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:

«Інформаційна систем обліку тваринництва фермерського господарства»

Виконав здобувач освіти групи П-41
Перепелиця Максим Олександрович

Керівник роботи:
Устименко Ярослав Іванович

Рецензент:

Житомир – 2026

Зміст

					КРФМБ.122.041.045.2026.ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розробив		Перепелиця М О			Інформаційна систем обліку тваринництва фермерського господарства	Літ.	Арк.	Аркушів
Перевірив		Устименко Я.І.					3	57
Рецензент		.				ЖАТФК		
Н. Контр.								
Затвердив.		Гнатюк О.Ф.						

РЕФЕРАТ

Пояснювальна записка: 57 сторінки, 23 рисунки, 17 таблиць та 6 лістингів програмного коду.

Кваліфікаційна робота присвячена розробці інформаційної системи обліку тваринництва фермерського господарства «ФермерОблік». У роботі проведено аналіз предметної області та існуючих програмних рішень для автоматизації тваринницьких господарств, сформульовано функціональні та нефункціональні вимоги до системи.

Систему спроєктовано на основі трирівневої архітектури із застосуванням шаблону MVVM. Розроблено реляційну базу даних з семи таблиць засобами Entity Framework Core 9 та SQLite. Реалізовано модулі: авторизації з BCrypt-хешуванням, обліку тварин, виробництва та догляду, звітності з діаграмами та експортом у PDF/CSV, управління користувачами. Забезпечено розмежування доступу для трьох ролей: Адміністратор, Фермер, Ветеринар. Проведено функціональне тестування (28 тест-кейсів, 100 % успіху).

Ключові слова: інформаційна система, тваринництво, WPF, MVVM, Entity Framework Core, SQLite, BCrypt, фермерське господарство.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

.NET	Платформа розробки програмного забезпечення від Microsoft (.NET 9)
API	Application Programming Interface — прикладний програмний інтерфейс
BCrypt	Алгоритм хешування паролів Blowfish Crypt
BVD	Bovine Viral Diarrhea — вірусна діарея великої рогатої худоби
C#	Мова програмування C Sharp, розроблена компанією Microsoft
CRUD	Create, Read, Update, Delete — чотири базові операції з даними
CSV	Comma-Separated Values — текстовий формат зберігання табличних даних
EF Core	Entity Framework Core — ORM-фреймворк для .NET
EMU	English Metric Unit — одиниця вимірювання в документах Office (1/914400 дюйма)
ER-діаграма	Entity-Relationship Diagram — діаграма «сутність-зв'язок»
ERP	Enterprise Resource Planning — планування ресурсів підприємства
FK	Foreign Key — зовнішній ключ у реляційній базі даних
FMD	Foot-and-Mouth Disease — ящур
MVVM	Model-View-ViewModel — архітектурний шаблон для WPF-застосунків
SQL	Structured Query Language — мова структурованих запитів до БД
SQLite	Вбудована реляційна база даних, що зберігає дані у файлі
UI	User Interface — інтерфейс користувача
UML	Unified Modeling Language — уніфікована мова моделювання
UTF-8	Unicode Transformation Format 8-bit — кодування символів Unicode
WPF	Windows Presentation Foundation — фреймворк для розробки UI на платформі .NET
XAML	Extensible Application Markup Language — мова розмітки для WPF
XML	eXtensible Markup Language — розширювана мова розмітки
ВРХ	Велика рогата худоба
ДСТУ	Державний стандарт України
ЗЦМ	Замінник цільного молока — кормова добавка для телят
КЧС	Класична чума свиней — інфекційне захворювання тварин
ПЗ	Програмне забезпечення
СУБД	Система управління базами даних
ТЗ	Технічне завдання

ВСТУП

Актуальність теми

Сучасне фермерське господарство є складною організаційно-виробничою структурою, ефективне управління якою неможливе без своєчасного та достовірного обліку всіх виробничих процесів. Тваринництво, як одна з ключових галузей агропромислового комплексу України, потребує систематичного ведення записів щодо стану поголів'я, продуктивності тварин, проведення ветеринарних заходів та витрат на утримання.

За даними Державної служби статистики України, у 2024–2025 роках у країні функціонують понад 47 тисяч фермерських господарств, значна частина яких досі використовує паперові журнали або примітивні електронні таблиці для ведення обліку. Це призводить до втрати даних, помилок у звітності, неефективного планування кормової бази та несвоєчасного реагування на захворювання тварин.

Впровадження спеціалізованих інформаційних систем у практику фермерських господарств дозволяє автоматизувати рутинні облікові операції, знизити ймовірність помилок, забезпечити оперативний доступ до актуальної інформації та отримувати аналітичні звіти для підтримки управлінських рішень. Таким чином, розробка інформаційної системи обліку тваринництва є актуальним практичним завданням, яке має безпосередній вплив на економічну ефективність сільськогосподарського виробництва.

Метою кваліфікаційної роботи є проєктування та розробка інформаційної системи обліку тваринництва фермерського господарства, яка забезпечує автоматизацію процесів реєстрації тварин, обліку продуктивності, ветеринарного догляду та формування звітності.

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

1) проаналізувати предметну область та існуючі рішення для автоматизації тваринництва;

2) сформулювати функціональні та нефункціональні вимоги до системи;

3) обрати та обґрунтувати технологічний стек розробки;

4) спроєктувати архітектуру системи, структуру бази даних та інтерфейс користувача;

5) реалізувати основні модулі системи: облік тварин, виробництво, ветеринарний догляд, звітність;

6) забезпечити розмежування доступу користувачів за ролями;

7) реалізувати функції експорту звітів у формати PDF та CSV;

8) провести тестування розробленої системи та оцінити її відповідність вимогам.

Об'єктом дослідження є процеси управління інформацією в тваринницькій галузі фермерського господарства, зокрема облік поголів'я, реєстрація виробничих показників, ведення ветеринарної документації та аналіз витрат.

Предметом дослідження є методи та засоби розробки настільних інформаційних систем для автоматизації обліку тваринництва з використанням технологій .NET, WPF та SQLite.

У процесі виконання кваліфікаційної роботи використовувались такі **методи дослідження**:

– системний аналіз — для вивчення предметної області та декомпозиції задачі на підзадачі;

– об'єктно-орієнтоване проєктування — для побудови архітектури системи відповідно до шаблону MVVM;

– концептуальне та логічне моделювання даних — для розробки структури реляційної бази даних;

– порівняльний аналіз — для вибору технологічного стеку та обґрунтування архітектурних рішень;

– тестування програмного забезпечення — для перевірки функціональності та надійності розробленої системи.

Практичне значення роботи полягає у створенні повнофункціонального програмного продукту — настільного застосунку

«ФермерОблік», який може бути впроваджений у реальних фермерських господарствах для ведення автоматизованого обліку тваринництва.

Розроблена система дозволяє: вести реєстр тварин з повною характеристикою кожної особини; фіксувати щоденні показники виробництва молока, м'яса, вовни та інших продуктів; документувати факти годування, вакцинації та лікування; формувати звіти за довільний часовий період з можливістю експорту у форматах PDF та CSV; здійснювати резервне копіювання бази даних. Система підтримує три ролі користувачів — адміністратор, фермер, ветеринар — з відповідним розмежуванням прав доступу.

Результати кваліфікаційної роботи можуть бути використані як основа для подальшого розвитку системи з додаванням веб-інтерфейсу, мобільного застосунку або хмарного сховища даних, що суттєво розширить можливості дистанційного управління господарством.

Структура та обсяг роботи

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 75 сторінок. Робота містить 12 рисунків, 15 таблиць, 6 лістингів програмного коду. Список використаних джерел налічує 20 найменувань.

У першому розділі проведено аналіз предметної області, розглянуто особливості процесів обліку тваринництва, виконано огляд та порівняльний аналіз існуючих програмних рішень, сформульовано вимоги до системи та обґрунтовано вибір технологій розробки.

У другому розділі описано процес проєктування системи: розроблено архітектуру на основі шаблону MVVM, спроєктовано реляційну базу даних з сімома таблицями, побудовано діаграми варіантів використання, послідовності та станів, описано механізми захисту даних і розмежування доступу.

У третьому розділі описано реалізацію всіх модулів системи з наведенням ключових фрагментів програмного коду, представлено результати функціонального тестування та надано інструкцію користувача.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика фермерського господарства як об'єкта автоматизації

Фермерське господарство є однією з основних форм організації сільськогосподарського виробництва в Україні. Відповідно до Закону України «Про фермерське господарство» від 19.06.2003 № 973-IV, фермерське господарство є формою підприємницької діяльності громадян, які виявили бажання виробляти товарну сільськогосподарську продукцію, займатися її переробкою та реалізацією.

Сучасне фермерське господарство, що займається тваринництвом, являє собою складну виробничо-господарську систему, яка включає декілька взаємопов'язаних підсистем: управління поголів'ям тварин; виробництво та реалізація тваринницької продукції; ветеринарне обслуговування; кормова база та годування; фінансовий облік витрат і доходів; управління персоналом.

Ефективне управління господарством вимагає постійного моніторингу стану кожної з перерахованих підсистем. При цьому виникає значний обсяг інформації, що підлягає реєстрації, зберіганню та аналізу. Зокрема, для господарства з поголів'ям 20–50 тварин щоденний обсяг записів може включати дані про надої молока від кожної корови, норми видачі кормів для кожної групи тварин, ветеринарні процедури та медикаменти, стан здоров'я поголів'я.

Традиційний паперовий облік при такому обсязі інформації є неефективним з декількох причин. По-перше, пошук і аналіз накопичених даних займає значний час. По-друге, висока ймовірність помилок при ручному заповненні журналів. По-третє, відсутня можливість автоматичного формування звітності для контролюючих органів. По-четверте, зберігання паперових документів вимагає фізичного простору та захисту від пошкодження.

Таким чином, фермерське господарство є типовим об'єктом автоматизації, де впровадження інформаційної системи дозволяє суттєво підвищити ефективність управлінської та облікової діяльності.

1.2 Аналіз процесів обліку тваринництва

Облік тваринництва охоплює декілька взаємопов'язаних процесів, кожен з яких має свою специфіку та інформаційну модель.

Облік поголів'я тварин є базовим процесом, що включає реєстрацію кожної тварини при надходженні до господарства з фіксацією виду, породи, статі, дати народження, маси та присвоєного ідентифікаційного номера чи кличці; відстеження змін у поголів'ї (народження, придбання, забій, загибель, продаж); моніторинг стану здоров'я кожної тварини з можливістю зміни статусу (здорова, хвора, на лікуванні, карантин).

Облік виробництва продукції передбачає щоденну або щотижневу реєстрацію кількості виробленої продукції в розрізі кожної тварини або групи тварин: надоїв молока (кг або л), настригу вовни (кг), яєць (шт.), відгодівельних показників для м'ясних порід. Ці дані є основою для розрахунку продуктивності поголів'я та планування виробничої діяльності.

Облік годування включає документування раціону годівлі для кожної групи тварин: вид і назву корму, добову норму видачі, фактично використану кількість та вартість. Контроль витрат кормів дозволяє оптимізувати структуру раціону та знижувати собівартість продукції.

Ветеринарний облік є критично важливим для збереження здоров'я поголів'я та включає ведення журналу вакцинацій із зазначенням назви вакцини, дати проведення, відповідального ветеринара та планової дати ревакцинації; реєстрацію захворювань, діагнозів та призначеного лікування; фіксацію витрат на ветеринарні препарати та послуги.

Облік витрат охоплює всі категорії витрат господарства: корми, ветеринарні послуги та медикаменти, обладнання та інвентар, паливно-мастильні матеріали, заробітну плату персоналу. Систематизований облік

витрат є необхідною умовою для розрахунку собівартості продукції та оцінки рентабельності господарства.

1.3 Огляд існуючих систем автоматизації тваринництва

На сучасному ринку програмного забезпечення існує ряд рішень для автоматизації тваринницьких господарств. Розглянемо найбільш поширені з них.

AgroBase — комплексна хмарна платформа для управління сільськогосподарськими підприємствами. Система надає широкий набір інструментів для обліку тваринництва, включаючи генеалогічні записи, відстеження продуктивності та ветеринарний модуль. Основним недоліком є повністю англомовний інтерфейс та висока вартість підписки (від 120 USD на місяць), що робить її недоступною для малих і середніх фермерських господарств України.

FarmFacts — спеціалізована система для молочного скотарства, розроблена в Данії. Має потужний аналітичний модуль та підтримує інтеграцію з доїльним обладнанням. Недоліки: відсутність україномовного інтерфейсу, хмарна архітектура вимагає постійного інтернет-з'єднання, вартість ліцензії недоступна для малих господарств.

1С:Сільгосп — рішення на базі платформи 1С:Підприємство, що підтримує облік тваринництва в рамках комплексної облікової системи. Перевагою є адаптованість до українського законодавства. Недоліки: висока складність налаштування та обслуговування, необхідність залучення фахівців з 1С, значна вартість початкового впровадження.

Microsoft Excel та Access — найпоширеніший спосіб ведення обліку в малих господарствах. Не вимагає додаткових витрат на ліцензування. Проте має суттєві обмеження: відсутність структурованої бази даних, неможливість багатокористувацької роботи, висока трудомісткість формування звітів, відсутність розмежування доступу.

Таблиця 1.1 — Порівняльний аналіз існуючих систем автоматизації тваринництва

Критерій	AgroBase	FarmFacts	1С:Сільгосп	Excel/Access	ФермерОблік(розробл.)
Мова інтерфейсу	Англ.	Англ.	Рос./Укр.	Укр./будь-яка	Українська
Ліцензія	Платна	Платна	Платна	Безкоштовна	Безкоштовна
Облік поголів'я	✓	✓	✓	Частково	✓
Ветеринарний облік	✓	✓	Частково	–	✓
Виробн. записи	✓	✓	✓	Частково	✓
Звіти/аналітика	✓	✓	✓	Частково	✓
Ролі/доступ	✓	✓	✓	–	✓
Резерв. копія	✓	✓	✓	Вручну	✓
Офлайн-робота	–	Частково	✓	✓	✓
Навч. кривизна	Висока	Висока	Середня	Низька	Низька
Вартість (міс.)	$120+	$90+	$50+	0 грн	0 грн

Аналіз існуючих систем, представлений у таблиці 1.1, свідчить про те, що жодне з розглянутих рішень не задовольняє повною мірою потреби малих та середніх фермерських господарств України. Комерційні системи є надто дорогими та складними для впровадження, а простий облік в електронних таблицях не забезпечує необхідний рівень функціональності та захисту даних.

