

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Програмування мобільних пристроїв»

за темою

«Створення мобільного додатку для пошуку зниклих тварин PawPatrol»

Пояснювальна записка

Виконали:
студентки 3-го курсу
групи НАІ-196
Кожухарь К.О.
Резвіна А.С.
Перевірив:
Годовіченко М. А.

Одеса-2022

ЗМІСТ

ЗАВДАННЯ	3
АНОТАЦІЯ	4
ВСТУП.....	5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ..	6
1.1 Проблемний аналіз існуючих програмних продуктів	6
1.2 Вибір інструментів розробки	7
1.2.1 Вибір мови програмування	7
1.2.2 Вибір середовища розробки.....	9
1.2.3 Вибір системи управління базами даних.....	10
2 ПРОЕКТУВАННЯ.....	12
2.1 Функціональні вимоги.....	12
2.2 Діаграма використання (Use cases diagram) програми та її опис	12
2.3 Ідентифікація архетипу додатка	15
2.4 Логічне представлення (Logical View).....	16
2.5 Уявлення процесів (Process View).....	17
2.5.1 Діаграма діяльності.....	18
2.5.2 Діаграма послідовності	23
2.6 Подання даних (Data View).....	26
3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ.....	28
3.1 Структура проекту	28
3.2 Опис класів додатку	29
3.3 Збереження даних у Firebase.....	32
3.4 Результати виконання у вигляді скриншотів додатку.....	35
ВИСНОВКИ.....	43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	44
ДОДАТКИ.....	45
Додаток А.....	45
Додаток Б	46

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ

Студенткам Резвіній Анні Сергіївні, група НАІ-196
Кожухарь Ксенії Олександрівні група НАІ-196

1. Тема роботи «Створення мобільного додатку для пошуку зниклих тварин PawPatrol»
2. Термін здачі студентом закінченої роботи 17.06.2022
3. Початкові дані до проекту (роботи): методичні вказівки до виконання курсової роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити) постановка задачі, аналіз предметної області, вибір та опис інструментів розробки, проектування, розробка та тестування програмного продукту
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) діаграма класів

Завдання видано 10.03.2022

(підпис викладача)

Завдання прийнято до виконання 10.03.2022

АНОТАЦІЯ

В роботі розглянуто розробку мобільного додатку для операційної системи Android PawPatrol, його конструювання та реалізацію на мові програмування Kotlin. Для роботи з даними було використано структуру даних у сервісі для зберігання даних Firebase Realtime. В ході виконання завдань було отримано нові навички розробки мобільних додатків для операційної системи Android на мові програмування Kotlin, роботи з сервісом Firebase Realtime, аналізу цільової аудиторії, закріплено навички проектування програмного продукту.

ANNOTATION

This term paper considers the development of a mobile application for the Android PawPatrol operating system, its design and implementation in the Kotlin programming language. The data structure in the Firebase Realtime data storage service was used to work with the data. During the tasks, new skills of developing mobile applications for the Android operating system in the Kotlin programming language, working with the Firebase Realtime service, analyzing the target audience, and skills of software product design were acquired.

ВСТУП

В курсовій роботі розглядається процес створення програмного продукту «PawPatrol» на етапах визначення вимог до програмного продукту та планування процесів розробки. Робота виконувалась в команді з двох учасниць: Кожухарь К.О, Резвіна А.С.

Робота пов'язана з такими потребами споживача як пошук загублених тварин або пошук хазяїв знайдених тварин. Аналіз вказаних потреб визначив інформаційну потребу – доступність інформації про місцезнаходження загубленої або знайденої тварини, а також про доступність комунікації для досягнення головної мети - повернення тварини її господарю. Вміст пояснювальної записки розділів «Аналіз предметної області та вибір інструментів розробки», «Проектування програмного продукту» та «Реалізація програмного продукту» співпадає зі вмістом пояснювальної записки інших учасників проектної команди. У розділах детальніше наведено опис особливостей конструювання: структур даних у сервісі для зберігання даних Firebase Realtime; програмних модулів в інструментальному середовищі Android Studio з використанням мови програмування Kotlin. При визначенні ступеня готовності існуючих програмних продуктів до вирішення інформаційної потреби проаналізовано наступні програмні продукти: NextDoor Pet Directory, PawBoost, PetFinder, PawScout.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ІНСТРУМЕНТІВ РОЗРОБКИ

