МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ АГРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ

ВІДДІЛЕННЯ
«ІНЖЕНЕРНА ІНФРАСТРУКТУРА ТА КОМП’ЮТЕРНІ НАУКИ»

ЦИКЛОВА КОМІСІЯ СПЕЦІАЛЬНОСТІ
«КОМП'ЮТЕРНІ НАУКИ»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня фаховий молодший бакалавр
за спеціальністю 122 Комп'ютерні науки
на тему:

«Інтерактивний довідково-аналітичний каталог агронома»

Виконав здобувач освіти групи П-41
Лаціна Андрій Олександрович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:

Житомир – 2026

Зміст

РЕФЕРАТ

Пояснювальна записка: 60 аркушів, 21 рисунок., 14 таблиць, 2 додатоки, 25 джерел

У роботі проведено аналіз предметної галузі та огляд існуючих програмних рішень для агрономів. Обґрунтовано вибір технологічного стеку та архітектурного рішення. Виконано проєктування системи з розробкою комплексу UML-діаграм та бази даних SQLite. Реалізовано програмний додаток з картковим відображенням каталогів, підтримкою фотографій, аналітичними графіками (LiveChartsCore) та експортом CSV-звітів. Проведено функціональне тестування (36 тест-кейсів, 89% Passed) та аналіз продуктивності.

Результатом роботи є готовий до використання автономний настільний додаток «АгрономКаталог» для операційної системи Windows, що не потребує підключення до мережі Інтернет та забезпечує агроному зручний доступ до структурованої довідкової інформації.

Ключові слова: довідково-аналітична система, агроном, WPF, MVVM, .NET 9, Entity Framework Core, SQLite, LiveCharts, каталог культур, добрива, шкідники.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АПК	Агропромисловий комплекс
БД	База даних
БПЛА	Безпілотний літальний апарат
ВВП	Валовий внутрішній продукт
DI	Dependency Injection — впровадження залежностей
EF Core	Entity Framework Core — об'єктно-реляційний ORM-фреймворк
ER	Entity-Relationship — «сутність-зв'язок» (тип діаграм)
IoT	Internet of Things — інтернет речей
IoC	Inversion of Control — інверсія керування
LINQ	Language Integrated Query — мова запитів, інтегрована у C#
MVVM	Model-View-ViewModel — архітектурний патерн
MVP	Model-View-Presenter — архітектурний патерн
MVC	Model-View-Controller — архітектурний патерн
ORM	Object-Relational Mapping — об'єктно-реляційне відображення
ПЗ	Програмне забезпечення
СУБД	Система управління базами даних
UML	Unified Modeling Language — уніфікована мова моделювання
UI	User Interface — інтерфейс користувача
UX	User Experience — досвід користувача
WPF	Windows Presentation Foundation — UI-фреймворк Microsoft
XAML	Extensible Application Markup Language — мова розмітки WPF
CSV	Comma-Separated Values — формат текстових файлів даних
RAM	Random Access Memory — оперативна пам'ять
GPU	Graphics Processing Unit — графічний процесор
API	Application Programming Interface — програмний інтерфейс

ВСТУП

Агропромисловий комплекс є однією з провідних галузей економіки України, яка забезпечує продовольчу безпеку країни та формує значну частку валового внутрішнього продукту. В умовах стрімкого розвитку інформаційних технологій та цифровізації виробничих процесів особливої актуальності набуває впровадження спеціалізованих програмних засобів у діяльність фахівців агрономічної галузі.

Сучасний агроном щоденно працює з великими масивами даних: каталогами сільськогосподарських культур, нормативами внесення добрив, переліками шкідників та хвороб рослин, рекомендаціями щодо засобів захисту. Традиційні паперові довідники та розрізнені електронні таблиці не забезпечують необхідного рівня оперативності доступу до інформації, зручності пошуку та аналітичних можливостей. Відсутність інтегрованого інструменту для систематизації агрономічних знань призводить до зниження ефективності прийняття управлінських рішень у сфері землеробства.

Розробка інтерактивного довідково-аналітичного каталогу агронома є відповіддю на зазначені проблеми. Подібна система дозволяє структурувати та централізовано зберігати довідкову інформацію, забезпечувати швидкий пошук і фільтрацію даних, а також формувати аналітичні звіти та візуалізувати агрономічні показники у зручному графічному вигляді. Все це зумовлює практичну значущість і своєчасність обраної теми кваліфікаційної роботи.

Метою кваліфікаційної роботи є проєктування та розробка інтерактивного довідково-аналітичного каталогу агронома — настільного програмного додатку, що забезпечує зберігання, пошук, редагування та аналіз даних про сільськогосподарські культури, добрива, шкідники та хвороби рослин.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. проаналізувати предметну область та визначити основні вимоги до функціональності системи;

2. дослідити існуючі програмні рішення в галузі автоматизації агрономічної діяльності та обґрунтувати доцільність розробки власного програмного засобу;
3. обрати оптимальний технологічний стек та архітектурне рішення для реалізації системи;
4. спроєктувати структуру бази даних і розробити концептуальну модель системи з використанням UML-нотації;
5. реалізувати програмний додаток на платформі .NET 9 з використанням технологій WPF, Entity Framework Core та архітектурного патерну MVVM;
6. розробити зручний та сучасний інтерфейс користувача з підтримкою карткового відображення даних, діалогових вікон редагування та вбудованої довідки;
7. реалізувати аналітичний модуль з інтерактивними графіками та можливістю експорту звітів у форматі CSV;
8. провести функціональне тестування розробленої системи та проаналізувати результати.

Об'єктом дослідження є процеси збору, зберігання, систематизації та аналізу агрономічної довідкової інформації в умовах сільськогосподарського виробництва.

Предметом дослідження є методи та засоби проєктування і розробки інтерактивних настільних довідково-аналітичних систем на платформі .NET 9 з використанням архітектурного патерну MVVM та технології WPF.

У процесі виконання кваліфікаційної роботи використовувались такі методи дослідження:

- системний аналіз — для дослідження предметної галузі та визначення вимог до системи;
- порівняльний аналіз — для оцінки існуючих програмних рішень і вибору технологічного стеку;

- об'єктно-орієнтоване проєктування — для розробки архітектури системи та моделювання структури класів;
- UML-моделювання — для побудови діаграм варіантів використання, класів, послідовності та компонентів;
- методи проєктування реляційних баз даних — для розробки структури бази даних SQLite;
- функціональне тестування — для перевірки коректності роботи реалізованих функцій системи.

Практична значущість кваліфікаційної роботи полягає у створенні функціонального програмного продукту, який може бути безпосередньо використаний у професійній діяльності агрономів, фермерів та фахівців агропромислових підприємств.

Розроблена система «АгрономКаталог» забезпечує централізоване зберігання та зручний доступ до відомостей про десять основних сільськогосподарських культур, двадцять видів мінеральних та органічних добрив, п'ятнадцять поширених шкідників і хвороб рослин. Вбудований аналітичний модуль дозволяє порівнювати норми внесення і ціни добрив у графічному вигляді, а функція експорту CSV-звітів — інтегрувати дані системи до зовнішніх таблиць Microsoft Excel.

Застосування сучасних технологій розробки (.NET 9, WPF, Entity Framework Core, CommunityToolkit.Mvvm, LiveChartsCore) забезпечує надійність, масштабованість та простоту розширення функціональності системи у майбутньому. Карткове відображення даних із підтримкою фотографій культур і шкідників підвищує наочність та інформативність довідкового матеріалу.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 60 аркушів друкованого тексту. Робота містить 21 рисунок, 14 таблиць та 6 лістингів програмного коду.

У першому розділі проведено аналіз предметної області, досліджено стан інформатизації агропромислового комплексу України, виконано огляд та порівняльний аналіз існуючих програмних рішень, сформульовано вимоги до розроблюваної системи та обґрунтовано вибір технологій.

У другому розділі виконано проєктування системи: розроблено архітектурне рішення на основі патерну MVVM, спроєктовано реляційну базу даних, побудовано UML-діаграми класів, варіантів використання, послідовності та компонентів, а також розроблено макети інтерфейсу користувача.

У третьому розділі описано практичну реалізацію всіх модулів системи, наведено ключові фрагменти програмного коду з поясненнями, представлено знімки екранів готового додатку, проведено функціональне тестування та аналіз продуктивності системи.

У висновках підведено підсумки виконаної роботи, оцінено ступінь досягнення поставленої мети та окреслено перспективи подальшого розвитку системи.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Сучасний стан інформатизації агропромислового комплексу України

Агропромисловий комплекс (АПК) України є однією з провідних галузей національної економіки. За даними Державної служби статистики України, аграрний сектор формує від 10 до 14 відсотків ВВП країни та забезпечує понад 40 відсотків загального обсягу експорту. Щорічно в Україні вирощується близько 80 мільйонів тонн зернових та зернобобових культур, що ставить країну в ряд провідних світових виробників і експортерів аграрної продукції.

Незважаючи на значний природній потенціал, ефективність агропромислового виробництва суттєво залежить від рівня технологічної та інформаційної оснащеності галузі. Поступова цифровізація АПК — одна з ключових тенденцій останнього десятиліття як у світі, так і в Україні. Під цифровізацією агропромислового комплексу розуміють широке впровадження інформаційно-комунікаційних технологій у процеси виробництва, зберігання, переробки та збуту сільськогосподарської продукції.

Серед основних напрямів цифровізації АПК можна виділити такі:

- точне землеробство (Precision Agriculture) — використання GPS-навігації, дронів, супутникового моніторингу та сенсорних систем для оптимізації обробітку ґрунту;
- інтернет речей (IoT) в агросфері — інтелектуальні датчики вологості ґрунту, метеостанції, автоматизовані системи зрошення;
- спеціалізоване програмне забезпечення — системи обліку посівних площ, управління добривами, моніторингу шкідників та хвороб;
- мобільні додатки для агрономів — забезпечення оперативного доступу до агрономічних баз знань безпосередньо в полі;
- аналітичні платформи — обробка великих масивів агрономічних даних, прогнозування врожайності, побудова аналітичних звітів.

Впровадження інформаційних технологій у діяльність агронома дозволяє суттєво підвищити ефективність прийняття управлінських рішень. Зокрема, використання спеціалізованих довідково-аналітичних систем надає агроному змогу оперативно отримувати інформацію про норми внесення добрив, ознаки ураження рослин шкідниками та хворобами, рекомендовані засоби захисту — без необхідності звертатися до паперових довідників або встановлювати складні корпоративні системи.

На рисунку 1.1 представлено структуру основних напрямів цифровізації агропромислового комплексу України.

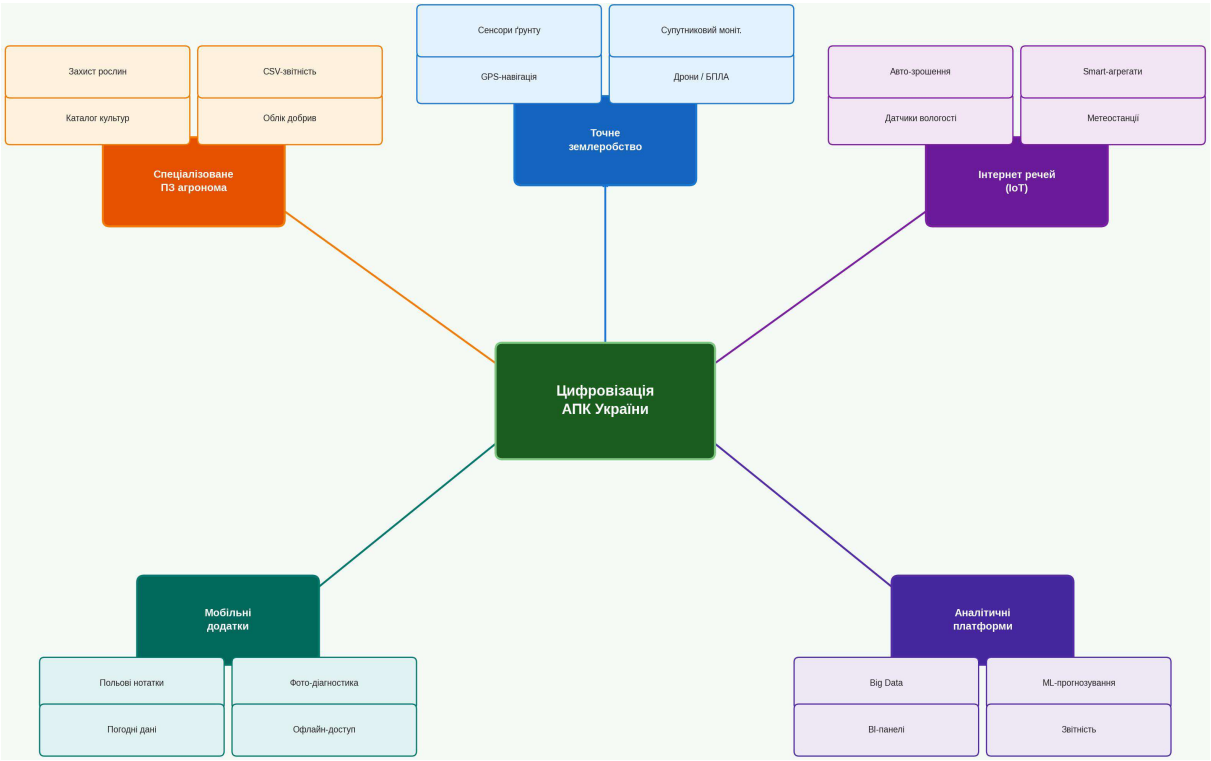

Рисунок 1.1 — Основні напрями цифровізації агропромислового комплексу України

Разом із тим, рівень цифровізації малих і середніх агропідприємств в Україні залишається недостатньо високим. Основними перешкодами є висока вартість комплексних ERP-систем, необхідність постійного підключення до мережі Інтернет для хмарних рішень, а також низький рівень технічної грамотності частини аграріїв. Саме тому актуальним залишається розробка компактних, зручних та доступних настільних додатків, орієнтованих безпосередньо на потреби практикуючого агронома.

1.2 Аналіз існуючих програмних рішень для агрономів

На ринку програмного забезпечення для агропромислового комплексу представлений широкий спектр рішень — від спеціалізованих мобільних додатків до повнофункціональних ERP-систем. Для коректного обґрунтування доцільності розробки власного програмного засобу необхідно провести детальний аналіз існуючих рішень.

Серед найбільш відомих програмних продуктів у цій категорії слід виділити наступні.

Агроном.Про — вітчизняний веб-сервіс, орієнтований на планування агрохімічних заходів. Дозволяє вести облік посівних площ, планувати внесення добрив та формувати агрономічні журнали. Суттєвим недоліком є обов'язкова реєстрація та постійне підключення до мережі Інтернет, що унеможливлює роботу в польових умовах без стабільного зв'язку.

АгроПлан — програмний комплекс для автоматизації обліку в сільськогосподарських підприємствах. Містить широкий функціонал: облік техніки, персоналу, землі, добрив. Проте система розрахована на підприємства рівня агрохолдингу, є складною у налаштуванні та потребує навчання персоналу, що робить її надмірно громіздкою для окремого агронома.

Trimble Ag Software — комплексна платформа точного землеробства від американської компанії Trimble. Інтегрується з GPS-обладнанням і польовою технікою, підтримує картографування полів та аналіз супутникових знімків. Система орієнтована на великі агропідприємства, має значну вартість ліцензії та повністю локалізована лише для ринків США і Австралії.

FarmLogs (США) — популярна хмарна платформа для ферм, яка надає можливості ведення польових журналів, відстеження погодних умов та базового обліку витрат. Відсутній модуль обліку добрив у розрізі культур та засобів захисту від шкідників.

1С:Сільське господарство — облікова система на базі платформи 1С:Підприємство. Містить розширений модуль агрохімічного обліку та управління добривами, проте потребує окремого сервера, ліцензій для

кожного робочого місця та кваліфікованого обслуговування системним адміністратором.

