Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка інформаційної системи обліку тварин у притулку

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–22–2

_____/ С. С. Пуляєв /

Керівник кваліфікаційної роботи _____/ І. О. Доценко /

В.о. завідувача кафедри _____/ А. М. Стрюк /

Кривий Ріг

2026

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

В.о. зав. кафедри

_____ А. М. Стрюк

«____» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ–22–2 Пуляєву Сергію Сергійовичу

1. Тема: «Розробка інформаційної системи обліку тварин у притулку» затверджена наказом по КНУ № 285с від «28» травня 2026 р.

2. Термін подання студентом закінченої роботи: «15» червня 2026 р.

3. Вихідні дані по роботі: створене програмне забезпечення має комп’ютеризувати облік тварин у притулку.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): провести аналіз існуючих на ринку аналогічних програмних продуктів, обґрунтувати функціонал розроблюваної системи, створити програмне забезпечення, здійснити тестування розробленого додатку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема алгоритму, зображення екранних форм додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Розгляд літературних джерел та пошук інтернет-ресурсів з заданої тематики	15.12.25 – 18.01.26
2	Аналіз існуючих методів вирішення проблеми	19.01.26 – 07.02.26
3	Формулювання актуальності роботи і постановка завдань	08.02.26 – 18.02.26
4	Оформлення матеріалів першого розділу роботи	18.02.26 – 02.03.26
5	Створення функціональної системи та алгоритму додатку	03.03.26 – 15.03.26
6	Оформлення матеріалів другого розділу роботи	16.04.26 – 09.04.26
7	Розробка баз даних, інтерфейсу програмного забезпечення, програмних модулів	10.04.26 – 01.05.26
8	Оформлення додатків	02.05.26 – 07.05.26
9	Тестування розробленої програми	08.05.26 – 15.05.26
10	Оформлення пояснювальної записки	16.05.26 – 14.06.26

Дата видачі завдання: «14» грудня 2025 р.

Студент: _____ / С. С. Пуляєв /

Керівник роботи: _____ / І. О. Доценко /

РЕФЕРАТ

ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК ТВАРИН, БАЗА ДАНИХ, ДЕСКТОПНИЙ ДОДАТОК, АВТОМАТИЗАЦІЯ ПРОЦЕСІВ.

Пояснювальна записка: 61 с., 16 рис., 1 дод., 15 джерел.

Об’єкт дослідження – процес автоматизації обліку та обробки інформації про тварин у спеціалізованих зоозахисних організаціях.

Предмет дослідження – методи, підходи та програмні засоби розробки настільних інформаційних систем із використанням об'єктно-орієнтованих мов програмування та локальних баз даних.

У ході виконання роботи було проаналізовано поточний стан справ у сфері обліку тварин та досліджено недоліки існуючого програмного забезпечення. Обґрунтовано необхідність створення автономного додатка, який не потребує підключення до інтернету чи регулярної оплати.

Спроєктовано логічну та фізичну структуру програми за принципом трьохшарової архітектури. Розроблено структуру реляційної бази даних SQLite для збереження відомостей про тварин, біологічні види, породи та поточні статуси перебування.

Програмну реалізацію виконано мовою C# з використанням платформи .NET Framework та фреймворку Windows Forms. Створена інформаційна система дозволяє вести повноцінний облік: додавати нові картки, редагувати дані, завантажувати фотографії, здійснювати багатокритеріальний пошук, а також формувати зведену статистику та експортувати інформацію. Програма має інтуїтивно зрозумілий інтерфейс, захищена від неправильного введення даних користувачем та повністю готова до практичного використання у повсякденній роботі волонтерів.

ABSTRACT

INFORMATION SYSTEM, ANIMAL RECORD-KEEPING, DATABASE, DESKTOP APPLICATION, PROCESS AUTOMATION.

Explanatory note: 61 pages, 16 figures, 1 appendix, 15 sources.

The object of research is the process of automating the record-keeping and processing of information about animals in specialized animal welfare organizations.

The subject of research is the methods, approaches, and software tools for developing desktop information systems using object-oriented programming languages and local databases.

In the course of the work, the current state of animal record-keeping was analyzed, and the shortcomings of existing software were investigated. The necessity of creating a standalone application that does not require an internet connection or regular subscription fees was substantiated.

The logical and physical structure of the software was designed based on the three-tier architecture principle. An SQLite relational database structure was developed to store information about animals, biological species, breeds, and current placement statuses.

The software implementation was executed in C# using the .NET Framework platform and the Windows Forms framework. The developed information system enables comprehensive record-keeping: adding new records, editing data, uploading photographs, performing multi-criteria searches, as well as generating summary statistics and exporting information. The application features an intuitive interface, is protected against incorrect user data entry, and is fully ready for practical use in the daily work of volunteers.

ЗМІСТ

ВСТУП

Діяльність сучасних притулків для тварин пов'язана з постійним надходженням нових підопічних, які потребують огляду, лікування, догляду та, зрештою, пошуку нових господарів. Організація такого процесу вимагає збереження та регулярного оновлення значних обсягів інформації про кожну тварину: її вид, вік, стан здоров'я, особливі прикмети, поточний статус та дати перебування. Ведення такого обліку в паперових журналах або за допомогою неспеціалізованих електронних таблиць є незручним, забирає багато часу в персоналу та підвищує ймовірність випадкової втрати чи спотворення даних.

Тому розробка повноцінної інформаційної системи для автоматизації рутинних процесів є достатньо актуальним завданням. Впровадження спеціалізованого програмного забезпечення дозволить оптимізувати роботу волонтерів та адміністраторів, прискорити пошук потрібної інформації та значно спростити підготовку статистичної звітності.

Об'єкт дослідження – процес автоматизації обліку та обробки інформації про тварин у спеціалізованих зоозахисних організаціях.

Предмет дослідження – методи, підходи та програмні засоби розробки настільних інформаційних систем із використанням об'єктно-орієнтованих мов програмування та локальних баз даних.

Мета дипломної роботи полягає у створенні зручної та надійної десктопної програми для управління інформацією про тварин у притулку, яка забезпечить збереження даних, швидку фільтрацію, генерацію звітів та можливість експорту записів для подальшого аналізу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПИТАННЯ ЦИФРОВІЗАЦІЇ ОБЛІКУ ТВАРИН У ПРИТУЛКУ

1.1. Особливості діяльності притулків для тварин

Основне завдання будь-якої зоозахисної організації полягає у забезпеченні безпечного місця перебування для безпритульних, покинутих або травмованих тварин. Такі установи займаються не лише наданням даху над головою та організацією харчування, а й наданням ветеринарного догляду, що включає первинний огляд, вакцинацію та необхідні медичні процедури. Зрештою, головною метою їхньої роботи є соціалізація підопічних та пошук для них нових, відповідальних господарів.

Для правильної організації щоденної роботи життєво необхідний детальний облік усіх мешканців притулку. Адміністрація та волонтери повинні володіти актуальними даними про кількість підопічних, їхній розподіл за видами (собаки, коти, птахи тощо), породами, віковими категоріями та станом здоров'я. Без належної систематизації записів існує великий ризик втрати важливих медичних відомостей, інформації про особливі прикмети або плутанини під час передачі тварин новим власникам.

Життєвий цикл перебування тварини в організації включає кілька етапів, кожен з яких супроводжується зміною її статусу. Під час надходження фіксується відповідна дата, проводиться огляд, визначаються стать, приблизний вік, вага та колір. Залежно від стану здоров'я, тварина може бути розміщена у загальному вольєрі або направлена на лікування.

Коли потенційна родина знайдена, підопічний може отримати статус забронюваного на час оформлення формальностей, а після успішної передачі – статус прилаштованого. Кожен такий перехід є важливою подією, яку необхідно правильно задокументувати для ведення внутрішньої статистики та звітності організації.

Традиційно інформація в багатьох подібних установах зберігається у паперових картках, журналах або звичайних електронних таблицях.

Проте такий підхід має суттєві недоліки: паперові носії швидко зношуються і ускладнюють пошук за певними критеріями, а електронні таблиці стають незручними при великій кількості записів і не гарантують цілісності даних.

Крім того, додавання фотографій або розгорнутих приміток до текстових документів є вкрай непрактичним.

Рисунок 1.1 – Типовий притулок для тварин [1]

Саме тому виникає об'єктивна потреба у використанні спеціалізованих баз даних, які здатні структурувати всі ці відомості в одному місці.

Це дозволить надійно зберігати інформацію, швидко фільтрувати її за потрібними параметрами та забезпечувати зручний доступ до карток тварин для всього персоналу.

1.2. Аналіз існуючих програмних рішень для обліку тварин

Перш ніж переходити до проєктування власного програмного забезпечення, доцільно дослідити вже існуючі на ринку системи. На сьогодні створено чимало продуктів, призначених для автоматизації роботи зоозахисних організацій. Їх можна умовно поділити на кілька категорій: загальні офісні додатки, великі міжнародні платформи та локальні спеціалізовані програми.

Як частково згадувалося раніше, багато невеликих установ намагаються адаптувати звичайні електронні таблиці [2,3] для ведення бази даних. Хоча цей підхід не вимагає фінансових витрат і додаткового навчання персоналу, він зовсім не підходить для довгострокового використання. У таблицях дуже складно налаштувати правильні зв'язки між сутностями – наприклад, зробити так, щоб при виборі певного виду тварини для заповнення пропонувалися лише відповідні йому породи. Крім того, зі збільшенням кількості записів робота з таким файлом стає незручною, а процес формування зведеної статистики займає багато часу.

Іншою категорією є масштабні міжнародні системи управління притулками. Вони володіють величезним арсеналом можливостей: від медичних карток до інтеграції з фінансовими сервісами та онлайн-базами. Розглянемо деякі з них.

Однією з популярних платформ є Shelterluv [4,5]. Вона розроблена переважно для великих західних організацій та ветеринарних центрів і працює як хмарне рішення. Програма містить розвинені інструменти для створення цифрових карток тварин, відстеження графіків вакцинації та автоматизації процесу передачі вихованців новим господарям.

Проте для вітчизняних притулків вона має вагомі недоліки. Платформа працює за платною моделлю підписки або знімає комісію за кожну успішну передачу тварини. Це є важким фінансовим тягарем для організацій, що існують за рахунок благодійності. Інтерфейс програми повністю англомовний, а офіційна підтримка української мови відсутня, що створює значні труднощі

під час щоденного використання персоналом. До того ж додаток міцно прив’язаний до американських реєстрів мікрочипів та платіжних сервісів, які в наших умовах не працюють, але перевантажують інтерфейс і заплутують волонтерів.

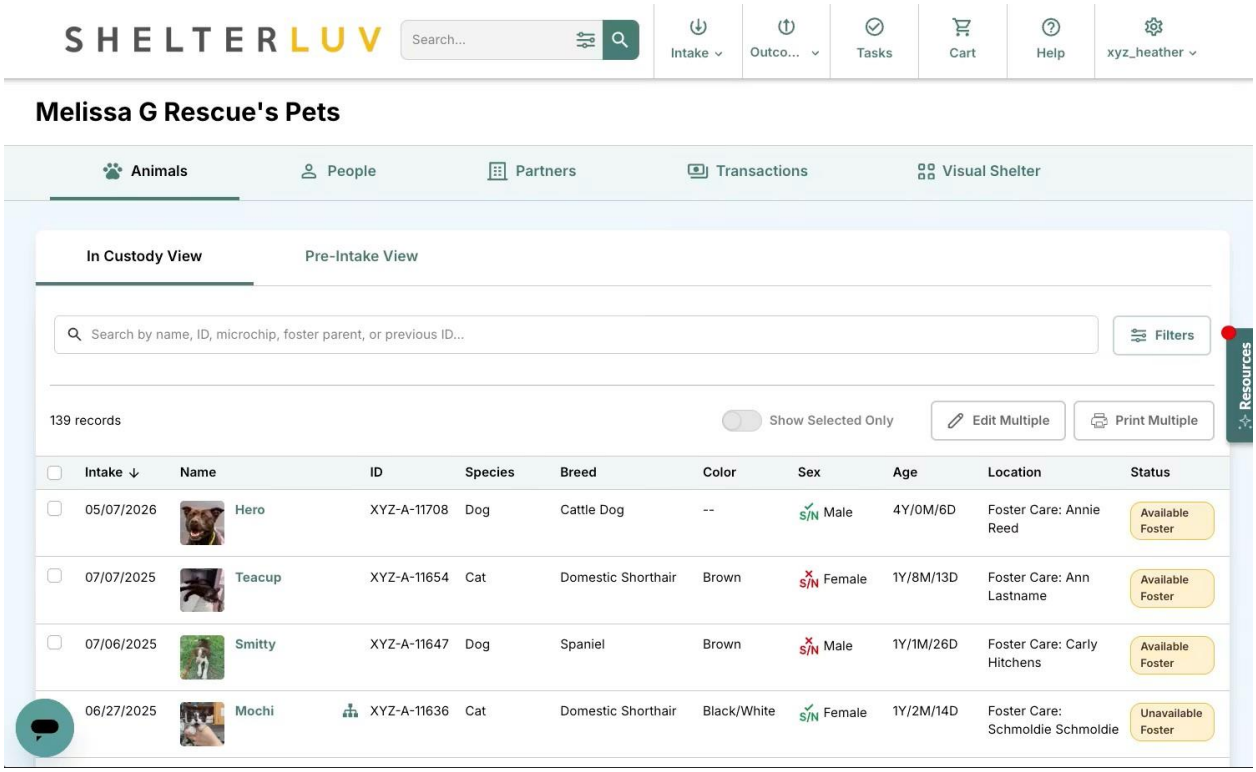

Рисунок 1.2 – Shelterluv

Іншим відомим прикладом є масштабна хмарна система PetPoint [6,7]. Вона дозволяє детально фіксувати раціон харчування, медичну історію та поведінкові особливості кожної особини, а також вести облік фінансових пожертвувань. Через намагання розробників охопити всі можливі сфери діяльності додаток став занадто громіздким. Картка тварини містить велику кількість обов'язкових полів для заповнення, більшість із яких є специфічними і непотрібними для повсякденного обліку.

Крім того, звичайний користувач без попередньої підготовки навряд чи зможе швидко розібратися в навігації програми. Оскільки система зберігає дані у хмарі, будь-які проблеми зі зв'язком або вимкнення електрики повністю

блокують доступ до бази даних, унеможливлюючи роботу організації в такі моменти.