1.1 Проблемний аналіз існуючих програмних продуктів

Для аналізу аналогів програмного продукту, що створюється, було проаналізовано деякі існуючі додатки та сервіси: якщо версія додатку була доступна в Україні, то його було завантажено на смартфон та протестовано власноруч; якщо версія недоступна, то було прочитано відкуги з офіційних сторінок розробника та на сторінках у соціальних мережах. Усього було розглянуто 4 додатки: NextDoor Pet Directory, PawBoost, PetFinder, PawScout. Результати порівняння детально наведені у таблиці 1.1

Таблиця 1.1 - Аналіз аналогів створюваного продукту

Назва продукту	Доступність	Переваги	Недоліки
NextDoor Pet Directory	Додаток доступний тільки у США за персональним запрошенням	Високий рівень надійності та безпеки; Велика база типів тварин для сортування; Дуже детальна форма для створення заявки про зниклу тварину	Важкий процес верифікації користувача; Користувачі мають доступ лише до переліку тварин в окрузі, в якому їх запросили; Використовується також як платформа для продажу тварин
PawBoost	Додаток недоступний в Україні	Наявність веб-версії; Велика кількість вхідних заявок; Відображаються заявки лише в радіусі 5 км від поточного місця знаходження користувача	Активність у додатку та на веб-сайті дуже низька у порівнянні з Фейсбук сторінкою, де також публікують заявки
PetFinder	Доступний тільки для iOS	Велике різноманіття функцій та глобальна мапа з можливістю побачити	Низька активність під постами учасників; Користувачі з Android

Назва продукту	Доступність	Переваги	Недоліки
		місцезнаходження загубленої або знайденої тваринки; Багато позитивних відгуків та достатньо користувачів по всьому світі, що шукають тут тваринок; Декілька типів міток залежно від статусу оголошення;	не можуть мати доступ до пошуку тварин;
PawScout	Є безкоштовна версія додатку в PlayMarket	Можливість поділитися постом у соц. мережах; Наявність чату з користувачами; Можливість додати винагороду за знаходження; Мапа з локацією, де тварина була втрачена	Відгуки з фейсбук сторінки свідчать про: постійні проблеми з додаванням нової тварини; багато фейкових заявок з номерами онлайн-магазинів; дзвінки від шахраїв через те, що номер телефону доступний всім користувачам.

1.2 Вибір інструментів розробки

1.2.1 Вибір мови програмування

З точки зору Google Play Store, Kotlin є офіційною мовою програмування для розробки додатків для Android. Однак Java залишається мовою програмування загального призначення для Android та інших платформ. У таблиці 1.2 наведено результати аналізу двох мов програмування.

Таблиця 1.2 - Порівняння Java та Kotlin

Параметр	Java	Kotlin
Стислість коду	Не дуже стислий в порівнянні	Компактніший за Java на 30-40%
Підтримка платформ	Java до останнього часу не використовувалася при розробці програм під iOS.	Kotlin можна використовувати для написання нативних програм для Android та iOS. Kotlin Multiplatform Mobile (KMM) працює в Android та iOS.
Примітивні типи	Змінні примітивного типу не є об'єктами	Змінні примітивного типу є об'єктами
Функції вищого порядку та Lambdas	Функції вищого порядку реалізуються за допомогою Callables. Вирази Lambdas представлені Java 8.	Попередньо вбудовані функції
Функції розширення	Ці функції не доступні Java.	Kotlin дозволяє розробникам розширювати клас новими функціями за допомогою функції розширення.
Null-Safety	наявність NullPointerException	всі типи за замовчуванням не мають значення NULL
Розумне приведення типів	Перш ніж об'єкт може бути наведений у Java, обов'язково потрібно перевірити тип. Це також правильно в сценаріях, де явно необхідно наводити об'єкт.	Kotlin має функцію розумного приведення, яка автоматично обробляє такі надлишкові приведення.
Підтримка конструкторів	Класи Java не можуть мати один або кілька вторинних конструкторів, крім первинного конструктора.	Класи Kotlin можуть мати один або кілька вторинних конструкторів, крім первинного конструктора.

Обидві мови програмування мають свої переваги та недоліки. І Java, і Kotlin компілюються в байткод, що означає, що можна використовувати Java-код у Kotlin і навпаки. Оскільки однією з ключових переваг Kotlin перед Java є те, що він був створений з огляду на необхідність підвищення продуктивності розробників, яке пов'язане з лаконічністю самого коду, у тому числі з інтуїтивно зрозумілим синтаксисом і загальним чистим дизайном мови, то написання нового коду на Kotlin займе менше часу. Тому саме з огляду на те, якою мовою легше користуватися у процесі розробки, було й обрано Kotlin.