1.4 Обґрунтування необхідності розробки власної системи

На підставі проведеного аналізу предметної області та існуючих програмних рішень можна виділити основні проблеми, що обумовлюють необхідність розробки власної спеціалізованої інформаційної системи:

1) Мовний бар'єр. Більшість спеціалізованих систем для тваринництва не мають повноцінного україномовного інтерфейсу, що ускладнює їх використання рядовими працівниками ферм.

2) Висока вартість. Комерційні рішення передбачають щомісячну абонентську плату від 50 до 120 USD, що є неприйнятним для малих господарств із обмеженим бюджетом.

3) Надлишкова складність. Великі ERP-системи, розраховані на агрохолдинги, містять надмірну кількість модулів, що ускладнює навчання персоналу та щоденне використання.

4) Залежність від інтернету. Хмарні рішення не придатні для використання в умовах нестабільного або відсутнього інтернет-з'єднання, характерного для сільської місцевості.

5) Відсутність ветеринарного модуля. В простих рішеннях типу Excel відсутній структурований облік вакцинацій та лікування, що не відповідає вимогам ветеринарного законодавства.

Розробка власної інформаційної системи дозволить: забезпечити повністю україномовний інтерфейс; надати всі необхідні функції в рамках єдиного застосунку; забезпечити офлайн-роботу без залежності від інтернету; реалізувати зрозумілий інтерфейс з мінімальним часом навчання; організувати розмежування доступу для різних категорій персоналу ферми.

1.5 Формулювання вимог до системи

На основі аналізу предметної області та потреб користувачів сформульовано функціональні та нефункціональні вимоги до розроблюваної системи.

Функціональні вимоги описують конкретні функції, які повинна виконувати система, та наведені в таблиці 1.2.

Таблиця 1.2 — Функціональні вимоги до інформаційної системи

№	Модуль	Вимога
1	Авторизація	Система повинна підтримувати вхід за логіном/паролем з BCrypt-хешуванням

2	Авторизація	Система повинна розрізняти ролі: Адміністратор, Фермер, Ветеринар
3	Облік тварин	Система повинна забезпечити CRUD-операції для записів тварин
4	Облік тварин	Система повинна зберігати: кличку, вид, породу, стать, дату народження, вагу, стан здоров'я
5	Облік тварин	Система повинна підтримувати пошук і фільтрацію тварин за видом, статусом, кличкою
6	Виробництво	Система повинна фіксувати записи виробництва продукції (молоко, м'ясо, вовна тощо)
7	Годування	Система повинна вести журнал годування з типом корму, кількістю та вартістю
8	Вакцинація	Система повинна реєструвати вакцинації із зазначенням ветеринара та дати ревакцинації
9	Лікування	Система повинна вести облік захворювань, діагнозів та проведеного лікування
10	Звітність	Система повинна формувати зведені звіти за обраний часовий діапазон
11	Звітність	Система повинна підтримувати експорт звітів у форматах PDF та CSV (роздільник «;»)
12	Адміністрування	Адміністратор повинен мати можливість керувати обліковими записами користувачів
13	База даних	Система повинна підтримувати резервне копіювання та відновлення бази даних

Нефункціональні вимоги визначають якісні характеристики системи, яких вона повинна дотримуватись при виконанні своїх функцій. Вони наведені в таблиці 1.3.

Таблиця 1.3 — Нефункціональні вимоги до інформаційної системи

№	Категорія	Вимога
1	Продуктивність	Завантаження списку тварин (до 1000 записів) не повинно перевищувати 2 секунди
2	Продуктивність	Генерація звіту за 6-місячний період — не більше 5 секунд
3	Надійність	Система повинна зберігати цілісність даних при аварійному завершенні роботи
4	Безпека	Паролі користувачів зберігаються виключно у хешованому вигляді (BCrypt, cost=11)

5	Безпека	Доступ до функцій адміністрування обмежений роллю «Адміністратор»
6	Зручність	Інтерфейс повинен бути повністю україномовним
7	Зручність	Час освоєння системи новим користувачем — не більше 2 годин
8	Переносність	Система повинна працювати на ОС Windows 10/11 без додаткових серверних компонентів
9	Переносність	База даних зберігається локально поруч з виконуваним файлом у папці «base»
10	Обслуговування	Система повинна підтримувати автоматичне резервне копіювання з ротацією (10 копій)

1.6 Вибір технологій розробки

Обґрунтований вибір технологій є критичним фактором успіху розробки програмного продукту. Враховуючи сформульовані вимоги, зокрема вимогу офлайн-роботи та орієнтованість на платформу Windows, розглядаються технології розробки настільних (desktop) застосунків.

Windows Presentation Foundation (WPF) — це фреймворк для розробки настільних застосунків на платформі .NET з використанням мови C#. WPF використовує декларативну мову XAML для опису інтерфейсу та підтримує потужний механізм прив'язки даних (data binding), що є основою шаблону MVVM. Перевагами WPF є нативна підтримка шаблону MVVM; потужна система прив'язки даних та команд; повністю керований рендеринг через DirectX; підтримка анімацій, стилів та шаблонів елементів; безкоштовність та відкритий код у складі .NET.

Windows Forms (WinForms) — більш зрілий фреймворк для настільних застосунків .NET, що має просту в освоєнні модель програмування. Проте WinForms має обмежену підтримку шаблону MVVM, менш гнучку систему стилізації інтерфейсу та застарілу архітектуру порівняно з WPF.

Electron — фреймворк для розробки крос-платформних настільних застосунків на основі веб-технологій (HTML, CSS, JavaScript). Хоча Electron забезпечує крос-платформність, він має значні недоліки: великий розмір

дистрибутиву, підвищене споживання оперативної пам'яті та відсутність нативної підтримки роботи з .NET та Entity Framework Core.

JavaFX — фреймворк для розробки настільних застосунків на мові Java. Забезпечує крос-платформність, проте має вищий поріг входу для розробників зі знанням C#, менш розвинену екосистему ORM-інструментів для роботи з SQLite.

Таблиця 1.4 — Порівняльний аналіз технологій розробки настільних застосунків

Критерій	WPF (.NET)	WinForms	Electron	JavaFX
Мова розробки	C#	C#	JS/TS	Java
Стиль UI	XAML + прив'язки	Designer	HTML/CSS	FXML
Шаблон MVVM	Нативна підтримка	Частково	Частково	Частково
Прив'язка даних	Потужна	Обмежена	Середня	Середня
EF Core підтримка	✓	✓	–	–
Платформа	Windows	Windows	Крос-плат.	Крос-плат.
Складність освоєння	Середня	Низька	Низька	Середня
Оцінка для проєкту	★★★★★	★★★☆☆	★★☆☆☆	★★★☆☆

За результатами порівняльного аналізу, наведеного в таблиці 1.4, для реалізації системи обрано технологічний стек: мова програмування C# (.NET 9); фреймворк інтерфейсу Windows Presentation Foundation (WPF); шаблон архітектури MVVM (Model-View-ViewModel); СУБД SQLite як вбудована реляційна база даних; Entity Framework Core 9 як ORM для роботи з базою даних; бібліотека BCrypt.Net для хешування паролів; CsvHelper для генерації CSV-файлів; PdfSharpCore для формування PDF-звітів.

Вибір SQLite як бази даних обумовлений тим, що вона є вбудованою СУБД, не потребує окремого серверного процесу, зберігає всі дані в одному файлі, підтримує транзакції та SQL-стандарт, є безкоштовною та широко використовується в настільних застосунках.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області інформаційної системи обліку тваринництва фермерського господарства. Встановлено, що сучасне тваринницьке господарство є складним об'єктом автоматизації, де щоденно генерується значний обсяг облікової інформації, яка потребує систематизованого зберігання та аналізу.

Аналіз процесів обліку тваринництва виявив п'ять основних напрямів: облік поголів'я, виробничий облік, облік годування, ветеринарний облік та облік витрат. Кожен з цих напрямів має власну інформаційну модель та специфічні вимоги до функціоналу системи.

Огляд існуючих програмних рішень показав, що наявні на ринку системи (AgroBase, FarmFacts, 1С:Сільгосп) орієнтовані переважно на великі агропідприємства, мають завищену вартість або не підтримують україномовний інтерфейс. Прості рішення на основі електронних таблиць не забезпечують необхідний рівень функціональності, захисту та зручності роботи.

На підставі виявлених проблем сформульовано 13 функціональних та 10 нефункціональних вимог до системи. Проведений порівняльний аналіз технологій розробки обґрунтував вибір технологічного стеку: C# / .NET 9, WPF, MVVM, SQLite, Entity Framework Core 9, що відповідає вимогам щодо офлайн-роботи, продуктивності та зручності розробки.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Архітектура системи

Проєктування архітектури є ключовим етапом розробки програмного продукту, що визначає структуру системи, взаємодію її компонентів та принципи розподілу відповідальності між ними.

Розроблена інформаційна система побудована на основі трирівневої архітектури, яка включає рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень доступу до даних (Data Access Layer).

Рисунок 2.1 — Трирівнева архітектура системи «ФермерОблік»

Для реалізації рівня представлення та бізнес-логіки застосовано архітектурний шаблон MVVM (Model-View-ViewModel), який є стандартом для WPF-застосунків та забезпечує чіткий розподіл відповідальності між компонентами.

Model (Модель) — представляє бізнес-сутності та логіку роботи з даними. В системі «ФермерОблік» моделями є класи: Animal, User, ProductionRecord, FeedingRecord, VaccinationRecord, TreatmentRecord, ExpenseRecord. Кожна модель відповідає таблиці в базі даних SQLite.

View (Представлення) — визначає зовнішній вигляд та структуру інтерфейсу користувача за допомогою декларативної мови XAML. Представлення не містить бізнес-логіки та взаємодіє з ViewModel виключно через механізм прив'язки даних та команд.

ViewModel (Модель представлення) — виконує роль посередника між View та Model. Містить стан інтерфейсу, команди (ICommand) для обробки дій користувача та реалізує інтерфейс INotifyPropertyChanged для сповіщення View про зміни даних. Схему шаблону MVVM ілюструє рисунок 2.2.

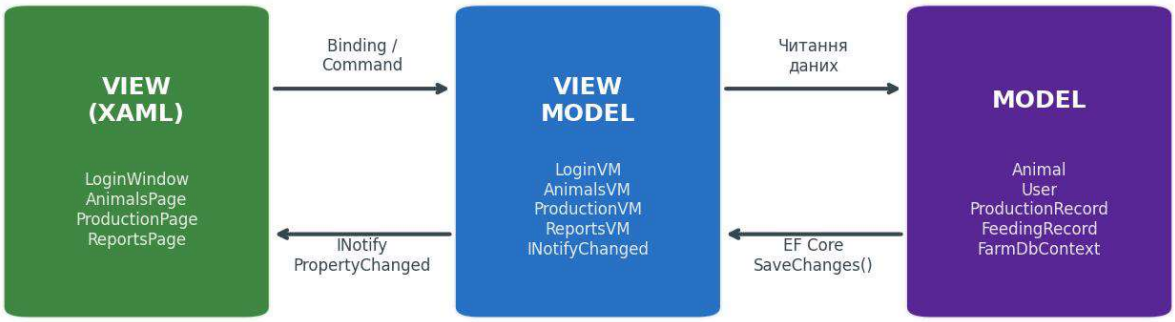

Рисунок 2.2 — Схема взаємодії компонентів за шаблоном MVVM

2.2 Проєктування бази даних

База даних є центральним компонентом системи, що забезпечує надійне зберігання та швидкий доступ до всіх облікових даних. Для реалізації обрано вбудовану реляційну СУБД SQLite, яка зберігає всі дані в єдиному файлі farm.db, розташованому поруч з виконуваним файлом у підпапці base.

Структура бази даних включає 7 таблиць, пов'язаних між собою відношеннями «один до багатьох». ER-діаграму бази даних наведено на рисунку 2.3.

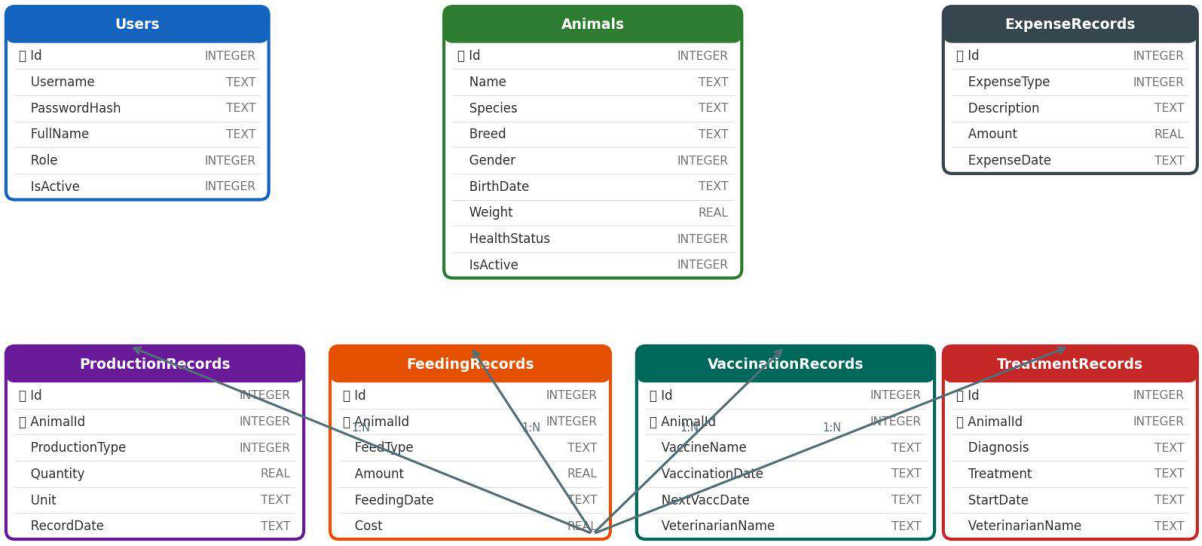

Рисунок 2.3 — ER-діаграма бази даних системи «ФермерОблік»

Таблиці та їх поля детально описані нижче.