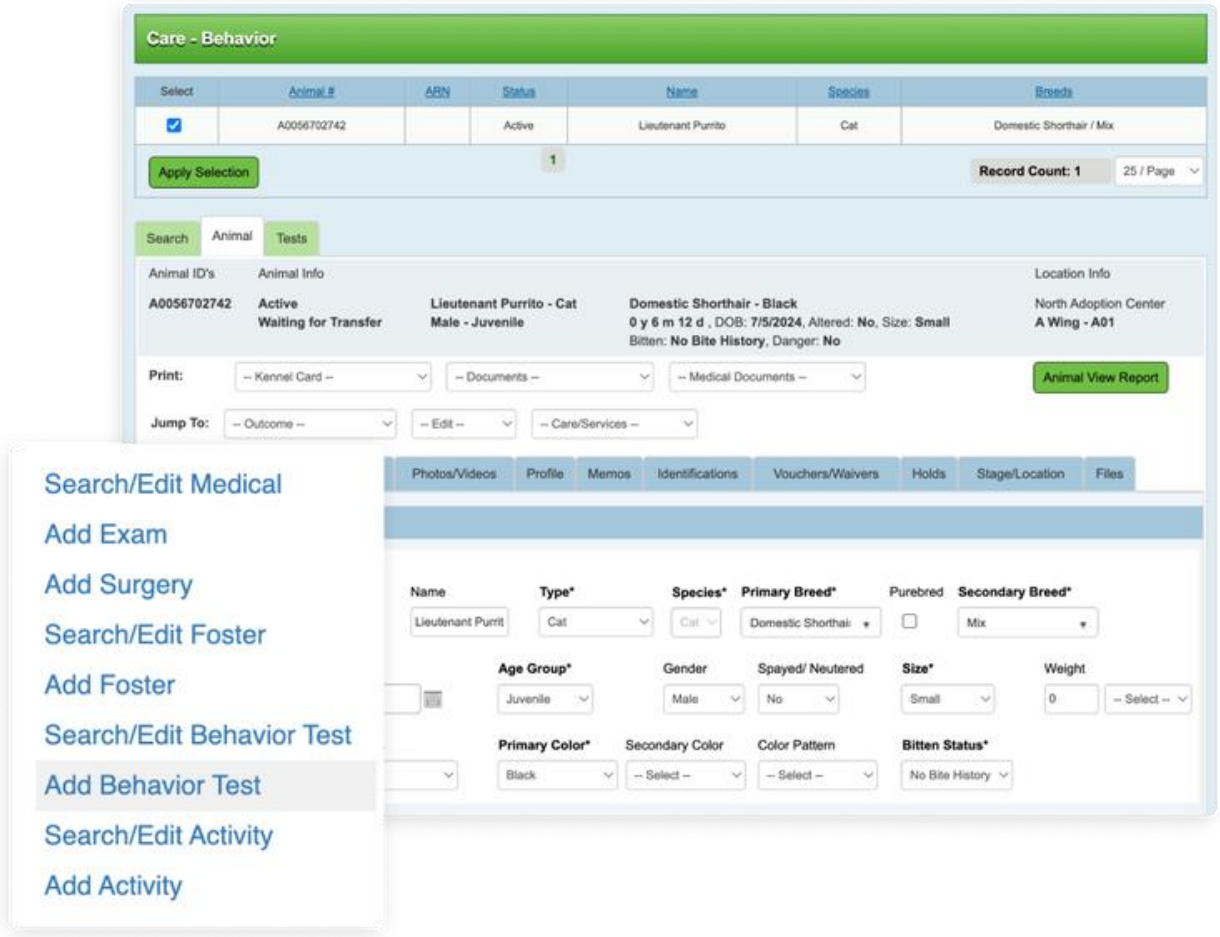

Рисунок 1.3 – PetPoint

Також варто згадати Animal Shelter Manager [8,9]. Це безкоштовний програмний продукт із відкритим вихідним кодом, що надає можливість відстежувати переміщення тварин між вольєрами та формувати статистику. Але для запуску цієї програми необхідно самостійно розгортати вебсервер та налаштовувати середовище бази даних. Для більшості працівників притулків без інженерної освіти такий процес є занадто складним, а будь-яка технічна помилка вимагає залучення сторонніх фахівців. Інтерфейс системи створювався досить давно, він не відповідає сучасним стандартам зручності та

виглядає архаїчно. Хоча розробники додали можливість вибору мови, переклад українською є неповним і часто містить некоректні терміни.

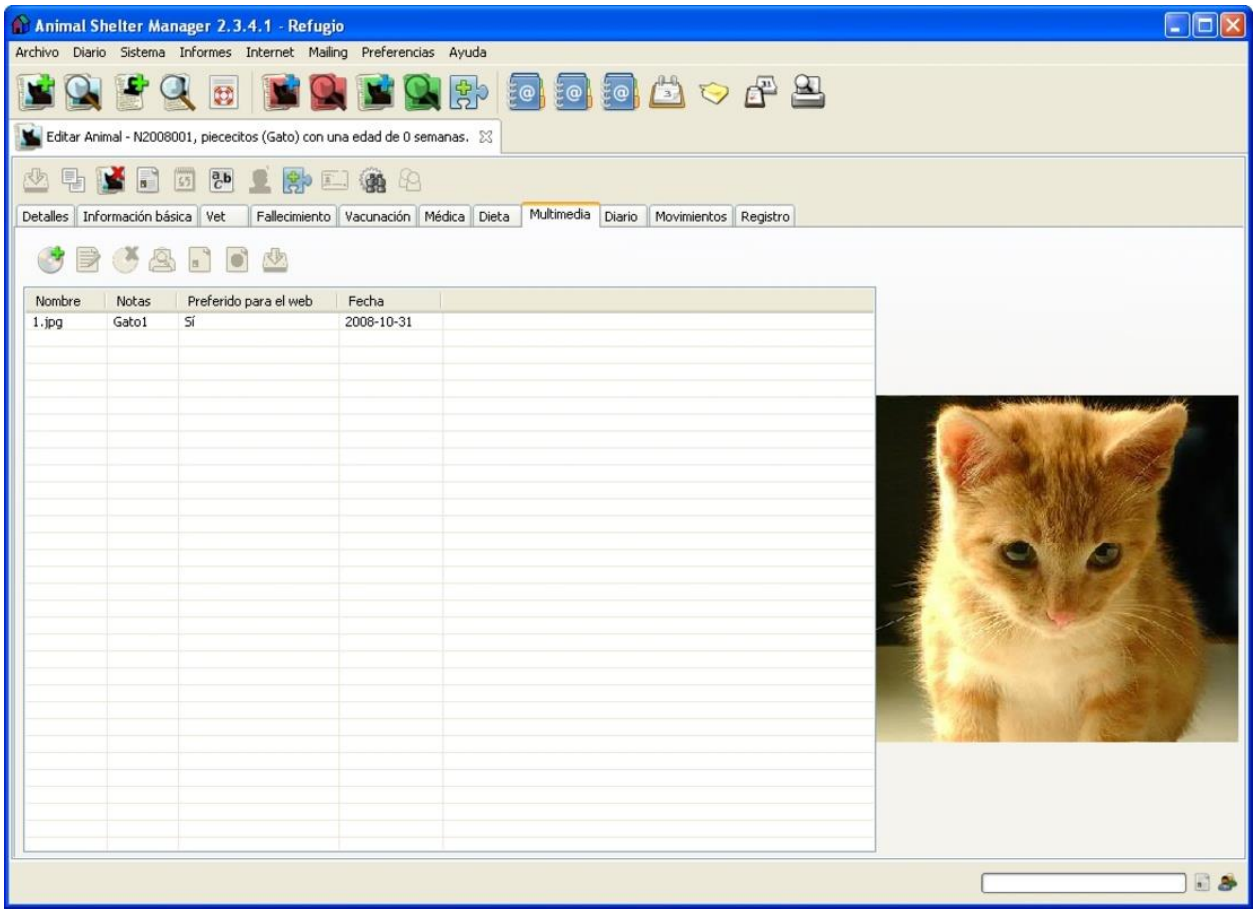

Рисунок 1.4 – Animal Shelter Manager

Проведений аналіз показує, що існуючі міжнародні аналоги або є занадто дорогими та складними, або вимагають постійного підключення до мережі та високої технічної кваліфікації для налаштування.

Також на ринку трапляються вітчизняні розробки, однак найчастіше вони створюються програмістами-ентузіастами під специфічні потреби конкретного закладу. Але оскільки програма пишеться під конкретну установу, її логіка часто є занадто прив'язаною до її внутрішніх процесів. Наприклад, у самій системі можуть бути намертво прописані назви певних приміщень, вольєрів або специфічні категорії статусів тварин, які використовує лише цей притулок. Якщо інша організація спробує застосувати

такий продукт для своїх задач, вона стикнеться з тим, що змінити ці налаштування або додати нові види тварин без втручання у програмний код просто неможливо.

Також найпоширеніша проблема волонтерських розробок – це їхній короткий життєвий цикл. Співпраця розробника з притулком зазвичай триває недовго. Після її завершення працівники залишаються з програмою сам на сам. Якщо виникають непередбачувані помилки, змінюється операційна система на робочих комп'ютерах або з'являється потреба в нових формах звітності, внести зміни до додатка стає неможливо, оскільки технічна підтримка відсутня.

Такі локальні рішення дуже рідко супроводжуються повноцінною архітектурною документацією. Вихідний код зазвичай залишається в одного розробника, він може бути неструктурованим, а пояснювальні коментарі в ньому відсутні. Через це залучити іншого програміста для виправлення помилок або розширення можливостей програми стає вкрай важко – фахівцю простіше написати нову систему з нуля, ніж намагатися розібратися в чужому проєкті без інструкцій.

Підсумовуючи результати аналізу, можна зробити висновок про наявність вільної ніші для простого та автономного інструменту. Для багатьох організацій найбільш раціональним вибором стане класична настільна програма з локальною базою даних. Такий формат забезпечує незалежність від стабільності інтернет-з'єднання, гарантує безпечне зберігання інформації безпосередньо на комп'ютері користувача та виключає будь-які абонентські платежі. Тому створення спеціалізованого додатку з рідною локалізацією, продуманим функціоналом для фільтрації записів і зручною системою експорту даних є цілком виправданим кроком.

1.3. Аналіз інформаційних потоків у притулку

Для успішного проєктування майбутньої програми важливо детально проаналізувати, як саме інформація циркулює всередині організації. Кожна подія в житті притулку породжує певні інформаційні потоки, які потребують

фіксації, обробки та подальшого збереження в базі даних. Увесь процес роботи з даними можна розділити на кілька логічних етапів:

а) реєстрація тварини;

б) зміна статусу тварини;

в) пошук інформації;

г) формування звітності.

Перший етап інформаційного обміну починається в момент надходження нового підопічного. Працівник проводить первинний огляд та збирає базові відомості: вид, стать, приблизну дату народження, дату надходження, вагу, особливі прикмети та забарвлення. За можливості робиться фотографія. У цей момент створюється новий запис у системі. Інформаційний потік спрямовується від користувача до бази даних, де інформація структурується та отримує унікальний ідентифікатор. На цьому етапі дуже важливою є валідація – програма повинна перевіряти правильність введених значень, коректність дат та обов'язковість заповнення основних полів, щоб уникнути збереження неповної картки.

У процесі перебування в установі стан та життєві обставини тварини постійно змінюються. Підопічний може потребувати медичного втручання і перейти в статус «На лікуванні», або ж для нього знайдеться нова родина, що означатиме перехід до категорії «Заброньована» чи «Прилаштована». З точки зору інформаційних потоків, це операція оновлення вже існуючого запису. Користувач ініціює зміну певного параметра через форму редагування, програма фіксує нове значення (а іноді й додає нові текстові примітки) та зберігає оновлену версію картки. Цей процес вимагає зручного механізму доступу до запису, щоб адміністратор міг вносити зміни без ризику втратити попередню інформацію.

Щоденна рутина волонтерів передбачає регулярні звернення до збережених даних. Коли приходять потенційні господарі, які хочуть взяти, наприклад, дорослого собаку певної породи, виникає зворотний інформаційний потік – від бази даних до користувача. Система повинна

приймати критерії запиту, обробляти їх і виводити на екран лише релевантні результати. Працівникам необхідна можливість як прямого текстового пошуку за кличкою, так і багатокритеріальної фільтрації за видом, породою та поточним статусом. Правильно налаштований пошуковий потік значно економить час персоналу і є однією з головних переваг автоматизованого обліку.

Останнім великим інформаційним блоком є агрегація даних. Керівництву, волонтерам та спонсорам регулярно потрібні зведені показники діяльності організації. Це стосується підрахунку загальної кількості тварин, їхнього розподілу за видами та статтю, кількості прилаштованих підопічних або визначення середнього віку мешканців. Інформаційний потік тут полягає у зборі розрізнених записів з таблиць бази даних, їх математичній обробці та виведенні у вигляді зручної статистики на екран.

Крім того, часто виникає потреба передати ці дані зовнішнім користувачам. Тому потік завершується можливістю експорту підготовленої інформації у загальноприйняті формати, такі як файли електронних таблиць або текстові документи з роздільниками, для подальшого аналізу чи друкування.

1.4. Визначення функціональних вимог до системи

На основі проведеного аналізу інформаційних потоків та виявлених недоліків існуючих рішень можна сформувати перелік основних функціональних вимог до розроблюваної інформаційної системи. Для того, щоб програма стала справді корисним інструментом у повсякденній роботі притулку, вона має забезпечувати повний цикл роботи з базою даних і підтримувати такі базові можливості.

Однією з найважливіших функцій є можливість додавання записів та створення карток для нових підопічних. Користувач повинен мати змогу вводити вичерпну інформацію: кличку, вид, породу, стать, дати народження та надходження, фізичні параметри (вага, колір), особливі прикмети,

початковий статус, а також завантажувати фотографію та залишати текстові примітки. При цьому програма має здійснювати перевірку (валідацію) правильності заповнення обов'язкових полів і коректності введених дат, щоб запобігти збереженню неповної інформації.

Оскільки інформація про тварину та її стан постійно змінюється, система повинна надавати зручні інструменти для редагування даних існуючих карток. Це дозволить адміністраторам актуалізувати медичні дані, додавати нові спостереження або змінювати статус перебування (наприклад, переводити тварину до категорії лікування чи позначати її як прилаштовану в нову родину).

Для випадків помилкового введення даних або створення дублів необхідно передбачити функцію видалення запису з бази. Щоб уникнути випадкової втрати важливої інформації через необережність волонтера, ця операція обов'язково повинна супроводжуватися вікном попередження та вимагати додаткового підтвердження дій користувача.

