ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмних засобів для ведення електронного реєстру свійських тварин»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ілля ЧЕРКЕСОВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Ілля ЧЕРКЕСОВ

Керівник: Ігор ГАМАНЮК
Ст. викладач кафедри

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Черкесову Іллі Валерійовичу

1. Тема кваліфікаційної роботи: «Розробка програмних засобів для ведення електронного реєстру свійських тварин»
керівник кваліфікаційної роботи ст. викладач кафедри Ігор ГАМАНЮК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про автоматизацію обліку поголів'я худоби у фермерському господарстві, опис методів і засобів проєктування інформаційних вебсистем, технічна документація з описом технологій Angular, Spring Boot, MySQL.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих інформаційних систем і підходів до автоматизації обліку поголів'я худоби у фермерських господарствах.
2. Проєктування інформаційної вебсистеми обліку поголів'я худоби для сільськогосподарського підприємства.

3. Програмна реалізація та опис функціонування інформаційної вебсистеми обліку поголів'я худоби.
4. Тестування інформаційної вебсистеми.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Моделювання прецедентів.
5. Моделювання предметної галузі.
6. Логічна архітектура.
7. Алгоритм фільтрації за типом тварини.
8. Проектування фільтрації за типом тварин.
9. Діаграма класів стосовно фільтрації
10. Моделювання меню.
11. Проєктування бази даних
12. Результати тестування основних модулів
13. Екранні форми.
14. Апробація результатів дослідження

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Формування функціональних та нефункціональних вимог до вебсистеми	08.03-10.03.2026	
4	Проєктування архітектури системи, бази даних та UML-діаграм	11.03-14.03.2026	
5	Програмна реалізація застосунку	15.03-02.04.2026	
6	Тестування застосунку	04.04-06.04.2026	
7	Оформлення роботи: вступ, висновки, реферат	10.04-25.04.2026	
8	Розробка демонстраційних матеріалів	26.04-05.05.2026	
9	Попередній захист роботи	11.05-22.05.2026	

Здобувач(ка) вищої освіти _____ Ілля ЧЕРКЕСОВ
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 81 стор., 4 табл., 10 рис., 22 джерела.

Мета роботи – спрощення процесу обліку поголів`я худоби шляхом впровадження вебсистеми “MyCattle+”.

Об’єкт дослідження – процес обліку поголів`я худоби.

Предмет дослідження – засоби, методи та технології реалізації автоматизації обліку поголів'я худоби.

Короткий зміст роботи: В роботі проаналізовано предметну область фермерського господарства та існуючі підходи до автоматизації обліку поголів’я худоби, приміщень, обладнання й пов’язаних даних. Проаналізовано програмні засоби та сучасні технології, що можуть бути використані для створення інформаційної системи такого типу. Розроблено структуру вебсистеми та програмно реалізовані ключові функціональні можливості, зокрема: облік тварин, ведення інформації про елементи ферми, пошук і перегляд даних, розмежування прав доступу користувачів, а також засоби фільтрації за типом тварин. Проведено модульне тестування застосунку. В роботі використано Angular для створення клієнтської частини, Spring Boot для реалізації серверної логіки, MySQL для збереження даних, а також у системі передбачено можливість подальшого розвитку веб застосунку для інтеграції з IoT-пристроями через модулі Device та DeviceData.

Сферою використання застосунку є фермерське господарство.

КЛЮЧОВІ СЛОВА: ОБЛІК ПОГОЛІВ’Я ХУДОБИ, ВЕБСИСТЕМА, ФЕРМЕРСЬКЕ ГОСПОДАРСТВО, ANGULAR, SPRING BOOT, MYSQL

ЗМІСТ

ВСТУП

Сільське господарство є однією з важливих галузей економіки, а ефективність роботи фермерських господарств значною мірою залежить від якості обліку та керування наявними ресурсами. Одним із ключових напрямів діяльності тваринницьких господарств є ведення обліку поголів'я худоби, контроль інформації про тварин, місця їх утримання, виробничі об'єкти та персонал. Зі збільшенням обсягів даних використання паперових журналів або неструктурованих електронних таблиць стає менш ефективним, оскільки ускладнює пошук інформації, підвищує ризик виникнення помилок та потребує значних витрат часу на виконання облікових операцій.

Сучасні інформаційні технології дозволяють автоматизувати значну частину процесів зберігання та оброблення даних. Використання веборієнтованих інформаційних систем забезпечує централізований доступ до інформації, спрощує роботу користувачів та створює умови для подальшого розвитку функціональних можливостей програмного забезпечення. Особливої актуальності набуває впровадження таких систем у сфері тваринництва, де необхідно працювати з великими обсягами інформації про тварин, ферми, приміщення, обладнання та персонал.

Актуальність теми кваліфікаційної роботи полягає в необхідності створення сучасної інформаційної вебсистеми, яка дозволить автоматизувати процеси обліку поголів'я худоби, забезпечити централізоване зберігання даних та підвищити ефективність роботи фермерського господарства.

Метою кваліфікаційної роботи є спрощення процесу обліку поголів'я худоби шляхом впровадження вебсистеми "MyCattle+".

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області та особливостей обліку поголів'я худоби;
- виконати огляд існуючих програмних рішень та визначити їх переваги

й недоліки;

- сформувати функціональні та нефункціональні вимоги до системи;
- спроєктувати архітектуру вебзастосунку та структуру бази даних;
- реалізувати клієнтську частину вебсистеми із використанням Angular;
- реалізувати серверну частину із використанням Spring Boot;
- організувати взаємодію між компонентами системи за допомогою REST API;
- реалізувати механізми авторизації та контролю доступу користувачів;
- провести тестування створеного програмного забезпечення та оцінити результати його роботи.

Об'єктом дослідження є процес обліку поголів'я худоби у фермерському господарстві.

Предметом дослідження є методи та засоби проєктування й розроблення інформаційних вебсистем для автоматизації процесів обліку в галузі тваринництва.

Під час виконання роботи використовувалися методи аналізу предметної області, порівняльного аналізу програмних рішень, об'єктно-орієнтованого проєктування, моделювання інформаційних систем, технології розроблення вебзастосунків та методи тестування програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні інформаційної вебсистеми, яка може використовуватися для автоматизації процесів обліку поголів'я худоби, зберігання інформації про об'єкти фермерського господарства та централізованого доступу до даних. Розроблена архітектура дозволяє надалі розширювати функціональність системи, зокрема шляхом розвитку аналітичних модулів.

У першому розділі виконано аналіз предметної області та існуючих аналогів. Другий розділ присвячений проєктуванню інформаційної системи. У третьому розділі розглянуто програмну реалізацію вебсистеми. У четвертому розділі наведено результати тестування та апробації розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ Й ІСНУЮЧИХ ЗАСОБІВ АВТОМАТИЗАЦІЇ ОБЛІКУ ПОГОЛІВ'Я ХУДОБИ

1.1 Аналіз предметної області фермерського господарства та процесу обліку поголів'я худоби

Фермерське господарство, діяльність якого пов'язана з утриманням худоби, являє собою складну систему, у межах якої постійно виконуються виробничі, організаційні та облікові процеси. Одним із найважливіших напрямів такої діяльності є ведення обліку поголів'я.

Основним об'єктом обліку виступає тварина. Для кожної тварини необхідно зберігати набір характеристик: ідентифікаційний номер, вид, породу, дату народження, стать, масу, стан здоров'я, продуктивність та інші параметри. Крім цього, важливими є відомості про місце утримання, історію переміщень і зміни стану тварини протягом її життєвого циклу.

Структура сучасного фермерського господарства зазвичай включає кілька взаємопов'язаних елементів: ферми, приміщення, секції утримання, персонал та технічні засоби контролю. У багатьох господарствах дедалі частіше використовуються цифрові датчики та автоматизовані системи моніторингу, які дозволяють отримувати інформацію про температуру, вологість, активність тварин або інші показники в режимі реального часу.

У наукових дослідженнях та сучасній аграрній практиці поширюється концепція Precision Livestock Farming (PLF). Вона передбачає використання датчиків, камер, алгоритмів аналізу даних і програмного забезпечення для постійного моніторингу тварин та умов їх утримання. Дослідники зазначають, що такі технології допомагають покращувати контроль за здоров'ям тварин і підвищувати ефективність роботи ферми.

У процесі ведення обліку працівники ферми регулярно виконують

однотипні операції: додають нових тварин, оновлюють інформацію про них, здійснюють пошук записів, формують списки та звіти, контролюють розподіл поголів’я між

приміщеннями. Якщо ці процеси виконуються вручну, зростає ризик втрати інформації, дублювання записів або виникнення помилок.

Ще однією особливістю предметної області є постійна зміна даних. Кількість тварин може змінюватися через народження, продаж, переведення між фермами або вибракування. Через це система обліку повинна забезпечувати швидке оновлення інформації та зручний доступ до актуальних даних.

Таким чином, процес обліку поголів’я худоби пов’язаний із великою кількістю взаємопов’язаних даних, які необхідно систематизувати та контролювати. Саме тому використання спеціалізованої вебсистеми є доцільним рішенням для автоматизації основних операцій та підвищення ефективності управління фермерським господарством.

1.2 Огляд існуючих інформаційних систем обліку в аграрній сфері

Розвиток цифрового сільського господарства сприяв появі великої кількості програмних рішень для автоматизації роботи фермерських господарств. Сьогодні на ринку представлені як комплексні системи управління агробізнесом, так і вузькоспеціалізовані продукти для ведення обліку тварин.

До найбільш відомих рішень належать AgriWebb, CattleMax, FarmIQ та інші системи управління тваринництвом. Вони дозволяють вести електронні картки тварин, відстежувати переміщення поголів’я, контролювати виробничі показники та формувати аналітичні звіти. У багатьох випадках такі платформи також підтримують мобільний доступ і синхронізацію даних між пристроями.

Загалом існуючі системи можна поділити на три основні групи:

комплексні системи управління агропідприємством;

спеціалізовані системи обліку тварин;

цифрові платформи з інтеграцією датчиків та засобів моніторингу.

Більшість сучасних рішень забезпечує централізоване зберігання інформації, формування історії подій, пошук, сортування, фільтрацію та створення звітів. Також поширеною стає інтеграція з технологіями цифрового моніторингу.

За даними досліджень у сфері цифрового тваринництва, сучасні системи моніторингу дозволяють автоматично відстежувати стан тварин у режимі реального часу, аналізувати поведінку та виявляти потенційні проблеми ще до появи очевидних симптомів захворювань.

Разом із тим на практиці використання готових програмних продуктів не завжди є оптимальним рішенням. Частина систем орієнтована на великі підприємства та містить надлишковий функціонал. Це ускладнює впровадження, збільшує вартість використання та потребує додаткового навчання персоналу.

Для невеликих і середніх фермерських господарств важливішими є простота інтерфейсу, швидкий доступ через браузер, зручне редагування інформації та можливість адаптації системи до власної структури господарства.

Тому навіть за наявності великої кількості готових рішень залишається потреба у створенні спеціалізованих вебсистем, які враховують конкретні потреби користувачів і не перевантажені другорядними функціями.

Таблиця 1.2

Таблиця порівняння аналогів

Характеристика	**CattleMax**	**Herdwatch**	**Cattlytics**	**Livestock Analytics**	**MyCattle+**
Облік поголів'я худоби	Так	Так	Так	Так	Так
Підтримка кількох ферм	Обмежена	Часткова	Так	Так	Так
Робота з користувачами	Так	Так	Так	Так	Так
Підтримка IoT-пристроїв	Часткова	Часткова	Обмежена	Відсутня	Так
Гнучкість налаштування системи	Низька	Низька	Середня	Середня	Висока

Продовження таблиці 1.2

Таблиця порівняння аналогів

Характеристика	**CattleMax**	**Herdwatch**	**Cattlytics**	**Livestock Analytics**	**MyCattle+**
Можливість розширення функціоналу	Обмежена	Обмежена	Так	Так	Так
Вартість використання	Умовно безкоштовна підписка	Підписка	Підписка	Комерційна	Безкоштовна або низька вартість
Простота використання	Висока	Середня	Середня	Середня	Висока

1.3 Порівняльний аналіз аналогів та обґрунтування необхідності розроблення вебсистеми

Перед початком розроблення власної інформаційної системи доцільно проаналізувати вже існуючі рішення та визначити їх сильні й слабкі сторони.

Під час порівняння аналогів варто враховувати такі критерії:

функціональні можливості;

зручність інтерфейсу;

доступність через веббраузер;

підтримку обліку тварин;

підтримку структури ферми;

наявність пошуку та фільтрації;

можливість масштабування;

інтеграцію із зовнішніми сервісами.

Проведений аналіз показує, що багато сучасних платформ мають широкий функціонал і підтримують велику кількість бізнес-процесів. Однак саме через це вони часто стають складними для освоєння та потребують значних ресурсів для впровадження.

Водночас простіші рішення іноді не забезпечують достатньої гнучкості під час організації структури ферми або не підтримують необхідні функції керування

даними.

Окрему увагу варто приділити веборієнтованим системам. Їх перевага полягає в тому, що користувач може працювати із системою практично з будь-якого пристрою без встановлення додаткового програмного забезпечення. Для фермерського господарства це особливо зручно, оскільки доступ до даних можна отримати як з офісного комп’ютера, так і безпосередньо під час роботи на фермі.

У сучасних дослідженнях цифрового сільського господарства зазначається, що впровадження цифрових технологій дозволяє підвищити продуктивність, покращити прийняття рішень та ефективніше використовувати ресурси господарства.

Саме тому виникає потреба у створенні спеціалізованої вебсистеми, яка поєднуватиме необхідний функціонал без надлишкової складності. Така система повинна забезпечувати зручний облік тварин, керування структурою ферми, роботу з користувачами, швидкий пошук інформації та зрозумілий інтерфейс.

У результаті можна зробити висновок, що розроблення власної інформаційної вебсистеми є доцільним, оскільки вона дозволить максимально адаптувати функціонал до потреб конкретного фермерського господарства.

1.4 Постановка задачі дослідження

Проведений аналіз предметної області показав, що процес обліку поголів’я худоби потребує сучасного програмного рішення, яке забезпечить централізоване зберігання інформації, зручну роботу з даними та підтримку основних операцій користувачів.

Метою роботи є розроблення інформаційної вебсистеми обліку поголів’я худоби для фермерського господарства, яка дозволить автоматизувати процеси ведення, оновлення, пошуку та перегляду інформації про тварин, елементи ферми та користувачів системи.

Для досягнення поставленої мети необхідно виконати такі завдання:

проаналізувати предметну область фермерського господарства;

дослідити особливості обліку поголів’я худоби;

виконати огляд сучасних інформаційних систем аграрного спрямування;

провести порівняльний аналіз існуючих аналогів;

сформувати функціональні та нефункціональні вимоги до системи;

обґрунтувати вибір технологій розроблення;

спроєктувати архітектуру вебсистеми;

реалізувати клієнтську та серверну частини застосунку;

реалізувати модулі керування тваринами, фермами та користувачами;

реалізувати засоби пошуку, сортування та фільтрації інформації;

провести тестування та оцінити результати роботи системи.

Об’єктом дослідження є процес обліку поголів’я худоби у фермерському господарстві.

Предметом дослідження є методи, засоби та програмні рішення автоматизації обліку поголів’я худоби з використанням вебтехнологій.

Практичне значення роботи полягає у створенні вебсистеми, яка може використовуватися для впорядкування інформації про тварин, ферми, приміщення та користувачів, а також для підвищення ефективності управління даними в межах господарства.

Отже, результати проведеного аналізу підтверджують актуальність розроблення спеціалізованої інформаційної вебсистеми для автоматизації обліку поголів’я худоби та визначають основні напрями її подальшого проєктування й реалізації.

Висновки до розділу 1

Проведений порівняльний аналіз аналогів дозволив обґрунтувати потребу у створенні спеціалізованої вебсистеми, яка поєднує базові функції обліку поголів'я, керування елементами ферми, роботу з користувачами, пошук, фільтрацію та можливість подальшого розширення. За результатами розділу сформульовано

основні напрями подальшого проєктування та визначено завдання, які мають бути реалізовані в межах кваліфікаційної роботи.

У межах аналізу було розглянуто існуючі інформаційні системи аграрного спрямування та визначено їх основні переваги й недоліки. Встановлено, що готові програмні продукти часто мають широкий набір функцій, але не завжди є зручними для невеликих і середніх фермерських господарств. Частина систем орієнтована на великі підприємства, має складну структуру впровадження або передбачає комерційну модель використання, що може бути недоцільним для окремих господарств.

У першому розділі було розглянуто предметну область фермерського господарства та визначено особливості процесу обліку поголів'я худоби. Показано, що облік тварин пов'язаний не лише зі зберіганням окремих записів, а й з постійним оновленням інформації про стан тварин, місця їх утримання, переміщення, відповідальних працівників та інші об'єкти господарства. Така предметна область потребує структурованого підходу до організації даних, оскільки ручне ведення обліку або використання неузгоджених таблиць поступово ускладнює пошук, редагування та контроль інформації.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ВЕБСИСТЕМИ ОБЛІКУ ПОГОЛІВ'Я ХУДОБИ

2.1 Формування функціональних та нефункціональних вимог до системи

Першим етапом проєктування інформаційної вебсистеми є визначення вимог, яким вона повинна відповідати. Від правильності їх формування залежить структура майбутнього програмного продукту, перелік реалізованих функцій та загальна ефективність роботи системи після впровадження.

У результаті аналізу предметної області було встановлено, що працівникам фермерського господарства необхідний інструмент для централізованого зберігання та оброблення інформації про поголів’я худоби. Крім даних про тварин, система повинна забезпечувати роботу з інформацією про ферми, приміщення, пристрої моніторингу та користувачів.