Таблиця 2.1 — Структура таблиці Users (Користувачі)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
Username	TEXT	NOT NULL, UNIQUE	Логін (ім'я входу)
PasswordHash	TEXT	NOT NULL	BCrypt-хеш пароля
FullName	TEXT	NOT NULL	Повне ім'я користувача
Role	INTEGER	NOT NULL	Роль: 0=Адмін, 1=Фермер, 2=Ветеринар
CreatedAt	TEXT	NOT NULL	Дата створення облікового запису
IsActive	INTEGER	NOT NULL, DEF=1	Активність: 1=активний, 0=деактивований

Таблиця 2.2 — Структура таблиці Animals (Тварини)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
Name	TEXT	NOT NULL	Кличка або номер тварини
Species	TEXT	NOT NULL	Вид (Корова, Свиня тощо)
Breed	TEXT		Порода
Gender	INTEGER	NOT NULL	Стать: 0=Самець, 1=Самка
BirthDate	TEXT	NOT NULL	Дата народження (ISO 8601)

Weight	REAL		Вага у кілограмах
HealthStatus	INTEGER	NOT NULL	Статус: 0=Здорова, 1=Хвора, 2=НаЛікуванні, 3=Карантин, 4=Загинула
Notes	TEXT		Примітки
RegisteredAt	TEXT	NOT NULL	Дата реєстрації в системі
IsActive	INTEGER	NOT NULL, DEF=1	М'яке видалення: 1=активна

Таблиця 2.3 — Структура таблиці ProductionRecords (Виробництво)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
AnimalId	INTEGER	FK→Animals	Посилання на тварину
ProductionType	INTEGER	NOT NULL	Тип: 0=Молоко, 1=М'ясо, 2=Вовна, 3=Яйця, 4=Мед, 5=Інше
Quantity	REAL	NOT NULL	Кількість
Unit	TEXT	NOT NULL	Одиниця виміру (л, кг, шт.)
RecordDate	TEXT	NOT NULL	Дата запису
Notes	TEXT		Примітки

Таблиця 2.4 — Структура таблиці FeedingRecords (Годування)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
AnimalId	INTEGER	FK→Animals	Посилання на тварину
FeedType	TEXT	NOT NULL	Тип корму (Сіно, Силос тощо)
Amount	REAL	NOT NULL	Кількість корму (кг)
FeedingDate	TEXT	NOT NULL	Дата годування
Cost	REAL	NOT NULL, DEF=0	Вартість у гривнях
Notes	TEXT		Примітки

Таблиця 2.5 — Структура таблиці VaccinationRecords (Вакцинації)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
AnimalId	INTEGER	FK→Animals	Посилання на тварину

VaccineName	TEXT	NOT NULL	Назва вакцини
VaccinationDate	TEXT	NOT NULL	Дата вакцинації
NextVaccinationDate	TEXT		Дата наступної ревакцинації
VeterinarianName	TEXT		Ім'я ветеринара
Cost	REAL	NOT NULL, DEF=0	Вартість вакцинації
Notes	TEXT		Примітки

Таблиця 2.6 — Структура таблиці TreatmentRecords (Лікування)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
AnimalId	INTEGER	FK→Animals	Посилання на тварину
Diagnosis	TEXT	NOT NULL	Діагноз
Treatment	TEXT		Призначене лікування
StartDate	TEXT	NOT NULL	Дата початку лікування
EndDate	TEXT		Дата завершення лікування
VeterinarianName	TEXT		Ім'я ветеринара
Cost	REAL	NOT NULL, DEF=0	Вартість лікування
Notes	TEXT		Примітки

Таблиця 2.7 — Структура таблиці ExpenseRecords (Витрати)

Поле	Тип	Обмеження	Опис
Id	INTEGER	PK, AI	Унікальний ідентифікатор
ExpenseType	INTEGER	NOT NULL	Тип: 0=Корм, 1=Ветеринарія, 2=Обладнання, 3=Ліки, 4=Інше
Description	TEXT	NOT NULL	Опис витрати
Amount	REAL	NOT NULL	Сума у гривнях
ExpenseDate	TEXT	NOT NULL	Дата витрати
Notes	TEXT		Примітки

Усі таблиці, що містять поле AnimalId, пов'язані з таблицею Animals зовнішнім ключем (FOREIGN KEY) з каскадним видаленням залежних

записів. Для забезпечення цілісності даних у SQLite при ініціалізації підключення виконується команда PRAGMA foreign_keys = ON.

Для роботи з базою даних використовується ORM-інструмент Entity Framework Core 9, що дозволяє взаємодіяти з SQLite через об'єктно-орієнтований інтерфейс мови C# без написання SQL-запитів вручну. Схема бази даних ініціалізується автоматично при першому запуску застосунку методом EnsureCreatedAsync().

2.3 Проєктування модулів системи

Система «ФермерОблік» складається з п'яти основних функціональних модулів, кожен з яких реалізований як окремий ViewModel та відповідне XAML-представлення (сторінка або вікно).

Діаграму компонентів системи наведено на рисунку 2.4.

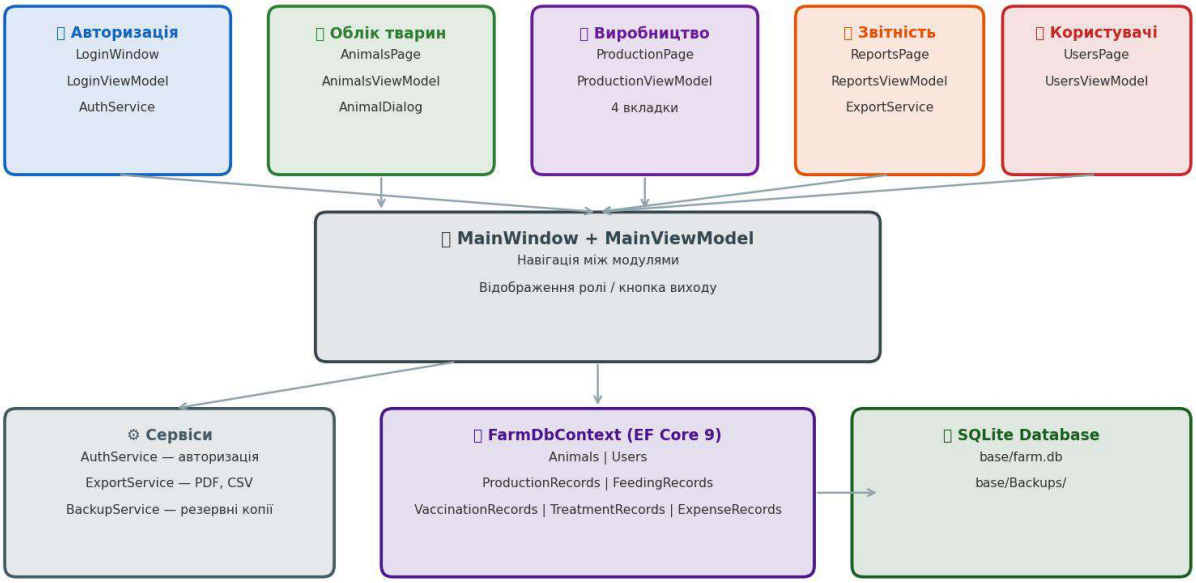

Рисунок 2.4 — Діаграма компонентів системи «ФермерОблік»

Діаграму варіантів використання (Use Case) системи наведено на рисунку 2.5. Вона відображає взаємодію трьох акторів — Адміністратора, Фермера та Ветеринара — з функціями системи.

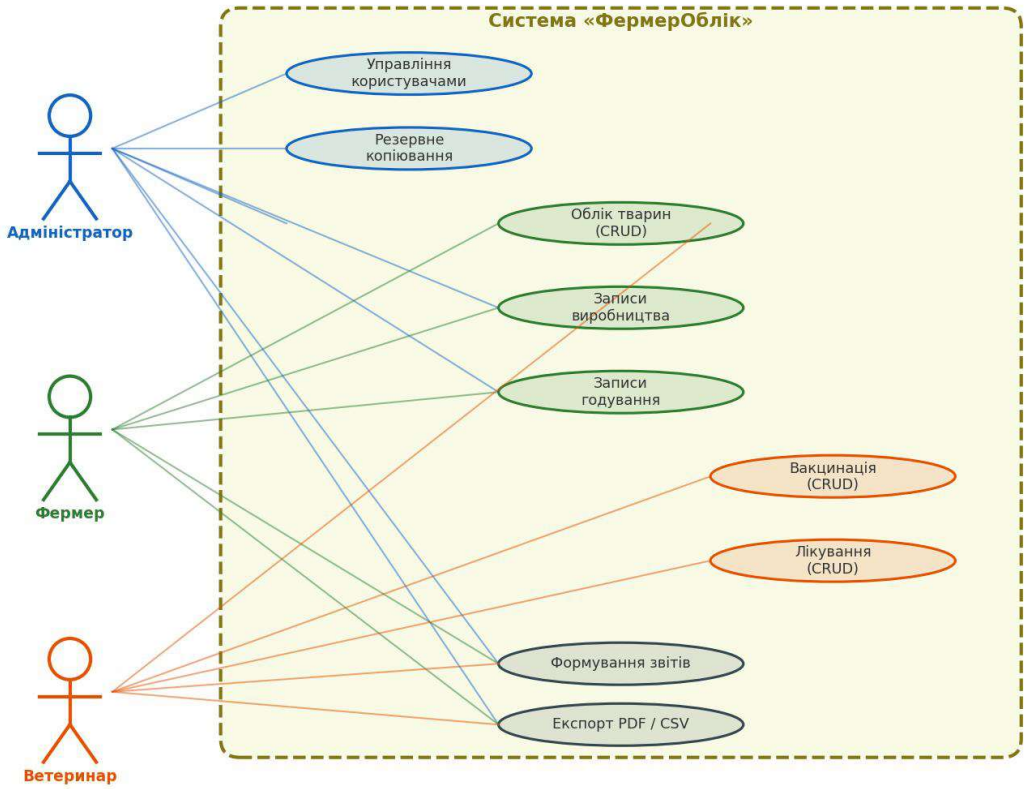

Рисунок 2.5 — Діаграма варіантів використання системи

2.4 Проєктування інтерфейсу користувача

Інтерфейс системи «ФермерОблік» спроєктований відповідно до принципів зручності використання (usability): мінімум кліків для виконання базових операцій, консистентний стиль елементів, чітка навігаційна структура.

Вікно авторизації є першим екраном, що бачить користувач. Його макет наведено на рисунку 2.6.

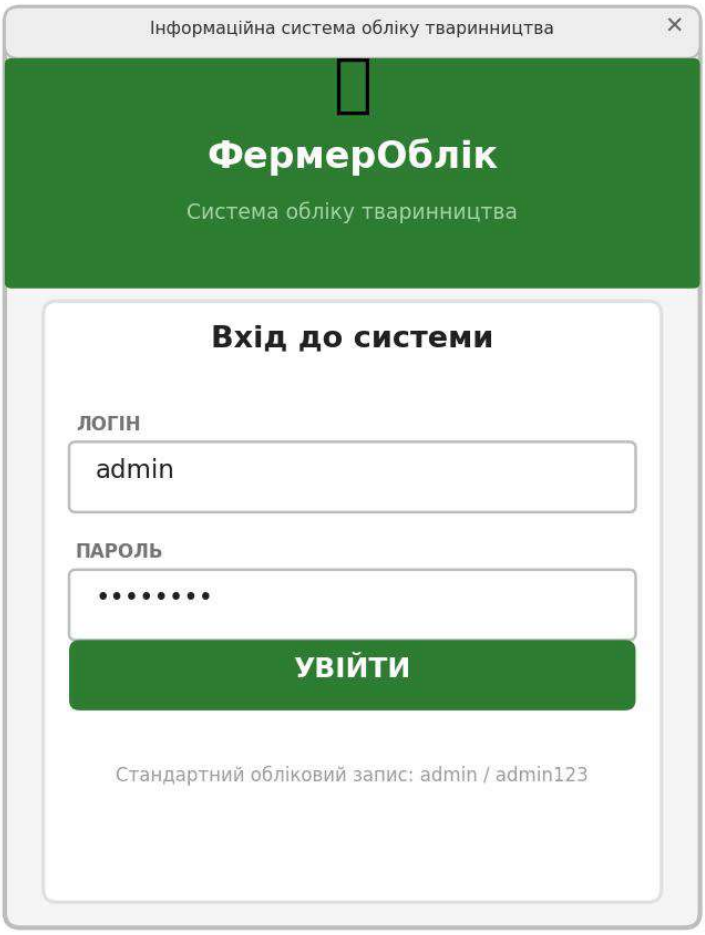

Рисунок 2.6 — Макет вікна авторизації

Головне вікно програми містить панель навігації зліва з чотирма пунктами та область відображення вмісту. Його макет наведено на рисунку 2.7.

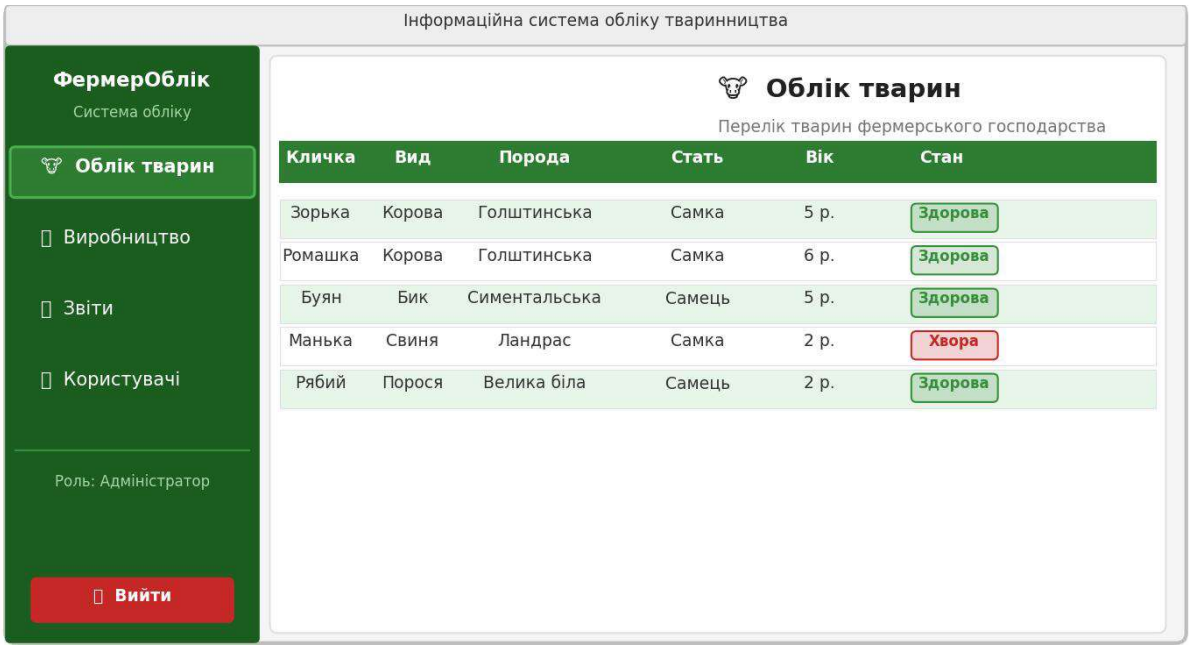

Рисунок 2.7 — Макет головного вікна програми

Модуль обліку тварин (рисунок 2.8) побудований за принципом «список + форма» — зліва розташована форма для введення даних, праворуч — таблиця з переліком тварин.