1.2.2 Вибір середовища розробки

Середовище розробки повинно бути адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. Незважаючи на те, що є велика кількість IDE для розробки додатків Android і Android Studio вважається абсолютним лідером, інші середовища також активно використовуються розробниками. У таблиці 1.3 наведено результати аналізу двох середовищ розробки.

Таблиця 1.3 - Порівняння Android Studio та Visual Studio

Параметр	Android Studio	Visual Studio
Інтерфейс	Зручний інтерфейс і так як він базується на IntelliJ IDEA, що спрощує процес освоєння для розробників, які знайомі з продуктами JetBrains.	Деякі налаштування та функції іноді може бути важко знайти. Інтерфейс не завжди інтуїтивно зрозумілий.
Функціональність	Розроблено спеціально для технологій Android, тому включає тільки потрібні для розробки інструменти.	Достатньо універсальний інструмент, тому часто можна зустріти непотрібні опції та інструментами для технологій Android.
Системні вимоги	Достатньо важкий інструмент, тому потребує	Відрізняється меншими системними вимогами та більш

Параметр	Android Studio	Visual Studio
	багато оперативної пам'яті та вимагає наявності потужного процесора. У протилежному випадку виникають проблеми, що пов'язані з продуктивністю.	швидкою роботою.
Дизайнер макетів	Зручний у використанні	Відсутній
Доступність	Повністю безкоштовна IDE.	Безкоштовний для особистого використання, але дуже дорогий для підприємств.

Після аналізу, в якості середі розробки була вибрана IDE Android Studio. Причиною вибору є досить зручний інтерфейс, легкість написання коду та те, що вона спеціально розроблена для створення додатків Android.

1.2.3 Вибір системи управління базами даних

Firebase – платформа для розробки програмних додатків, яку розробила компанія Firebase. Firebase Realtime – це сервіс, який надає БД, в основі якої лежить хмарна технологія. Дані зберігаються у форматі JSON і синхронізуються в реальному часі. Використовуючи SDK Firebase для розробки під мобільну ОС, то розробники отримують автоматичне оновлення з новими даними.

Основні функції Firebase:

- 1) синхронізація даних в режимі реального часу;
- 2) швидкий доступ до документів і колекцій, що зберігаються в БД;
- 3) простота у використанні, що є пріоритетом для всіх продуктів Firebase;

При цьому підході продуктивність запитів не залежить від обсягу даних, і результат роботи з базою з 100 і 100 мільйонів документів буде однаково ефективний.

MySQL – реляційна система управління реляційними базами даних, є СУБД з відкритим вихідним кодом. Метою створення MySQL було підвищити швидкість обробки великих об’ємів даних. Вона підтримується багатьма відомими мовами програмування. MySQL – це найбільш поширена серверна СУБД.

Основні функції MySQL:

- крос-платформне використання та підтримка;
- відповідність стандартам SQL;
- використання тригерів;
- використання курсорів для полегшення обробки запитів вилучення,
- наявність статистики про роботу сервера та продуктивність запитів;

Недоліками MySQL є менша надійність транзакцій через те, що при створенні СУБД їх не було розроблено, більш повільний процес розробки в порівнянні з сучасними БД, обмеження функціоналу, який не вистачає для розробки сучасних програмних додатків, невідповідність стандартам SQL в певних принципових питаннях. MySQL не може виконувати багато процесів одночасно, що ставить її ефективність під сумнів.

Таким чином, проаналізувавши інформацію про сервіс Firebase Realtime та СУБД MySQL та порівнявши їх можливості, було виділено наступні переваги Firebase над MySQL:

- 1) швидкодія роботи з даними. Розробник та користувач може дуже швидко доступитись до даних;
- 2) простота в роботі з системою. Не потрібно створювати велику кількість файлів, треба лише зареєструвати програмний додаток на офіційному сайті Firebase;
- 3) активна підтримка від Google та безкоштовний зворотній зв’язок від спеціалістів по розробці.

Враховуючи переваги та недоліки кожного способу зберігання даних в цій роботі буде доцільно використовувати сервіс для зберігання даних Firebase Realtime.

2 ПРОЕКТУВАННЯ

2.1 Функціональні вимоги

З метою упорядкування функцій ПП була створена класифікація функціональних вимог, виявлених в сценаріях використання прецедентів.

В таблиці 2.1 перераховано всі функції, які було визначено в рамках розробки програмного продукту. Кожна функція в ієрархії позначено унікальним ієрархічним ідентифікатором FRi.