Для швидкої навігації великою базою даних додаток має підтримувати функцію прямого текстового пошуку за ім'ям тварини. Пошуковий механізм повинен бути чутливим до регістру введених символів, швидко обробляти запит і містити кнопку для швидкого скидання результатів, щоб користувач міг легко повернутися до загального списку всіх підопічних.

Окрім звичайного пошуку, необхідна розвинена система фільтрації, яка дозволить сортувати тварин за кількома параметрами одночасно. Користувач повинен мати можливість відбирати записи за видом, її породою та поточним статусом. Дуже важливою вимогою є динамічна залежність параметрів: при виборі певного виду (наприклад, «Собака») випадаючий список доступних порід має оновлюватися автоматично, приховуючи варіанти, що стосуються інших тварин.

Програма повинна самостійно обробляти наявну інформацію та генерувати зведену статистику. Сюди входить підрахунок загальної кількості мешканців притулку, їх детальний розподіл за статусами, видами та статтю.

Також система має автоматично вираховувати загальний середній вік усіх тварин у базі та окремі показники середнього віку для кожного виду.

Для подальшої роботи з інформацією поза межами розроблюваної системи потрібна можливість вивантаження бази даних. Додаток повинен підтримувати експорт карток у формати електронних таблиць та текстових файлів із роздільниками. При цьому має гарантуватися збереження правильного кодування символів, щоб український текст та всі спеціальні символи коректно відображалися на будь-яких комп'ютерах чи пристроях для друку.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТВАРИН У ПРИТУЛКУ

2.1 Постановка задачі розробки

Головною метою розробки є створення надійної та простої у використанні десктопної програми, яка автоматизує рутинну роботу працівників зоозахисної організації. Система має стати єдиним центром для збереження всієї інформації про підопічних, замінивши собою паперові журнали та розрізнені електронні таблиці. Завдяки впровадженню цього програмного забезпечення очікується значне прискорення пошуку потрібних даних, спрощення процедури інвентаризації та зручне формування звітів для керівництва чи потенційних власників. Програма повинна допомагати волонтерам зосередитися на догляді за тваринами, а не на тривалій роботі з документами.

Програма розробляється з урахуванням умов, у яких зазвичай працюють невеликі притулки. Вона має бути повністю автономною та повноцінно функціонувати без доступу до мережі Інтернет. Серед технічних вимог варто виділити невимогливість до апаратних ресурсів: додаток повинен стабільно працювати на комп'ютерах з операційними системами Windows 7 або новіших версій, займаючи мінімум вільного місця на диску.

Інтерфейс має бути інтуїтивно зрозумілим, розробленим виключно українською мовою, щоб будь-який новий працівник міг почати роботу без тривалого навчання. Також вимагається забезпечення стійкості до помилок користувача, наявність перевірок правильності введених даних під час збереження карток та зручна система взаємодії з таблицями для перегляду великих масивів інформації.

Для реалізації поставлених завдань було обрано мову програмування C# [10,11] у поєднанні з платформою .NET Framework [12,13]. Цей вибір пояснюється тим, що мова C# має потужні об'єктно-орієнтовані можливості, а

сама платформа добре оптимізована й уже присутня в більшості сучасних операційних систем сімейства Windows. Це означає, що користувачам не доведеться встановлювати багато додаткових компонентів, щоб запустити готовий продукт.

Створення графічного інтерфейсу базується на технології Windows Forms. Вона дозволяє розробляти класичні віконні додатки зі стандартними елементами управління, такими як кнопки, текстові поля та випадаючі списки. Такий підхід робить зовнішній вигляд програми звичним для будь-якого користувача персонального комп'ютера, що скорочує час на адаптацію персоналу.

Для збереження інформації найкращим рішенням стало використання локальної бази даних SQLite [14] разом із провайдером доступу System.Data.SQLite [15]. На відміну від великих серверних баз, SQLite зберігає всі таблиці у звичайному невеликому файлі безпосередньо на комп'ютері. Це позбавляє необхідності налаштовувати окремий сервер, що є ідеальним варіантом для портативної та локальної програми.

З точки зору проєктування коду, розробка спиратиметься на трьохшарову архітектуру. Вона передбачає розділення програми на шар інтерфейсу, шар бізнес-логіки та шар доступу до даних. Такий поділ дозволить організувати код максимально правильно: візуальні форми відповідатимуть лише за відображення інформації, репозиторії та сервіси займатимуться обробкою запитів, а спеціальний клас управління базою відповідатиме за підключення. Це спростить процес написання програми та полегшить її подальше вдосконалення.

2.2 Загальна архітектура програмної системи

Для забезпечення надійності та зручності подальшого супроводу коду, розроблена інформаційна система побудована за класичним принципом трьохшарової архітектури. Такий підхід передбачає логічне розділення всієї програми на окремі ізольовані рівні: візуальний інтерфейс, бізнес-логіку та

механізми доступу до даних. Кожен із цих блоків виконує свою вузьку задачу, що дозволяє уникнути плутанини та спрощує розширення функціоналу в майбутньому.

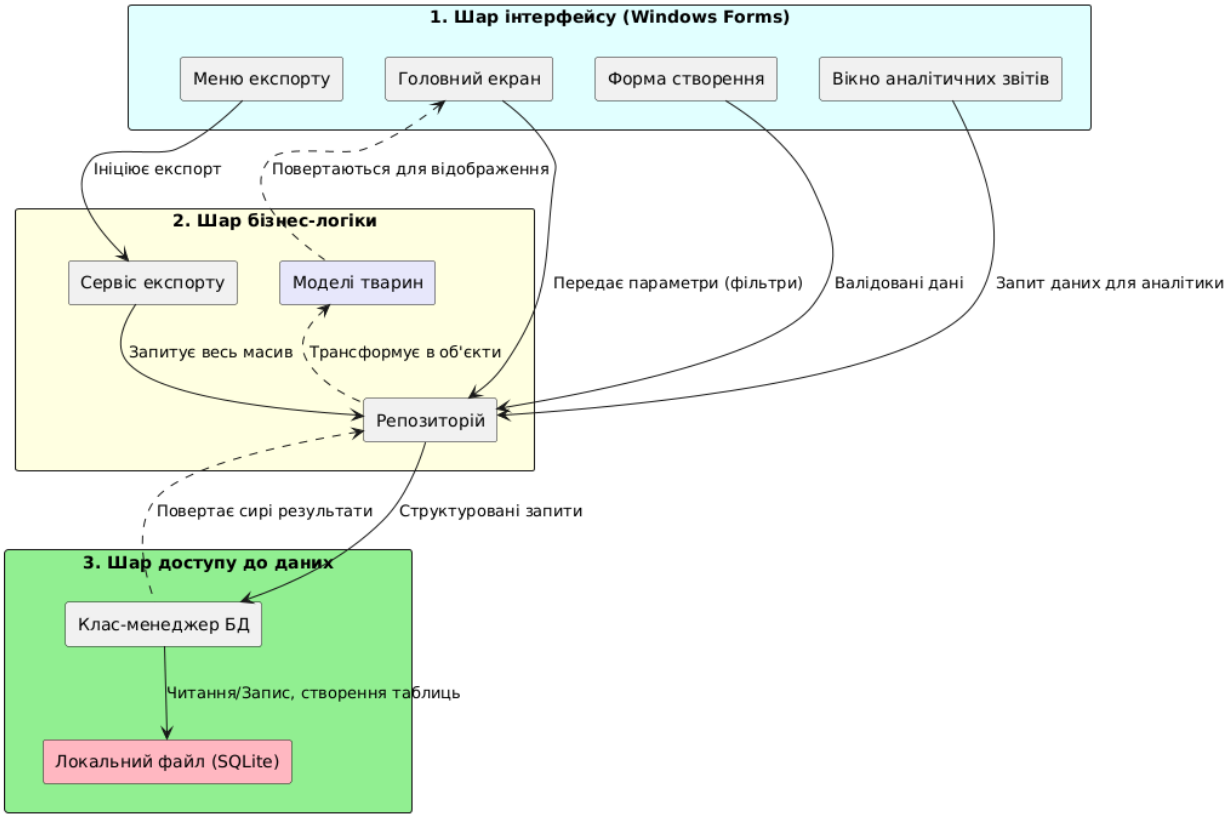

Рисунок 2.1 – Архітектура інформаційної системи обліку тварин у притулку

Клієнтська частина є видимою для користувача оболонкою програми, та реалізована за допомогою візуальних компонентів Windows Forms. Вона складається з набору взаємопов'язаних вікон: головного екрана з таблицею всіх підопічних, форми для створення та редагування карток тварин, вікна перегляду аналітичних звітів та меню налаштування експорту.

Основне завдання цього рівня полягає у правильному відображенні інформації, зборі введених волонтером даних та їх первинній валідації. Саме клієнтська частина відповідає за перевірку того, чи заповнені обов'язкові поля (такі як ім'я, вид, стать), чи правильно обрані дати надходження та народження. Важливою особливістю є те, що графічний інтерфейс не містить

складних обчислювальних алгоритмів і не звертається безпосередньо до бази даних – він лише реагує на дії користувача та передає відповідні команди на нижчі рівні архітектури.

За фізичне збереження інформації на найнижчому рівні відповідає шар доступу до даних, який у програмі представлений спеціальним класом-менеджером. Його основна мета – інкапсулювати всі технічні нюанси роботи з локальним SQLite-файлом.

Менеджер бере на себе організацію життєвого циклу бази: він перевіряє наявність файлу під час запуску додатка, створює необхідні таблиці (для видів, порід, статусів та загального переліку тварин) і заповнює довідники початковими значеннями. Також цей компонент здійснює управління підключеннями, приймаючи структуровані запити від бізнес-логіки, виконуючи їх і повертаючи сирі результати для подальшої обробки. Завдяки такому відокремленню, інші частини програми взагалі не залежать від специфіки мови запитів до бази.

Зв'язувальною ланкою між діями користувача та сховищем інформації є середній рівень – шар бізнес-логіки. Він містить клас репозиторію та спеціалізовані сервіси, де реалізовані основні правила роботи системи.

Взаємодія між рівнями відбувається за таким сценарієм: коли адміністратор, наприклад, обирає в інтерфейсі фільтр для відображення лише собак, які перебувають на лікуванні, візуальна форма передає ці параметри до репозиторію. Репозиторій формує відповідне завдання для менеджера бази даних. Після отримання результатів від менеджера, репозиторій перетворює сирі дані на зручні об'єкти (моделі тварин) і повертає їх назад на клієнтську частину для відображення в таблиці.

Аналогічно працюють і додаткові модулі, зокрема сервіс експорту. Візуальна форма лише запитує шлях для збереження файлу, після чого передає керування сервісу, який самостійно витягує весь масив інформації з репозиторію, форматує його та генерує готовий документ. Така модульна

взаємодія дозволяє розвантажити візуальні вікна, залишаючи їхній код лаконічним і зосередженим виключно на графічних елементах.

2.3 UML-моделювання інформаційної системи

2.3.1 Розробка логічної моделі даних

На етапі проєктування бази даних важливою складовою є розробка логічної моделі, яка зазвичай візуалізується у вигляді діаграми «сутність-зв'язок» (ER-діаграми). Цей інструмент допомагає абстрагуватися від конкретних технічних деталей реалізації та графічно показати, з яких інформаційних об'єктів складається предметна область і як саме вони взаємодіють між собою.

Головними елементами логічної моделі є сутності – об'єкти реального світу, інформацію про які необхідно зберігати. Кожна з них наділена певним набором атрибутів, що описують її властивості. У межах розроблюваної системи виділено чотири основні сутності:

а) Тварина. Це найголовніший об'єкт моделі. Її атрибути включають унікальний ідентифікатор, кличку, стать, дату народження, дату надходження, забарвлення, вагу, особливі прикмети, шлях до збереженої фотографії та текстові примітки.

б) Вид. Описує біологічну приналежність (собаки, коти, птахи тощо). Має лише базові атрибути: ідентифікатор, назву та загальний опис.

в) Порода. Виступає деталізацією виду. Її атрибутами є ідентифікатор та назва.

г) Статус. Характеризує стан перебування підопічного в установі на даний момент. Складається з ідентифікатора, назви статусу та його розширеного опису.

Для того щоб інформація була цілісною, а в базі не виникало дублювання або суперечностей, між сутностями встановлено логічні зв'язки. У цій моделі застосовується найбільш поширений тип відношення – «один до багатьох».

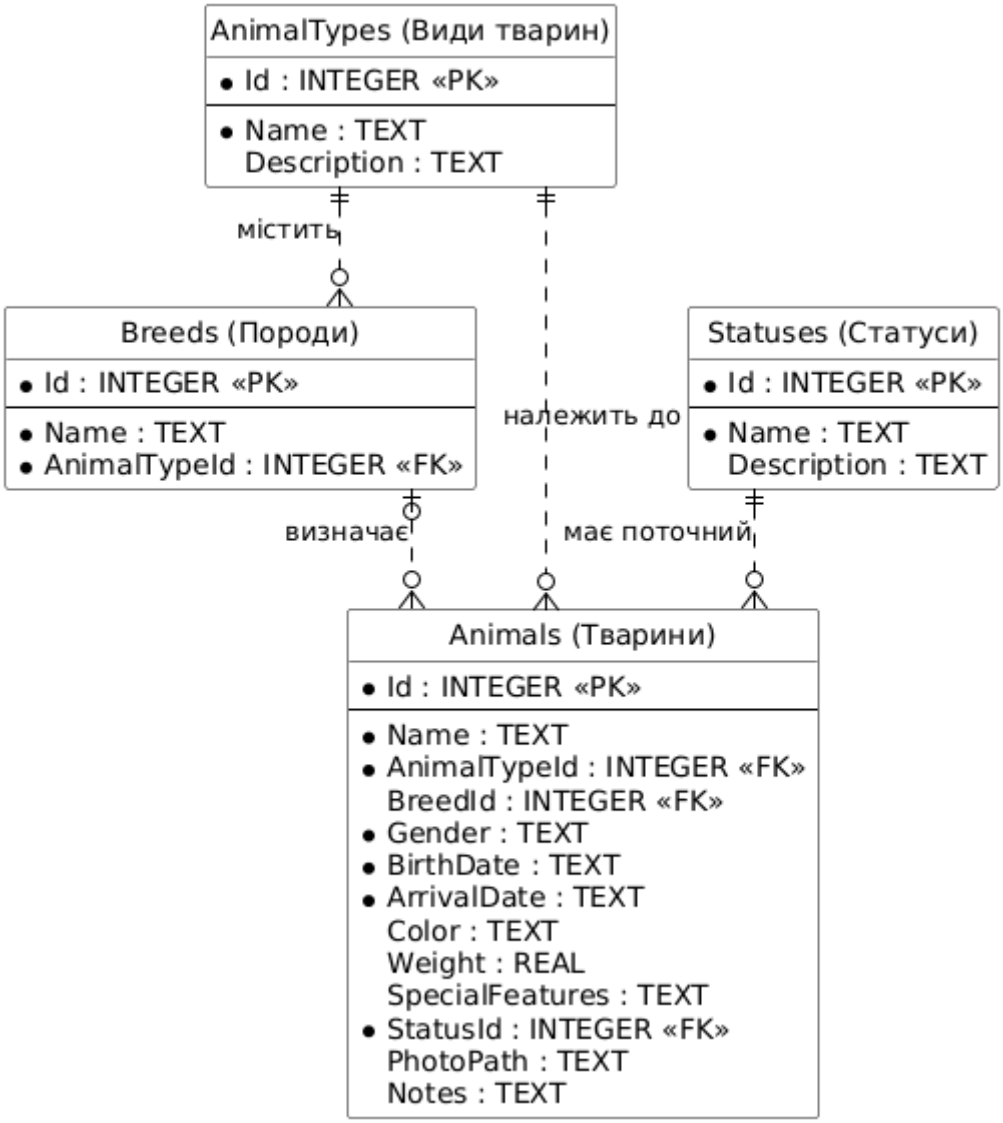

Рисунок 2.2 – ER-діаграма інформаційної системи обліку тварин у притулку

а) «Вид» – «Тварина». Цей зв'язок показує, що до одного біологічного виду може належати безліч зареєстрованих в установі тварин, але кожна конкретна тварина має належати лише до одного виду.