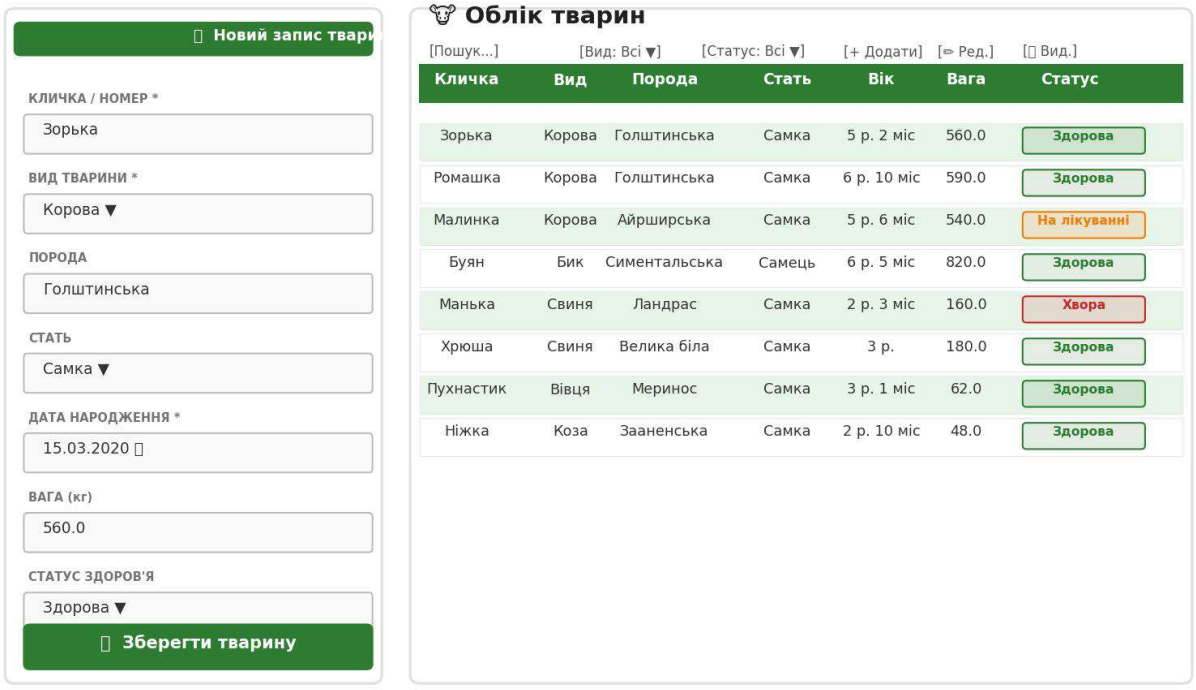

Рисунок 2.8 — Макет модуля обліку тварин

2.5 Моделювання процесів системи

Для документування ключових бізнес-процесів системи розроблено діаграми послідовності (Sequence Diagrams) та діаграму станів тварини.

Процес авторизації користувача описує взаємодію між компонентами системи при введенні облікових даних (рисунок 2.9).

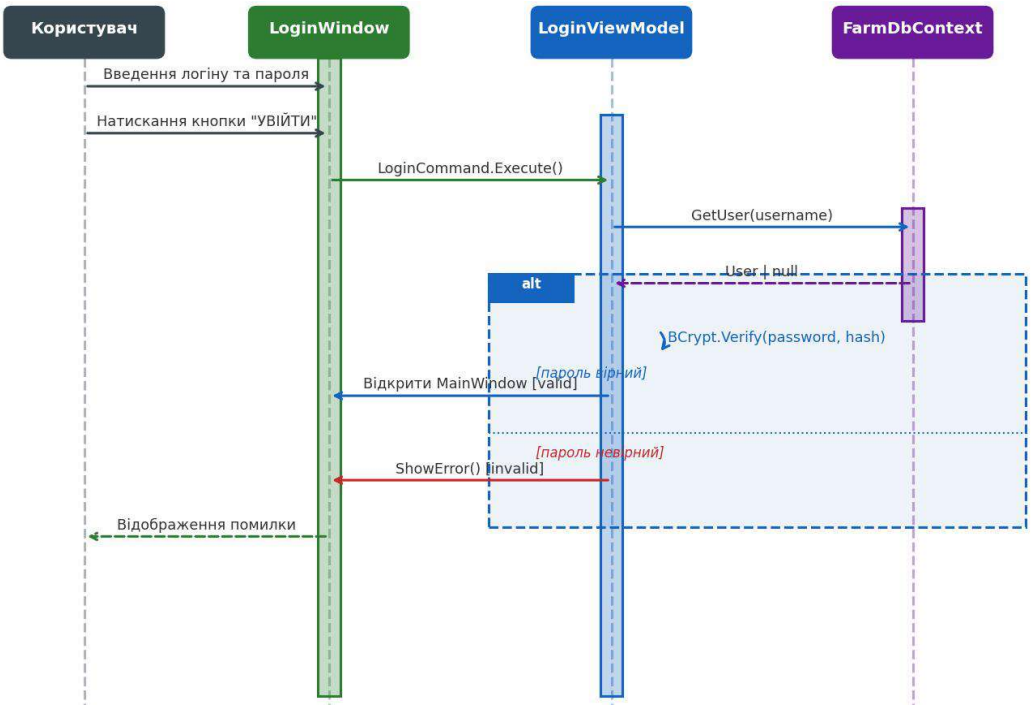

Рисунок 2.9 — Діаграма послідовності: процес авторизації

Процес додавання нової тварини до системи (рисунок 2.10) демонструє взаємодію між модулем обліку тварин та базою даних.

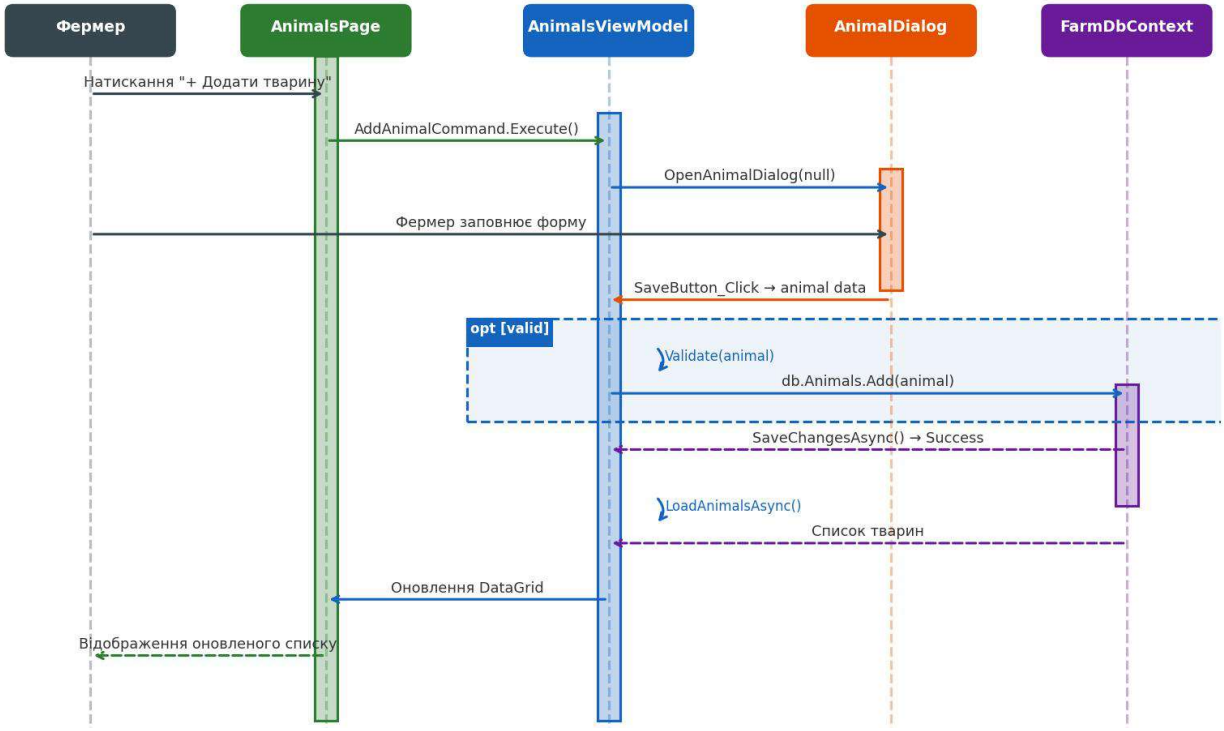

Рисунок 2.10 — Діаграма послідовності: додавання тварини

Діаграма станів тварини (рисунок 2.11) описує можливі стани, в яких може перебувати тварина в системі, та переходи між ними.

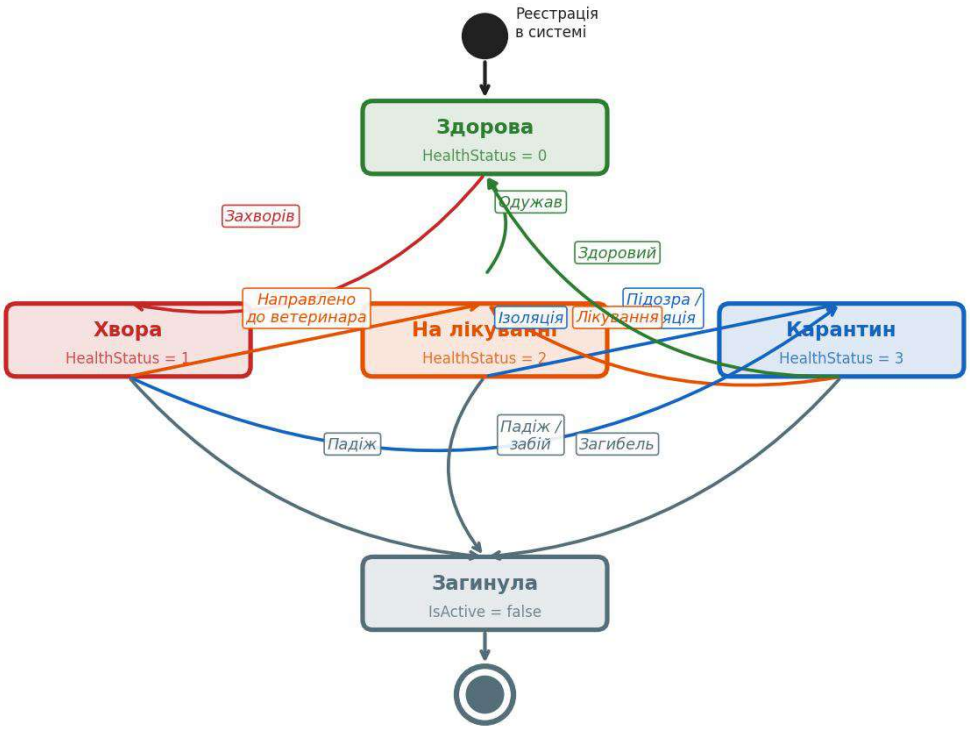

Рисунок 2.11 — Діаграма станів тварини

2.6 Захист даних та розмежування доступу

Безпека даних є критичним аспектом інформаційної системи, що оперує особистими даними персоналу та конфіденційною виробничою інформацією господарства. В системі «ФермерОблік» реалізовано декілька механізмів захисту.

Хешування паролів. Паролі користувачів ніколи не зберігаються у відкритому вигляді. При створенні облікового запису пароль хешується алгоритмом BCrypt з коефіцієнтом складності (work factor) рівним 11, що означає виконання $2^{11} = 2048$ ітерацій. Це забезпечує стійкість до атак перебору навіть у разі витоку бази даних. При авторизації введений пароль порівнюється з хешем функцією BCrypt.Verify().

Розмежування доступу реалізоване на рівні ViewModel через перевірку ролі поточного користувача (AuthService.Instance.CurrentUser.Role). Система підтримує три ролі: Адміністратор — повний доступ до всіх функцій системи; Фермер — робота з тваринами, виробничими записами та звітністю; Ветеринар — перегляд тварин, повний доступ до модулів вакцинації та лікування.

Матрицю прав доступу наведено в таблиці 2.8.

Таблиця 2.8 — Матриця прав доступу користувачів

Функція системи	Адміністратор	Фермер	Ветеринар
Управління користувачами	✓ Повний	–	–
Перегляд списку тварин	✓	✓	✓
Додавання/редагування тварин	✓	✓	–
Видалення тварин	✓	✓	–
Записи виробництва/годування	✓	✓	Перегляд
Вакцинація — перегляд	✓	✓	✓
Вакцинація — CRUD	✓	✓	✓
Лікування — CRUD	✓	✓	✓
Формування звітів	✓	✓	✓
Експорт PDF/CSV	✓	✓	✓
Резервне копіювання	✓	–	–

Резервне копіювання. Для захисту від втрати даних у системі реалізовано автоматичне резервне копіювання бази даних за командою користувача. Резервні копії зберігаються в підпапці base/Backups/ у форматі farm_backup_РРРР-ММ-ДД_ГГ-хх-сс.db. Система автоматично видаляє надлишкові резервні копії, зберігаючи не більше 10 останніх, що дозволяє контролювати дисковий простір.