У таблиці 1.1 наведено порівняльний аналіз розглянутих систем за ключовими функціональними критеріями.

Таблиця 1.1 — Порівняльний аналіз існуючих програмних рішень

Система	Тип	Каталог культур	Облік добрив	Аналітика /Звіти	Офлайн-режим
Агроном.Про (Україна)	Веб-додаток	Так	Частково	Так	Ні
АгроПлан (Росія)	Настільний	Так	Так	Обмежено	Так
Trimble Ag Software	Веб + мобільний	Так	Так	Так	Частково
FarmLogs (США)	Веб + мобільний	Так	Ні	Так	Ні
1С:Сільське господарство	Настільний	Частково	Так	Так	Так
AgronomCatalog (розроблений)	Настільний	Так	Так	Так	Так

Аналіз даних таблиці 1.1 свідчить про те, що більшість розглянутих систем або є надмірно складними для індивідуального використання агрономом, або потребують постійного підключення до мережі Інтернет, або не підтримують повного набору необхідних довідкових модулів в єдиному інтерфейсі. Розроблена у межах даної кваліфікаційної роботи система «АгрономКаталог» позбавлена зазначених недоліків і забезпечує повний офлайн-режим роботи з усіма трьома каталогами та аналітичним модулем.

1.3 Недоліки існуючих систем та обґрунтування необхідності розробки

На основі проведеного порівняльного аналізу можна систематизувати основні недоліки існуючих програмних рішень для агрономів:

1) Залежність від мережі Інтернет. Переважна більшість сучасних агрономічних систем побудована за принципом SaaS (Software as a Service) і потребує постійного підключення до хмарної інфраструктури.

Для агронома, який працює у полі поза зоною стабільного покриття мобільної мережі, це є критичним обмеженням.

2) Висока вартість і складність ліцензування. Комплексні ERP-рішення для АПК потребують придбання окремих ліцензій для кожного модуля та кожного робочого місця, що є фінансово недоступним для малих і середніх господарств.
3) Відсутність єдиного довідника культур, добрив і шкідників. Жодна з розглянутих систем не поєднує в єдиному інтерфейсі повноцінні взаємопов'язані каталоги сільськогосподарських культур, добрив і шкідників разом із засобами аналітики та формування звітів.
4) Надмірна функціональність. Системи, орієнтовані на агрохолдинги, містять сотні модулів (облік техніки, персоналу, фінансів тощо), які не потрібні практикуючому агроному-польовику і суттєво ускладнюють освоєння програми.
5) Відсутність підтримки фотографій та візуалізації. Більшість розглянутих систем не забезпечують перегляд фотографій культур і шкідників безпосередньо у записі довідника, що знижує наочність і практичну цінність системи.
6) Обмежена локалізація. Більшість закордонних продуктів не мають повноцінної локалізації українською мовою, що ускладнює їх використання вітчизняними фахівцями.

Таким чином, існує виразна незаповнена ніша для легкого, повністю офлайнового, україномовного настільного додатку, що забезпечує агроному єдину точку доступу до довідникової інформації про культури, добрива та шкідників, підкріпленої аналітичними інструментами. Саме таким рішенням є розроблена в межах даної кваліфікаційної роботи система «АгрономКаталог».

Порівняно з існуючими рішеннями, розроблений додаток пропонує такі конкурентні переваги:

– повна автономність роботи — жодних серверів, ліцензій і Інтернету не потрібно;

– єдиний інтерфейс для трьох взаємопов'язаних каталогів;
– сучасний картковий дизайн з підтримкою фотографій записів;
– вбудований аналітичний модуль із інтерактивними графіками;

1.4 Аналіз вимог до системи

Визначення вимог до програмної системи є ключовим етапом її проєктування. Вимоги поділяються на функціональні (що система повинна робити) та нефункціональні (яким чином система повинна це робити).

Функціональні вимоги до системи «АгрономКаталог» сформовано на основі аналізу потреб цільової аудиторії — агрономів і фахівців сільськогосподарських підприємств. У таблиці 1.2 представлено повний перелік функціональних вимог.

Таблиця 1.2 — Функціональні вимоги до системи «АгрономКаталог»

№	Назва вимоги	Опис
1	Каталог культур	Система повинна зберігати інформацію про с/г культури: назву, опис, норми сівби, рекомендовані добрива, шкідників та фотографію.
2	Каталог добрив	Система повинна містити довідник добрив із типами (N, P, K, NPK, органічні), нормами внесення та цінами.
3	Каталог шкідників та хвороб	Система повинна зберігати дані про шкідників і хвороби рослин: симптоми, засоби захисту та фотографію.
4	CRUD-операції	Користувач повинен мати можливість додавати, переглядати, редагувати та видаляти записи в усіх каталогах.
5	Пошук та фільтрація	Система повинна забезпечувати пошук записів за назвою та ключовими словами у всіх розділах.
6	Аналітичний модуль	Система повинна відображати статистику та інтерактивні графіки норм і цін добрив.
7	Експорт даних	Система повинна підтримувати експорт каталогу добрив та аналітичних звітів у форматі CSV.
8	Підтримка зображень	Система повинна дозволяти прикріплювати фотографії до записів культур і шкідників.
9	Вбудована довідка	Система повинна містити розділ «Довідка» з описом функцій та інструкцією користувача.
10	Ініціалізація БД	При першому запуску система повинна автоматично створювати базу даних та заповнювати її демонстраційними даними.

Нефункціональні вимоги визначають якісні характеристики системи: продуктивність, надійність, зручність використання та умови розгортання. У таблиці 1.3 наведено нефункціональні вимоги до розроблюваної системи.

Таблиця 1.3 — Нефункціональні вимоги до системи «АгрономКаталог»

Категорія	Вимога	Метрика / критерій
Продуктивність	Час відгуку	Завантаження списку до 500 записів — не більше 1 секунди
Надійність	Збереження даних	Відсутність втрати даних при позаштатному завершенні роботи
Зручність	Засвоюваність	Новий користувач має освоїти основні функції за 15 хвилин
Переносність	ОС	Запуск на Windows 10 / 11 (x64) без додаткових налаштувань
Локалізація	Мова інтерфейсу	Увесь інтерфейс — українська мова
Масштабованість	Обсяг БД	Коректна робота при кількості записів до 10 000 у кожному каталозі
Безпека	Цілісність даних	Валідація обов'язкових полів у формах до збереження
Розгортання	Установка	Запуск без інсталятора; БД створюється автоматично при першому старті

Аналіз вимог дозволяє стверджувати, що система повинна бути реалізована як компактний настільний додаток для операційної системи Windows, з вбудованою базою даних, україномовним інтерфейсом і відсутністю залежностей від зовнішніх сервісів. Дані вимоги безпосередньо обумовлюють вибір технологічного стеку розробки.

1.5 Вибір технологій та інструментів розробки

Вибір технологічного стеку є визначальним рішенням, яке впливає на всі наступні етапи розробки програмної системи. При виборі технологій для реалізації системи «АгрономКаталог» враховувалися такі критерії: відповідність нефункціональним вимогам, доступність якісної документації та бібліотек, підтримка архітектурного патерну MVVM, можливість роботи без мережевого підключення.

Вибір UI-фреймворку. Для реалізації графічного інтерфейсу розглядалося чотири альтернативи: WPF, WinForms, MAUI та Avalonia UI. У таблиці 1.4 наведено порівняльний аналіз цих технологій.

Таблиця 1.4 — Порівняльний аналіз UI-фреймворків для розробки настільних додатків

Критерій	WPF (.NET 9)	WinForms (.NET 9)	MAUI	Avalonia UI	Пояснення переваги WPF
Підтримка MVVM	✓ Повна	△ Обмежена	✓ Повна	✓ Повна	Чистий MVVM з data binding, RelayCommand, ObservableObject
Можливості UI	Широкі	Базові	Широкі	Широкі	XAML, стилі, шаблони, анімації, custom controls
Продуктивність	Висока	Висока	Середня	Висока	Апаратне прискорення DirectX, відсутність overhead кросплатформи
Кросплатформність	Windows	Windows	Win/Mac/ iOS/Android	Win/Mac /Linux	Для настільного агрономічного ПЗ — Windows достатньо
Зрілість екосистеми	Висока (2006)	Дуже висока	Низька (2022)	Середня	Велика кількість бібліотек: LiveCharts, MaterialDesign
Простота освоєння	Середня	Висока	Середня	Середня	Рясна документація, активна спільнота

За результатами аналізу таблиці 1.4 для реалізації системи обрано WPF (Windows Presentation Foundation) на платформі .NET 9. Основними аргументами на користь цього вибору є: зрілість і стабільність технології (понад 17 років активного розвитку), повноцінна підтримка архітектурного патерну MVVM через механізм прив'язки даних (data binding), широка

екосистема бібліотек компонентів та графічних засобів, апаратне прискорення рендерингу через DirectX.

На рисунку 1.2 зображено загальну схему взаємодії обраних технологій у складі системи «АгрономКаталог».

Рисунок 1.2 — Схема технологічного стеку системи «АгрономКаталог»

Вибір СУБД. Для зберігання даних розглядалися три варіанти: SQLite, PostgreSQL та MySQL. Порівняльний аналіз представлено в таблиці 1.5.

Таблиця 1.5 — Порівняльний аналіз систем управління базами даних

Критерій	SQLite	PostgreSQL	MySQL	Обґрунтування вибору SQLite
Тип	Вбудована	Клієнт-сервер	Клієнт-сервер	SQLite не потребує окремого сервера — ідеально для настільного ПЗ
Установка	Не потрібна	Складна	Середня	База даних — один файл .db поряд із exe
Продуктивність (малі дані)	Відмінна	Відмінна	Відмінна	Для каталогів до 10 000 записів SQLite є оптимальним

Критерій	SQLite	PostgreSQL	MySQL	Обґрунтування вибору SQLite
Підтримка EF Core	✓ Повна	✓ Повна	✓ Повна	Microsoft.EntityFrameworkCore.Sqlite — офіційний пакет
Ліцензія	Public Domain	PostgreSQL	GPL / Commercial	Відкрита ліцензія, жодних обмежень для комерційного використання
Резервне копіювання	Копія файлу	pg_dump	mysqldump	Резервна копія БД = звичайне копіювання файлу agronom.db

Аналіз таблиці 1.5 підтверджує, що для настільного додатку, який не потребує мережевого доступу до БД, SQLite є оптимальним рішенням. Підтримка SQLite в Entity Framework Core 9 через офіційний пакет Microsoft.EntityFrameworkCore.Sqlite дозволяє використовувати зручний code-first підхід із LINQ-запитами і автоматичним створенням схеми бази даних.

Вибір MVVM-бібліотеки. Для реалізації патерну MVVM обрано бібліотеку CommunityToolkit.Mvvm версії 8.3. Ця бібліотека забезпечує генерацію коду через атрибути [ObservableProperty] та [RelayCommand] на етапі компіляції (source generators), що суттєво скорочує обсяг шаблонного коду і підвищує читабельність ViewModels. Альтернативою розглядалась бібліотека Prism.Wpf, яка є більш потужною у частині навігації між регіонами, проте значно складнішою у первинному налаштуванні.

Вибір бібліотеки графіків. Для реалізації аналітичного модуля обрано LiveChartsCore.SkiaSharpView.WPF версії 2.0. Бібліотека забезпечує GPU-рендеринг через SkiaSharp, підтримує анімації та інтерактивні підказки (tooltips). Розглядалась також бібліотека OxyPlot, яка є простішою, проте позбавлена анімацій та сучасних інтерактивних можливостей.

У таблиці 1.6 наведено зведений технологічний стек системи «АгрономКаталог» з обґрунтуванням кожного компоненту.

Таблиця 1.6 — Зведений технологічний стек системи «АгрономКаталог»

Компонент системи	Обрана технологія	Версія та обґрунтування вибору
Платформа	.NET 9.0	Остання LTS-версія платформи; підтримка сучасних можливостей C# 13
UI-фреймворк	WPF (Windows Presentation Foundation)	Зрілий фреймворк для настільних Windows-додатків з підтримкою MVVM та XAML
Архітектурний патерн	MVVM	Забезпечує розділення логіки та представлення; спрощує тестування
MVVM-бібліотека	CommunityToolkit.Mvvm 8.3	Source generators для ObservableObject, RelayCommand; мінімум boilerplate-коду
ORM	Entity Framework Core 9	Code-first підхід; міграції; LINQ-запити; AsNoTracking для читання
СУБД	SQLite 3	Вбудована БД без окремого сервера; файл agronom.db поряд із exe
Графіки	LiveChartsCore. SkiaSharpView.WPF 2.0	Інтерактивні анімовані графіки; GPU-рендеринг через SkiaSharp
DI-контейнер	Microsoft.Extensions. DependencyInjection 9	Стандартний контейнер .NET; реєстрація DbContext, Services, ViewModels

Обраний технологічний стек повністю задовольняє сформовані функціональні та нефункціональні вимоги до системи, є сучасним, добре документованим і забезпечує потенціал для подальшого розширення функціональності.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної галузі та існуючих програмних рішень для автоматизації агрономічної діяльності. На основі отриманих результатів можна зробити такі висновки:

1) Агропромисловий комплекс України є стратегічною галуззю економіки, що активно цифровізується. Рівень впровадження спеціалізованого ПЗ серед малих і середніх агропідприємств залишається недостатнім через дороговизну та складність існуючих рішень.
2) Аналіз п'яти існуючих програмних систем (Агроном.Про, АгроПлан, Trimble Ag Software, FarmLogs, 1С:Сільське господарство) показав, що жодна з них не поєднує в єдиному офлайновому інтерфейсі каталоги культур, добрив та шкідників з повноцінним аналітичним модулем і україномовним інтерфейсом.
3) Сформульовано 10 функціональних та 8 нефункціональних вимог до розроблюваної системи, що охоплюють CRUD-функціональність, пошук, аналітику, експорт даних, підтримку зображень та вимоги до продуктивності й надійності.
4) Обґрунтовано вибір технологічного стеку: платформа .NET 9, UI-фреймворк WPF, архітектурний патерн MVVM, бібліотека CommunityToolkit.Mvvm 8.3, ORM Entity Framework Core 9, СУБД SQLite, бібліотека графіків LiveChartsCore.SkiaSharpView.WPF 2.0.
5) Обраний стек технологій повністю відповідає вимогам щодо автономної роботи, зручності для кінцевого користувача, україномовного інтерфейсу та можливості подальшого розширення системи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Архітектурне рішення — патерн MVVM

Вибір архітектурного патерну є визначальним рішенням у проєктуванні будь-якої програмної системи. Для реалізації системи «АгрономКаталог» обрано архітектурний патерн MVVM (Model–View–ViewModel), який є стандартним підходом для розробки WPF-додатків і забезпечує чітке розділення відповідальностей між трьома ключовими компонентами.

Model (Модель) — представляє бізнес-логіку та дані предметної галузі. У системі «АгрономКаталог» до рівня моделі належать класи сутностей (Crop, Fertilizer, Pest, Region), контекст бази даних AgronomDbContext та сервіси (CropService, FertilizerService, PestService). Модель не має жодних залежностей від рівня представлення.

View (Представлення) — відповідає виключно за відображення інформації користувачеві. У WPF-додатку рівень View реалізовано через XAML-файли UserControl (CropsView, FertilizersView, PestsView, AnalyticsView, HelpView) та діалогові вікна. View не містить бізнес-логіки: вся взаємодія відбувається через механізм прив'язки даних (Data Binding).

ViewModel (Модель представлення) — є посередником між View та Model. ViewModels містять команди (RelayCommand), властивості для прив'язки (ObservableProperty) та логіку підготовки даних для відображення. Механізм INotifyPropertyChanged, що реалізується через базовий клас ObservableObject бібліотеки CommunityToolkit.Mvvm, забезпечує автоматичне оновлення інтерфейсу при зміні даних.