б) «Вид» – «Порода». Тут діє аналогічна логіка: один вид включає багато різних порід. Завдяки цьому зв'язку програма завжди знає, які саме породи належить пропонувати користувачеві при виборі, наприклад, кота.

в) «Порода» – «Тварина». До однієї породи може належати багато особин. Водночас логічна модель враховує реалії роботи волонтерів, тому

передбачає, що тварина може взагалі не мати визначеної породи (наприклад, звичайні вуличні тварини).

г) «Статус» – «Тварина». Один стан (скажімо, «У притулку») одночасно призначається багатьом підопічним. При зміні життєвих обставин тварини її зв'язок просто перемикається на інший статус із довідника.

Розроблена ER-модель дозволяє уникнути багаторазового введення одних і тих самих текстових значень (як-от назв порід чи статусів), що зменшує ймовірність помилок з боку персоналу.

2.3.2 Діаграма компонентів

Для відображення фізичної організації розробленого програмного забезпечення та взаємозв'язків між його окремими модулями було побудовано діаграму компонентів. Вона наочно демонструє практичну реалізацію обраної трьохшарової архітектури та розподіл відповідальності між частинами коду.

Структурно програма розділена на кілька логічних пакетів, кожен з яких виконує свою частину роботи:

а) Шар інтерфейсу користувача (UI Layer), що об'єднує всі візуальні форми, з якими безпосередньо взаємодіє персонал установи. До нього належать: MainForm – головне вікно з таблицею всіх підопічних та елементами пошуку; AnimalEditForm – вікно створення та зміни карток тварин; ReportForm – форма перегляду зведеної статистики та ExportForm – вікно налаштування параметрів вивантаження даних.

Ці компоненти займаються виключно збором інформації від користувача та відображенням результатів. Вони не містять в собі алгоритмів збереження чи обробки, а передають усі запити на виконання до наступного рівня.

б) Шар бізнес-логіки (Business Logic Layer) містить основні правила роботи системи. Головним компонентом тут виступає Repository, який приймає команди від усіх візуальних форм, перевіряє їх і формує відповідні завдання для бази даних. Також на цьому рівні розміщено ExportService –

окремий модуль, який відповідає за форматування інформації для подальшого запису в табличні файли. Для того щоб отримати необхідний масив записів для експорту, цей сервіс також звертається за допомогою до компонента Repository.

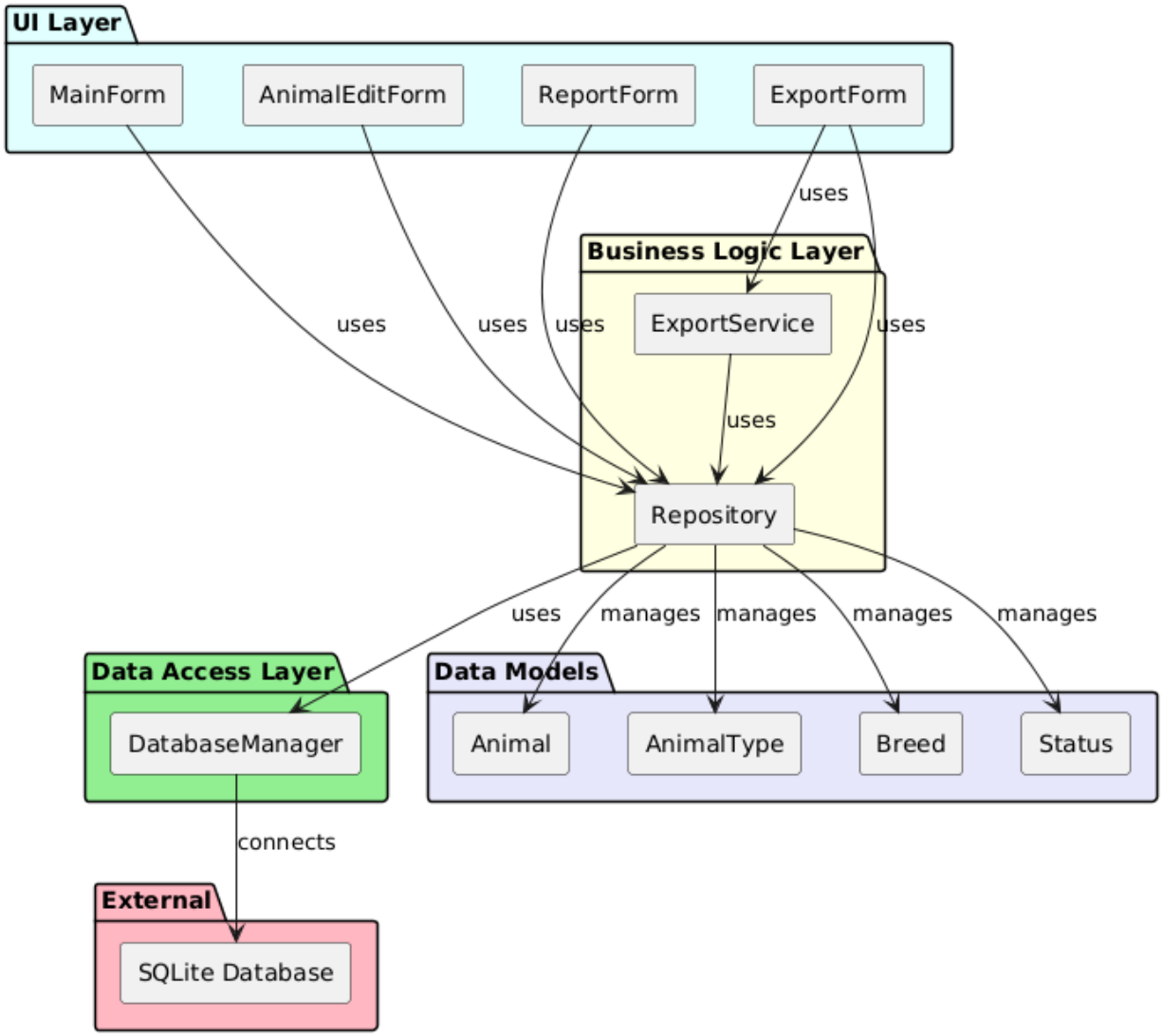

Рисунок 2.3 – Діаграма компонентів інформаційної системи обліку тварин у притулку

в) Моделі даних (Data Models) є допоміжним пакетом, що містить класи-контейнери, які описують сутності бази: Animal, AnimalType, Breed та Status. Репозиторій керує цими моделями: він створює такі об'єкти на основі

отриманих з бази рядків і передає їх на рівень інтерфейсу, щоб форми могли вивести інформацію на екран.

г) Шар доступу до даних (Data Access Layer) представлений єдиним класом DatabaseManager. Його завдання полягає у технічному обслуговуванні роботи з даними. Він приймає команди від репозиторію та безпосередньо підключається до зовнішнього сховища – локального файлу SQLite Database.

Такий поділ взаємозв'язків гарантує коректну ізоляцію компонентів. Наприклад, якщо в майбутньому виникне потреба повністю змінити дизайн вікон або додати нові кнопки, це ніяк не вплине на код роботи з базою даних, оскільки візуальний шар відділений від інфраструктури програми.

2.3.3 Діаграма послідовності

Щоб краще зрозуміти динаміку роботи розробленої програми та взаємодію її компонентів у часі, було побудовано діаграму послідовності. Як показовий приклад на схемі зображено один із найчастіших робочих сценаріїв – процес реєстрації нового підопічного.

Весь шлях від першого кліку до появи нового рядка в таблиці можна розділити на кілька послідовних етапів, що описані нижче.

Відкриття вікна та підготовка довідників. Процес починається, коли користувач натискає кнопку додавання на головній формі. Програма створює екземпляр вікна редагування. Перш ніж показатися на екрані, це вікно ініціалізує шар бізнес-логіки (репозиторій) та просить надати списки видів тварин і статусів. Репозиторій звертається до файлу бази даних SQLite, отримує необхідну інформацію і передає її назад. Завдяки цьому, коли форма нарешті відкривається, волонтер одразу бачить готові випадаючі списки для зручного вибору.

Введення інформації та перевірка відбуваються, коли користувач заповнює всі необхідні поля, за потреби додає фотографію та натискає кнопку збереження. Перед тим, як відправити інформацію далі, форма проводить валідацію: перевіряє, чи не залишилися порожніми обов'язкові параметри та

чи правильно обрані дати. Якщо все добре, візуальний компонент об’єднує зібрані дані в єдиний об’єкт тварини.

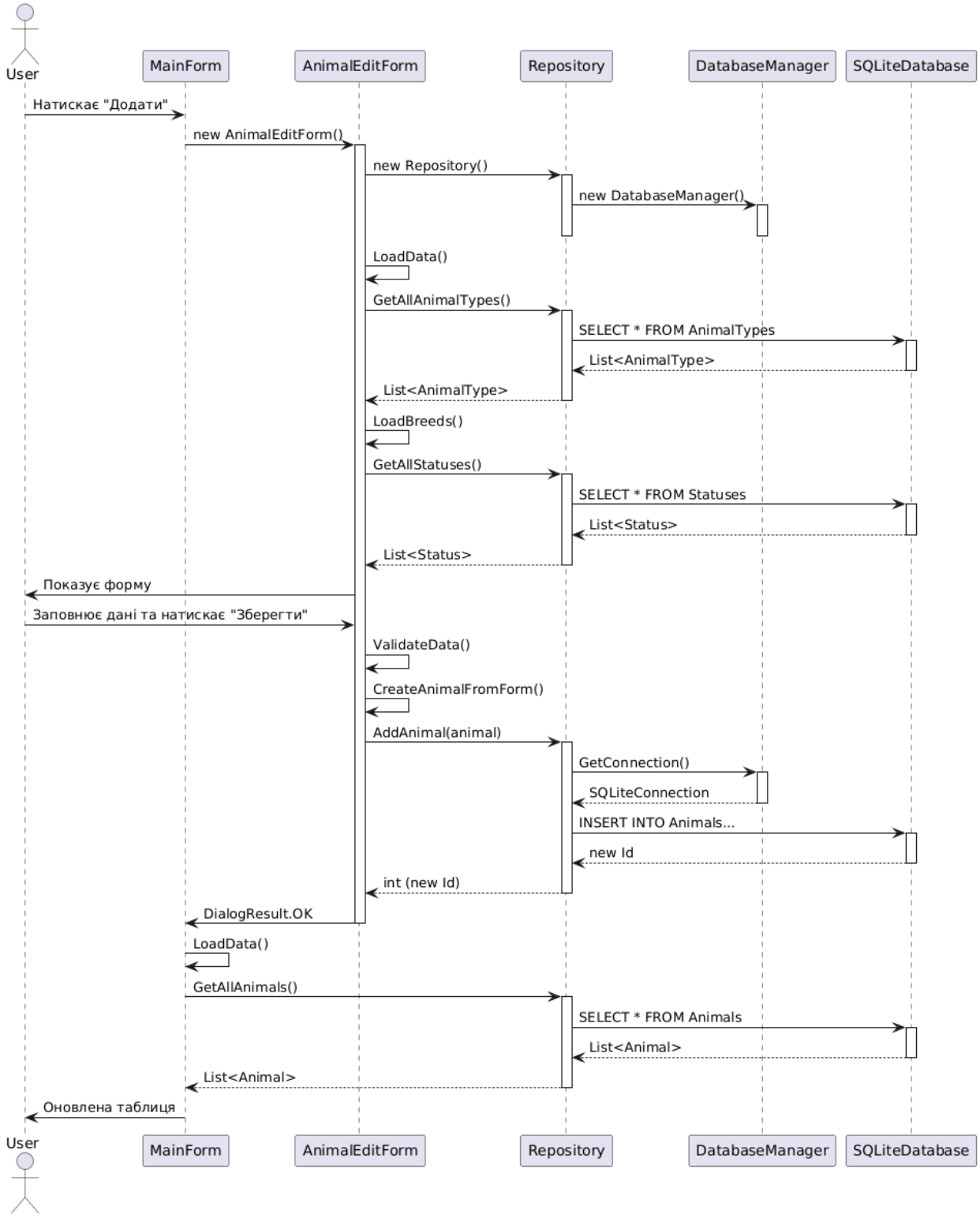

Рисунок 2.4 – Діаграма послідовності інформаційної системи обліку тварин у притулку (додавання нової тварини)

Взаємодія з базою даних відбувається, коли створений об'єкт передається до репозиторію. Останній звертається до менеджера бази даних для отримання активного підключення, після чого виконує команду додавання (INSERT) у таблицю Animals. База даних опрацьовує цей запит, присвоює запису унікальний ідентифікатор і повідомляє репозиторій про успішне виконання операції.