До функціональних вимог належать можливості, які система повинна надавати користувачеві в процесі роботи. Основною функцією є ведення обліку тварин. Користувач повинен мати можливість створювати нові записи про тварин, переглядати наявну інформацію, вносити зміни та видаляти непотрібні записи.

Для кожної тварини в системі зберігаються такі дані, як ідентифікатор, тип тварини, порода, стать, дата народження, місце утримання та додаткові характеристики. Це дозволяє формувати повну електронну картку об’єкта обліку та швидко отримувати необхідну інформацію.

Крім роботи з тваринами, система повинна підтримувати керування іншими сутностями предметної області. Передбачається можливість створення та редагування записів про ферми, приміщення, пристрої та користувачів. Завдяки цьому структура господарства відображається у системі максимально наближено до реальних умов експлуатації.

Окрему увагу приділено механізмам пошуку та фільтрації інформації.

Оскільки кількість записів може поступово збільшуватися, користувач повинен мати можливість швидко знаходити необхідні дані. Для цього передбачено

фільтрацію тварин за типом, а також пошук потрібних записів серед наявних об’єктів.

Ще однією важливою функцією є авторизація користувачів. Перед початком роботи користувач проходить процедуру автентифікації, після чого отримує доступ до функцій системи. Такий підхід дозволяє забезпечити контроль доступу до даних та створює основу для подальшого впровадження ролей користувачів.

Крім функціональних можливостей, система повинна відповідати певним нефункціональним вимогам. Насамперед це стосується зручності використання. Інтерфейс має бути зрозумілим навіть для користувачів без спеціальної технічної підготовки. Розміщення елементів керування повинно бути логічним, а виконання типових операцій не повинно вимагати значних часових витрат.

Важливою характеристикою є швидкодія. Система повинна оперативно опрацьовувати запити користувача, швидко відображати інформацію та забезпечувати комфортну роботу навіть за збільшення кількості даних.

Також під час проєктування враховувалася можливість подальшого розвитку системи. Архітектура повинна дозволяти додавання нових модулів, сутностей або сервісів без необхідності повного перепроєктування вже реалізованих компонентів.

Таким чином, сформовані вимоги визначають функціональні межі вебсистеми та виступають основою для вибору технологій і побудови архітектури програмного продукту.

Таблиця 2.1

Функціональні і нефункціональні вимоги

Тип вимоги	**Найменування вимоги**
Функціональна	Ведення обліку подій життєвого циклу тварин
Функціональна	Забезпечення пошуку і перегляду інформації про тварин за різними параметрами (ID, порода, статус)

Продовження таблиці 2.1

Функціональні і нефункціональні вимоги

Тип вимоги	Найменування вимоги
Функціональна	Створення звітів про стан поголів’я (кількість, структура, зміни за період)
Функціональна	Формування ієрархічної системи доступу користувачів до кількості данних у БД та можливостей редагування інформації у БД відповідно до рівня допуску
Функціональна	Підтримка додавання/редагування/видалення елементів ферми, користувачів та худоби
Функціональна	Визначення адміністратором прав доступу користувачів.
Функціональна	Можливість експорту даних про конкретну тварину.
Нефункціональна	Забезпечення продуктивності. Обробка запитів користувача не більше 2 секунд.
Нефункціональна	Система має забезпечувати одночасну роботу до 1000 користувачів.
Нефункціональна	Захист доступу до даних через систему авторизації з ролями.

2.2 Програмні засоби реалізації вебсистеми

Для реалізації інформаційної вебсистеми обліку поголів’я худоби було обрано сучасний стек технологій, який забезпечує створення масштабованого, надійного та зручного програмного продукту. Розробка системи виконувалась із використанням технологій HTML5, CSS3, TypeScript, Angular, Java, Spring Boot, MySQL та Swagger.

Мова розмітки HTML5 використовується для створення структури вебінтерфейсу системи. За допомогою HTML формуються сторінки відображення інформації про тварин, ферми, приміщення, датчики та користувачів. Технологія забезпечує коректне відображення контенту у сучасних веббраузерах та підтримує адаптивну побудову інтерфейсу.

Для оформлення інтерфейсу застосовується CSS3. Використання каскадних таблиць стилів дозволяє створити зручний користувацький інтерфейс,

налаштовувати розташування елементів сторінок, кольорову схему, адаптивність та забезпечувати єдиний стиль оформлення всієї системи.

Мова програмування TypeScript використовується для реалізації клієнтської логіки вебзастосунку. Вона розширює можливості JavaScript за рахунок статичної типізації, що підвищує надійність програмного коду та спрощує його супровід. TypeScript використовується як основна мова розробки фронтенд-частини проєкту.

Фреймворк Angular застосовується для створення односторінкового вебзастосунку. Angular забезпечує компонентний підхід до розробки інтерфейсу, підтримує маршрутизацію, двостороннє зв’язування даних та взаємодію з серверною частиною через HTTP-запити. За допомогою Angular реалізовано сторінки управління поголів’ям худоби, фермами, приміщеннями, датчиками та користувачами системи.

Для реалізації серверної частини використовується мова програмування Java. Дана технологія характеризується високою продуктивністю, переносимістю та широким використанням у корпоративних інформаційних системах. Java забезпечує виконання бізнес-логіки вебсистеми та взаємодію із базою даних.

Фреймворк Spring Boot використовується для побудови серверного застосунку. Він дозволяє швидко створювати REST API, реалізовувати механізми автентифікації, обробки запитів та управління даними. За допомогою Spring Boot реалізовано функціональність роботи з тваринами, фермами, приміщеннями, датчиками, користувачами та журналом подій системи.

Для зберігання інформації використовується система управління базами даних MySQL. У базі даних зберігаються відомості про поголів’я худоби, характеристики тварин, дані ферм, приміщень, показники датчиків, облікові записи користувачів та журнал подій. MySQL забезпечує надійне та ефективне зберігання інформації із можливістю швидкого доступу до даних.

Для документування та тестування програмного інтерфейсу використовується Swagger. Даний інструмент автоматично формує документацію REST API на основі серверного коду, дозволяє переглядати доступні методи, параметри запитів та структуру відповідей. Використання Swagger значно

спрощує процес інтеграції клієнтської та серверної частин системи, а також полегшує тестування програмного забезпечення під час розробки.

Обраний набір технологій дозволяє створити сучасну інформаційну вебсистему обліку поголів'я худоби, яка забезпечує ефективне управління даними, підтримує масштабування функціоналу та відповідає вимогам до сучасних вебзастосунків.

Рис. 2.1 Програмні засоби реалізації (логотипи)

2.3 Моделювання прецедентів використання системи

Для визначення основних сценаріїв взаємодії користувача із системою було виконано моделювання прецедентів використання. Використання діаграми прецедентів дозволяє наочно відобразити функціональні можливості системи та показати, які дії можуть виконувати користувачі під час роботи.

У системі передбачено два типи користувачів — працівник та супервайзер. Працівник використовує основні функції системи для роботи з інформацією про тварин та пов'язаними даними, тоді як супервайзер додатково виконує функції адміністрування, керування довідниками та налаштуваннями системи.

Після успішного входу до системи користувач отримує доступ до основних

модулів. Одним із найважливіших сценаріїв є робота з тваринами. У межах цього сценарію користувач може переглядати список тварин, створювати нові записи, редагувати наявну інформацію або видаляти записи, які більше не

використовуються.

Окрім цього, користувач має можливість працювати з інформацією про ферми, приміщення та пристрої. Для кожної із зазначених сутностей підтримуються операції створення, перегляду, редагування та видалення записів.

Важливим елементом взаємодії є використання пошуку та фільтрації. Наприклад, користувач може обрати певний тип тварини та отримати список записів, які відповідають зазначеному критерію. Це дозволяє значно скоротити час пошуку інформації під час роботи з великими обсягами даних.

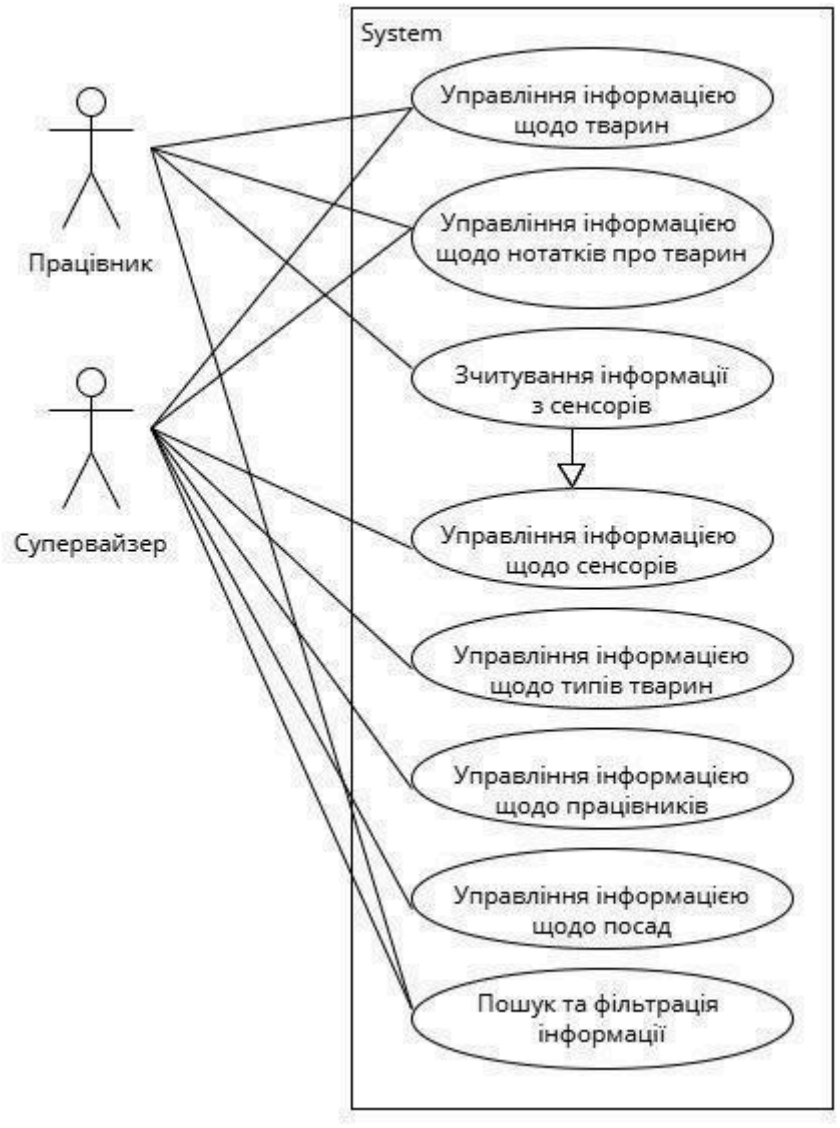


Рис. 2.2 діаграма прецедентів використання “MyCattle+”

Частина сценаріїв пов’язана між собою. Наприклад, редагування або видалення запису можливе лише після відкриття відповідного списку та вибору конкретного об’єкта. Аналогічно використання фільтрації відбувається в межах роботи зі списками даних і є допоміжною функцією для швидкого пошуку інформації.

Результатом моделювання стала діаграма прецедентів використання, яка відображає основні функціональні можливості системи та взаємодію користувача з її компонентами. Отримана модель буде використана під час подальшого проєктування структури меню, інтерфейсу та серверної логіки вебзастосунку.

Опис діаграми прецедентів

Діаграма прецедентів відображає функціональні можливості вебсистеми обліку поголів'я худоби та взаємодію користувачів із системою. На діаграмі представлені два актори — працівник та супервайзер, які мають різні рівні доступу до функцій системи.

Актор Працівник

Працівник є користувачем системи, який виконує щоденну роботу з інформацією про тварин. Він має доступ до управління інформацією щодо тварин, управління інформацією щодо нотатків про тварин та зчитування інформації з сенсорів. За допомогою цих функцій працівник може переглядати та оновлювати відомості про поголів'я, працювати із записами щодо тварин та отримувати актуальні дані від пристроїв моніторингу.

Актор Супервайзер

Супервайзер має розширені повноваження та відповідає за адміністрування даних системи. Крім можливостей, доступних працівнику, він може здійснювати управління інформацією щодо сенсорів, типів тварин, працівників та посад. Це дозволяє контролювати структуру довідкових даних та забезпечувати актуальність інформації, що використовується в системі.

Управління інформацією щодо тварин

Даний прецедент забезпечує створення, перегляд, редагування та видалення інформації про тварин. Користувачі можуть працювати з відомостями про тип

тварини, дату народження, стан здоров'я та іншими характеристиками.

Управління інформацією щодо нотатків про тварин

Прецедент призначений для ведення додаткових записів щодо тварин. Користувачі можуть створювати, переглядати та редагувати нотатки, які містять важливу інформацію про події, спостереження або зміни стану тварин.

Зчитування інформації з сенсорів

Прецедент забезпечує отримання даних від сенсорів та пристроїв моніторингу. Отримана інформація використовується для контролю параметрів, пов'язаних із тваринами та умовами їх утримання.

Управління інформацією щодо сенсорів

Прецедент призначений для налаштування та адміністрування сенсорів. Супервайзер може додавати нові пристрої, змінювати їх параметри та контролювати їхню роботу. На діаграмі цей прецедент пов'язаний із прецедентом зчитування інформації з сенсорів відношенням узагальнення, оскільки робота з даними сенсорів є складовою частиною управління сенсорами.

Управління інформацією щодо типів тварин

Даний прецедент дозволяє керувати довідником типів тварин. Супервайзер може додавати нові типи, редагувати існуючі записи та видаляти непотрібні дані.

Управління інформацією щодо працівників

Прецедент забезпечує управління відомостями про працівників підприємства. До його функцій належать створення, перегляд, редагування та видалення інформації про персонал.

Управління інформацією щодо посад

Прецедент використовується для ведення довідника посад працівників. Супервайзер може створювати нові посади, змінювати існуючі записи та підтримувати актуальність організаційної структури підприємства.

Пошук та фільтрація інформації

Даний прецедент призначений для швидкого пошуку необхідних даних серед наявних записів системи. Користувач може виконувати пошук інформації та застосовувати фільтрацію за визначеними критеріями, зокрема за типом тварини.

Використання даного прецеденту дозволяє скоротити час пошуку потрібних відомостей та підвищує ефективність роботи з великими обсягами даних.

Зв'язки між акторами і прецедентами

Працівник взаємодіє з прецедентами управління інформацією щодо тварин, управління інформацією щодо нотатків про тварин та зчитування інформації з сенсорів. Супервайзер взаємодіє з усіма прецедентами системи та має розширені права доступу до функцій адміністрування. Такий розподіл повноважень забезпечує розмежування доступу між звичайними користувачами та адміністративним персоналом, що підвищує безпеку та ефективність роботи системи.

2.4 Побудова діаграми предметної галузі

Після визначення функціональних вимог та основних сценаріїв використання системи наступним етапом проєктування є побудова діаграми предметної галузі. Вона дозволяє формалізувати структуру майбутньої інформаційної системи та відобразити взаємозв’язки між її основними сутностями.

Під час аналізу предметної області було встановлено, що центральним об’єктом системи є тварина. Саме навколо обліку тварин формується основна функціональність вебсистеми. Для кожної тварини зберігаються індивідуальні характеристики, необхідні для її ідентифікації та подальшого обліку.

Тварина належить до певного місця утримання та пов’язана з конкретною фермою. Такий підхід дозволяє відображати реальну структуру господарства та контролювати розміщення поголів’я. Крім того, одна ферма може містити декілька будівель або приміщень, у яких утримуються різні групи тварин.

Окремою сутністю є пристрій моніторингу. Пристрої використовуються для збору інформації про тварин та пов’язані з ними показники. Отримані дані можуть використовуватися для контролю стану тварин та автоматизованого моніторингу параметрів, необхідних для ведення обліку поголів’я худоби.

Для забезпечення роботи із системою передбачена сутність користувача. Користувачі виконують операції зі створення, перегляду та редагування даних. У перспективі ця сутність може бути розширена механізмом ролей та розмежування прав доступу.

Побудована діаграма предметної галузі дозволяє визначити основні зв’язки між об’єктами системи та створює основу для подальшого проєктування структури бази даних. На її основі формуються таблиці, атрибути та зовнішні ключі, які використовуватимуться під час реалізації програмного продукту.

Використання діаграми предметної галузі також дозволяє перевірити повноту моделі та переконатися, що всі важливі елементи фермерського господарства враховані під час проєктування системи.

Опис діаграми предметної галузі

Діаграма предметної галузі відображає основні сутності вебсистеми обліку поголів'я худоби та зв'язки між ними. Модель описує структуру даних, що використовується для зберігання інформації про тварин, працівників, ферми, будівлі, датчики та службові записи.

Сутність Farm

Сутність призначена для зберігання інформації про ферми підприємства. Вона містить атрибути ID, FarmTypeID та Name. Кожна ферма може містити декілька будівель, що використовуються для утримання тварин та розміщення обладнання.

Сутність FarmBuilding

Містить інформацію про будівлі, що належать фермам. До її складу входять атрибути ID, FarmID, BuildingID та BuildingName. Кожна будівля належить одній фермі, тоді як одна ферма може містити багато будівель.

Сутність Employee

Використовується для зберігання інформації про працівників підприємства. Вона містить атрибути ID, EmployeeFirstName, EmployeeLastName, Occupation, PhoneNumber та Email. Працівники здійснюють догляд за тваринами та виконують обов'язки відповідно до займаної посади.

Сутність Occupation

Містить інформацію про посади працівників. Вона включає атрибути ID, OccupationName та ChiefOccupation. Одна посада може бути призначена багатьом працівникам, тоді як кожен працівник займає лише одну посаду.

