Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА-ПОШУКОВА СИСТЕМА ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ОБМІНУ ДОМАШНІМИ УЛЮБЛЕНЦЯМИ»

Виконав: студент 4 курсу, групи 2ІСТ-21б спеціальності 126 – «Інформаційні системи та технології»
_____ Олег ВОЛОЩУК
Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«03» 06 2025 р.
Рецензент: к.т.н., доц. каф. КСУ
_____ Олексій СІЛАГІН
«07» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«06» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
"28" 03 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Волощук Олегу Володимировичу

1. Тема роботи: «Інформаційна-пошукова система для підтримки прийняття рішень з обміну домашніми улюбленцями»
керівник роботи: Козачко О.М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «20» 03 2025 року № 97
2. Термін подання студентом роботи 21.05.25
3. Вихідні дані до роботи:
1) Дані про існуючі платформи для адопції;
2) Інформація про поточну ситуацію з безпритульними тваринами;
3) Типові характеристики тварин зібрані на основі публічних оголошень
4. Зміст текстової частини:
1) Аналіз проблематики процесів адопції в Україні;
2) Вибір оптимальних інформаційних технологій для розробки;
3) Розроблення інформаційно-пошукової системи для підтримки прийняття рішень з обміну домашніми улюбленцями;
5. Перелік ілюстративного матеріалу:
1) Діаграма архітектури MVVM;
2) Структура роботи інформаційної системи;
3) Блок-схема алгоритму роботи системи;
4) Діаграма варіантів використання системи;
5) Тестування роботи системи.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.03.25

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	10.04	вик
2	Порівняльний аналіз проблематики	10.04	20.04	вик
3	Вибір оптимальних інформаційних технологій	20.04	30.04	вик
4	Розроблення інформаційної системи автоматизації роботи користувачів	1.05	15.05	вик
5	Оформлення матеріалів до захисту БКР	15.05	30.05	вик

Студент Олег ВОЛОЩУК

Керівник роботи Олексій КОЗАЧКО

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 80 сторінок формату А4, на яких є 29 рисунків, список використаних джерел містить 20 найменувань.

Мета роботи полягає у створенні інформаційної технології, яка дозволить спростити процес пошуку, перегляду та додавання інформації про тварин, тим самим забезпечуючи зручний і швидкий обмін даними між користувачами, зацікавленими в адопції.

В роботі наведено розроблення мобільного додатку на платформі Android, за допомогою мови програмування Kotlin. Розроблено дизайн та функціонал мобільного додатку. Розроблена система забезпечує можливість керування режимом обмеженого доступу на мобільному пристрої.

Ключові слова: інформаційна система, адопція, мобільний додаток, Kotlin, Android Studio

ABSTRACT

The bachelor's thesis consists of 80 A4 pages with 29 figures and a list of references containing 20 titles.

The aim of the work is to create an information technology that will simplify the process of searching, viewing and adding information about animals, thereby providing convenient and fast data exchange between users interested in adoption.

The paper presents the development of a mobile application on the Android platform using the Kotlin programming language. The design and functionality of the mobile application have been developed. The developed system provides the ability to control the restricted access mode on a mobile device.

Keywords: information system, application, mobile application, Kotlin, Android Studio Translated with DeepL.com (free version)

ЗМІСТ

ВСТУП ... 3
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ... 5
1.1 Аналіз предметної області ... 5
1.2 Огляд аналогів ... 8
1.3 Потенційні користувачі інформаційної системи та їх потреби ... 14
1.4 Висновки ... 16
2 ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ... 18
2.1 Аналіз можливостей мови програмування Kotlin ... 18
2.2 Аналіз середовища розробки Android Studio ... 21
2.3 Опис IT-Інфраструктури ... 23
2.4 Вибір допоміжних технологій для реалізації проєкту ... 27
2.5 Висновки ... 29
3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ОБМІНУ ДОМАШНІМИ УЛЮБЛЕНЦЯМИ ... 31
3.1 Розроблення архітектури інформаційної системи ... 31
3.2 Візуалізація інтерфейсу користувача ... 33
3.3 Практична реалізація мобільного додатку ... 42
3.4 Тестування роботи системи ... 45
3.5 Висновки ... 52
ВИСНОВКИ ... 53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ... 54
Додаток А (обов’язковий) Технічне завдання ... 56
Додаток Б (обов’язковий) Протокол перевірки кваліфікаційної роботи ... 59
Додаток В (доповідниковий) Фрагмент лістингу програми ... 60
Додаток Г (обов’язковий) Ілюстративна частина ... 73

ВСТУП

Актуальність теми. У сучасному світі цифрові технології щодня збільшують свій вплив на світ, виступаючи засобом забезпечення швидкого доступу до інформації, комунікації та онлайн-послуг. Одним з питань, що сьогодні безпосередньо стоїть перед світом, є питання безпритульних тварин і ускладнення їх адопції. Особливо, коли кожен день людські звірства переростають у війни від яких страждають не тільки люди а й тварини. Пропозиції з цього питання часто розрізнені або просто малозручні для користувачів, що ускладнює знайти для тварини нового власника або самостійно вибрати собі вихованця. З іншого боку, зростає потреба у цифрових методах швидко та ефективно поєднати потенційних власників і тварин, які потребують прихистку. У цій ситуації, створення мобільного додатку з пошуку й обміну домашніх улюбленців видається надзвичайно актуальним, оскільки це зробить процес адопції цікавим, простим і доступним для багатьох користувачів.

Мета і задачі дослідження. Мета дослідження полягає у створенні інформаційної технології, яка дозволить спростити процес пошуку, перегляду та додавання інформації про тварин, тим самим забезпечуючи зручний і швидкий обмін даними між користувачами, зацікавленими в адопції. Для досягнення поставленої мети необхідно вирішити наступні задачі:

– проаналізувати наявні рішення в сфері адопції тварин і виявити їхні сильні та слабкі сторони;

– визначити найоптимальніші технології і платформу, за допомогою яких доцільно буде втілити в життя функціонал мобільного додатку;

– спроектувати архітектуру додатку, що охоплює реалізацію пошуку, обраного списку та з можливістю додавати своїх тваринок;

– розробити функціональний прототип додатку; провести тестування з урахуванням легкості у використанні, надійності та ефективності його функціонування.

Об'єктом дослідження є процес розробки мобільного додатка для взаємодії з користувачами в контексті пошуку та обміну домашніми тваринами.

Предметом дослідження є методи, інструменти й технології створення мобільного додатка в функціональному стані інформаційної платформи для адопції тварин.

Публікації. За результатами виконання даної роботи опубліковано тези доповідей на тему: «Інформаційна платформа для пошуку та обміну домашніми улюбленцями» на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2024-2025 рр.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Предметна область даної роботи охоплює сферу пошуку, передачі та адопції домашніх тварин за допомогою інформаційних технологій. Це надзвичайно важлива соціальна та етично відповідальна сфера, оскільки мільйони тварин щороку опиняються без домівок. Більшість із них - покинуті або загублені собаки та коти, які потрапляють у притулки, де умови утримання не завжди відповідають належному рівню, або залишаються на вулиці, де їхнє життя піддається випробуванням. Проблема безпритульних тваринок в Україні завжди була болючою. Песики з голодними очима біля супермаркетів та під'їздів, котики, що відігріваються у підвалах і харчуються на смітниках, ця картина стала для нас вже майже звичною. Завжди знаходилися добрі люди, які дбали про чотирилапих безхатченків. Все ж з початком війни становище безхатніх тварин погіршилося. Військові дії не лише спричинили гуманітарну кризу для людської спільноти, але й залишили тисячі тварин без даху над головою. Розгублені та голодні собаки й коти, були залишені людьми під час евакуації або втратили домівку через руйнування, опиняються на вулицях міст і сіл. В наш час ця проблема потребує негайного вирішення, адже безпритульні тварини не тільки страждають самі, але й становлять небезпеку для людей, будучи носіями різних захворювань [2].

У цьому контексті використання інформаційних технологій стає одним із ефективних інструментів боротьби з проблемою безпритульних тварин. Цифрові платформи, що об'єднують власників, волонтерів, захисні організації та потенційних опікунів, дозволяють значно пришвидшити процес пошуку нових домівок для тварин. Мобільні застосунки, зокрема, надають зручний спосіб перегляду анкет тварин, фільтрації за параметрами, встановлення контакту між користувачами та поширення актуальної інформації у реальному часі. Таким

чином, інтеграція сучасних технологій у цю сферу дозволяє не лише оптимізувати процес адопції, але й формувати у суспільстві культуру відповідального ставлення до тварин.

На рисунку 1.1 наведена статистика, яка вказує на те що зафіксовано більш ніж 140 тисяч безпритульних тварин в Україні.

З них:

– 84% тваринок викинуті на вулицю попередніми власниками.
– 11% були загублені, були змушені опинитись на вулиці випадково.
– 5% тварин народжені на вулиці.

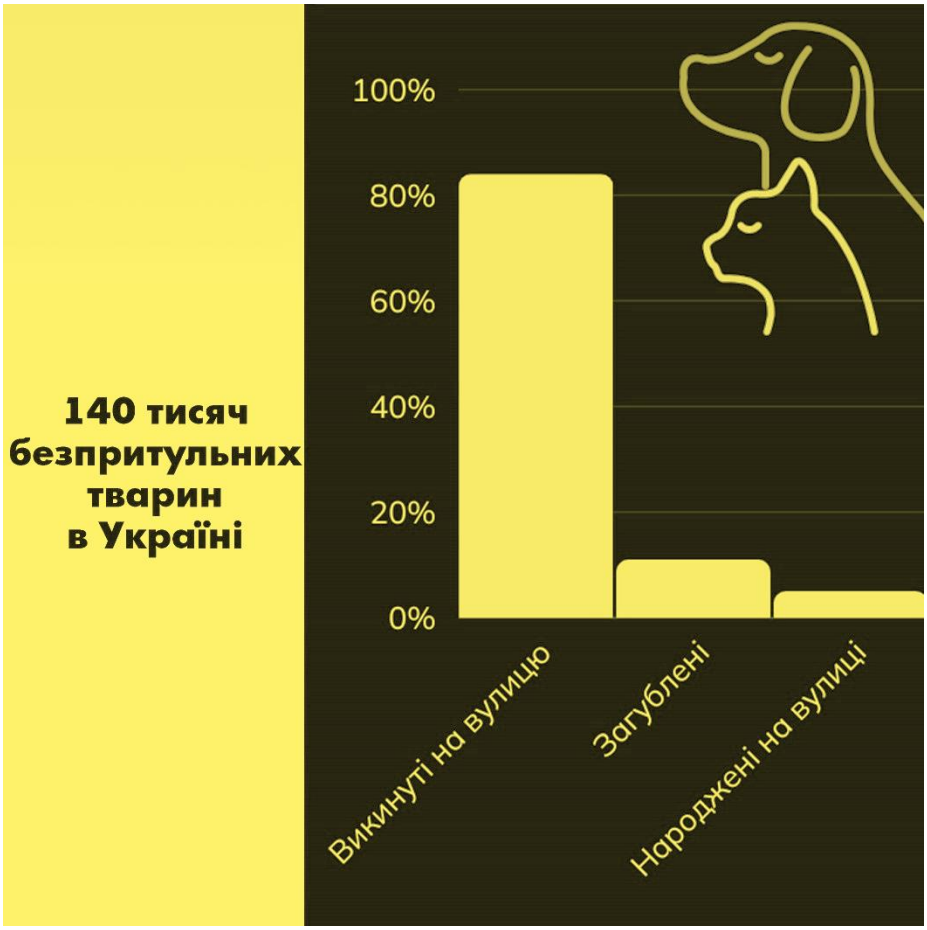

Рисунок 1.1 – Статистика Безпритульних тварин в Україні

Понад половина українців мають домашніх улюбленців, причому більшість із них мешкає в сільській місцевості. Водночас у містах налічується близько 4,4 мільйона власників тварин віком від 16 до 65 років. Смаки щодо тварин відрізняються: чоловіки частіше обирають собак, птахів зазвичай

заводять люди з нижчим рівнем доходу, а утримання акваріумів є досить коштовним хобі [3].

На рисунку 1.2 показано, які саме домашні тваринки є найпопулярнішими серед українців. До 50% власників тварин вони найшли свого улюбленця саме онлайн. Попри активний розвиток цифрових технологій та стрімке зростання

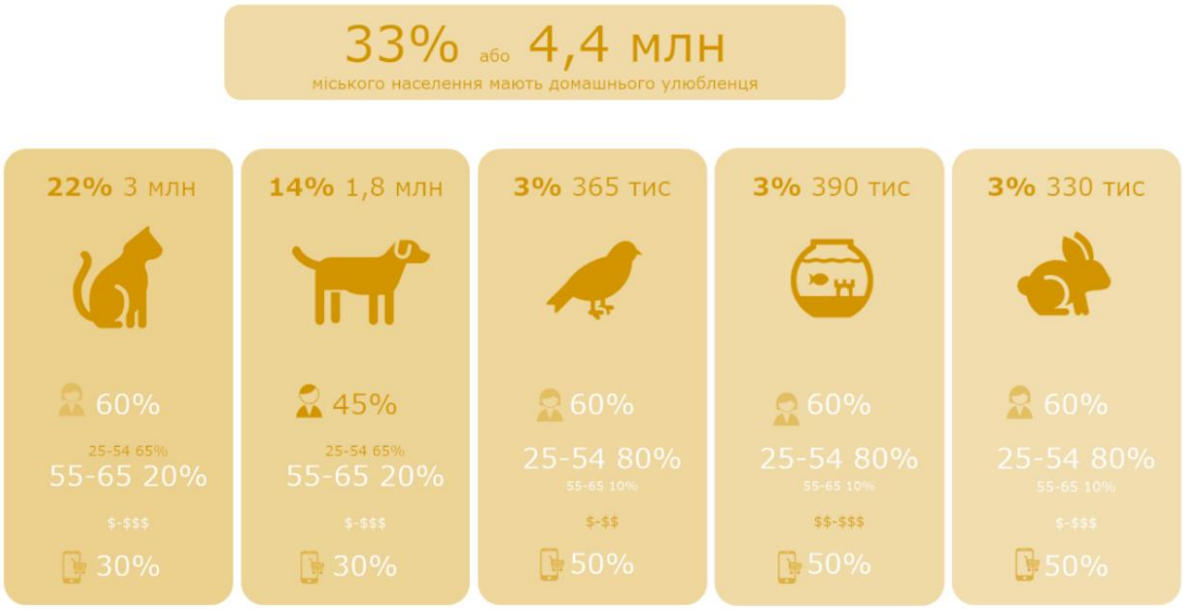

Рисунок 1.2 – Статистика популярності домашніх тваринок серед українців

популярності мобільних додатків, сфера адопції тварин в Україні досі залишається недостатньо діджиталізованою. На ринку практично відсутні Android-застосунки, що б дозволяли швидко й зручно шукати, переглядати і передавати домашніх тварин іншим власникам.

Це є особливо актуальним з огляду на те, що українські користувачі все частіше обирають онлайн-формати для таких процесів: вони шукають тварин через соціальні мережі, різні сайти, маркетплейси або тематичні спільноти. Все ж такі методи є неструктурованими, не завжди безпечними та не забезпечують зручного функціоналу для фільтрації та зв’язку.

Ось, саме та низка труднощів, з якою стикаються люди, шукаючи, нового домашнього улюбленця, не на перевірених сайтах чи спільнотах:

- Брак єдиного ресурсу: Відсутня централізована система, яка акумулювала інформацію про тварин, доступних для адопції в межах певного регіону.

- Обмежена фільтрація: Більшість доступних платформ не дають інструментів для зручного пошуку по параметрам, наприклад вид, порода, вік, розмір, стать, місцезнаходження і тд.
- Неактуальність даних: Часто інформація на платформах дуже давня, і оголошення залишаються активними навіть після того, як тварина вже знайшла нового власника.
- Геолокаційні обмеження: Пошук тварин у межах зручного місцезнаходження часто ускладнений через відсутність відповідних фільтрів або геолокаційної підтримки

1.2 Огляд аналогів

На ринку вже існують певні веб-платформи та мобільні застосунки, які дозволяють розміщувати оголошення про тварин, що потребують нового дому. Однак багато з них мають обмежений функціонал, неадаптований або незручний інтерфейс для мобільних пристроїв, або ж охоплюють лише певні регіони чи категорії тварин.

Коли ви вирішуєте завести домашнього улюбленця, перед вами стоїть два шляхи: придбати тварину у продавця або взяти її з притулку чи рятувальної організації. Хоча покупка може здаватися легшим варіантом, варто усвідомлювати цінність адопції. Вона не лише рятує життя тварини, але й суттєво змінює долю обох - і улюбленця, і його нового власника [4].

Мобільні застосунки, спеціалізовані саме на адопції тварин в межах конкретного міста або регіону, є не надто поширеними, а ті, що існують, часто мають недосконалу систему пошуку, відсутність інтерактивних можливостей та обмежену базу користувачів.

Для кращого розуміння ситуації на ринку, в рамках проєкту було проведено аналіз декількох існуючих сервісів, що мають подібний функціонал. Це дозволило виявити недоліки наявних рішень і визначити потенційні переваги розробленого мобільного застосунку, спрямованого на забезпечення зручного, швидкого та ефективного процесу пошуку та обміну домашніми улюбленцями.

Першим оглянемо додаток «Animal ID», зображений на рисунку 1.3. Це безкоштовний застосунок для iOS та Android, створений в Україні для ідентифікації домашніх та безпритульних тваринок, для об’єднання ветеринарів, кінологів, притулків та господарів. Ціль – підрахувати безпритульних тварин, щоб зменшити їх кількість та збільшити усиновлення таких собак [5].

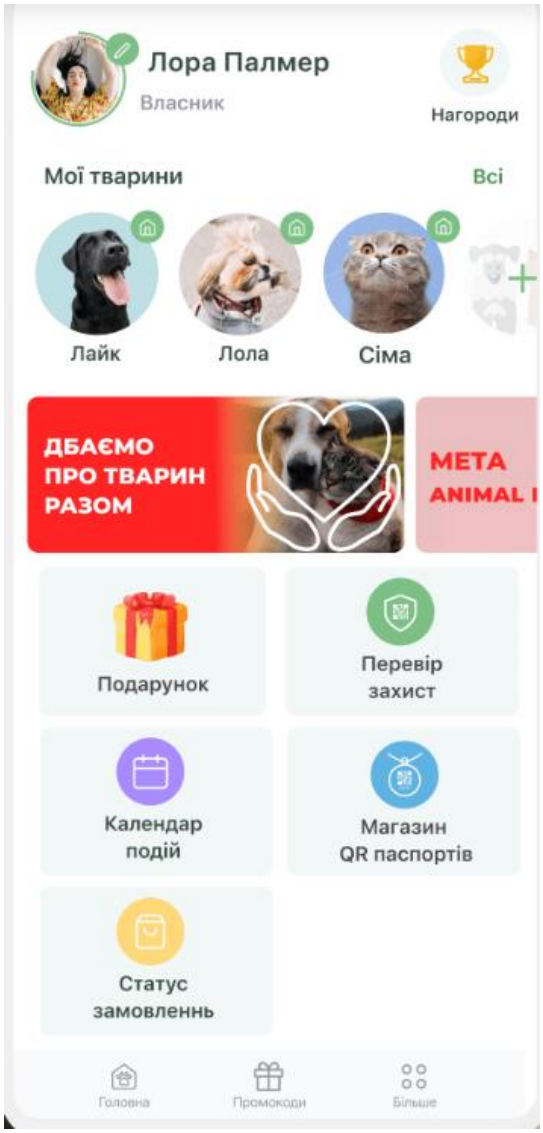

Рисунок 1.3 – Інтерфейс додатку «Animal ID»

Наступний додаток – «PetFinder», зображений на рисунку 1.4. Це платформа для пошуку та усиновлення домашніх тварин, яка дозволяє користувачам легко знайти собаку, кота чи інших пухнастих або шершавих друзів. Користувачі можуть шукати тварин для усиновлення з тисяч притулків та рятувальних організацій, фільтруючи за місцезнаходженням, породою, віком, розміром та статтю.