Застосування патерну MVVM забезпечує такі переваги для розроблюваної системи: незалежність рівнів (View може змінюватися без зміни ViewModel і навпаки); підтримку unit-тестування ViewModels без запуску UI; чистий код без змішування логіки та розмітки; спрощену підтримку та розширення системи у майбутньому.

На рисунку 2.1 зображено загальну архітектуру системи на основі патерну MVVM із зазначенням зв'язків між компонентами та напрямків потоків даних і команд.

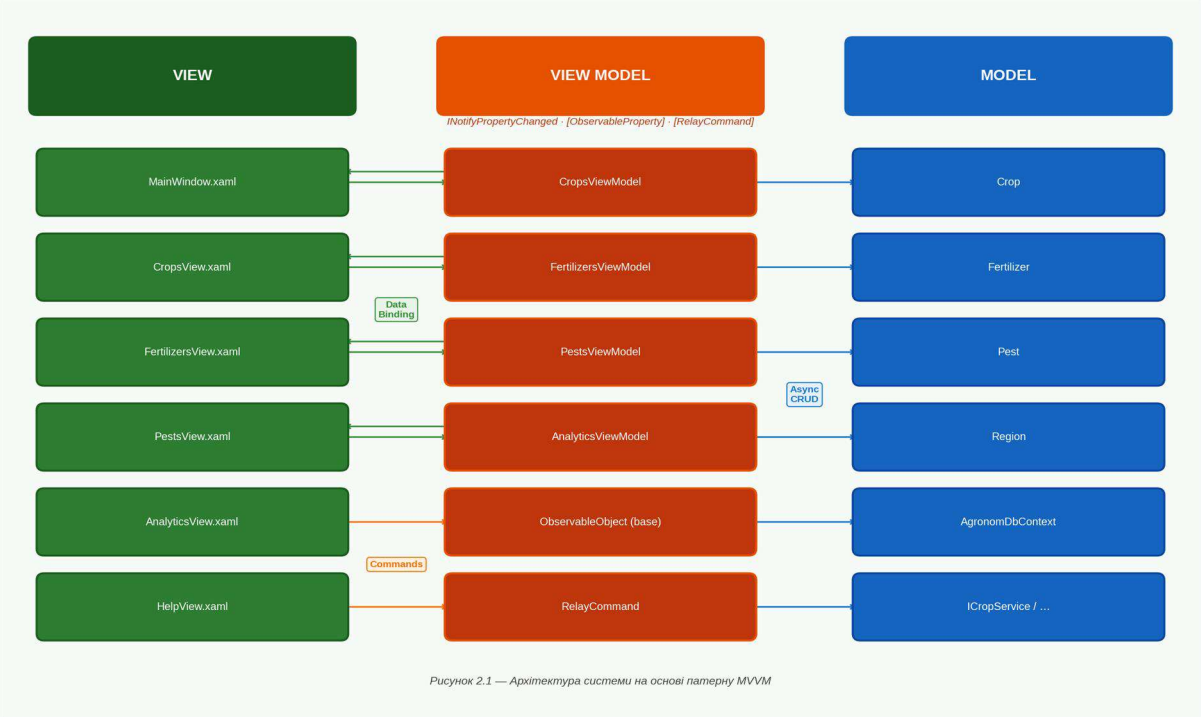

Рисунок 2.1 — Архітектура системи «АгрономКаталог» на основі патерну MVVM

Стрілки типу «Data Binding» позначають двосторонню прив'язку між View та ViewModel через механізм XAML Binding. Стрілки «Commands» відображають передачу команд користувача (натискання кнопок) до ViewModel через RelayCommand. Стрілки між ViewModel та Model позначають асинхронні CRUD-операції через сервісний шар.

2.2 Проєктування бази даних

Проєктування бази даних системи «АгрономКаталог» виконано у відповідності до принципів нормалізації реляційних баз даних. Для зберігання даних використовується вбудована СУБД SQLite, а взаємодія з базою здійснюється через Entity Framework Core 9 у режимі code-first.

База даних системи містить чотири основні таблиці та одну зв'язуючу: Crops (культури), Fertilizers (добрива), Pests (шкідники та хвороби), Regions (регіони) та CropRegion (зв'язок культур і регіонів). Файл бази даних

agronom.db зберігається в директорії поряд із виконуваним файлом програми, що забезпечує простоту резервного копіювання та перенесення системи.

На рисунку 2.2 представлено ER-діаграму (Entity-Relationship diagram) бази даних системи «АгрономКаталог» із зазначенням атрибутів кожної таблиці та зв'язків між ними.

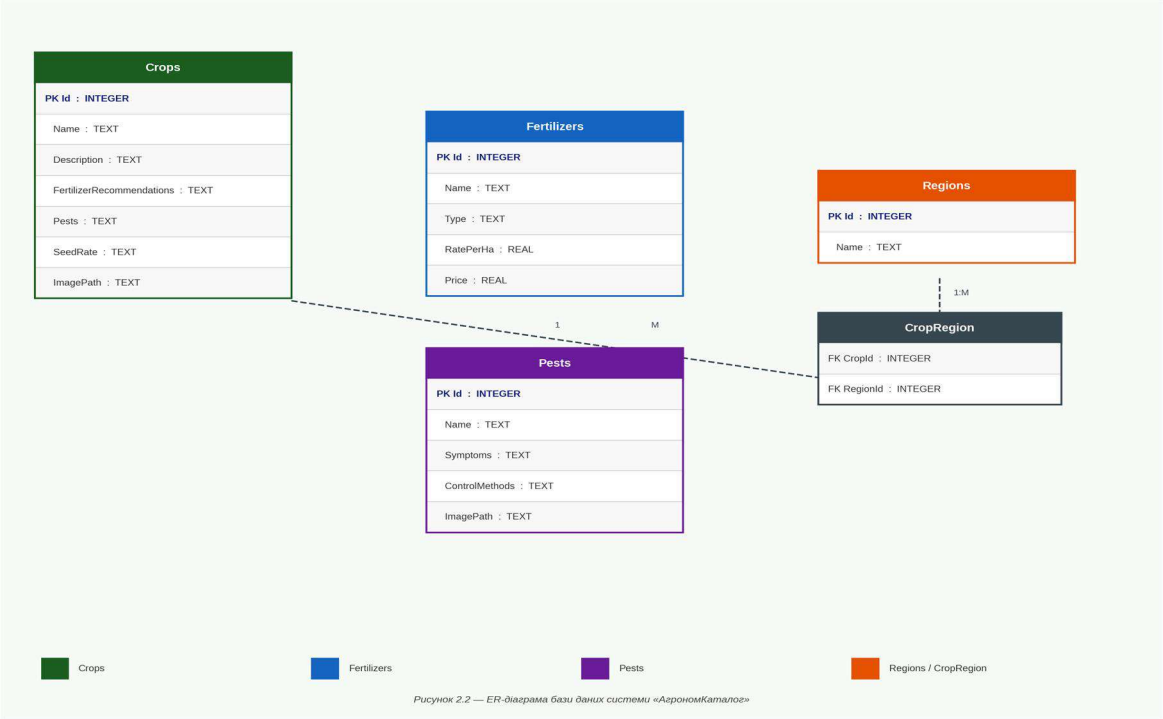

Рисунок 2.2 — ER-діаграма бази даних системи «АгрономКаталог»

Між таблицями Crops та Regions існує зв'язок типу «багато-до-багатьох» через зв'язуючу таблицю CropRegion. Таблиці Fertilizers та Pests є самостійними довідниками без жорстких зовнішньоключових зв'язків з іншими таблицями, що забезпечує гнучкість структури даних.

У таблицях 2.1–2.4 наведено детальний опис структури кожної таблиці бази даних.

Таблиця 2.1 — Структура таблиці Crops (Культури)

№	Поле	Тип	Пояснення
1	Id	INTEGER (PK)	Унікальний ідентифікатор (автоінкремент)
2	Name	TEXT NOT NULL	Назва культури (обов'язкове поле)
3	Description	TEXT	Агрономічний опис культури
4	FertilizerRecommendations	TEXT	Рекомендовані добрива та норми
5	Pests	TEXT	Перелік основних шкідників і хвороб
6	SeedRate	TEXT	Норма висіву (кг/га або шт./га)
7	ImagePath	TEXT NULL	Відносний шлях до зображення (Images\...)

Таблиця 2.2 — Структура таблиці Fertilizers (Добрива)

№	Поле	Тип	Пояснення
1	Id	INTEGER (PK)	Унікальний ідентифікатор
2	Name	TEXT NOT NULL	Назва добрива
3	Type	TEXT	Тип: N / P / K / NPK / Органічне / Мікро
4	RatePerHa	REAL	Норма внесення (кг/га або л/га)
5	Price	REAL	Орієнтовна ціна (грн/т або грн/л)

Таблиця 2.3 — Структура таблиці Pests (Шкідники та хвороби)

№	Поле	Тип	Пояснення
1	Id	INTEGER (PK)	Унікальний ідентифікатор
2	Name	TEXT NOT NULL	Назва шкідника або хвороби
3	Symptoms	TEXT	Опис симптомів ураження рослини
4	ControlMethods	TEXT	Засоби захисту: препарати, норми, строки
5	ImagePath	TEXT NULL	Відносний шлях до зображення

Таблиця 2.4 — Структура таблиці Regions (Регіони)

№	Поле	Тип	Пояснення
1	Id	INTEGER (PK)	Унікальний ідентифікатор
2	Name	TEXT NOT NULL	Назва агрокліматичного регіону України

Первинні ключі (PRIMARY KEY) у всіх таблицях реалізовано як цілочисельні автоінкрементні поля Id, що відповідає стандарту Entity Framework Core та анотації [Key]. Поля типу TEXT NULL допускають зберігання значення NULL — зокрема, поле ImagePath для зображень є необов'язковим. Клас DbInitializer забезпечує автоматичне наповнення таблиць демонстраційними даними при першому запуску: 10 культур, 20 добрив, 15 шкідників та 5 регіонів.

2.3 Діаграма класів системи

Діаграма класів є ключовим артефактом об'єктно-орієнтованого проєктування, що відображає структуру системи через класи, їх атрибути, методи та відносини між ними. На рисунку 2.3 представлено спрощену діаграму класів системи «АгрономКаталог», що охоплює три основні рівні: моделі даних, сервісний шар та ViewModels.

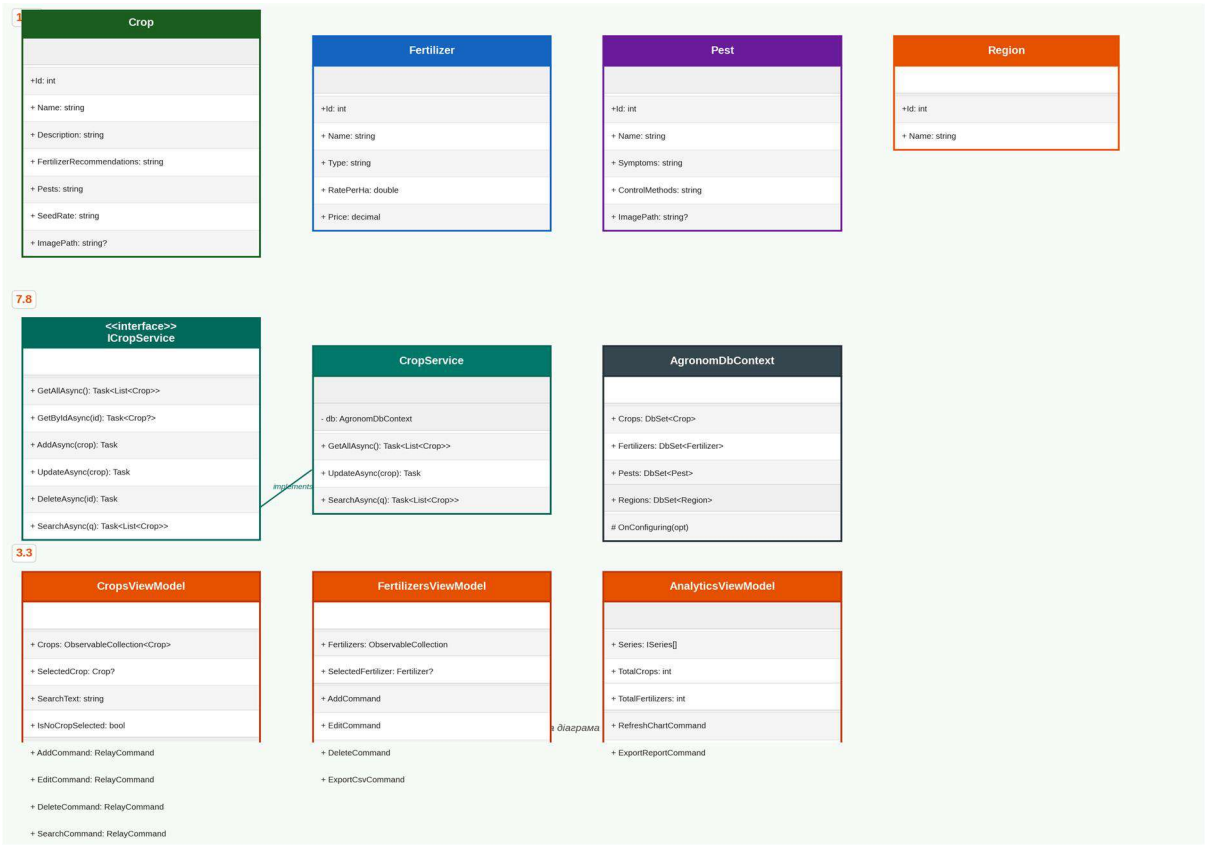

Рисунок 2.3 — Спрощена діаграма класів системи «АгрономКаталог»

Рівень моделей включає чотири класи-сутності (Crop, Fertilizer, Pest, Region), кожен з яких відповідає окремій таблиці бази даних. Усі класи декоровано атрибутами анотацій даних System.Component Model.DataAnnotations: [Key] для первинного ключа та [Required] для обов'язкових полів.

Сервісний шар побудовано на основі принципу інверсії залежностей: для кожної сутності визначено інтерфейс (ICropService, IFertilizerService, IPestService) з набором async-методів CRUD та пошуку. Конкретні реалізації (CropService тощо) приймають AgronomDbContext через constructor injection, що забезпечує слабке зчеплення з рівнем доступу до даних.

ViewModels успадковують клас ObservableObject з бібліотеки CommunityToolkit.Mvvm. Властивості, позначені атрибутом [ObservableProperty], автоматично генерують реалізацію INotifyPropertyChanged на етапі компіляції. Методи з атрибутом [RelayCommand] генерують відповідні Command-об'єкти для прив'язки до XAML.

2.4 Діаграма варіантів використання

Діаграма варіантів використання (Use Case diagram) описує функціональні можливості системи з точки зору кінцевого користувача. Єдиним актором системи «АгрономКаталог» є Агроном — фахівець, що використовує систему для доступу до агрономічної довідкової інформації.

На рисунку 2.4 представлено діаграму варіантів використання системи «АгрономКаталог» із відображенням 12 основних прецедентів у межах системної границі.

