог полів;
- Каталог датчиків;
- Каталог звітів;
- Каталог планів

Бекенд - перевіряє доступи, обробляє запити, формує відповіді та генерує звіти.

- Аутентифікація;
- Управління полями;
- Управління датчиками;
- Планування добрив;
- Обробка звітів;

База даних - це структурована частина системи, вона зберігає всю інформацію.

- Таблиця полів;
- Таблиця датчиків;
- Таблиця планів добрива;
- Таблиця звітів;

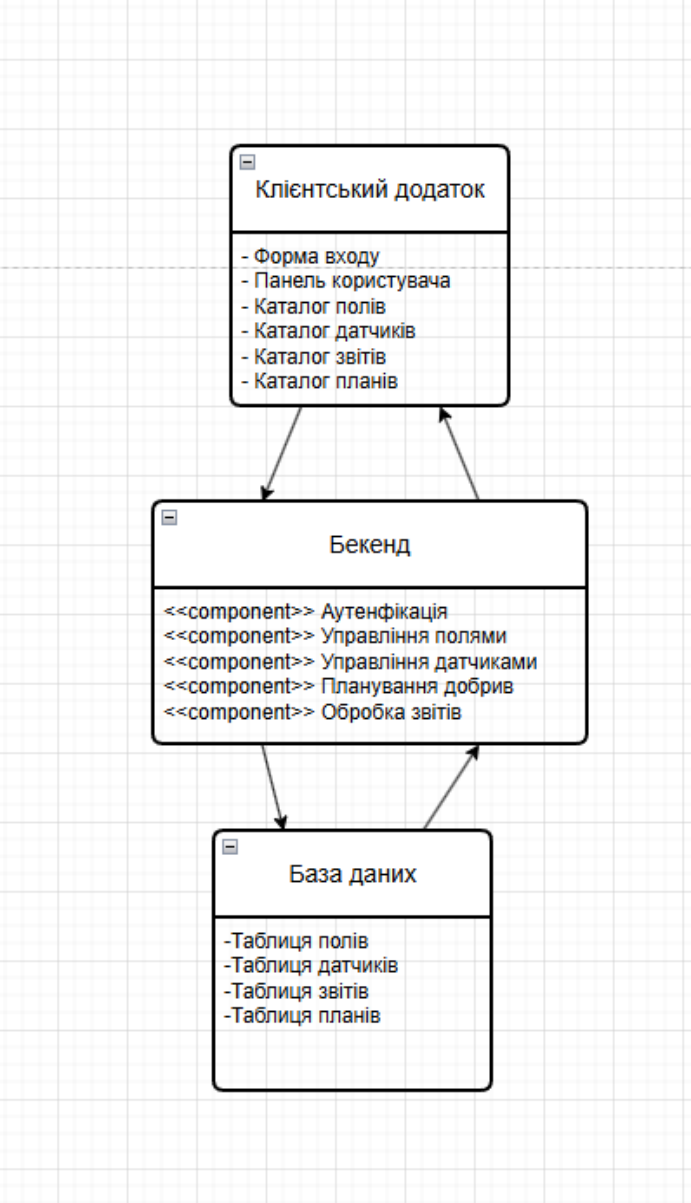

Рис. 2.7 Діаграма компонентів

3 Розробка інформаційного та програмного забезпечення

3.1 Система управління інформаційною базою

Інформаційна база є центральною частиною в програмному забезпечені системи моніторингу агрохіманалізу. Вона зберігає всі дані, котрі необхідні для роботи програмного забезпечення, аналізу ґрунтів, створення планів добрива та генерації звітів.

Для створення інформаціїної бази я обрав Firebase Cloud Firestore - це сучасна хмара NoSQL-бази даних. Вона забезпечує зручну та маштабну структуру зберігання даних, а також синхронізацію даних у реальному часі.

Firebase був обраний з таких причин:

- Підтримка роботи в реальному часі - зміни які вносяться зберігаються та відображаються миттєво;
- Висока швидкість розробки - SDK та REST API легко інтегруються в .NET-додатки для C#;
- Хмарна структура - не потребує локального серверу;

Таблиця 1

Компоненти Firebase

Компонент	Призначення
🔹 **Firebase Realtime Database** або **Cloud Firestore**	Зберігання структурованих даних у форматі JSON (поля, датчики, звіти).
🔹 **Firebase Authentication**	Реєстрація, вхід користувачів, авторизація за ролями (агроном, адміністратор).
🔹 **Firebase Storage**	Зберігання зображень полів, супутникових фото тощо.
🔹 **Firebase Hosting (опційно)**	Розміщення фронтенду (якщо веб-додаток).
🔹 **Firebase Functions (опційно)**	Серверна логіка: генерація звітів, автоматичний розрахунок добрив.

Структура бази даних Firebase виглядає так:

Поля

1. ІД поля;
   - Ід поля;
   - Номер поля;
   - Тип ґрунту;
   - Площа;

2. Датчики;
   - Ід датчика;
   - Ід поля;
   - Номер поля;
   - Тип датчику;

3. Звіт;
    - Ід звіту;
    - Ід поля;
    - Ід датчика;
    - Ід плану добрив;
    - Результати;
    - Дата;

4. План добрив;
    - Ід плану;
    - Ід поля;
    - Ід датчика;
    - Результати;
    - Рекомендації;

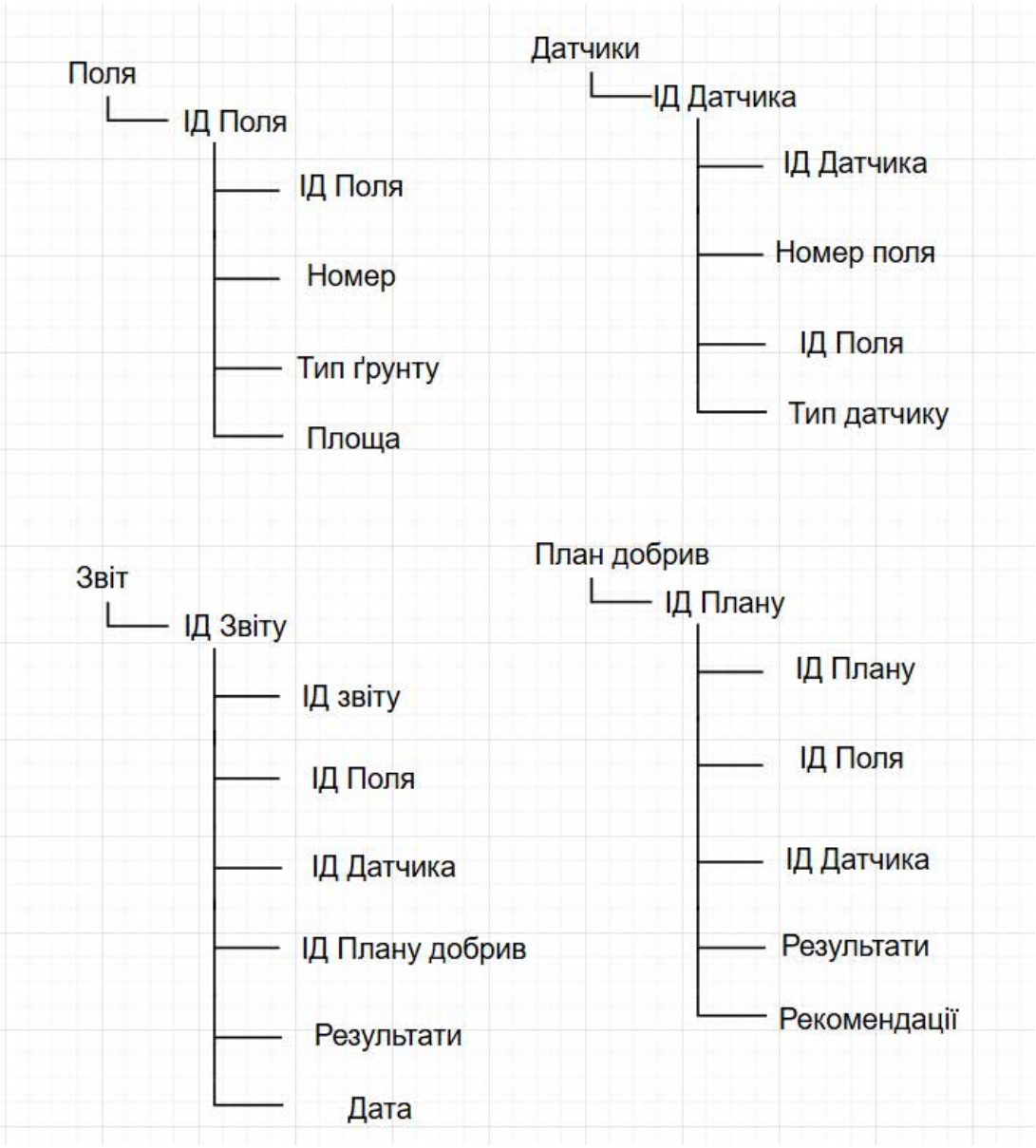

Рис 3.1 Структура бази даних

3.2 Розробка інформаційної бази

Щоб підключити Firebase до середовища C# Windows Forms потрібно використовувати бібліотеку Firebase.Batebase.
Для підключення бази використовується код з Таблиці 3.1:

Таблиця 3.1

```
client = new FirebaseClient("https://diplom-75aba-default-rtdb.europe-west1.firebasedatabase.app/");
```

Після цього з інформаціїною базою даних можна вже працювати.

Робота з датчиками

Для збереження результатів датчиків у інформаційну базу даних, використовувався код з Таблиці 3.2. Нові дані зберігаються у гілці results/sensor_1.

Таблиця 3.2

```
public async Task SaveSensorData(SensorResult data)
{
    await client
        .Child("results/sensor_1")
        .PostAsync(data);
}
```

Щоб зберігати данні їх потрібно отримати. Для отримання даних з агродатчику, використовувався код з Таблиці 3.3. Цей код повертає дані, які надійшли від датчиків.

Таблиця 3.3

```
public async Task<SensorResult> GetLastSensorData()
{
   var data = await client
       .Child("results/sensor_1")
       .OrderByKey()
       .LimitToLast(1)
       .OnceAsync<SensorResult>();
   return data.FirstOrDefault()?.Object;
}
```

Для отримання всіх даних з агродатчиків, використовувався код з Таблиці 3.4.
Цей код потрібен для підтягування даних у таблицю.

Таблиця 3.4

```
public async Task<List<Sensor>> GetAllSensors()
{
    var sensors = await client
        .Child("sensors")
        .OnceAsync<Sensor>();

    return sensors.Select(x => new Sensor
    {
        Id = x.Key,
        Тип = x.Object.Тип,
        ПолеId = x.Object.ПолеId,
        НомерПоля = x.Object.НомерПоля
    }).ToList();
}
```

Для отримання датчика по ІД датчика та отримання датчика за ІД поля. Використовувався код з Таблиць 3.5, та 3.6.