«PetFinder» об'єднує сотні тисяч організацій, які мають справу усиновлення та прийом тварин по всіх країнах. У базі присутні групи порятунку собак і котів, спеціалізуються на певних породах, таких як Німецькі вівчарки, Французькі бульдоги та Біглі або Пуделі [6].

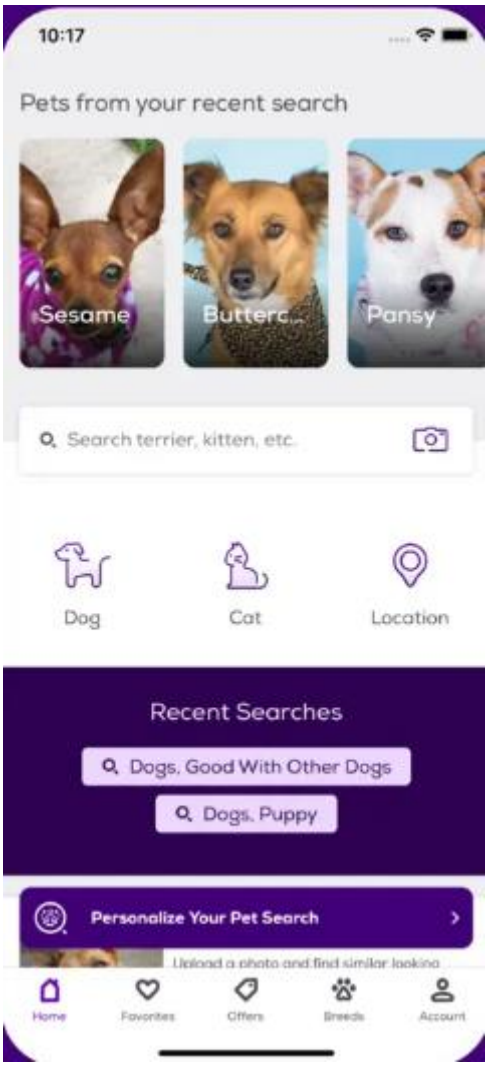

Рисунок 1.4 – Інтерфейс додатку «PetFinder»

Додаток «Домівка Врятованих Тварин», який зображений на рисунку 1.5, позиціонує себе як платформа, у якій ви можете частково, або ж повністю стати опікуном будь-якого мешканця домівки, обравши серед диких, екзотичних чи свійських! Таким чином забезпечити харчування, догляд на лікування обраної тварини.

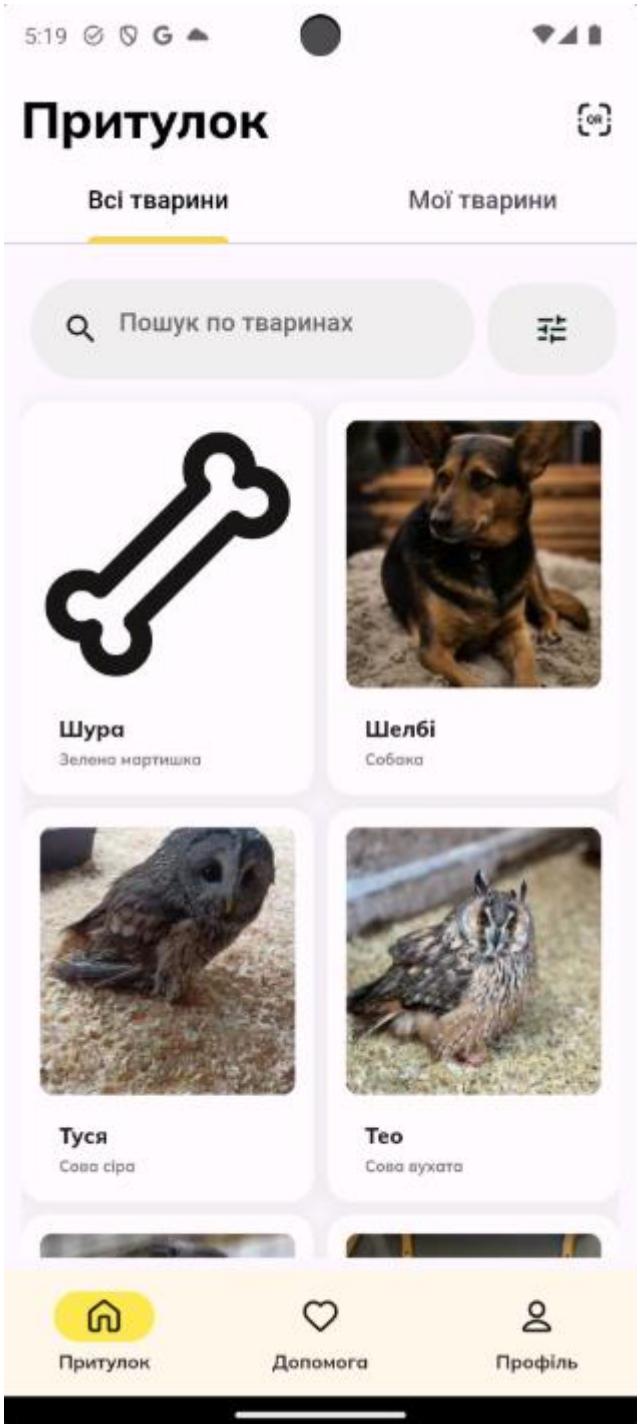

Рисунок 1.5 – Інтерфейс додатку «Домівка Врятованих Тварин»

Ще один додаток наведений на рисунку 1.6 - Pets Adoption - Adopt a Pet, він призначений для пошуку та адопції домашніх тварин, таких як собаки, коти, птахи, кролики та коні. Він надає користувачам можливість знаходити тварин, які потребують нового дому, та зв'язуватися з їхніми власниками або притулками.

Ключові особливості додатку «Pets Adoption - Adopt a Pet»

- Дозволяє переглядати домашніх тварин різних порід, доступних для усиновлення.
- Надає детальну інформацію про кожного вихованця.
- Надає контактну інформацію про власників тварин.
- Надає список порід домашніх тварин.
- Дозволяє вибрати породу, щоб знайти домашніх тварин цієї породи для усиновлення.

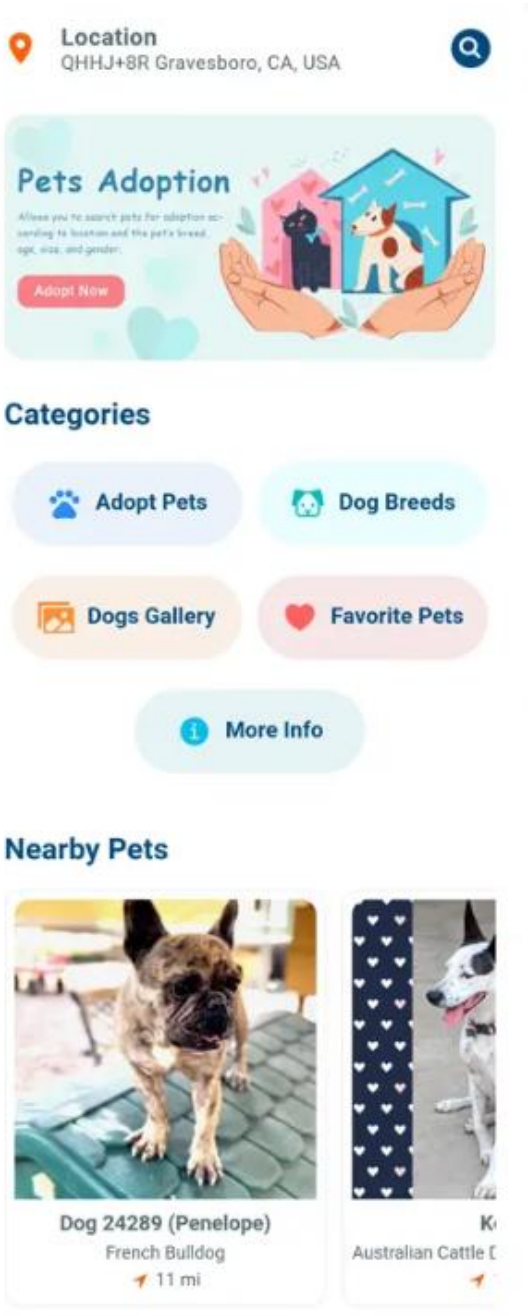

Рисунок 1.5 – Інтерфейс додатку «Pets Adoption - Adopt a Pet»

Аналізуючи ці додатки, отримав цінні ідеї для ідей, дизайну та досвіду, які можна втілити в розробку інформаційно-пошукової системи для пошуку та обміну домашніми улюбленцями. Зокрема, були враховані зручність користувацького інтерфейсу, інтуїтивна навігація, ефективні фільтри для пошуку тварин за різними критеріями чи категоріями, а також методи взаємодії між юзерами.

Окрім візуальних та функціональних аспектів, окрему увагу приділено загальному користувацькому досвіду: швидкодії додатків, адаптивності до різних типів пристроїв, рівню персоналізації контенту, а також доступності функцій для різних вікових і соціальних груп. Було проаналізовано, які підходи сприяють залученню користувачів, підвищують їхню активність та довіру до платформи. Вивчення підходів до подання інформації, таких як використання зрозумілих іконок, логічних сценаріїв взаємодії та мультимедійного контенту (фото, відео, відгуки), дало змогу ширше подивитись на можливості підвищення ефективності комунікації між користувачами та платформою.

Ці спостереження дозволили мені сформувати бачення майбутнього додатку як ефективного інструменту для адопції тварин, що поєднує функціональність, доступність та емоційний зв’язок із користувачем. Отримані висновки - це основа для проектування власного продукту, що враховує найкращі практики та уникає поширених недоліків існуючих рішень.

1.3 Потенційні користувачі інформаційної системи та їх потреби

Інформаційна система FluffyFind орієнтована на користувачів, які мають якийсь зв’язок із адопцією домашніх тварин. Такими користувачами можуть бути люди, які шукають тварин для усиновлення, власники, які змушені передати своїх тварин іншим особам, представники притулків і волонтери та зоозахисники, які піклуються про добробут безпритульних тварин. Кожна з цих груп має певні вимоги, які повинні бути враховані під час проектування та впровадження системи, щоб зробити платформу зручною та ефективною для використання.

Люди, які хочуть мати домашнього улюбленця, є однією з основних груп користувачів. Зазвичай люди очікують від них системи з простим інтерфейсом, який дозволяє шукати за різними критеріями, такими як вид тварини, порода, вік, стать, розмір, колір, особливі потреби або місце перебування. Для них надзвичайно важливо мати можливість переглядати високоякісні фотографії тварин, щоб дізнатися про їхні характери, звички, історії, стан здоров’я та наявність вакцин. Крім того, важливим є функціонал, який дозволяє легко зв’язуватися з притулком або власником через вбудований месенджер, отримувати оперативні відповіді та зберігати вподобаних тварин у списку, обраному для подальшого перегляду.

На рисунку 1.6 показана діаграма, на якій видно кількість усиновлених тварин під час двох Днів адопції в Києві у 2024 році [7]. Бачимо розподіл між собаками, котами та загальною кількістю прилаштованих тварин.

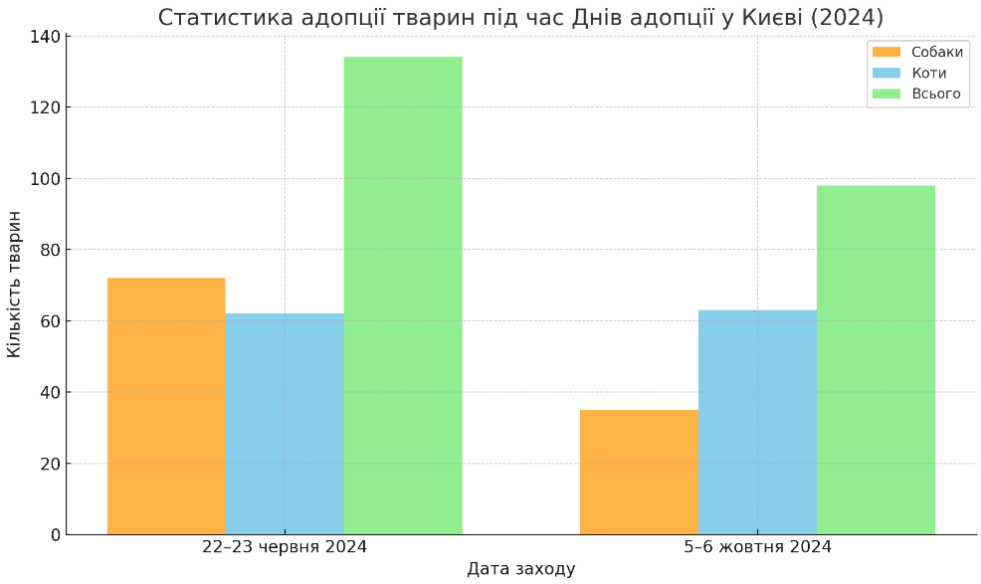

Рисунок 1.6 – Діаграма кількості усиновлених тварин в Києві

Користувачі, які шукають нове житло для свого улюбленця, також є важливою категорією. Причини можуть бути різними, наприклад зміни в житті, проблеми зі здоров'ям, переїзд або алергії у родичів. Таким особам потрібна платформа, яка дозволяє швидко та просто розмістити оголошення про передачу тварин. Система повинна дозволити завантажити фотографії, вказати всі важливі характеристики тварини, додати контактну інформацію та керувати своїми рекламами, редагуючи, оновлюючи або видаляючи їх у разі потреби. Крім того, користувачі хочуть, щоб платформа забезпечувала безпеку, наприклад, показуючи відгуки та рейтинги нових власників.

Притулки для тварин, які забезпечують тимчасове утримання безпритульних або врятованих тварин, також є важливими користувачами системи. Оскільки вони часто мають велику кількість тварин одночасно та потребують інструментів для централізованого управління інформацією, їхні потреби є більш складними. Система повинна підтримувати функцію масового завантаження оголошень, управління профілем організації, надання статистики щодо переглядів і кількості успішно прилаштованих тварин. Притулки також зацікавлені в наявності системи верифікації, яка допоможе людям, які шукають тварин для адопції, бути більш впевненими.

Волонтери та зоозахисники відіграють важливу роль у взаємодії з платформою. Вони активно підтримують порятунок тварин, організовують тимчасову перетримку, транспортування, лікування та шукають відповідальних власників. Їм потрібна система, яка дозволяє координувати свої дії, створювати термінові оголошення, оперативно реагувати на нові запити та організовувати взаємодію між притулками, приватними особами та іншими волонтерами. Швидкість, зручна навігація та можливості спілкування та сповіщень є важливими для таких користувачів.

Таким чином, створювати вимоги до архітектури, інтерфейсу, функціоналу та моделі взаємодії платформи FluffyFind можна, лише розуміючи потенційних користувачів і їхні потреби.

1.4 Висновки

У результаті аналізу предметної області було можливо повно охарактеризувати поточні проблеми пошуку, передачі та адопції домашніх тварин у цифровому середовищі. Існує проблема з якісними мобільними цифровими інструментами для адопції тварин, особливо на рівні Android-платформи, яка в Україні є найпоширенішою.

Вивчення поточних рішень показало, що більшість популярних платформ або мають глобальний характер і не адаптовані до потреб українців, або мають обмежений функціонал, який не охоплює всі аспекти взаємодії між різними сторонами процесу адопції. Це створює чітку нішу для впровадження спеціалізованої інформаційної системи, орієнтованої на локальні реалії, зосередженої на гуманному ставленні до тварин, зручною мовною підтримкою та простим інтерфейсом.

Додатково було виявлено, що потенційні користувачі часто стикаються з недовірою до платформ, які не забезпечують прозорості та верифікації користувачів або самих оголошень. Бракує ефективних механізмів зворотного зв’язку, рейтингів, підтверджених історій адопції - усіх тих функцій, які могли б підвищити довіру та відповідальність усіх учасників процесу. Також існує потреба в інтеграції освітніх та інформативних матеріалів, що підвищують

обізнаність користувачів про правила утримання тварин, правові аспекти адопції та важливість стерилізації й вакцинації. Всі ці компоненти значною мірою недооцінені в існуючих рішеннях, але є критично важливими для формування свідомої та відповідальної спільноти.

Цільова аудиторія була чітко визначена: це як звичайні люди, які шукають домашніх улюбленців, так і ті, хто змушений їх віддати. Крім того, враховано вимоги волонтерів, зоозахисників і притулків, які відіграють важливу роль у цьому соціально важливому процесі. Усі ці категорії мають різні запити до системи, що свідчить про те, що створення універсального, але адаптивного продукту є правильним.

Окремо варто відзначити недостатній рівень цифрової інклюзивності існуючих платформ. Часто вони не враховують потреби людей з обмеженими можливостями - наприклад, відсутність озвучування елементів інтерфейсу або адаптації для слабозорих користувачів. У рамках розробки системи доцільно закласти принципи доступності з самого початку. Також спостерігається нестача локалізованого контенту: навіть у багатомовних платформах українська мова або відсутня, або реалізована формально, що ускладнює взаємодію для частини користувачів. Водночас, інтерфейс має не просто перекладати елементи, а й адаптувати функції до культурного контексту, зокрема через використання знайомих образів, назв і структур поведінки. Такий підхід не лише спрощує користування, а й підсилює довіру до цифрового продукту як до свого, рідного ресурсу.

Таким чином, на основі аналізу предметної області очевидно, що розробка інформаційної системи FluffyFind, має полегшувати спілкування між учасниками процесу адопції, полегшувати пошук тварин за певними критеріями, забезпечувати зручне розміщення оголошень і створювати безпечну зону взаємодії користувачів. У результаті цього рішення тварини будуть прилаштовані краще. Крім того, це вирішення матиме важливий вплив на розвиток культури, в якій люди будуть більш відповідальними за своїх менших братів.

2 ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

2.1 Аналіз можливостей мови програмування Kotlin

Kotlin – це сучасна, багатоцільова мова програмування, вона зародилася в JetBrains, і звичайно що набула широкої популярності серед розробників. Мова призначена для створення додатків на платформі Java, але і також підтримує багато інших цілей, включно з Android-розробкою, веброзробкою та багато іншого [8].

Через численні переваги мови програмування Kotlin була обрана як основна технологія для реалізації мобільних додатків. Це особливо зручно та ефективно для створення Android-додатків. Компанія JetBrains розробила мову програмування Kotlin, яка є сучасною, статично типізованою мовою програмування. Google офіційно оголосила Kotlin однією з рекомендованих мов для розробки під Android у 2017 році, що лише підтверджує його актуальність і перспективність.

Основною перевагою Kotlin є її повна сумісність із Java, що дозволяє використовувати широкий спектр бібліотек, фреймворків і інструментів, які є наявні в Java. Це особливо важливо для проектів, у яких інтеграція з уже створеними модулями або використання сторонніх бібліотек Java. Kotlin легко компілюється в байт-код Java Virtual Machine (JVM) і може взаємодіяти з Java-кодом у будь-якому напрямку. Kotlin і Java-файли можуть існувати разом у одному проєкті без обмежень.

Читабельність і лаконічність коду також є великою перевагою. Kotlin дозволяє розробникам писати менше коду, щоб досягти тих самих функцій, які Java вимагає великої кількості шаблонного або «бюрократичного» коду. Це не тільки зменшує ймовірність помилок, але й покращує підтримку проекту, оскільки код стає простішим для інших розробників для розуміння та зміни. Наприклад, Kotlin пропонує data-класи, які автоматично створюють всі необхідні методи, а не пропонує гетерів, сетерів і конструкторів.

Крім того, важливою особливістю Kotlin є система типів, яка запобігає NullPointerException, що є однією з найбільш поширених проблем у Java-

розробці. У Kotlin типи за замовчуванням не допускають нулів, що означає, що компілятор не дозволяє розробнику випадково використати змінну, яка може бути нулем, без попереднього оброблення цього випадку. Таким чином, надійність програмного продукту підвищується, оскільки значна частина помилок виявляється вже на етапі компіляції.