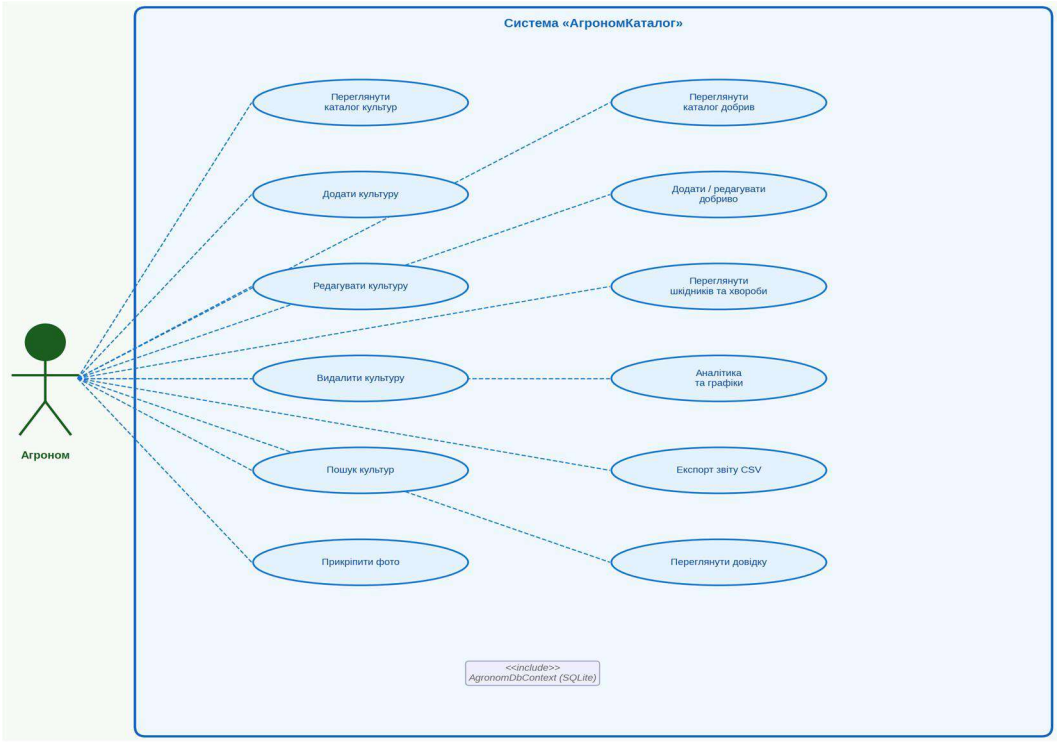

Рисунок 2.4 — Діаграма варіантів використання системи «АгрономКаталог»

Аналіз діаграми дозволяє виокремити три логічні групи варіантів використання. Перша група стосується роботи з каталогом культур: перегляд, додавання, редагування, видалення, пошук та прикріплення фотографій. Друга група охоплює аналогічні операції з каталогом добрив та каталогом шкідників і хвороб. Третя група — аналітичні функції: перегляд графіків, порівняння норм добрив та експорт звітів у форматі CSV. Усі варіанти використання включають (<<include>>) взаємодію з AgronomDbContext для читання або запису даних у SQLite.

2.5 Діаграми послідовності

Діаграми послідовності (Sequence diagrams) описують динаміку взаємодії компонентів системи у часі для виконання конкретних сценаріїв. Розроблено дві ключові діаграми послідовності: для операції додавання культури та для операції пошуку.

Сценарій 1: Додавання нової культури. На рисунку 2.5 зображено послідовність взаємодії компонентів при додаванні нової культури

користувачем. Сценарій охоплює 13 кроків від натискання кнопки до відображення нової картки у списку.

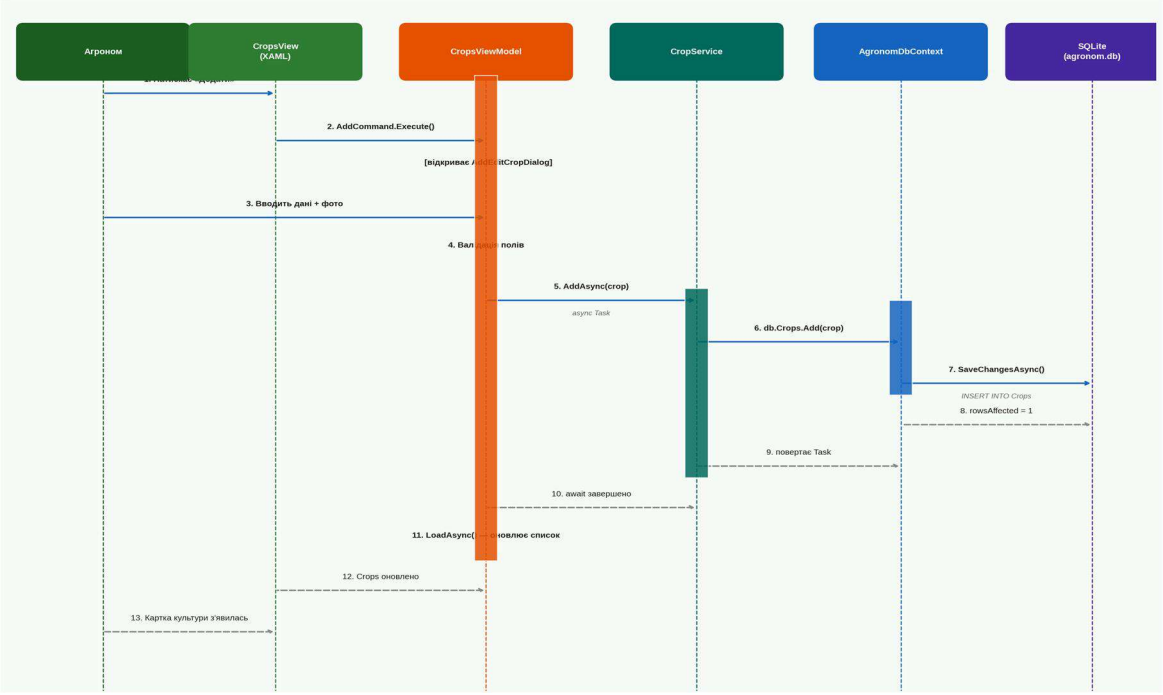

Рисунок 2.5 — Діаграма послідовності: операція додавання нової культури

Ключові аспекти реалізованого сценарію: відкриття модального діалогу AddEditCropDialog з порожніми полями; валідація обов'язкового поля «Назва» перед збереженням; асинхронне збереження через CropService.AddAsync(); автоматичне перезавантаження списку після успішного збереження за допомогою LoadAsync(); відновлення стану SelectedCrop після оновлення ObservableCollection.

Сценарій 2: Пошук культур за ключовим словом. На рисунку 2.6 зображено послідовність операції пошуку. Пошук відбувається за полями Name та Description культури через LINQ-вираз з методом Contains().

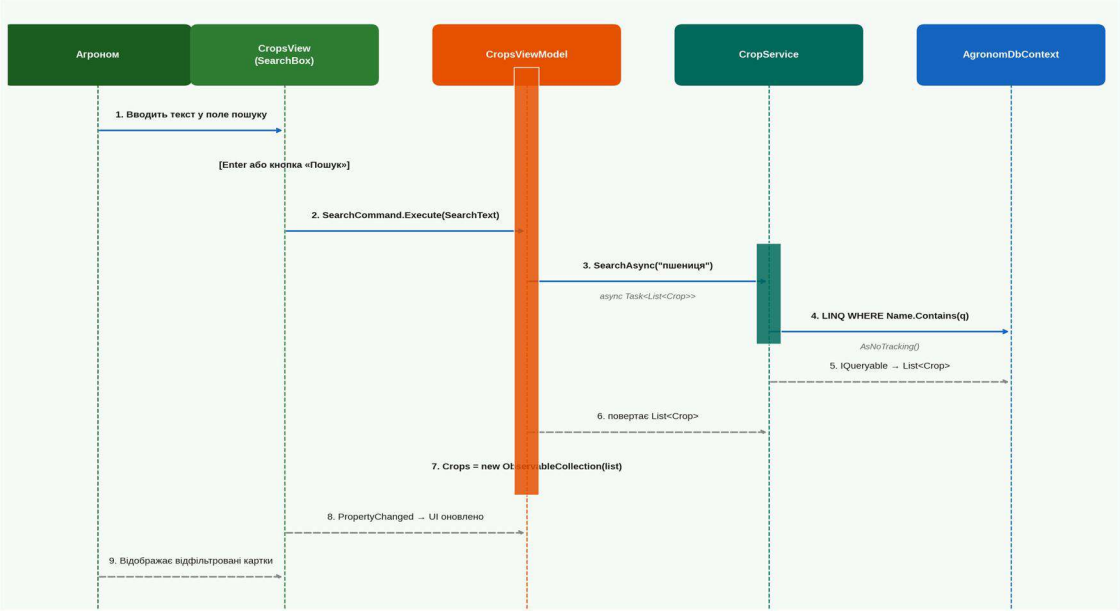

Рисунок 2.6 — Діаграма послідовності: операція пошуку та фільтрації культур

Особливістю реалізованого пошуку є застосування методу AsNoTracking() при формуванні LINQ-запиту до Entity Framework Core. Це гарантує відсутність конфліктів трекінгу сутностей при наступних операціях редагування — вирішена проблема, що виникла при первинній реалізації через використання db.Crops.Update() для відокремлених (detached) сутностей.

2.6 Компонентна діаграма

Компонентна діаграма відображає фізичну структуру системи: склад компонентів, їх залежності та зв'язки із зовнішніми ресурсами. На рисунку 2.7 наведено компонентну діаграму системи «АгрономКаталог».

Система «АгрономКаталог» упакована в єдиний виконуваний файл AgronomCatalog.exe і складається з шести внутрішніх компонентів. Компонент Views/XAML містить усі UserControl та діалогові вікна. Компонент ViewModels реалізує бізнес-логіку представлення. Компонент Services забезпечує CRUD-операції через інтерфейси. Компонент Data містить AgronomDbContext та DbInitializer. Компонент Models визначає класи-сутності. Компонент Resources містить стилі XAML та конвертори значень.

Рисунок 2.7 — Компонентна діаграма системи «АгрономКаталог»

Зовнішніми компонентами системи є: файл бази даних SQLite agronom.db, папка Images/ для зображень культур і шкідників, а також середовище виконання .NET 9 Runtime. Всі зовнішні ресурси розташовуються в директорії поряд із виконуваним файлом програми, що забезпечує портативність системи без необхідності встановлення.

2.7 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача (UI) системи «АгрономКаталог» виконувалося з урахуванням принципів UX-дизайну: простота освоєння, візуальна ієрархія інформації, мінімальна кількість кроків для виконання основних операцій та узгодженість кольорового оформлення.

Головне вікно системи побудовано на основі TabControl із п'ятьма вкладками, що забезпечує інтуїтивну навігацію між розділами. Для кожної вкладки вибрано унікальну колірну схему: зелена для культур, синя для добрив, фіолетова для шкідників, помаранчева для аналітики. Це дозволяє користувачу миттєво орієнтуватися в розділах.

Ключовою особливістю інтерфейсу є картковий вигляд каталогів (card-based layout) замість традиційних рядків таблиці DataGrid. Кожна картка

відображає фотографію запису, назву та ключові параметри. При виборі картки праворуч розгортається панель деталей з повною інформацією. Такий підхід підвищує наочність і робить систему більш привабливою для кінцевого користувача.

На рисунку 2.8 представлено детальний макет (wireframe) головного вікна системи «АгрономКаталог» для вкладки «Культури» з позначенням основних UI-елементів.

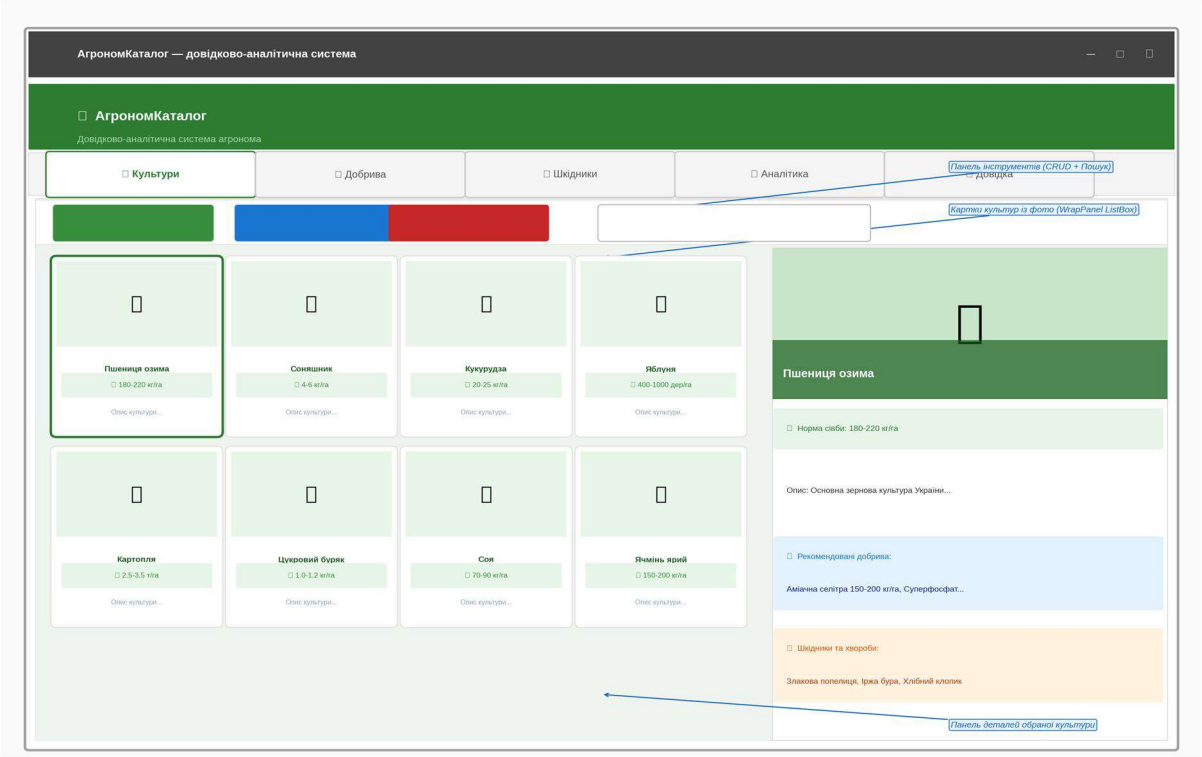

Рисунок 2.8 — Макет (wireframe) головного вікна системи «АгрономКаталог»

Макет ілюструє такі структурні елементи інтерфейсу: рядок заголовка вікна з назвою системи; зелена шапка з логотипом та підзаголовком; панель вкладок із підсвіченою активною вкладкою; панель інструментів із кнопками CRUD та полем пошуку; сітка карток культур у WrapPanel (ListBox); права панель деталей обраної культури з фото-шапкою, градієнтом, нормою сівби та інформаційними блоками добрив і шкідників.

Діалогові вікна редагування (AddEditCropDialog, AddEditFertilizerDialog, AddEditPestDialog) реалізовано як модальні вікна із кольоровою шапкою, відповідною до теми розділу. Вікно редагування

культури додатково містить блок вибору зображення з превью та кнопками «Обрати файл» і «Видалити фото».

Висновки до розділу 2

У другому розділі виконано комплексне проєктування системи «АгрономКаталог». На основі проведеної роботи можна зробити такі висновки:

1) Обрано архітектурний патерн MVVM як основу побудови системи. Застосування CommunityToolkit.Mvvm дозволяє реалізувати патерн з мінімальною кількістю шаблонного коду завдяки source generators для ObservableProperty та RelayCommand.
2) Спроєктовано реляційну базу даних із п'ятьма таблицями (Crops, Fertilizers, Pests, Regions, CropRegion). Структуру таблиць детально описано в таблицях 2.1–2.4. Автоматична ініціалізація БД при першому запуску забезпечує зручність розгортання системи.
3) Побудовано комплекс UML-діаграм: діаграма класів (рисунок 2.3) відображає статичну структуру системи; діаграма варіантів використання (рисунок 2.4) — функціональні вимоги; діаграми послідовності (рисунки 2.5–2.6) — динаміку взаємодії компонентів; компонентна діаграма (рисунок 2.7) — фізичну організацію системи.
4) Спроєктовано сучасний інтерфейс користувача на основі карткового відображення даних із підтримкою фотографій, колірного кодування розділів та детальної панелі перегляду. Wireframe головного вікна (рисунок 2.8) підтвердив відповідність архітектурних рішень нефункціональним вимогам щодо зручності та освоюваності системи.
5) Усі проєктні рішення, прийняті в даному розділі, безпосередньо реалізовано у програмному коді, описаному в розділі 3.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Реалізація моделей даних та Entity Framework Core

Реалізацію системи «АгрономКаталог» розпочато з визначення моделей даних — класів C#, що описують структуру доменних сутностей та відображаються на таблиці бази даних SQLite через Entity Framework Core 9. Усі моделі розміщено у папці Models проєкту та декоровано атрибутами анотацій даних з простору імен System.ComponentModel.DataAnnotations.