Сутність Animal

Є центральним елементом предметної галузі та призначена для зберігання інформації про тварин. Вона містить атрибути ID, EmployeeID, AnimalType, AnimalName, AnimalBirthday, HealthStatus та Notes. Для кожної тварини зберігаються відомості про її тип, дату народження, стан здоров'я та відповідального працівника.

Сутність AnimalType

Використовується для класифікації тварин за видами. Вона містить атрибути ID та TypeName. Один тип тварини може бути пов'язаний із багатьма тваринами, тоді як кожна тварина належить лише до одного типу.

Сутність Note

Сутність призначена для зберігання додаткових записів та приміток щодо тварин. Вона містить атрибути ID, Text та AnimalID. Одна тварина може мати багато приміток, які використовуються для фіксації важливої інформації про її стан або події.

Сутність Devices

Використовується для зберігання інформації, отриманої від датчиків та пристроїв моніторингу. Вона містить атрибути ID, DeviceID, Value, Time, Timestamp та DeviceName. Дані пристроїв дозволяють здійснювати автоматизований контроль параметрів, пов'язаних із тваринами.

Зв'язки між сутностями

Між сутностями предметної галузі існує система взаємозв'язків. Ферма містить багато будівель, а кожна будівля належить одній фермі. Працівник займає одну посаду, тоді як одна посада може бути призначена багатьом працівникам. Кожен працівник доглядає за декількома тваринами, а кожна тварина закріплена за одним працівником. Кожна тварина належить до одного типу тварин, тоді як один

тип може бути пов'язаний із багатьма тваринами. Для кожної тварини може зберігатися декілька приміток та використовуватися декілька пристроїв моніторингу. Сукупність цих зв'язків забезпечує цілісність даних та дозволяє ефективно організувати процес обліку поголів'я худоби на підприємстві.

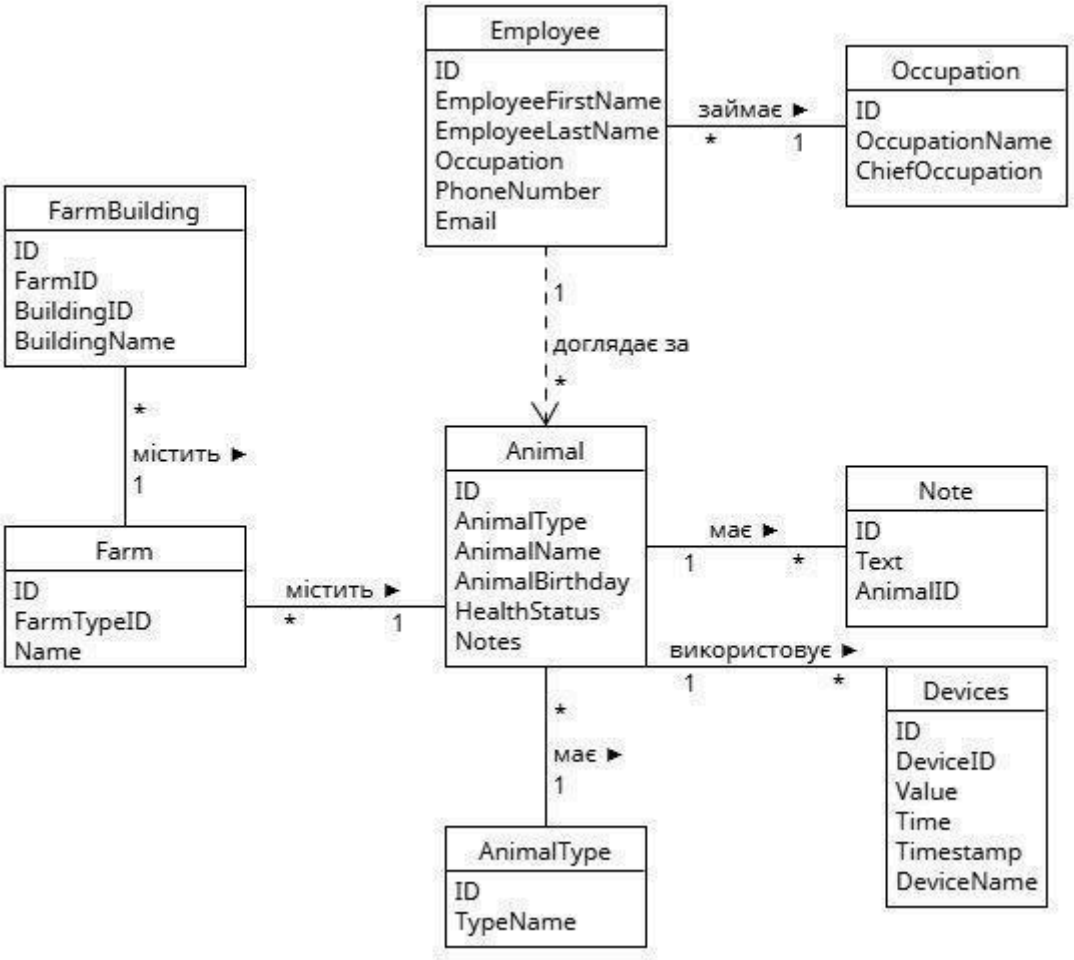


Рис. 2.3 діаграма предметної галузі “MyCattle+”

2.5 Проєктування логічної архітектури системи

Одним із найважливіших етапів проєктування є визначення логічної архітектури програмного продукту. Архітектура визначає структуру системи, спосіб взаємодії її компонентів та розподіл відповідальності між окремими рівнями застосунку.

Для вебсистеми обліку поголів’я худоби було обрано багаторівневу

архітектуру, яка складається з клієнтського рівня, серверного рівня та рівня зберігання даних. Подібний підхід широко використовується під час розроблення сучасних вебзастосунків, оскільки забезпечує чіткий розподіл функціональності між компонентами системи.

Клієнтський рівень реалізований за допомогою Angular та відповідає за взаємодію користувача із системою. Саме на цьому рівні відображаються сторінки вебзастосунку, форми введення даних, таблиці з інформацією про тварин, ферми, пристрої та інші об’єкти. Також клієнтська частина відповідає за перевірку введених даних і формування запитів до сервера.

Серверний рівень реалізується на основі Spring Boot і забезпечує виконання бізнес-логіки системи. Він приймає запити від клієнтської частини, виконує необхідні перевірки, обробляє інформацію та формує відповіді для користувача. На цьому рівні реалізуються операції створення, редагування, видалення та пошуку записів.

Для обміну даними між клієнтом і сервером використовується REST API. Такий підхід дозволяє розділити інтерфейс користувача та серверну логіку на незалежні компоненти, що спрощує підтримку та модернізацію програмного забезпечення.

Рівень зберігання даних представлений системою керування базами даних MySQL. Саме тут розміщуються всі дані про тварин, ферми, приміщення, пристрої та користувачів. Використання реляційної моделі дозволяє підтримувати цілісність інформації та забезпечувати коректні зв’язки між сутностями.

Під час проєктування архітектури також враховувалася можливість подальшого розвитку системи. У майбутньому до неї можуть бути додані нові модулі, наприклад журнал подій, система сповіщень, засоби аналітики або інтеграція з реальними датчиками моніторингу. Завдяки багаторівневій архітектурі такі

зміни можуть бути реалізовані без суттєвої перебудови вже створених компонентів.

Важливою перевагою обраного підходу є можливість незалежного розвитку

окремих частин системи. Наприклад, зміни інтерфейсу користувача не потребують модифікації структури бази даних, а зміни бізнес-логіки можуть виконуватися без повного перепроєктування клієнтської частини.

Таким чином, використання багаторівневої архітектури забезпечує гнучкість, масштабованість та зручність супроводу вебсистеми обліку поголів'я худоби, що є важливим для її подальшого розвитку та експлуатації.

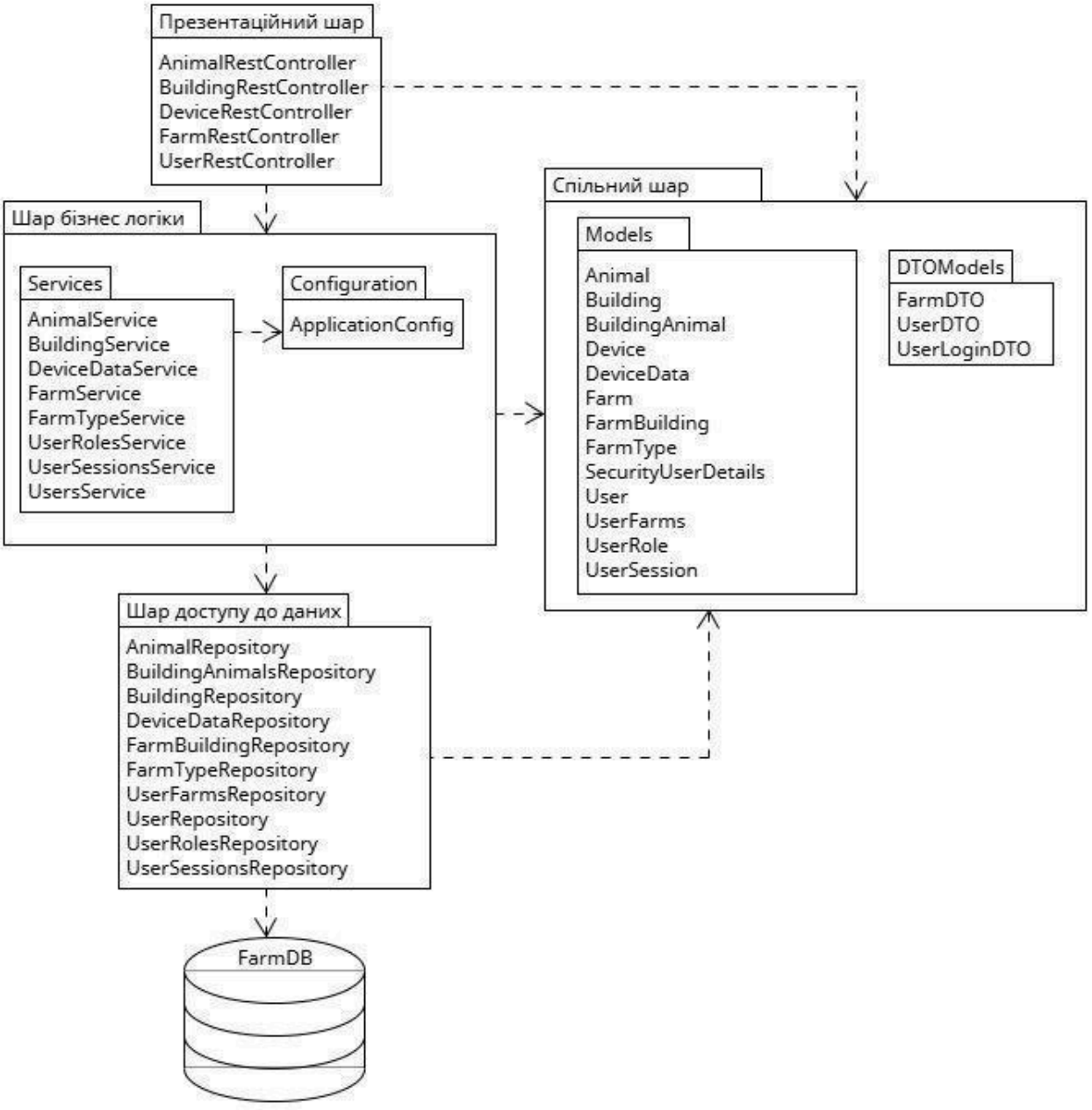


Рис. 2.3 логічна архітектура "MyCattle+"

2.6 Проєктування алгоритму фільтрації за типом тварини

Під час роботи з інформаційною системою користувачеві часто необхідно швидко знаходити потрібні записи серед великої кількості наявних даних. У системі обліку поголів’я худоби такою задачею є пошук тварин певного типу.

Саме тому одним із важливих елементів функціональності було передбачено механізм фільтрації.

Фільтрація дозволяє відображати лише ті записи, які відповідають заданому критерію. У межах розроблюваної вебсистеми основним критерієм є тип тварини. Наприклад, користувач може окремо переглядати інформацію лише про корів, свиней, коней або інші категорії тварин, що зберігаються в базі даних.

Робота алгоритму починається з вибору користувачем необхідного типу тварини у відповідному елементі інтерфейсу. Після цього формується запит на отримання даних, які відповідають обраному значенню. Запит передається до серверної частини системи, де здійснюється пошук записів із відповідним типом тварини.

На сервері отримане значення використовується як критерій відбору. Система аналізує записи, що містяться в базі даних, та формує список об’єктів, які відповідають обраному типу. Після завершення обробки результат повертається до клієнтської частини та відображається користувачеві у вигляді оновленого списку.

Якщо записів із зазначеним типом не знайдено, система повинна повідомити користувача про відсутність результатів. Це дозволяє уникнути непорозумінь під час роботи та забезпечує зрозумілий зворотний зв’язок.

Передбачено також можливість скасування фільтрації. У такому випадку система повторно завантажує повний перелік тварин та відображає всі наявні записи незалежно від їхнього типу.

З точки зору користувача алгоритм є досить простим, однак його реалізація суттєво підвищує зручність роботи із системою. Використання фільтрації скорочує час пошуку необхідної інформації та дозволяє ефективніше працювати навіть зі значними обсягами даних.

Отже, алгоритм фільтрації за типом тварини є важливою складовою функціональності вебсистеми та забезпечує швидкий доступ до потрібної інформації в процесі ведення обліку поголів’я худоби.

Опис діаграми діяльності фільтрації тварин за типом

Діаграма діяльності відображає алгоритм виконання фільтрації тварин за їх типом у системі обліку поголів'я худоби. Вона демонструє послідовність дій, що виконуються під час обробки запиту користувача та формування результуючої вибірки даних.

Початок виконання алгоритму

Процес розпочинається з введення користувачем типу тварини, за яким необхідно виконати пошук. Отримане значення використовується як критерій фільтрації для подальшої обробки даних.

Створення результуючої колекції

Після отримання параметра пошуку створюється порожня результуюча колекція, до якої будуть додаватися об'єкти, що відповідають заданому типу тварини.

Перегляд колекції тварин

Система послідовно переходить до кожного об'єкта із загальної колекції тварин. Для кожного елемента виконується перевірка відповідності його типу значенню, введеному користувачем.

Перевірка умови фільтрації

Для поточного об'єкта виконується перевірка умови відповідності типу тварини заданому критерію. Якщо тип чергового об'єкта збігається із введеним типом, цей об'єкт додається до результуючої колекції. Якщо умова не виконується, система переходить до наступного об'єкта без виконання додаткових дій.

Формування результату

Після завершення перевірки поточного об'єкта виконується перехід до наступного елемента загальної колекції. Процес повторюється доти, доки не буде оброблено всі записи про тварин.

Повернення результатуючої колекції

Після завершення перегляду всієї колекції тварин система повертає сформовану результуючу колекцію, яка містить лише ті об'єкти, що відповідають обраному типу тварини. Отриманий результат передається користувачу для подальшого перегляду та роботи з відфільтрованими даними.

Завершення алгоритму

Після повернення результуючої колекції процес фільтрації завершується.

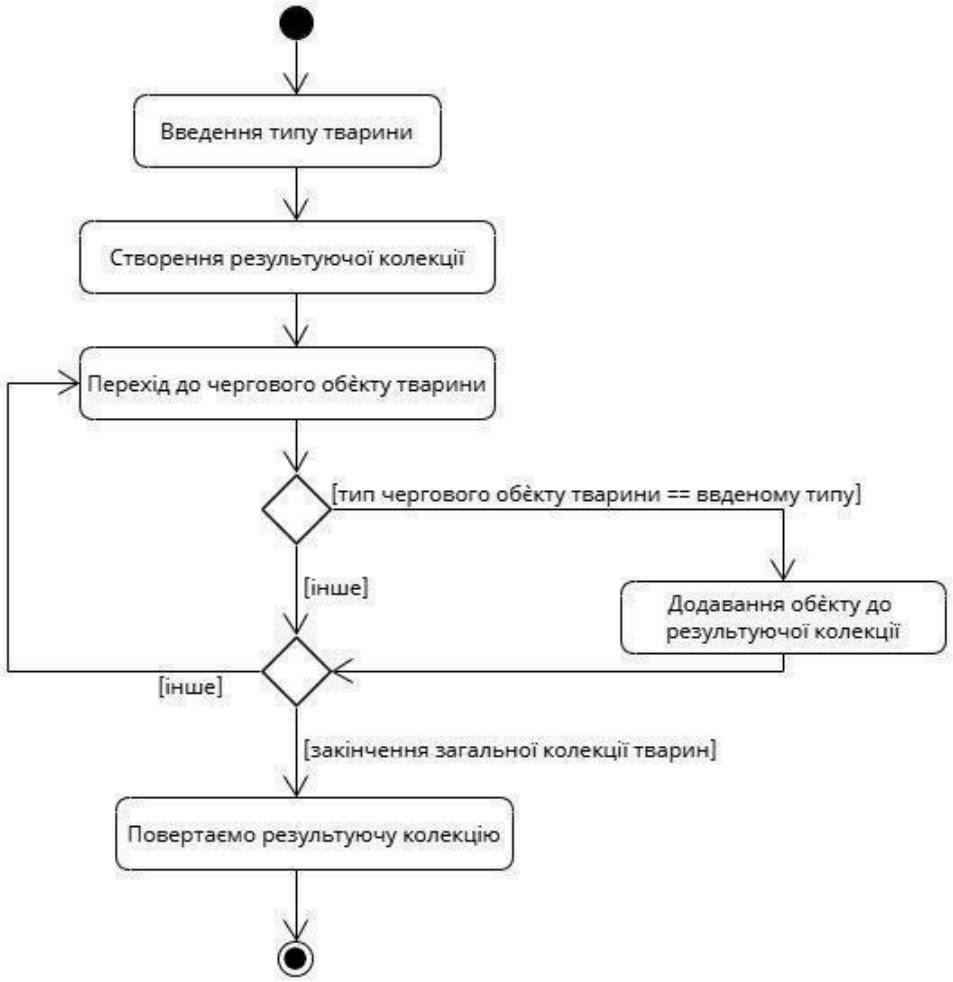


Рис. 2.4 алгоритм фільтрації за типом тварини

Опис діаграми послідовності фільтрації тварин за типом

Діаграма послідовності відображає процес взаємодії між користувачем, контролером, сервісним шаром та репозиторієм під час виконання фільтрації тварин за обраним типом. Діаграма демонструє порядок виклику методів та передачі даних між компонентами системи.