Kotlin також пропонує вбудовану підтримку корутин, механізму, який дозволяє ефективно реалізовувати асинхронне програмування та багатопоточність. Асинхронність є життєво важливою для мобільної розробки, оскільки багато процесів (наприклад, мережеві запити, зчитування з бази даних, обробка великих кількостей даних) повинні виконуватися в фоновому режимі, не блокуючи головного потоку. Корутини дозволяють розробникам писати асинхронний код так, ніби він синхронний; це полегшує розуміння та підтримку коду.

На рисунку 2.1 наведена статистика за 2024 рік якій мові програмування надають перевагу.

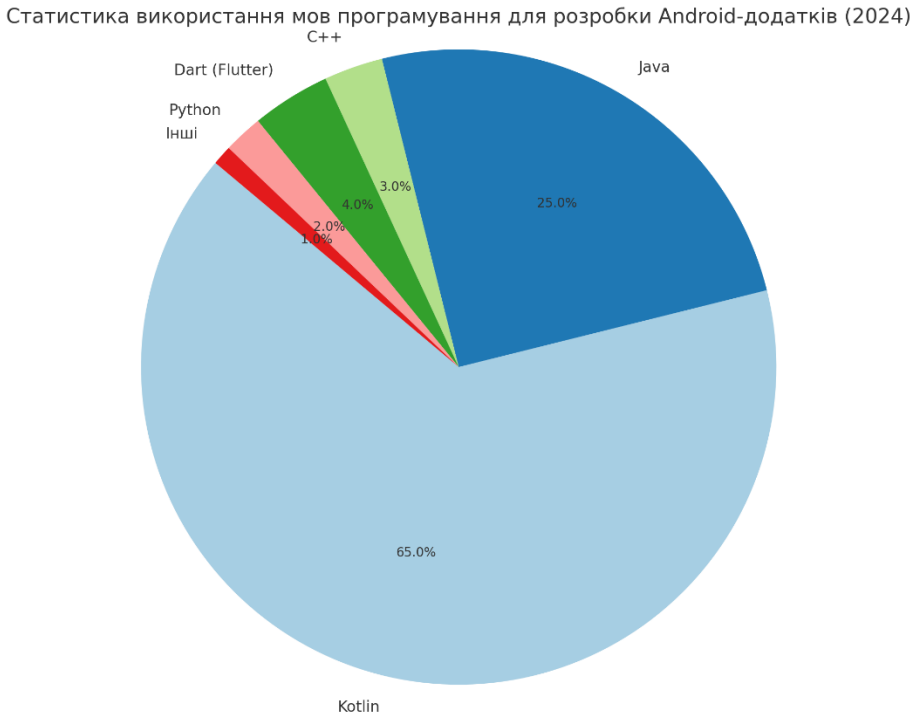

Рисунок 2.1 – Статистика використання мов програмування для розробки Android додатків

Крім того, варто відзначити високу швидкість розробки та наявність сучасного інструментарію. Kotlin повністю працює з Android Studio, офіційним середовищем розробки програмного забезпечення під керівництвом Android. Зручне автодоповнення, інтелектуальне підсвічування синтаксису, відладка та інші інструменти Android Studio покращують продуктивність розробників. Крім того, велика кількість відкритих ресурсів, плагінів і бібліотек Kotlin дозволяє швидко знайти готові рішення для стандартних завдань.

Функціональне програмування, яке підтримує Kotlin, дозволяє створювати більш чисту, модульну та тестовану архітектуру. Завдяки функціям вищого порядку, лямбда-виразам, розширенням для класів, операторам роботи зі колекціями та багатьом іншим функціям Kotlin робить виконання складних логік більш простим.

Крім технічних переваг Kotlin має активну спільноту та ефективну документацію, що полегшує навчання та вирішення проблем під час розробки. Багато прикладів, інструкцій і туторіалів доступні в Інтернеті, а також багато відкритих проєктів з відкритим кодом дозволяють вивчати найкращі практики програмування Kotlin. Підводячи підсумок, можна сказати, що використання мови Kotlin для створення мобільного застосування є цілком обґрунтованим рішенням, оскільки вона поєднує в собі наступне:

- Актуальність і постійний розвиток,
- Інтеграція повністю з Android-платформою,
- Високий рівень розробки,
- Безпечний догляд за типами,
- Добре працює асинхронне програмування,
- Зрозумілий і зручний синтаксис.

Kotlin ідеально підходить для розробки мобільного додатку в рамках конкретного проекту завдяки його перевагам.

Зважаючи на всі перелічені переваги мови, а також офіційну підтримку компанії Google та активний інтерес спільноти розробників, вважаємо, що Kotlin має всі шанси стати свого часу не менш популярною мовою у сфері створення мобільних додатків, ніж Java [9].

2.2 Аналіз середовища розробки Android Studio

Android Studio – являється офіційним інтегрованим середовищем розробки (IDE) Google для ОС Android, створене на основі програмного забезпечення JetBrains IntelliJ IDEA і розроблене для розробки Android [10].

Для створення додатку в якості основного середовища розробки було обрано Android Studio – інтегроване середовище розробки виробництва Google, використовуючи його, розробникам стають доступні інструменти для створення програм на платформі Android OS. Android Studio можна встановити на Windows, Mac та Linux [11].

Порівняємо Android Studio з іншими інтегрованими середовищами розробки (IDE) :

- Eclipse: До появи Android Studio основним середовищем для розробки Android-додатків було Eclipse. Проте нині Google офіційно рекомендує Android Studio, оскільки воно пропонує сучасніші інструменти, швидший процес збирання додатків і тіснішу інтеграцію з Android-платформою
- IntelliJ IDEA: Android Studio побудоване на базі IntelliJ IDEA - потужного середовища для розробки на Java від JetBrains. Хоча IntelliJ також підтримує Android через плагіни, Android Studio є спеціалізованим рішенням, яке містить додаткові функції, орієнтовані саме на Android-розробку, та глибшу інтеграцію з відповідними бібліотеками й інструментами.
- NetBeans: Java підтримується IDE NetBeans з відкритим кодом, яка доступна для завантаження. Крім того, для підтримки розробки додатків Android за допомогою плагінів він надає функціональні можливості для розробки Java. Але завдяки покращеній взаємодії з Android SDK та інструментами Android Studio пропонує більш спеціалізоване та багатофункціональне середовище для розробки Android.

Ось деякі ключові функції та компоненти Android Studio :

- Інтерфейс користувача: Android Studio пропонує зручний інтерфейс із різними панелями, такими як панель «Проект», панель «Редактор» і панель «Logcat», які дозволяють вам ефективно переміщатися та редагувати код.
- Система збірки Gradle: Android Studio використовує систему збірки Gradle, яка автоматизує процес створення, тестування та упаковки вашої програми.
- Він керує залежностями вашого проекту та обробляє процес збирання, полегшуючи керування складними проектами.
- Редактор коду: Android Studio містить потужний редактор коду з такими функціями, як доповнення коду, рефакторинг коду, підсвічування синтаксису та шаблони коду. Він підтримує кілька мов програмування, включаючи Java і Kotlin.
- Редактор макета: Редактор макета в Android Studio надає візуальний інтерфейс для розробки інтерфейсів користувача (UI) для вашої програми.
- Емулятор і диспетчер пристроїв: Android Studio містить емулятор, який дозволяє тестувати вашу програму на віртуальних пристроях з різними версіями Android і конфігураціями пристроїв.
- Налагоджувач: Android Studio постачається з потужним налагоджувачем, який допомагає знаходити та виправляти помилки у коді.
- Контроль версій: Android Studio легко інтегрується з такими популярними системами контролю версій, як Git, дозволяючи керувати кодовою базою, відстежувати зміни та співпрацювати з іншими розробниками.
- Розширюваність: Android Studio підтримує плагіни та розширення, що дозволяє налаштовувати та розширювати функціональні можливості IDE відповідно до ваших потреб.
- Враховуючи ці переваги, вибір Android Studio є найкращим та перевіреним рішенням для розробки мого мобільного застосунку.

2.3 Опис IT-Інфраструктури

Інформаційна система FluffyFind побудована з урахуванням сучасних вимог до мобільних застосунків: високої продуктивності, гнучкості, безпеки та масштабованості. IT-інфраструктура проєкту охоплює як архітектурні рішення, так і практичну реалізацію фронтенд та бекенд-компонентів, механізми обміну даними та підтримки користувачів.

Ключовою основою застосунку є архітектурний шаблон який зображений на рисунку 2.2 MVVM (Model–View–ViewModel), який забезпечує чітке розділення логіки інтерфейсу та обробки даних. Це значно полегшує супровід і масштабування проєкту. Такий підхід дозволяє розробникам змінювати або оновлювати інтерфейс без впливу на бізнес-логіку, а також спрощує тестування окремих компонентів.

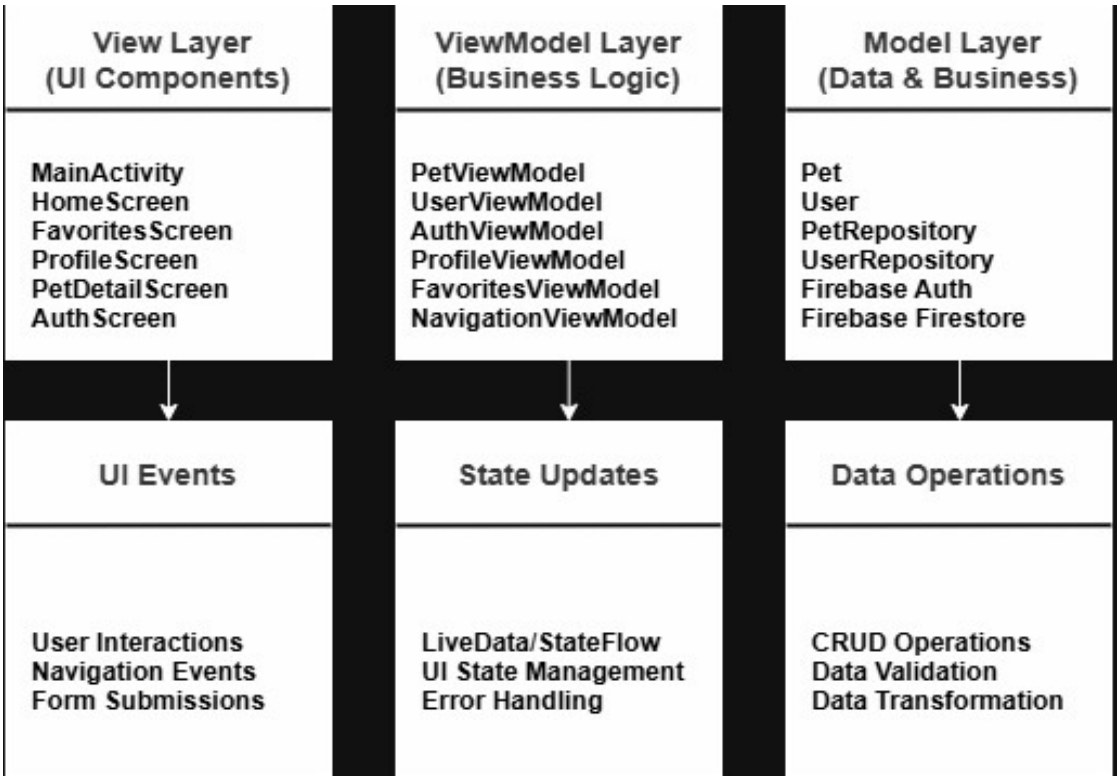

Рисунок 2.2 – Діаграма архітектури MVVM для FluffyFind

Jetpack Compose, який використовується як основна технологія побудови UI, забезпечує декларативний підхід до створення інтерфейсів. Це дає змогу швидко реалізовувати адаптивні, інтерактивні та візуально привабливі компоненти, які реагують на зміни стану в режимі реального часу. Застосунок

підтримує темну та світлу теми, що покращує користувацький досвід та відповідає стандартам Material Design 3.

ІТ-інфраструктура поділена на два ключові рівні: клієнтський (frontend) та даних (backend). На рівні клієнта реалізовані такі екрани: головний, профілю користувача, улюблених тварин, авторизації, перегляду детальної інформації про тварину. Навігація між екранами здійснюється за допомогою централізованого контролера та нижньої навігаційної панелі. Структура застосунку яка зображена на рисунку 2.3 організована модульно, що дозволяє додавати нові екрани чи функції без порушення існуючої логіки.

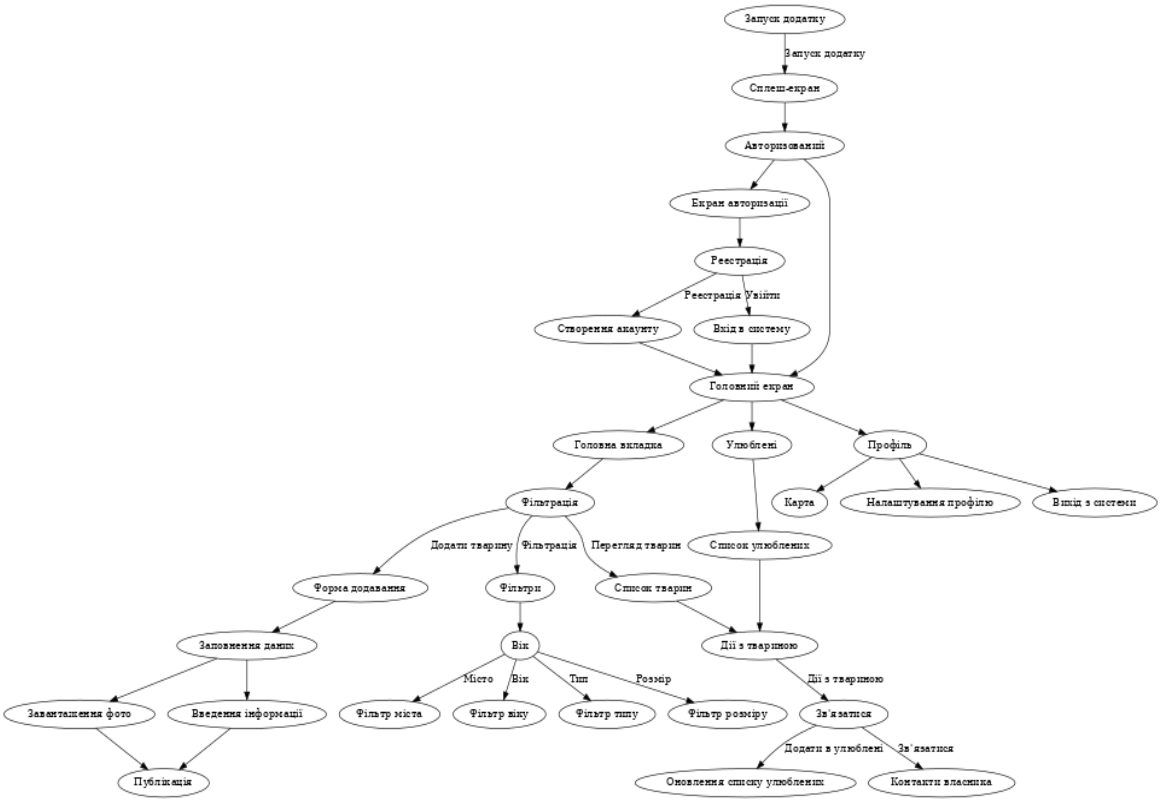

Рисунок 2.3 - Структура роботи інформаційної системи «FluffyFind»

З боку обробки та зберігання даних функціонують репозиторії, які взаємодіють із хмарними сервісами та локальними джерелами. Такий підхід відповідає принципам "Single Source of Truth" та "Separation of Concerns", що сприяє підтримуваності коду та забезпечує стабільність роботи застосунку.

Особливу увагу в інфраструктурі приділено забезпеченню безпеки. Усі операції з чутливими даними виконуються через захищені канали. Для автентифікації користувачів використано сервіси, що гарантують безпечний вхід

та зберігання інформації. Додатково, на рівні застосунку реалізовано валідацію вхідних даних, що запобігає помилкам і зловживанням.

Важливою складовою IT-інфраструктури є інтеграція з зовнішніми сервісами. Використання інтерактивних карт дозволяє відображати місцезнаходження тварин, а можливість зворотного зв'язку через email забезпечує пряме спілкування користувача з розробниками. Вся система орієнтована на максимальну зручність для кінцевого користувача: доступна українська локалізація, підтримуються кастомні налаштування, такі як сповіщення, звук тощо.

Для забезпечення продуктивності та оптимізації реалізовано кешування зображень за допомогою бібліотеки Coil, що дозволяє зменшити кількість запитів до серверу та пришвидшує завантаження контенту. Управління станом відбувається централізовано, що підвищує стабільність застосунку навіть при великій кількості активних користувачів. На рисунку 2.4 наведено статистику популярності та кількість використань бібліотек Coil, Glide, Fresco та Picasso на GitHub за останній місяць.

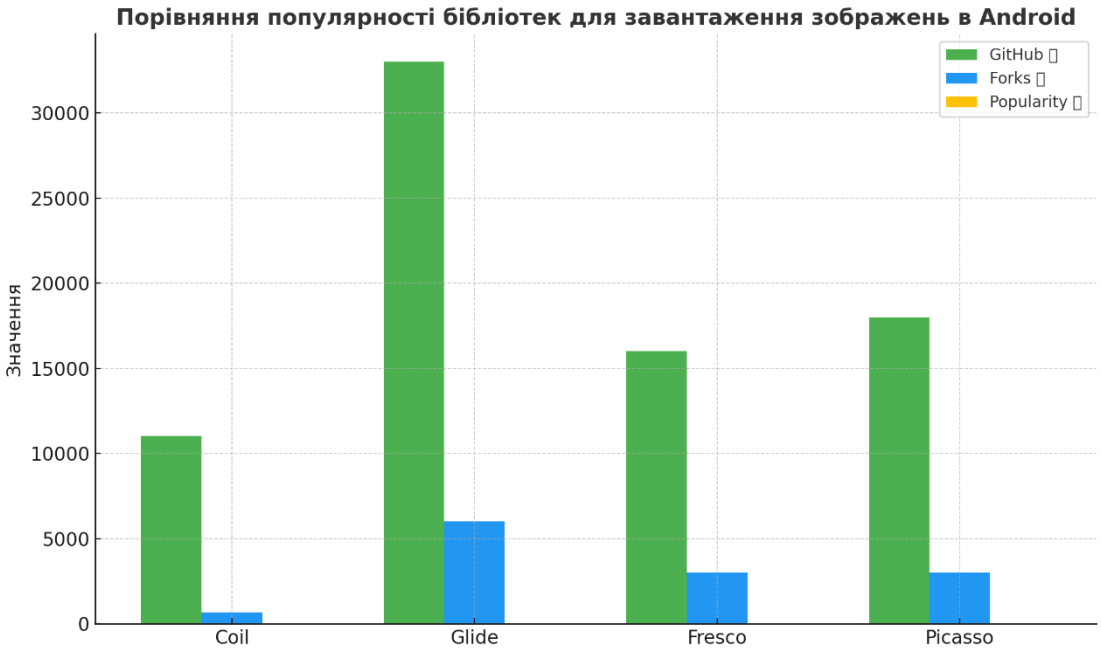

Рисунок 2.4 – Порівняння бібліотек Coil, Glide, Fresco та Picasso

Хоча бібліотека Coil поступається за кількістю зірок таким популярним рішенням як Glide чи Picasso, її вибір обґрунтований сучасним підходом до розробки, повною підтримкою Kotlin та Jetpack Compose, а також ефективністю

при роботі з ресурсами пристрою. Це робить її ідеальним вибором для сучасного Android-додатку, такого як FluffyFind.

З огляду на можливе зростання кількості користувачів, інфраструктура повинна бути здатною до масштабування. Обрані сервіси підтримують як вертикальне масштабування (збільшення ресурсів окремих компонентів), так і горизонтальне (додавання нових екземплярів), що дозволяє підтримувати стабільну роботу системи за високого навантаження. Наприклад, Cloud Firestore автоматично адаптується до збільшення кількості зчитувань і записів без потреби ручного втручання.