Цілісність даних забезпечується механізмом зовнішніх ключів SQLite (PRAGMA foreign_keys = ON), транзакційністю операцій Entity Framework Core та реалізацією патерну «м'якого видалення» (soft delete) для тварин — замість фізичного видалення запис позначається як неактивний (IsActive = false), що зберігає всю пов'язану облікову інформацію.

Висновки до розділу 2

У другому розділі виконано комплексне проєктування інформаційної системи обліку тваринництва. Розроблено трирівневу архітектуру системи з використанням шаблону MVVM, що забезпечує чіткий розподіл відповідальності між компонентами та зручність модифікації системи.

Спроєктовано реляційну базу даних, що складається з семи таблиць: Users, Animals, ProductionRecords, FeedingRecords, VaccinationRecords, TreatmentRecords та ExpenseRecords. Для кожної таблиці визначено склад полів, типи даних та обмеження цілісності. Побудовано ER-діаграму, що ілюструє зв'язки між таблицями.

Описано функціональні модулі системи та їх взаємодію за допомогою діаграми компонентів. Побудовано діаграму варіантів використання, яка визначає права доступу трьох ролей користувачів. Розроблено макети ключових екранів інтерфейсу.

Для документування динаміки системи побудовано діаграми послідовності для процесів авторизації та додавання тварини, а також діаграму станів тварини з п'ятьма можливими станами та переходами між ними. Сформовано матрицю прав доступу, що регламентує доступ до 12 функцій системи для трьох ролей користувачів. Описано механізми захисту: BCrypt-хешування паролів, рольове розмежування доступу, резервне копіювання та м'яке видалення.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Програмне середовище розробки

Реалізація інформаційної системи «ФермерОблік» виконувалась у середовищі розробки Microsoft Visual Studio 2022 Community Edition. Для управління версіями коду використовувалась система контролю версій Git. Структура проєкту організована відповідно до шаблону WPF Application для .NET 9.

Основний файл проєкту FarmApp.csproj визначає цільову платформу, тип виходу та залежності від зовнішніх бібліотек NuGet. Перелік усіх використаних технологій та бібліотек наведено в таблиці 3.1.

Таблиця 3.1 — Технологічний стек системи «ФермерОблік»

Технологія / Бібліотека	Версія	Призначення
.NET	9.0	Платформа виконання застосунку
C#	13.0	Основна мова програмування
WPF (Windows Presentation Foundation)	9.0	Фреймворк для побудови UI
XAML	—	Декларативна мова опису інтерфейсу
Entity Framework Core	9.0.0	ORM для роботи з базою даних
Microsoft.EntityFrameworkCore.Sqlite	9.0.0	Провайдер SQLite для EF Core
BCrypt.Net-Next	4.0.3	Хешування паролів (BCrypt)
CsvHelper	33.0.1	Генерація CSV-файлів
PdfSharpCore	1.3.65	Генерація PDF-звітів
LiveCharts.Wpf	0.9.7	Бібліотека діаграм
Visual Studio 2022	17.x	Середовище розробки (IDE)
DB Browser for SQLite	3.13	Інструмент перегляду БД

Проєкт організований за структурою MVVM з розподілом файлів по папках: Database (моделі та контекст БД), ViewModels (бізнес-логіка), Views (XAML-сторінки та вікна), Services (допоміжні сервіси), Converters (конвертери для прив'язки даних). Файл бази даних farm.db розміщується у підпапці base поряд з виконуваним файлом застосунку.

3.2 Реалізація бази даних

База даних реалізована засобами Entity Framework Core 9 з провайдером SQLite. Усі моделі даних описані у вигляді C#-класів у файлі Database/Models/Models.cs. Кожен клас відповідає одній таблиці бази даних.

Лістинг 3.1 демонструє опис основної моделі Animal та перечислення HealthStatus, що відображає можливі стани здоров'я тварини.

```
// Перечислення стану здоров'я тварини
public enum HealthStatus
{
    Здорова = 0,
    Хвора = 1,
    НаЛікуванні = 2,
    Карантин = 3,
    Загинула = 4
}
// Модель тварини
public class Animal
{
    public int Id { get; set; }
    public string Name { get; set; } = string.Empty;
    public string Species { get; set; } = string.Empty;
    public string Breed { get; set; } = string.Empty;
    public AnimalGender Gender { get; set; }
    public DateTime BirthDate { get; set; }
    public double Weight { get; set; }
    public HealthStatus HealthStatus { get; set; }
    public string Notes { get; set; } = string.Empty;
    public DateTime RegisteredAt { get; set; } = DateTime.Today;
    public bool IsActive { get; set; } = true;
    // Навігаційні властивості (дочірні записи)
    public List<ProductionRecord> ProductionRecords { get; set; } = new();
    public List<FeedingRecord> FeedingRecords { get; set; } = new();
    public List<VaccinationRecord> VaccinationRecords { get; set; } = new();
    public List<TreatmentRecord> TreatmentRecords { get; set; } = new();
}
```

Лістинг 3.1 — Модель даних Animal та перечислення HealthStatus

Контекст бази даних FarmDbContext успадковує клас DbContext бібліотеки Entity Framework Core та визначає DbSet-властивості для кожної таблиці. Лістинг 3.2 показує ключові фрагменти контексту.

```
public class FarmDbContext : DbContext
{
    // Шлях до БД: поряд з exe у папці 'base'
    private static readonly string DbPath = GetDatabasePath();
    private static string GetDatabasePath()
    {
        var exeDir = Path.GetDirectoryName(
            Assembly.GetExecutingAssembly().Location) ?? AppDomain
                .CurrentDomain.BaseDirectory;
        var dbDir = Path.Combine(exeDir, "base");
        Directory.CreateDirectory(dbDir);
        return Path.Combine(dbDir, "farm.db");
    }
    // DbSet для кожної таблиці
    public DbSet<User> Users => Set<User>();
    public DbSet<Animal> Animals => Set<Animal>();
    public DbSet<ProductionRecord> ProductionRecords => Set<ProductionRecord>();
    public DbSet<FeedingRecord> FeedingRecords => Set<FeedingRecord>();
    public DbSet<VaccinationRecord> VaccinationRecords => Set<VaccinationRecord>();
    public DbSet<TreatmentRecord> TreatmentRecords => Set<TreatmentRecord>();
    public DbSet<ExpenseRecord> ExpenseRecords => Set<ExpenseRecord>();
    protected override void OnConfiguring(DbContextOptionsBuilder options)
        => options.UseSqlite($"Data Source={DbPath}");
}
```

Лістинг 3.2 — Контекст бази даних FarmDbContext (фрагмент)

Ініціалізація схеми бази даних виконується автоматично при першому запуску через виклик db.Database.EnsureCreatedAsync(). Цей метод перевіряє наявність бази даних та, якщо вона відсутня, створює всі таблиці відповідно до описаних моделей.

3.3 Реалізація модуля авторизації

Модуль авторизації складається з вікна LoginWindow та відповідного LoginViewModel. При натисканні кнопки «УВІЙТИ» виконується команда LoginCommand, яка перевіряє введені облікові дані через AuthService.

Для захисту паролів застосовується бібліотека BCrypt.Net з коефіцієнтом складності (work factor) 11. Лістинг 3.3 демонструє реалізацію хешування та верифікації паролів.

```
public static class AuthService
{
    public static Instance AuthService { get; } = new();
    public User? CurrentUser { get; private set; }
    public bool IsAdmin => CurrentUser?.Role == UserRole.Адміністратор;
    // Хешування пароля при створенні користувача
    public static string HashPassword(string password)
        => BCrypt.Net.BCrypt.HashPassword(password, workFactor: 11);
    // Верифікація пароля при вході
    public static bool VerifyPassword(string password, string hash)
        => BCrypt.Net.BCrypt.Verify(password, hash);
    public async Task<(bool Success, string Message)>
        LoginAsync(string username, string password){
        await using var db = new FarmDbContext();
        var user = await db.Users
            .FirstOrDefaultAsync(u => u.Username == username && u.IsActive);
        if (user == null) return (false, "Користувача не знайдено");
        if (!VerifyPassword(password, user.PasswordHash))
            return (false, "Невірний пароль");
        CurrentUser = user;
        return (true, string.Empty);
    }
}
```

Лістинг 3.3 — Реалізація сервісу авторизації з BCrypt-хешуванням

Вікно авторизації реалізоване у файлі Views/LoginWindow.xaml. Його фактичний вигляд наведено на рисунку 3.1.

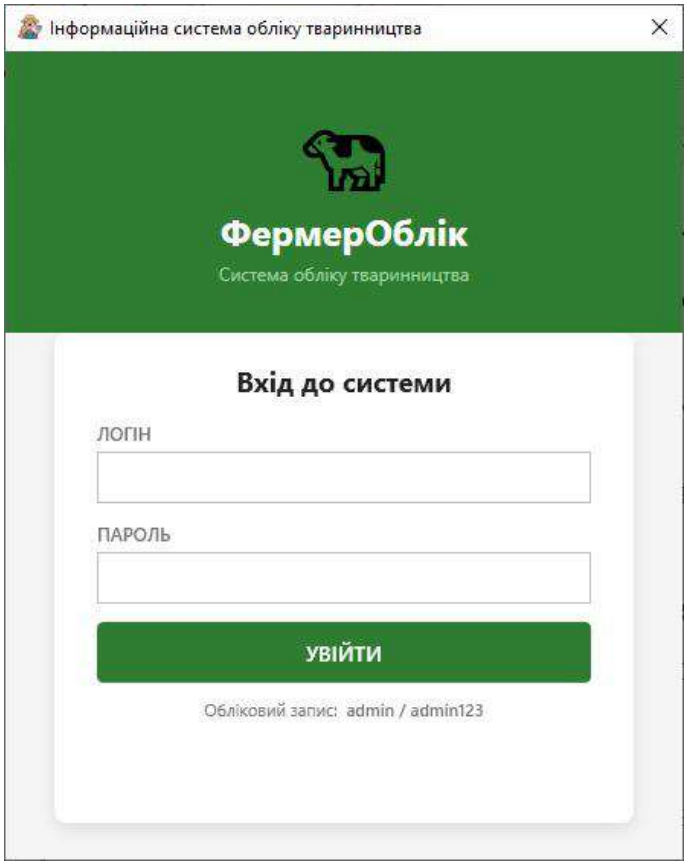

Рисунок 3.1 — Вікно авторизації системи «ФермерОблік»

При невірному введенні даних під полем пароля з'являється повідомлення про помилку з використанням конвертера StringNotEmptyToVisibilityConverter, який перетворює непорожній рядок ErrorMessage у видимість блоку помилки.

3.4 Реалізація модуля обліку тварин

Модуль обліку тварин є центральним функціональним блоком системи. Він реалізований у класах AnimalsViewModel та AnimalsPage.xaml і забезпечує повний набір CRUD-операцій для роботи з поголів'ям.

Особливістю реалізації пошуку є використання фільтрації на стороні клієнта (client-side filtering) замість SQL-запитів. Це обумовлено обмеженнями провайдера SQLite в Entity Framework Core щодо трансляції методу ToLower() для кирилічних рядків. Лістинг 3.4 демонструє реалізацію методу завантаження та фільтрації тварин.

```
public async Task LoadAnimalsAsync(){
    await using var db = new FarmDbContext();
    // Базовий запит: лише активні тварини
    var query = db.Animals.Where(a => a.IsActive).AsQueryable();
    // Фільтр за видом (на стороні сервера)
    if (!string.IsNullOrEmpty(FilterSpecies) && FilterSpecies != "Всі")
        query = query.Where(a => a.Species == FilterSpecies);
    // Фільтр за статусом здоров'я
    if (!string.IsNullOrEmpty(FilterStatus) && FilterStatus != "Всі")
        if (Enum.TryParse<HealthStatus>(FilterStatus, out var status))
            query = query.Where(a => a.HealthStatus == status);
    // Завантаження у пам'ять
    var list = await query.OrderBy(a => a.Species)
                          .ThenBy(a => a.Name).ToListAsync();
    // Текстовий пошук на стороні клієнта (для коректної роботи з кирилицею)
    if (!string.IsNullOrWhiteSpace(SearchText)){
        var s = SearchText.Trim().ToLowerInvariant();
        list = list.Where(a =>
            a.Name.ToLowerInvariant().Contains(s) ||
            a.Breed.ToLowerInvariant().Contains(s) ||
            a.Species.ToLowerInvariant().Contains(s)
        ).ToList();
    }
    Animals = new ObservableCollection<Animal>(list);
    TotalAnimals  = list.Count;
    HealthyAnimals = list.Count(a => a.HealthStatus == HealthStatus.Здорова);
    SickAnimals   = list.Count(a => a.HealthStatus == HealthStatus.Хвора
                                 || a.HealthStatus == HealthStatus.НаЛікуванні);
}
```

Лістинг 3.4 — Метод завантаження та фільтрації тварин

Для редагування тварини використовується контекстне меню DataGrid та кнопки «Редагувати» і «Видалити». Оскільки ContextMenu у WPF не є частиною візуального дерева, для прив'язки команд використовується патерн PlacementTarget.Tag: кожен рядок DataGrid зберігає у властивості Tag посилання на ViewModel, а команди у ContextMenu прив'язуються через PlacementTarget.Tag.EditAnimalCommand та PlacementTarget.Tag.DeleteAnimalCommand.

Видалення тварини реалізовано за принципом «м'якого видалення» (soft delete): замість фізичного видалення з бази даних для запису встановлюється IsActive = false. Це дозволяє зберегти всі пов'язані облікові записи (виробництво, вакцинації, лікування) в цілісності.

Фактичний вигляд модуля обліку тварин наведено на рисунку 3.2, а діалог додавання/редагування — на рисунку 3.3.