Користувач

Ініціює процес фільтрації, надсилаючи запит FiltrationByAnimalType(animalType), де animalType містить тип тварини, за яким необхідно виконати пошук.

AnimalRestController

Після отримання запиту контролер AnimalRestController викликає однойменний метод FiltrationByAnimalType(animalType) сервісного шару. Контролер не виконує обробку даних самостійно, а лише передає параметри до бізнес-логіки та очікує результат виконання.

AnimalService

Клас AnimalService отримує запит від контролера та виконує основну логіку фільтрації. Для отримання даних сервіс звертається до репозиторію за допомогою виклику методу GetAnimals(). Після отримання списку всіх тварин створюється нова колекція List<Animal>, яка буде містити результати фільтрації.

AnimalRepository

Репозиторій AnimalRepository виконує отримання всіх записів про тварин та повертає колекцію animals до сервісного шару. Репозиторій відповідає лише за доступ до даних і не бере участі у виконанні фільтрації.

List<Animal>

Після створення нової колекції сервіс починає послідовний перегляд усіх елементів списку animals. Для кожного об'єкта Animal перевіряється умова відповідності типу тварини значенню animalType. Якщо умова animal.AnimalType == animalType виконується, поточний об'єкт додається до результуючого списку за допомогою операції Add(animal). Цей процес повторюється для всіх записів колекції.

Повернення результату

Після завершення циклу сформований список AnimalFilterResult повертається з AnimalService до AnimalRestController. Далі контролер передає результат користувачу. У підсумку користувач отримує перелік тварин, тип яких відповідає заданому критерію фільтрації.

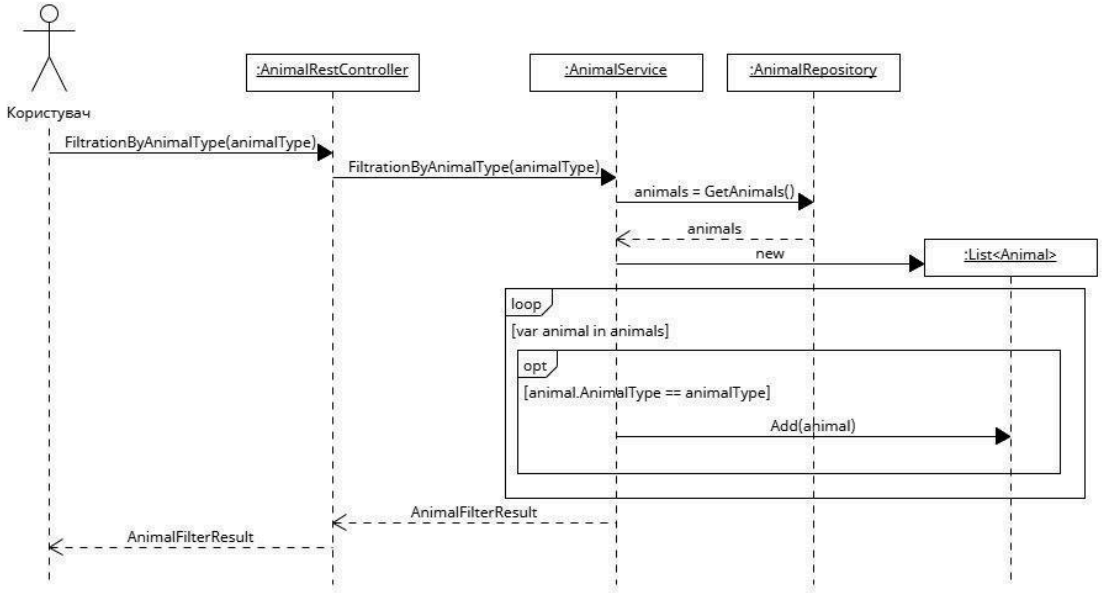


Рис. 2.5 Діаграма послідовності

Опис діаграми класів для реалізації фільтрації тварин за типом

Діаграма класів відображає архітектуру програмного модуля фільтрації тварин за їх типом у вебсистемі обліку поголів'я худоби. Реалізація побудована відповідно до багатошарової архітектури та використовує патерн Repository-Service-Controller.

Клас AnimalRestController

Клас відповідає за обробку HTTP-запитів від користувача.

Атрибут:

animalService : IAnimalService — посилання на сервісний шар.

Методи:

FiltrationByAnimalType(animalType) — виконує запит на отримання тварин певного виду; GetAllAnimals() — повертає список усіх тварин.

Контролер використовує інтерфейс IAnimalService для взаємодії з бізнес-логікою системи.

Інтерфейс IAnimalService

Визначає набір операцій сервісного шару.

Методи:

FiltrationByAnimalType(animalType) GetAllAnimals()

Інтерфейс забезпечує незалежність контролера від конкретної реалізації сервісу.

Клас AnimalService

Реалізує інтерфейс IAnimalService та містить бізнес-логіку фільтрації.

Атрибут:

animalRepository : IAnimalRepository

Методи:

FiltrationByAnimalType(animalType) : List — відбирає тварин за вказаним типом; GetAllAnimals() : List — отримує всі записи про тварин.

Для доступу до даних сервіс використовує інтерфейс репозиторію IAnimalRepository.

Інтерфейс IAnimalRepository Визначає операції доступу до даних.

Метод:

GetAnimals() : List

Інтерфейс приховує деталі роботи з джерелом даних від сервісного шару.

Клас AnimalRepository

Реалізує інтерфейс IAnimalRepository та забезпечує доступ до колекції тварин.

Атрибут:

Animals : List

Метод:

GetAnimals() : List

Саме цей клас взаємодіє з базою даних або іншим джерелом зберігання інформації.

Клас Animal

Представляє сутність тварини.

Атрибути:

ID — унікальний ідентифікатор; AnimalType — тип тварини; AnimalName — назва або кличка; AnimalBirthday — дата народження; HealthStatus — стан здоров'я; Notes — додаткова інформація. Кожен об'єкт цього класу є записом про

окрему тварину.

Клас AnimalType

Містить інформацію про типи тварин.

Атрибути:

ID — унікальний ідентифікатор типу;

TypeName — назва типу тварини.

Зв'язки між класами AnimalRestController використовує (use) інтерфейс IAnimalService. AnimalService реалізує інтерфейс IAnimalService. AnimalService використовує (use) інтерфейс IAnimalRepository. AnimalRepository реалізує інтерфейс IAnimalRepository. Один об'єкт AnimalType може відповідати багатьом об'єктам Animal (зв'язок **1 : ***). AnimalRepository агрегує колекцію об'єктів Animal, що відображено відношенням агрегації. Алгоритм роботи фільтрації Користувач надсилає запит на фільтрацію тварин за певним типом. AnimalRestController викликає метод FiltrationByAnimalType() сервісного шару. AnimalService отримує список усіх тварин через AnimalRepository. Сервіс виконує відбір об'єктів, у яких значення AnimalType відповідає заданому параметру. Відфільтрований список повертається контролеру. Контролер надсилає результат користувачу у вигляді списку тварин.

Опис проєктування бази даних

Діаграма класів відображає архітектуру програмного модуля фільтрації тварин за їх типом у вебсистемі обліку поголів'я худоби. Реалізація побудована відповідно до багатошарової архітектури та використовує патерн Repository-Service-Controller.

Клас AnimalRestController

Клас відповідає за обробку HTTP-запитів від користувача.

Атрибут:

animalService : IAnimalService — посилання на сервісний шар.

Методи:

FiltrationByAnimalType(animalType) — виконує запит на отримання тварин певного виду; GetAllAnimals() — повертає список усіх тварин.

Контролер використовує інтерфейс IAnimalService для взаємодії з бізнес-логікою системи.

Інтерфейс IAnimalService

Визначає набір операцій сервісного шару.

Методи:

FiltrationByAnimalType(animalType) GetAllAnimals()

Інтерфейс забезпечує незалежність контролера від конкретної реалізації сервісу.

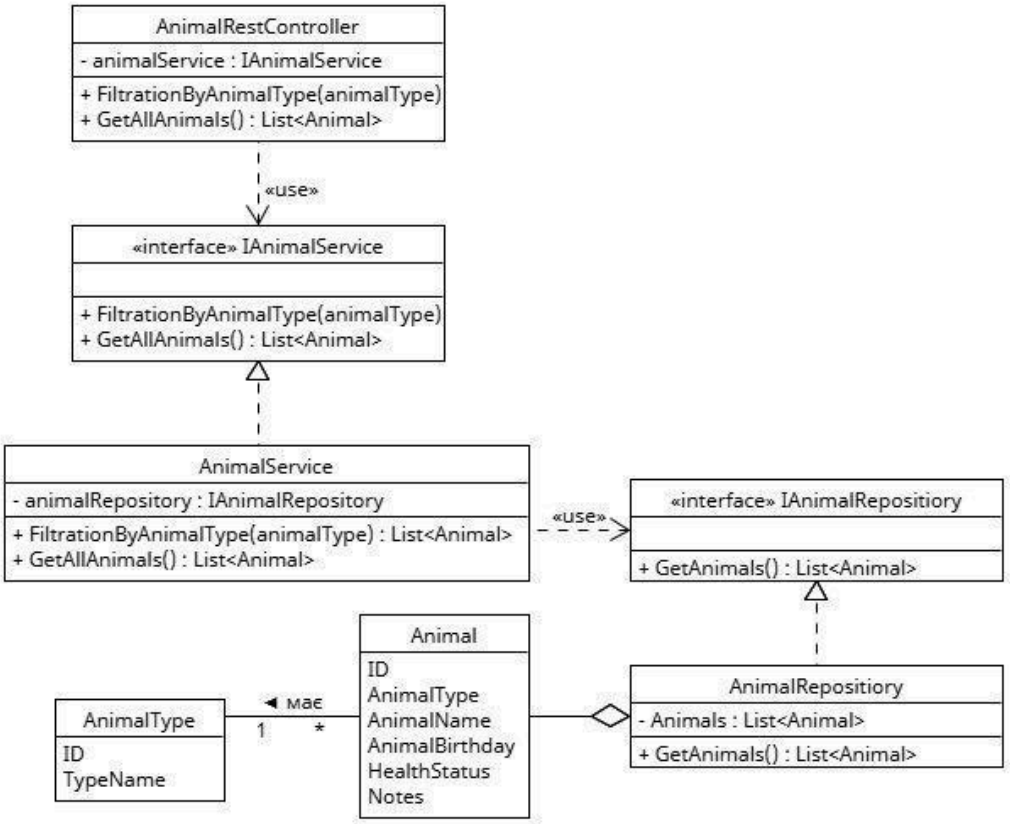


Рис. 2.6 Діаграма класів стосовно фільтрації

Клас AnimalService

Реалізує інтерфейс IAnimalService та містить бізнес-логіку фільтрації.

Атрибут:

animalRepository : IAnimalRepository

Методи:

FiltrationByAnimalType(animalType) : List — відбирає тварин за вказаним

типом; GetAllAnimals() : List — отримує всі записи про тварин.

Для доступу до даних сервіс використовує інтерфейс репозиторію IAnimalRepository.

Інтерфейс IAnimalRepository

Визначає операції доступу до даних.

Метод:

GetAnimals() : List

Інтерфейс приховує деталі роботи з джерелом даних від сервісного шару.

Клас AnimalRepository

Реалізує інтерфейс IAnimalRepository та забезпечує доступ до колекції тварин.

Атрибут:

Animals:List

Метод:

GetAnimals():List

Саме цей клас взаємодіє з базою даних або іншим джерелом зберігання інформації.

Клас Animal

Представляє сутність тварини.

Атрибути:

ID — унікальний ідентифікатор;

AnimalType — тип тварини;

AnimalName — назва або кличка; AnimalBirthday — дата народження; HealthStatus — стан здоров'я; Notes — додаткова інформація.

Кожен об'єкт цього класу є записом про окрему тварину.

Клас AnimalType містить інформацію про типи тварин.

Атрибути:

ID — унікальний ідентифікатор типу; TypeName — назва типу тварини. Зв'язки між класами AnimalRestController використовує (use) інтерфейс IAnimalService. AnimalService реалізує інтерфейс IAnimalService. AnimalService

використовує (use) інтерфейс IAnimalRepository. AnimalRepository реалізує інтерфейс IAnimalRepository. Один об'єкт AnimalType може відповідати багатьом об'єктам Animal (зв'язок **1 : ***). AnimalRepository агрегує колекцію об'єктів Animal, що відображено відношенням агрегації. Алгоритм роботи фільтрації Користувач надсилає запит на фільтрацію тварин за певним типом. AnimalRestController викликає метод FiltrationByAnimalType() сервісного шару. AnimalService отримує список усіх тварин через AnimalRepository. Сервіс виконує відбір об'єктів, у яких значення AnimalType відповідає заданому параметру. Відфільтрований список повертається контролеру. Контролер надсилає результат користувачу у вигляді списку тварин.

2.7 Проєктування інтерфейсу користувача та структури меню

Ефективність роботи будь-якої інформаційної системи значною мірою залежить від якості інтерфейсу користувача. Навіть за наявності широкого функціоналу система може бути незручною у використанні, якщо навігація є складною або необхідні функції важко знайти. Саме тому під час проєктування вебсистеми значну увагу було приділено структурі меню та організації інтерфейсу.

Основною метою проєктування інтерфейсу було забезпечення швидкого доступу до найбільш використовуваних функцій системи. Для цього всі елементи меню були згруповані відповідно до структури предметної області. Такий підхід дозволяє користувачеві швидко орієнтуватися в системі та переходити до потрібного розділу без виконання зайвих дій.

Після проходження авторизації користувач отримує доступ до головного інтерфейсу системи. Основне меню містить розділи, пов’язані з керуванням тваринами, фермами, приміщеннями, пристроями та користувачами. Кожен розділ відкриває окремий модуль, який відповідає за роботу з відповідною групою даних.

Для відображення списків об’єктів використовуються таблиці, що містять основну інформацію про записи. Біля кожного запису передбачено кнопки перегляду, редагування та видалення. Такий підхід є звичним для більшості

вебзастосунків і не потребує додаткового навчання користувачів.

Особлива увага була приділена формам введення даних. Під час створення або редагування записів користувачеві відображаються лише необхідні поля, що спрощує процес заповнення інформації та зменшує ймовірність помилок.

Для підвищення зручності роботи в інтерфейс інтегровано механізми пошуку та фільтрації. Вони розташовуються безпосередньо на сторінках зі списками об’єктів, що дозволяє користувачеві швидко змінювати критерії відображення інформації без переходу до додаткових розділів.

Під час проєктування також враховувалися принципи єдиного стилю оформлення. Однакові елементи керування, схожий зовнішній вигляд сторінок та передбачувана поведінка інтерфейсу дозволяють скоротити час адаптації користувачів до системи.

Оскільки клієнтська частина реалізується за допомогою Angular, структура інтерфейсу побудована на основі компонентного підходу. Кожен функціональний модуль представлений окремим набором компонентів, що спрощує підтримку та подальший розвиток програмного забезпечення.

Таким чином, спроєктований інтерфейс користувача забезпечує зручний доступ до функціональних можливостей системи, підтримує логічну структуру меню та створює комфортні умови для виконання основних операцій обліку поголів’я худоби.

Опис діаграми діяльності навігації користувача в системі

Діаграма діяльності відображає процес взаємодії користувача з основними функціональними модулями вебсистеми обліку поголів'я худоби. Вона демонструє послідовність переходів між меню системи та можливості керування окремими категоріями даних.

Логін користувача

Робота із системою розпочинається з процедури авторизації. Після успішного входу користувач переходить до головного меню, яке є центральною точкою навігації між усіма функціональними модулями системи.

Головне меню

Головне меню забезпечує доступ до управління ролями, тваринами, користувачами, фермами та датчиками. Також з головного меню користувач може завершити роботу із системою шляхом виконання виходу.

Меню щодо ролей

Під час переходу до модуля ролей система завантажує список ролей та відкриває меню щодо ролей. Із цього меню користувач може перейти до меню з управління інформацією щодо ролей для перегляду та редагування відповідних даних. Після завершення роботи виконується повернення до головного меню.

Меню щодо тварин

Під час відкриття модуля тварин система завантажує список тварин та відображає меню щодо тварин. Із цього меню користувач може перейти до меню з управління інформацією щодо тварин, де здійснюється перегляд та редагування відомостей про поголів'я. Також доступний перехід до меню сервісів щодо тварин, яке надає додаткові функції роботи з даними. Після завершення роботи користувач повертається до головного меню.

Меню з управління інформацією щодо користувачів

Для роботи з користувачами система завантажує список користувачів та відкриває меню з управління інформацією щодо користувачів. У цьому модулі здійснюється перегляд, створення, редагування та видалення облікових записів користувачів. Після завершення операцій виконується повернення до головного меню.

Меню щодо ферм

Під час переходу до модуля ферм система завантажує список ферм та відкриває меню щодо ферм. Звідси користувач може перейти до меню з управління інформацією щодо ферм для виконання операцій над даними ферм або до меню сервісів ферм для використання додаткових функціональних можливостей. Після завершення роботи здійснюється повернення до головного меню.