Ще однією перевагою IT-інфраструктури FluffyFind є її масштабованість. Завдяки модульній побудові та продуманій структурі даних, додаток легко адаптується до нових потреб. Наприклад, у майбутньому можна буде додати функції чату, фільтрації за геолокацією, розширення профілів тварин тощо, без значної перебудови існуючої системи.

Важливим аспектом є наявність системи резервного копіювання. У межах інфраструктури Firebase реалізовано механізми автоматичного створення резервних копій, що зберігаються у географічно розподілених дата-центрах. Це дозволяє швидко відновити роботу системи у разі збоїв або втрати даних. Додатково, створення плану аварійного відновлення та його тестування підвищує загальну надійність проєкту.

З боку технічної підтримки реалізовано механізми логування помилок та можливість оновлення застосунку. Це дозволяє розробникам оперативно реагувати на збої або звернення користувачів. Система зворотного зв'язку, інтегрована в інтерфейс, забезпечує безперебійний канал комунікації між користувачами та підтримкою.

2.4 Вибір допоміжних технологій для реалізації проєкту

Для реалізації мобільного застосунку FluffyFind - інформаційної платформи для пошуку та обміну домашніми улюбленцями - було обрано мову програмування Kotlin. Додатково використано низку допоміжних технологій та інструментів, таких як Firebase, Room, Material Design Components та інші. У цьому підпункті буде докладно розглянуто причини такого вибору та обґрунтовано його доцільність.

Вибір мови програмування Kotlin:

Kotlin є сучасною, статично типізованою мовою програмування, створеною компанією JetBrains. У 2017 році компанія Google офіційно оголосила Kotlin однією з основних мов для Android-розробки, а з 2019 року Kotlin визнано пріоритетною мовою для Android. Такий високий рівень підтримки та розвитку пояснюється низкою її переваг:

- Лаконічність синтаксису. Kotlin дозволяє писати менше коду порівняно з Java завдяки використанню розширень, функцій вищого порядку та скороченого синтаксису. Це не лише пришвидшує розробку, а й покращує читабельність та супровідність коду.
- Безпечність щодо null-значень. Kotlin пропонує вбудовані механізми боротьби з помилками, пов’язаними з null-значеннями (одна з найчастіших причин падіння програм у Java). Завдяки системі Nullable / Non-nullable типів, більшість таких помилок можна виявити ще на етапі компіляції.
- Повна сумісність із Java. Kotlin працює поверх JVM (Java Virtual Machine) і повністю сумісний із Java. Це дозволяє використовувати всі існуючі Java-бібліотеки та поступово мігрувати проєкти без потреби повного переписування коду.
- Підтримка Android Studio. Android Studio має повну інтеграцію з Kotlin - автоматичну генерацію коду, рефакторинг, налагодження, підтримку Gradle тощо.

- Широке ком’юніті та документація. Мова активно розвивається, має відкритий вихідний код та велику спільноту, що забезпечує наявність рішень для більшості поширених завдань.

- У контексті розробки застосунку FluffyFind Kotlin забезпечує високу продуктивність, безпечну обробку даних користувача, а також спрощує реалізацію сучасного, адаптивного інтерфейсу. Мова також добре підходить для роботи з асинхронними запитами, що важливо при роботі з хмарними сервісами (наприклад, Firebase).

Використання Firebase:

Firebase - це набір хмарних сервісів, які надають Google для створення, розгортання та обслуговування мобільних і веб-застосунків. У FluffyFind було використано кілька ключових компонентів Firebase, які забезпечують критично важливу функціональність.

Для автентифікації користувачів було використано Firebase Authentication. Цей сервіс дозволяє швидко реалізувати реєстрацію та вхід користувачів за допомогою електронної пошти, номеру телефону або соціальних мереж. Це знижує час на розробку та підвищує безпеку даних користувача.

Для зберігання даних про тварин, користувачів та їхні обрані оголошення використовується Cloud Firestore - масштабована, гнучка база даних NoSQL. Firestore дозволяє зберігати структуровані дані у вигляді колекцій і документів. Основні переваги:

- Миттєва синхронізація даних у реальному часі;
- Хмарне зберігання з можливістю офлайн-доступу;
- Потужна система запитів, що дозволяє фільтрувати тварин за параметрами (тип, порода, вік тощо).

Для зберігання фотографій домашніх улюбленців використовується Firebase Storage. Цей сервіс дозволяє зберігати великі файли (зображення, відео) з високою швидкістю доступу та вбудованими механізмами безпеки.

Firebase Realtime Database, дозволяє розробникам створювати реальний обмін даними між додатком та сервером, і звичайно що без необхідності в оновленні сторінки. Firebase , має і інші компоненти, такі як Firebase Cloud

Storage, він дозволяє зберігати файли в хмарі, Firebase Cloud Messaging - дозволяє розробникам відправляти сповіщення на мобільні пристрої, та Firebase Hosting, який в свою чергу дозволяє розміщувати веб-додатки в хмарі і забезпечує їх швидку та безпечну доставку [12].

Для реалізації сповіщень, наприклад, про нові повідомлення в чаті або нові оголошення, використовується Firebase Cloud Messaging. Це дозволяє оперативно інформувати користувачів без потреби постійного опитування сервера.

Локальне зберігання: Room

Для локального кешування даних, зокрема обраних тварин або історії переглядів, у проєкті використовується Room являється офіційною бібліотекою Google для роботи з SQLite. Room виступає як ORM-рішення (Object-Relational Mapping) і дозволяє працювати з базою даних через Kotlin-класи.

Room — одна з базових бібліотек, без якої не рекомендують починати роботу над android-проєктом. Вона, входить в Jetpack, це список популярних і використовуваних бібліотек. Завдання, допомогти писати менше коду, і звичайно що зробити його узгодженим з актуальними версіями ОС і методологічно правильним. Взагальному у ньому є все, що критично важливо для спрощення роботи і Room [13].

Переваги Room:

- Автоматична генерація SQL-запитів на основі анотацій;
- Перевірка запитів під час компіляції;
- Зручна інтеграція з LiveData та Coroutines;
- Підтримка міграцій бази даних при оновленнях застосунку.

2.5 Висновки

Проведений аналіз дозволив сформувати чітке розуміння щодо вибору основних інструментів та технологій, які використовуються для реалізації мобільного застосунку FluffyFind. Обраний технологічний стек - це поєднання мови програмування Kotlin, середовища розробки Android Studio та таких сучасних рішень, як Firebase і Room.

Kotlin забезпечує лаконічний, безпечний і продуктивний підхід до програмування, що є особливо важливим при створенні надійних мобільних застосунків. Його інтеграція з Android Studio та повна сумісність із Java надають гнучкість у реалізації як нових рішень, так і використанні перевірених практик. Завдяки статичній типізації, обробці null-значень та підтримці корутин Kotlin дає змогу зменшити кількість помилок і спростити роботу з асинхронними операціями.

Використання Android Studio як середовища розробки забезпечує розробнику потужний інструментарій для ефективного створення, тестування та налагодження додатку. Зокрема, її глибока інтеграція з Kotlin та Firebase значно прискорює процес розробки та зменшує ймовірність технічних помилок.

Додатково було доцільним використання таких технологій, як Firebase - для організації авторизації, зберігання даних та сповіщень - та бібліотеки Room - для реалізації локального зберігання даних. Усе це дозволяє створити застосунок, що відповідає сучасним стандартам якості, має високу продуктивність, зручний інтерфейс та забезпечує комфортне користування як у режимі онлайн, так і офлайн.

Таким чином, обрані мова програмування, інструменти та платформи повністю відповідають цілям і вимогам проєкту. Вони дозволяють реалізувати усі заплановані функціональні можливості застосунку FluffyFind та забезпечують умови для його подальшого масштабування й підтримки.

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ОБМІНУ ДОМАШНІМИ УЛЮБЛЕНЦЯМИ

3.1 Розроблення архітектури інформаційної системи

Інформаційно-пошукова система для пошуку та обміну домашніми улюбленцями, була розроблений шляхом складної архітектури, яка поєднує сучасні технології клієнт-серверної взаємодії, реактивне управління станом і високу модульність. Така архітектура гарантує безпеку обробки користувацьких даних, масштабованість, зручність використання та стабільність застосунку.

Загальна структура застосунку базується на архітектурі патерні MVVM, що дозволяє розділити логіку користувача (UI), бізнес-логіку та роботу з джерелами даних. Це полегшує процес тестування та гарантує, що окремі частини системи є незалежними. У проекті реалізовано три основні рівні: зображення, зображенняМодель і модель. Кожен з них містить відповідні репозиторії для роботи з Firebase Firestore та локальним сховищем SharedPreferences.

Функціональна архітектура застосунку складається з багатьох основних екранів. Ці екрани включають головний екран зі списком тварин (HomeScreen), екран улюблених (FavoritesScreen), екран профілю користувача (ProfileScreen), екран детального перегляду тварини (PetDetailScreen), екран авторизації (AuthScreen) і екран з інтегрованою картою (MapScreen). Для навігації між цими екранами Jetpack Navigation Compose має підтримку вкладеної навігації, нижньої панелі навігації та посилань. Архітектура, яка базується на підході Single Activity, покращує керування навігаційним стеком і знижує навантаження на систему.

Реалізований на основі хмарного сервісу Firebase Firestore, зберігання даних дозволяє зберігати оголошення, профілі користувачів і списки улюблених тварин. SharedPreferences є локальною альтернативою для зберігання налаштувань, таких як звуки, сповіщення та мова. Він дозволяє швидко отримати доступ до персоналізованих налаштувань без потреби у підключенні до мережі.

Для організації доступу до даних створено PetRepository і UserRepository. Ці репозиторії об’єднали логіку взаємодії з локальним сховищем і Firebase. З цієї причини ViewModel-и залишаються незмінними та відповідають лише за бізнес-логіку. StateFlow є інструментом для управління системним станом, який дозволяє відслідковувати зміни та оновлювати інтерфейс у режимі реального часу. Compose-функція запам’ятовування використовується для короткострокового локального стану.

Firebase Authentication забезпечує систему автентифікації, яка підтримує кілька способів входу, включаючи анонімну авторизацію, використання Google Sign-In та використання електронної пошти та пароля. Модуль UserRepository дозволяє керувати сесією користувача, вхід і вихід із системи.

Безпека була дуже важливою під час проєктування. Firebase Security Rules дозволяють контролювати доступ до даних Firestore. Ці правила обмежують операції з даними відповідно до автентифікаційного статусу користувача. Додатковими функціями є валідація вхідних даних, очищення вводу користувача та обробка помилок, яка забезпечує відповідні повідомлення користувача.

Крім того, система враховує локалізацію та доступність, тому є підтримка української мови, але можна розширити підтримку до інших мов. Coil, бібліотека для асинхронного завантаження зображень з кешуванням, покращує продуктивність. Це значно скорочує час очікування та ресурси пристрою.

Дизайн системи гарантує високий рівень тестованості. Здійснено тести взаємодії компонентів інтерфейсу, а також модульні тести для ViewModel і репозиторіїв. Увага приділялася тестуванню навігації та реакції системи на різні ситуації.

Таким чином, архітектура інформаційної системи FluffyFind поєднує сучасні підходи до розробки мобільних застосунків із масштабованою структурою, безпечним зберіганням даних і гнучким управлінням. Структура була розроблена з огляду на модульність, надійність і готовність до подальшого розширення функціональності.

3.2 Візуалізація інтерфейсу користувача

Розробка дизайну графічного інтерфейсу для мобільного застосунку якому я дав назву "FluffyFind" здійснювалася в онлайн-інструменті для дизайну Figma. Figma - це сайт, застосунок для роботи із векторною графікою, що дозволяє створювати різні макети, інтерфейси, логотипи, лендінги, проектування і багато інших креативів чи дизайнів. Інструмент цей популярний серед Web чи UI/UX-дизайнерів, SMM-менеджерів, stories-креєйторів та інших digital-експертів. Бо він зручний для використання, доступний, тут є функція працювати над проєктом віддалено разом з іншими учасниками та багато чого іншого[14].

Основною метою етапу проєктування UI/UX було створити інтуїтивно зрозумілий, привабливий та ефективний інтерфейс, орієнтованого на потреби основних користувачів, які в пошуках тваринок. UI-дизайн і UX-дизайн включають в себе різні навички, але вони є невід'ємною частиною успіху один одного. Гарний дизайн це про те завжди може врятувати інтерфейс, який незручний і заплутаний в навігації, а блискучий інтерфейс це якраз таки те що ідеально підходить для користувача, може бути зіпсований поганим візуальним інтерфейсом, який робить використання програми не дуже неприємним [15].

Я обрав фіолетово-рожеву кольорову гаму для мого додатку з огляду на психологічний та емоційний вплив цих кольорів на користувача. Фіолетовий колір традиційно асоціюється з чимось чарівним, спокійним і водночас інноваційним. Він пробуджує уяву, викликає почуття довіри та комфорту, що є надзвичайно важливим для застосунку, пов’язаного з тваринами. Адже користувачі звертаються до такого сервісу не тільки з практичною метою, але й емоційною - вони шукають нового улюбленця або хочуть знайти новий дім для тварини, яка їм небайдужа.

Рожевий, своєю чергою, символізує доброту, ніжність, турботу та любов - саме ті цінності, які є ключовими у процесі адопції. Поєднання фіолетового й рожевого утворює гармонійну палітру, що викликає позитивні емоції,

асоціюється з теплом, безпекою та доброзичливою атмосферою. Такий вибір кольорів створює враження турботливого, дружнього й сучасного сервісу. Крім того, ця палітра виділяється серед стандартних інтерфейсів, роблячи додаток впізнаваним і привабливим, особливо для молодої аудиторії.

Логотип відіграє центральну роль у формуванні візуальної ідентичності додатку. Він є першою точкою контакту користувача з брендом, тому має бути простим, легко запам’ятовуваним і водночас таким, що передає основне повідомлення сервісу. У моєму випадку логотип поєднує образ тварини (що є очевидним натяком на ціль додатку) із м’якими, округлими формами, які символізують дружність і безпеку. Використання цих форм підкреслює ненав’язливість та відкритість інтерфейсу, що спонукає користувача довіряти платформі.

Округлий дизайн інтерфейсу також не є випадковим. Закруглені кути, плавні лінії та м’які форми асоціюються з дружністю, доступністю та сучасністю. Такий підхід у дизайні значно покращує користувацький досвід, адже зменшує візуальний тиск і створює відчуття легкості під час взаємодії з додатком. Це особливо важливо у контексті мобільних застосунків, де користувачі очікують інтуїтивної навігації та комфортного інтерфейсу.

Початком розробки дизайну слугував розташований на рисунку 3.1 логотип інформаційно-пошукової системи – «FluffyFind», літери F розташовані дзеркально один до одної і символізують вуха, що допомогає і підтримує орієнтуватись користувачу що додаток створений саме для процесів адопції або процесів обміну тваринами.

Рисунок 3.1 – Логотип Інформаційно-пошукової системи «FluffyFind»

Дизайн охоплює такі основні сторінки:

1. Головна сторінка (HomeScreen), яка зображена на рисунку 3.2.

Основна функція: Відображення списку тварин та управління фільтрами

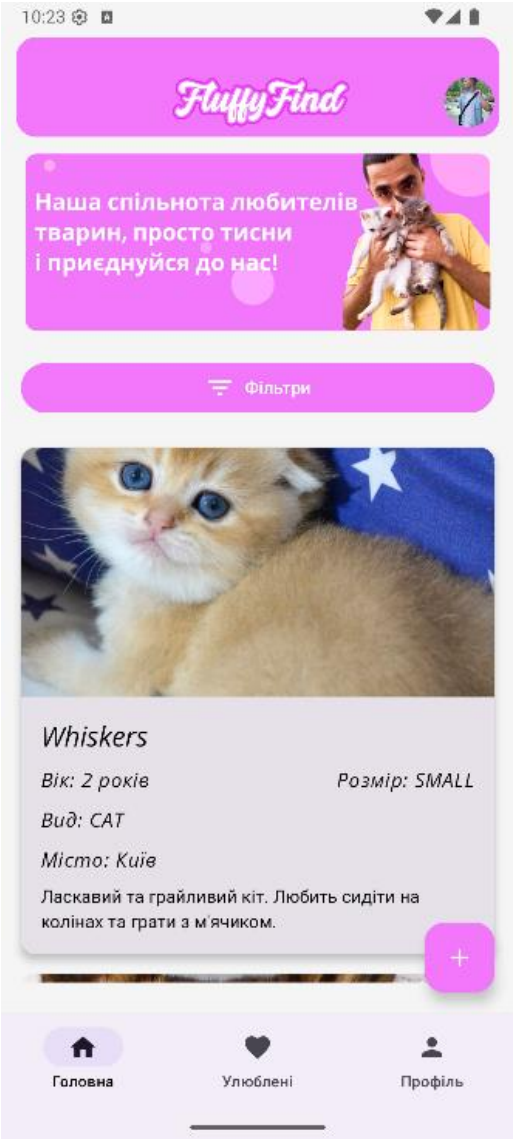

Рисунок 3.2 – Головна сторінка

Містить: Верхню панель з логотипом та профілем користувача, Кнопки соціальних мереж «Instagram» та «Monobank»), Систему фільтрів для пошуку тварин, Список карток тварин.

Функціонал: Фільтрація тварин за різними параметрами, Додавання тварин до улюблених, Перегляд детальної інформації про тварину, Перехід до профілю користувача, Перехід до соціальних мереж

2. Функціонал фільтрів

- Основна функція: Налаштування параметрів пошуку тварин
- Містить: Фільтр за видом тварини (собаки, коти, інші), Фільтр за розміром (маленький, середній, великий), Фільтр за містом, Фільтр за віком
- Функціонал: Зображений на рисунку 3.3, де можна побачити вибір одного або декількох фільтрів, Скидання фільтрів, Автоматичне оновлення списку тварин

Рисунок 3.3 – Функціонал фільтрів

3. Сторінка тварини

- Основна функція: Відображення інформації про конкретну тварину
- Містить : Фото тварини, Ім’я тварини, Вік, Розмір, Вид, Місто, Опис, Кнопку «Улюблене»
- Функціонал: Зображений на рисунку 3.4 Додавання/видалення з улюблених, Перехід до детальної інформації про тварину

Рисунок 3.4 – Сторінка тварини

4. Профіль користувача

- Основна функція: Відображення інформації про користувача
- Містить: Фото профілю, Інформацію про користувача, FAQ та налаштування, можливість вийти з профілю.
- Функціонал: Проілюстрований на рисунку 3.5, містить перегляд профілю, виключення звуку та сповіщень, кнопка FAQ, яка перенаправляє користувача на можливість надіслати лист на пошту olegvolosh04@gmail.com, кнопка виходу.

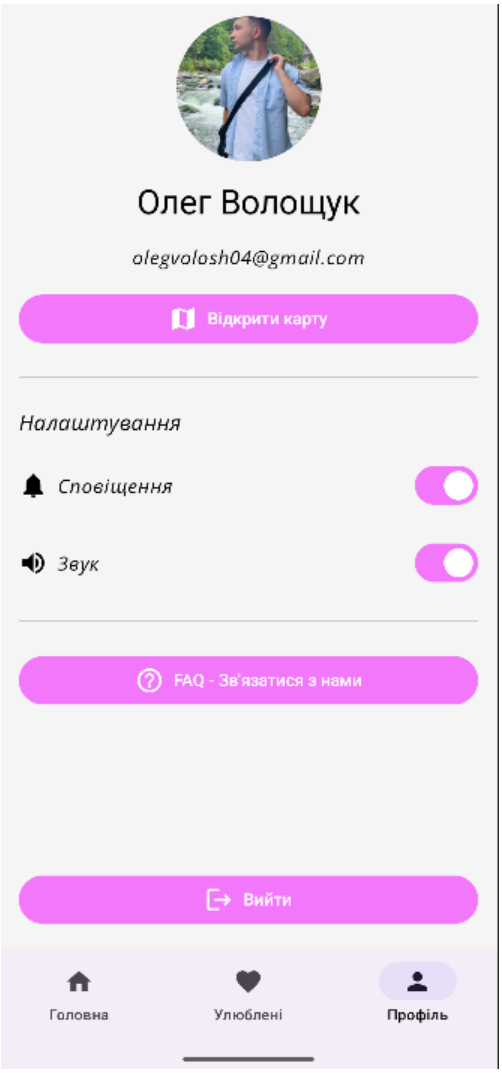

Рисунок 3.5 – Профіль користувача

5. Кнопки соціальних мереж

- Основна функція: Взаємодія з соціальними мережами
- Містить: Кнопку Instagram, Кнопку Monobank для донатів
- Функціонал: Перехід до сторінки Instagram, Перехід до сторінки донатів, Свайп між кнопками (Рисунок 3.6 та Рисунок 3.7).