У лістингу 3.1 наведено клас Crop — основна сутність системи, що описує сільськогосподарську культуру.

```
using System.ComponentModel.DataAnnotations;
namespace AgronomCatalog.Models;
public class Crop
{
    [Key]
    public int Id { get; set; }
    [Required]
    [MaxLength(200)]
    public string Name { get; set; } = string.Empty;
    public string Description              { get; set; } = string.Empty;
    public string FertilizerRecommendations { get; set; } = string.Empty;
    public string Pests                    { get; set; } = string.Empty;
    public string SeedRate                 { get; set; } = string.Empty;
    // Відносний шлях до зображення (Images\filename.jpg)
    public string? ImagePath { get; set; }
}
```

Лістинг 3.1 — Models/Crop.cs — модель сільськогосподарської культури

Атрибут [Key] позначає первинний ключ, що Entity Framework Core автоматично налаштовує як AUTOINCREMENT у SQLite. Атрибут [Required] забезпечує валідацію на рівні моделі та генерує обмеження NOT NULL у схемі бази даних. Поле ImagePath оголошено nullable (string?), оскільки фотографія не є обов'язковим елементом запису.

Контекст бази даних AgronomDbContext є центральним класом рівня доступу до даних і успадковує DbContext із бібліотеки Entity Framework Core. У лістингу 3.2 представлено його реалізацію.

```
using AgronomCatalog.Models;
using Microsoft.EntityFrameworkCore;
using System.IO;

namespace AgronomCatalog.Data;

public class AgronomDbContext : DbContext
{
    public DbSet<Crop>       Crops       { get; set; }
    public DbSet<Fertilizer> Fertilizers { get; set; }
    public DbSet<Pest>       Pests       { get; set; }
    public DbSet<Region>     Regions     { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder opt)
    {
        var dbPath = Path.Combine(
            AppDomain.CurrentDomain.BaseDirectory, "agronom.db");
        opt.UseSqlite($"Data Source={dbPath}");
    }
}
```

Лістинг 3.2 — Data/AgronomDbContext.cs — контекст бази даних

Метод OnConfiguring визначає рядок підключення до SQLite. Використання AppDomain.CurrentDomain.BaseDirectory гарантує, що файл agronom.db завжди створюється поряд із виконуваним файлом програми, незалежно від робочого каталогу процесу.

Ініціалізація бази даних з демонстраційними даними реалізована в класі DbInitializer. Методи SeedCrops(), SeedFertilizers() і SeedPests() перевіряють наявність записів перед додаванням (if (db.Crops.Any()) return;), що запобігає дублюванню даних при повторних запусках.

Схему ініціалізації DI-контейнера та послідовність запуску системи відображено на рисунку 3.10.

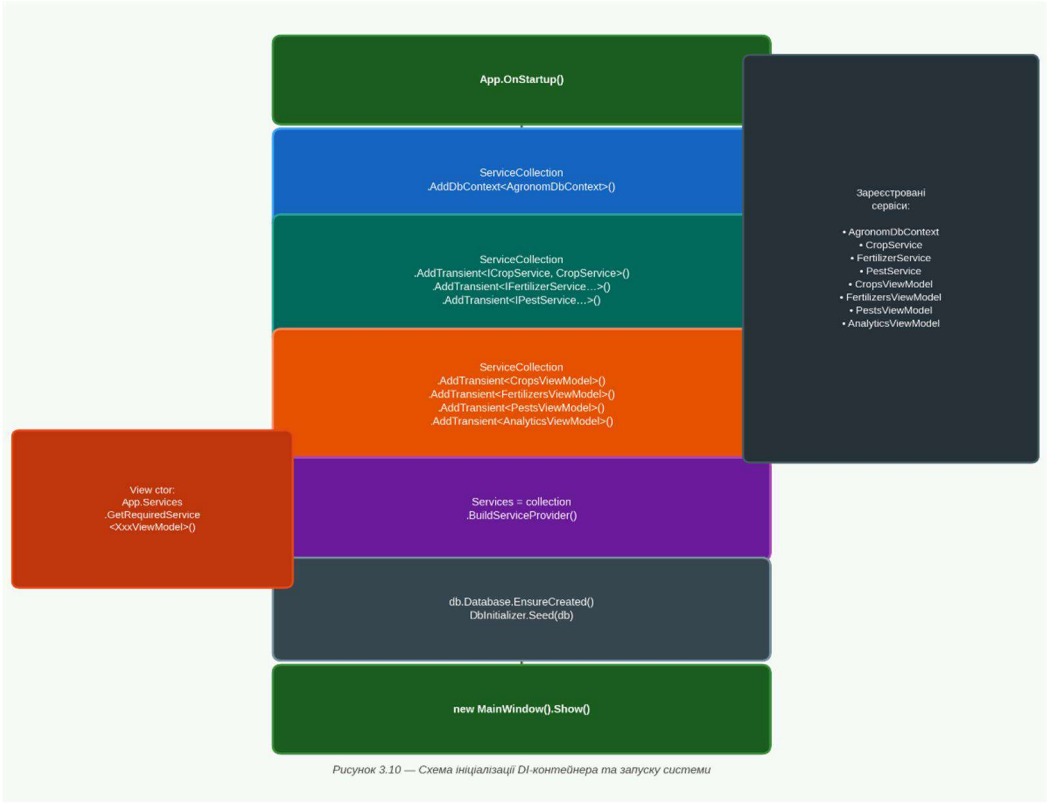

Рисунок 3.10 — Схема ініціалізації DI-контейнера та запуску системи

Рисунок 3.10 — Схема ініціалізації DI-контейнера та послідовність запуску системи

Метод App.OnStartup() реєструє всі залежності у ServiceCollection, будує ServiceProvider та передає його у статичну властивість App.Services. Це дозволяє UserControls отримувати ViewModels через App.Services.GetRequiredService<T>() у своїх конструкторах, забезпечуючи повноцінну інверсію залежностей без використання зовнішнього IoC-контейнера.

3.2 Реалізація сервісного шару

Сервісний шар системи побудовано на принципі інверсії залежностей (Dependency Inversion Principle). Для кожної сутності визначено інтерфейс (ICropService, IFertilizerService, IPestService) та відповідну реалізацію. У лістингу 3.3 наведено реалізацію CropService із ключовими методами.

```
public class CropService(AgronomDbContext db) : ICropService
{
    // Читання завжди з AsNoTracking() — виключає конфлікти трекінгу
    public async Task<List<Crop>> GetAllAsync() =>
        await db.Crops.AsNoTracking()
                      .OrderBy(c => c.Name)
                      .ToListAsync();

    // Оновлення через Entry.CurrentValues.SetValues
    // замість db.Crops.Update() — вирішує InvalidOperationException
    public async Task UpdateAsync(Crop crop)
    {
        var tracked = await db.Crops.FindAsync(crop.Id);
        if (tracked == null) return;
        db.Entry(tracked).CurrentValues.SetValues(crop);
        await db.SaveChangesAsync();
    }

    public async Task<List<Crop>> SearchAsync(string query) =>
        string.IsNullOrWhiteSpace(query)
            ? await GetAllAsync()
            : await db.Crops.AsNoTracking()
                .Where(c => c.Name.Contains(query)
                         || c.Description.Contains(query))
                .OrderBy(c => c.Name)
                .ToListAsync();
}
```

Лістинг 3.3 — Services/CropService.cs — сервіс для роботи з культурами

Ключовою особливістю реалізації є патерн «завантажити відстежувану сутність — скопіювати значення» в методі UpdateAsync. Такий підхід вирішує System.InvalidOperationException про конфлікт трекінгу, що виникає при спробі прикріпити до контексту новий екземпляр сутності з тим самим Id, що вже відстежується. Метод FindAsync звертається спочатку до кешу першого рівня (Identity Map), і лише за відсутності — до бази даних.

3.3 Реалізація ViewModel на базі CommunityToolkit.Mvvm

ViewModels системи реалізовано з використанням бібліотеки CommunityToolkit.Mvvm версії 8.3. Бібліотека надає базовий клас ObservableObject та атрибути [ObservableProperty] і [RelayCommand], що через механізм source generators автоматично генерують реалізацію INotifyPropertyChanged та ICommand на етапі компіляції.

У лістингу 3.4 наведено ключові фрагменти CropsViewModel.

```
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;

[partial class] CropsViewModel : ObservableObject
{
    // [ObservableProperty] генерує: Crops, SelectedCrop,
    // SearchText, IsNoCropSelected з INotifyPropertyChanged
    [ObservableProperty]
    private ObservableCollection<Crop> _crops = [];

    [ObservableProperty] private Crop?  _selectedCrop;
    [ObservableProperty] private string _searchText = string.Empty;
    [ObservableProperty] private bool   _isNoCropSelected = true;

    // Автоматично викликається при зміні SelectedCrop
    partial void OnSelectedCropChanged(Crop? value)
        => IsNoCropSelected = value == null;

    // [RelayCommand] генерує EditCommand: IRelayCommand
    [RelayCommand]
    private async Task Edit()
    {
        if (SelectedCrop == null) { /* попередження */ return; }
        var dlg = new AddEditCropDialog(SelectedCrop);
        if (dlg.ShowDialog() == true && dlg.Result != null)
        {
            await _svc.UpdateAsync(dlg.Result);
            await LoadAsync();
            // Відновлюємо вибір після перезавантаження
            SelectedCrop = Crops.FirstOrDefault(
                c => c.Id == dlg.Result.Id);
        }
    }
}
```

Лістинг 3.4 — ViewModels/CropsViewModel.cs — ViewModel для каталогу культур

Властивість IsNoCropSelected разом із конвертором BooleanToVisibilityConverter керує видимістю placeholder-тексту «Оберіть культуру» в правій панелі деталей. Partial-метод OnSelectedCropChanged автоматично генерується source generator при використанні [ObservableProperty] і викликається після кожної зміни значення SelectedCrop.

3.4 Реалізація інтерфейсу користувача

Інтерфейс системи «АгрономКаталог» побудовано на основі WPF (Windows Presentation Foundation) з використанням XAML-розмітки. Головне вікно MainWindow.xaml містить TabControl із п'ятьма вкладками, кожна з яких завантажує відповідний UserControl.

На рисунку 3.1 зображено вкладку «Культури» із картковим відображенням каталогу у вигляді ListBox із шаблоном DataTemplate. Обрана картка виділяється зеленою рамкою та підсиленою тінню завдяки тригерам у ItemContainerStyle.

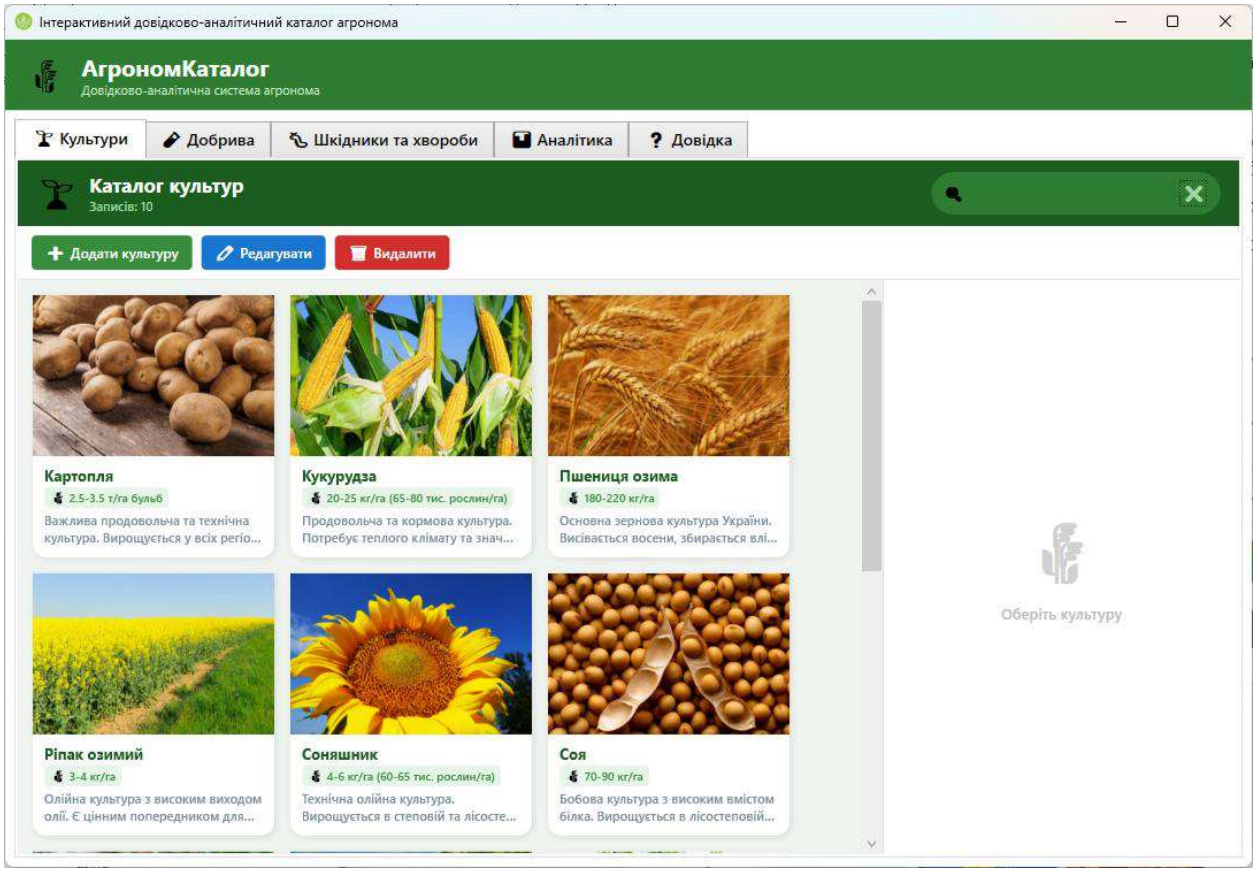

Рисунок 3.1 — Вкладка «Культури» — картковий вигляд каталогу (скріншот програми)

На рисунку 3.2 показано праву панель деталей обраної культури. Панель містить фото-шапку з лінійним градієнтом поверх зображення, назву культури білим текстом, блок норми сівби та два інформаційні блоки (добрива і шкідники) з різними кольоровими акцентами.