Меню щодо датчиків

Під час відкриття модуля датчиків система завантажує список датчиків та

відображає меню щодо датчиків. Із цього меню доступний перехід до меню з управління інформацією щодо датчиків, де виконується робота з параметрами та характеристиками пристроїв моніторингу. Також користувач може перейти до меню сервісів датчиків для виконання додаткових операцій. Після завершення роботи виконується повернення до головного меню.

Завершення роботи

Після завершення взаємодії з функціональними модулями користувач може виконати вихід із системи через головне меню. Після цього процес роботи із вебсистемою завершується.

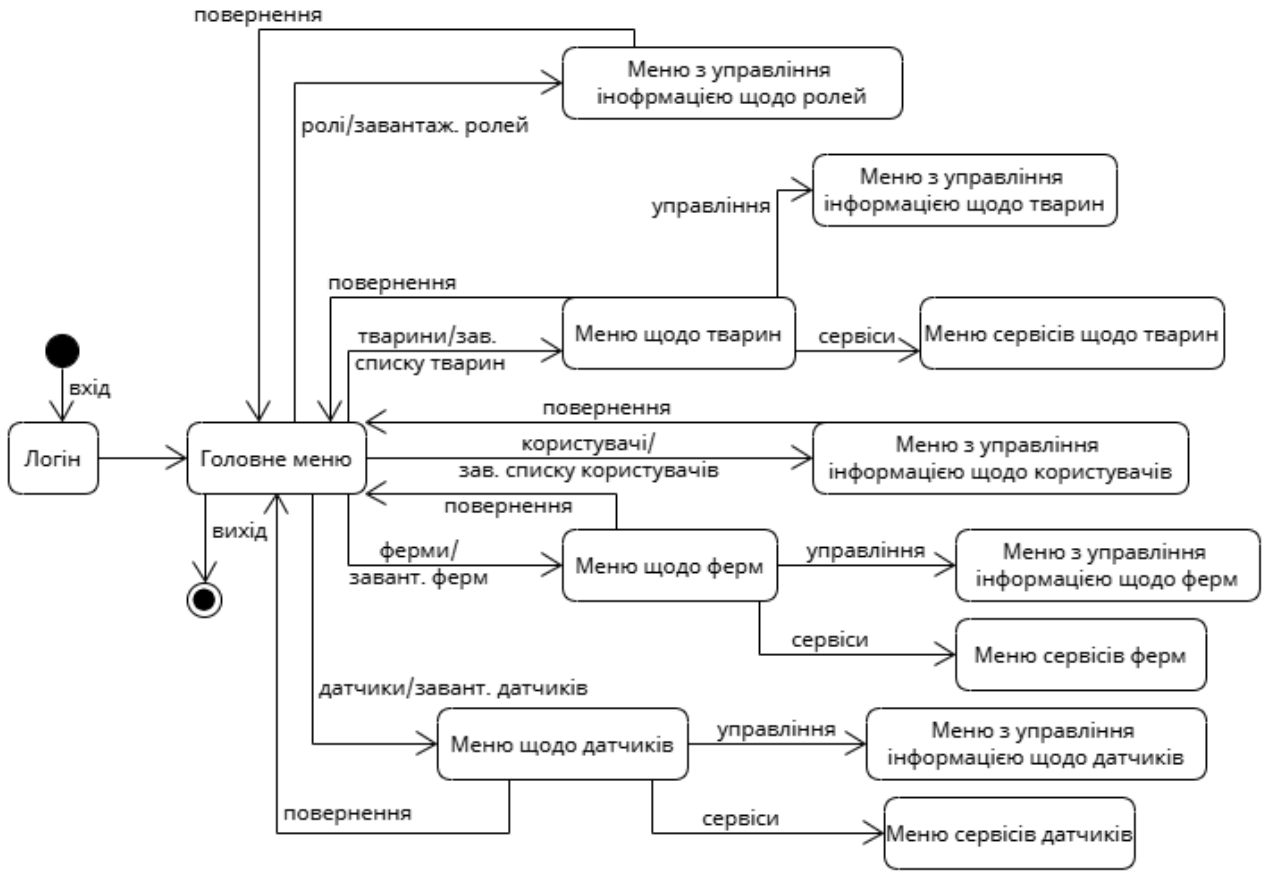


Рис. 2.7 моделювання меню “MyCattle+”

2.8 Проєктування бази даних

База даних призначена для інформаційної вебсистеми обліку поголів'я худоби на сільськогосподарському підприємстві. Вона містить сутності для зберігання інформації про тварин, працівників, ферми, будівлі, датчики та службові записи.

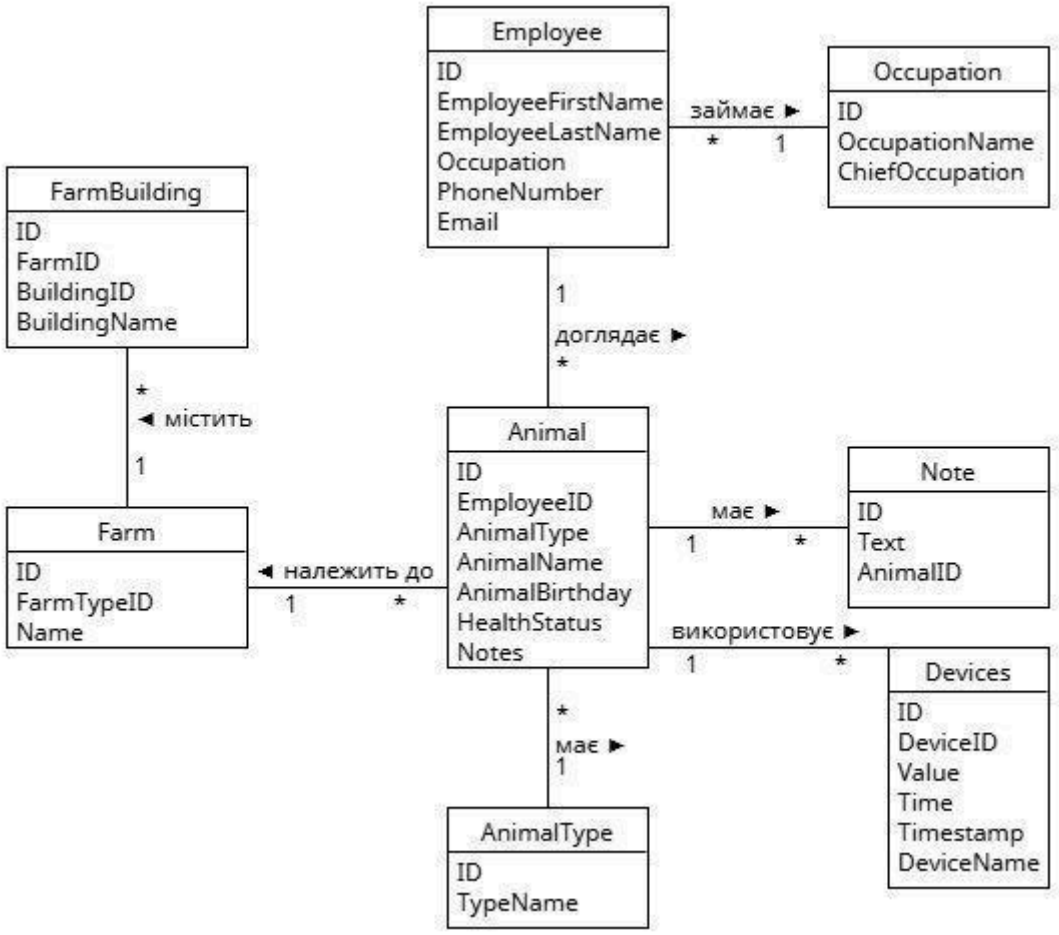


Рис. 2.8 Проєктування бази даних

Таблиця Farm

Містить інформацію про ферми підприємства.

Поля:

ID — унікальний ідентифікатор ферми;

FarmTypeID — посилання на тип ферми;

Name — назва ферми.

Зв'язок:

одна ферма може містити багато будівель (FarmBuilding).

Таблиця FarmBuilding

Містить інформацію про будівлі, що належать фермам.

Поля:

ID — унікальний ідентифікатор будівлі;

FarmID — ідентифікатор ферми;

BuildingID — ідентифікатор типу будівлі;

BuildingName — назва будівлі.

Зв'язок:

кожна будівля належить одній фермі;

одна ферма може містити багато будівель.

Таблиця Occupation

Зберігає перелік посад працівників.

Поля:

ID — унікальний ідентифікатор посади;

OccupationName — назва посади;

ChiefOccupation — ознака керівної посади.

Зв'язок:

одна посада може бути призначена багатьом працівникам.

Таблиця Employee

Містить інформацію про працівників підприємства.

Поля:

ID — унікальний ідентифікатор працівника;

EmployeeFirstName — ім'я;

EmployeeLastName — прізвище;

Occupation — посада працівника;

PhoneNumber — номер телефону;

Email — адреса електронної пошти.

Зв'язки:

кожен працівник займає одну посаду (Occupation);

один працівник може відповідати за багато тварин (Animal).

Таблиця AnimalType

Містить класифікацію видів тварин.

Поля:

ID — унікальний ідентифікатор виду;

TypeName — назва виду тварини.

Зв'язок:

один тип може належати багатьом тваринам.

Таблиця Animal

Є центральною сутністю системи та містить дані про поголів'я.

Поля:

ID — унікальний ідентифікатор тварини;

EmployeeID — відповідальний працівник;

AnimalType — вид тварини;

AnimalName — кличка або назва;

AnimalBirthday — дата народження;

HealthStatus — стан здоров'я;

Notes — додаткові відомості.

Зв'язки:

кожна тварина належить до одного виду (AnimalType);

за кожною твариною закріплений один працівник (Employee);

одна тварина може мати багато приміток (Note);

одна тварина може використовувати багато пристроїв (Devices).

Таблиця Note

Призначена для зберігання службових записів щодо тварин.

Поля:

ID — унікальний ідентифікатор запису;

Text — текст примітки;

AnimalID — ідентифікатор тварини.

Зв'язок:

одна тварина може мати багато приміток;

кожна примітка належить одній тварині.

Таблиця Devices

Містить дані, отримані від датчиків та пристроїв моніторингу.

Поля:

ID — унікальний ідентифікатор запису;

DeviceID — ідентифікатор пристрою;

Value — зафіксоване значення;

Time — час вимірювання;

Timestamp — дата та час запису;

DeviceName — назва пристрою.

Зв'язок:

один пристрій може бути пов'язаний з багатьма тваринами;

одна тварина може використовувати декілька пристроїв для моніторингу стану та збору даних.

Загальна характеристика

Структура бази даних реалізує облік тварин, відповідальних працівників, ферм та будівель, а також забезпечує зберігання даних від датчиків і журналу подій. Центральним елементом моделі є сутність Animal, навколо якої побудовані зв'язки з працівниками, видами тварин, пристроями моніторингу та службовими примітками. Така структура дозволяє організувати централізований контроль поголів'я та автоматизувати процеси ведення обліку на сільськогосподарському підприємстві.

Висновки до розділу 2

Результати другого розділу створюють основу для програмної реалізації вебсистеми. Запропоновані проєктні рішення дозволяють перейти до розроблення клієнтської та серверної частин застосунку, реалізації бази даних, механізмів роботи з користувачами та функцій обліку тварин.

У розділі також розглянуто моделювання прецедентів використання,

побудову діаграми предметної галузі та логічної архітектури системи. Окрему увагу приділено алгоритму фільтрації тварин за типом, діаграмі послідовності та діаграмі класів, що описують реалізацію відповідного функціонального модуля. Додатково було описано структуру бази даних, зв'язки між основними сутностями та принципи захисту інформації.

У другому розділі було виконано проєктування інформаційної вебсистеми обліку поголів'я худоби. Сформовано функціональні та нефункціональні вимоги до системи, визначено основні можливості, які повинні бути реалізовані під час розроблення програмного продукту. Обґрунтовано вибір технологій Angular, Spring Boot та MySQL, які забезпечують побудову клієнтської частини, серверної логіки та рівня зберігання даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ВЕБСИСТЕМИ ОБЛІКУ ПОГОЛІВ'Я ХУДОБИ

3.1 Реалізація клієнтської частини вебсистеми

Клієнтська частина вебсистеми реалізована за допомогою фреймворку Angular 17.3. Використання Angular дозволило створити односторінковий вебзастосунок, який забезпечує швидку навігацію між сторінками та зручну взаємодію користувача із системою.

Основна структура клієнтської частини розміщена в каталозі src/app та поділена на окремі функціональні модулі. Для реалізації сторінок використовуються компоненти, згруповані в каталозі pages. У системі передбачені сторінки авторизації, головна сторінка, сторінки роботи з таблицями даних та допоміжні екрани для керування об'єктами предметної області.

Для взаємодії із серверною частиною використовуються Angular-сервіси. Вони відповідають за надсилання HTTP-запитів до REST API та отримання даних для подальшого відображення в інтерфейсі. Окремо реалізовано сервіси авторизації, роботи з таблицями даних та отримання метаданих інтерфейсу.

Важливою особливістю реалізації є використання механізму динамічного формування частини інтерфейсу. Серверна частина надає інформацію про доступні секції, подання, поля та зв'язки між ними, що дозволяє формувати окремі елементи адміністративного інтерфейсу без жорсткого програмування кожної сторінки окремо.

Для організації навігації використовується Angular Router. Налаштування маршрутів дозволяє користувачу переходити між сторінками без перезавантаження браузера, що позитивно впливає на швидкодію та зручність роботи із системою.

З метою захисту окремих сторінок застосовано механізм guards. Перед відкриттям сторінки система перевіряє стан авторизації користувача та за

необхідності обмежує доступ до функціоналу.

Таким чином, клієнтська частина забезпечує відображення інформації, взаємодію користувача із системою та обмін даними із серверною частиною.

3.2 Реалізація серверної частини та взаємодії з базою даних

Серверна частина реалізована на базі Spring Boot 3.5 та побудована за багаторівневою архітектурою. Основними складовими є REST-контролери, сервісний рівень, репозиторії доступу до даних, моделі предметної області та модулі безпеки.

Рівень REST-контролерів відповідає за приймання HTTP-запитів від клієнтської частини та формування відповідей у форматі JSON. Для роботи з окремими функціональними напрямами створено контролери AnimalRestController, FarmRestController, UserRestController, AdminTableRestController, UiMetaRestController, BuildingRestController та DeviceRestController.

Основна бізнес-логіка реалізована у сервісному шарі DataSource. До його складу входять сервіси AnimalService, FarmService, BuildingService, DeviceDataService, UsersService, UserRolesService, UserSessionsService, UiMetaService та інші компоненти. Саме на цьому рівні виконуються перевірка вхідних даних, оброблення запитів та взаємодія з репозиторіями.

Для роботи з базою даних використовується рівень репозиторіїв, побудований на основі Spring Data JPA. Репозиторії забезпечують виконання операцій створення, пошуку, оновлення та видалення даних. У системі реалізовано AnimalRepository, FarmRepository, UserRepository, FarmBuildingRepository, DeviceDataRepository, UserRolesRepository та інші репозиторії.

Зберігання даних здійснюється в базі даних MySQL. Структура бази побудована відповідно до моделі предметної області та підтримує зв'язки між усіма основними сутностями системи.

Обмін даними між клієнтом і сервером здійснюється через REST API. Клієнт надсилає HTTP-запит до відповідного контролера, після чого запит передається до сервісного шару. Після виконання необхідних операцій система звертається до бази даних і повертає результат користувачеві.

Таким чином, серверна частина виступає центральним елементом вебсистеми та забезпечує узгоджену роботу всіх її компонентів.

3.3 Реалізація модулів обліку тварин, елементів ферми та користувачів

У процесі розроблення було реалізовано кілька функціональних модулів, кожен із яких відповідає за окрему групу завдань.

Реалізовано повноцінний модуль обліку тварин. Він забезпечує зберігання та опрацювання інформації про поголів'я худоби. Для кожної тварини можуть зберігатися ідентифікаційні дані, тип, порода, дата народження та інші характеристики.

Модуль керування фермами забезпечує роботу з інформацією про фермерські господарства та їхню структуру. Через відповідний REST-контролер реалізовано отримання списку ферм, створення нових записів і перегляд інформації про окремі ферми.

Окремо реалізовано модуль користувачів. Він забезпечує роботу з обліковими записами, ролями та сесіями користувачів. У межах цього модуля підтримуються операції авторизації, отримання інформації про поточного користувача та робота з ролями доступу.

Важливою особливістю системи є модуль адміністративних таблиць. Його реалізація дозволяє працювати з різними наборами даних через універсальний механізм відображення таблиць. Для цього використовується контролер AdminTableRestController, який підтримує операції створення, оновлення, перегляду та видалення записів.

Ще одним спеціалізованим модулем є система метаданих інтерфейсу. Вона забезпечує отримання інформації про секції, подання, поля та зв'язки між ними.

Завдяки цьому частина інтерфейсу формується динамічно на основі даних, отриманих від сервера.

До основних сутностей системи належать Animal, Farm, Building, Device, DeviceData та User. Для реалізації зв'язків між ними використовуються додаткові сутності FarmBuilding, BuildingAnimal, UserFarms, UserRole та UserSession.

Таким чином, реалізовані модулі забезпечують роботу з основними об'єктами предметної області та формують функціональну основу вебсистеми.

3.4 Реалізація механізмів доступу та роботи з даними

Для забезпечення безпеки вебсистеми реалізовано механізм автентифікації користувачів на основі JWT-токенів. Такий підхід дозволяє перевіряти особу користувача під час кожного звернення до серверної частини без необхідності постійного зберігання стану сесії на сервері.

Модуль безпеки містить класи SecurityConfig та JwtAuthenticationFilter, які відповідають за налаштування правил доступу та перевірку отриманих токенів. Для оброблення виняткових ситуацій використовується GlobalExceptionHandler.