Дані коди використовувалися для отримання інформації про датчик за його ІД, та отримання інформації, до якого ІД поля конкретні датчикі прив’язані.

Таблиця 3.5

```
public async Task<Sensor> GetSensorById(string sensorId)
{
   var sensors = await client
       .Child("sensors")
       .OnceAsync<Sensor>();
   return sensors
       .Where(s => s.Key == sensorId)
       .Select(s =>
       {
           s.Object.Id = s.Key;
           return s.Object;
       })
       .FirstOrDefault();
}
```

Таблиця 3.6

```
public async Task<List<Sensor>> GetSensorsByFieldId(string fieldId)
{
    var sensors = await client
        .Child("sensors")
        .OnceAsync<Sensor>();

    return sensors
        .Where(s => s.Object.ПолеId == fieldId)
        .Select(s => s.Object)
        .ToList();
}
```

Код з Таблиці 3.7 використовувався для додавання датчику до інформаційної бази даних.

Таблиця 3.7

```
public async Task AddSensor(Sensor sensor)
{
    await client
        .Child("sensors")
        .PostAsync(sensor);

}
```

Також датчики можна видалити, для цього використовувався код з Таблиці 3.8.

Таблиця 3.8

```
public async Task DeleteSensor(string sensorId)
{
    await client
        .Child("sensors")
        .Child(sensorId)
        .DeleteAsync();
}
```

Робота з полями

Щоби додавати нові поля у інформаційну базу даних використовується код з Таблиці 3.9.

Таблиця 3.9

```
public async Task AddField(Field field)
{
    string fieldId = $"field{field.Номер}";
    field.Id = fieldId;

    await client
        .Child("fields")
        .Child(fieldId)
        .PutAsync(field);
}
```

Для отримання даних використовується ІД поля, для виконання цього використовується код з Таблиці 3.10.

Таблиця 3.10

```
public async Task<Field> GetFieldById(string fieldId)
{
    var field = await client
.Child("fields")
.Child(fieldId)
.OnceSingleAsync<Field>();

    return field;
}
```

Робота з планами добрив

Щоби створити новий план використовується код з Таблиці 3.11. Він створює новий план добрив з унікальним ІД.

Таблиця 3.11

```
public async Task SavePlan(Plan plan)
{
    var id = Guid.NewGuid().ToString(); // Унікальний ідентифікатор
    plan.Id = id;

    await client
        .Child("Plans")
        .Child(id)
        .PutAsync(plan);
}
```

Для отримання даних використовується два коди з Таблиць 3.12, та 3.13. В Таблиці 3.12 знаходиться код для отримання всіх планів.

Таблиця 3.12

```
public async Task<List<Plan>> GetAllPlans()
{
    var plans = await client
        .Child("Plans")
        .OnceAsync<Plan>();

    return plans.Select(p =>
    {
        var plan = p.Object;
        plan.Id = p.Key;
        return plan;
    }).ToList();
}
```

В Таблиці 1.13 код для отримання даних про конкретний план.

Таблиця 3.13

```
public async Task<Plan> GetPlanById(string planId)
{
    var plan = await client
        .Child("Plans")
        .Child(planId)
        .OnceSingleAsync<Plan>();

    plan.Id = planId;
    return plan;
}
```

Робота зі звітами

Щоб зберегти звіт використовується код з Таблиці 3.14.

Таблиця 3.14

```
public async Task SaveReport(Report report)
{
   var id = Guid.NewGuid().ToString(); // Генеруємо ID
   report.Id = id;

   await client
       .Child("Reports")
       .Child(id)
       .PutAsync(report);
}
```

Отримання звітів за структурою схоже на отримання планів добрив. Для отримання всіх звітів код з Таблиці 3.15.

Таблиця 3.15

```
public async Task<List<Report>> GetAllReports()
{
   var reports = await client
      .Child("Reports")
      .OnceAsync<Report>();

   return reports.Select(r =>
   {
      var report = r.Object;
      report.Id = r.Key;
      return report;
   }).ToList();
}
```

Для отримання конкретного звіту код з Таблиці 3.16.

Таблиця 3.16

```
public async Task<Report> GetReportById(string id)
{
    var report = await client
       .Child("Reports")
       .Child(id)
       .OnceSingleAsync<Report>();
    report.Id = id;
    return report;
}
```

Також у системі реалізовано моделі даних для коректної десеріалізації.

- Модель для полів

Таблиця 3.17

```
public class Field
{
   public string Id { get; set; }
   public string Номер { get; set; }
   public string ТипГрунту { get; set; }
   public string Площа { get; set; }
}
```

- Модель для датчиків

Таблиця 3.18

```
public class Sensor
{
   public string Id { get; set; }
   public string Тип { get; set; }
   public string ПолеId { get; set; }
   public string НомерПоля { get; set; }
}
```

- Модель для планів добрив

Таблиця 3.19

```
public class Plan
{
   public string Id { get; set; }
   public string ПолеId { get; set; }
   public string ДатчикId { get; set; }
   public string ТипДатчика { get; set; }
   public double Значення { get; set; }
   public double Мін { get; set; }
   public double Макс { get; set; }
   public string Рекомендації { get; set; }
   public DateTime Дата { get; set; }
}
```

- Модель для звітів

Таблиця 3.20

```
public class Report
{
    public string Id { get; set; }
    public string ПолеId { get; set; }
    public string НомерПоля { get; set; }
    public string НазваПоля { get; set; }
    public string ДатчикId { get; set; }
    public string ПланId { get; set; }
    public DateTime Дата { get; set; }
}
```

- Модель для результату датчиків

Таблиця 3.21

```
public class SensorResult
{
    public double Nitrogen { get; set; }
    public double PH { get; set; }
    public double Moisture { get; set; }
    public DateTime Timestamp { get; set; }
}
```

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для розробки системи моніторингу агрохіманалізу були вибрані інструменти, які найкраще забезпечують масштабність, зручну розробку та інтеграцію з хмарними сервісами:

Мова програмування

Як основною мовою для розробки прикладної частини проєкту було обрано c# (C-Sharp).

Вона надає:

- Зручну роботу з графічними інтерфейсами;
- Можливість використання бібліотек для підключення Firebase;
- Об’єктно-орієнтований підхід до розробки;

Технології інтерфейсу

Щоб реалізувати графічний інтерфейс було використано Windows Forms, вона є частиною .NET Framework. Також вона надає змогу швидко створювати візуальні елементи.

Її перевагами є:

- Інтеграція з Visual Studio;
- Швидкий старт для десктоп-додатків;
- Простота у відлагоджені;

Хмарна база даних

Основу для системи зберігання даних було використано Firebase Realtime Database.

Яка в свою чергу забезпечує:

- Асинхронний обмін даними;
- Миттєву синхронізацію між хмарою та користувачем;
- Підтримку клієнтских бібліотек для .NET;
- Зберігання даних у формати JSON;

Бібліотека

Щоби взаємодіяти c# з Firebase використовується бібліотека Fiarebase.Database.

Вона дозволяє:

- здійснювати CRUD операції;
- Працювати з асинхронними запитами;
- Використовувати LINQ;

Середовище розробки

Для реалізації застосунку використовувалося середовище розробки Microsoft Visual Studio 2022.

Її преваги:

- Повна підтримка Windows Forms;
- Вбудовані засоби роботи з базами даних;
- Зручне середовище розробки;
- Опит роботи з цим середовищем;

3.4 Алгоритмізація та програмування програмних модулів

Програмне забезпечення складається з таких модулів:

- Модуль авторизації
- Модуль датчиків
- Модуль полів
- Модуль планування добрив
- Модуль звітів
- Модуль додавання даних

Модуль датчиків

Цей модуль призначений для перегляду усіх наявних датчиків. Він дає можливість переглядати таблицю з усіма наявними датчиками та їхніми параметрами.

Основна логіка полягає в отриманні всіх датчиків та інформації про них з бази даних Firebase та відображає у таблиці DataGridView.
В даній таблиці стовбці створюються в ручну, що дозволяє контролювати їх назви та порядок, за допомогою коду в Таблиці 3.22.

Спочатку таблиця очищується від попередніх даних, далі створюються колонки:

- ІД
- Тип
- ІД поля
- Номер поля

Далі таблиця заповнюється даними з бази даних, за допомогою методу firebase.GetAllSensors().

Таблиця 3.22

```
private async Task LoadSensorTable()
{
   dataGridView1.Columns.Clear();
   dataGridView1.Rows.Clear();
   dataGridView1.AutoGenerateColumns = false;

   dataGridView1.Columns.Add("Id", "ID");
   dataGridView1.Columns.Add("Тип", "Тип");
   dataGridView1.Columns.Add("ПолеId", "ID поля");
   dataGridView1.Columns.Add("НомерПоля", "Номер поля");
   dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;

   var sensorList = await firebase.GetAllSensors();
   foreach (var s in sensorList)
   {
      dataGridView1.Rows.Add(s.Id, s.Тип, s.ПолеId, s.НомерПоля);
   }
}
```

Метод полів

Він складається з двох пов’язаних форм.

- Форма вибору поля. Тут користувач вибирає потрібне поле.
- Форма перегляду інформації про поле. На цій формі користувач може переглянути інформацію про вибране ним поле та про датчики які знаходяться на цьому полі.

Реалізація вибору поля

Коли користувач натискає на конкретне поле то запускається метод OpenFieldDetails зі вказаним ІД поля наприклад ("field1"), код кнопки поля Таблиця 3.23.

Таблиця 3.23

```
private void button1_Click(object sender, EventArgs e)
{
    OpenFieldDetails("field1");
}
```

Коли запускається метод, він перевіряє ІД вибраного поля, і відкриває форму FieldDetailsForm з інформацією про конкретне поле, на основі ІД поля GetFieldById, код методу в Таблиці 3.24.