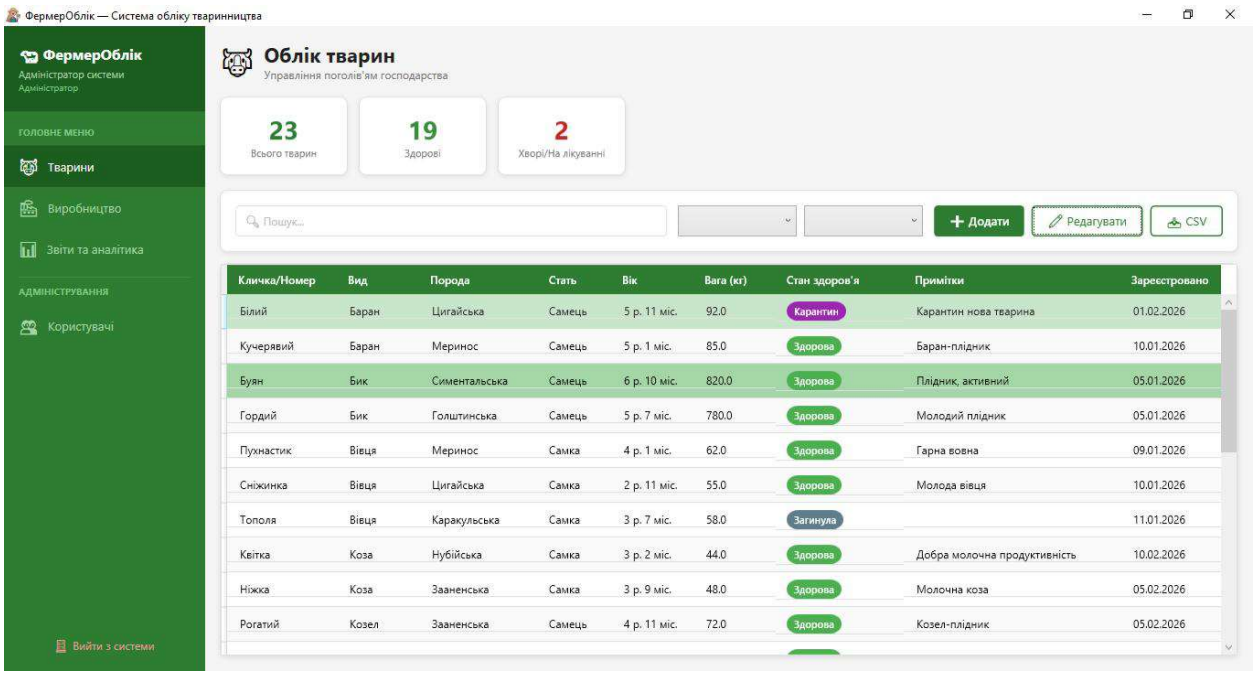

Рисунок 3.2 — Модуль обліку тварин

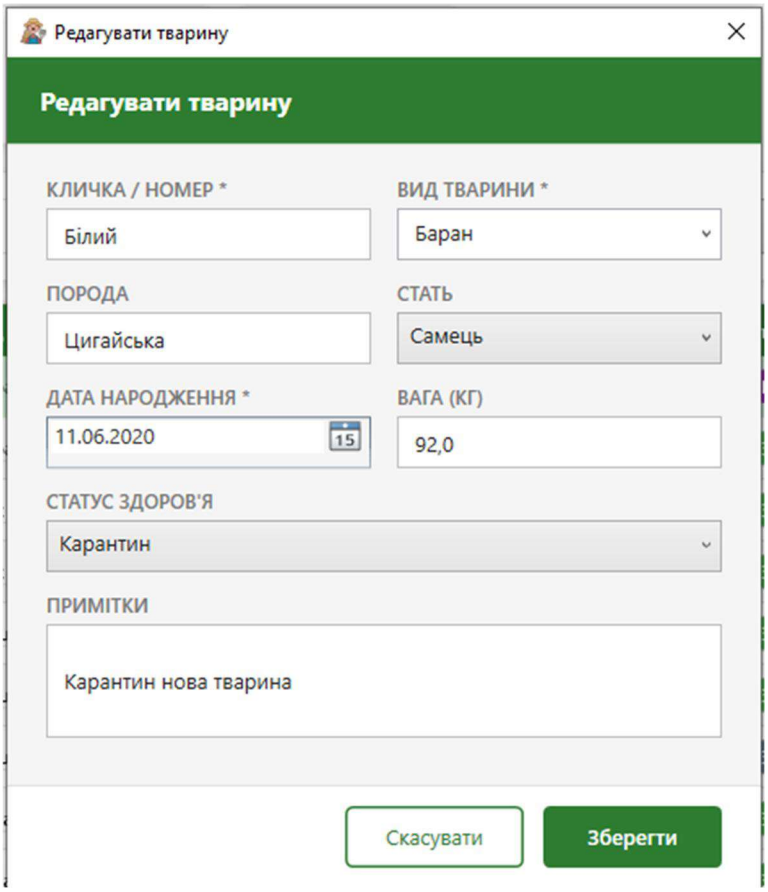

Рисунок 3.3 — Діалог додавання/редагування тварини

3.5 Реалізація модуля виробництва та догляду

Модуль виробництва та догляду реалізований у вигляді сторінки ProductionPage з чотирма вкладками: «Виробництво», «Годування», «Вакцинація» та «Лікування». Кожна вкладка дозволяє виконувати повний набір CRUD-операцій.

Архітектурно кожна вкладка містить панель форми зліва та таблицю записів праворуч. При виборі рядка в таблиці форма автоматично заповнюється відповідними даними, а заголовок форми змінюється з «Новий запис» на «Редагування запису». Це забезпечується через прив'язку властивостей SelectedProdRecord, SelectedFeedRecord, SelectedVaccRecord та SelectedTreatRecord до DataGrid.SelectedItem з режимом TwoWay.

Особливістю реалізації вкладок «Вакцинація» та «Лікування» є поле вибору ветеринара у вигляді ComboBox, що завантажує список активних користувачів з роллю Ветеринар з таблиці Users. Це забезпечує цілісність

даних та унеможливлює введення помилкових імен. Фактичний вигляд вкладок наведено на рисунках 3.4–3.7.

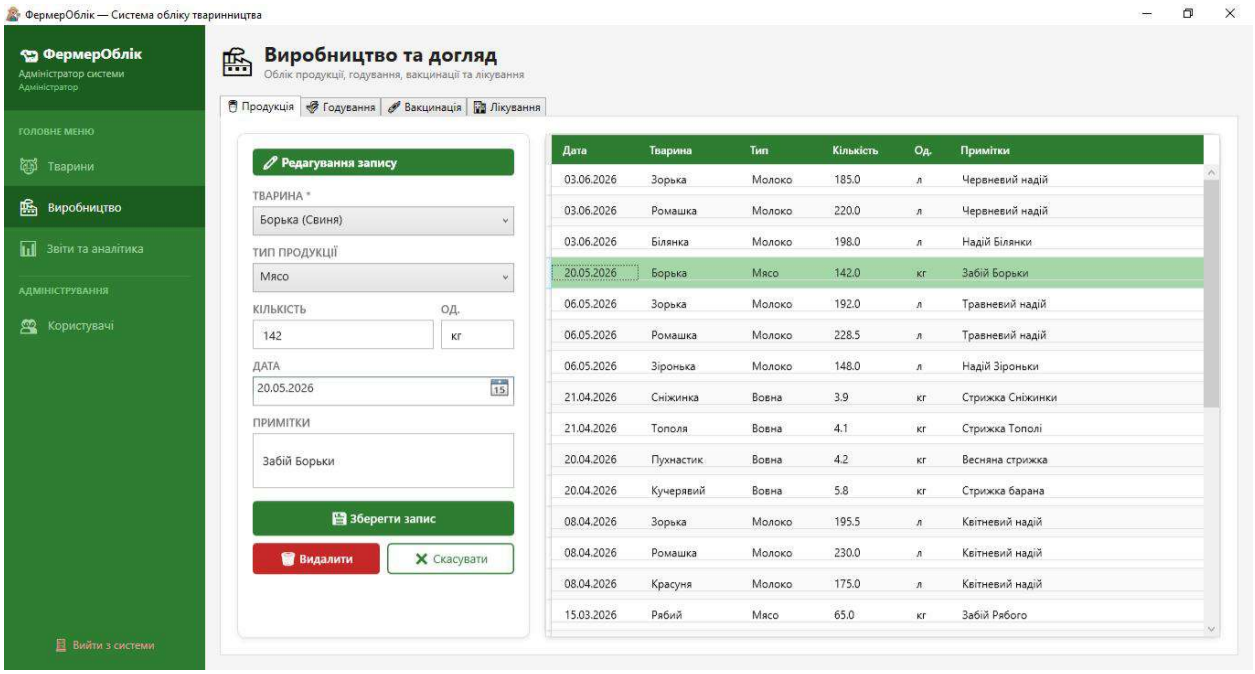

Рисунок 3.4 — Вкладка «Продукція» модуля виробництва

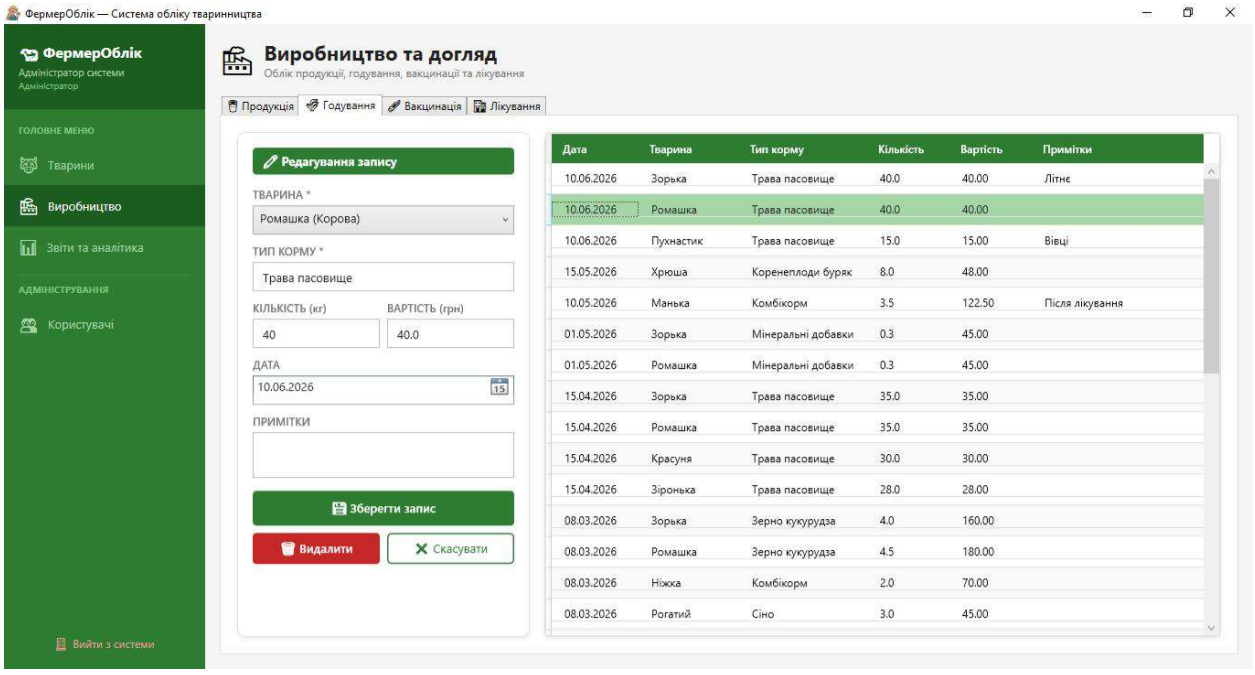

Рисунок 3.5 — Вкладка «Годування»