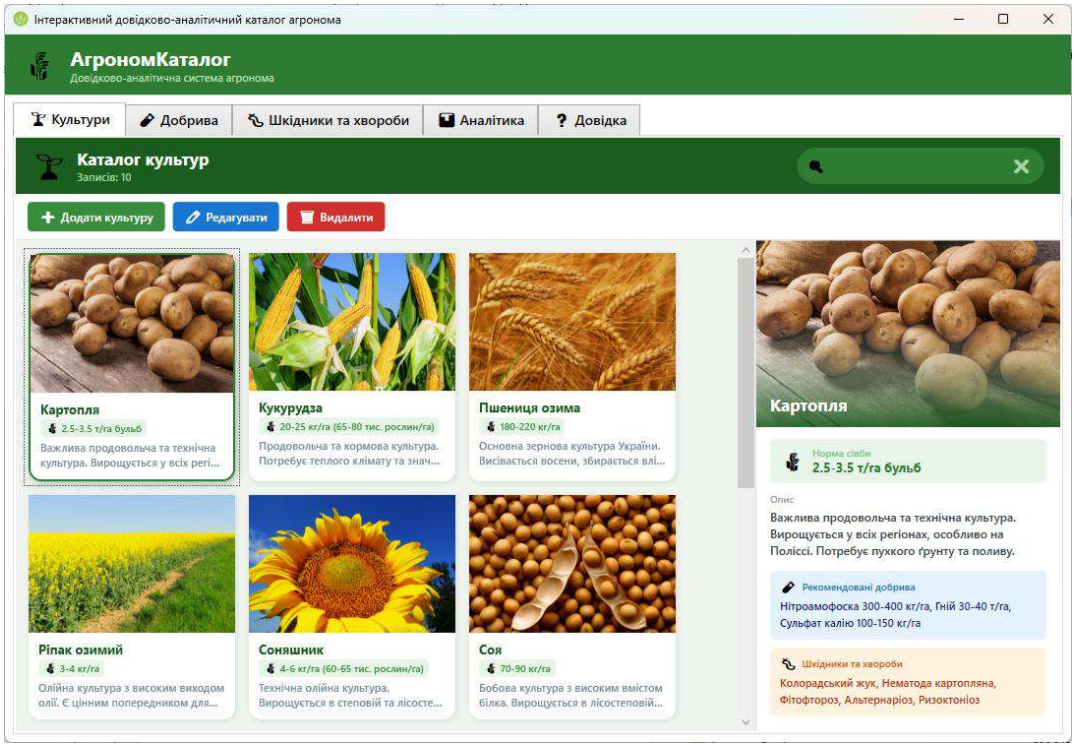

Рисунок 3.2 — Панель деталей обраної культури (скріншот програми)

Діалогове вікно AddEditCropDialog, зображене на рисунку 3.3, реалізовано як модальне вікно з темно-зеленою шапкою, превью зображення та кнопками вибору і видалення фото. Зображення копіюється до підпапки Images\ поряд із exe із захистом від дублювання імен файлів.

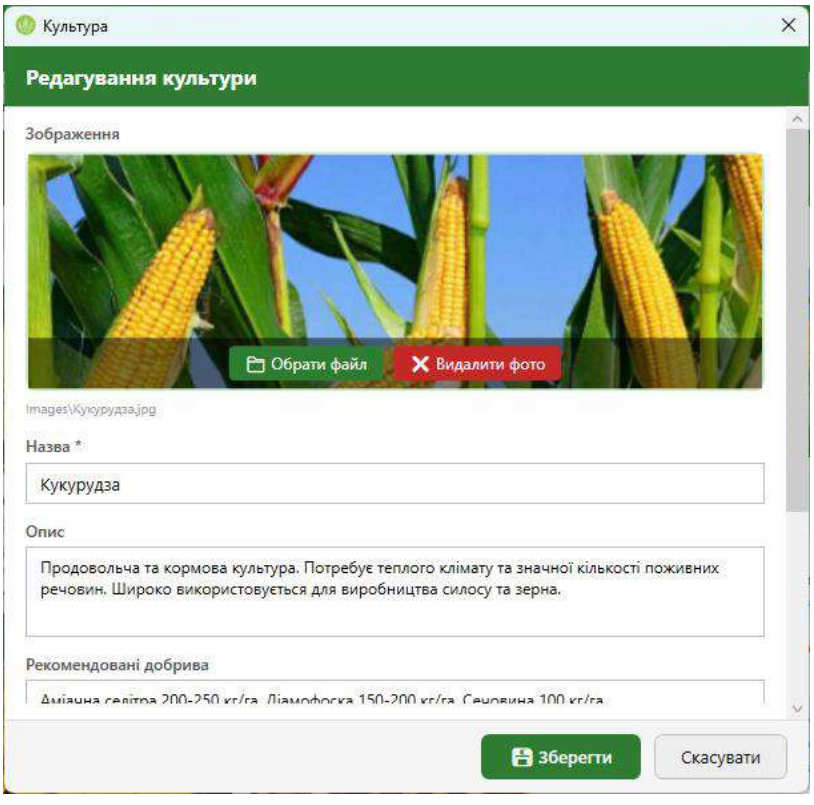

Рисунок 3.3 — Діалогове вікно редагування культури (скріншот програми)

Вкладка «Добрива» (рисунок 3.4) реалізована у вигляді DataGrid із сортуванням та фільтрацією. Панель інструментів містить кнопки CRUD та кнопку «Експорт CSV», яка викликає метод ExportCsvCommand і зберігає звіт у форматі UTF-8 із роздільником «;» для сумісності з Microsoft Excel.

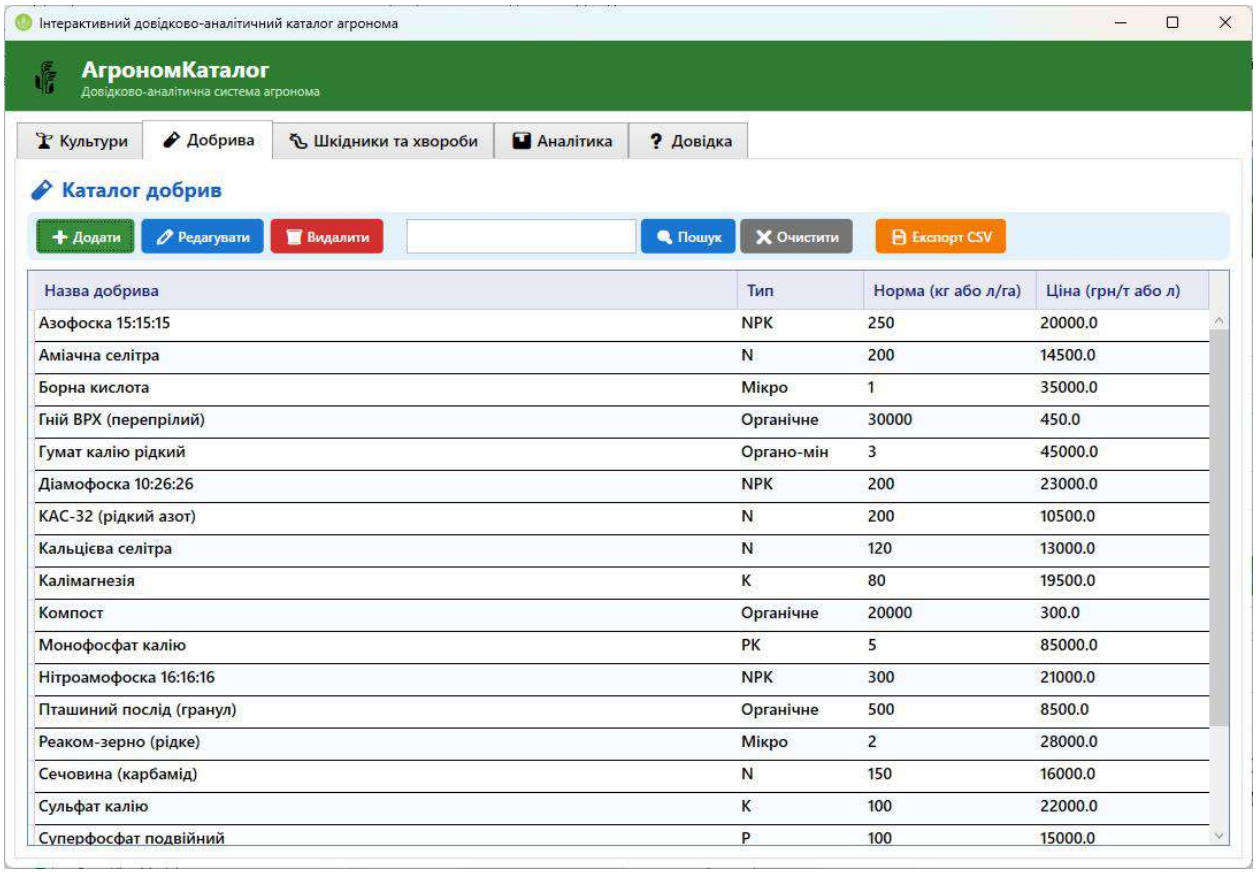

Рисунок 3.4 — Вкладка «Добрива» (скріншот програми)

Вкладка «Шкідники та хвороби» (рисунок 3.5) виконана в єдиному стилі з каталогом культур — той самий картковий ListBox з фіолетовою кольоровою схемою, права панель деталей з блоками «Симптоми ураження» та «Засоби захисту».

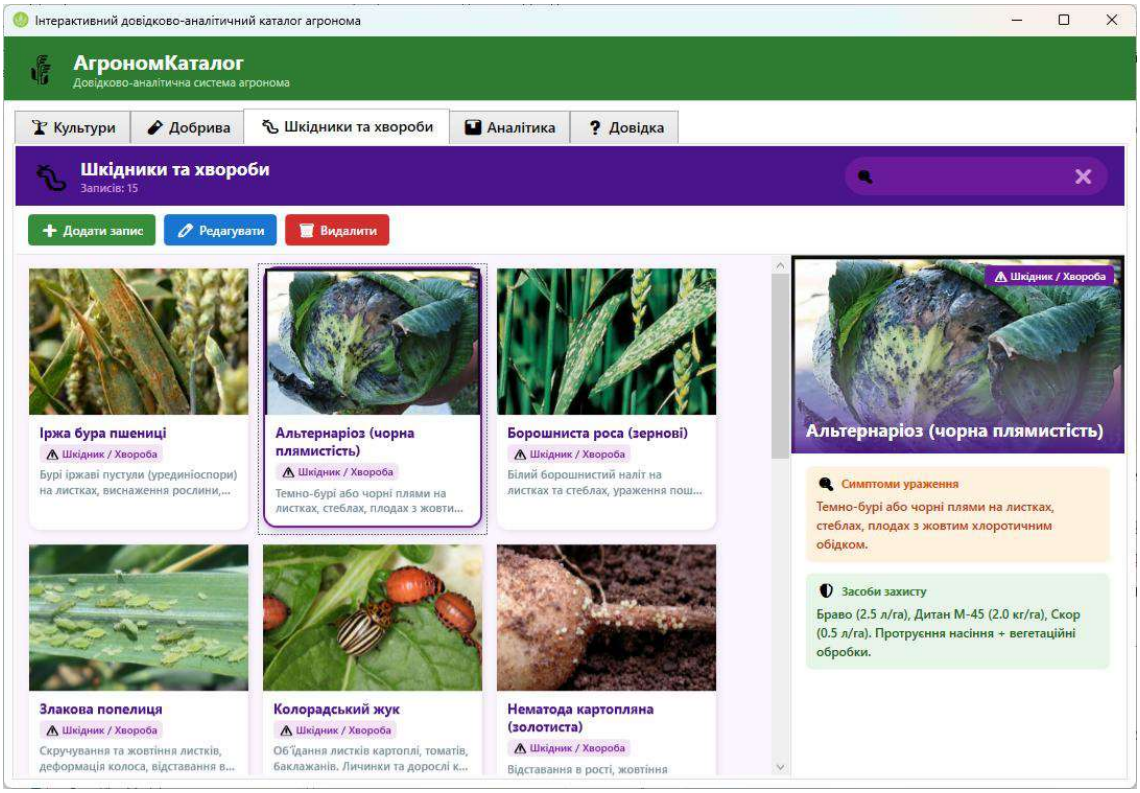

Рисунок 3.5 — Вкладка «Шкідники та хвороби» (скріншот програми)

Аналітичний модуль (рисунок 3.6) відображає три статистичні картки із загальною кількістю записів у кожному каталозі, ComboBox для вибору типу графіку та CartesianChart від LibChartsCore з анімованими стовпцями.

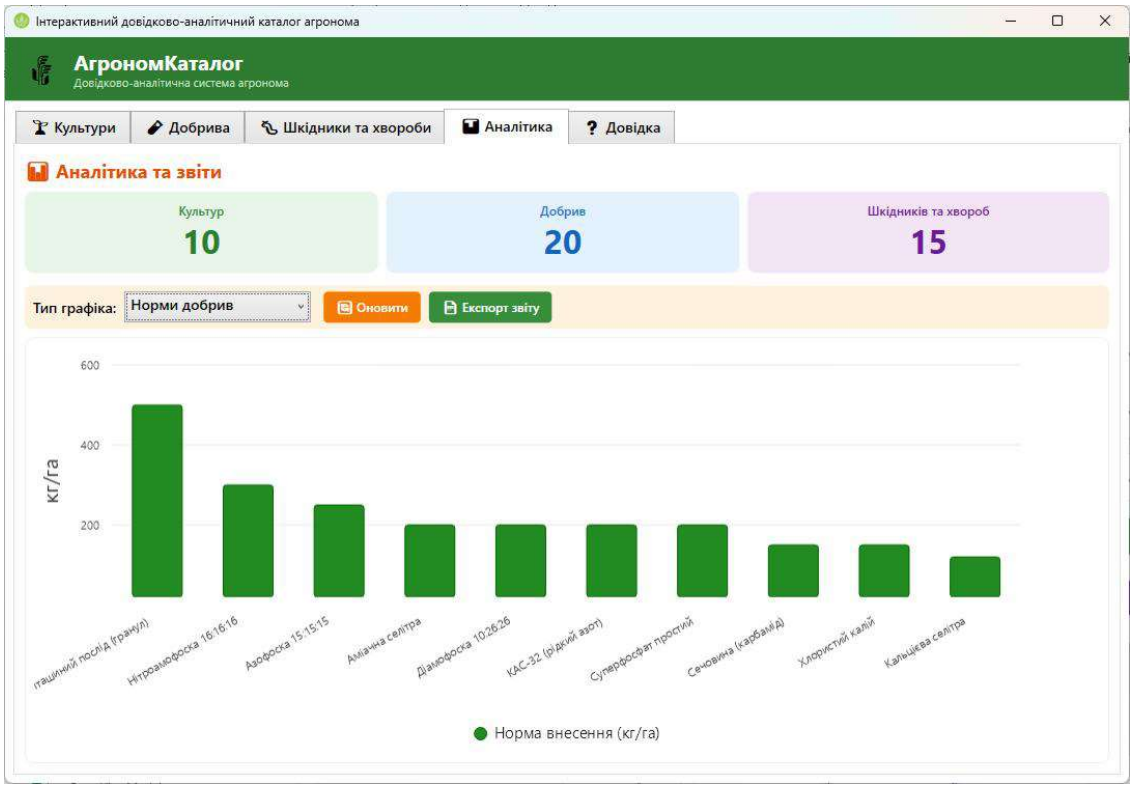

Рисунок 3.6 — Вкладка «Аналітика» (скріншот програми)

Розділ «Довідка» (рисунок 3.7) реалізовано як ScrollViewer з форматованим описом всіх розділів системи, інструкцій з використання CRUD-операцій та технічних відомостей про стек технологій. Весь текст викладено українською мовою.