Таблиця 3.24

```
private async void OpenFieldDetails(string fieldId)
{


   var field = await firebase.GetFieldById(fieldId);
   if (field != null)
   {
      var detailsForm = new FieldDetailsForm(field);
      detailsForm.Show();
   }
   else
   {
      MessageBox.Show("Поле не знайдено!");
   }
}
```

Після відкриття форми FieldDetailsForm, запускається метод FieldDetailsForm_Load, який на основі ІД поля підтягує всю потрібну інформацію:

- Інформація про поле. ІД, Номер поля, Тип ґрунту, Площа;
- Інформація про датчикі;
- Фільтрація датчиків, які прив’язані до вибраного поля, та заповнення таблиця цими датчиками;

Таблиця 3.25

```
private async void FieldDetailsForm_Load(object sender, EventArgs e)
{
      if (field == null)
   {
      MessageBox.Show("Поле не знайдено");
      this.Close();
      return;
   }
   label1.Text = $"ID: {field.Id}";
   label2.Text = $"Номер: {field.Номер}";
   label3.Text = $"Тип ґрунту: {field.ТипҐрунту}";
   label4.Text = $"Площа: {field.Площа}";

      var allSensors = await firebase.GetAllSensors();
      var sensors = allSensors.Where(s => s.ПолеId == field.Id).ToList();

   dataGridView1.DataSource = null;
   dataGridView1.Columns.Clear();
   dataGridView1.AutoGenerateColumns = false;

   dataGridView1.Columns.Add("Id", "ID");
   dataGridView1.Columns.Add("Тип", "Тип");
   dataGridView1.Columns.Add("ПолеId", "ID поля");
   dataGridView1.Columns.Add("НомерПоля", "Номер поля");
   foreach (var s in sensors)
   {
      dataGridView1.Rows.Add(s.Id, s.Тип, s.ПолеId, s.НомерПоля);
   }
   dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
}
```

Створення звіту

На формі FieldDetailsForm також реалізовано створення звіту Таблиця 3.26. Звіт буде створюватися на основі поля яке переглядає користувач, з алгоритмом:

- Отримання планів
- Фільтрація планів за вибраним полем
- Вибір найновішого плану
- Формування звіту та його збереження у базу даних

Таблиця 3.27

```
private async void button2_Click(object sender, EventArgs e)
{
    var allPlans = await firebase.GetAllPlans();
    var fieldPlans = allPlans
        .Where(p => p.ПолеId == field.Id)
        .OrderByDescending(p => p.Дата)
        .ToList();

    if (fieldPlans.Count == 0)
    {
        MessageBox.Show("Немає збережених планів добрив для цього
поля.");
        return;
    }

    var latestPlan = fieldPlans.First();
    var newReport = new Report
    {
        ПолеId = latestPlan.ПолеId,
        ДатчикId = latestPlan.ДатчикId,
        ПланId = latestPlan.Id,
        НомерПоля = field.Номер,
        Дата = DateTime.Now
    };

    // Зберігаємо у Firebase
    await firebase.SaveReport(newReport);

    MessageBox.Show("✅ Звіт успішно створено!");
}
```

Плани добрив

На формі PlansForm користувач переглядає таблицю dataGridView1 з показниками датчика та рекомендаціями щодо добрив, для покращення уваги дані підсвічуються коляром.

Для перегляду планів користувач повинен скласти плани добрив Таблиця 3.28. Плани добрив складаються з перевірки показників датчиків, далі показники звіряються з нормою, та надається рекомендація щодо підвищення або зниження плану добрив. Після чого сформовані плани зберігаються у базу даних.

Таблиця 3.28

```
    private async void button1_Click(object sender, EventArgs e)
    {
        dataGridView1.Rows.Clear();
        dataGridView1.Columns.Clear();

        dataGridView1.Columns.Add("ПланId", "ID Плану");
        dataGridView1.Columns.Add("ПолеId", "ID Поля");
        dataGridView1.Columns.Add("ДатчикId", "ID Датчика");
        dataGridView1.Columns.Add("ТипДатчика", "Тип датчика");
        dataGridView1.Columns.Add("Значення", "Значення");
        dataGridView1.Columns.Add("Мін", "Мін");
        dataGridView1.Columns.Add("Макс", "Макс");
        dataGridView1.Columns.Add("Рекомендації", "Рекомендації");

        var sensors = await firebase.GetAllSensors();
        Random random = new Random();

        Dictionary<string, (double min, double max)> sensorLimits = new
Dictionary<string, (double, double)>
{
    { "Температурний", (5, 35) },
    { "Вологість ґрунту", (30, 70) },
    { "pH рівень", (6, 7.5) },
    { "Електропровідність", (0.2, 2.0) },
    { "Освітленість", (200, 800) }
};
```

```
foreach (var sensor in sensors)
      {
         string тип = string.IsNullOrEmpty(sensor.Тип) ? "Невідомий" :
sensor.Тип;
         double value = 0, min = 0, max = 0;
         string рекомендація = "Немає даних";

         if (sensorLimits.ContainsKey(тип))
         {
            (min, max) = sensorLimits[тип];
            double range = max - min;
            double lower = min - range * 0.2;
            double upper = max + range * 0.2;
            value = Math.Round(random.NextDouble() * (upper - lower) +
lower, 2);
            if (value < min)
               рекомендація = $"Низьке значення — підвищити параметр";
            else if (value > max)
               рекомендація = $"Високе значення — зменшити вплив";
            else
               рекомендація = "Показник у нормі";
         }
         else
         {
            рекомендація = "Невідомий тип датчика";
         }

         string planId = Guid.NewGuid().ToString().Substring(0, 8);
         int rowIndex = dataGridView1.Rows.Add(planId, sensor.ПолеId,
sensor.Id, тип, value, min, max, рекомендація);
         DataGridViewRow row = dataGridView1.Rows[rowIndex];

         if (value < min)
            row.DefaultCellStyle.BackColor = Color.LightBlue;
         else if (value > max)
            row.DefaultCellStyle.BackColor = Color.LightCoral;
         else
            row.DefaultCellStyle.BackColor = Color.LightGreen;
      }
      dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
```

```
foreach (DataGridViewRow row in dataGridView1.Rows)
      {
```

```
        if (row.IsNewRow) continue;

        var plan = new Plan
        {
            ПолеId = row.Cells["ПолеId"].Value?.ToString(),
            ДатчикId = row.Cells["ДатчикId"].Value?.ToString(),
            ТипДатчика = row.Cells["ТипДатчика"].Value?.ToString(),
            Значення = Convert.ToDouble(row.Cells["Значення"].Value),
            Мін = Convert.ToDouble(row.Cells["Мін"].Value),
            Макс = Convert.ToDouble(row.Cells["Макс"].Value),
            Рекомендації = row.Cells["Рекомендації"].Value?.ToString()
        };

        await firebase.SavePlan(plan);
    }
}
```

Звітність

Звітність складається з двох форм:

- Форма вибору звіту ReportsForm. На цій формі користувач обирає з таблиці dataGridView1 який звіт йому переглянути.
- Форма перегляду звіту. Після вибору звіту користувач преглядає сам звіт.

При запуску форми ReportsForm, запускається метод ReportsForm_Load, Таблиця 2.19. Він заповняє таблицю наявними звітами.

Таблиця 3.29

```
private async void ReportsForm_Load(object sender, EventArgs e)
{
   reports = await firebase.GetAllReports();

   dataGridView1.DataSource = reports.Select(r => new
   {
      r.Id,
      НомерПоля = r.НомерПоля,
      Дата = r.Дата.ToString("yyyy-MM-dd HH:mm")
   }).ToList();

   dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
}
```

Для перегляду всієї інформації використовується метод button2_Click, Таблиця 3.30. Запускається форма ReportDetailsForm на основі вибраного звіту.

Таблиця 3.30

```
private async void button2_Click(object sender, EventArgs e)
{
   if (dataGridView1.SelectedRows.Count > 0)
   {
      string reportId =
dataGridView1.SelectedRows[0].Cells["Id"].Value.ToString();
      var report = reports.FirstOrDefault(r => r.Id == reportId);
      if (report != null)
      {
         var form = new ReportDetailsForm(report);
         form.ShowDialog();
      }
   }
   else
   {
      MessageBox.Show("Оберіть звіт для перегляду.");}
}
```

Коли завантажується форма ReportDetailsForm, запускається метод ReportDetailsForm_Load для заповнення всіх даних на основі вибраного звіту. Таблиця 3.31.

Таблиця 3.31

```
private async void ReportDetailsForm_Load(object sender, EventArgs e)
{
    label1.Text = $"ID звіту: {report.Id}";
    label3.Text = $"ID поля: {report.ПолеId}";
    label4.Text = $"ID датчика: {report.ДатчикId}";
    label5.Text = $"ID плану: {report.ПланId}";
    label2.Text = $"Дата: {report.Дата:yyyy-MM-dd HH:mm}";

    var поле = await firebase.GetFieldById(report.ПолеId);
    var датчик = await firebase.GetSensorById(report.ДатчикId);
    var план = await firebase.GetPlanById(report.ПланId);

    dataGridView1.Columns.Add("ТипДатчика", "Тип датчика");
    dataGridView1.Columns.Add("Значення", "Значення");
    dataGridView1.Columns.Add("НомерПоля", "Номер поля");
    dataGridView1.Columns.Add("Рекомендації", "Рекомендації");

    dataGridView1.Rows.Add(датчик.Тип, план.Значення, поле.Номер,
план.Рекомендації);
}
```

Панель адміністратора

Для додавання поля, додавання та видалення датчика є окремі форми adminPanelForm, AddFieldForm, AddSensorForm.

На формі adminPanelForm, адміністратор може додати поля, додати датчики та видалити датчики.
При додаванні нового поля відкривається форма AddFieldForm, та запускається метод AddFieldForm_Load для заповнення даними comboBox, Таблиця 3.32.

Таблиця 3.32

```
private void AddFieldForm_Load(object sender, EventArgs e)
{
   comboBox1.Items.AddRange(new string[] { "Чорнозем", "Торф'яний",
"Суглинковий" });
   comboBox2.Items.AddRange(new string[] { "Поле 1", "Поле 2", "Поле
3", "Поле 4", "Поле 5" });
}
```

Далі при додаванні поля запускається метод button1_Click, вибрані дані зберігаються у базу даних як нове поле, Таблиця 3.33.

Таблиця 3.33

```
private async void button1_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(comboBox2.Text) ||
string.IsNullOrEmpty(comboBox1.Text) ||
string.IsNullOrEmpty(textBox1.Text))
   {
      MessageBox.Show("Заповни всі поля!", "Помилка");
      return;
   }
   string номер = comboBox2.Text.Replace("Поле ", "").Trim();
   var field = new Field
   {
      Номер = номер,
      ТипГрунту = comboBox1.Text,
      Площа = textBox1.Text};
   await firebase.AddField(field);
   MessageBox.Show("Поле успішно додано до бази даних!", "Успіх");
   this.Close();}
```

При запуску форми AddSensorForm, запускається метод AddSensorForm_Load для заповнення даних у comboBox, Таблиця 3.34.

Таблиця 3.34

```
private void AddSensorForm_Load(object sender, EventArgs e)
{
   comboBox1.Items.AddRange(new string[] { "Температурний",
"Вологість ґрунту", "pH рівень", "Електропровідність", "Освітленість" });
   comboBox2.Items.AddRange(new string[] { "Поле 1", "Поле 2", "Поле
3", "Поле 4", "Поле 5" });
}
```

При додавання датчика запускається метод button2_Click, як із додаванням поля, вибрані дані зберігаються у базу даних як новий датчик, Таблиця 3.35.