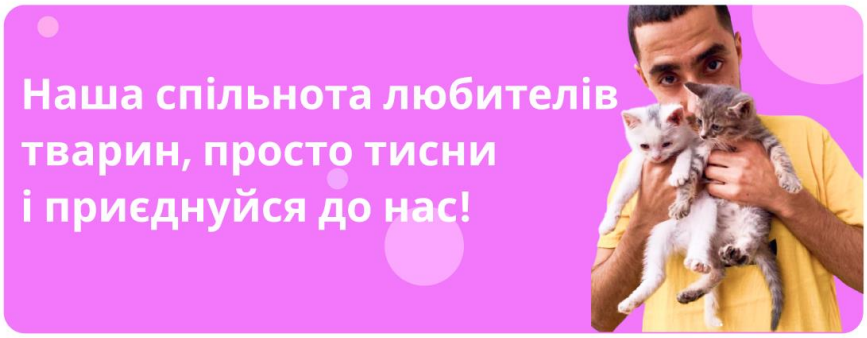

Рисунок 3.6 – Кнопка Інстаграм сторінки спільноти

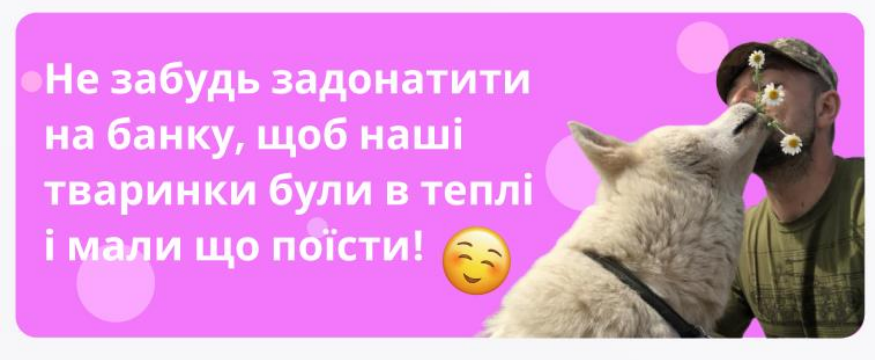

Рисунок 3.7 - Кнопка Монобанки для допомоги тваринам з прифронтових територій.

6. Сторінка улюблених

- Основна функція: Відображення тварин, доданих до улюблених
- Містить: Список улюблених тварин
- Функціонал: Зображений на рисунку 3.8, являє собою перегляд улюблених тварин, Видалення з улюблених.

Рисунок 3.8 – Сторінка улюблених

7. Сторінка Google Карт з Ветернинарними клініками

- Основна функція: Відображення інформації і розташування клінік у містах України

- Функціонал: Зображений на рисунку 3.9, де бачим можливість перегляду повної інформації, Розташування.

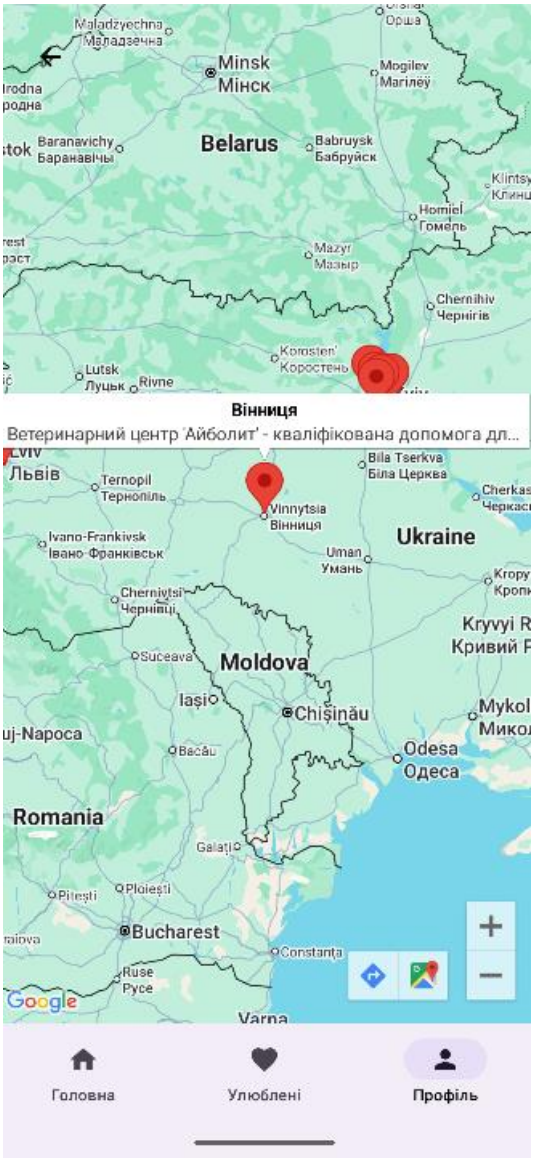

Рисунок 3.9 - Сторінка Google Карт з Ветернинарними клініками

8. Сторінка додавання тварини

- Основна функція: Можливість добавити тварину на головну додатку,

- Функціонал: Додавання за віком, типом, опис, імя, фото, розмір – ці всі можливості показані на рисунку 3.10.

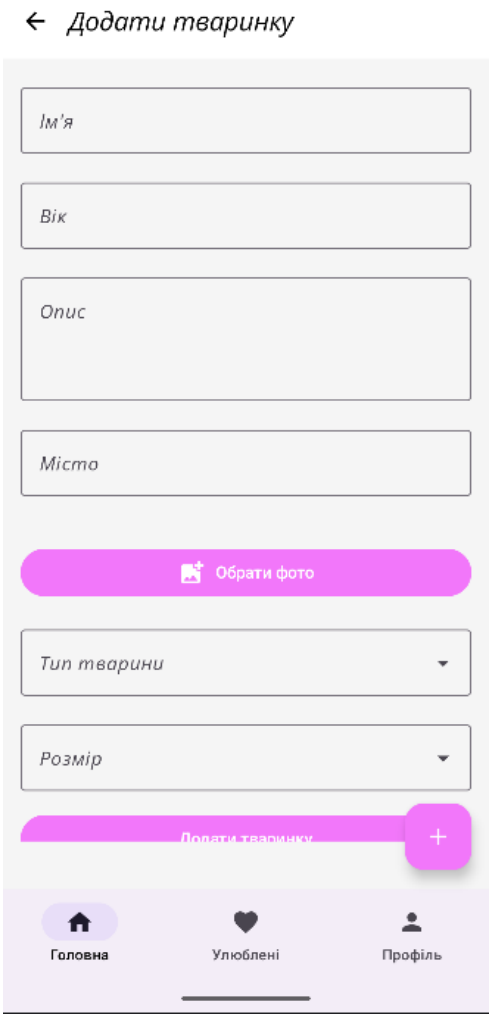

Рисунок 3.10 – Сторінка додавання тварини

Дизайн мобільного додатку сам по собі включає доволі великий і необмежений спектр завдань. Авжеж і крім візуального та технічного оформлення проводиться вивчення поведінкових факторів юзерів, аналітика конкурентів чи інших компаній чи творців, розробка додаткового функціонала і так далі [16].

Загальний дизайн застосунку виконаний у світлій колірній гамі з використанням фіолетових акцентів, що створює відчуття легкості та сучасності. Елементи інтерфейсу виглядають лаконічно та функціонально. Навігаційна панель, розміщена в нижній частині екрану на всіх основних сторінках, забезпечує швидкий та зручний доступ до ключових розділів застосунку, що сприяє позитивному користувацькому досвіду. Кожна сторінка розроблена з урахуванням основних сценаріїв використання та надає

користувачеві необхідну інформацію та інструменти для ефективного пошуку тваринок.

Кожна сторінка має свій унікальний функціонал та призначена для виконання конкретних завдань. Всі сторінки взаємопов'язані між собою та утворюють єдину систему для пошуку та взаємодії з тваринами.

3.3 Практична реалізація мобільного додатку

Реалізація інформаційної системи для пошуку та обміну домашніми улюбленцями передбачає поетапний процес розробки, який охоплює створення користувацького інтерфейсу, реалізацію функціональності, інтеграцію з базою даних та тестування. Враховуючи вибрані технології, зокрема Android Studio та Kotlin, розробка організована з урахуванням сучасних архітектурних підходів.

Для наочності та кращого розуміння логіки роботи системи наведено блок-схему, яка відображає основні етапи взаємодії користувача з додатком та послідовність обробки даних. У процесі розробки особлива увага приділяється створенню зручного інтерфейсу користувача, який дозволяє легко орієнтуватися в додатку. Інтерфейс реалізовано з урахуванням принципів адаптивного дизайну, що забезпечує коректне відображення на різних пристроях і роздільних здатностях екрана. Функціональні можливості додатку включають пошук тварин за різними параметрами (вид, вік, стать, місцезнаходження), додавання оголошень про улюбленців, перегляд детальної інформації, а також можливість зберігати вподобаних тварин до обраного для подальшого перегляду. Інтеграція з базою даних реалізована на основі технології Firebase Realtime Database, що дозволяє забезпечити швидку синхронізацію даних між клієнтською частиною додатку та хмарним сховищем. Для локального зберігання обраних тварин використовується Room Database, яка дозволяє зберігати дані на пристрої навіть без доступу до інтернету. Такий підхід дозволяє забезпечити стабільну роботу додатку навіть в умовах обмеженого або нестабільного інтернет-з'єднання.

Ця блок-схема демонструє на рисунку 3.11 алгоритм роботи системи.

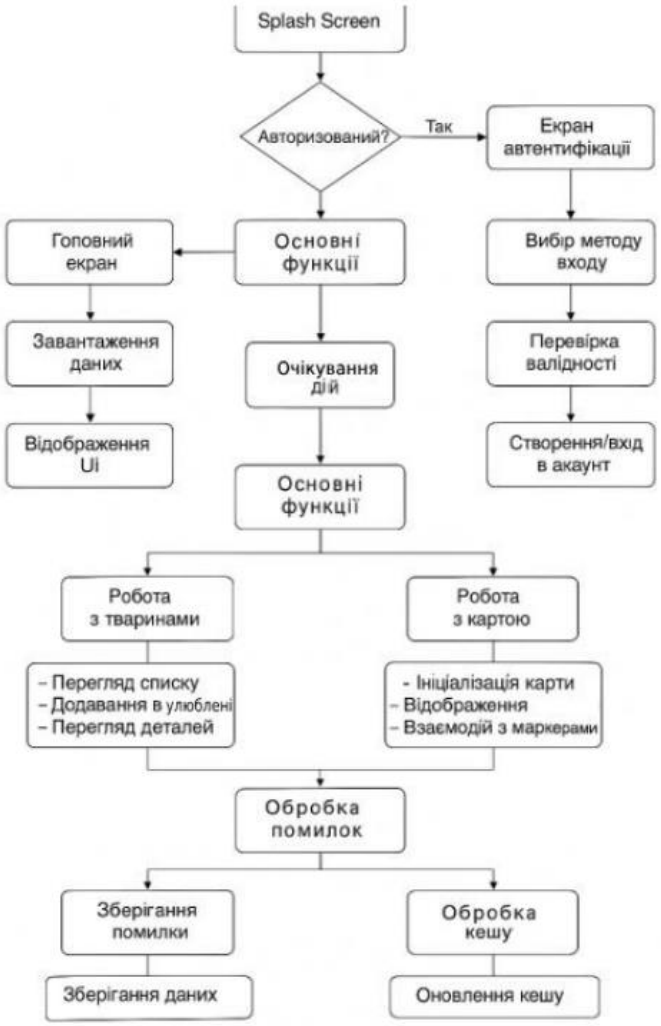

Рисунок 3.11 - Блок-схема алгоритму роботи системи

Архітектура інформаційно-пошукової системи базується на Clean Architecture, вона забезпечує чітке розділення відповідальностей шарами domain, data та presentation. Застосовується патерн MVVM (Model-View-ViewModel) для управління користувацьким інтерфейсом та бізнес-логікою. Використовуються і компоненти Android Architecture Components для побудови архітектури. Навігація між екранами реалізована завдяки Jetpack Navigation. Для управління залежностями використовується Hilt/Dagger. Джерела даних абстраговані через Repository Pattern, а бізнес-логіка інкапсульована в Use Cases, які описують кроки користувача для досягнення певних цілей. Use Case – методологія, вона в свою чергу описує всі кроки, зроблені юзером для досягнення цілей [17].

На основі цих Use Cases побудована і на рисунку 3.12 продемонстрована діаграма варіантів використання системи, яка наочно ілюструє взаємодію

користувачів із основними функціями додатку. Діаграма відображає ключові сценарії, такі як пошук домашніх улюбленців, додавання нового оголошення, перегляд детальної інформації про тварину, управління списком улюбленців та навігацію між цими функціями. Такий підхід дозволяє зрозуміло структурувати функціональність додатку і забезпечити прозорість у розумінні процесів, що відбуваються під час користування системою

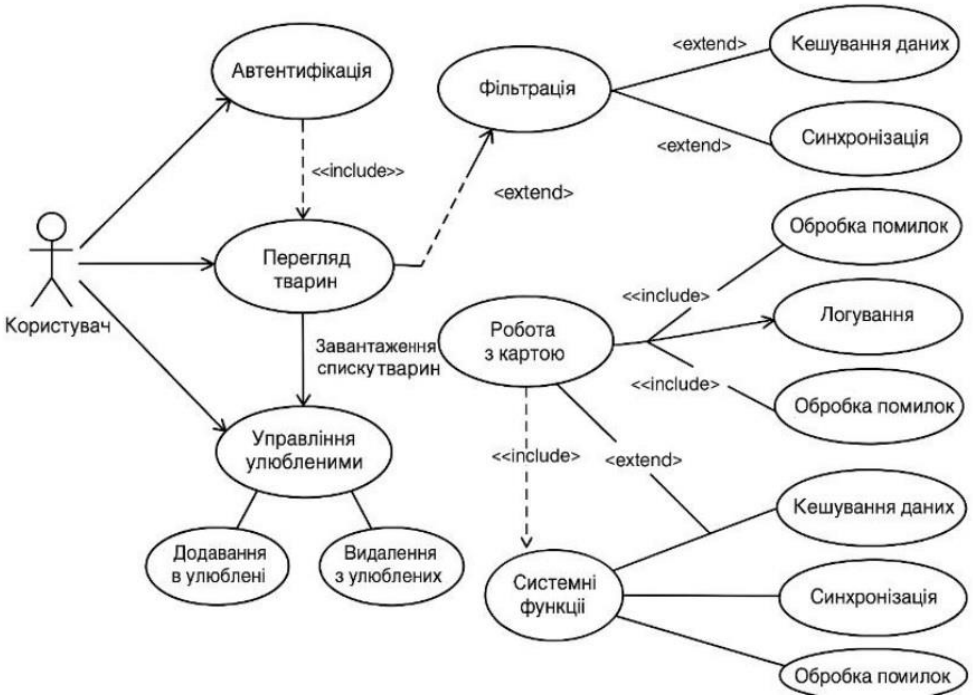

Рисунок 3.12 - Діаграма варіантів використання застосунку

Користувацький інтерфейс реалізовано з використанням Jetpack Compose - сучасного фреймворку для побудови UI, що базується на декларативному підході. Дизайн побудований відповідно до принципів Material Design 3. Основними будівельними блоками є Composable функції. Управління станом UI забезпечується через відповідні механізми, що дозволяють ефективно контролювати зміни інтерфейсу. Для створення плавних анімацій використано вбудовані засоби Compose та бібліотеки MotionLayout і ConstraintLayout. Для завантаження зображень застосовується легка бібліотека Coil, а розширені компоненти для Compose надає Accompanist.

Функціональна частина реалізована на основі сервісів Firebase. Firebase Authentication відповідає за автентифікацію користувачів, Firestore - за зберігання даних, а Firebase Storage - за збереження файлів. Для виявлення

помилок у додатку інтегровано Firebase Crashlytics. Геолокаційні функції реалізовані за допомогою Google Maps SDK, а також API-сервісів Places, Directions і Geocoding, які дозволяють працювати з місцями, прокладати маршрути та визначати адреси.

Асинхронне програмування реалізоване з використанням Kotlin Coroutines, а реактивні потоки - через Flow. Для локального зберігання даних використано Room Database, що забезпечує роботу з базою даних SQLite. Виконання фонових задач забезпечує WorkManager, а для збереження налаштувань додатку - DataStore.

У сфері безпеки реалізовано кілька важливих механізмів. Дані користувача захищені завдяки шифруванню за допомогою Encrypted Shared Preferences та бібліотеки Android Security, а також використанню ProGuard для захисту коду від реверс-інжинірингу. Автентифікація реалізована через Firebase Auth, а також підтримується біометрична автентифікація завдяки Biometric Auth.

3.4 Тестування роботи системи

У інформаційній системі FluffyFind основними компонентами інтерфейсу є кілька ключових екранів, які забезпечують користувачам зручний доступ до функціоналу платформи і без роботи яких користувач не зміг би повноцінно взаємодіяти з системою. Кожен з цих екранів виконує свою унікальну роль у процесі користування додатком, формуючи логічну та послідовну взаємодію між користувачем і сервісом.

Інтерфейс побудований з урахуванням принципів зручності (usability), інтуїтивності та швидкого доступу до основних функцій. Навігація між екранами проста та логічна, що дозволяє навіть недосвідченим користувачам легко орієнтуватися в системі.

Тому перед тестуванням системи розглянемо п’ять основних екранів інформаційної системи, які забезпечують повний цикл користування додатком - від перегляду та пошуку тварин до додавання власних оголошень і збереження

улюбленців. Ці екрани є базовими для реалізації ключових сценаріїв використання:

– MainActivity - головна активність показана на рисунку 3.13, яка ініціалізує додаток і керує навігацією між екранами.