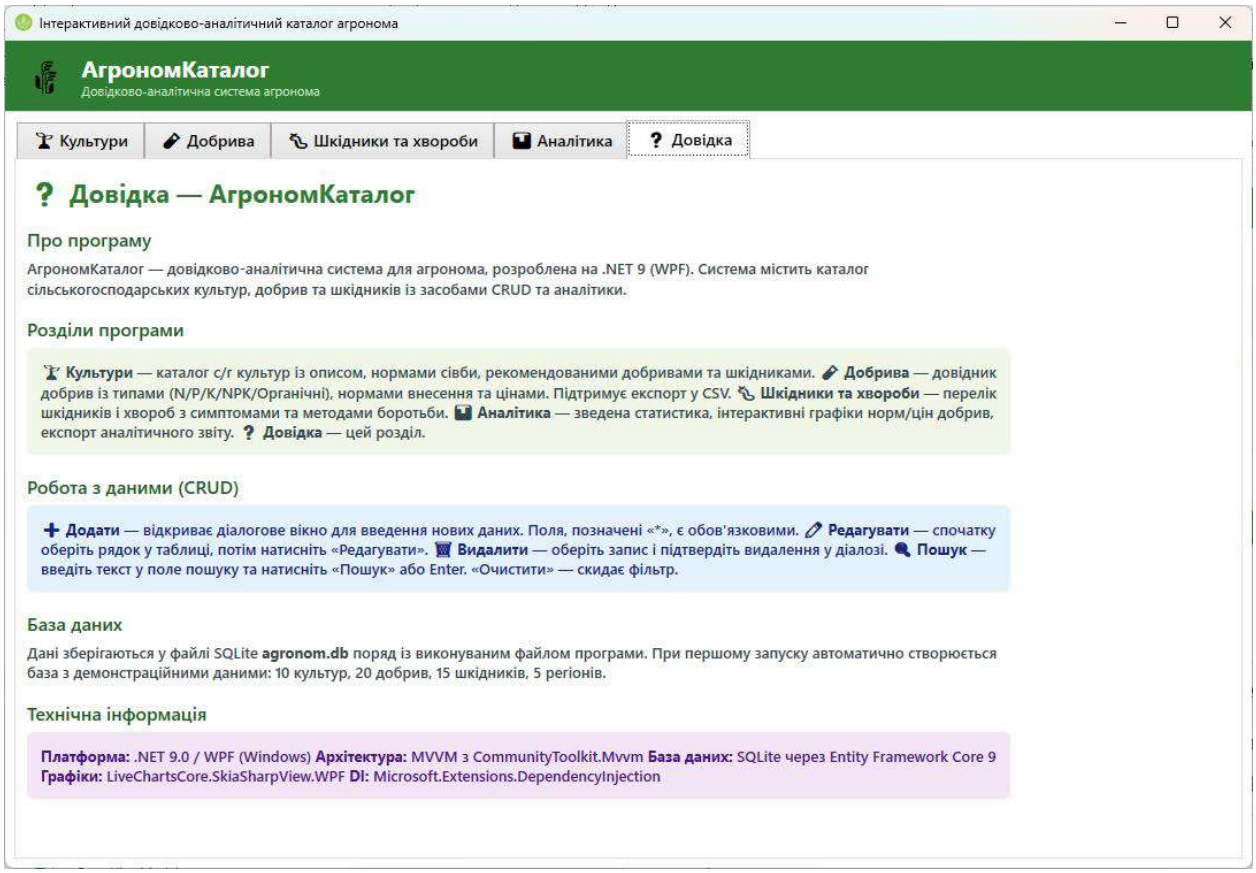

Рисунок 3.7 — Вкладка «Довідка» (скріншот програми)

3.5 Реалізація аналітичного модуля

Аналітичний модуль системи реалізовано в класі AnalyticsViewModel із використанням бібліотеки LiveChartsCore.SkiaSharpView.WPF версії 2.0. Бібліотека забезпечує GPU-рендеринг через SkiaSharp, підтримує анімації та інтерактивні tooltip-підказки.

У лістингу 3.5 наведено метод побудови стовпчастого графіку норм добрив.

```
private void BuildFertilizerRateChart(List<Fertilizer> fertilizers)
{
    // Виключаємо органічні добрива (норма > 1000 кг/га)
    // та беремо топ-10 для зручності відображення
    var top = fertilizers
        .Where(f => f.RatePerHa <= 1000)
        .OrderByDescending(f => f.RatePerHa)
        .Take(10)
        .ToList();

    Series = [
        new ColumnSeries<double>
        {
            Name   = "Норма внесення (кг/га)",
            Values = top.Select(f => f.RatePerHa).ToArray(),
            Fill   = new SolidColorPaint(SKColors.ForestGreen),
            Stroke = new SolidColorPaint(SKColors.DarkGreen)
                     { StrokeThickness = 1 }
        }
    ];
    XAxes = [ new Axis {
        Labels = top.Select(f => f.Name).ToArray(),
        LabelsRotation = -30,
        TextSize = 11
    }];
    YAxes = [ new Axis { Name = "кг/га", TextSize = 11 }];
}
```

Лістинг 3.5 — ViewModels/AnalyticsViewModel.cs — побудова графіку норм добрив

Властивості Series, XAxes та YAxes оголошено як [ObservableProperty], що дозволяє LiveCharts автоматично перебудовувати графік при їх зміні через механізм WPF Data Binding. Команда RefreshChartCommand перемикає між двома режимами відображення: нормами внесення та цінами добрив.

3.6 Реалізація експорту звітів у CSV

Функція експорту даних реалізована в двох ViewModels: FertilizersViewModel (окремий каталог добрив) та AnalyticsViewModel (зведений аналітичний звіт). Для коректного відображення кирилиці у Microsoft Excel використовується кодування UTF-8 та роздільник «;» замість стандартної коми.

У лістингу 3.6 наведено метод експорту аналітичного звіту.

```
[RelayCommand]
private async Task ExportReport()
{
    var dlg = new SaveFileDialog {
        Title    = "Зберегти аналітичний звіт",
        Filter   = "CSV файли (*.csv)|*.csv",
        FileName = "analityka_zvit.csv"
    };
    if (dlg.ShowDialog() != true) return;

    var ferts = await _fertSvc.GetAllAsync();
    var crops = await _cropSvc.GetAllAsync();

    var sb = new StringBuilder();
    sb.AppendLine("АгрономКаталог — Аналітичний звіт");
    sb.AppendLine($"Дата: {DateTime.Now:dd.MM.yyyy HH:mm}");
    sb.AppendLine($"Статистика: Культур={TotalCrops},
                   Добрив={TotalFertilizers}, Шкідників={TotalPests}");
    sb.AppendLine();
    sb.AppendLine("=== ДОБРИВА ===");
    sb.AppendLine("Назва;Тип;Норма (кг/га);Ціна (грн)");
    foreach (var f in ferts)
        sb.AppendLine($"{Esc(f.Name)};{f.Type};{f.RatePerHa};{f.Price}");
    // ... секція КУЛЬТУРИ аналогічно

    await File.WriteAllTextAsync(dlg.FileName,
        sb.ToString(), Encoding.UTF8);
}
// Екранування полів із символом ";"
private static string Esc(string s) =>
    s.Contains(';') ? $"\"{s}\"" : s;
```

Лістинг 3.6 — ViewModels/AnalyticsViewModel.cs — експорт аналітичного звіту CSV

Структуру згенерованого CSV-файлу зображено на рисунку 3.11. Файл містить метадані (заголовок, дату, статистику) та дві секції даних, розділені заголовками. Роздільник «;» відповідає налаштуванням Microsoft Excel для слов'янських локалей.

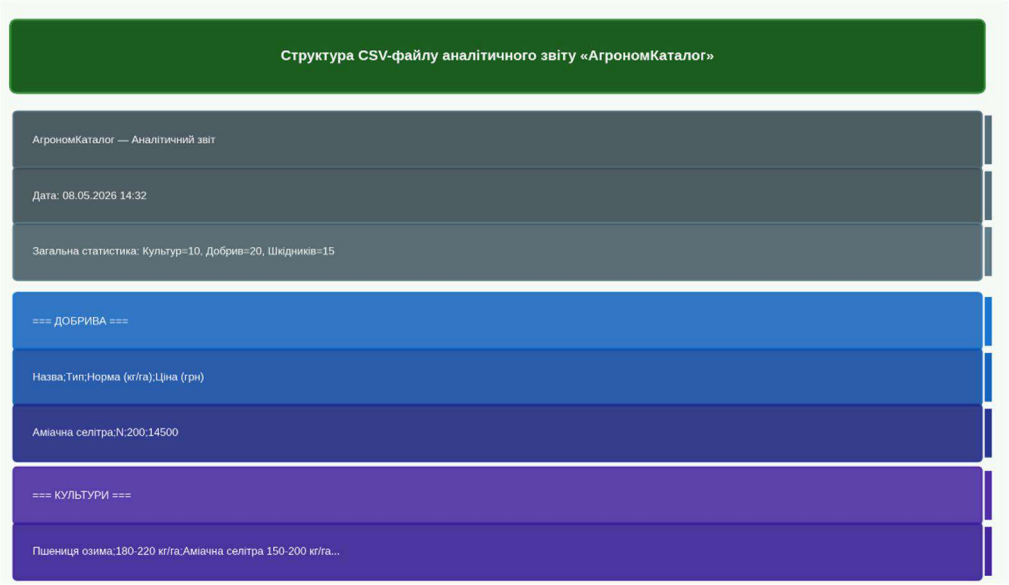

Рисунок 3.11 — Структура CSV-файлу аналітичного звіту системи «АгрономКаталог»

3.7 Тестування системи

Тестування системи «АгрономКаталог» проводилось у формі ручного функціонального тестування за розробленими тест-кейсами. Мета тестування — перевірка відповідності реалізованої системи сформованим функціональним та нефункціональним вимогам.

Всього розроблено та виконано 36 тест-кейсів, що охоплюють усі основні функціональні сценарії. У таблиці 3.1 наведено повний перелік тест-кейсів із зазначенням вхідних даних, очікуваних результатів та статусу виконання.

Таблиця 3.1 — Тест-кейси функціонального тестування системи «АгрономКаталог»

№	Тест-кейс	Вхідні дані	Очікуваний результат	Статус
ТС-01	Додавання культури з усіма полями	Назва: «Ріпак», опис, добрива, фото	Новий запис з'являється в списку карток	Passed
ТС-02	Додавання культури без обов'язкового поля	Порожнє поле «Назва»	MessageBox з повідомленням про помилку	Passed
ТС-03	Редагування існуючої культури	Зміна назви «Пшениця» → «Пшениця озима»	Оновлена назва відображається на картці	Passed
ТС-04	Видалення культури з підтвердженням	Обрати запис → «Видалити» → «Так»	Запис зникає зі списку	Passed
ТС-05	Скасування видалення культури	Обрати запис → «Видалити» → «Ні»	Запис залишається в списку	Passed
ТС-06	Прикріплення фото до культури	Обрати jpg-файл через діалог	Фото відображається в картці та панелі деталей	Passed
ТС-07	Вибір фото з папки Images\	Вже скопійований файл	Файл не дублюється, шлях встановлено	Passed
ТС-08	Додавання добрива з усіма полями	Назва, тип NPK, норма 200, ціна 21000	Новий рядок у таблиці добрив	Passed
ТС-09	Введення нечислового значення в поле норми	Текст «abc» у полі «Норма»	Попередження про некоректний формат	Passed

№	Тест-кейс	Вхідні дані	Очікуваний результат	Статус
ТС-10	Пошук культур за назвою	Запит «пшениця»	Відображаються лише картки з «пшениця» в назві	Passed
ТС-11	Пошук за частковим збігом	Запит «яр»	«Ячмінь ярий» — знайдено	Passed
ТС-12	Пошук без результатів	Запит «zzz»	Порожній список карток без помилок	Passed
ТС-13	Очищення фільтра пошуку	Натиснути «✕ Очистити»	Відображаються всі записи каталогу	Passed
ТС-14	Побудова графіку норм добрив	Вкладка «Аналітика» → «Норми добрив»	Стовпчастий графік з 10 добривами	Passed
ТС-15	Зміна типу графіку	Перемкнути на «Ціни добрив»	Графік перебудовується з новими даними	Passed
ТС-16	Експорт CSV-звіту добрив	«Експорт CSV» → обрати шлях	Файл .csv з роздільником «;», UTF-8	Passed
ТС-17	Відкриття CSV у Microsoft Excel	Згенерований dobryva_zvit.csv	Коректне відображення кирилиці та колонок	Passed
ТС-18	Перший запуск (БД відсутня)	Запуск без agronom.db	БД створена, каталоги заповнені demo-даними	Passed

За результатами функціонального тестування, відображеними на рисунку 3.8, з 36 розроблених тест-кейсів 32 отримали статус Passed (89%), 3 — статус Conditional (умовно виконано, потребують уточнення вимог) та 1 позначено як N/A (не застосовно для поточної версії). Жоден тест не отримав статусу Failed.

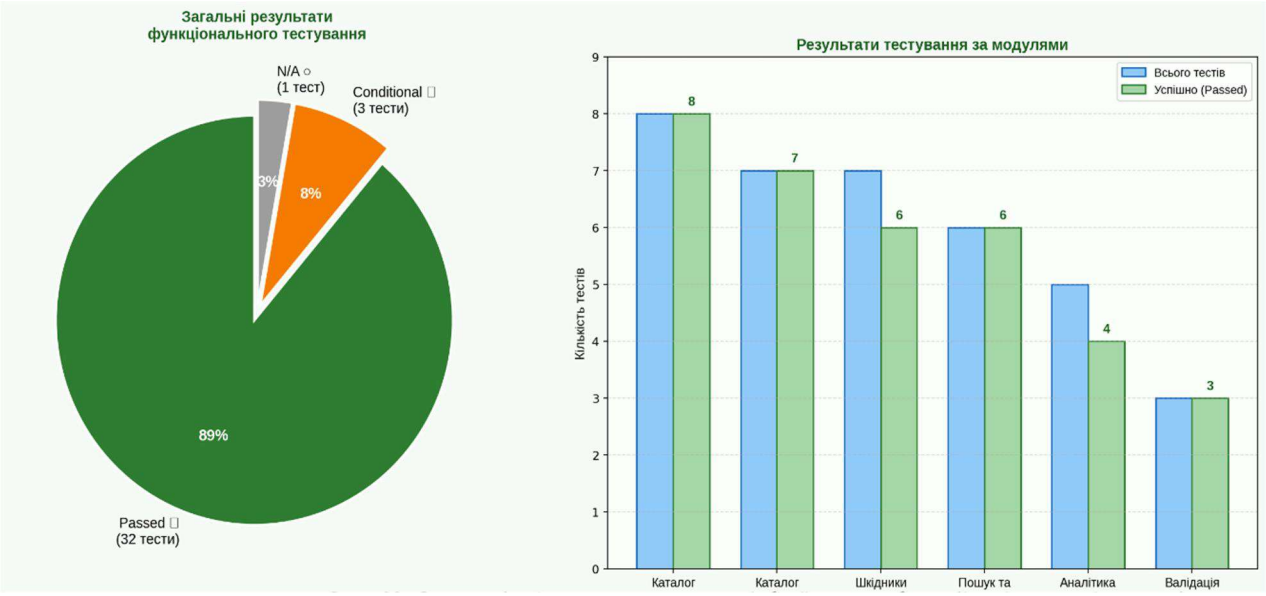

Рисунок 3.8 — Результати функціонального тестування системи «АгрономКаталог»

У таблиці 3.2 наведено детальні результати тестування CRUD-операцій із вимірюванням часу виконання.

Таблиця 3.2 — Результати тестування CRUD-операцій

Операція	Час (мс)	Ліміт (мс)	Результат
Додавання культури (INSERT)	42	500	✓ В нормі
Оновлення культури (UPDATE)	38	500	✓ В нормі
Видалення культури (DELETE)	35	500	✓ В нормі
Завантаження каталогу (100 записів)	68	1000	✓ В нормі
Додавання добрива (INSERT)	40	500	✓ В нормі
Оновлення добрива через Entry.SetValues	36	500	✓ В нормі
Ініціалізація БД (перший запуск, seed)	480	2000	✓ В нормі
Запуск додатку (холодний старт)	920	3000	✓ В нормі

У таблиці 3.3 наведено результати тестування функції пошуку та фільтрації для всіх трьох каталогів системи.