Таблиця 2.1 – Функціональні вимоги до розробки

Ідентифікатор функції(назва)	Назва функції
FR1	Авторизація користувача
FR2	Створення заявки про зниклу тварину
FR3	Редагування заявки про зниклу тварину
FR4	Видалення заявки про зниклу тварину
FR5	Перегляд заявки про зниклу тварину
FR6	Перегляд загального списку заявок залежно від вхідних параметрів
FR7	Додавання коментаря-репорту до заявки

2.2 Діаграма використання (Use cases diagram) програми та її опис

Діаграма використання — це графічне зображення можливої взаємодії користувача з системою. Діаграма використання показує різні варіанти використання та різні типи користувачів, яких система має і часто супроводжуватиметься іншими типами діаграм.

На рисунку 2.1 зображено діаграму прецедентів для застосунку PawPatrol.

Визначено акторів: Авторизований користувач, FireBase, Google Map API.
Прецеденти: Створення оголошення, Редагування оголошення, Видалення оголошення, Коментування оголошень, Авторизація.

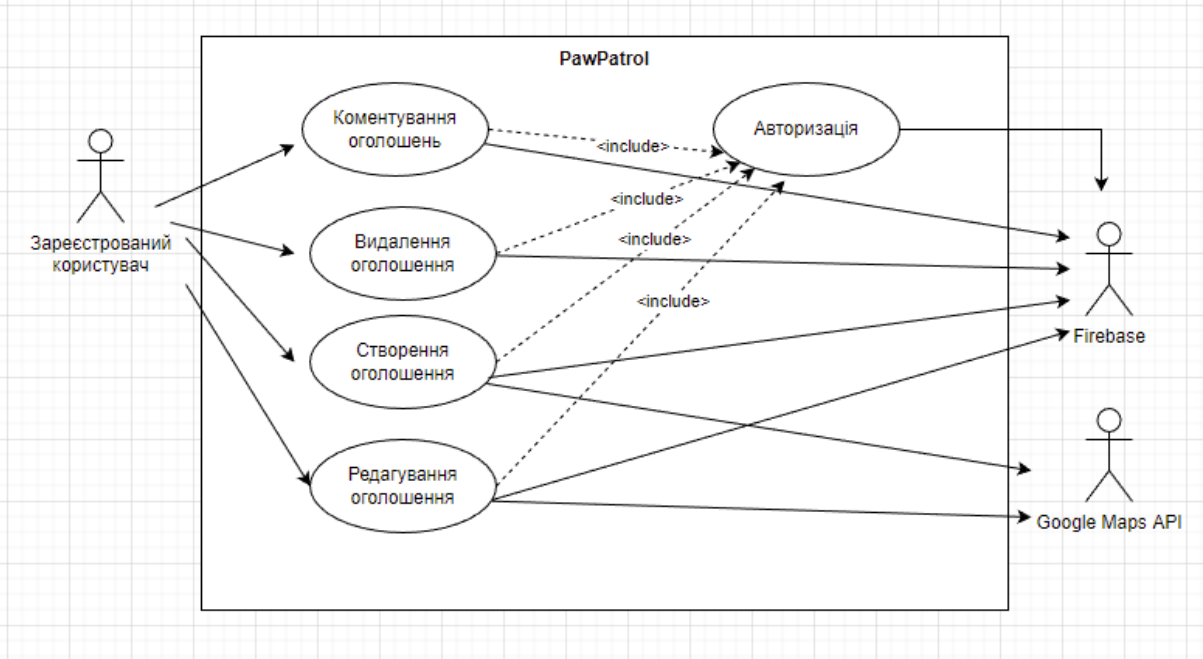


Рисунок 2.1 - Use Case UML-діаграма

Встановлені зв’язки щодо рисунку 2.1 з діаграмою використання наведено у таблиці 2.2.

Таблиця 2.2– Встановлені зв’язки у use case діаграмі

Актор	Тип зв’язку	Прецедент
Авторизований користувач	асоціація	Редагування оголошення
Авторизований користувач	асоціація	Створення оголошення
Авторизований користувач	асоціація	Видалення оголошення
Авторизований користувач	асоціація	Коментування оголошень
Прецедент	Тип зв’язку	Актор
Створення оголошення	асоціація	Firebase

Актор	Тип зв'язку	Прецедент
Редагування оголошення	асоціація	Firebase
Видалення оголошення	асоціація	Firebase
Коментування оголошення	асоціація	Firebase
Авторизація	асоціація	Firebase
Редагування оголошення	асоціація	Google Maps API
Створення оголошення	асоціація	Google Maps API
Прецедент	Тип зв'язку	Прецедент
Редагування оголошень	включення	Авторизація
Створення оголошень	включення	Авторизація
Коментування оголошень	включення	Авторизація
Видалення оголошень	включення	Авторизація

Опис даної діаграми наведено нижче у форматі user story для більш чіткого та детального розуміння зв'язків між користувачами та додатком.