Таблиця 3.35

```
private async void button2_Click(object sender, EventArgs e)
{
   string type = comboBox1.SelectedItem?.ToString();
   string selectedField = comboBox2.SelectedItem?.ToString();
   if (string.IsNullOrWhiteSpace(type) ||
string.IsNullOrWhiteSpace(selectedField))
   {
      MessageBox.Show("Будь ласка, заповніть усі поля.");
      return;
   }
      string полеId = "field" + selectedField.Replace("Поле ", "").Trim();
   var newSensor = new Sensor
   {
      Id = Guid.NewGuid().ToString(),
      Тип = type,
      ПолеId = полеId,
      НомерПоля = selectedField
   };
   await firebase
      .Child("sensors")
      .Child(newSensor.Id)
      .PutAsync(newSensor);
   MessageBox.Show("Датчик успішно додано!");
   this.Close();}
```

Для видалення датчика, на формі adminPanelForm, запускається метод button4_Click він видаляє вибраний датчик у comboBox, та видаляє його на основі його ІД, Таблиця 3.36.

Таблиця 3.36

```
private async void button4_Click(object sender, EventArgs e)
{
    if (comboBox1.SelectedItem != null)
    {
        string sensorId = comboBox1.SelectedItem.ToString();
        await firebase.DeleteSensor(sensorId);

        MessageBox.Show("Датчик успішно видалено");

        // Оновлюємо список у ComboBox
        var sensors = await firebase.GetAllSensors();
        comboBox1.Items.Clear();
        foreach (var sensor in sensors)
        {
            comboBox1.Items.Add(sensor.Id);
        }
    }
    else
    {
        MessageBox.Show("Виберіть датчик для видалення.");
    }
}
```

4 Рекомендації щодо впровадження та експлуатації системи

4.1 Тестування системи

Після розробки програмного забезпечення системи моніторингу агрохіманалізу було проведено комплексне тестування, яке включало в себе модульне, інтеграційне та функціональне тестування. Мета цього тестування в тому, щоб перевірити коректність роботи програмного забезпечення, переконатися у відповідності вимогам та виявити помилки.

Модульне тестування

Модульне тестування проводилося для окремих компонентів системи:

- Взаємодія з Firebase
- Логіка генерації планів добрив
- Форми для роботи з датчиками, полями, планами та звітами.

Для проведення модульного тестування застосовано методи логічного тестування, із даними.

Інтеграційне тестування

Інтеграційне тестування включало в себе:

- Взаємодію між формами
- Збереження, оновлення та завантаження пов’язаних між собою сутностей
- Перевірка коректних даних на основі зв’язків між об’єктами

Взаємодія між формами виконана за допомогою menuStrip1, на ній добавлені кнопки:

- Головна
- Меню (Звіти, Датчики, Карта полів, Плани добрив, Адмін панель)
- Закрити (Вийти, Закрити)

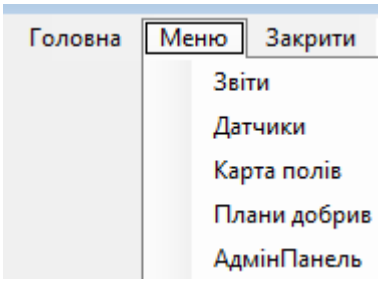

Рис 4.1 Меню

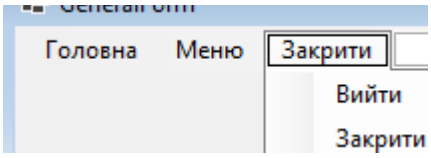

Рис 4.2 Закрити

При натисканні на кнопку відкривається відповідна форма. Кнопка АдмінПанель активна тільки для користувача адмін. При натисканні на кнопку Вийти користувача перекидає на форму авторизації. При натисканні на Закрити програма закривається.

Всі форми пов’язані між собою. Датчики пов’язані з полями, на формі датчики знаходяться всі датчики, Рис 4.3. вони прив’язані до полів, тобто в кожного поля є свої датчики, це наглядно зображено на Рис.4.4 - 4.7 .

	ID	Тип	ID поля	Номер поля
	109c86ea-beea-42fa-8964-b09...	Вологість ґрунту	field1	Поле 1
	12bec867-2b33-4f89-99ce-3cb...	pH рівень	field3	Поле 3
	2fc66d09-ef05-4f28-b9e9-da2...	Освітленість	field4	Поле 4
	457376cc-efa3-490a-b1c4-53a...	pH рівень	field1	Поле 1
	cf15458b-494b-4bf3-88ed-45b...	Електропровідність	field3	Поле 3
	e6008397-07ea-476b-b644-98...	Електропровідність	field2	Поле 2
	f6c67a9d-68f9-424f-8359-714...	Електропровідність	field1	Поле 1
▸*				

Рис 4.3 Всі датчики.

Номер: 1 ID: field1 Тип ґрунту: Чорнозем Площа: 2.2

	ID	Тип	ID поля	Номер поля
▸	109c86ea-beea-42fa-8964-b099...	Вологість ґрунту	field1	Поле 1
	457376cc-efa3-490a-b1c4-53a8...	pH рівень	field1	Поле 1
	f6c67a9d-68f9-424f-8359-714b1...	Електропровідність	field1	Поле 1
*				

Рис 4.4 Поле 1.

Номер: 2 ID: field2 Тип ґрунту: Чорнозем Площа: 2

	ID	Тип	ID поля	Номер поля
▸	e6008397-07ea-476b-b644-9837...	Електропровідність	field2	Поле 2
*				

Рис 4.5 Поле 2.

Номер: 3 ID: field3 Тип ґрунту: Торф'яний Площа: 2.7

	ID	Тип	ID поля	Номер поля
▸	12bec867-2b33-4f89-99ce-3cbc...	pH рівень	field3	Поле 3
	cf15458b-494b-4bf3-88ed-45ba...	Електропровідність	field3	Поле 3
*				

Рис 4.6 Поле 3.

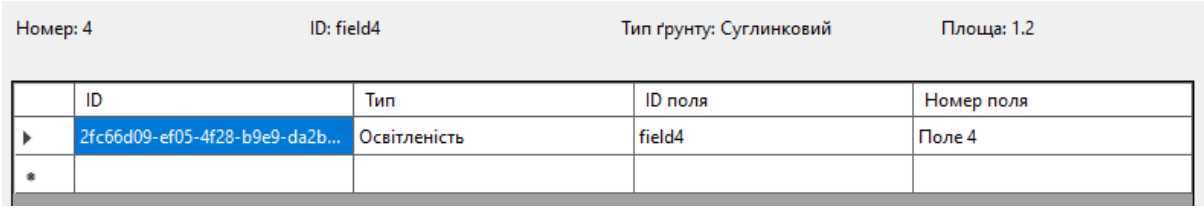
Номер: 4
ID: field4
Тип ґрунту: Суглинковий
Площа: 1.2
ID
Тип
ID поля
Номер поля
2fc66d09-ef05-4f28-b9e9-da2b...
Освітленість
field4
Поле 4

Рис 4.7 Поле 4.

План добрив напряму залежить від Датчиків, тобто якщо в базі немає жодного датчику то і плану не буде з чого складати. План добрив перевіряє всі наявні датчики в туж секунду коли користувач натисне Розрахувати план добрив, Рис 4.8.

Рис 4.8 Плани добрив

Звіти залежать від усіх, вони беруть дані з усіх інших форм для створення звіту, Рис 4.9 - 4.11.