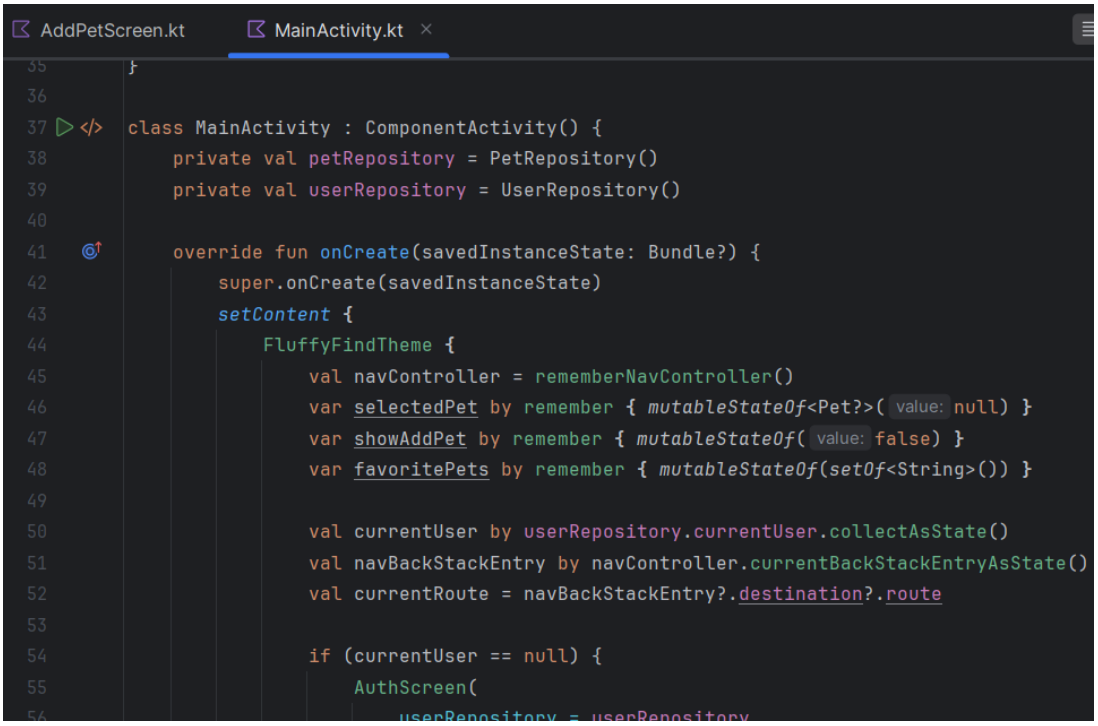

Рисунок 3.13 – Частина коду екрану MainActivity

– HomeScreen - головний екран, який відображає список доступних тварин, основну навігацію, а також можливість перегляду короткої інформації про кожну тварину. Саме з цього екрану який зображений на рисунку 3.14 користувач розпочинає роботу з додатком.

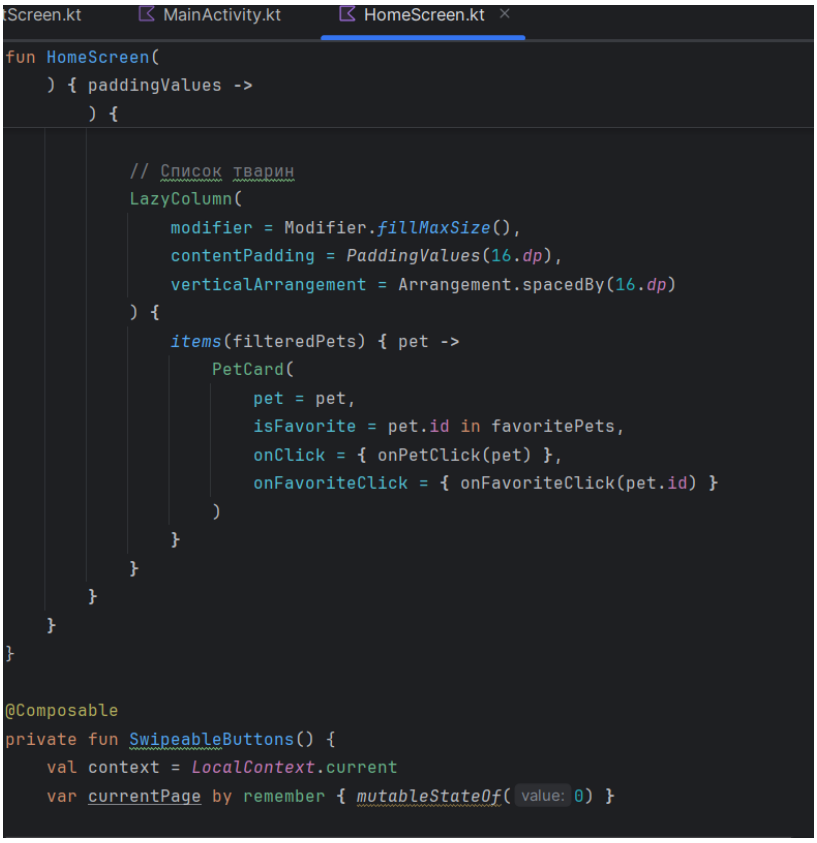

Рисунок 3.14 – Частина коду екрану HomeScreen

– AddPetScreen - екран для додавання нового оголошення про тварину. Користувач може вказати всі необхідні параметри: ім’я, тип тварини, опис, фото тощо. Цей екран, який можем бачити на рисунку 3.15, є ключовим для формування контенту в системі.

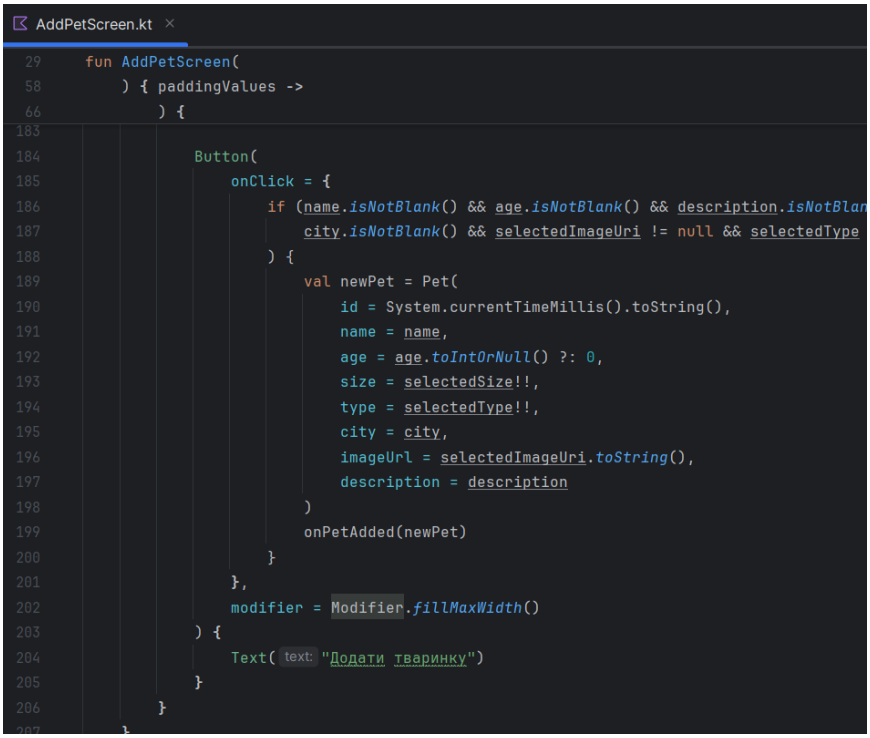

Рисунок 3.15 – Частина коду екрану AddPetScreen

– PetDetailsScreen - екран з детальною інформацією про вибрану тварину. На рисунку 3.16 відображаються всі характеристики тварини, контактні дані власника, а також можливість додати оголошення до улюблених.

Рисунок 3.16 – Частина коду екрану PetDetailScreen

– FavoritesScreen - зображений на рисунку 3.17 екран, де зберігаються всі тварини, які користувач відмітив як улюблені. Це дозволяє швидко повертатися до них у майбутньому без необхідності повторного пошуку.

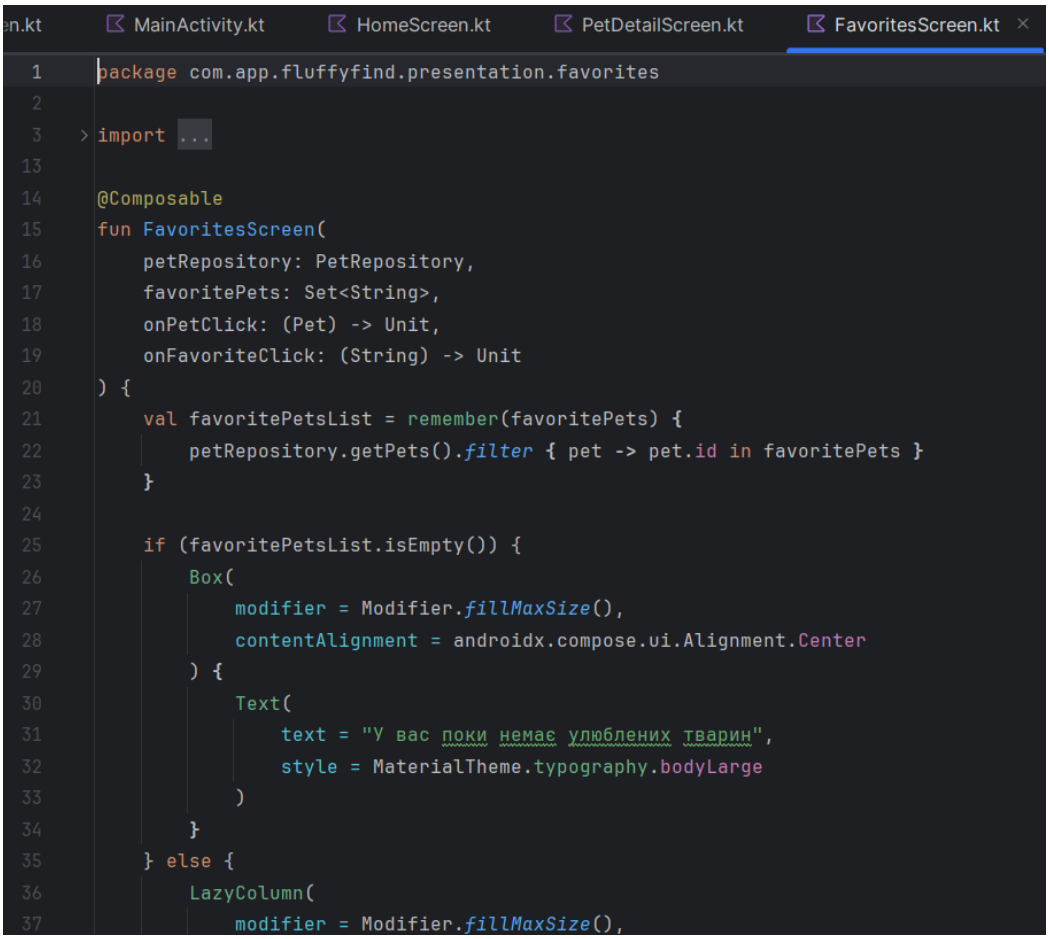

Рисунок 3.17 - Частина коду екрану FavoritesScreen

Після детального огляду ключових екранів додатку, що забезпечують основну функціональність системи — перегляд, пошук, додавання, збереження та детальний перегляд інформації про тварин — можемо перейти до фінального етапу розробки, який включає компіляцію та тестування інформаційно-пошукової системи для пошуку та обміну домашніми улюбленцями.

На цьому етапі перевіряється коректність роботи всіх компонентів, зручність користувацької взаємодії та стабільність роботи додатку в реальних умовах використання.

Спершу запускаємо інструментальне тестування All Test, застосунку в Android Studio [18].

Результати тестування зображено на рисунку 3.18.

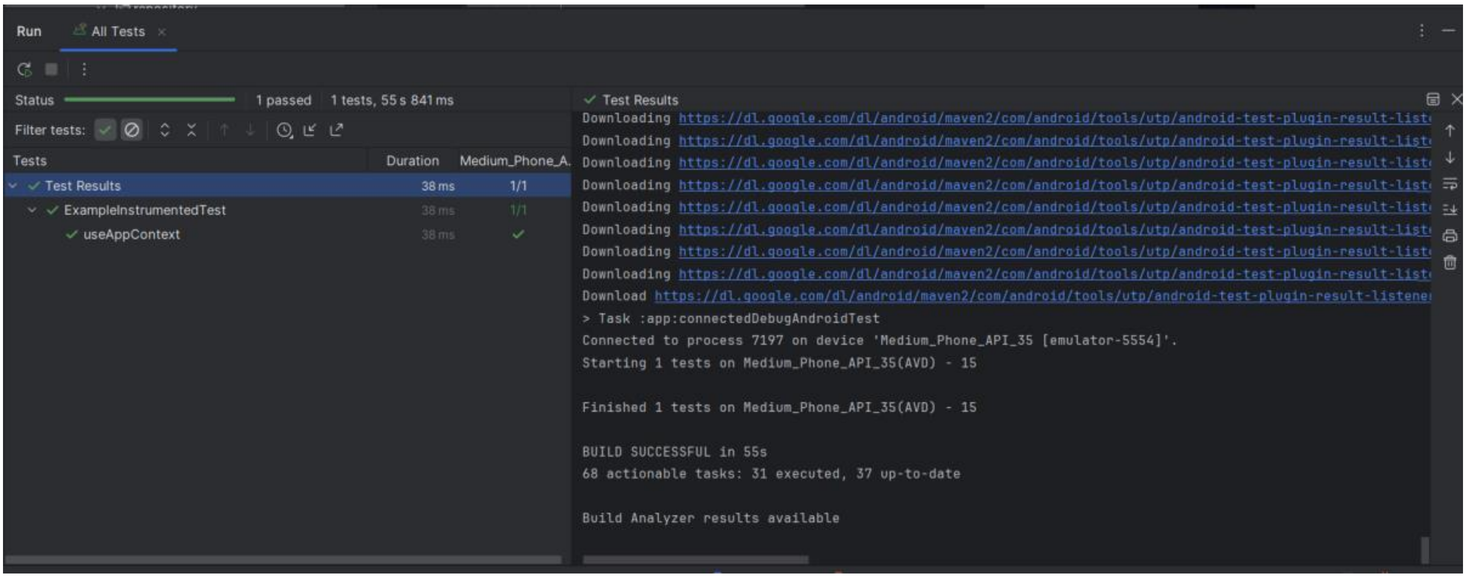

Рисунок 3.18 – Результат інструментального тестування

"1 passed, 1 tests, 55 s 841 ms": Це ключовий показник успішного тестування. Він чітко вказує на те, що один тест був запущений і успішно пройдений. Загальний час виконання тесту (близько 56 секунд) є прийнятним для одного інструментального тесту, хоча для більшої кількості тестів час виконання може зрости.

"BUILD SUCCESSFUL in 4m 14s": Це повідомлення від Gradle (система для автоматизації додатків та збору статистики про використання бібліотек, що застосовує такі мови як Groovy, Java, JavaScript, Kotlin, так далі) [19], яке свідчить про успішне завершення процесу збірки додатку. Усього процес тривав 4 хвилини та 14 секунд.

- "37 actionable tasks: 37 up-to-date": Повідомлення від Gradle показує, що всі 37 завдань збірки були вже актуальними (up-to-date) і не потребували повторного виконання, що свідчить про ефективність і оптимізовану збірку.
- "Task :app:assembleDebug": Це основне завдання, яке виконується для створення дебаг-версії додатку. Завдання завершено успішно.
- "Task :app:compileDebugKotlin": Завдання, яке відповідає за компіляцію коду Kotlin у дебаг-режимі. Цей етап також пройшов без помилок.

- "Task :app:processDebugManifest": Завдання, яке обробляє Android маніфест для дебаг-версії додатку. Всі необхідні зміни та налаштування були успішно внесені.

- "Task :app:mergeDebugResources": Завдання, яке об'єднує всі ресурси додатку для дебаг-версії. Це включає стилі, макети, зображення тощо, що успішно оброблено.

- "Task :app:packageDebug": Завдання, яке створює пакет дебаг-версії додатку, готовий до встановлення на пристрій або емулятор.

- "Task :app:validateSigningDebug": Це завдання перевіряє налаштування підпису для дебаг-версії додатку, щоб переконатися, що він правильно підписаний для тестування.

- "Task :app:assembleDebug UP-TO-DATE": Останнє завдання збірки, яке вказує на те, що дебаг-версія додатку була успішно зібрана і готова до тестування.

- "BUILD SUCCESSFUL": Фінальне повідомлення, воно підтверджує, що всі етапи збірки були успішно виконані, і дебаг-версія додатку готова для подальшого тестування та використання [20].

Проведене інструментальне тестування підтвердило коректну роботу мобільного додатку, а також стабільність його основних функціональних компонентів. У ході перевірки було встановлено, що всі передбачені функції працюють відповідно до заданих вимог і не викликають помилок під час використання.

Тестування охоплювало основні сценарії взаємодії користувача із системою, що дало змогу переконатися в надійності реалізованого функціоналу. Окрім цього, успішно виконано збірку дебаг-версії додатку без жодних помилок, що свідчить про правильну інтеграцію всіх модулів проєкту, відсутність конфліктів у залежностях та стабільну роботу програмного коду в середовищі розробки. Система поводилася передбачувано в усіх протестованих ситуаціях, що є важливим критерієм її якості. Це дозволяє зробити висновок про загальну готовність інформаційної системи до подальшого використання, зокрема її впровадження у практичну діяльність для вирішення поставлених задач.

3.5 Висновки

У цьому розділі було детально розглянуто процес проєктування та реалізації мобільного застосунку FluffyFind, який є інформаційною системою для пошуку та обміну домашніми улюбленцями. По-перше, було сформовано ефективну архітектуру застосунку, що поєднує патерн MVVM, реактивне управління станом, Client-Server взаємодію, хмарні сервіси Firebase та локальне зберігання налаштувань через SharedPreferences.

Завдяки використанню ViewModel, Repository, StateFlow та Firebase Security Rules, досягнуто високого рівня модульності, безпеки та масштабованості системи. По-друге, у процесі візуалізації інтерфейсу користувача було створено прототипи основних екранів у Figma з урахуванням принципів UX/UI дизайну. Особливу увагу приділено інтуїтивній навігації, легкому доступу до основного функціоналу, привабливому візуальному оформленню та зручності взаємодії з користувачем.

По-третє, у межах практичної реалізації застосунку реалізовано повний функціонал: від побудови інтерфейсу на основі Jetpack Compose до підключення Firebase Authentication, Firestore, Google Maps SDK, Room та інших інструментів. Застосовано сучасні практики - Clean Architecture, Flow, Coroutines, Hilt, що забезпечує стійкість, продуктивність і гнучкість додатку.

Таким чином, створений мобільний застосунок відповідає сучасним вимогам до програмних продуктів - він є безпечним, надійним, функціональним і зручним для користувачів. Результати розробки демонструють високий рівень інтеграції дизайну, архітектурного планування та технічної реалізації.

ВИСНОВКИ

Створено систему у виді додатку FluffyFind, який виконує роль як інформаційна платформа з пошуку й обміну домашніми улюбленцями. Проаналізовано наявні рішення в сфері адопції тварин і виявилено сильні і слабкі сторони, визначено найоптимальнішу технологію і платформу для створення додатку. Проведено тестування з урахуванням легкості у використанні, надійності та ефективності його функціонування.

Розроблена інформаційно-пошукова система FluffyFind демонструє ефективність у вирішенні актуальної соціальної проблеми адопції безпритульних тварин. Завдяки інтуїтивному інтерфейсу, локальній роботі без потреби постійного підключення до Інтернету та зручним інструментам для пошуку та публікації оголошень, додаток забезпечує зниження часових витрат користувача та підвищення загальної доступності процесу обміну домашніми улюбленцями.

Застосування сучасних технологій - Kotlin у поєднанні з Jetpack-компонентами дало змогу досягти високої стабільності, масштабованості та гнучкості розробки. Вибір архітектурних рішень був обґрунтований аналізом існуючих альтернатив і здійснений з урахуванням критеріїв продуктивності, підтримки спільнотою та відповідності вимогам Android платформи.

FluffyFind вигідно відрізняється від деяких наявних рішень спрощеною структурою взаємодії з користувачем, локалізацією під українського споживача та націленістю на конкретні потреби адопції. Критеріями ефективності розробки стали точність фільтрації результатів, швидкодія інтерфейсу та потенціал до масштабування - усі ці аспекти підтвердили задовільний рівень реалізації під час тестування.

За результатами даної роботи зроблено доповідь на тему «Інформаційна платформа для пошуку та обміну домашніми улюбленцями» на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2024-2025 рр.)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Волощук О.В., Козачко О.М. Інформаційна платформа для пошуку та обміну домашніми улюбленцями. «*LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації*». URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24349 (дата звернення: 24.03.2025).