Після успішної авторизації система створює користувацьку сесію та надає доступ до функціональних можливостей відповідно до ролі користувача. Інформація про ролі та сесії зберігається у відповідних сутностях UserRole і UserSession.

Для роботи з даними використовується технологія Spring Data JPA. Вона дозволяє виконувати операції доступу до бази даних через репозиторії без необхідності створення великої кількості SQL-запитів вручну.

Операції створення, читання, оновлення та видалення записів виконуються через сервісний шар. Це дозволяє централізувати бізнес-логіку та уникнути дублювання коду в контролерах.

Завдяки такій організації забезпечується безпечна та стабільна робота системи навіть у разі збільшення кількості користувачів і обсягу даних.

3.5 Реалізація інтерфейсу користувача та основних екранних форм

Інтерфейс користувача реалізований у вигляді набору сторінок і форм, призначених для виконання основних операцій із даними. Головною метою під час розроблення інтерфейсу було забезпечення простоти використання та швидкого доступу до функціональних можливостей системи.

Після авторизації користувач отримує доступ до головної сторінки, з якої може перейти до необхідного розділу системи. Навігація побудована таким чином, щоб користувач міг швидко знаходити потрібні дані та виконувати типові операції.

Для відображення інформації використовуються таблиці даних. Кожна таблиця містить перелік записів і набір дій для роботи з ними. Користувач може переглядати інформацію, створювати нові записи, редагувати існуючі або видаляти непотрібні дані.

Для введення інформації використовуються окремі форми. Під час заповнення форми виконується перевірка коректності введених даних, що дозволяє зменшити кількість помилок і забезпечити цілісність інформації в базі даних.

Окремо реалізовано механізми пошуку та фільтрації, які дозволяють швидко знаходити потрібні записи серед великої кількості даних. Це особливо важливо під час роботи з інформацією про тварин і ферми.

Завдяки використанню Angular усі сторінки мають єдину структуру та спільні елементи керування, що забезпечує зручність використання та цілісність інтерфейсу.

3.6 Реалізація контролю даних та оброблення помилок

Таким чином, реалізована взаємодія між клієнтською частиною, серверною логікою та базою даних забезпечує стабільну роботу вебсистеми, спрощує підтримку програмного коду та створює умови для подальшого розширення

функціональності.

Важливою частиною реалізації є підтримка єдиного підходу до структури запитів. Це спрощує розроблення нових модулів, оскільки кожен новий компонент може використовувати вже сформований принцип взаємодії. Наприклад, модулі тварин, ферм, пристроїв і користувачів можуть мати схожу структуру запитів і відповідей.

Реалізація взаємодії між рівнями системи також передбачає оброблення помилок. Помилки можуть виникати через некоректні дані, відсутність доступу, проблеми з підключенням до бази або неправильний запит. У таких випадках сервер повинен повертати зрозумілу відповідь, а клієнтська частина має відобразити її користувачеві.

Під час реалізації взаємодії з базою даних важливо забезпечити коректність виконання операцій. Якщо користувач створює новий запис, сервер повинен перевірити отримані дані та лише після цього передати їх до бази. Якщо користувач редагує запис, система повинна переконатися, що такий об'єкт існує. Якщо запис не знайдено, користувач має отримати відповідне повідомлення.

Для передачі даних між клієнтом і сервером використовуються об'єкти, що містять необхідну інформацію про сутності системи. Наприклад, під час роботи з тваринами передаються дані про ідентифікатор, тип, ім'я, дату народження та стан здоров'я. Така структура дозволяє уникнути передачі зайвої інформації та забезпечує більш зрозумілу взаємодію між компонентами.

Такий підхід дозволяє зробити код більш зрозумілим і підтримуваним. Якщо в майбутньому потрібно змінити логіку фільтрації, це можна зробити на рівні сервісу без зміни контролера або інтерфейсу користувача. Якщо потрібно змінити спосіб отримання даних, зміни можуть бути внесені на рівні репозиторію.

Серверна частина системи побудована за принципом поділу на контролери, сервіси, репозиторії та моделі. Контролери приймають запити від клієнта та передають їх далі на рівень бізнес-логіки. Сервіси виконують основну обробку даних, перевіряють умови виконання операцій та координують роботу з репозиторіями. Репозиторії забезпечують доступ до бази даних і виконують

операції читання або запису.

Клієнтська частина надсилає запити до серверної частини за допомогою HTTP. Кожен запит відповідає певній операції, наприклад отриманню списку тварин, створенню нового запису, редагуванню інформації або видаленню об'єкта. Сервер обробляє запит, звертається до відповідного сервісу та після виконання операції повертає результат у форматі, зручному для використання на фронтенді.

Під час програмної реалізації вебсистеми важливо забезпечити узгоджену взаємодію між клієнтською частиною, серверною частиною та базою даних. У розробленій системі клієнтський рівень відповідає за відображення даних і приймання дій користувача, серверний рівень виконує бізнес-логіку, а база даних забезпечує зберігання інформації. Такий поділ дозволяє зменшити складність програмного продукту та спростити його супровід.

Висновки до розділу 3

Результати розділу підтверджують, що обраний технологічний стек та архітектурні рішення дозволяють реалізувати основні функції вебсистеми. Створена програмна структура забезпечує роботу з даними, виконання облікових операцій і взаємодію користувача із системою через зрозумілий вебінтерфейс.

У межах реалізації було використано багаторівневу структуру, що включає REST-контролери, сервісний рівень, репозиторії та моделі даних. Такий підхід дозволив розмежувати відповідальність між компонентами системи та забезпечити зручність подальшого супроводу. Окремо описано механізми взаємодії між рівнями системи, контроль введених даних, оброблення помилок і підготовку до подальшого розширення функціональності.

У третьому розділі було розглянуто програмну реалізацію інформаційної вебсистеми обліку поголів'я худоби. Описано реалізацію клієнтської частини на основі Angular, серверної частини з використанням Spring Boot, а також взаємодію з базою даних MySQL. Розглянуто структуру основних модулів, які забезпечують роботу з тваринами, фермами, приміщеннями, пристроями та користувачами.

Таким чином, реалізація контролю даних і оброблення помилок є важливою частиною програмного забезпечення. Вона підвищує надійність системи, покращує зручність використання та забезпечує стабільну роботу основних функціональних модулів.

Контроль даних також важливий для підтримки цілісності бази даних. Наприклад, система не повинна дозволяти створення посилань на неіснуючі об'єкти або видалення записів, які використовуються в інших частинах системи. Для цього застосовуються перевірки на рівні бізнес-логіки та обмеження на рівні бази даних.

У процесі подальшого розвитку системи доцільно реалізувати журналювання помилок і дій користувачів. Це дозволить адміністраторам аналізувати проблемні ситуації та швидше виявляти причини некоректної роботи. Для інформаційної системи, що використовується в господарстві, така можливість є корисною, оскільки вона підвищує контрольованість програмного продукту.

Окремо слід враховувати помилки доступу. Якщо користувач не має прав на виконання певної операції, система повинна заблокувати таку дію. Це стосується видалення записів, зміни прав користувачів або доступу до службових розділів системи. Такий підхід підвищує безпеку та знижує ризик випадкових змін.

Оброблення помилок реалізується таким чином, щоб користувач отримував зрозуміле повідомлення. Якщо операція не може бути виконана, система не повинна просто припиняти роботу або показувати технічну інформацію. Користувач має бачити пояснення, яке допоможе виправити помилку або повторити дію.

Серверна частина виконує додаткову перевірку отриманих даних. Це необхідно, оскільки перевірка лише на клієнтському рівні не є достатньою. Сервер повинен самостійно переконатися, що дані відповідають вимогам системи та можуть бути безпечно збережені у базі даних.

Клієнтська частина може виконувати первинну перевірку ще до надсилання даних на сервер. Це дозволяє швидко повідомити користувача про помилку та не створювати зайве навантаження на сервер. Наприклад, якщо поле залишено

порожнім, система може відразу показати відповідне повідомлення.

Для зменшення кількості помилок використовується перевірка обов'язкових полів. Наприклад, під час створення запису про тварину користувач повинен вказати основні характеристики, без яких облік буде неповним. До таких даних можуть належати тип тварини, дата народження, стан здоров'я або інші параметри, передбачені структурою системи.

Під час розроблення вебсистеми обліку поголів'я худоби важливо забезпечити не лише виконання основних операцій, а й контроль правильності введення та оброблення даних. У системі передбачається робота з великою кількістю записів, тому помилки у введених значеннях можуть призвести до неточностей у подальшому обліку.

4 ТЕСТУВАННЯ, АПРОБАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1 Організація та методика тестування системи

Після завершення розроблення вебсистеми обліку поголів'я худоби було проведено тестування її основних функціональних компонентів. Метою тестування стала перевірка коректності роботи клієнтської та серверної частин застосунку, взаємодії з базою даних, механізмів авторизації користувачів та реалізованих REST API.

Тестування проводилося в середовищі локального розгортання системи. Клієнтська частина запускалася за допомогою Angular 17, серверна частина працювала на платформі Spring Boot 3.5 із використанням бази даних MySQL. Для перевірки роботи REST API використовувалися вбудовані засоби браузера та інструменти надсилання HTTP-запитів.

Основна увага приділялася перевірці таких компонентів:

авторизація користувачів;

робота REST API;

модуль керування фермами;

модуль керування користувачами;

отримання адміністративних таблиць;

отримання метаданих інтерфейсу;

взаємодія з базою даних;

механізми навігації клієнтської частини.

Під час тестування використовувалися сценарії, що моделюють типові дії користувача. Для кожного сценарію визначався очікуваний результат та перевірявся фактичний результат роботи системи.

Таблиця 4.1

Основні тестові сценарії

Тестовий сценарій	Очікуваний результат
Авторизація користувача	Успішний вхід до системи
Отримання інформації про користувача	Повернення даних користувача
Отримання списку ферм	Відображення інформації про ферми
Створення нового запису ферми	Збереження запису в БД
Редагування інформації про ферму	Оновлення даних у БД
Отримання метаданих інтерфейсу	Повернення структури UI
Отримання адміністративної таблиці	Коректне відображення даних
Перевірка JWT-автентифікації	Доступ лише для авторизованих користувачів

Проведене тестування дозволило оцінити як окремі функції системи, так і їх спільну роботу в межах єдиного програмного середовища.

4.2 Перевірка функціонування основних модулів вебсистеми

Першим етапом стала перевірка роботи модуля авторизації користувачів. Під час тестування було підтверджено коректність оброблення облікових даних та формування JWT-токена після успішної автентифікації. Також перевірено обмеження доступу до захищених ресурсів у разі відсутності авторизації.

Наступним етапом стала перевірка модуля керування фермами. Було протестовано операції отримання списку ферм, створення нового запису, перегляду інформації та редагування існуючих даних. Усі перевірені операції виконувалися коректно та супроводжувалися відповідним оновленням інформації в базі даних.

Окремо перевірявся модуль користувачів. Під час тестування було підтверджено коректне отримання інформації про користувачів, роботу механізмів ролей та взаємодію із сервісом користувацьких сесій.

Особливу увагу приділено модулю адміністративних таблиць. Перевірка показала коректне отримання даних через AdminTableRestController та їх подальше відображення в клієнтському інтерфейсі. Даний механізм дозволяє

працювати з різними типами даних через універсальний інтерфейс без необхідності створення окремої сторінки для кожної сутності.

Також було протестовано модуль метаданих інтерфейсу. Через UiMetaRestController система отримує інформацію про доступні секції, поля, зв'язки та структуру інтерфейсу. Результати перевірки підтвердили правильність формування та передачі цих даних між сервером і клієнтом.

Під час перевірки роботи бази даних було підтверджено коректне збереження інформації, виконання операцій пошуку та оновлення записів через репозиторії Spring Data JPA.

У результаті тестування основних модулів критичних помилок, які б унеможливлювали роботу системи, виявлено не було.

4.3 Аналіз результатів тестування

За результатами проведених перевірок встановлено, що реалізована вебсистема забезпечує коректне виконання основних функцій, передбачених під час проєктування.

Усі перевірені REST API повертали коректні відповіді та забезпечували взаємодію між клієнтською та серверною частинами застосунку. Також підтверджено працездатність механізму авторизації на основі JWT-токенів та коректність роботи системи контролю доступу.

Позитивні результати отримано під час перевірки модулів роботи з фермами, користувачами та адміністративними таблицями. Усі зміни даних успішно зберігалися в базі даних та коректно відображалися в інтерфейсі користувача.

Таблиця 4.2

Результати тестування функціональних можливостей системи

Функція	Результат
Авторизація користувачів	Успішно
Отримання інформації про користувача	Успішно
Взаємодія з базою даних	Успішно
REST API	Успішно
Перевірка доступу за JWT	Успішно
Робота з адміністративними таблицями	Успішно
Отримання UI-метаданих	Успішно

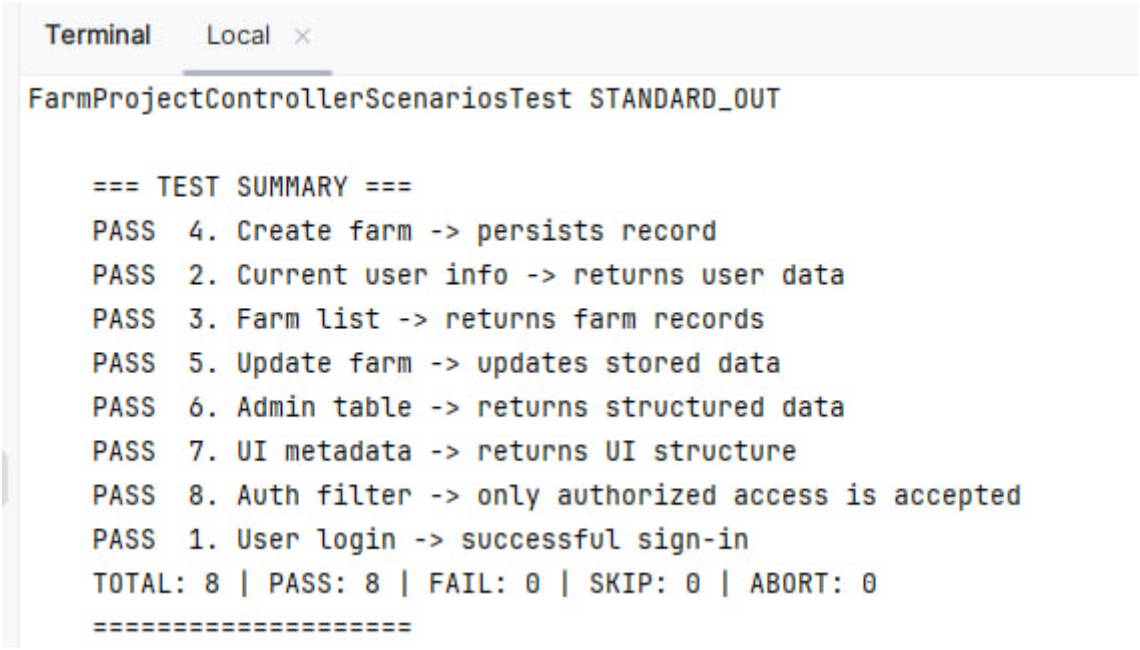


Рис. 4.2 Результати програмного тестування основних модулів

Отримані результати свідчать про працездатність розробленої системи та відповідність її основних компонентів поставленим вимогам.

Водночас проведене тестування дозволило визначити напрями подальшого розвитку проєкту. До них належать розширення функціональності модулів Animal, Building та Device, удосконалення механізмів фільтрації, а також розширення аналітичних можливостей системи.

4.4 Апробація результатів дослідження

Апробація результатів дослідження проводилася шляхом демонстрації роботи розробленої вебсистеми та аналізу реалізованих функціональних можливостей.

Під час апробації було продемонстровано роботу клієнтської частини на базі Angular, взаємодію із серверною частиною Spring Boot та обмін даними через REST API. Також було показано процес авторизації користувача, отримання інформації про ферми та роботу адміністративного інтерфейсу.

Особливий інтерес становив механізм динамічного формування інтерфейсу на основі метаданих, отриманих із сервера. Такий підхід дозволяє спростити подальше розширення системи та зменшити обсяг дублювання програмного коду.

Практична цінність розробленої системи полягає в тому, що вона забезпечує централізоване зберігання інформації про основні об'єкти фермерського господарства та надає єдиний інтерфейс для роботи з ними.

Результати апробації підтвердили можливість використання обраної архітектури та технологічного стеку для створення інформаційних систем аграрного спрямування. Крім того, реалізовані рішення можуть бути використані як основа для подальшого розвитку системи та розширення її функціональних можливостей.

Висновки до розділу 4

Проведена апробація дозволила оцінити практичну придатність розробленої вебсистеми. Отримані результати свідчать, що програмний продукт може використовуватися для впорядкування облікових процесів у фермерському господарстві та має потенціал для подальшого розвитку, зокрема шляхом додавання аналітичних функцій, журналу подій та інтеграції з пристроями моніторингу.

Результати тестування показали, що система коректно виконує основні операції з даними, забезпечує відображення інформації через вебінтерфейс та підтримує сценарії, передбачені під час проєктування. Перевірка фільтрації підтвердила можливість швидкого пошуку тварин за типом, а тестування авторизації показало правильність роботи механізму доступу користувачів.

У четвертому розділі було проведено тестування, апробацію та аналіз результатів роботи інформаційної вебсистеми обліку поголів'я худоби. Перевірено роботу основних функціональних модулів, зокрема модулів обліку тварин, ферм, користувачів, авторизації, пошуку та фільтрації. Також розглянуто взаємодію між клієнтською частиною, серверною логікою та базою даних.

Таким чином, результати тестування підтвердили працездатність основних функціональних можливостей вебсистеми та показали, що запропоноване рішення може бути використане як основа для подальшого розвитку інформаційної системи фермерського господарства.