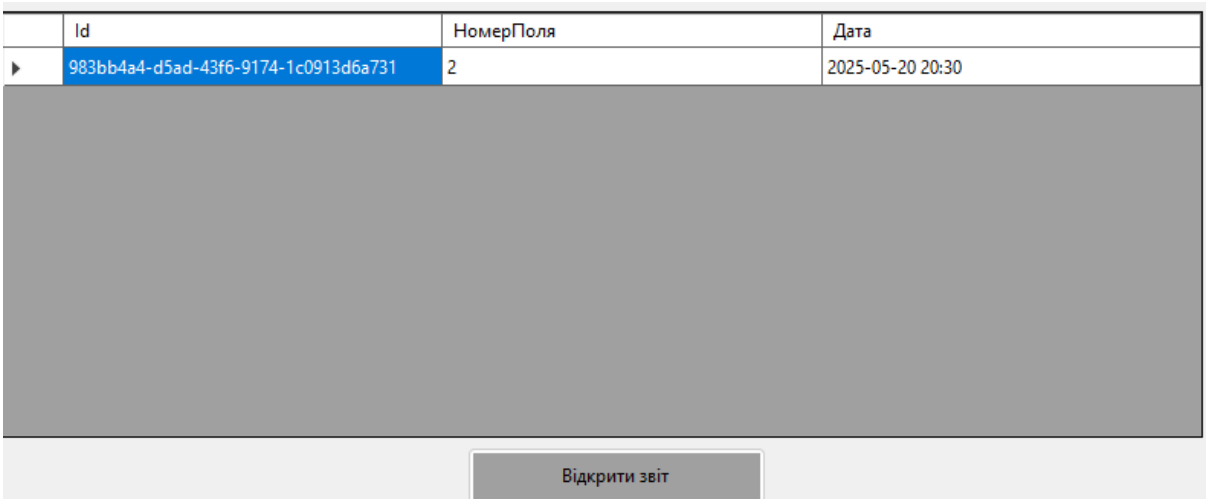

Рис 4.9 Список звітів

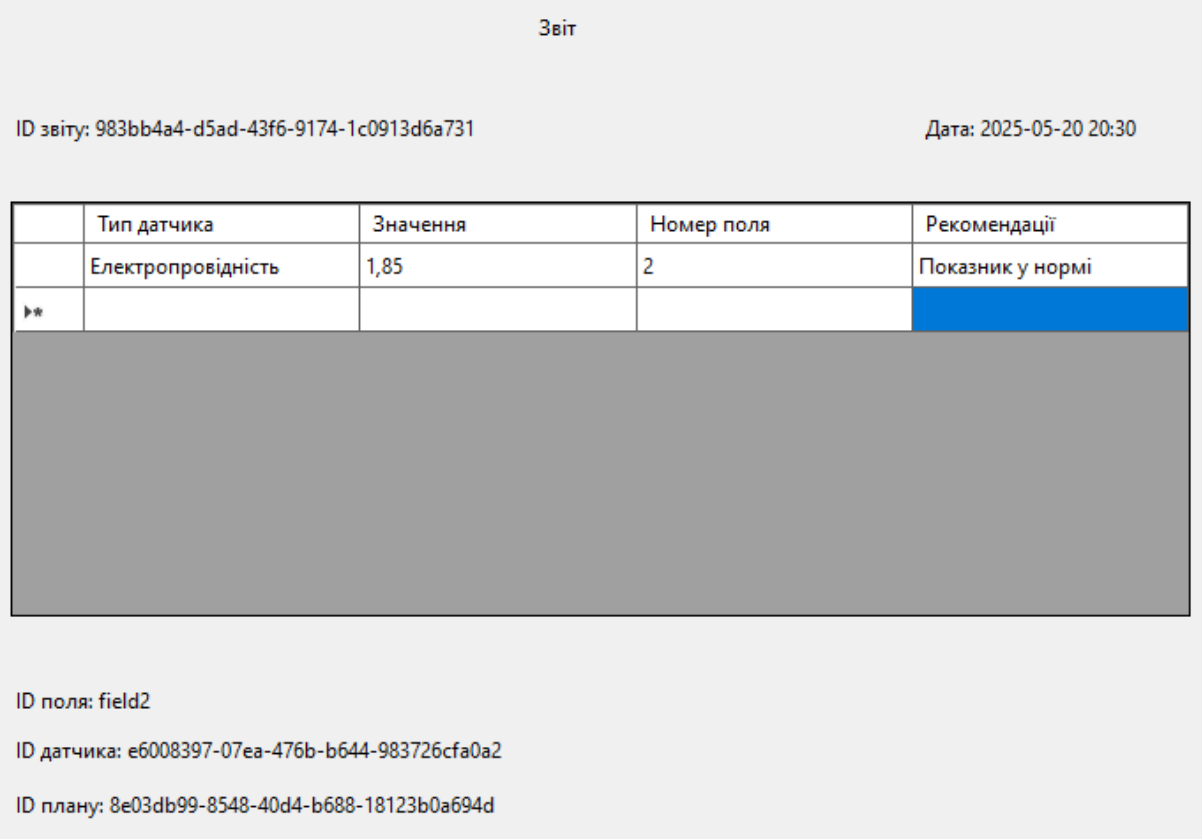
Звіт
ID звіту: 983bb4a4-d5ad-43f6-9174-1c0913d6a731
Дата: 2025-05-20 20:30
Тип датчика
Значення
Номер поля
Рекомендації
Електропровідність
1,85
2
Показник у нормі
ID поля: field2
ID датчика: e6008397-07ea-476b-b644-983726cfa0a2
ID плану: 8e03db99-8548-40d4-b688-18123b0a694d

Рис 4.11 Звіт

Коректність даних

Дані планів в базі Рис 4.12 повністю збігаються з даними в таблиці Рис 4.8. Єдина відмінність між рисунками в тому що базі знаходяться ще два старих плана добрив, які не відображаються на формі Плани добрив.

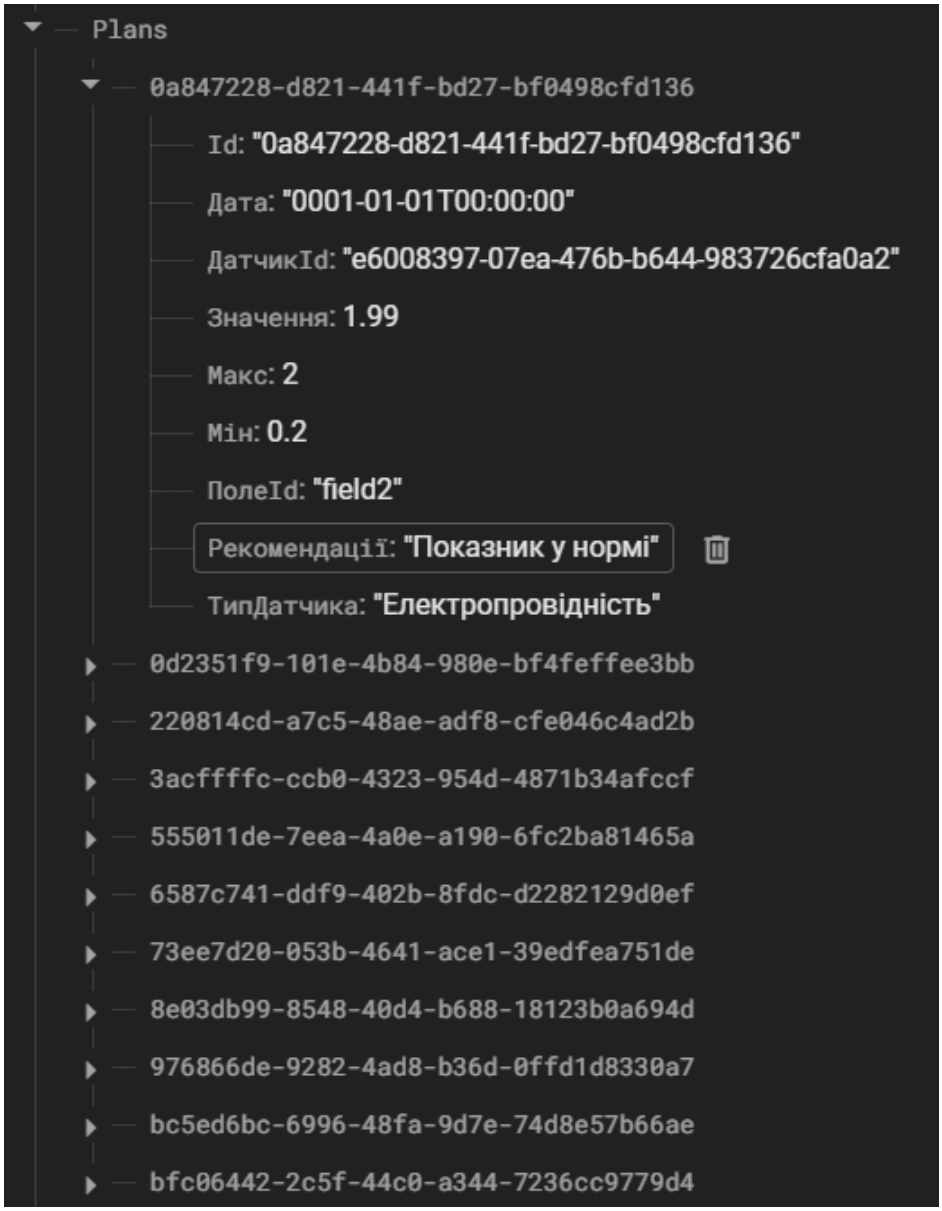

Рис 4.12 Список планів у базі

Дані звіту в базі Рис 4.13 повністю збігаються з даними звіту в програмі Рис 4.11.

Рис 4.13 Звіт у базі даних

Дані полів з бази Рис 4.14 збігаються з даними в програмі Рис 4.4.

Рис 4.14 Поля в базі даних

Дані датчиків Рис 4.15 також збігаються з даним з програми Рис 4.3.

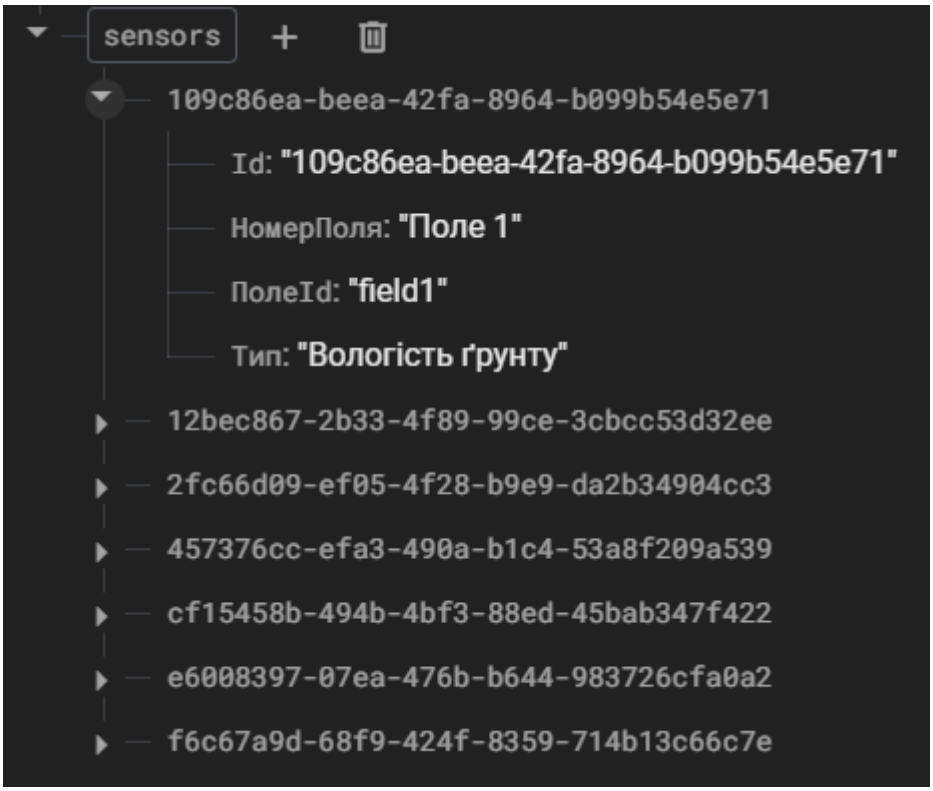

Рис 4.15 Датчики в базі даних

Функціональне тестування

Функціональне тестування полягає в тому щоб перевірити систему на відповідність до вимог користувача:

- Створення, перегляд та видалення датчиків і створення полів;
- Створення планів добрив;
- Виділення порушень кольором;
- Формування та перегляд звітів;
- Перевірка меню навігації;

Тестові сценарії

ІД	Назва сценарію	Вхідні дані	Очікуваний результат	Результат
TC1	Додавання нового датчика	Тип = тип датчику Поле = №	Датчик збережено у базі даних та відображено у таблиці	Успішно
TC2	Створення плану добрив	Всі датчики	План добрив створено, відображено результати датчиків та рекомендації щодо добрив	Успішно
TC3	Створення звіту	Дані про поля, датчики та плани	Звіт створено на основі вибраного поля	Успішно
TC4	Перегляд звіту	Список звітів	Відкрито форму із вибраним звітом з списку усіх звітів	Успішно
TC5	Кольорові відхилення	Значення датчиків	План з нормальним результатом закрашений зеленим, план з перевищенням норми закрашений червоним, план з результатами нижче норми горить синім	Успішно