2. Ми відповідальні за тих, кого приручили: проблема безпритульних тварин в Україні (інфографіка). URL: https://fact-news.com.ua/mi-vidpovidalni-za-tih-kogo-priruchili-problema-bezpritulnih-tvarin-v-ukraini-infografika/

3. Домашні тварини українців та їхні вподобання. URL: https://media-systems.ua/news/pet_food_consumers

4. Важливість Адопції домашнії тварин. URL: https://varangiandogs.org/blog/the-importance-of-pet-adoption--saving-lives-and-making-a-difference

5. Як мобільний додаток допомагає безпритульним тваринам. URL: https://hromadske.ua/posts/yak-mobilnyi-dodatok-dopomahaie-bezprytulnym-tvarynam

6. PetFinder.URL:https://play.google.com/store/apps/details?id=com.discovery.petfinder

7. Дні адопції у Києві 5-6 жовтня: скільки тварин знайшли новий дім. URL:https://nashkiev.ua/news/dni-adoptsii-u-kievi-5-6-zhovtnya-skilki-tvarin-znaishli-novii-dim

8. Єгоров, Сергій Юрійович. "Розробка Android застосунку засобами Kotlin та Firebase ML Kit." (2023) – 62с. URL: https://dspace.znu.edu.ua/jspui/bitstream/12345/18440/1/%D0%84%D0%B3%D0%BE%D1%80%D0%BE%D0%B2.pdf

9. Костіков, Микола Павлович. "Перспективи розроблення мобільних додатків із Kotlin." (2022). - 226 с. URL: https://dspace.nuft.edu.ua/items/aa776487-a402-496c-9d9c-fa849ce218b8

10. Android Studio: що це таке і для чого потрібна. URL: https://uk.androidayuda.com/андроїд-студія/

11. Android Studio: переваги та особливості. URL: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/

12. Тягунова, Марія Юріївна, and Р. В. Бойко. *Основне використання Firebase*. НУ" Запорізька політехніка", 2023. URL: https://eir.zp.edu.ua/server/api/core/bitstreams/53b94cb9-124e-42f6-a18f-d61e5f0dddbb/content

13. Technologies Room. URL: https://brander.ua/technologies/room

14. Що таке Figma та для кого вона потрібна. URL: https://cases.media/article/sho-take-figma-ta-dlya-kogo-vona-potribna?srsltid=AfmBOorwxESYu_f2Eh1zlwmD3KHpAO2qun97vkgwT8dpRN52n9V0LVUX%20

15. Що таке UI та UX дизайн? URL: https://te.itstep.org/blog/ui-and-ux-design

16. Дизайн мобільних додатків: 7 кроків створення з прикладами. URL: https://wezom.com.ua/ua/blog/dizajn-mobilnyh-prilozhenij-pochemu-on-vazhen-i-gde-zakazat

17. Use cases. Як покращити взаємодію користувача з системою. URL: https://surl.li/ltffdq

18. Introduction to Testing in Android Studio. URL: https://hyperskill.org/learn/step/40233

19. Коротко про Gradle. URL: https://abitap.com/16-4-korotko-pro-gradle/

20. Understanding Build Analyzer in Android. URL: https://blog.mindorks.com/build-analyzer-in-android/

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

ІНФОРМАЦІЙНА-ПОШУКОВА СИСТЕМА ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ОБМІНУ ДОМАШНІМИ УЛЮБЛЕНЦЯМИ

08-34.БКР.004.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Олексій КОЗАЧКО
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-21б
_____ Олег ВОЛОЩУК
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1)Android Studio, навчальний посібник. URL: https://developer.android.com/guide;

2) Як Мобільний додаток допомагає безпритульним тваринам, стаття. URL: https://hromadske.ua/posts/yak-mobilnyi-dodatok-dopomahaie-bezprytulnym-tvarynam

3) Костіков, Микола Павлович. "Перспективи розроблення мобільних додатків із Kotlin." URL: https://dspace.nuft.edu.ua/items/aa776487-a402-496c-9d9c-fa849ce218b8

3. Мета і призначення роботи.

Метою дослідження полягає у створенні інформаційної технології, яка дозволить спростити процес пошуку, перегляду та додавання інформації про тварин, тим самим забезпечуючи зручний і швидкий обмін даними між користувачами, зацікавленими в адопції.

4. Вихідні дані для проведення робіт:

База даних Firebase Realtime Database підключена до серверної частини Android Studio (Kotlin), платформа Jetpack Compose в середовищі розробки Android Studio з даними про домашніх тварин (коти, собаки, хом'яки, птахи, черепахи) та про користувачів додатку FluffyFind.

5. Методи дослідження:

Дослідження існуючих інформаційних систем, аналіз вимог, аналіз потреб користувача розробка Діаграм, створення макетів мобільного застосунку, тестування застосунку.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Аналіз потенційних користувачів і ситуації з адопцією _____ – _____
c) Вибір оптимальних інформаційних технологій _____ – _____
d) Розроблення інформаційно-пошукової системи _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційно-пошукової системи для підтримки прийняття рішень з обміну домашніми улюбленцями

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.

Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-21б _____ Олег ВОЛОЩУК

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна-пошукова система для підтримки прийняття рішень з обміну домашніми улюбленцями»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,25 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Євгеній КРИЖАНОВСЬКИЙ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ

Здобувач _____ (підпис) Олег ВОЛОЩУК

Додаток В

(довідниковий)

Фрагмент лістингу програми

Код Файлу додатку Navigation.kt

```
package com.app.fluffyfind.navigation

import androidx.compose.foundation.layout.padding
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.automirrored.filled.Message
import androidx.compose.material.icons.filled.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.navigation.NavDestination.Companion.hierarchy
import androidx.navigation.NavGraph.Companion.findStartDestination
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.currentBackStackEntryAsState
import androidx.navigation.compose.rememberNavController
import com.app.fluffyfind.data.repository.PetRepository
import com.app.fluffyfind.data.repository.UserRepository
import com.app.fluffyfind.presentation.auth.AuthScreen
import com.app.fluffyfind.presentation.favorites.FavoritesScreen
import com.app.fluffyfind.presentation.home.HomeScreen
import com.app.fluffyfind.presentation.messages.MessagesScreen
import com.app.fluffyfind.presentation.profile.ProfileScreen
import com.app.fluffyfind.presentation.splash.SplashScreen

@Composable
fun AppNavigation() {
    val navController = rememberNavController()
    val petRepository = PetRepository()
    val userRepository = UserRepository()
    var favoritePets by remember { mutableStateOf<Set<String>>(setOf()) }
    var selectedPet by remember { mutableStateOf<com.app.fluffyfind.data.model.Pet?>(null) }

    val currentUser by userRepository.currentUser.collectAsState()

    NavHost(
        navController = navController,
        startDestination = "splash"
    ) {
        composable("splash") {
            SplashScreen(navController = navController)
        }
```

```
composable("auth") {
    if (currentUser == null) {
        AuthScreen(
            userRepository = userRepository,
            onAuthSuccess =
        )
    } else {
        navController.navigate("home") {
            popUpTo("auth") { inclusive = true }
        }
    }
}

composable("home") {
    HomeScreen(
        petRepository = petRepository,
        userRepository = userRepository,
        favoritePets = favoritePets,
        onPetClick = { pet -> selectedPet = pet },
        onFavoriteClick = { petId ->
            favoritePets = if (petId in favoritePets) {
                favoritePets - petId
            } else {
                favoritePets + petId
            }
        },
        onProfileClick = {
            navController.navigate("profile")
        }
    )
}

composable("favorites") {
    FavoritesScreen(
        petRepository = petRepository,
        favoritePets = favoritePets,
        onPetClick = { pet -> selectedPet = pet },
        onFavoriteClick = { petId ->
            favoritePets = if (petId in favoritePets) {
                favoritePets - petId
            } else {
                favoritePets + petId
            }
        }
    )
}

composable("profile") {
    ProfileScreen(
        userRepository = userRepository,
```

```
                onLogout = { userRepository.logout() },
                onOpenMap = { /* Nothing to do for now */ }
            )
        }
    }

    // Показуємо екран деталей тварини, якщо вона вибрана
    selectedPet?.let { pet ->
        com.app.fluffyfind.presentation.detail.PetDetailScreen(
            pet = pet,
            isFavorite = pet.id in favoritePets,
            onBackClick = { selectedPet = null },
            onFavoriteClick = {
                favoritePets = if (pet.id in favoritePets) {
                    favoritePets - pet.id
                } else {
                    favoritePets + pet.id
                }
            }
        )
    }
}

sealed class Screen(val route: String, val icon: androidx.compose.ui.graphics.vector.ImageVector,
val label: String) {
    object Home : Screen("home", Icons.Default.Home, "Головна")
    object Favorites : Screen("favorites", Icons.Default.Favorite, "Улюблені")
    object Messages : Screen("messages", Icons.AutoMirrored.Filled.Message, "Повідомлення")
    object Profile : Screen("profile", Icons.Default.Person, "Профіль")
}

private val items = listOf(
    Screen.Home,
    Screen.Favorites,
    Screen.Messages,
    Screen.Profile
)
```