Окремо слід зазначити, що тестування також дозволило визначити напрями подальшого вдосконалення. У майбутньому до системи можуть бути додані розширені звіти, журнал подій, аналітичні модулі, автоматичні сповіщення та більш глибока інтеграція з IoT-пристроями.

Практичне значення результатів тестування полягає у підтвердженні того, що розроблена система може використовуватися для автоматизації базових облікових процесів. Вона забезпечує централізоване зберігання інформації, доступ через вебінтерфейс, роботу з основними сутностями та виконання операцій над даними.

Під час тестування взаємодії з базою даних перевірялося, чи коректно зберігаються та оновлюються записи. Окремо аналізувалося, чи відображаються зміни на клієнтській частині після виконання операцій. Це дозволяє переконатися, що між фронтендом, бекендом і базою даних існує узгоджена взаємодія.

Також важливою була перевірка авторизації користувачів. Система повинна надавати доступ лише тим особам, які мають відповідні облікові дані. Це дозволяє захистити внутрішню інформацію та забезпечити контрольовану роботу з даними.

У разі неправильного введення даних користувач не повинен отримати доступ до захищених розділів.

Перевірка механізму фільтрації показує, що система може відбирати записи відповідно до заданого типу тварини. Це важливо для роботи з великими наборами даних, оскільки користувачеві не потрібно переглядати повний список вручну. Завдяки фільтрації підвищується швидкість пошуку інформації та зручність роботи з таблицями.

Одним із ключових результатів тестування є підтвердження коректності виконання операцій створення, перегляду, редагування та видалення записів. Такі операції складають основу функціональності системи. Якщо вони працюють стабільно, користувач може виконувати основні облікові дії без необхідності застосування сторонніх інструментів.

Під час тестування особлива увага приділялася модулям, які безпосередньо пов'язані з основними задачами системи. До них належать модуль обліку тварин, модуль роботи з фермами, модуль користувачів, механізми авторизації, пошуку та фільтрації. Перевірка цих компонентів є важливою, оскільки саме вони визначають практичну корисність вебсистеми.

Після виконання основних перевірок функціональних можливостей системи доцільно провести додатковий аналіз отриманих результатів. Такий аналіз дозволяє не лише встановити факт працездатності окремих модулів, а й оцінити загальну готовність продукту до практичного використання.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання автоматизації обліку поголів'я худоби шляхом розроблення інформаційної вебсистеми для сільськогосподарського підприємства.

У процесі виконання роботи було проаналізовано предметну область фермерського господарства та особливості ведення обліку тварин. Проведено огляд сучасних інформаційних систем аграрного спрямування, виконано порівняльний аналіз існуючих аналогів та визначено їх переваги й недоліки. За результатами аналізу обґрунтовано доцільність створення власної вебсистеми, орієнтованої на централізоване зберігання та оброблення інформації про поголів'я худоби, ферми, приміщення, пристрої та користувачів.

На етапі проєктування сформовано функціональні та нефункціональні вимоги до системи, обґрунтовано вибір програмних засобів реалізації, побудовано діаграму прецедентів використання, діаграму предметної галузі та логічну архітектуру застосунку. Також спроєктовано алгоритм фільтрації тварин за типом і структуру інтерфейсу користувача.

У результаті програмної реалізації створено клієнтську частину вебсистеми на основі Angular та серверну частину із використанням Spring Boot. Для зберігання даних використано систему керування базами даних MySQL. Реалізовано REST API для взаємодії між компонентами системи, механізм авторизації користувачів на основі JWT-токенів, а також модулі роботи з тваринами, фермами, користувачами, адміністративними таблицями та метаданими інтерфейсу.

Проведене тестування підтвердило працездатність основних функціональних можливостей системи, коректність взаємодії клієнтської та серверної частин, стабільність роботи REST API та механізмів доступу до даних. Результати перевірки показали, що розроблена вебсистема успішно виконує поставлені завдання та забезпечує зручне керування інформацією в межах

фермерського господарства.

Практичне значення роботи полягає у створенні вебсистеми, яка може використовуватися для автоматизації процесів обліку поголів'я худоби, зберігання інформації про елементи ферми та організації централізованого доступу до даних. Запропонована архітектура дозволяє надалі розширювати функціональність системи, зокрема шляхом розвитку аналітичних модулів, удосконалення механізмів моніторингу та інтеграції з IoT-пристроями.

Таким чином, мету кваліфікаційної роботи досягнуто, а поставлені завдання виконано в повному обсязі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Atzori L., Iera A., Morabito G. The Internet of Things: A Survey. Computer Networks. 2010. Vol. 54, No. 15. P. 2787-2805. URL: https://www.sciencedirect.com/science/article/pii/S1389128610001568 (дата звернення: 31.05.2026).
2. Alreshidi E. Smart Sustainable Agriculture Solution Underpinned by Internet of Things (IoT) and Artificial Intelligence (AI). 2019. URL: https://www.researchgate.net/publication/336611233 (дата звернення: 31.05.2026).
3. Hasan M. M., Islam M. U., Sadeq M. J. Towards Technological Adaptation of Advanced Farming through AI, IoT and Robotics: A Comprehensive Overview. 2022. URL: https://www.mdpi.com/2673-2688/3/3/52 (дата звернення: 31.05.2026).
4. Babar A. Z., Akan O. B. Sustainable and Precision Agriculture with the Internet of Everything (IoE). 2024. URL: https://ieeexplore.ieee.org/document/10478146 (дата звернення: 31.05.2026).
5. Sahoo J., Barrett K. Internet of Things (IoT) Application Model for Smart Farming. 2021. URL: https://www.mdpi.com/2624-831X/2/3/22 (дата звернення: 31.05.2026).
6. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 810 p.
7. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill, 2020. 880 p.
8. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2004. 208 p.
9. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. Boston : Pearson, 2017. 1272 p.
10. Richardson C. Microservices Patterns. New York : Manning Publications, 2018. 520 p.
11. Документація Spring Boot. URL: https://docs.spring.io/spring-boot/documentation.html

(дата звернення: 31.05.2026).

12. Angular Documentation. URL: https://angular.dev (дата звернення: 31.05.2026).
13. Angular CLI Documentation. URL: https://angular.dev/tools/cli (дата звернення: 31.05.2026).
14. Oracle Corporation. Java Platform, Standard Edition Documentation. URL: https://docs.oracle.com/en/java (дата звернення: 31.05.2026).
15. Oracle Corporation. MySQL Reference Manual. URL: https://dev.mysql.com/doc (дата звернення: 31.05.2026).
16. OpenAPI Initiative. OpenAPI Specification. URL: https://spec.openapis.org/oas/latest.html (дата звернення: 31.05.2026).
17. Міністерство аграрної політики та продовольства України. Стратегія розвитку аграрного сектору економіки України. URL: https://minagro.gov.ua (дата звернення: 31.05.2026).
18. Дія.Бізнес. Цифровізація аграрного сектору України. URL: https://business.diia.gov.ua (дата звернення: 31.05.2026).
19. Національна академія аграрних наук України. Цифрові технології в сільському господарстві. URL: https://naas.gov.ua (дата звернення: 31.05.2026).
20. AgroPortal.ua. Цифровізація агробізнесу в Україні. URL: https://agroportal.ua (дата звернення: 31.05.2026).
21. Черкесов І.В., Гаманюк І.М. Розробка вебзастосунку для обліку поголів’я худоби в сільськогосподарських підприємствах. Матеріали VII Всеукраїнської науково-конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 2026 рік, Київ, ДУІКТ. Збірник тез.
22. Черкесов І.В. Проєктування інформаційної системи тваринницького господарства. Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація»(подано до друку).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

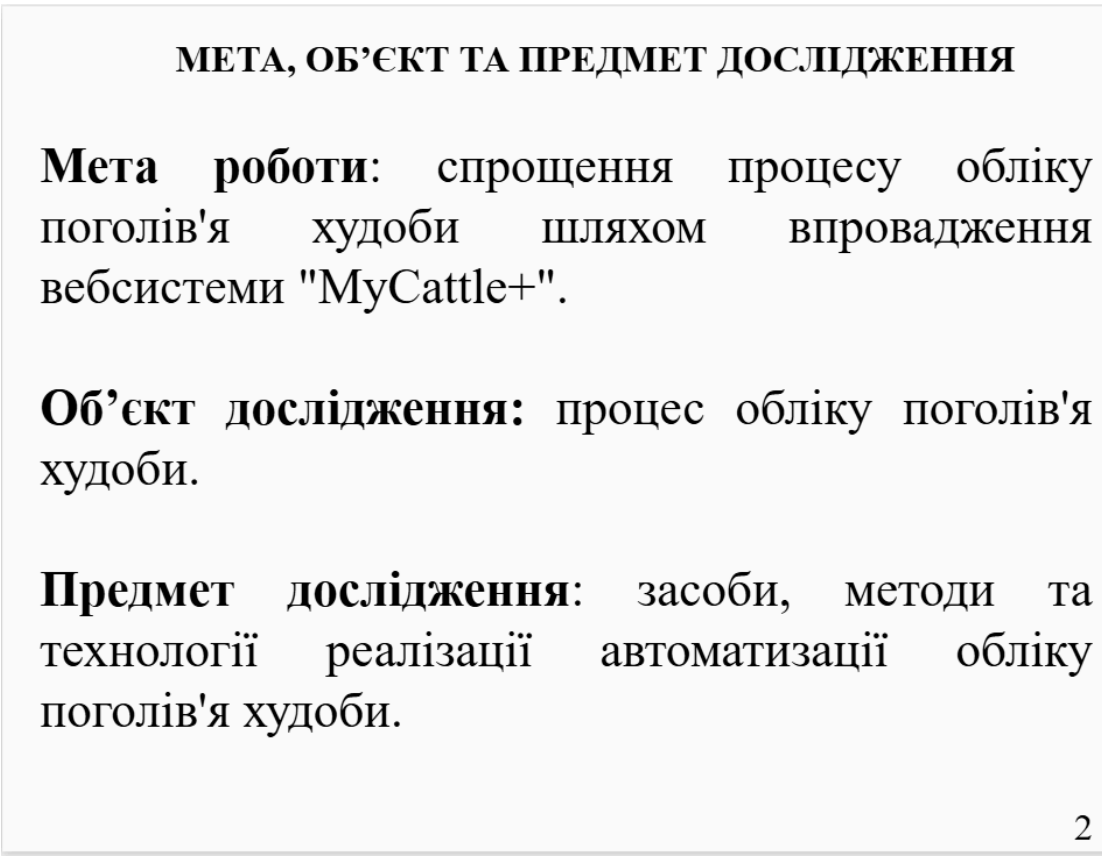
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмних засобів для ведення електронного реєстру свійських тварин

Виконав студент 4 курсу
групи ПД-44
Черкесов Ілля Валерійович
Керівник роботи
Старший викладач кафедри Гаманюк Ігор Михайлович

Київ – 2026



МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення процесу обліку поголів'я худоби шляхом впровадження вебсистеми "MyCattle+".

Об'єкт дослідження: процес обліку поголів'я худоби.

Предмет дослідження: засоби, методи та технології реалізації автоматизації обліку поголів'я худоби.

2

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну область фермерського господарства та особливості обліку тварин, приміщень, обладнання й пов'язаних даних.
2. Дослідити існуючі підходи та програмні засоби автоматизації обліку в аграрній сфері, визначити їх переваги та недоліки.
3. Обґрунтувати необхідність розроблення інформаційної системи для автоматизації облікових процесів фермерського господарства.
4. Сформувати функціональні та нефункціональні вимоги до майбутньої системи.
5. Спроєктувати структуру інформаційної системи, базу даних і взаємодію основних модулів.
6. Розробити програмний застосунок для ведення обліку об'єктів фермерського господарства та реалізувати механізми збереження, обробки, перегляду та оновлення інформації в системі.
7. Провести тестування розробленого програмного продукту та оцінити результати його впровадження.

3

АНАЛІЗ АНАЛОГІВ

Технічні характеристики	CattleMax	Herdwatch	Cattlytics	Livestock Analytics	MyCattle+
Мультиферменність (декілька господарств за людину)	обмежена	Слабка підтримка	наявна	наявна	наявна
Складність використанн	простий	середній	складний	складний	простий
Підтримка IoT / датчиків	Є можливість інтегрувати датчики, але не має можливості додавати свої датчики чи управляти ними	Є можливість під`єднати якорні маячки тварин, але не підтримує інтеграцію більш складних систем	Неповна і не надто гнучка інтеграція	Не має повноцінного IoT менеджмента	Є повна можливість додавати будь які IoT системи для ферм і налаштування типу данних
Гнучкість налаштування системи	Надто обмежена	мінімальна	наявна	Наявна, але надто переускладнена	наявна
Цінова доступність	Безкоштовний тариф або 7.99/міс. за розширений функціонал	Безкошт. тариф або $80-200/рік	$80–150/рік або $7.99/міс	Зазвичай $300–1000, Але ціна залежить від спроможностей бізнесу і його розміру	Безкоштовний тариф або $5.99/міс чи $60/рік. Масштабування ціни залежно від кількості поголов'я худоби.

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Ведення обліку подій життєвого циклу тварин
2. Забезпечення пошуку і перегляду інформації про тварин за різними параметрами (ID, порода, статус)
3. Створення звітів про стан поголів'я (кількість, структура, зміни за період)
4. Формування ієрархічної системи доступу користувачів до кількості данних у БД та можливостей редагування інформації у БД відповідно до рівня допуску
5. Підтримка додавання/редагування/видалення елементів ферми, корсистувачів та худоби
6. Визначення адміністратором прав доступу користувачів.
7. Можливість експорту даних про конкретну тварину.

Нефункціональні вимоги:

1. Забезпечення продуктивності. Обробка запитів користувача не більше 2 секунд.
2. Система має забезпечувати одночасну роботу до 1000 користувачів.
3. Захист доступу до даних через систему авторизації з ролями.
4. Сумісність із вебпереглядачами: Microsoft Edge, Google Chrome, Firefox.

5

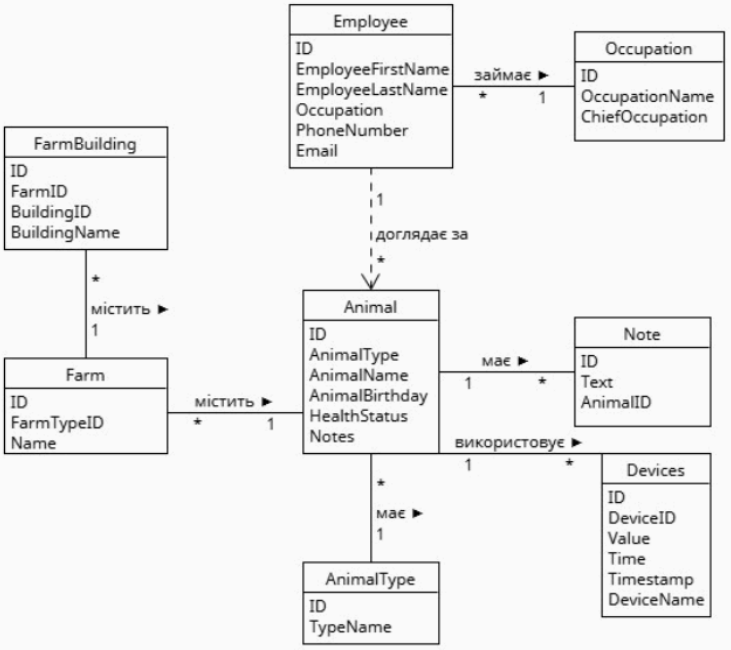
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