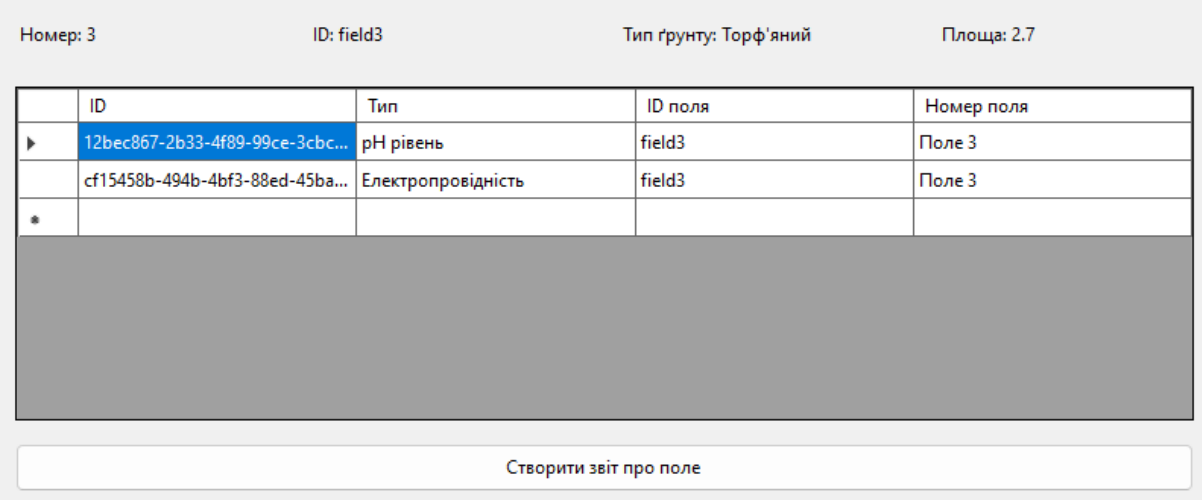

Рис 4.16 ТС1 Список датчиків на полі 3

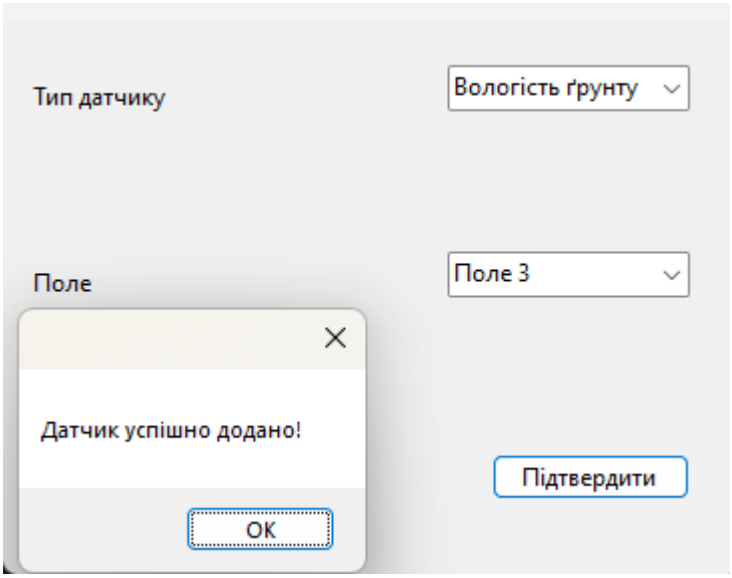

Рис 4.17 ТС1 Створення датчику на полі 3

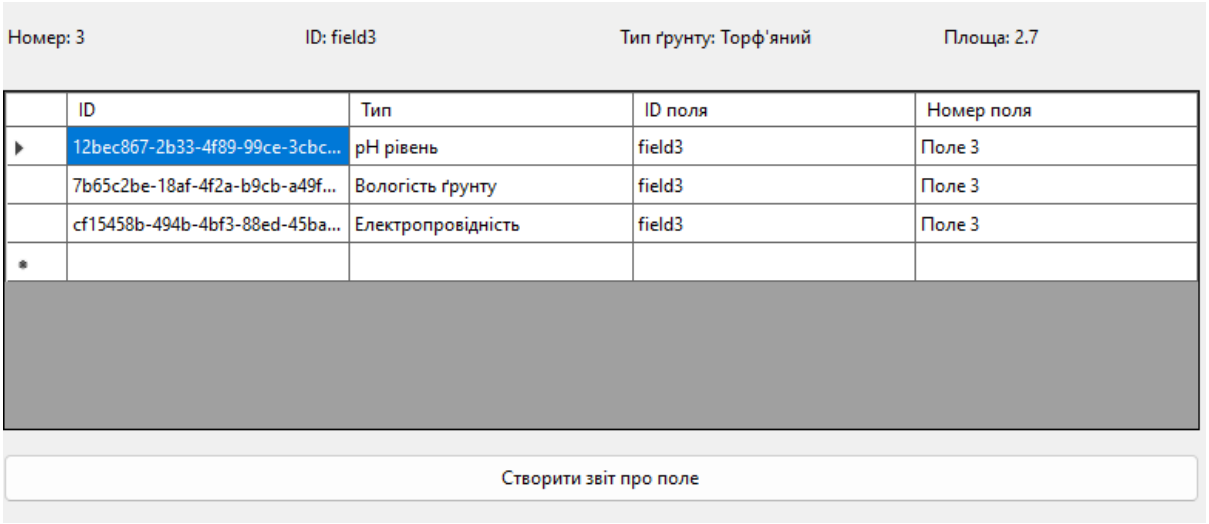

Рис 4.18 ТС1 Датчик створений на полі 3

Рис 4.19 ТС2 Створення планів

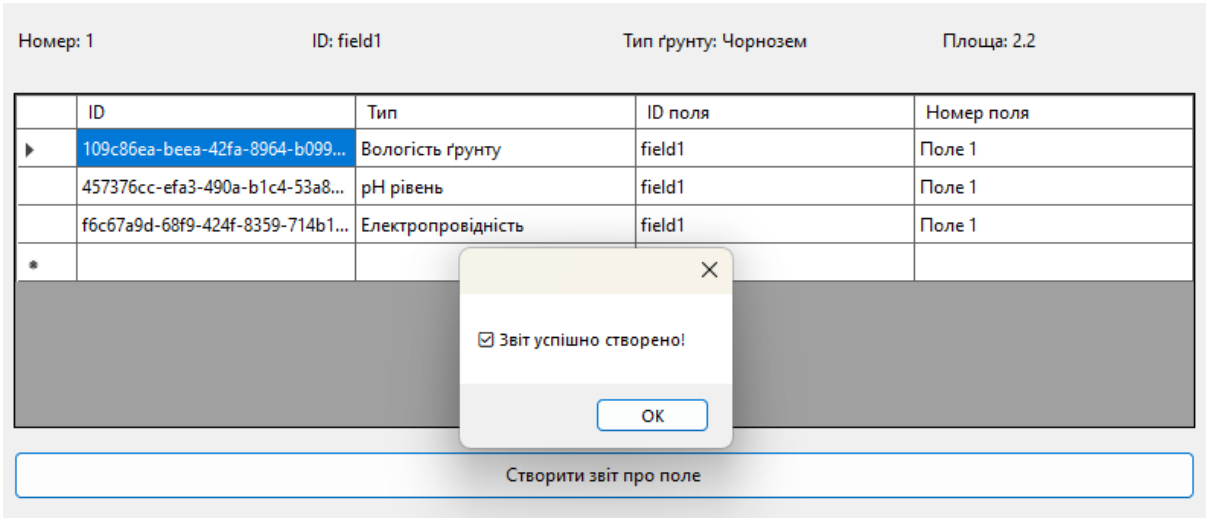

Рис 4.20 ТС3 Створення звіту

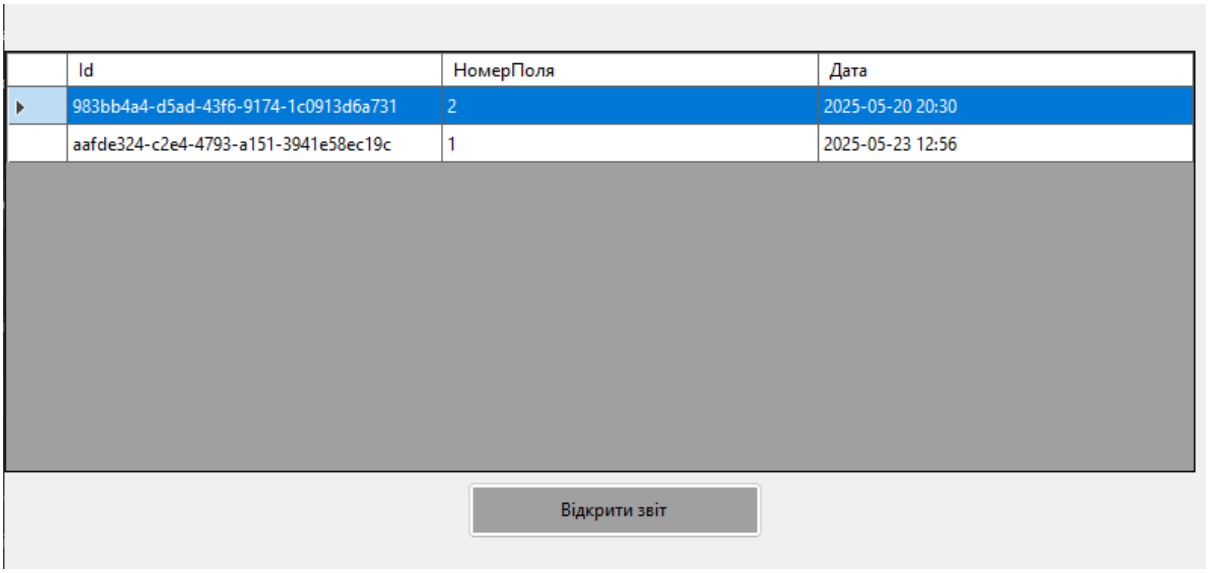

Рис 4.21 ТС4 список звітів

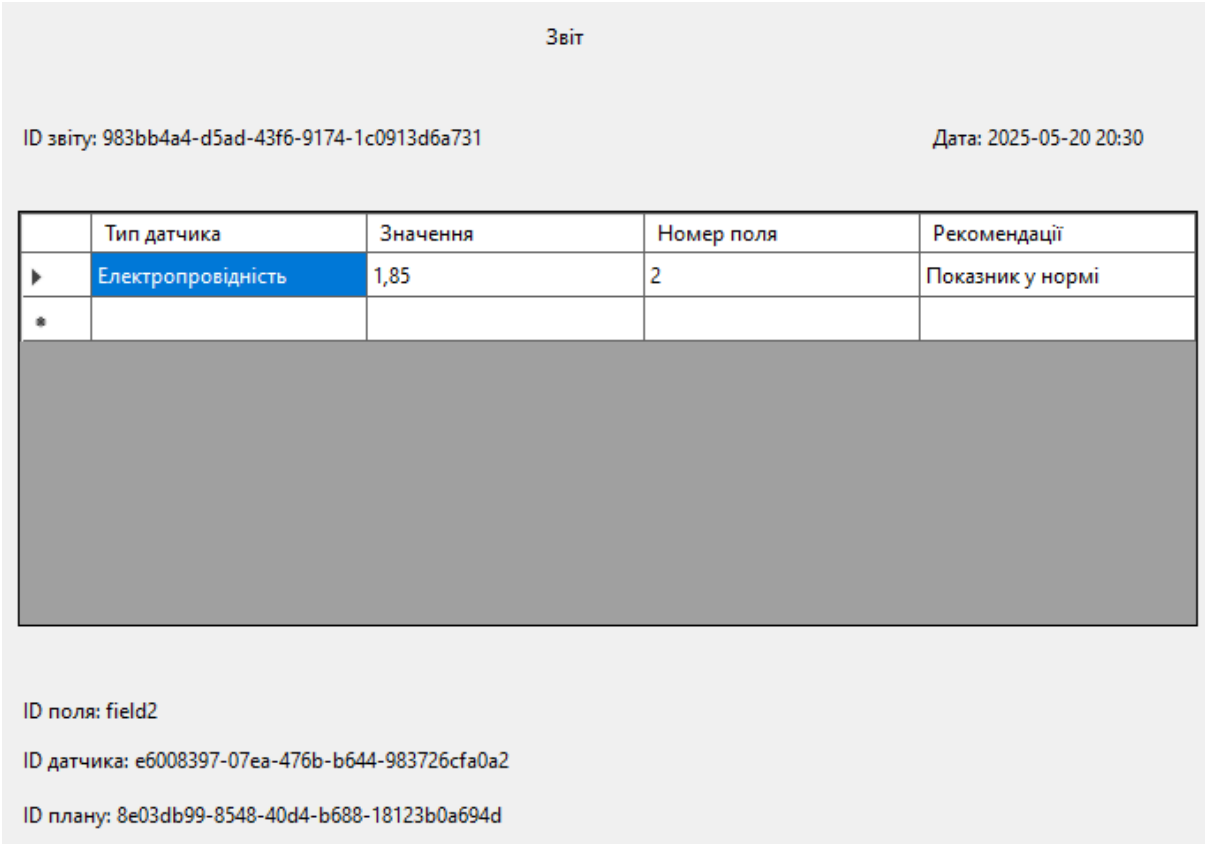

Рис 4.22 ТС4 Звіт

Рис 4.23 ТС5 Кольорове закрашування планів

Результати

Функціональні модулі працюють згідно до вимог. Підчас тестування було виявлено кілька незначних помилок, одна з яких була в неправильному відображені даних у таблиці на формі Звіт. Всі помилки були виправлені.

4.2 Вимоги до апаратного та програмного забезпечення

Щоби програмне забезпечення система моніторингу агрохіманалізу працювала коректно, потрібно притримуватися певних вимог до конфігурації комп’ютера.

Таблиця 4.1

Апаратні вимоги

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор (CPU)	2-ядерний, 1.8 ГГц	4-ядерний, 2.5 ГГц
Оперативна пам’ять (RAM)	4 ГБ	8 ГБ
Місце на диску	500 МБ	1 ГБ
З’єднання з Інтернетом	Обов’язкове (для роботи з Firebase)	Швидке постійне підключення

Таблиця 4.2

Програмні вимоги

Програмне забезпечення	**Версія**
Операційна система	Windows 8 / 10 / 11 (x64)
.NET Framework	Версія 4.7.2 або вище
Драйвери	Оновлені мережеві драйвери

4.3 Склад інсталяційного пакету

Інсталяційний пакет програмного забезпечення системи моніторингу агрохіманалізу складається з компонентів:

№	**Назва файлу/папки**	**Опис**
1	Setup.exe	Головний інсталятор для встановлення програми
2	ApplicationFile\	Каталог, що містить всі необхідні бібліотеки і ресурси додатку
4	readme.txt	Інструкція з авторизації
7	Diplom.application	Програма

Створення інсталяційного пакету було виконано за допомогою функції у Visual Studio.

- Відкрити Publish Selection , Рис 4.25. Build - Publish Selection.
- Створити профіль, Рис 4.26. New Profile – Folder Рис 4.26 – ClickOnce Рис 4.27 - вибрати шлях встановлення Рис 4.28 - From a CD, DVD, or USB drive Рис 4.29 - вибрати версію Рис 4.30 - Рис 4.31 - Рис 4.32 пропустити.