Код Файлу додатку HomeScreen.kt

```
package com.app.fluffyfind.presentation.home

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.gestures.detectHorizontalDragGestures
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.rememberScrollState
```

```
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Favorite
import androidx.compose.material.icons.filled.FavoriteBorder
import androidx.compose.material.icons.filled.FilterList
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.input.pointer.pointerInput
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import coil.compose.AsyncImage
import com.app.fluffyfind.R
import com.app.fluffyfind.data.model.Pet
import com.app.fluffyfind.data.model.PetSize
import com.app.fluffyfind.data.model.PetType
import com.app.fluffyfind.data.repository.PetRepository
import com.app.fluffyfind.data.repository.UserRepository
import androidx.compose.material3.MenuAnchorType
import androidx.compose.ui.graphics.Color
import androidx.compose.runtime.collectAsState

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun HomeScreen(
    petRepository: PetRepository,
    userRepository: UserRepository,
    favoritePets: Set<String>,
    onPetClick: (Pet) -> Unit,
    onFavoriteClick: (String) -> Unit,
    onProfileClick: () -> Unit
) {
    var showFilters by remember { mutableStateOf(false) }
    var selectedType by remember { mutableStateOf<PetType?>(null) }
    var selectedSize by remember { mutableStateOf<PetSize?>(null) }
    var selectedCity by remember { mutableStateOf<String?>(null) }
    var selectedAgeRange by remember { mutableStateOf<IntRange?>(null) }

    val currentUser by userRepository.currentUser.collectAsState(initial = null)
    val pets by petRepository.pets.collectAsState(initial = emptyList())

    LaunchedEffect(Unit) {
        petRepository.getPets()
    }
```

```
val availableTypes = remember(selectedSize, selectedCity, selectedAgeRange) {
    petRepository.getAvailableTypes(selectedSize, selectedCity, selectedAgeRange)
}
val availableSizes = remember(selectedType, selectedCity, selectedAgeRange) {
    petRepository.getAvailableSizes(selectedType, selectedCity, selectedAgeRange)
}
val availableCities = remember(selectedType, selectedSize, selectedAgeRange) {
    petRepository.getAvailableCities(selectedType, selectedSize, selectedAgeRange)
}
val availableAgeRanges = remember(selectedType, selectedSize, selectedCity) {
    petRepository.getAvailableAgeRanges(selectedType, selectedSize, selectedCity)
}

val filteredPets = remember(pets, selectedType, selectedSize, selectedCity, selectedAgeRange) {
    pets.filter { pet ->
        val typeMatch = selectedType == null || pet.type == selectedType
        val sizeMatch = selectedSize == null || pet.size == selectedSize
        val cityMatch = selectedCity == null || pet.city.equals(selectedCity, ignoreCase = true)
        val ageMatch = when (val range = selectedAgeRange) {
            null -> true
            else -> pet.age >= range.first && pet.age <= range.last
        }

        typeMatch && sizeMatch && cityMatch && ageMatch
    }
}

Scaffold(
    topBar = {
        TopAppBar(
            title = {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
                        .fillMaxHeight(),
                    contentAlignment = Alignment.Center
                ) {
                    Image(
                        painter = painterResource(id = R.drawable.label1),
                        contentDescription = "FluffyFind",
                        modifier = Modifier
                            .size(300.dp)
                    )
                }
            },
            actions = {
                Box(
                    modifier = Modifier
                        .fillMaxHeight(),
                    contentAlignment = Alignment.Center
```

```
            ) {
                currentUser?.photoUrl?.let { photoUrl ->
                    AsyncImage(
                        model = photoUrl,
                        contentDescription = "Profile Photo",
                        modifier = Modifier
                            .size(40.dp)
                            .clip(CircleShape)
                            .clickable(onClick = onProfileClick),
                        contentScale = ContentScale.Crop
                    )
                } ?: run {
                    IconButton(onClick = onProfileClick) {
                        Icon(
                            painter = painterResource(id = R.drawable.ic_profile),
                            contentDescription = "Profile",
                            modifier = Modifier.size(40.dp)
                        )
                    }
                }
            }
        },
        colors = TopAppBarDefaults.topAppBarColors(
            containerColor = MaterialTheme.colorScheme.primary
        ),
        modifier = Modifier
            .padding(horizontal = 12.dp, vertical = 4.dp)
            .clip(MaterialTheme.shapes.large)
            .height(80.dp)
    )
  }
) { paddingValues ->
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(paddingValues)
    ) {
        // Кнопки з можливістю свайпати
        SwipeableButtons()

        // Фільтри
        Button(
            onClick = { showFilters = !showFilters },
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp, vertical = 8.dp)
        ) {
            Row(
                verticalAlignment = Alignment.CenterVertically,
                horizontalArrangement = Arrangement.Center,
```

```
                modifier = Modifier.fillMaxWidth()
            ) {
                Icon(
                    imageVector = Icons.Default.FilterList,
                    contentDescription = "Фільтри"
                )
                Spacer(modifier = Modifier.width(8.dp))
                Text("Фільтри")
            }
        }

        if (showFilters) {
            FilterSection(
                selectedType = selectedType,
                selectedSize = selectedSize,
                selectedCity = selectedCity,
                selectedAgeRange = selectedAgeRange,
                availableTypes = availableTypes,
                availableSizes = availableSizes,
                availableCities = availableCities,
                availableAgeRanges = availableAgeRanges,
                onTypeSelected = { selectedType = it },
                onSizeSelected = { selectedSize = it },
                onCitySelected = { selectedCity = it },
                onAgeRangeSelected = { selectedAgeRange = it }
            )
        }

        // Список тварин
        LazyColumn(
            modifier = Modifier.fillMaxSize(),
            contentPadding = PaddingValues(16.dp),
            verticalArrangement = Arrangement.spacedBy(16.dp)
        ) {
            items(filteredPets) { pet ->
                PetCard(
                    pet = pet,
                    isFavorite = pet.id in favoritePets,
                    onClick = { onPetClick(pet) },
                    onFavoriteClick = { onFavoriteClick(pet.id) }
                )
            }
        }
    }
  }
}

@Composable
private fun SwipeableButtons() {
    val context = LocalContext.current
```

```
    var currentPage by remember { mutableStateOf(0) }

    Box(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp)
            .pointerInput(Unit) {
                detectHorizontalDragGestures { change, dragAmount ->
                    if (dragAmount > 50) { // Свайп вправо
                        currentPage = (currentPage - 1).coerceIn(0, 1)
                    } else if (dragAmount < -50) { // Свайп вліво
                        currentPage = (currentPage + 1).coerceIn(0, 1)
                    }
                }
            },
        contentAlignment = Alignment.Center
    ) {
        when (currentPage) {
            0 -> Image(
                painter = painterResource(id = R.drawable.button),
                contentDescription = "Instagram",
                modifier = Modifier
                    .clickable {
                        val intent = Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.instagram.com/fluffyfind/"))
                        context.startActivity(intent)
                    }
            )
            1 -> Image(
                painter = painterResource(id = R.drawable.button2),
                contentDescription = "Monobank",
                modifier = Modifier
                    .clickable {
                        val intent = Intent(Intent.ACTION_VIEW,
Uri.parse("https://send.monobank.ua/jar/85iN5uicrP"))
                        context.startActivity(intent)
                    }
            )
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun FilterSection(
    selectedType: PetType?,
    selectedSize: PetSize?,
    selectedCity: String?,
    selectedAgeRange: IntRange?,
    availableTypes: List<PetType>,
```

```
    availableSizes: List<PetSize>,
    availableCities: List<String>,
    availableAgeRanges: List<IntRange>,
    onTypeSelected: (PetType?) -> Unit,
    onSizeSelected: (PetSize?) -> Unit,
    onCitySelected: (String?) -> Unit,
    onAgeRangeSelected: (IntRange?) -> Unit
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp)
    ) {
        Column(
            modifier = Modifier.padding(16.dp),
            verticalArrangement = Arrangement.spacedBy(16.dp)
        ) {
            TypeFilterDropdown(
                selectedType = selectedType,
                availableTypes = availableTypes,
                onTypeSelected = onTypeSelected
            )
            SizeFilterDropdown(
                selectedSize = selectedSize,
                availableSizes = availableSizes,
                onSizeSelected = onSizeSelected
            )
            CityFilterDropdown(
                selectedCity = selectedCity,
                availableCities = availableCities,
                onCitySelected = onCitySelected
            )
            AgeRangeFilterDropdown(
                selectedAgeRange = selectedAgeRange,
                availableAgeRanges = availableAgeRanges,
                onAgeRangeSelected = onAgeRangeSelected
            )
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun TypeFilterDropdown(
    selectedType: PetType?,
    availableTypes: List<PetType>,
    onTypeSelected: (PetType?) -> Unit
) {
    FilterDropdown(
        label = "Вид тварини",
```

```
        selectedOption = selectedType?.name ?: "Bci",
        options = listOf("Bci") + availableTypes.map { it.name },
        onOptionSelected = { option ->
            onTypeSelected(if (option == "Bci") null else PetType.valueOf(option))
        }
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun SizeFilterDropdown(
    selectedSize: PetSize?,
    availableSizes: List<PetSize>,
    onSizeSelected: (PetSize?) -> Unit
) {
    FilterDropdown(
        label = "Розмір",
        selectedOption = selectedSize?.name ?: "Bci",
        options = listOf("Bci") + availableSizes.map { it.name },
        onOptionSelected = { option ->
            onSizeSelected(if (option == "Bci") null else PetSize.valueOf(option))
        }
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun CityFilterDropdown(
    selectedCity: String?,
    availableCities: List<String>,
    onCitySelected: (String?) -> Unit
) {
    FilterDropdown(
        label = "Місто",
        selectedOption = selectedCity ?: "Bci",
        options = listOf("Bci") + availableCities,
        onOptionSelected = { option ->
            onCitySelected(if (option == "Bci") null else option)
        }
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun AgeRangeFilterDropdown(
    selectedAgeRange: IntRange?,
    availableAgeRanges: List<IntRange>,
    onAgeRangeSelected: (IntRange?) -> Unit
) {
    FilterDropdown(
```

```
        label = "Вік",
        selectedOption = selectedAgeRange?.let { "${it.first}-${it.last} років" } ?: "Всі",
        options = listOf("Всі") + availableAgeRanges.map { "${it.first}-${it.last} років" },
        onOptionSelected = { option ->
            if (option == "Всі") {
                onAgeRangeSelected(null)
            } else {
                val (start, end) = option.split("-").map { it.toInt() }
                onAgeRangeSelected(start..end)
            }
        }
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
private fun FilterDropdown(
    label: String,
    selectedOption: String,
    options: List<String>,
    onOptionSelected: (String) -> Unit
) {
    var expanded by remember { mutableStateOf(false) }

    Column {
        Text(
            text = label,
            style = MaterialTheme.typography.labelMedium
        )
        ExposedDropdownMenuBox(
            expanded = expanded,
            onExpandedChange = { expanded = it }
        ) {
            OutlinedTextField(
                value = selectedOption,
                onValueChange = {},
                readOnly = true,
                trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded) },
                modifier = Modifier
                    .fillMaxWidth()
                    .menuAnchor(MenuAnchorType.PrimaryNotEditable)
            )
            ExposedDropdownMenu(
                expanded = expanded,
                onDismissRequest = { expanded = false }
            ) {
                options.forEach { option ->
                    DropdownMenuItem(
                        text = { Text(option) },
                        onClick = {
```

```
                    onOptionSelected(option)
                    expanded = false
                }
            )
        }
    }
  }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun PetCard(
  pet: Pet,
  modifier: Modifier = Modifier,
  isFavorite: Boolean = false,
  onFavoriteClick: () -> Unit,
  onClick: () -> Unit
) {
  Card(
    modifier = modifier
      .fillMaxWidth()
      .clickable(onClick = onClick),
    elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
  ) {
    Box {
      IconButton(
        onClick = onFavoriteClick,
        modifier = Modifier
          .align(Alignment.TopEnd)
          .padding(8.dp)
      ) {
        Icon(
          imageVector = if (isFavorite) Icons.Default.Favorite else
Icons.Default.FavoriteBorder,
          contentDescription = if (isFavorite) "Видалити з улюблених" else "Додати в
улюблені",
          tint = if (isFavorite) MaterialTheme.colorScheme.primary else
MaterialTheme.colorScheme.onSurface
        )
      }

      Column {
        AsyncImage(
          model = pet.imageUrl,
          contentDescription = "Photo of ${pet.name}",
          modifier = Modifier
            .fillMaxWidth()
            .height(200.dp),
          contentScale = ContentScale.Crop
```

```
            )

            Column(
                modifier = Modifier.padding(16.dp),
                verticalArrangement = Arrangement.spacedBy(8.dp)
            ) {
                Text(
                    text = pet.name,
                    style = MaterialTheme.typography.titleLarge,
                    fontWeight = FontWeight.Bold
                )

                Row(
                    modifier = Modifier.fillMaxWidth(),
                    horizontalArrangement = Arrangement.SpaceBetween
                ) {
                    Text("Вік: ${pet.age} років")
                    Text("Розмір: ${pet.size.name}")
                }

                Text("Вид: ${pet.type.name}")
                Text("Місто: ${pet.city}")

                Text(
                    text = pet.description,
                    style = MaterialTheme.typography.bodyMedium
                )
            }
        }
    }
}
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНО-ПОШУКОВА СИСТЕМА ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З ОБМІНУ ДОМАШНІМИ УЛЮБЛЕНЦЯМИ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

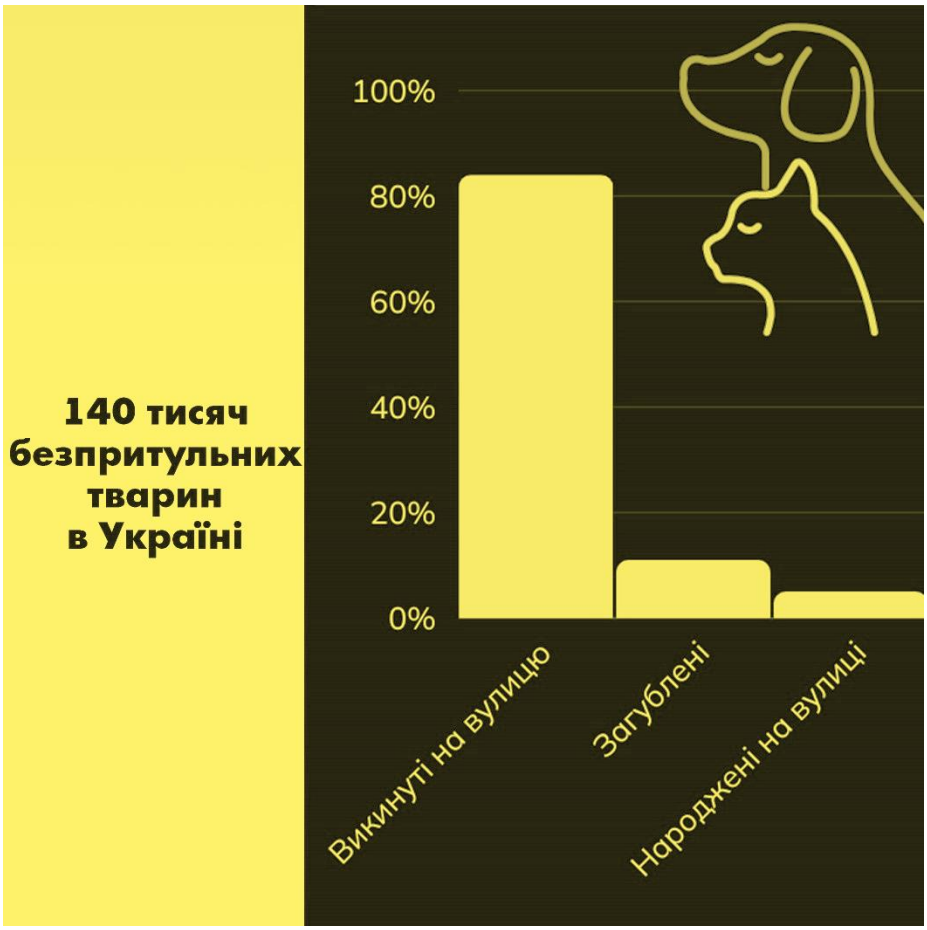

Рисунок Г.1 – Статистика Безпритульних Тварин в Україні

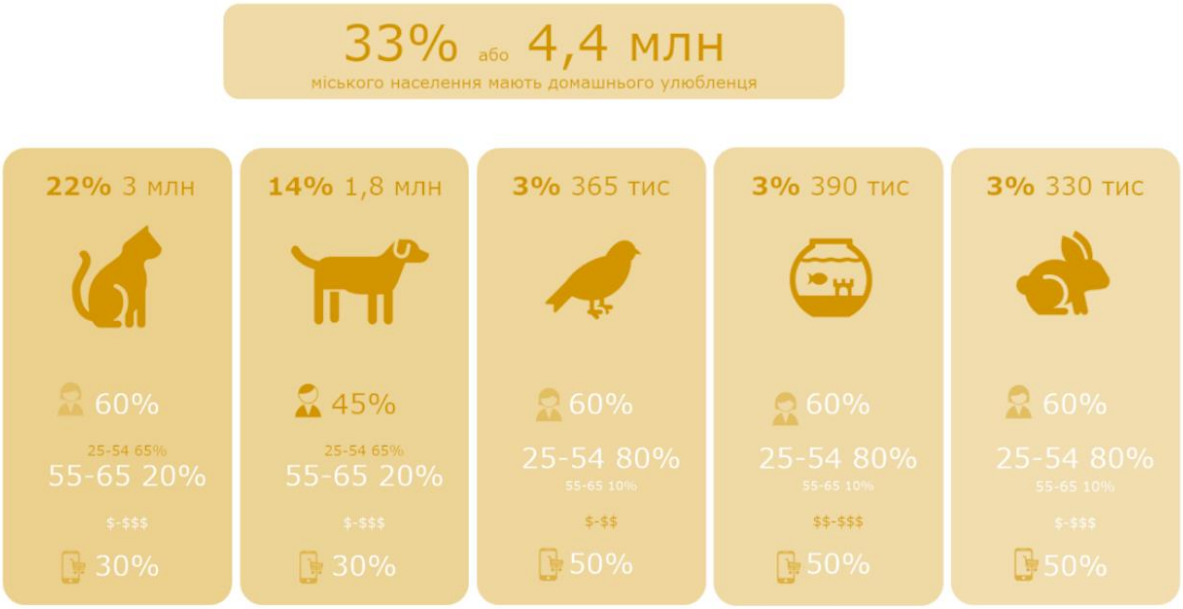

Рисунок Г.2 – Статистика популярності домашніх тваринок серед українців

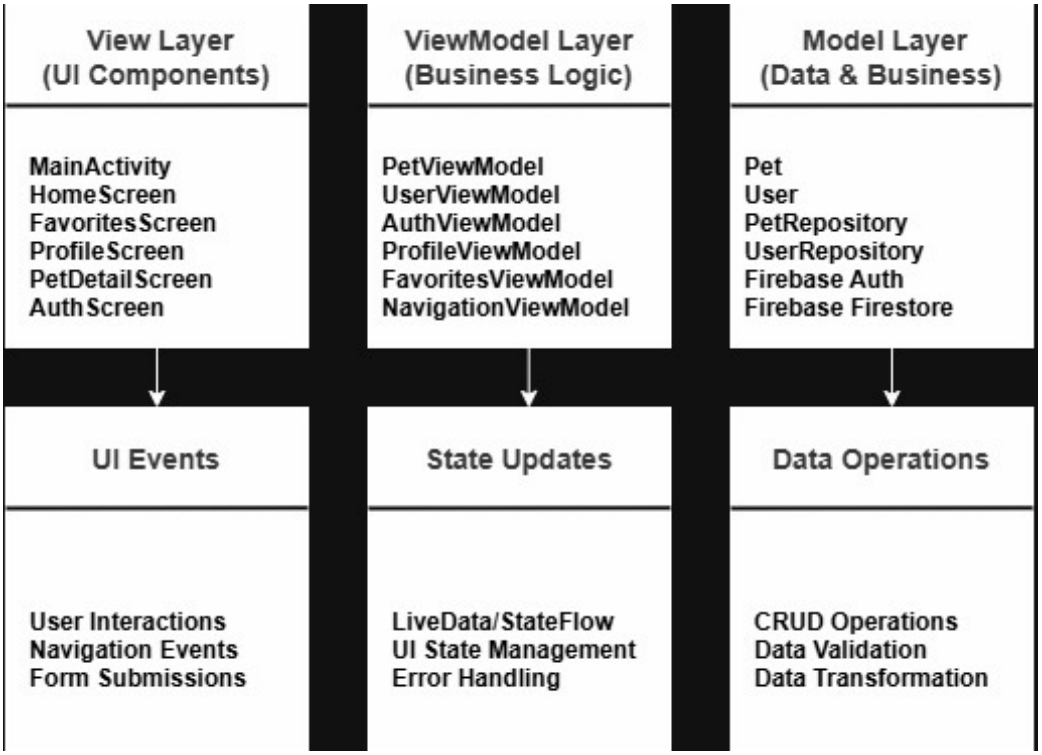

Рисунок Г.3 – Діаграма архітектури MVVM

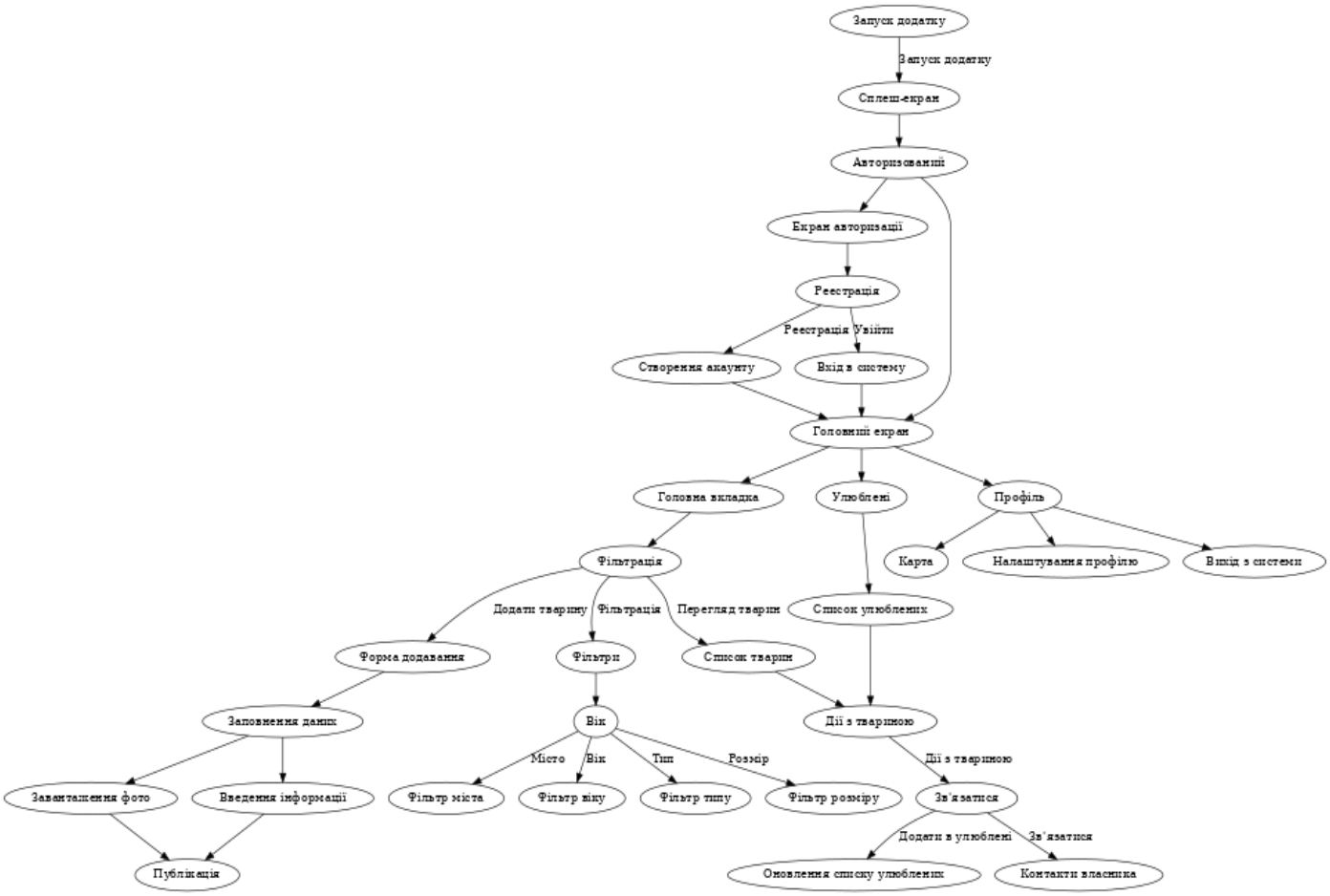

Рисунок Г.4 – Структура роботи інформаційної системи «FluffyFind»

Рисунок Г.5 - Логотип Інформаційно-пошукової системи «FluffyFind»

Рисунок Г.6 - Головна сторінка

Рисунок Г.7 - Функціонал фільтрів

Рисунок Г.8 – Сторінка тварини

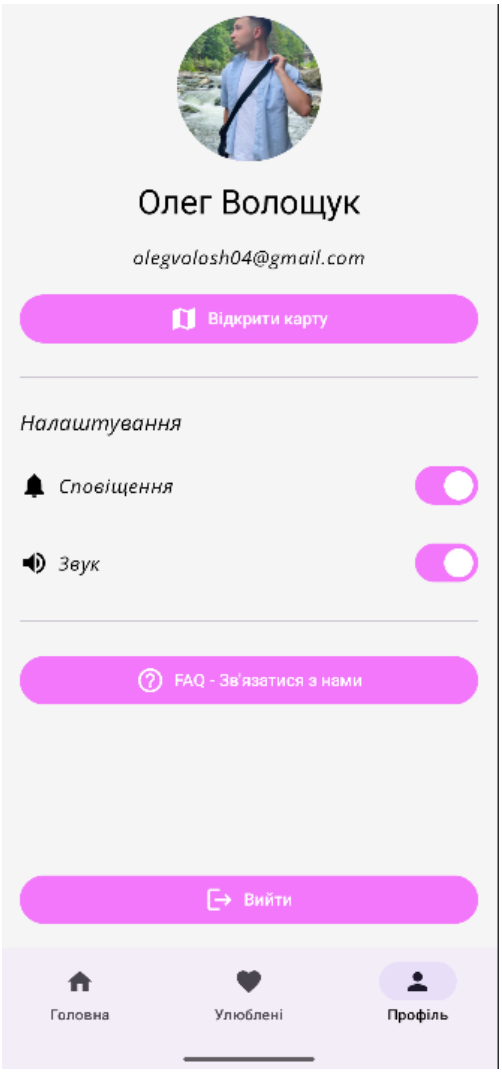

Рисунок Г.9 - Профіль користувача

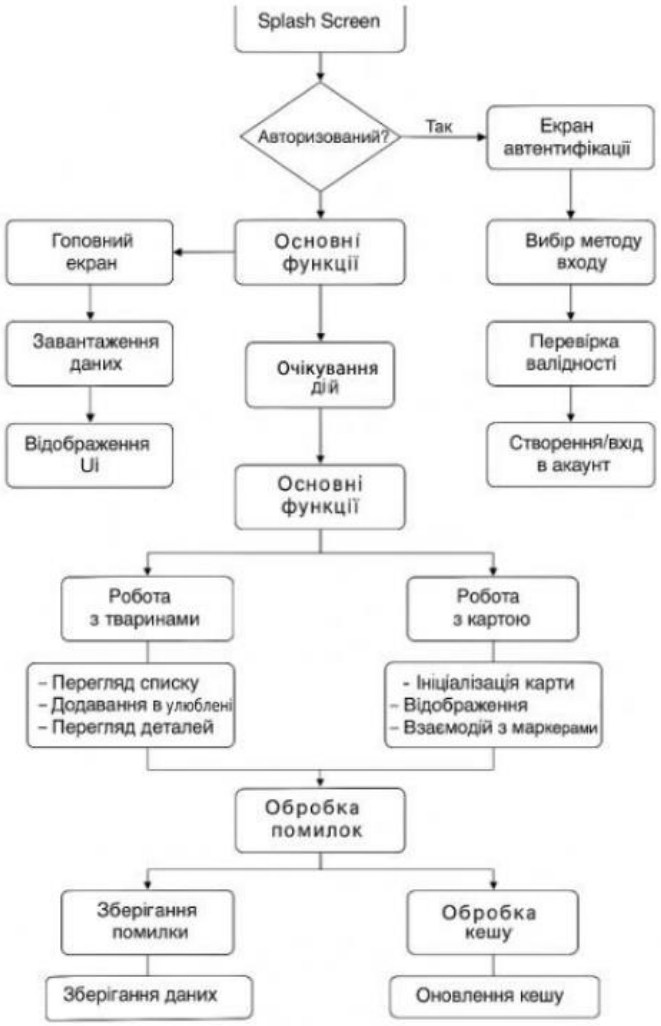

Рисунок Г.10 - Блок-схема алгоритму роботи системи

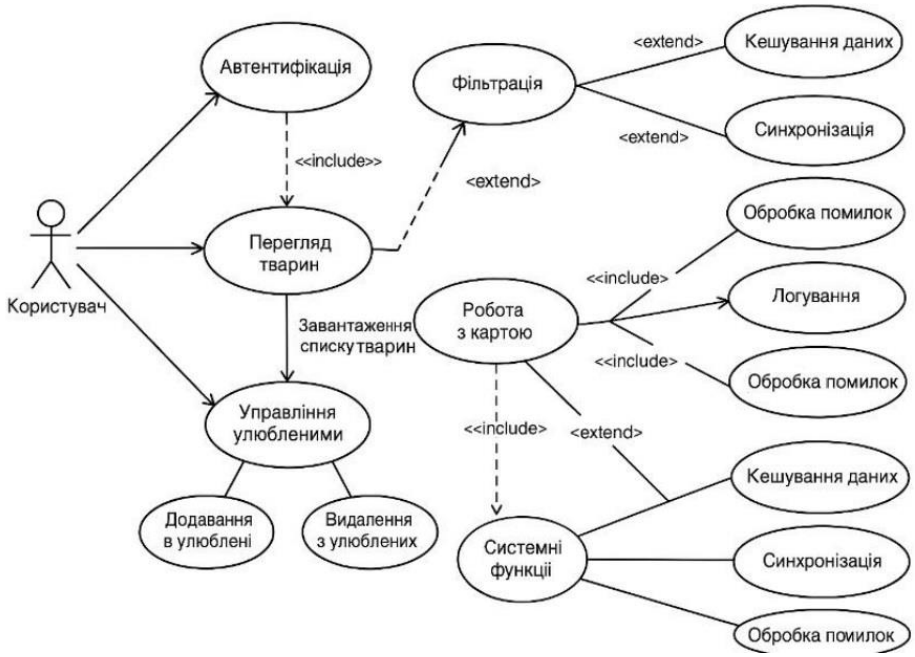

Рисунок Г.11 - Діаграма варіантів використання застосунку