Отримавши підтвердження про збереження, вікно додавання закривається і передає головній формі сигнал успішного завершення (DialogResult.OK). Головна форма розуміє, що дані змінилися, тому самостійно ініціює запит до репозиторію для отримання свіжого переліку всіх підопічних. Відбувається завантаження оновленої інформації, таблиця перемальовується, і користувач бачить щойно створену картку серед інших. Таким чином, відбувається оновлення головного екрана.

2.4 Проєктування структури бази даних

Для забезпечення надійного збереження інформації та уникнення дублювання даних, база розроблена за принципами реляційної моделі. Уся інформація розподілена між кількома взаємопов'язаними таблицями. Головна ідея полягає у винесенні повторюваних значень у відокремлені довідники, що значно спрощує подальшу підтримку та розширення системи.

Розглянемо перелік спроектованих таблиць.

а) Таблиця AnimalTypes (Види тварин) зберігає базовий перелік видів тварин, якими опікується установа (наприклад, собаки, коти, птахи). Вона складається з унікального ідентифікатора, назви виду та необов'язкового текстового опису. Такий підхід дозволяє легко додавати нові категорії в майбутньому без необхідності змінювати архітектуру самої бази даних.

б) Таблиця Breeds (Породи). Оскільки кожна порода логічно належить до певного виду, ця таблиця безпосередньо пов'язана з попередньою. Вона містить власний ідентифікатор, назву породи та зовнішній ключ, який вказує на відповідний запис із таблиці видів. Завдяки цьому структурному рішенню

в інтерфейсі програми вдалося реалізувати залежне оновлення списків: при виборі певного виду користувачеві пропонуються виключно відповідні йому породи.

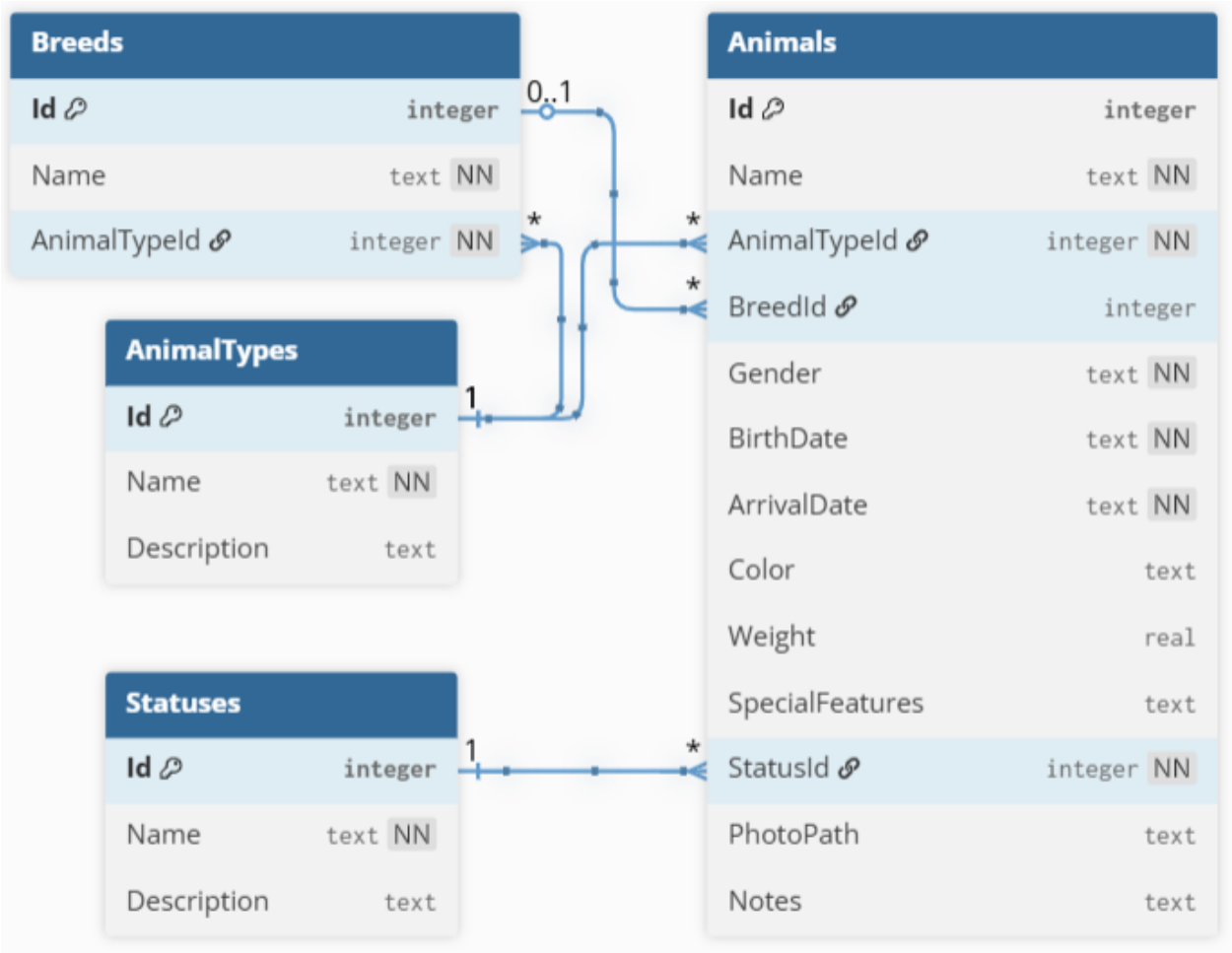

Рисунок 2.5 – Структура бази даних інформаційної системи обліку тварин у притулку

в) Таблиця Statuses (Статуси) зберігає можливі стани перебування підопічного (у притулку, на лікуванні, заброньована, прилаштована). Таблиця має просту структуру, що включає ідентифікатор, назву та опис. Використання окремого довідника для статусів замість звичайного текстового поля допомагає уникнути граматичних помилок під час ручного введення та значно спрощує математичне формування звітності.

г) Таблиця Animals (Тварини) є найбільшою таблицею системи, в якій зберігається вся детальна інформація про конкретних мешканців організації. Вона містить текстові поля для клички, статі, кольору та особливих прикмет, а також числове поле для фіксації ваги. Дати народження та надходження зберігаються у текстовому форматі за загальноприйнятим стандартом (рік-місяць-день), що є оптимальним підходом для обраної системи управління базами даних.

Окрім цього, таблиця містить зовнішні ключі, які посилаються на ідентифікатори виду, породи та поточного статусу. Важливою архітектурною особливістю є те, що поля породи, кольору, ваги, особливостей, шляху до фотографії та приміток налаштовані як такі, що можуть не містити даних (nullable). Це відповідає реальним умовам роботи волонтерів, адже під час прийому знайденої на вулиці тварини далеко не вся інформація про неї може бути відомою одразу.

2.5 Алгоритм роботи інформаційної системи

Життєвий цикл програми побудований за принципом очікування подій. Це означає, що додаток не виконує дії заздалегідь прописаним лінійним маршрутом, а реагує на конкретні команди користувача.

Робота починається з етапу ініціалізації, коли одразу після запуску система звертається до компонента керування базою даних для перевірки наявності локального файлу сховища. Якщо програма запускається вперше і цього файлу немає, він створюється автоматично разом із необхідними таблицями та початковими довідниками для видів, порід і статусів. Після успішного встановлення підключення завантажується головна візуальна форма, яка отримує весь наявний перелік тварин і відображає його на екрані.

Завершивши підготовку, програма переходить у стан спокою та починає чекати на дії адміністратора. Подальший розвиток подій залежить від того, якими елементами інтерфейсу скористається людина. Наприклад, якщо змінюється текст у полі пошуку або обирається інше значення у випадаючих

списках фільтрів, програма зчитує ці нові критерії, передає їх до репозиторію, отримує відфільтрований масив записів і перемальовує таблицю. У випадку, коли виникає необхідність додати нового підопічного або відредагувати існуючу картку, алгоритм відкриває відповідне вікно. Після заповнення інформації та спроби її збереження запускається механізм валідації. Якщо виявляються порожні обов'язкові поля або некоректні дати, процес призупиняється з виведенням повідомлення про помилку. Якщо ж інформація введена правильно, вона записується у файл бази, а головна форма самостійно оновлює таблицю.

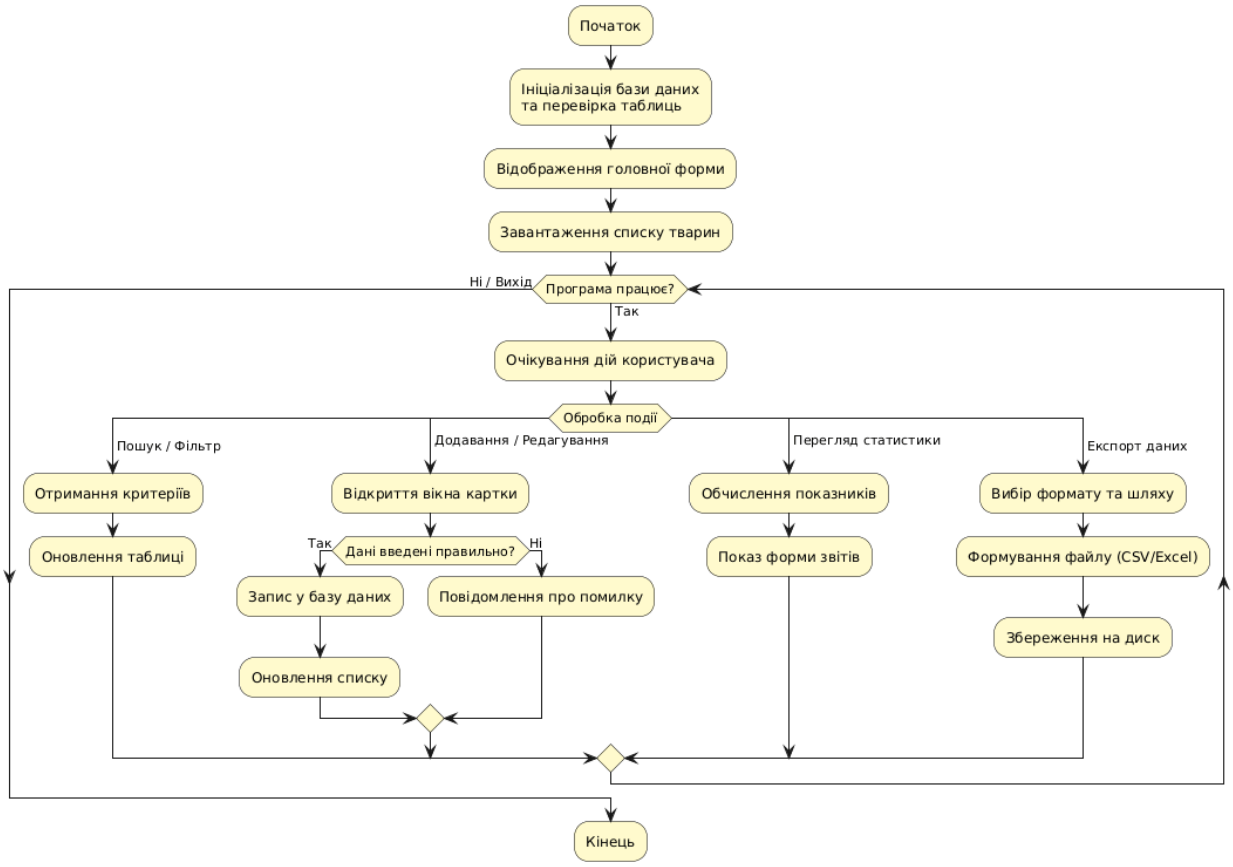

Рисунок 2.6 – Блок-схема алгоритму роботи інформаційної системи обліку тварин у притулку

Окрім роботи з картками, користувач може ініціювати генерацію звітів або експорт даних. При виклику статистики алгоритм збирає всі записи,

проводить необхідні математичні обчислення, такі як розрахунок середнього віку чи підрахунок за статусами, і виводить результати на екран. А під час експорту система запитує про бажаний формат файлу та місце його збереження, після чого формує готовий документ на основі активних записів.

Такий цикл обробки подій триває доти, доки користувач не вирішить завершити роботу і не натисне кнопку закриття головного вікна. Отримавши цю команду, алгоритм припиняє роботу графічного інтерфейсу, звільняє зайняту оперативну пам'ять, коректно закриває з'єднання з базою даних та повністю зупиняє виконання програми.

Для наочного представлення описаної логіки було підготовлено відповідний код для генерації блок-схеми алгоритму. Блоки початку та завершення на ній відображаються у вигляді овалів, що відповідає загальноприйнятим правилам оформлення.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ТВАРИН У ПРИТУЛКУ

3.1 Опис розробленої інформаційної системи

У результаті виконання етапів проєктування та написання програмного коду було створено готову до використання інформаційну систему. Її графічний інтерфейс побудований з урахуванням потреб працівників зоозахисних організацій та адаптований для зручної щоденної роботи. У цьому підрозділі розглянуто загальний вигляд програми та призначення її основних візуальних компонентів.

Під час запуску додатка відкривається головна форма, яка слугує основним робочим простором.