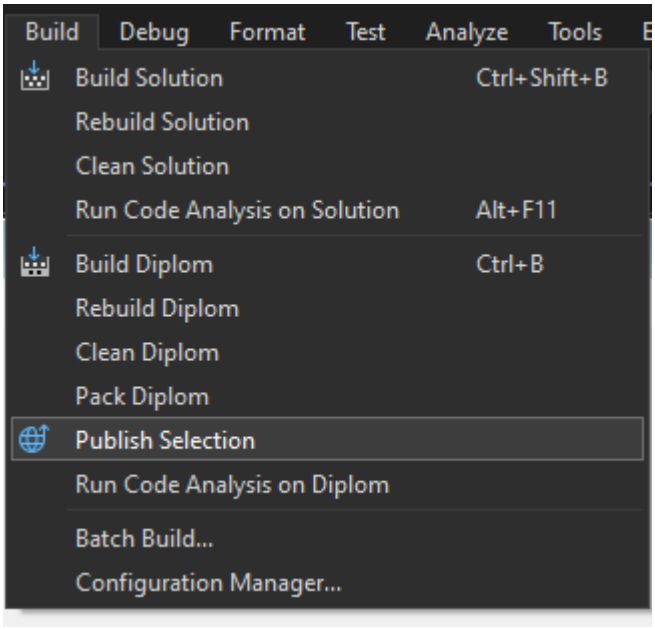

Рис 4.25 Створення пакету - публікування

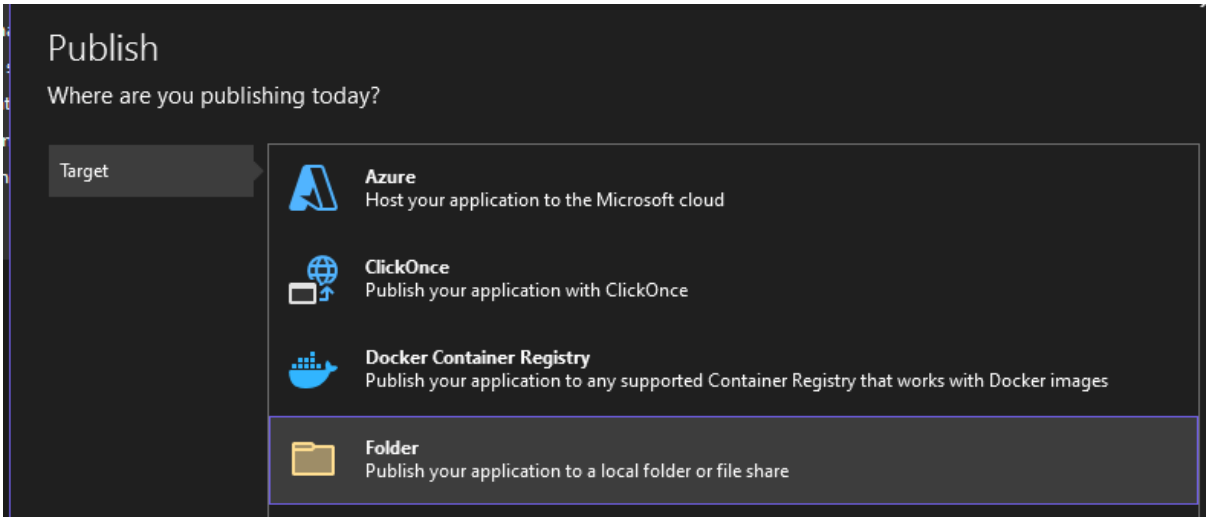

Рис 4.26 Folder

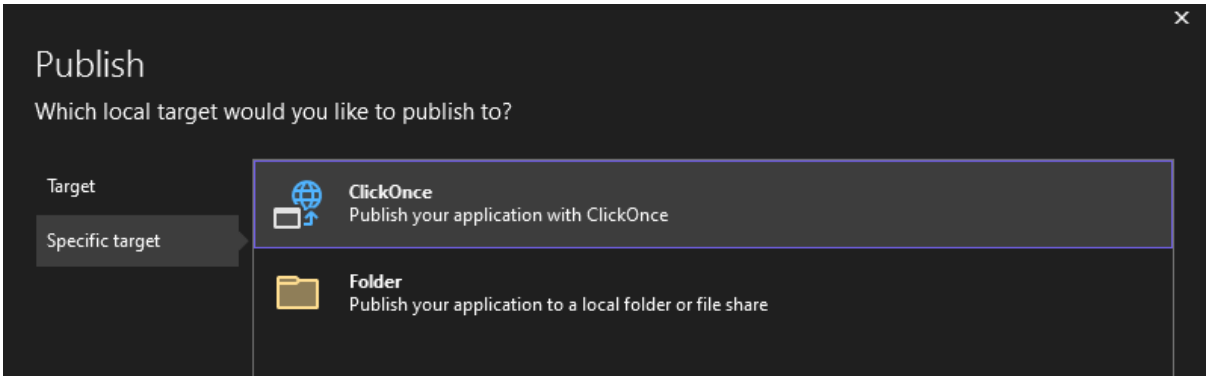

Рис 4.27 ClickOnce

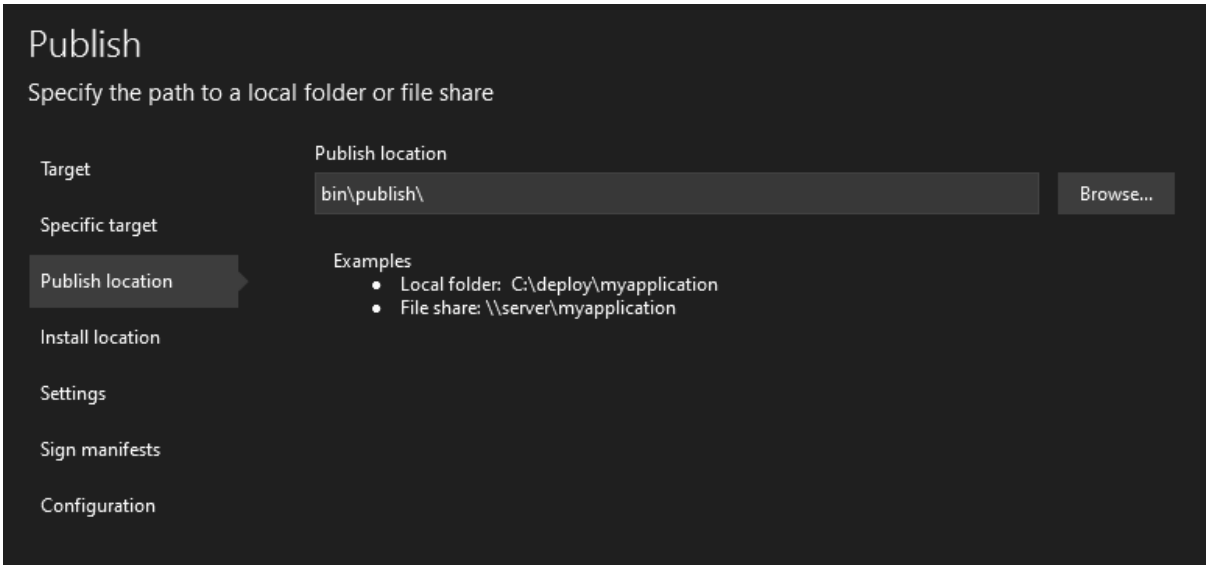

Рис 4.28 Шлях встановлення

Рис 4.29 From a CD, DVD, or USB drive

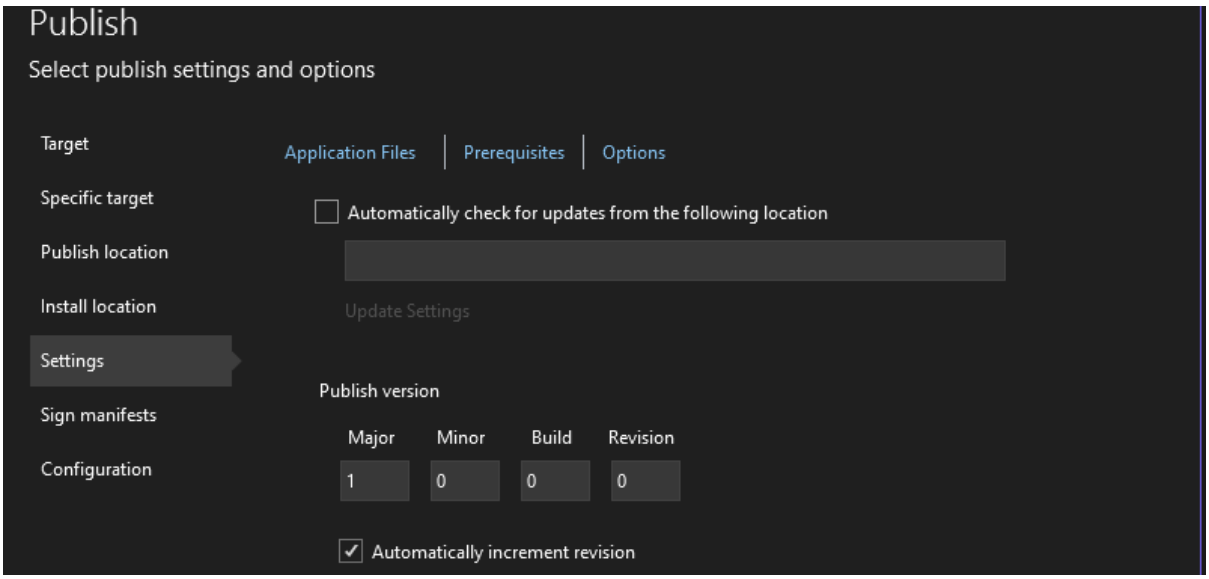

Рис 4.30 Версія пакету

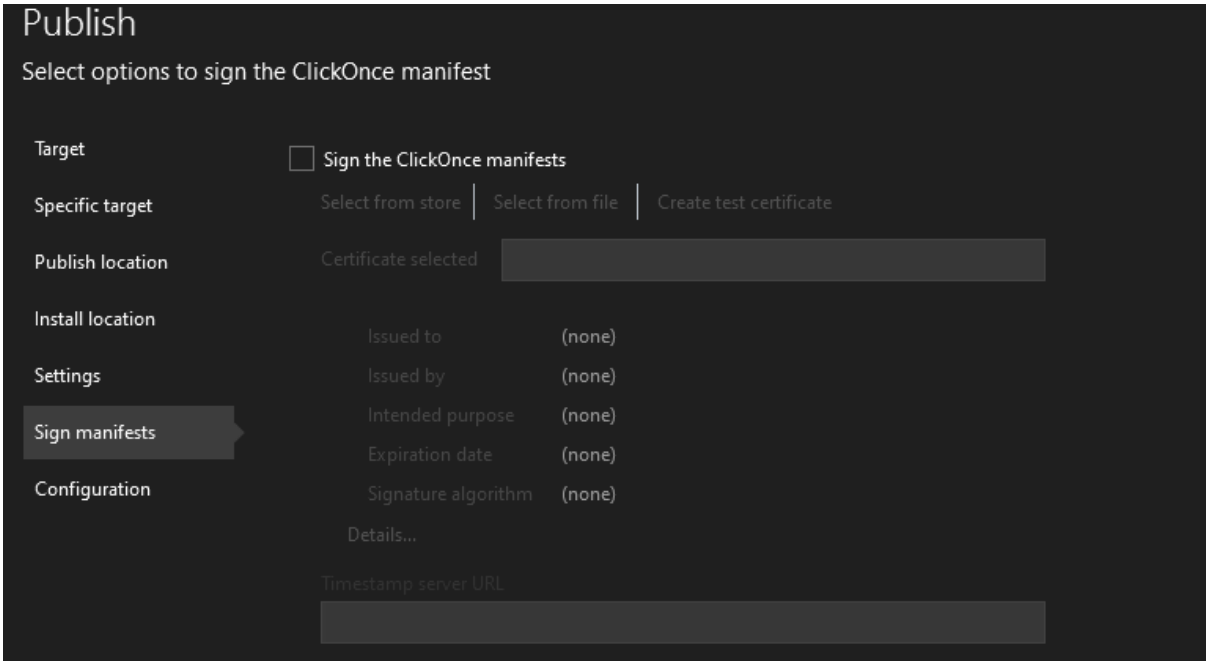

Рис 4.31 Пропустити

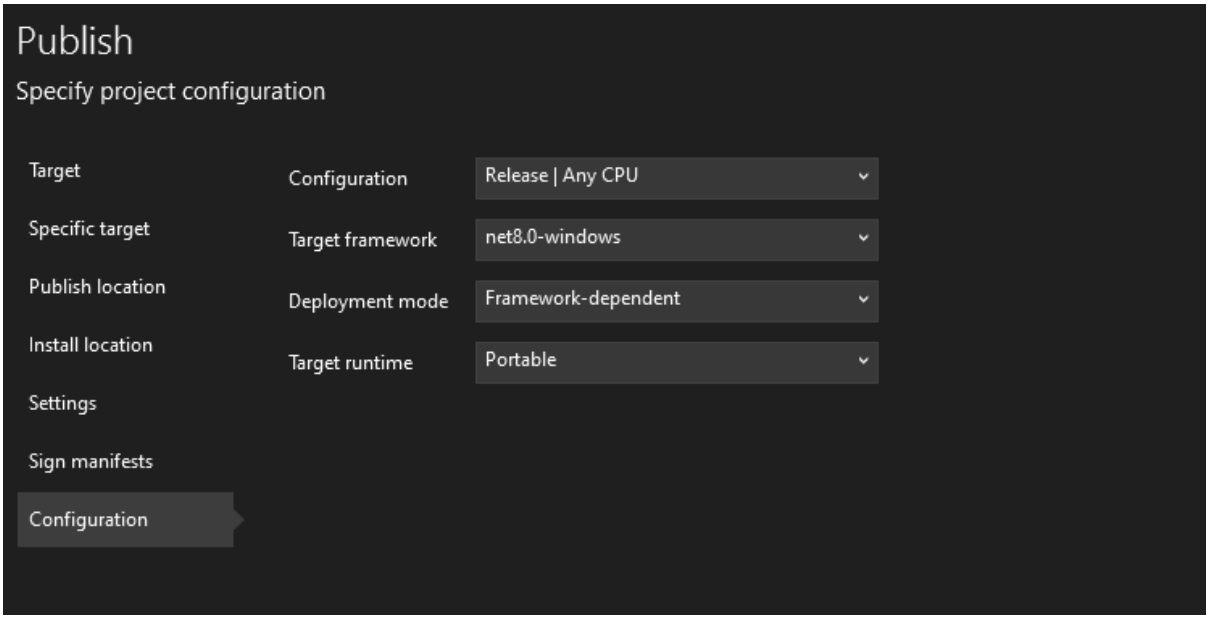

Рис 4.32 Фініш

Для завершення створення інсталяційного пакету потрібно натиснути Publish Рис 4.33.

Рис 4.33 Публікація

Висновок

У ході виконання дипломного проєкту було реалізовано програмне забезпечення системи моніторингу агрохіманалізу, яке забезпечує збір даних, їх обробку, аналіз та збереження, також формування планів добрива та звітів.

Основними досягненнями є:

- Інтеграція з Firebase;
- Зручний користувацький інтерфейс;
- Формування планів добрив;
- Формування звітів;
- Візуальне виділення відхилень;

Дана система дозволить аграріям приймати обґрунтовані рішення щодо добрив, та підвищувати ефективність роботи.
Впровадження такого рішення буде позитивно впливати на розвиток цифрового землеробства в Україні.

Список використаних джерел

Вікіпедія:

Вікіпедія діаграма класів - https://uk.wikipedia.org/wiki/Діаграма_класів

Вікіпедія UML діаграма - https://uk.wikipedia.org/wiki/Unified_Modeling_Language