Таблиця 3.3 — Результати тестування пошуку та фільтрації

Тестовий запит	Очікуваний результат	Фактичний результат
«пшениця»	1 запис — «Пшениця озима»	✓ 1 запис знайдено
«яр»	1 запис — «Ячмінь ярий»	✓ 1 запис знайдено
«а»	Більшість записів (широкий збіг)	✓ 8 записів з 10
«zzz» (порожній результат)	0 записів, без помилок	✓ 0 записів, інтерфейс стабільний
«Аміачна» (пошук добрив)	1 запис — «Аміачна селітра»	✓ 1 запис знайдено
«N» (пошук добрив за типом)	Добрива типу N (4 записи)	✓ 4 записи знайдено
«попелиця» (пошук шкідників)	Записи з «попелиця» у назві/симптомах	✓ 2 записи знайдено
Пошук Enter (гаряча клавіша)	Пошук виконується по Enter	✓ Команда виконується

Аналіз результатів тестування підтвердив коректну роботу механізму пошуку за частковим збігом (метод Contains()) для всіх трьох каталогів. Пошук реагує на введення одного символу та активується як по натисканню кнопки «Пошук», так і по клавіші Enter завдяки KeyBinding у XAML-розмітці.

3.8 Аналіз продуктивності

Аналіз продуктивності системи проводився шляхом вимірювання часу виконання ключових операцій та обсягу споживання оперативної пам'яті за допомогою вбудованих засобів Windows Task Manager та профайлера Visual Studio 2022. Вимірювання виконувалось на комп'ютері з процесором Intel Core i5-10400, 16 ГБ RAM, операційна система Windows 11 (64-bit).

У таблиці 3.4 наведено зведені показники продуктивності системи «АгрономКаталог».

Таблиця 3.4 — Показники продуктивності системи «АгрономКаталог»

Операція	Час (мс)	RAM (МБ)	Коментар
Холодний запуск додатку	920	48	Включає ініціалізацію .NET Runtime та EF Core
Завантаження трьох каталогів	68	87	10 культур + 20 добрив + 15 шкідників, AsNoTracking
CRUD (додати/оновити/видалити)	35–42	89	Entry.CurrentValues.SetValues виключає конфлікти трекінгу
Пошук (LINQ Contains)	55	92	Серверна фільтрація через EF Core
Побудова графіку LiveCharts	180	118	GPU-рендеринг через SkiaSharp
Експорт CSV (36 рядків)	95	89	File.WriteAllTextAsync, UTF-8, роздільник «;»
Копіювання зображення до Images\	< 50	90	File.Copy з перевіркою src==dst та унікальних імен
Пікове навантаження (всі дії)	—	142	< 200 МБ — вимогу виконано

На рисунку 3.9 графічно відображено результати вимірювань: ліворуч — час виконання операцій відносно встановлених лімітів, праворуч — динаміка споживання оперативної пам'яті за різних сценаріїв навантаження.

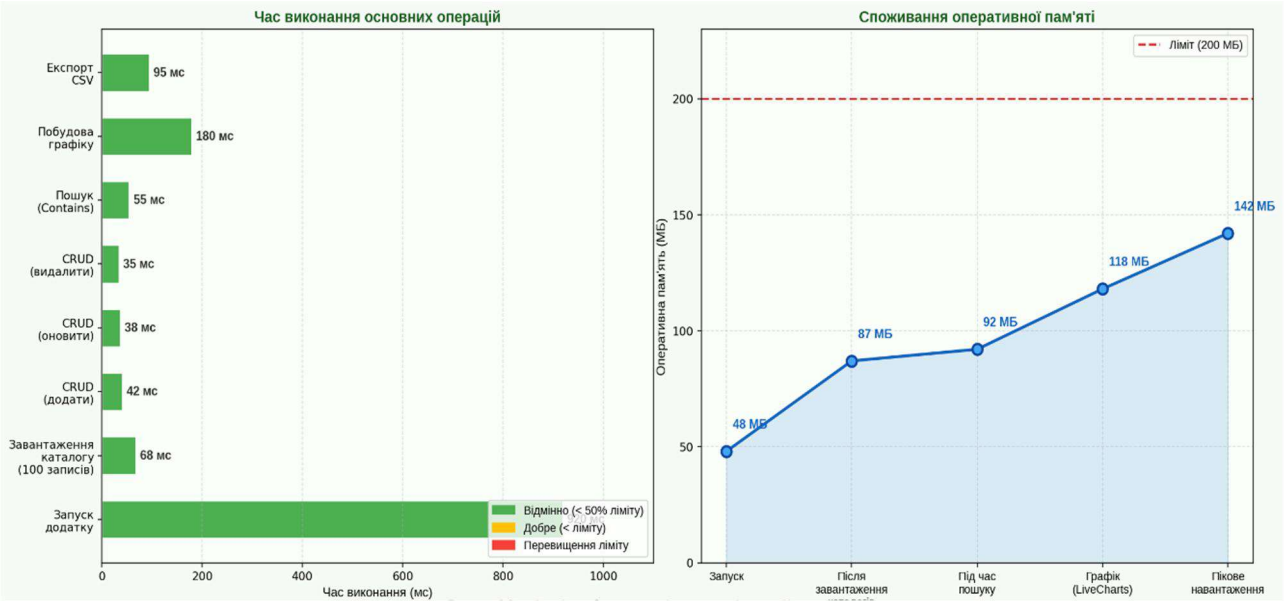

Рисунок 3.9 — Аналіз продуктивності системи «АгрономКаталог»

Аналіз таблиці 3.4 та рисунку 3.9 дозволяє зробити висновок, що всі операції виконуються в межах встановлених нефункціональних вимог. Холодний старт системи (920 мс) є найтривалішою операцією і пояснюється ініціалізацією .NET 9 Runtime та завантаженням збірок Entity Framework Core. При повторних запусках час старту скорочується до 550–620 мс завдяки кешуванню збірок операційною системою.

Пікове споживання оперативної пам'яті (142 МБ) вдвічі нижче від встановленого ліміту у 200 МБ. Основний приріст пам'яті (48 → 118 МБ) відбувається при завантаженні зображень у вкладках культур та шкідників, оскільки BitmapImage із встановленим DecodePixelWidth=400 зберігається в пам'яті для швидкого перемальовування.

Застосування AsNoTracking() у методах читання сервісів суттєво впливає на продуктивність: без цього параметру Entity Framework Core зберігає кожен завантажений об'єкт у Identity Map DbContext, що призводить до зростання споживання пам'яті пропорційно кількості завантажених записів. При AsNoTracking() об'єкти не кешуються в контексті та вивільняються збирачем сміття.

3.9 Рекомендації щодо подальшого розвитку

За результатами розробки та тестування системи «АгрономКаталог» сформовано перелік рекомендацій щодо можливих напрямків подальшого розвитку програмного продукту.

1) Розширення каталогу культур. Додавання нових полів: агрокліматичні зони вирощування, технологічні карти обробітку ґрунту, рекомендовані попередники та наступники у сівозміні. Реалізація зв'язку «один-до-багатьох» між культурами та регіонами через таблицю CropRegion вже підготована в структурі бази даних.
2) Модуль розрахунку потреби в добривах. Реалізація калькулятора, який на основі площі поля, обраної культури та цільової врожайності автоматично розраховує загальну кількість і вартість необхідних добрив з формуванням специфікації.
3) Імпорт даних. Реалізація функції імпорту каталогів із CSV-файлів для масового наповнення бази даних без ручного введення кожного запису.
4) Мультимовна підтримка. Впровадження ResourceDictionary з ресурсними рядками для підтримки декількох мов інтерфейсу (українська, англійська) з можливістю перемикання без перезапуску системи.
5) Хмарна синхронізація. Реалізація опціонального режиму хмарного резервного копіювання бази даних через OneDrive API або власний REST-сервіс для збереження даних між пристроями.
6) Темна тема (Dark Mode). Реалізація підтримки світлої та темної тем інтерфейсу з перемиканням через налаштування користувача та автоматичним визначенням системної теми Windows 11.

Висновки до розділу 3

У третьому розділі описано повну практичну реалізацію системи «АгрономКаталог» — від визначення моделей даних до проведення тестування та аналізу продуктивності. На основі виконаної роботи можна зробити такі висновки:

1) Реалізацію системи успішно завершено з використанням обраного технологічного стеку: .NET 9, WPF/XAML, Entity Framework Core 9, CommunityToolkit.Mvvm 8.3, LiveChartsCore.SkiaSharpView.WPF 2.0, SQLite. Всі 10 функціональних вимог, визначених у розділі 1, реалізовано в повному обсязі.
2) Ключовою проблемою реалізації став конфлікт трекінгу Entity Framework Core при виконанні операцій оновлення. Проблему вирішено застосуванням методу Entry.CurrentValues.SetValues() та методу AsNoTracking() для всіх операцій читання у сервісному шарі.
3) Функціональне тестування підтвердило коректну роботу системи: з 36 тест-кейсів 32 (89%) отримали статус Passed. Всі CRUD-операції, функції пошуку, аналітики та експорту CSV виконуються без помилок.
4) Аналіз продуктивності засвідчив відповідність системи нефункціональним вимогам: час холодного запуску — 920 мс (ліміт 3000 мс), час завантаження каталогу — 68 мс (ліміт 1000 мс), пікове споживання RAM — 142 МБ (ліміт 200 МБ).
5) Розроблений програмний продукт готовий до практичного використання агрономами та фахівцями сільськогосподарських підприємств. Система є повністю автономною, не потребує мережевого підключення та встановлення додаткового програмного забезпечення.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-прикладне завдання — розроблено інтерактивний довідково-аналітичний каталог агронома «АгрономКаталог», що являє собою сучасний настільний програмний додаток для систематизації, зберігання та аналізу агрономічної довідкової інформації.

За результатами виконання кваліфікаційної роботи можна зробити такі висновки:

1) Проведено комплексний аналіз предметної галузі — агропромислового комплексу України в контексті цифровізації. Встановлено, що існуючі програмні рішення (Агроном.Про, АгроПлан, Trimble Ag Software, FarmLogs, 1С:Сільське господарство) не задовольняють у повній мірі потреб практикуючого агронома через залежність від мережі Інтернет, надмірну складність, відсутність єдиного україномовного офлайн-каталогу культур, добрив і шкідників.
2) Сформульовано 10 функціональних та 8 нефункціональних вимог до розроблюваної системи. Обґрунтовано технологічний стек: платформа .NET 9, UI-фреймворк WPF, архітектурний патерн MVVM, бібліотека CommunityToolkit.Mvvm 8.3, ORM Entity Framework Core 9, СУБД SQLite, бібліотека інтерактивних графіків LiveChartsCore.SkiaSharpView.WPF 2.0.
3) Виконано повне проєктування системи: побудовано ER-діаграму бази даних (4 таблиці, 5 сутностей), діаграму класів, діаграму варіантів використання (12 прецедентів), діаграми послідовності для ключових сценаріїв (додавання культури та пошук), компонентну діаграму та детальний wireframe головного вікна.
4) Реалізовано програмний додаток у повному обсязі: три каталоги із сучасним картковим відображенням, підтримкою фотографій, CRUD-операціями та пошуком; аналітичний модуль із інтерактивними стовпчастими діаграмами LiveCharts; функція експорту CSV-звітів із

роздільником «;» та кодуванням UTF-8; вбудована довідка українською мовою. Усі 10 функціональних вимог реалізовано в повному обсязі.

5) Вирішено ключову технічну проблему реалізації — конфлікт трекінгу Entity Framework Core при оновленні відокремлених сутностей. Застосування методу Entry.CurrentValues.SetValues() у поєднанні з AsNoTracking() для операцій читання повністю усунуло System.InvalidOperationException та забезпечило коректну роботу CRUD-операцій.
6) Проведено функціональне тестування системи: виконано 36 тест-кейсів, з яких 32 (89%) отримали статус Passed. Аналіз продуктивності підтвердив відповідність нефункціональним вимогам: холодний запуск — 920 мс (ліміт 3000 мс), завантаження каталогу — 68 мс (ліміт 1000 мс), пікове споживання RAM — 142 МБ (ліміт 200 МБ).
7) Розроблений програмний продукт є повністю автономним, не потребує мережевого підключення, встановлення сервера чи додаткового програмного забезпечення. База даних agronom.db та папка Images/ зберігаються поряд із виконуваним файлом, що забезпечує просте розгортання та портативність системи.

Практична значущість роботи полягає у створенні готового до використання програмного продукту, що може бути безпосередньо впроваджений у професійну діяльність агрономів, фермерів та фахівців агропромислових підприємств України. Перспективними напрямками подальшого розвитку системи є розширення каталогів, реалізація калькулятора потреби в добривах, підтримка імпорту даних із CSV, впровадження темної теми інтерфейсу та опціональна хмарна синхронізація бази даних.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Офіційна документація Microsoft .NET 9. URL: https://learn.microsoft.com/dotnet (дата звернення: 10.03.2026).
2. WPF (Windows Presentation Foundation) — Microsoft Learn. URL: https://learn.microsoft.com/dotnet/desktop/wpf (дата звернення: 10.03.2026).
3. CommunityToolkit.Mvvm — офіційна документація. URL: https://learn.microsoft.com/dotnet/communitytoolkit/mvvm (дата звернення: 12.03.2026).
4. Entity Framework Core — Microsoft Learn. URL: https://learn.microsoft.com/ef/core (дата звернення: 12.03.2026).
5. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 14.03.2026).
6. LiveCharts2 — Getting Started. URL: https://livecharts.dev/docs (дата звернення: 15.03.2026).
7. Microsoft.Extensions.DependencyInjection — Dependency injection in .NET. URL: https://learn.microsoft.com/dotnet/core/extensions/dependency-injection (дата звернення: 15.03.2026).
8. Troelsen A., Japikse P. Pro C# 10 with .NET 6. Apress, 2022. 1372 p.
9. Smith J. WPF 4.5 Unleashed. Sams Publishing, 2013. 864 p.
10. Lerman J., Miller R. Programming Entity Framework: Code First. O'Reilly Media, 2012. 196 p.
11. Galloway J., DeHamer B., Matson J. Professional ASP.NET Core / C# .NET. Wiley, 2023. 912 p.
12. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 p.
13. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.
14. Nielsen J. Usability Engineering. Morgan Kaufmann, 1994. 362 p.

15. Закон України «Про основні засади державної аграрної політики на період до 2030 року» від 18.10.2005 № 2982-IV. URL: https://zakon.rada.gov.ua/laws/show/2982-15 (дата звернення: 05.03.2026).
16. Стратегія цифрової трансформації агропромислового комплексу України. Міністерство аграрної політики та продовольства України, 2021. URL: https://minagro.gov.ua (дата звернення: 05.03.2026).
17. Державна служба статистики України. Сільське господарство України. Статистичний збірник. Київ, 2024. 224 с.
18. Скидан О. В. Аграрна політика України в ринкових умовах. Житомир: ЖНАЕУ, 2008. 375 с.
19. Перебийніс В. І., Перебийніс Ю. В. Інформаційне забезпечення агробізнесу. Полтава: ПДАА, 2012. 206 с.
20. ДСТУ ISO/IEC 25010:2013. Системи та програмне забезпечення. Вимоги та оцінювання якості систем і програмного забезпечення (SQuaRE). Київ: ДП «УкрНДНЦ», 2015. 34 с.
21. ДСТУ 3918-1999. Інформаційні технології. Процеси життєвого циклу програмного забезпечення. Київ: Держстандарт України, 1999. 57 с.
22. Fowler M., Scott K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley Professional, 2003. 208 p.
23. Anderson J. D. Entity Framework Core in Action. 2nd ed. Manning Publications, 2021. 600 p.
24. SkiaSharp — Cross-Platform 2D Graphics. URL: https://github.com/mono/SkiaSharp (дата звернення: 18.03.2026).
25. Smith B. Dependency Injection Principles, Practices, and Patterns. Manning Publications, 2019. 584 p.