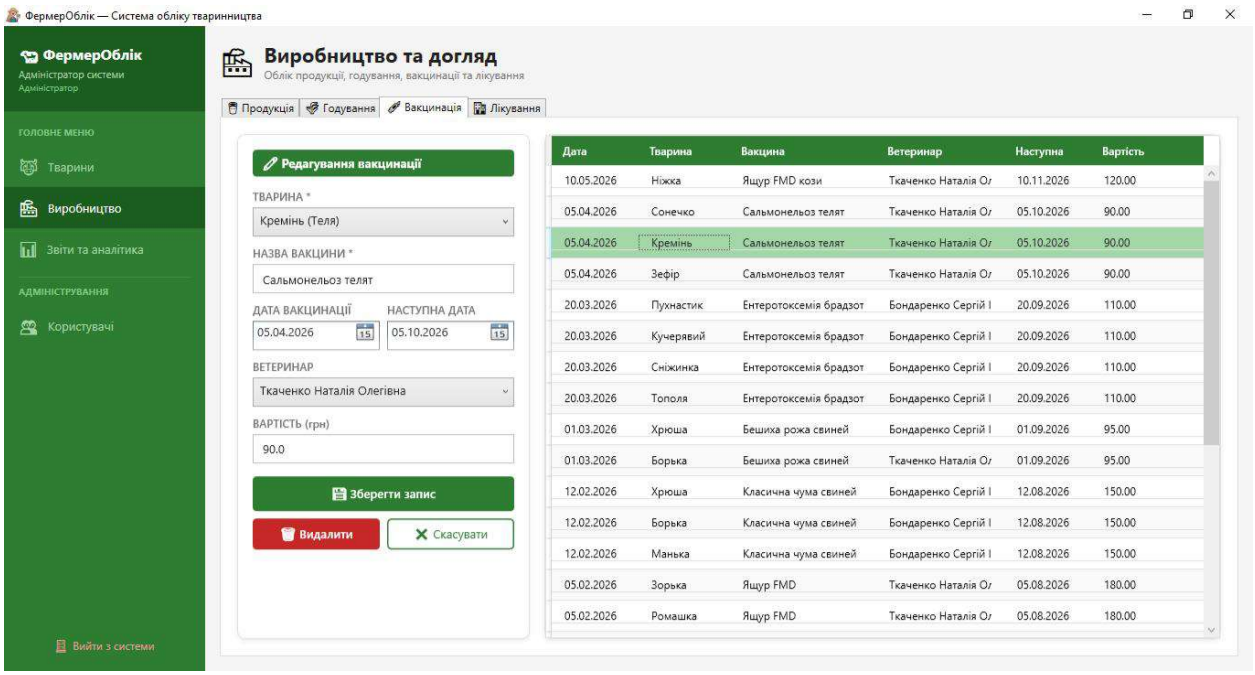

Рисунок 3.6 — Вкладка «Вакцинація» з ComboBox ветеринарів

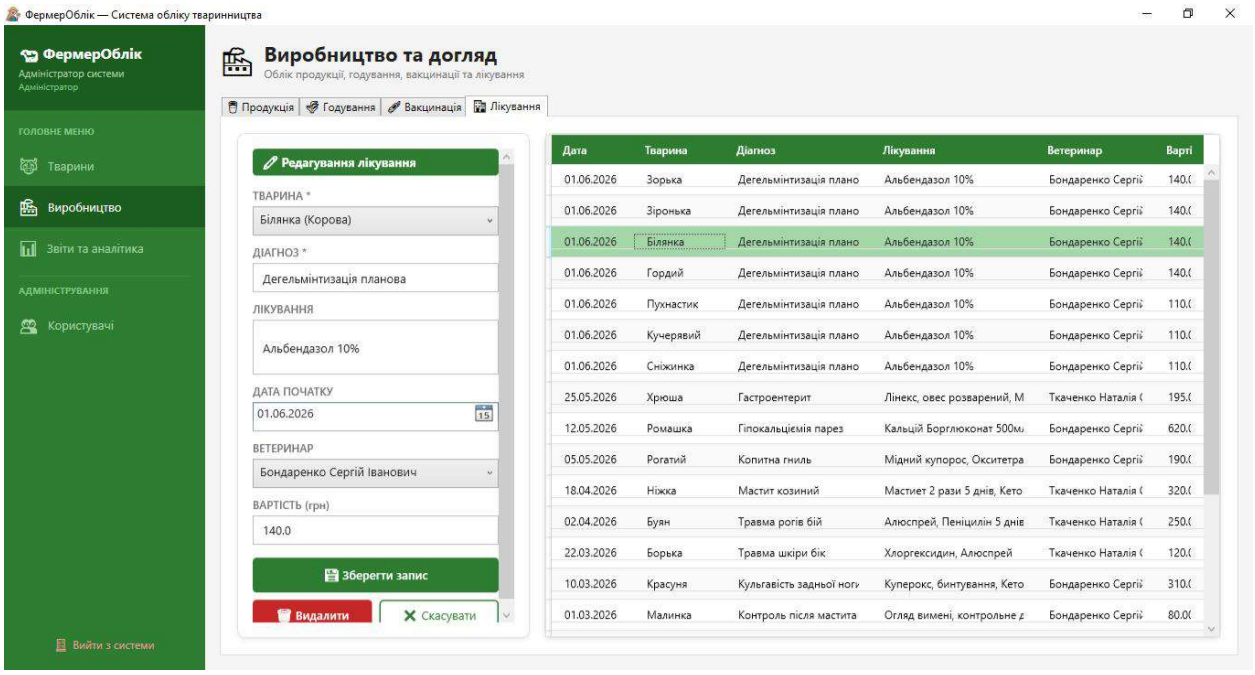

Рисунок 3.7 — Вкладка «Лікування»

3.6 Реалізація модуля звітності та аналітики

Модуль звітності реалізований у класі ReportsViewModel та сторінці ReportsPage.xaml. Він дозволяє формувати зведені звіти за довільний часовий діапазон, що задається через два поля DatePicker: «Дата з» та «Дата до».

Після натискання кнопки «Сформувати звіт» виконується асинхронний метод GenerateReportAsync(), який збирає дані з усіх семи таблиць БД та

формує колекцію рядків звіту. Звіт містить: загальну кількість та стан здоров'я тварин; розподіл поголів'я за видами; обсяги виробленої продукції за кожним типом; витрати на корми, вакцинацію та лікування.

Права частина сторінки містить дві діаграми — «Структура поголів'я» та «Витрати за період». Діаграми реалізовані як ItemsControl з шаблоном DataTemplate, де ширина кожної смуги (BarWidth) розраховується нормалізацією значення відносно максимального: BarWidth = Value / MaxValue * 180 пікселів. Фактичний вигляд модуля звітів наведено на рисунку 3.8.

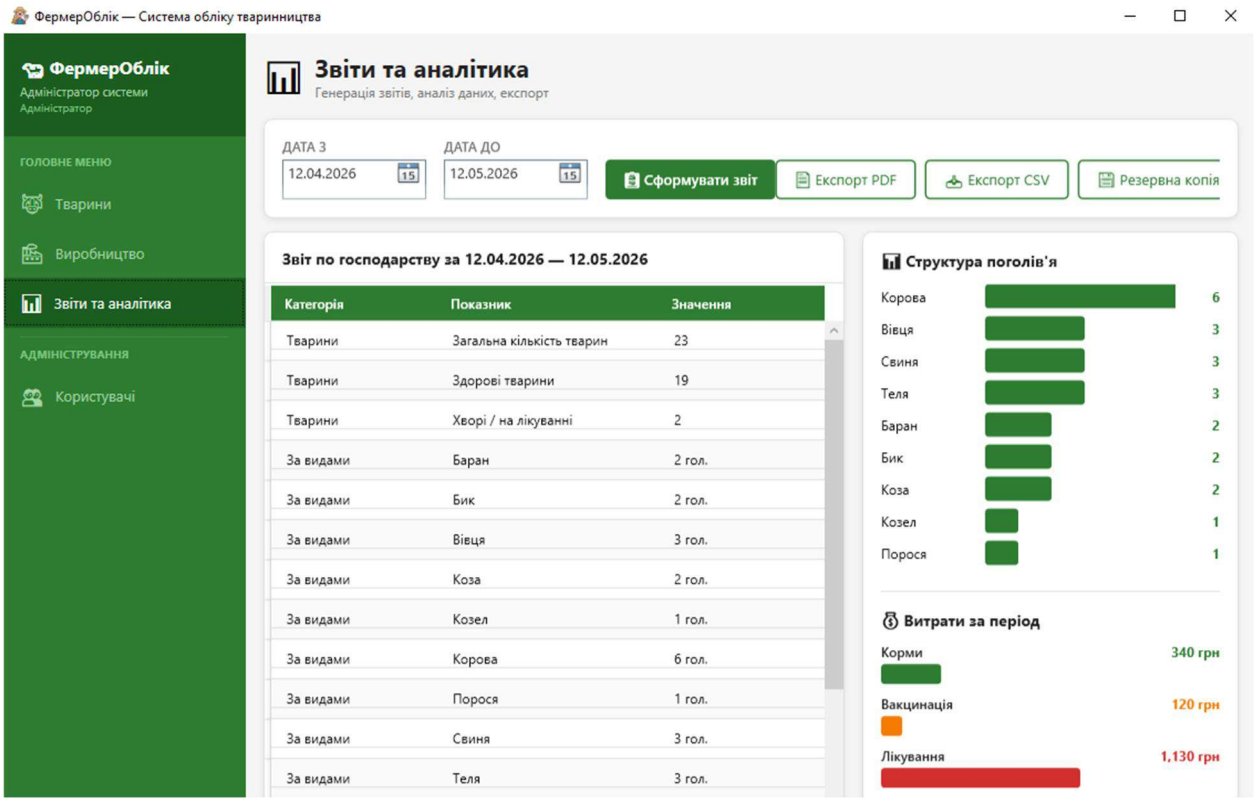

Рисунок 3.8 — Модуль звітності з діаграмами

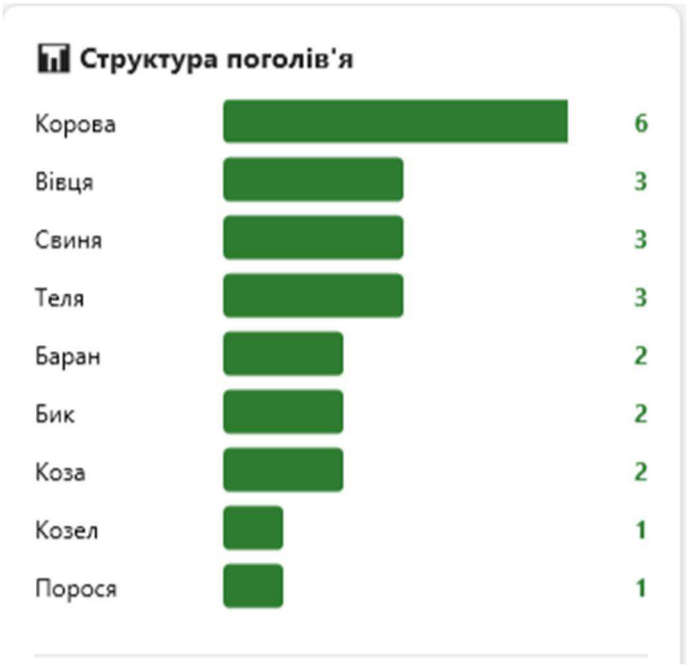

Рисунок 3.9 — Діаграма структури поголів'я

Рисунок 3.10 — Діаграма витрат за період

3.7 Реалізація експорту даних

Система підтримує два формати експорту звітів: PDF та CSV. Реалізацію обох форматів забезпечує клас ExportService у папці Services.

Експорт у формат CSV реалізований за допомогою бібліотеки CsvHelper з використанням роздільника «;» (крапка з комою), що забезпечує коректне відкриття файлів у Microsoft Excel з українською локалізацією. Лістинг 3.5 демонструє фрагмент методу експорту тварин у CSV.

```
public static async Task<(bool Success, string Message)>
    ExportAnimalsToCsvAsync(IEnumerable<Animal> animals, string filePath)       {
    try         {
        // UTF-8 кодування для коректного відображення кирилиці
        await using var writer = new StreamWriter(filePath, false, Encoding.UTF8);
        // Роздільник ';' для сумісності з Excel (українська локалізація)
        await using var csv = new CsvWriter(writer,
            new CsvConfiguration(CultureInfo.InvariantCulture)
            {
                Delimiter = ";"
            });
        // Запис заголовків та рядків
        csv.WriteHeader<AnimalCsvRecord>();
        await csv.NextRecordAsync();
        foreach (var animal in animals)
        {
            csv.WriteRecord(new AnimalCsvRecord(animal));
            await csv.NextRecordAsync();
        }
        return (true, $"Збережено: {filePath}");
    }
    catch (Exception ex) { return (false, ex.Message); }
}
```

Лістинг 3.5 — Метод експорту тварин у CSV-формат

Генерація PDF-звіту реалізована за допомогою бібліотеки PdfSharpCore. Лістинг 3.6 демонструє ключовий фрагмент методу формування PDF-документа.

```
public static (bool Success, string Message) GeneratePdfReport(
    string filePath, string title,
    List<string> headers, List<List<string>> rows)       {
    // Реєстрація кодування для кирилиці
    Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
    using var document = new PdfDocument();
    var page = document.AddPage();
    var gfx = XGraphics.FromPdfPage(page);
    // Шрифти
    var fontTitle  = new XFont("Arial", 14, XFontStyleEx.Bold);
    var fontHeader = new XFont("Arial", 10, XFontStyleEx.Bold);
    var fontBody   = new XFont("Arial",  9, XFontStyleEx.Regular);
    double y = 40;
    // Заголовок звіту
    gfx.DrawString(title, fontTitle, XBrushes.Black,
        new XRect(0, y, page.Width, 20), XStringFormats.TopCenter);
    y += 30;
    // Таблиця заголовків
    double colW = (page.Width - 80) / headers.Count;
    foreach (var (h, i) in headers.Select((h,i)=>(h,i)))              {
    y += 20;
    // Рядки даних
    foreach (var row in rows){
    document.Save(filePath);
    return (true, $"PDF збережено: {filePath}");
}
```

Лістинг 3.6 — Генерація PDF-звіту (фрагмент ExportService)

3.8 Тестування системи

Тестування програмного продукту виконувалось методом функціонального тестування — перевірки відповідності реалізованих функцій системи сформульованим вимогам. Для кожного модуля розроблено набір тест-кейсів, що охоплюють позитивні сценарії (коректні вхідні дані) та негативні сценарії (некоректні або граничні значення).

Таблиця 3.2 містить тест-кейси для модуля авторизації.

Таблиця 3.2 — Тест-кейси модуля авторизації

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Вхід з коректними даними	Логін: admin, пароль: admin123	Відкриття MainWindow	ПРОЙДЕНО: вікно відкрилось
2	Вхід з неправильним паролем	Логін: admin, пароль: wrongpass	Помилка 'Невірний пароль'	ПРОЙДЕНО: повідомлення відображено
3	Вхід з неіснуючим логіном	Логін: xyz, пароль: any	Помилка 'Користувача не знайдено'	ПРОЙДЕНО
4	Порожнє поле логіну	Логін: '', пароль: admin123	Помилка 'Введіть логін'	ПРОЙДЕНО
5	Порожнє поле паролю	Логін: admin, пароль: ''	Помилка 'Введіть пароль'	ПРОЙДЕНО
6	Вхід деактивованого користувача	IsActive=false	Помилка 'Акаунт деактивовано'	ПРОЙДЕНО

Таблиця 3.3 містить тест-кейси для модуля обліку тварин.

Таблиця 3.3 — Тест-кейси модуля обліку тварин

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Додавання нової тварини	Всі поля заповнені коректно	Тварина додана до списку	ПРОЙДЕНО
2	Порожнє поле 'Кличка'	Name=''	Помилка валідації	ПРОЙДЕНО
3	Майбутня дата народження	BirthDate > Today	Помилка 'Дата не може бути в майбутньому'	ПРОЙДЕНО

4	Редагування тварини	Зміна ваги та стану	Дані оновлено в БД	ПРОЙДЕНО
5	Видалення тварини	Підтвердження видалення	IsActive=false, зі списку зникла	ПРОЙДЕНО
6	Пошук за кличкою	Пошуковий рядок: 'Зор'	Список містить лише 'Зорька'	ПРОЙДЕНО
7	Фільтр за видом	Фільтр: 'Корова'	Показано лише корів	ПРОЙДЕНО

Таблиця 3.4 містить тест-кейси для модуля виробництва та догляду.

Таблиця 3.4 — Тест-кейси модуля виробництва та догляду

№	Опис тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	Додавання запису надою	Тварина: Зорька, 185 л	Запис збережено, відображено в таблиці	ПРОЙДЕНО
2	Кількість 0 або від'ємна	Quantity = 0	Помилка 'Кількість > 0'	ПРОЙДЕНО
3	Тварина не обрана	SelectedAnimal = null	Помилка 'Оберіть тварину'	ПРОЙДЕНО
4	Редагування запису	Клік на рядок + зміна даних	Форма заповнена, дані збережено	ПРОЙДЕНО
5	Видалення запису	Підтвердження	Запис видалено з БД	ПРОЙДЕНО
6	Вибір ветеринара зі списку	ComboBox ветеринарів	Список містить лише Role=Ветеринар	ПРОЙДЕНО

Зведені результати тестування всіх модулів системи наведено в таблиці 3.5.