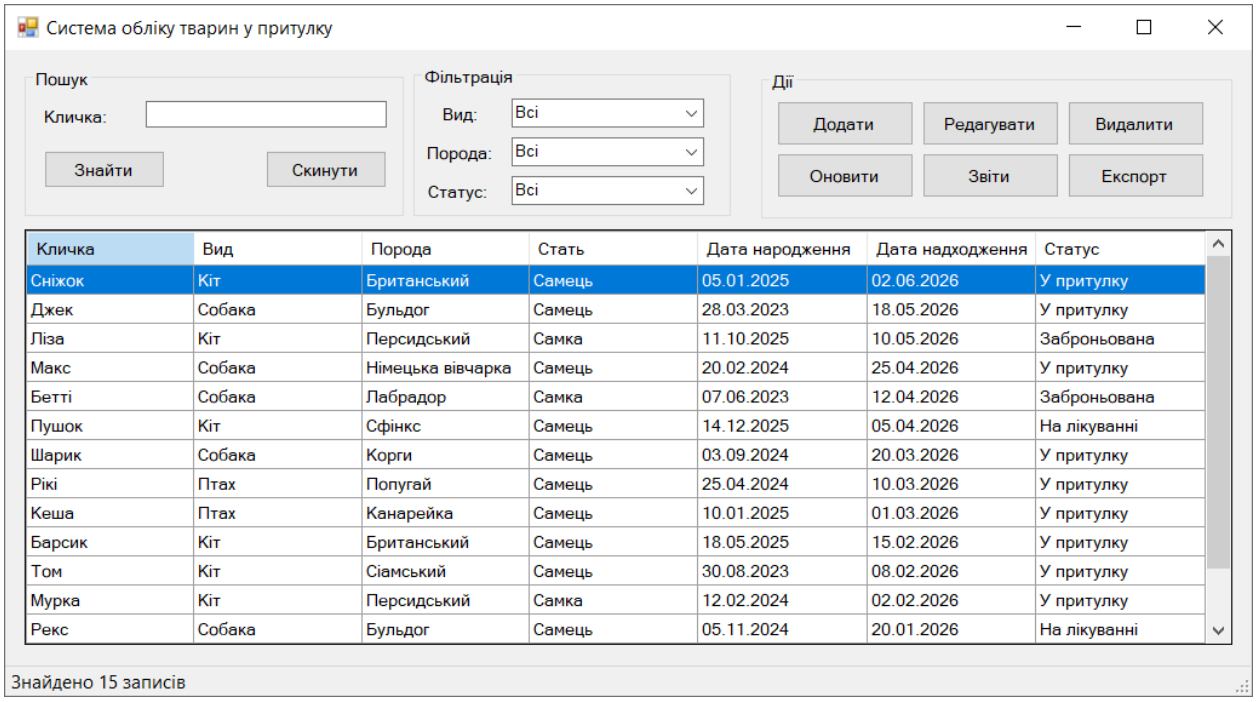

Рисунок 3.1 – Головний екран інформаційної системи

Візуально вона поділена на кілька функціональних зон, щоб не перевантажувати увагу користувача. У верхній частині розташована панель управління, яка містить інструменти для навігації базою даних. Тут

знаходяться випадаючі списки для вибору виду, породи та поточного статусу, а також текстове поле для пошуку за кличкою. З правого боку розміщено кнопки для виклику додаткових інструментів, таких як звіти та експорт.

Основну площу головного екрана займає велика таблиця. У ній відображається весь перелік тварин із детальною інформацією: унікальний номер, ім'я, біологічний вид, порода, стать, дати народження та надходження, вага, забарвлення, особливі прикмети та статус. Нижня частина вікна відведена під інформаційний рядок, де виводиться загальна кількість записів у базі.

Для перегляду детальної інформації про конкретну тварину або додавання нового запису розроблено окрему форму редагування.

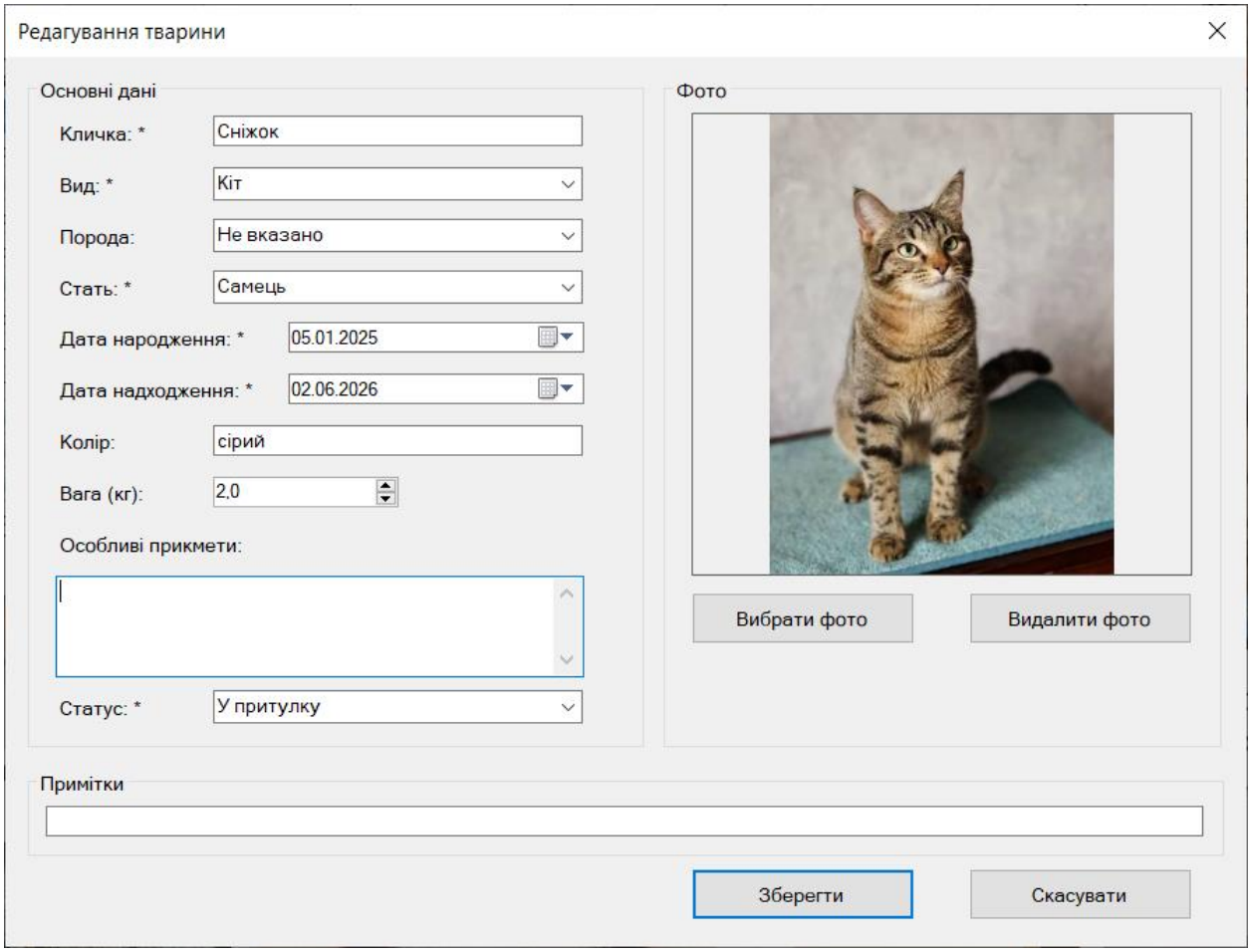

Рисунок 3.2 – Форма редагування інформації про тварину

Вона відкривається поверх головного вікна і містить структурований набір полів. Інтерфейс складається з текстових блоків для введення імені та прикмет, календарів для правильного вибору дат, а також списків для довідкових даних (стать, статус). Збоку передбачено спеціальну область для завантаження та відображення фотографії підопічного. Багаторядкове поле внизу форми призначене для збереження розгорнутих приміток або медичних спостережень.

Аналітична частина програми реалізована у вигляді форми звітів. Інтерфейс цього вікна створений для швидкого ознайомлення зі статистикою.

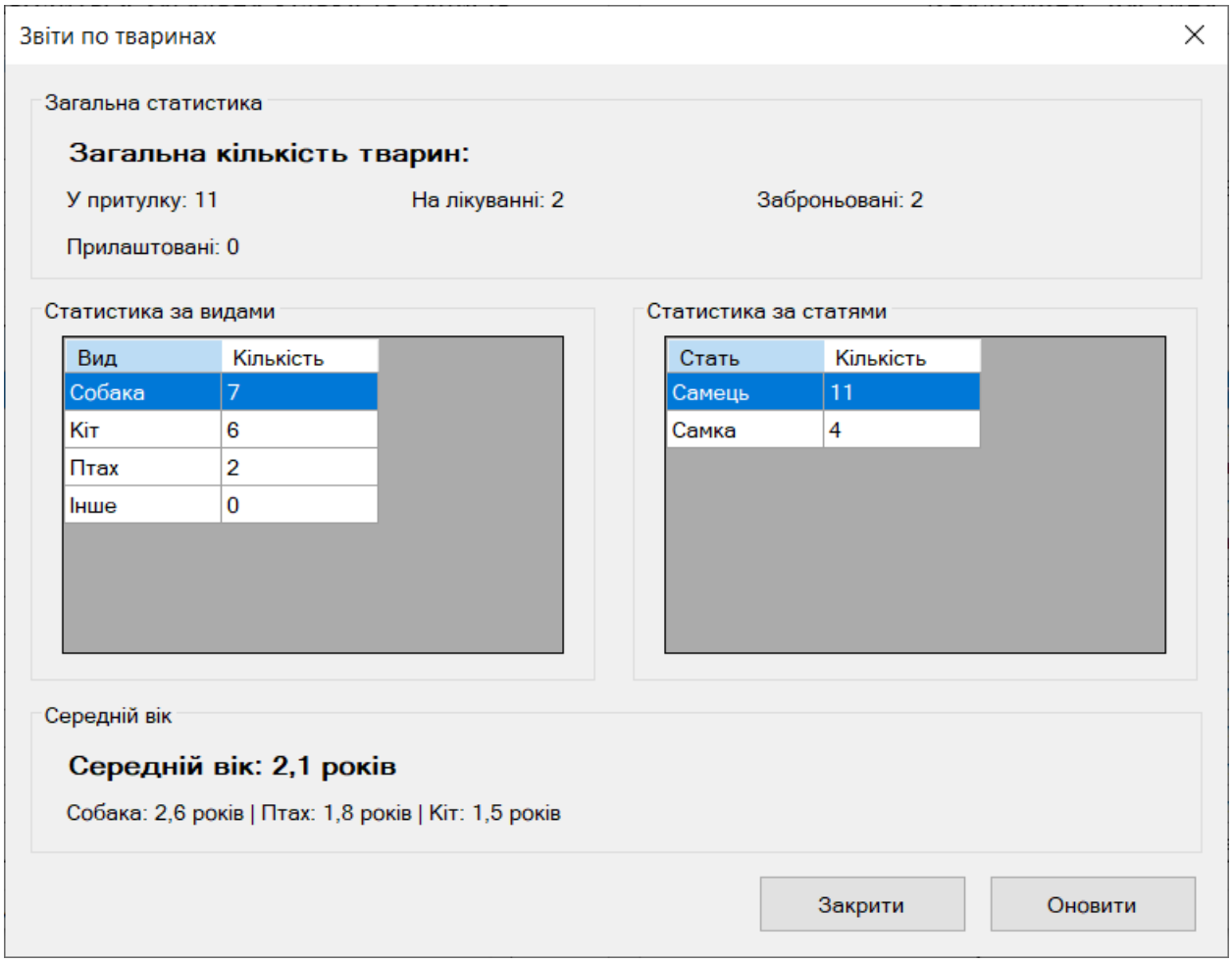

Рисунок 3.3 – Вікно перегляду зведеної статистики

Простір згруповано за смисловими блоками: загальна кількість тварин, таблиця розподілу за статусами перебування, розподіл за біологічними видами

та статтю. Окремим блоком виведено інформацію про середній вік мешканців. Усі дані подаються у вигляді компактних списків та числових показників, які легко сприймаються візуально.

Для взаємодії з іншими програмами передбачено вікно експорту. Воно має мінімалістичний дизайн і містить лише необхідні елементи управління. Користувач бачить кнопки для вибору бажаного формату збереження (електронна таблиця Excel або текстовий файл CSV).

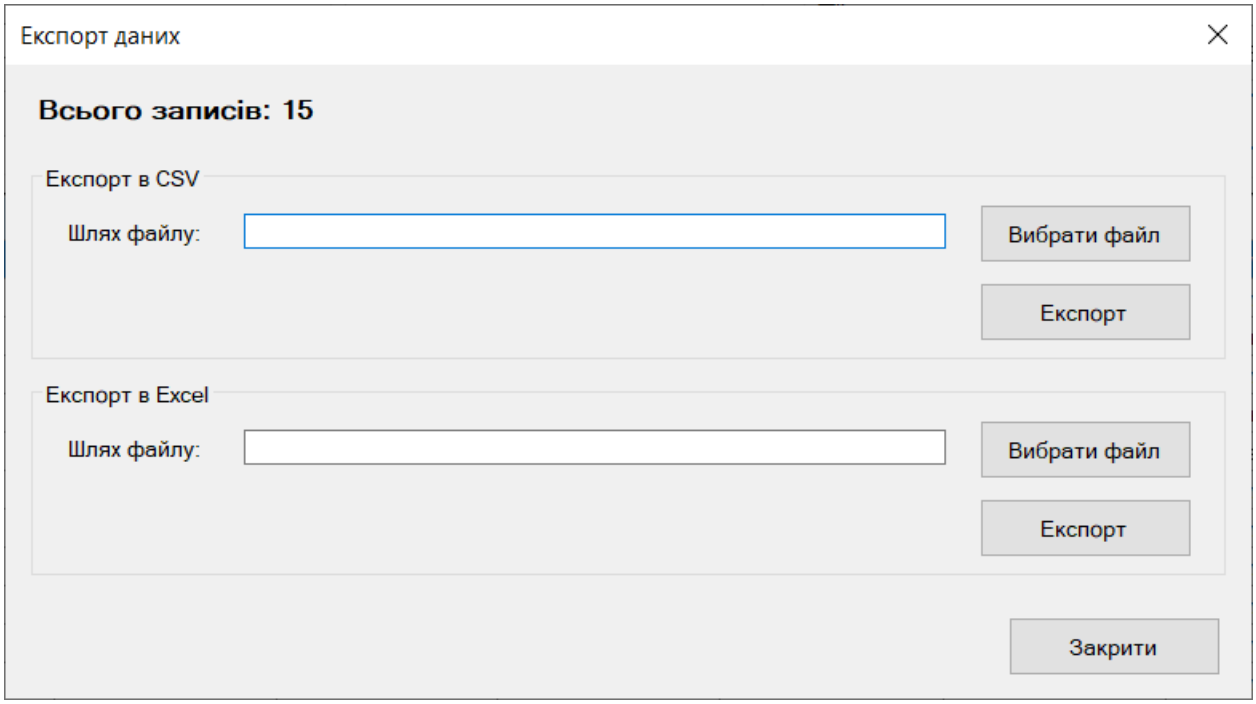

Рисунок 3.4 – Інтерфейс вікна експорту бази даних

Таким чином, головна форма забезпечує загальний огляд, форма редагування дозволяє працювати з деталями, а додаткові вікна вирішують вузькоспеціалізовані задачі аналітики та збереження.

3.2 Інструкція до використання інформаційної системи

Розроблена програма не потребує складного процесу встановлення чи налаштування сторонніх серверів. Для початку роботи достатньо перенести файли проєкту на комп'ютер та запустити основний виконуваний файл. Усі

необхідні операції зі створення бази даних система виконає самостійно під час першого запуску. Практичне використання програми базується на виконанні кількох простих алгоритмів.

Щоб швидко орієнтуватися у великому масиві записів, користувачеві доступні інструменти сортування на головному екрані:

а) Для пошуку конкретного підопічного достатньо ввести його ім'я в текстове поле. Таблиця автоматично відобразить лише ті рядки, що відповідають запиту.