Вікіпедія діаграма пакетів - https://uk.wikipedia.org/wiki/Діаграма_пакетів

Вікіпедія Windows Forms - https://uk.wikipedia.org/wiki/Windows_Forms

Вікіпедія .NET Framework - https://uk.wikipedia.org/wiki/.NET_Framework

Вікіпедія Firebase - https://uk.wikipedia.org/wiki/Firebase

Сайти:

FieldView - https://climate.com/en-us.html

Soft.Farm - https://soft.farm/uk

Статі:

Рекомендації за результатами аналізу ґрунту — експерт назвав фактори, які слід враховувати - https://superagronom.com/news/18362-rekomendatsiyi-za-rezultatami-analizu-gruntu--ekspert-nazvav-faktori-yaki-slid-vrahovuvati?

Книги

Системи точного землеробства - Силабус навчальної дисципліни «Адаптивні системи землеробства» - Гудзь В.П. 2020 - vkfkadapsyszeml24.pdf

Методи агрохімічного аналізу ґрунту - «Агроекологічний супутниковий моніторинг» - Кучма Т.Л. 2019 - https://www.agroeco.org.ua/wp-content/uploads/Publications/Monography/agroecologichniy%20suputnikoviy%20monitirind.pdf

Інформаційні системи в аграрній галузі- «Концептуальні оцінки реалізації засад інклюзивного розвитку сільських територій за участі агрохолдингових інтегрованих формувань» - https://www.researchgate.net/publication/353925767_Sonceptual_assessment_of_the_implementation_of_the_principles_of_inclusive_rural_areas_development_with_the_participation_of_agroholding_integrated_formations - Ігнатенко М.М. 2021