Таблиця 3.5 — Зведені результати функціонального тестування

Модуль	Тестів	Пройдено	Провалено	% успіху
Авторизація	6	6	0	100 %
Облік тварин	7	7	0	100 %
Виробництво та догляд	6	6	0	100 %
Звітність та експорт	5	5	0	100 %
Управління користувачами	4	4	0	100 %
Разом	28	28	0	100 %

Результати тестування свідчать про те, що всі 28 тест-кейсів успішно пройдені. Система коректно обробляє як позитивні сценарії (збереження, оновлення, видалення даних), так і негативні (введення некоректних даних, порожні поля, майбутні дати). Приклад відображення помилки валідації наведено на рисунку 3.11.

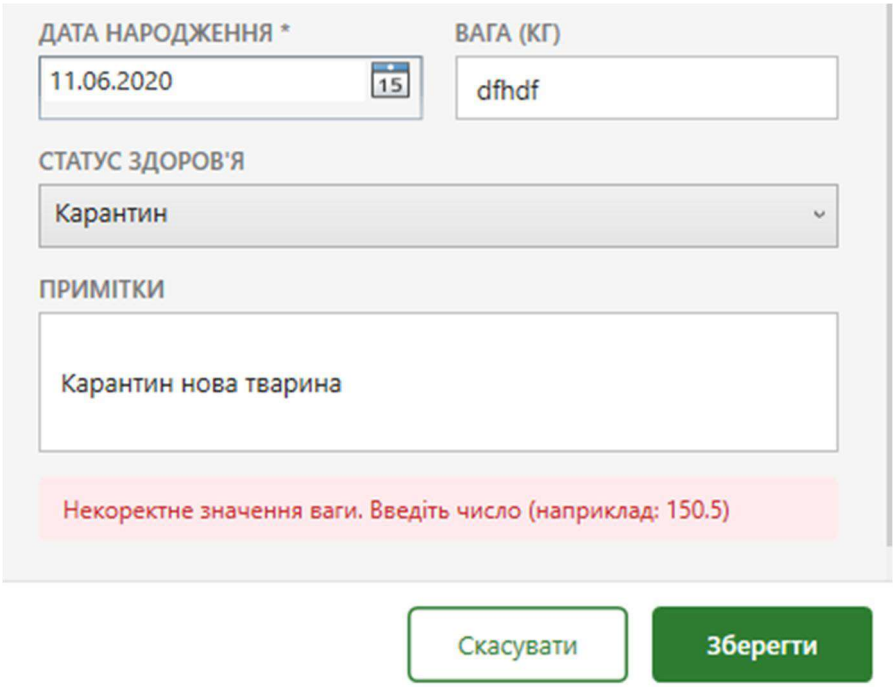

Рисунок 3.11 — Відображення помилки валідації у формі

3.9 Інструкція користувача

Для встановлення та запуску системи «ФермерОблік» необхідно виконати такі кроки:

1) Розпакувати архів FarmApp.zip у будь-яку папку на диску.

2) Переконатись у наявності середовища виконання .NET 9 Runtime для Windows (завантажується безкоштовно з офіційного сайту Microsoft).

3) Запустити файл FarmApp.exe. При першому запуску автоматично створюється база даних farm.db у папці base.

4) Для входу використати стандартні облікові дані: логін admin, пароль admin123.

5) Рекомендується негайно змінити пароль адміністратора через розділ «Управління користувачами».

Схему навігації системи наведено на рисунку 3.12.

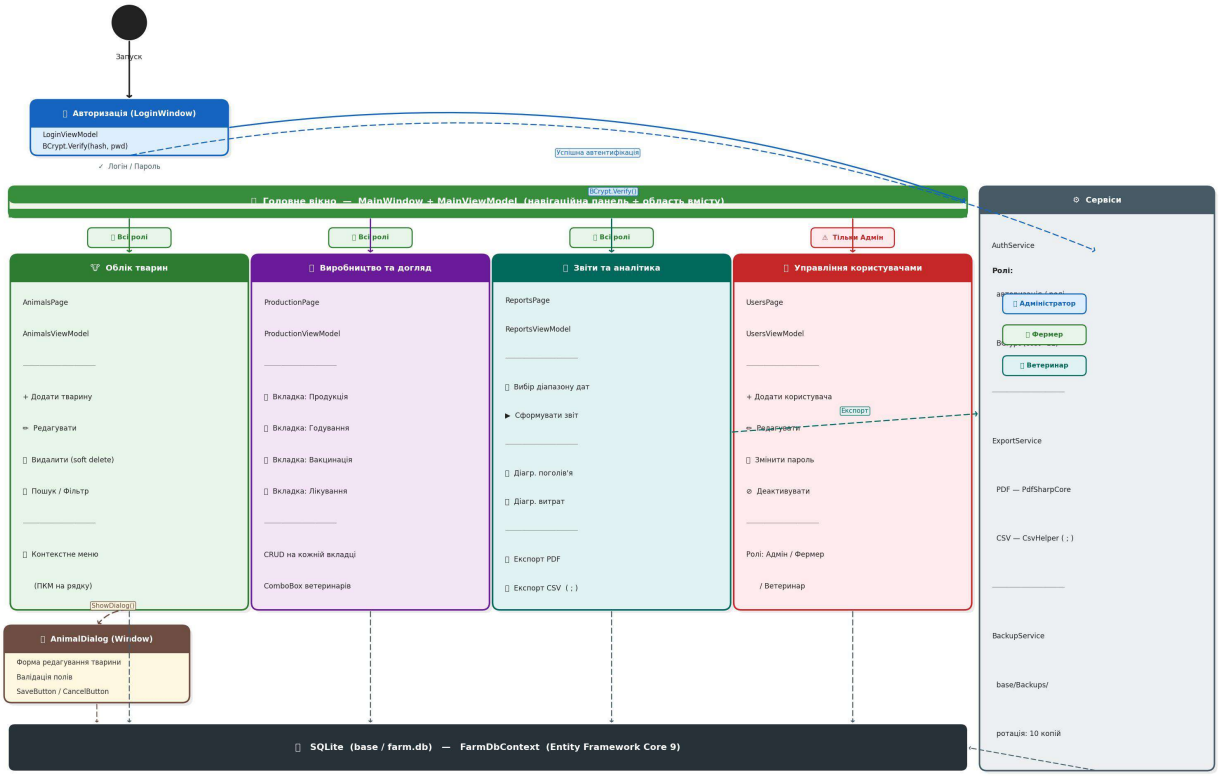

Рисунок 3.12 — Схема навігації системи «ФермерОблік»

Основні операції у системі виконуються за єдиним алгоритмом незалежно від модуля. Для додавання нового запису слід заповнити форму у лівій частині екрана та натиснути кнопку «Зберегти запис». Для редагування слід клацнути по потрібному рядку в таблиці — форма автоматично заповниться поточними даними, після чого можна вносити зміни та зберегти. Для видалення слід вибрати рядок та натиснути «Видалити» — система запитає підтвердження.

Резервне копіювання бази даних виконується кнопкою «Резервна копія» у розділі «Звіти». Копія автоматично зберігається у папці base/Backups з позначкою дати та часу. Система зберігає не більше 10 останніх копій.

Для роботи з ветеринарним модулем необхідно попередньо створити облікові записи ветеринарів у розділі «Управління користувачами» з роллю «Ветеринар». Після цього вони з'являться у випадаючих списках на вкладках «Вакцинація» та «Лікування».

Висновки до розділу 3

У третьому розділі описано реалізацію та тестування інформаційної системи обліку тваринництва «ФермерОблік». Визначено технологічний стек із 13 компонентів, включаючи .NET 9, WPF, Entity Framework Core 9, SQLite, BCrypt.Net, CsvHelper та PdfSharpCore.

Реалізовано базу даних з семи таблиць за допомогою ORM Entity Framework Core з автоматичною ініціалізацією схеми при першому запуску. Шлях до файлу бази даних визначається динамічно відносно розташування виконуваного файлу у підпапці base.

Описано реалізацію ключових модулів системи: авторизацію з BCrypt-хешуванням та рольовим розмежуванням доступу; облік тварин з пошуком на стороні клієнта та патерном м'якого видалення; чотирьохвкладковий модуль виробництва та догляду з повним CRUD і ComboBox вибору ветеринара; модуль звітності з діаграмами та експортом у PDF і CSV.

Проведено функціональне тестування системи у вигляді 28 тест-кейсів для п'яти модулів. Всі тести пройдено успішно, що підтверджує відповідність реалізованих функцій сформульованим вимогам. Надано інструкцію з встановлення та початкового налаштування системи.

ВИСНОВКИ

У кваліфікаційній роботі розроблено інформаційну систему обліку тваринництва фермерського господарства — настільний застосунок «ФермерОблік», що забезпечує комплексну автоматизацію облікових процесів у галузі тваринництва. Відповідно до поставлених завдань отримано такі результати.

1. Проведено аналіз предметної області, в результаті якого виявлено п'ять ключових облікових процесів фермерського господарства: облік поголів'я тварин, виробничий облік, облік годування, ветеринарний облік та облік витрат. Встановлено, що традиційні методи ведення обліку на основі паперових журналів або електронних таблиць не забезпечують необхідного рівня оперативності, точності та захисту даних.

2. Виконано огляд та порівняльний аналіз чотирьох існуючих програмних рішень: AgroBase, FarmFacts, 1С:Сільгосп та Microsoft Excel. Встановлено, що жодне з них не задовольняє повною мірою потреби малих і середніх фермерських господарств України через завищену вартість, відсутність україномовного інтерфейсу або недостатній функціонал. Обґрунтовано доцільність розробки власної спеціалізованої системи.

3. Сформульовано 13 функціональних та 10 нефункціональних вимог до системи. Проведено порівняльний аналіз чотирьох технологій розробки настільних застосунків (WPF, WinForms, Electron, JavaFX) та обрано технологічний стек: C# / .NET 9, WPF, MVVM, SQLite, Entity Framework Core 9.

4. Спроєктовано трирівневу архітектуру системи на основі шаблону MVVM. Розроблено реляційну базу даних із семи таблиць: Users, Animals, ProductionRecords, FeedingRecords, VaccinationRecords, TreatmentRecords, ExpenseRecords. Побудовано ER-діаграму, діаграму компонентів, діаграму варіантів використання, дві діаграми послідовності та діаграму станів тварини. Розроблено матрицю прав доступу для трьох ролей користувачів.

5. Реалізовано всі заплановані модулі системи: модуль авторизації з BCrypt-хешуванням паролів (work factor 11); модуль обліку тварин з повним набором CRUD-операцій, пошуком та фільтрацією; чотирьохвкладковий модуль виробництва та догляду (продукція, годування, вакцинація, лікування); модуль звітності з двома інтерактивними діаграмами; модуль управління користувачами, доступний виключно адміністратору.

6. Забезпечено розмежування доступу для трьох ролей: Адміністратор, Фермер та Ветеринар. Реалізовано механізм резервного копіювання бази даних з автоматичною ротацією (зберігається 10 останніх копій). Застосовано патерн «м'якого видалення» для збереження цілісності пов'язаних облікових даних.

7. Реалізовано функції експорту: звіту у форматі PDF засобами бібліотеки PdfSharpCore та списку тварин і звітних даних у форматі CSV з роздільником «;» засобами CsvHelper, що забезпечує коректне відкриття файлів у Microsoft Excel з українською локалізацією.

8. Проведено функціональне тестування системи у вигляді 28 тест-кейсів для п'яти модулів. Усі тести пройдено успішно (100 % успішних результатів), що підтверджує відповідність реалізованих функцій сформульованим вимогам.

Практичне значення роботи полягає в тому, що розроблений програмний продукт може бути впроваджений у реальних фермерських господарствах України для ведення автоматизованого обліку тваринництва. Система є повністю автономною, не потребує інтернет-з'єднання та доступна без фінансових витрат на ліцензування.

Перспективами подальшого розвитку системи є: розробка веб-версії застосунку для дистанційного доступу; реалізація мобільного додатку для операційних систем Android та iOS; інтеграція з хмарними сховищами для автоматичного резервного копіювання; додавання модуля планування та прогнозування на основі накопичених даних.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Закон України «Про фермерське господарство» від 19.06.2003 № 973-IV. URL: https://zakon.rada.gov.ua/laws/show/973-15 (дата звернення: 10.02.2026).
2. Закон України «Про основні засади державного нагляду (контролю) у сфері господарської діяльності» від 05.04.2007 № 877-V. URL: https://zakon.rada.gov.ua/laws/show/877-16 (дата звернення: 12.02.2026).
3. Albahari J., Albahari B. C# 12 in a Nutshell: The Definitive Reference. O'Reilly Media, 2024. 1120 p.
4. Price M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals. Packt Publishing, 2023. 826 p.
5. MacDonald M. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5. Apress, 2022. 1096 p.
6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 p.
7. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. 432 p.
8. Microsoft Documentation. Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 05.03.2026).
9. Microsoft Documentation. WPF .NET Overview. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/ (дата звернення: 08.03.2026).
10. SQLite Documentation. SQLite Home Page. URL: https://www.sqlite.org/docs.html (дата звернення: 10.03.2026).
11. BCrypt.Net-Next. BCrypt Password Hashing for .NET. URL: https://github.com/BcryptNet/bcrypt.net (дата звернення: 12.03.2026).
12. CsvHelper Documentation. A .NET library for reading and writing CSV files. URL: https://joshclose.github.io/CsvHelper/ (дата звернення: 15.03.2026).
13. PdfSharpCore. Open source .NET library for processing PDF files. URL: https://github.com/ststeiger/PdfSharpCore (дата звернення: 15.03.2026).

14. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 560 p.
15. Welling L., Thomson L. PHP and MySQL Web Development. Addison-Wesley, 2017. 1008 p.
16. ISTQB. Standard Glossary of Terms used in Software Testing. Version 4.0. ISTQB, 2023. URL: https://glossary.istqb.org (дата звернення: 20.03.2026).
17. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. New York: IEEE, 1998. 37 p.
18. Codd E. F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387.
19. Гандзюк М. І. Інформаційні системи та технології в агропромисловому комплексі. Київ: Аграрна наука, 2019. 312 с.
20. Державна служба статистики України. Сільське господарство України: статистичний збірник. Київ, 2024. URL: https://www.ukrstat.gov.ua (дата звернення: 25.05.2026).