б) Для відбору за категоріями необхідно скористатися випадаючими списками. Наприклад, якщо обрати вид «Кіт», сусідній список порід підлаштується під цей вибір. Додавши до цього статус «На лікуванні», можна отримати точний перелік котів, які наразі перебувають у ветеринарному блоці.

в) Щоб скасувати всі критерії і повернутися до загального переліку, передбачена спеціальна кнопка скидання параметрів пошуку.

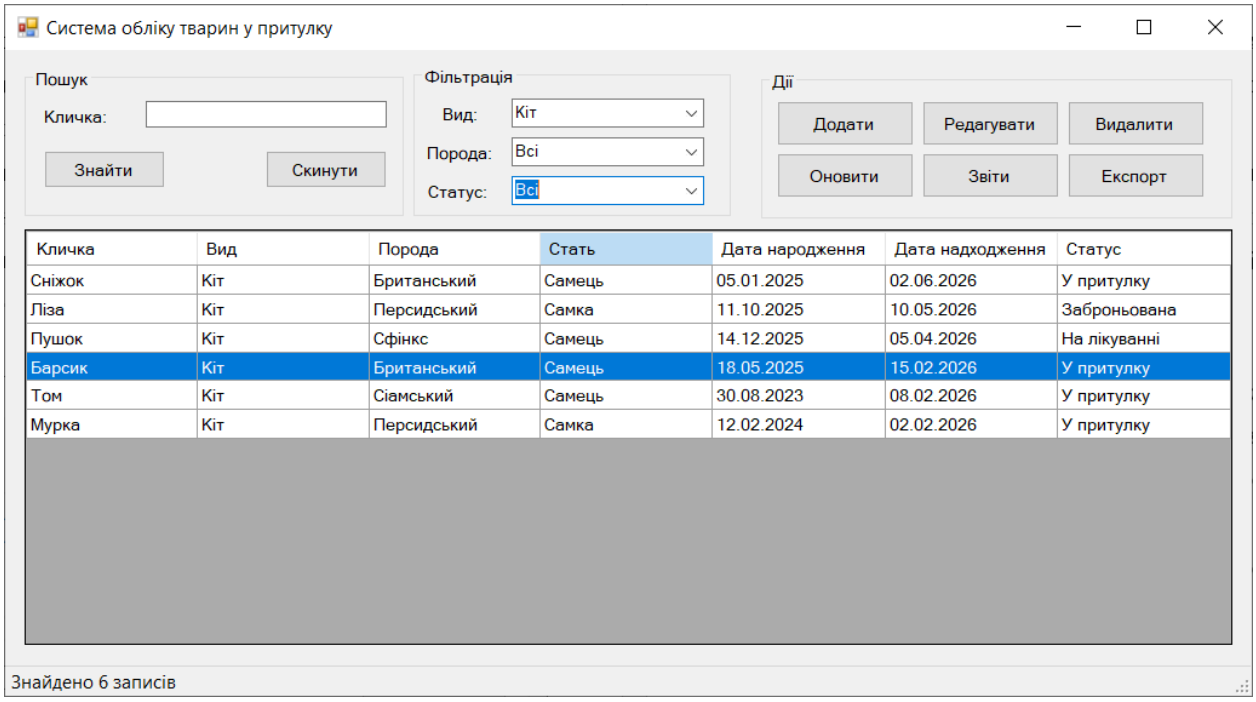

Рисунок 3.5 – Виконання фільтрації

Розглянемо процес додавання нового запису в інформаційну систему.

Процес реєстрації нового мешканця організації складається з таких кроків:

1. Натиснути кнопку додавання на панелі інструментів.

2. У формі, що з'явиться, заповнити обов'язкові параметри: ім'я, біологічний вид, стать та початковий статус.

3. За допомогою вбудованого календаря обрати правильні дати народження та надходження до установи.

4. За потреби вказати додаткові відомості (вагу, забарвлення, особливі прикмети) або написати текстові примітки.

5. Для прикріплення фотографії натиснути відповідну кнопку та обрати зображення на комп'ютері.

6. Підтвердити створення запису кнопкою збереження. Якщо всі вимоги щодо заповнення виконано, картка з'явиться у загальній таблиці.

Оскільки життєві обставини підопічних постійно змінюються, інформація потребує регулярної актуалізації. Щоб змінити дані (наприклад, перевести тварину до категорії прилаштованих), потрібно виділити її рядок у таблиці лівою кнопкою миші та натиснути кнопку редагування. Відкриється вікно з уже заповненими даними, де можна внести будь-які правки і зберегти результат.

У разі створення випадкового дубліката або помилкового запису його можна видалити. Після виділення рядка і натискання кнопки видалення програма обов'язково попросить додатково підтвердити цей намір.

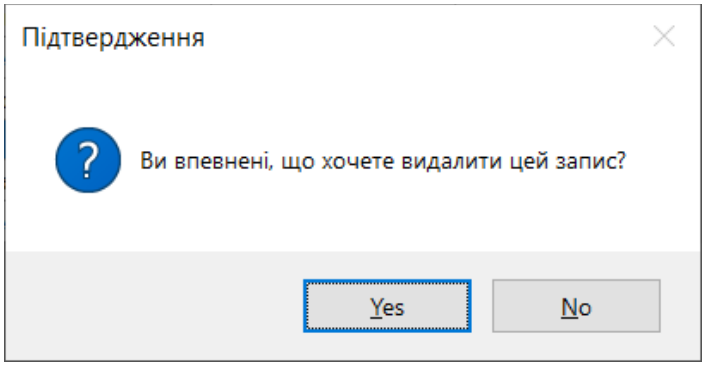

Рисунок 3.6 – Підтвердження видалення

Для отримання аналітичних даних про роботу установи потрібно натиснути кнопку формування звітів. У новому вікні програма самостійно підрахує і відобразить актуальну статистику. Якщо база даних була відредагована під час перегляду статистики, для перерахунку показників достатньо натиснути кнопку оновлення безпосередньо у вікні звітів.

Коли виникає необхідність зберегти інформацію для подальшого аналізу в інших додатках або для роздрукування, використовується функція експорту. Після натискання відповідної кнопки користувачеві потрібно обрати бажаний формат і вказати місце на диску, куди буде збережено готовий документ. Після завершення вивантаження система сповістить про успішне створення файлу.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було успішно вирішено актуальне завдання автоматизації роботи зоозахисних організацій. На початковому етапі розробки було проведено детальний аналіз процесів обліку тварин у притулках та досліджено недоліки існуючих програмних рішень. Це дозволило обґрунтувати необхідність створення власної інформаційної системи, яка не залежить від інтернет-з'єднання та повністю адаптована до реальних потреб вітчизняних волонтерів і адміністраторів притулків.

На основі сформованих вимог було спроєктовано логічну та фізичну структуру майбутньої програми. Використання класичної трьохшарової архітектури дозволило правильно відокремити візуальний інтерфейс від логіки та механізмів доступу до даних. Для надійного збереження інформації було розроблено реляційну базу даних на основі технології SQLite. Цей вибір виявився оптимальним, оскільки він забезпечує автономність програми та не вимагає складного налаштування серверного середовища з боку користувачів. Усі інформаційні сутності, такі як види, породи, статуси та самі картки тварин, були пов'язані між собою таким чином, щоб уникати дублювання тексту та гарантувати цілісність бази.

Практична реалізація системи здійснювалася мовою C# з використанням можливостей платформи .NET Framework та графічного фреймворку Windows Forms. Розроблений інтерфейс є україномовним і побудований за принципами доступності, щоб будь-який новий працівник міг швидко опанувати програму без попереднього навчання. Додаток забезпечує весь необхідний цикл роботи: від реєстрації нового підопічного та оновлення його життєвого статусу до складного пошуку карток за багатьма параметрами одночасно. Важливим досягненням є впровадження динамічних списків та автоматичної перевірки правильності введених даних, що значно знижує ймовірність людських помилок під час заповнення карток.

Окрім базового функціоналу збереження інформації, програма містить спеціальні інструменти для автоматичного формування зведеної статистики, підрахунку кількості підопічних за різними категоріями та визначення їхнього середнього віку. Наявність модуля експорту дозволяє легко вивантажувати базу даних у формати електронних таблиць та текстових документів для роботи в інших офісних додатках. У результаті було створено завершений і готовий до практичного використання програмний продукт.

Впровадження інформаційної системи здатне значно спростити ведення документообігу в притулку, зекономити час персоналу та допомогти організації зосередитися на своїй головній меті – порятунку тварин та швидкому пошуку для них нових родин.

ПЕРЕЛІК ПОСИЛАНЬ

1. Чотири лапи й мокрий ніс [Електронний ресурс] – Режим доступу до ресурсу: https://odesa.name/uk/eternal-4982-chotyry-lapy-j-mokryj-nis-top-7-odeskyh-prytulkiv-dlya-tvaryn

2. Microsoft Excel [Електронний ресурс] – Режим доступу до ресурсу: https://excel.cloud.microsoft

3. LibreOffice Calc [Електронний ресурс] – Режим доступу до ресурсу: https://www.libreoffice.org/

4. Shelterluv. The easy-to-use, modern shelter software you've been looking for [Електронний ресурс] – Режим доступу до ресурсу: https://www.shelterluv.com/

5. What is ShelterLuv? [Електронний ресурс] – Режим доступу до ресурсу: https://sumble.com/tech/shelterluv

6. Efficient Shelter Management Starts Here [Електронний ресурс] – Режим доступу до ресурсу: https://www.24pet.com/products/petpoint

7. PetPoint 2026 [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwareadvice.com/data-management/petpoints-profile/

8. sheltermanager.com [Електронний ресурс] – Режим доступу до ресурсу: https://sheltermanager.com/site/en_home.html

9. Animal Shelter Manager [Електронний ресурс] – Режим доступу до ресурсу: https://petcolove.org/shelter-partners/animal-shelter-management-certificate/

10. C#. The modern, innovative, open-source programming language for building all your apps. [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/languages/csharp

11. C# language reference [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/csharp/language-reference/

12. .NET Framework is a Windows-only version of .NET for building client and server applications [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/download/dotnet-framework

13. .NET Framework documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/dotnet/framework/
14. SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://sqlite.org/
15. System.Data.SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://system.data.sqlite.org/home/doc/trunk/www/index.md