Моделювання прецедентів

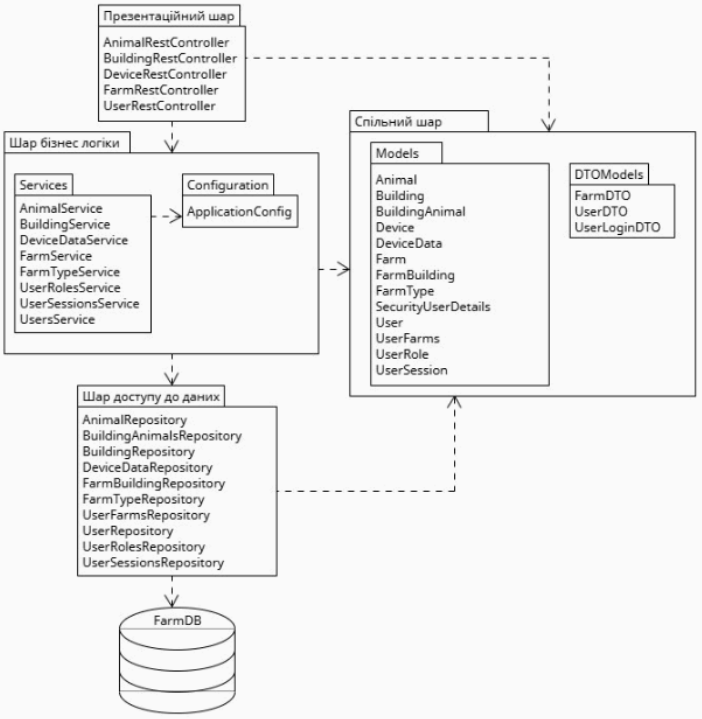
7

Моделювання предметної галузі



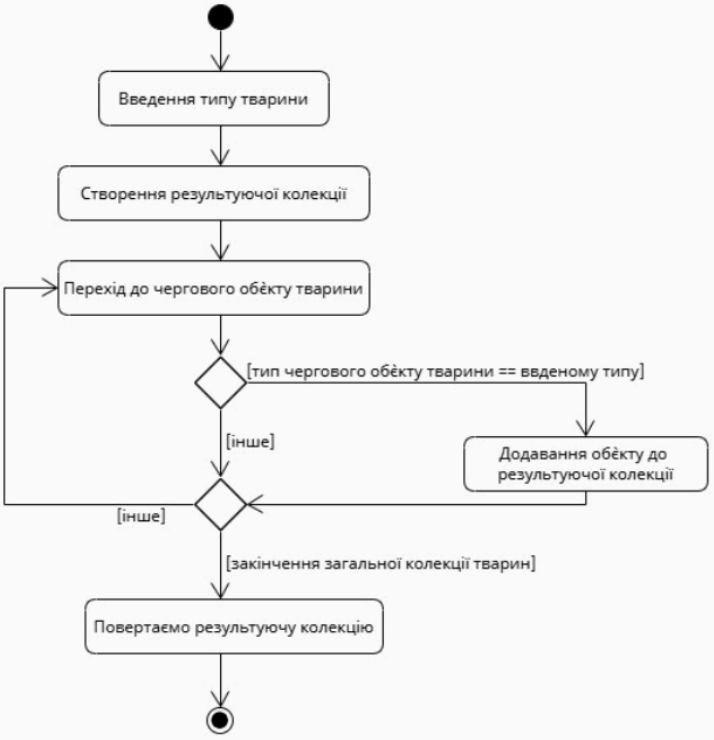
8

Логічна архітектура



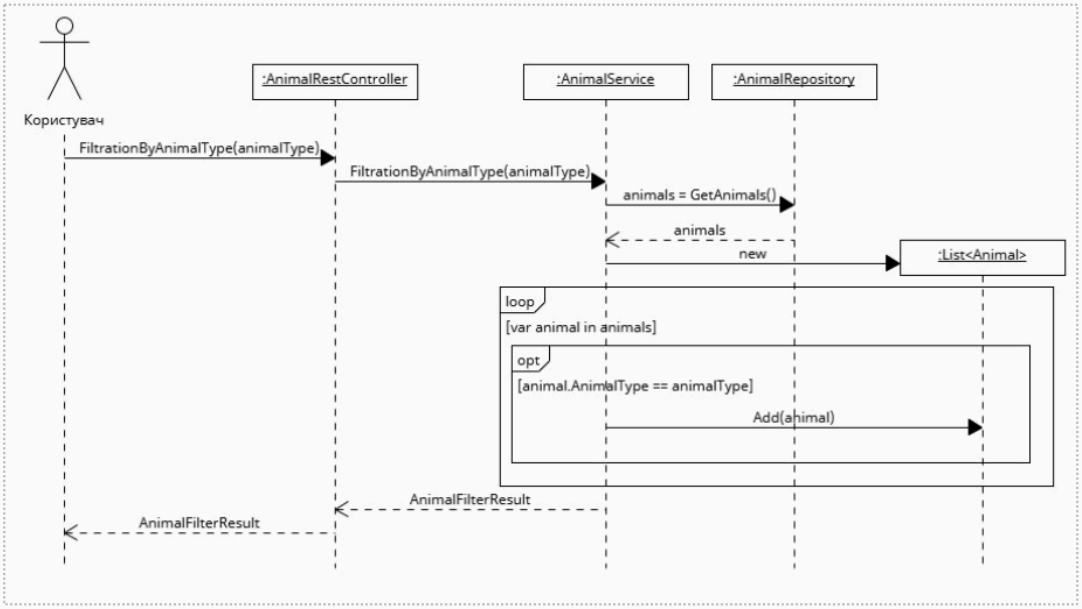
9

Алгоритм фільтрації за типом тварини



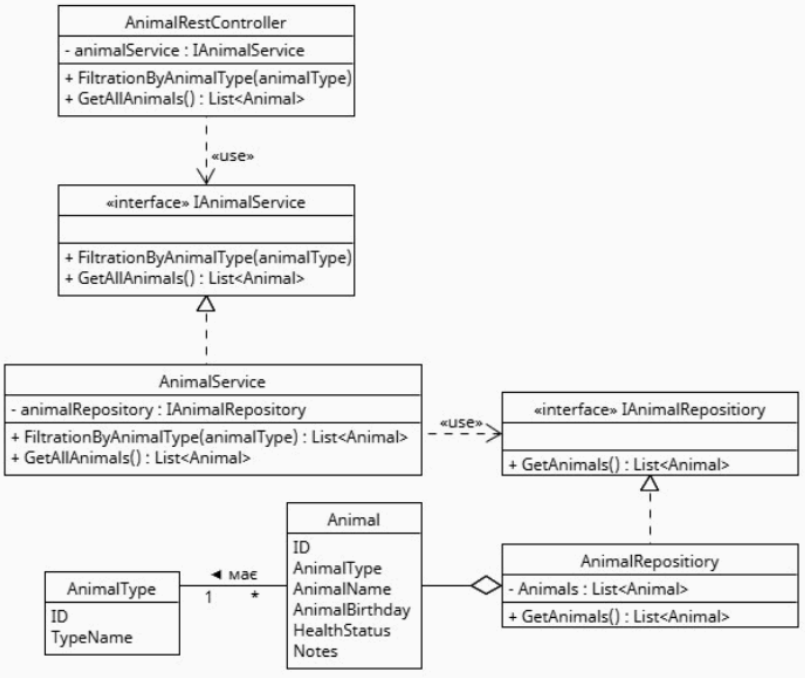
10

Проектування фільтрації за типом тварин



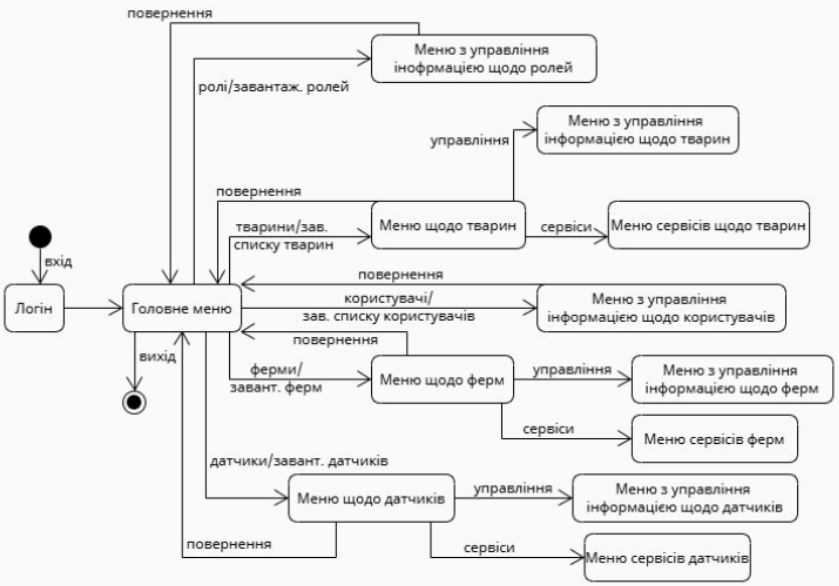
11

Діаграма класів стосовно фільтрації



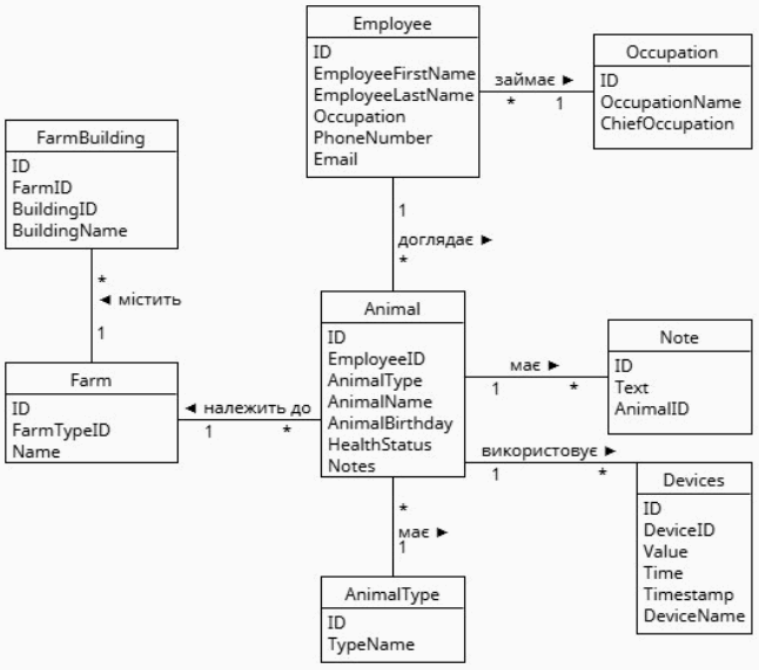
12

Моделювання меню



13

Проєктування бази даних



14

Результати тестування основних модулів

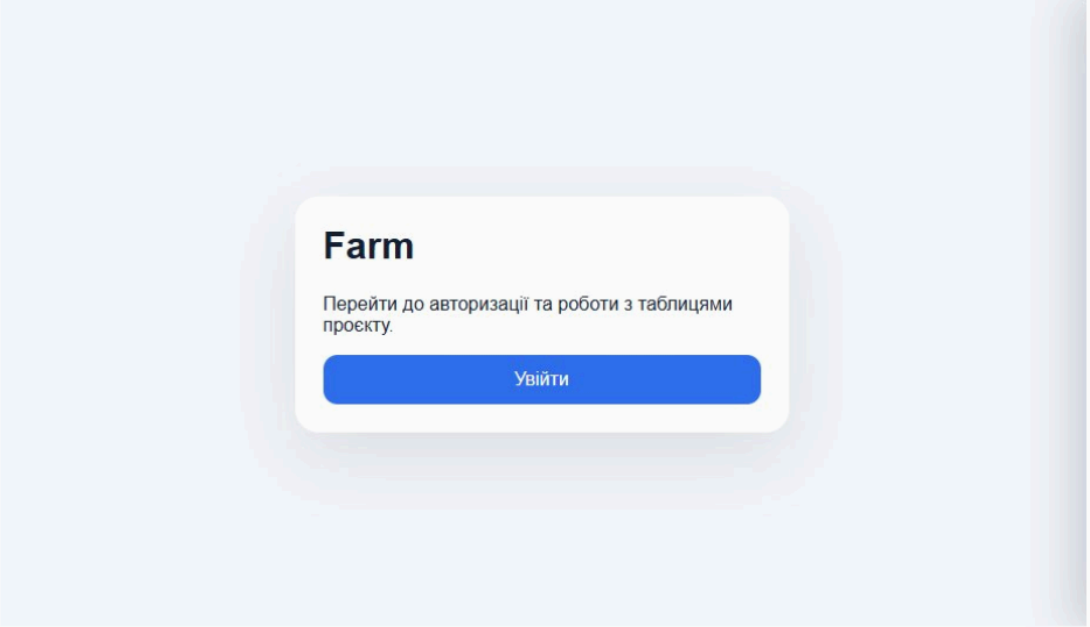
Terminal Local ×

```
FarmProjectControllerScenariosTest STANDARD_OUT

    === TEST SUMMARY ===
    PASS  4. Create farm -> persists record
    PASS  2. Current user info -> returns user data
    PASS  3. Farm list -> returns farm records
    PASS  5. Update farm -> updates stored data
    PASS  6. Admin table -> returns structured data
    PASS  7. UI metadata -> returns UI structure
    PASS  8. Auth filter -> only authorized access is accepted
    PASS  1. User login -> successful sign-in
    TOTAL: 8 | PASS: 8 | FAIL: 0 | SKIP: 0 | ABORT: 0
    ====================
```

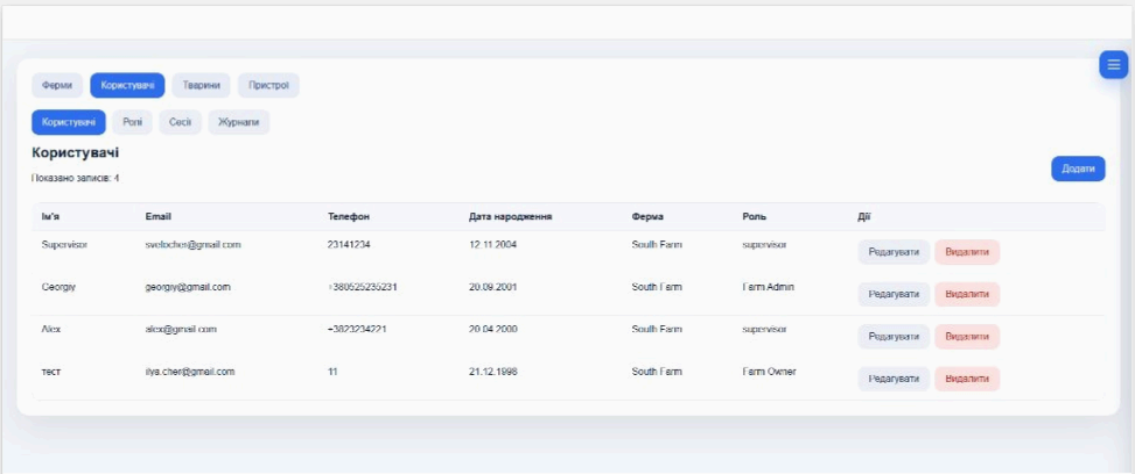
15

Екранні форми



Меню авторизації користувача

16



Головна сторінка користувача

17

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Черкесов І.В., Гаманюк І.М. Розробка вебзастосунку для обліку поголів`я худоби в сільскогосподарських підприємствах. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 23.04.2026, ДУІКТ, м.Київ. К.:ДУІКТ, 2026 С. 439-442.

2. Черкесов І.В., Гаманюк І.М. Проєктування інформаційної системи тваринницького господарства. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація»(подано до друку).

18

У ході виконання кваліфікаційної роботи було

1. Проаналізовано предметну область фермерського господарства. Визначено основні напрями автоматизації облікових процесів.

2. Досліджено існуючі підходи та програмні засоби автоматизації обліку в аграрній сфері. Визначено їх переваги та недоліки.

3. Сформовано вимоги до інформаційної системи. Спроєктовано її структуру, базу даних та взаємодію основних модулів.

4. Розроблено програмний застосунок для автоматизації обліку об'єктів фермерського господарства.

5. Проведено тестування розробленого програмного продукту. Продукт відповідає визначеним вимогам. Підтверджено доцільність використання програмного забезпечення у аграрній промисловості.

19

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

Б.1 Лістинг головного класу застосунку FarmBackApplication

```
@SpringBootApplication
@EnableJpaRepositories("com.FarmBack.Repositories")
public class FarmBackApplication {

    public static void main(String[] args) {

SpringApplication.run(FarmBackApplication.class, args);
    }
}
```

Б.2 Лістинг REST-контролера роботи з фермами FarmRestController

```
@RequestMapping("/")
@RestController
public class FarmRestController {

                private final FarmTypeService farmTypeService;
    private final FarmService farmService;
                private final FarmTypeRepository farmTypeRepository;

    FarmRestController(
        FarmTypeService farmTypeService,
        FarmService farmService,
        FarmTypeRepository farmTypeRepository
    ) {
        this.farmTypeService = farmTypeService;
        this.farmService = farmService;
                this.farmTypeRepository = farmTypeRepository;
    }

    @PostMapping("/farm")
        public Farm CreateFarm(@RequestBody FarmDTO newFarm)
            throws NotFoundException {

        List<FarmType> farmTypes =

this.farmTypeService.findAllFarmTypes();

        boolean found = farmTypes.stream()
            .anyMatch(farmType ->
                farmType.getId()

.equals(newFarm.farmTypeId));

        if (found) {
                                                return

this.farmService.CreateFarm(newFarm);
        }

            throw new NotFoundException("Farm not found");
    }

    @GetMapping("/farms")
        public List<Farm> getFarms() {return this.farmService.getAllFarms();}

    @GetMapping("/farm/{farmId}")
    public ResponseEntity<Farm> GetFarmById(
        @PathVariable String farmId) {

                                Farm farm = farmService.GetFarmById(farmId);
        return ResponseEntity.ok(farm);
    }
}
```

Б.3 Лістинг сервісу роботи з фермами FarmService

```
@Service
public class FarmService {

    private final FarmRepository farmRepository;

            public FarmService(FarmRepository farmRepository) {
        this.farmRepository = farmRepository;
    }

     public Farm CreateFarm(FarmDTO farmDTO) {

        Farm newFarm = new Farm();

                                newFarm.id = UUID.randomUUID().toString();
        newFarm.name = farmDTO.name;
                        newFarm.farmTypeId = farmDTO.farmTypeId;

        return this.farmRepository.save(newFarm);
    }

    public Farm GetFarmById(String id) {
        return farmRepository.findById(id)
            .orElseThrow(() ->
                new EntityNotFoundException(
                    "Ферма с id " + id +
                    " не найдена"));
    }

    public List<Farm> getAllFarms() {
        return this.farmRepository.findAll();
    }
}
```

Б.4 Лістинг конфігурації безпеки SecurityConfig

```
@Configuration
public class SecurityConfig {

    private final JwtAuthenticationFilter
        jwtAuthenticationFilter;

    public SecurityConfig(
                                JwtAuthenticationFilter
jwtAuthenticationFilter) {

        this.jwtAuthenticationFilter =
            jwtAuthenticationFilter;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(
        HttpSecurity http) throws Exception {

        return http
    }
            .csrf(AbstractHttpConfigurer::disable)
            .cors(cors -> {})
            .authorizeHttpRequests(auth -> auth
                .requestMatchers(
                    HttpMethod.POST,
                    "/user"
                ).permitAll()
                .requestMatchers(
                    "/login",
                    "/farms",
                    "/swagger-ui/**"
                ).permitAll()
                .anyRequest()
                .authenticated()
            )
            .addFilterBefore(
                jwtAuthenticationFilter,

UsernamePasswordAuthenticationFilter.class
            )
            .build();
    }
```