Додаток А

Програмний код

MainForm.cs

```
using System;
using System.Windows.Forms;
using AnimalShelterSystem.Data;
using AnimalShelterSystem.Models;

namespace AnimalShelterSystem.Forms
{
    // Головна форма програми для обліку тварин у притулку
    public partial class MainForm : Form
    {
        // Екземпляр репозиторію для доступу до даних
        private Repository _repository;

        // Конструктор форми
        public MainForm()
        {
            InitializeComponent();
            _repository = new Repository();
        }

        // Метод завантаження форми
        private void MainForm_Load(object sender, EventArgs e)
        {
            // Завантаження даних при запуску форми
            LoadData();
            LoadFilters();
        }

        // Метод завантаження даних в таблицю
        private void LoadData()
        {
            try
            {
                // Отримання списку всіх тварин
                var animals = _repository.GetAllAnimals();

                // Очищення таблиці
                dgvAnimals.DataSource = null;
                dgvAnimals.Rows.Clear();

                // Заповнення таблиці даними
                foreach (var animal in animals)
                {
                    dgvAnimals.Rows.Add(
                        animal.Id,
                        animal.Name,
                        animal.AnimalTypeName,
```

```
                        animal.BreedName,
                        animal.Gender,
                        animal.BirthDate.ToString("dd.MM.yyyy"),
animal.ArrivalDate.ToString("dd.MM.yyyy"),
                        animal.StatusName
                    );
                }

                UpdateStatus($"Завантажено {animals.Count}
записів");
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження даних:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Метод завантаження даних для фільтрів
        private void LoadFilters()
        {
            try
            {
                // Завантаження видів тварин
                var animalTypes =
_repository.GetAllAnimalTypes();
                cmbFilterAnimalType.DisplayMember = "Name";
                cmbFilterAnimalType.ValueMember = "Id";
                cmbFilterAnimalType.Items.Clear();
                cmbFilterAnimalType.Items.Add("Всі");
                foreach (var type in animalTypes)
                {
                    cmbFilterAnimalType.Items.Add(type);
                }
                cmbFilterAnimalType.SelectedIndex = 0;

                // Завантаження статусів
                var statuses = _repository.GetAllStatuses();
                cmbFilterStatus.DisplayMember = "Name";
                cmbFilterStatus.ValueMember = "Id";
                cmbFilterStatus.Items.Clear();
                cmbFilterStatus.Items.Add("Всі");
                foreach (var status in statuses)
                {
                    cmbFilterStatus.Items.Add(status);
                }
                cmbFilterStatus.SelectedIndex = 0;

                // Породи будуть завантажені при виборі виду
                cmbFilterBreed.Items.Clear();
```

```
                cmbFilterBreed.Items.Add("Всі");
                cmbFilterBreed.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження фільтрів:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Обробник кнопки "Додати"
        private void btnAdd_Click(object sender, EventArgs e)
        {
            // Відкриття форми для додавання нової тварини
            AnimalEditForm editForm = new AnimalEditForm();
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }

        // Обробник кнопки "Редагувати"
        private void btnEdit_Click(object sender, EventArgs e)
        {
            // Перевірка вибору запису
            if (dgvAnimals.SelectedRows.Count == 0)
            {
                MessageBox.Show("Виберіть запис для
редагування", "Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            // Отримання ідентифікатора вибраної тварини
            int animalId =
Convert.ToInt32(dgvAnimals.SelectedRows[0].Cells[0].Value);

            // Відкриття форми для редагування
            AnimalEditForm editForm = new
AnimalEditForm(animalId);
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadData();
            }
        }

        // Обробник кнопки "Видалити"
        private void btnDelete_Click(object sender, EventArgs e)
        {
            // Перевірка вибору запису
```

```
            if (dgvAnimals.SelectedRows.Count == 0)
            {
                MessageBox.Show("Виберіть запис для видалення",
"Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            // Підтвердження видалення
            DialogResult result = MessageBox.Show("Ви впевнені,
що хочете видалити цей запис?",
                "Підтвердження", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

            if (result == DialogResult.Yes)
            {
                try
                {
                    // Отримання ідентифікатора вибраної тварини
                    int animalId =
Convert.ToInt32(dgvAnimals.SelectedRows[0].Cells[0].Value);

                    // Видалення запису
                    bool deleted =
_repository.DeleteAnimal(animalId);

                    if (deleted)
                    {
                        MessageBox.Show("Запис успішно
видалено", "Успіх",
                            MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        LoadData();
                    }
                    else
                    {
                        MessageBox.Show("Не вдалося видалити
запис", "Помилка",
                            MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка видалення:
{ex.Message}", "Помилка",
                        MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
        }
```

```
        // Обробник кнопки "Оновити"
        private void btnRefresh_Click(object sender, EventArgs
e)
        {
            LoadData();
        }

        // Обробник кнопки "Звіти"
        private void btnReports_Click(object sender, EventArgs
e)
        {
            // Відкриття форми звітів
            ReportForm reportForm = new ReportForm();
            reportForm.ShowDialog();
        }

        // Обробник кнопки "Експорт"
        private void btnExport_Click(object sender, EventArgs e)
        {
            // Відкриття форми експорту
            ExportForm exportForm = new ExportForm();
            exportForm.ShowDialog();
        }

        // Обробник кнопки "Знайти"
        private void btnSearch_Click(object sender, EventArgs e)
        {
            SearchAnimals();
        }

        // Обробник кнопки "Скинути пошук"
        private void btnResetSearch_Click(object sender,
EventArgs e)
        {
            txtSearchName.Clear();
            cmbFilterAnimalType.SelectedIndex = 0;
            cmbFilterBreed.SelectedIndex = 0;
            cmbFilterStatus.SelectedIndex = 0;
            LoadData();
        }

        // Обробник зміни фільтру за видом
        private void
cmbFilterAnimalType_SelectedIndexChanged(object sender,
EventArgs e)
        {
            LoadBreedsForFilter();
            ApplyFilters();
        }

        // Обробник зміни фільтру за породою
```

```
        private void cmbFilterBreed_SelectedIndexChanged(object
sender, EventArgs e)
        {
            ApplyFilters();
        }

        // Обробник зміни фільтру за статусом
        private void cmbFilterStatus_SelectedIndexChanged(object
sender, EventArgs e)
        {
            ApplyFilters();
        }

        // Метод завантаження порід для фільтру
        private void LoadBreedsForFilter()
        {
            try
            {
                cmbFilterBreed.DisplayMember = "Name";
                cmbFilterBreed.ValueMember = "Id";
                cmbFilterBreed.Items.Clear();
                cmbFilterBreed.Items.Add("Всі");

                // Якщо вибрано конкретний вид
                if (cmbFilterAnimalType.SelectedIndex > 0)
                {
                    AnimalType selectedType =
(AnimalType)cmbFilterAnimalType.SelectedItem;
                    var breeds =
_repository.GetBreedsByAnimalType(selectedType.Id);
                    foreach (var breed in breeds)
                    {
                        cmbFilterBreed.Items.Add(breed);
                    }
                }

                cmbFilterBreed.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження порід:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Метод пошуку тварин
        private void SearchAnimals()
        {
            try
            {
                string searchName = txtSearchName.Text.Trim();
```

```
                int? breedId = null;
                int? animalTypeId = null;
                int? statusId = null;

                // Отримання значень фільтрів
                if (cmbFilterBreed.SelectedIndex > 0)
                {
                    Breed selectedBreed =
(Breed)cmbFilterBreed.SelectedItem;
                    breedId = selectedBreed.Id;
                }

                if (cmbFilterAnimalType.SelectedIndex > 0)
                {
                    AnimalType selectedType =
(AnimalType)cmbFilterAnimalType.SelectedItem;
                    animalTypeId = selectedType.Id;
                }

                if (cmbFilterStatus.SelectedIndex > 0)
                {
                    Status selectedStatus =
(Status)cmbFilterStatus.SelectedItem;
                    statusId = selectedStatus.Id;
                }

                // Виконання пошуку
                var animals =
_repository.SearchAnimals(searchName, breedId, animalTypeId,
statusId);

                // Очищення таблиці
                dgvAnimals.DataSource = null;
                dgvAnimals.Rows.Clear();

                // Заповнення таблиці результатами
                foreach (var animal in animals)
                {
                    dgvAnimals.Rows.Add(
                        animal.Id,
                        animal.Name,
                        animal.AnimalTypeName,
                        animal.BreedName,
                        animal.Gender,
                        animal.BirthDate.ToString("dd.MM.yyyy"),

animal.ArrivalDate.ToString("dd.MM.yyyy"),
                        animal.StatusName
                    );
                }
```

```
                UpdateStatus($"Знайдено {animals.Count}
записів");
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка пошуку: {ex.Message}",
"Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Метод застосування фільтрів
        private void ApplyFilters()
        {
            SearchAnimals();
        }

        // Метод оновлення статусу в рядку стану
        private void UpdateStatus(string message)
        {
            lblStatus.Text = message;
        }
    }
}
```

AnimalEditForm.cs

```
using System;
using System.Windows.Forms;
using System.Drawing;
using AnimalShelterSystem.Data;
using AnimalShelterSystem.Models;

namespace AnimalShelterSystem.Forms
{
    // Форма для додавання та редагування тварин
    public partial class AnimalEditForm : Form
    {
        // Екземпляр репозиторію для доступу до даних
        private Repository _repository;

        // Ідентифікатор тварини (null для додавання, має
значення для редагування)
        private int? _animalId;

        // Шлях до фотографії
        private string _photoPath;

        // Конструктор для додавання нової тварини
        public AnimalEditForm()
        {
            InitializeComponent();
```

```
            _repository = new Repository();
            _animalId = null;
            _photoPath = null;
            LoadData();
        }

        // Конструктор для редагування існуючої тварини
        public AnimalEditForm(int animalId)
        {
            InitializeComponent();
            _repository = new Repository();
            _animalId = animalId;
            _photoPath = null;
            LoadData();
            LoadAnimalData();
        }

        // Метод завантаження даних в елементи управління
        private void LoadData()
        {
            try
            {
                // Завантаження видів тварин
                var animalTypes =
_repository.GetAllAnimalTypes();
                cmbAnimalType.DisplayMember = "Name";
                cmbAnimalType.ValueMember = "Id";
                cmbAnimalType.Items.Clear();
                foreach (var type in animalTypes)
                {
                    cmbAnimalType.Items.Add(type);
                }

                // Завантаження статів
                var statuses = _repository.GetAllStatuses();
                cmbStatus.DisplayMember = "Name";
                cmbStatus.ValueMember = "Id";
                cmbStatus.Items.Clear();
                foreach (var status in statuses)
                {
                    cmbStatus.Items.Add(status);
                }

                // Завантаження статей
                cmbGender.Items.Clear();
                cmbGender.Items.Add("Самець");
                cmbGender.Items.Add("Самка");

                // Встановлення дати надходження за
замовчуванням - сьогодні
                dtpArrivalDate.Value = DateTime.Today;
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження даних:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Метод завантаження даних тварини для редагування
        private void LoadAnimalData()
        {
            try
            {
                if (_animalId.HasValue)
                {
                    Animal animal =
_repository.GetAnimalById(_animalId.Value);

                    if (animal != null)
                    {
                        // Заповнення полів даними
                        txtName.Text = animal.Name;

                        // Вибір виду
                        foreach (AnimalType item in
cmbAnimalType.Items)
                        {
                            if (item.Id == animal.AnimalTypeId)
                            {
                                cmbAnimalType.SelectedItem =
item;
                                break;
                            }
                        }

                        // Вибір породи
                        LoadBreeds();
                        if (animal.BreedId.HasValue)
                        {
                            foreach (Breed item in
cmbBreed.Items)
                            {
                                if (item.Id ==
animal.BreedId.Value)
                                {
                                    cmbBreed.SelectedItem =
item;
                                    break;
                                }
                            }
                        }
```

```
                        // Вибір статі
                        cmbGender.Text = animal.Gender;

                        // Дати
                        dtpBirthDate.Value = animal.BirthDate;
                        dtpArrivalDate.Value =
animal.ArrivalDate;

                        // Інші поля
                        txtColor.Text = animal.Color;
                        if (animal.Weight.HasValue)
                        {
                            nudWeight.Value =
animal.Weight.Value;
                        }
                        txtSpecialFeatures.Text =
animal.SpecialFeatures;

                        // Вибір статусу
                        foreach (Status item in cmbStatus.Items)
                        {
                            if (item.Id == animal.StatusId)
                            {
                                cmbStatus.SelectedItem = item;
                                break;
                            }
                        }

                        // Фото
                        _photoPath = animal.PhotoPath;
                        LoadPhoto();

                        // Примітки
                        txtNotes.Text = animal.Notes;

                        // Зміна заголовка форми
                        this.Text = "Редагування тварини";
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження даних
тварини: {ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Обробник зміни виду тварини
        private void cmbAnimalType_SelectedIndexChanged(object
sender, EventArgs e)
        {
```

```
            LoadBreeds();
        }

        // Метод завантаження порід для вибраного виду
        private void LoadBreeds()
        {
            try
            {
                cmbBreed.DisplayMember = "Name";
                cmbBreed.ValueMember = "Id";
                cmbBreed.Items.Clear();
                cmbBreed.Items.Add("Не вказано");

                if (cmbAnimalType.SelectedItem is AnimalType
selectedType)
                {
                    var breeds =
_repository.GetBreedsByAnimalType(selectedType.Id);
                    foreach (var breed in breeds)
                    {
                        cmbBreed.Items.Add(breed);
                    }
                }

                cmbBreed.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка завантаження порід:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        // Обробник кнопки вибору фото
        private void btnSelectPhoto_Click(object sender,
EventArgs e)
        {
            if (ofdPhoto.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    _photoPath = ofdPhoto.FileName;
                    LoadPhoto();
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка завантаження фото:
{ex.Message}", "Помилка",
                        MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
```

```
            }
        }

        // Обробник кнопки видалення фото
        private void btnRemovePhoto_Click(object sender,
EventArgs e)
        {
            _photoPath = null;
            pbPhoto.Image = null;
        }

        // Метод завантаження фото в PictureBox
        private void LoadPhoto()
        {
            try
            {
                if (!string.IsNullOrEmpty(_photoPath) &&
System.IO.File.Exists(_photoPath))
                {
                    pbPhoto.Image = Image.FromFile(_photoPath);
                }
                else
                {
                    pbPhoto.Image = null;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Помилка відображення фото:
{ex.Message}", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                pbPhoto.Image = null;
            }
        }

        // Обробник кнопки Зберегти
        private void btnSave_Click(object sender, EventArgs e)
        {
            if (ValidateData())
            {
                try
                {
                    Animal animal = CreateAnimalFromForm();

                    if (_animalId.HasValue)
                    {
                        // Редагування існуючої тварини
                        animal.Id = _animalId.Value;
                        bool updated =
_repository.UpdateAnimal(animal);

                        if (updated)
```

```
                        {
                            MessageBox.Show("Дані успішно
оновлено!", "Успіх",
                                MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.DialogResult = DialogResult.OK;
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("Не вдалося оновити
дані", "Помилка",
                                MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        }
                    }
                    else
                    {
                        // Додавання нової тварини
                        int newId =
_repository.AddAnimal(animal);

                        if (newId > 0)
                        {
                            MessageBox.Show("Тварину успішно
додано!", "Успіх",
                                MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.DialogResult = DialogResult.OK;
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("Не вдалося додати
тварину", "Помилка",
                                MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Помилка збереження даних:
{ex.Message}", "Помилка",
                        MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
        }

        // Обробник кнопки Скасувати
        private void btnCancel_Click(object sender, EventArgs e)
```

```
        {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }

        // Метод перевірки даних
        private bool ValidateData()
        {
            // Перевірка клички
            if (string.IsNullOrWhiteSpace(txtName.Text))
            {
                MessageBox.Show("Введіть кличку тварини",
"Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                txtName.Focus();
                return false;
            }

            // Перевірка виду
            if (cmbAnimalType.SelectedItem == null)
            {
                MessageBox.Show("Виберіть вид тварини",
"Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                cmbAnimalType.Focus();
                return false;
            }

            // Перевірка статі
            if (string.IsNullOrWhiteSpace(cmbGender.Text))
            {
                MessageBox.Show("Виберіть стать тварини",
"Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                cmbGender.Focus();
                return false;
            }

            // Перевірка дати народження
            if (dtpBirthDate.Value > DateTime.Today)
            {
                MessageBox.Show("Дата народження не може бути в
майбутньому", "Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                dtpBirthDate.Focus();
                return false;
            }
```

```
            // Перевірка дати надходження
            if (dtpArrivalDate.Value > DateTime.Today)
            {
                MessageBox.Show("Дата надходження не може бути в
майбутньому", "Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                dtpArrivalDate.Focus();
                return false;
            }

            // Перевірка дати надходження не раніше дати
народження
            if (dtpArrivalDate.Value < dtpBirthDate.Value)
            {
                MessageBox.Show("Дата надходження не може бути
раніше дати народження", "Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                dtpArrivalDate.Focus();
                return false;
            }

            // Перевірка статусу
            if (cmbStatus.SelectedItem == null)
            {
                MessageBox.Show("Виберіть статус тварини",
"Попередження",
                    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                cmbStatus.Focus();
                return false;
            }

            return true;
        }

        // Метод створення об'єкту Animal з даних форми
        private Animal CreateAnimalFromForm()
        {
            Animal animal = new Animal();

            animal.Name = txtName.Text.Trim();

            if (cmbAnimalType.SelectedItem is AnimalType
selectedType)
            {
                animal.AnimalTypeId = selectedType.Id;
            }

            if (cmbBreed.SelectedItem is Breed selectedBreed &&
cmbBreed.SelectedIndex > 0)
```

```
            {
                animal.BreedId = selectedBreed.Id;
            }
            else
            {
                animal.BreedId = null;
            }

            animal.Gender = cmbGender.Text;
            animal.BirthDate = dtpBirthDate.Value;
            animal.ArrivalDate = dtpArrivalDate.Value;
            animal.Color =
string.IsNullOrWhiteSpace(txtColor.Text) ? null :
txtColor.Text.Trim();
            animal.Weight = nudWeight.Value > 0 ?
nudWeight.Value : (decimal?)null;
            animal.SpecialFeatures =
string.IsNullOrWhiteSpace(txtSpecialFeatures.Text) ? null :
txtSpecialFeatures.Text.Trim();

            if (cmbStatus.SelectedItem is Status selectedStatus)
            {
                animal.StatusId = selectedStatus.Id;
            }

            animal.PhotoPath = _photoPath;
            animal.Notes =
string.IsNullOrWhiteSpace(txtNotes.Text) ? null :
txtNotes.Text.Trim();

            return animal;
        }
    }
}
```