Визначено тип користувача - авторизований користувач (далі просто користувач):

Як користувач, я хочу мати змогу переглядати список усіх зниклих тваринок, щоб швидко оглянути повну інформацію.

Як користувач, я хочу мати змогу переглядати оголошення конкретної тваринки, щоб отримати більш детальну інформацію про зовнішній вигляд та місце, де вона була загублена.

Як користувач, я хочу мати змогу створювати оголошення, щоб отримати допомогу у пошуку зниклої тваринки.

Як користувач, я хочу мати змогу залишати коментарі під заявками інших користувачів, щоб сприяти швидкому знаходженню тваринки.

Як користувач, я хочу мати змогу переглядати коментарі інших користувачів під своїми оголошеннями, щоб знати, де шукати тваринку.

Як користувач, я хочу мати змогу редагувати своє оголошення, щоб надати більш актуальну інформацію.

Як користувач, я хочу мати змогу видаляти свої оголошення, щоб не вводити в оману інших користувачів, якщо моя тваринка вже знайдена.

2.3 Ідентифікація архетипу додатка

Створений додаток належить до типу мобільних додатків. Мобільний додаток, як правило, структурується як багат шаровий додаток, що включає шар інтерфейсу користувача (представлення), бізнес-шар і шар доступу до даних.

На рисунку 2.2 можна побачити, що додаток було створено для операційної системи Android на мові програмування Kotlin. У якості бази даних було використано зберігання та синхронізацію даних у реальному часі за допомогою Firebase Realtime database. Firebase, що підтримується Google, бере на себе складність роботи з базами даних у реальному часі, аутентифікації користувачів та роботи з робочими процесами офлайн-синхронізації. Firebase не потребує налаштування конфігурації сервера інфраструктури, оскільки платформа піклується про хостинг і, у свою чергу, надає SDK.

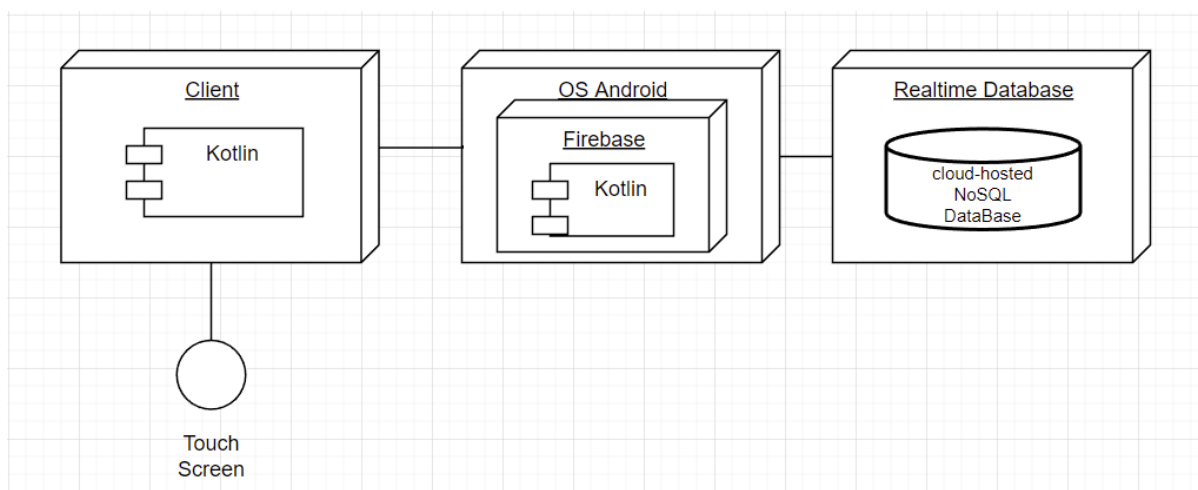


Рисунок 2.2 – Deploy diagram

2.4 Логічне представлення (Logical View)

Щоб забезпечити основу для розуміння структури та організації проектування системи, у робочому процесі Analysis & Design використовується архітектурний вигляд, який називається логічним представленням. Існує лише один логічний погляд на систему, який ілюструє ключові реалізації варіантів використання, підсистеми, пакети та класи, які охоплюють архітектурно значущу поведінку.

На рисунку 2.3 зображено логічне представлення додатку PawPatrol

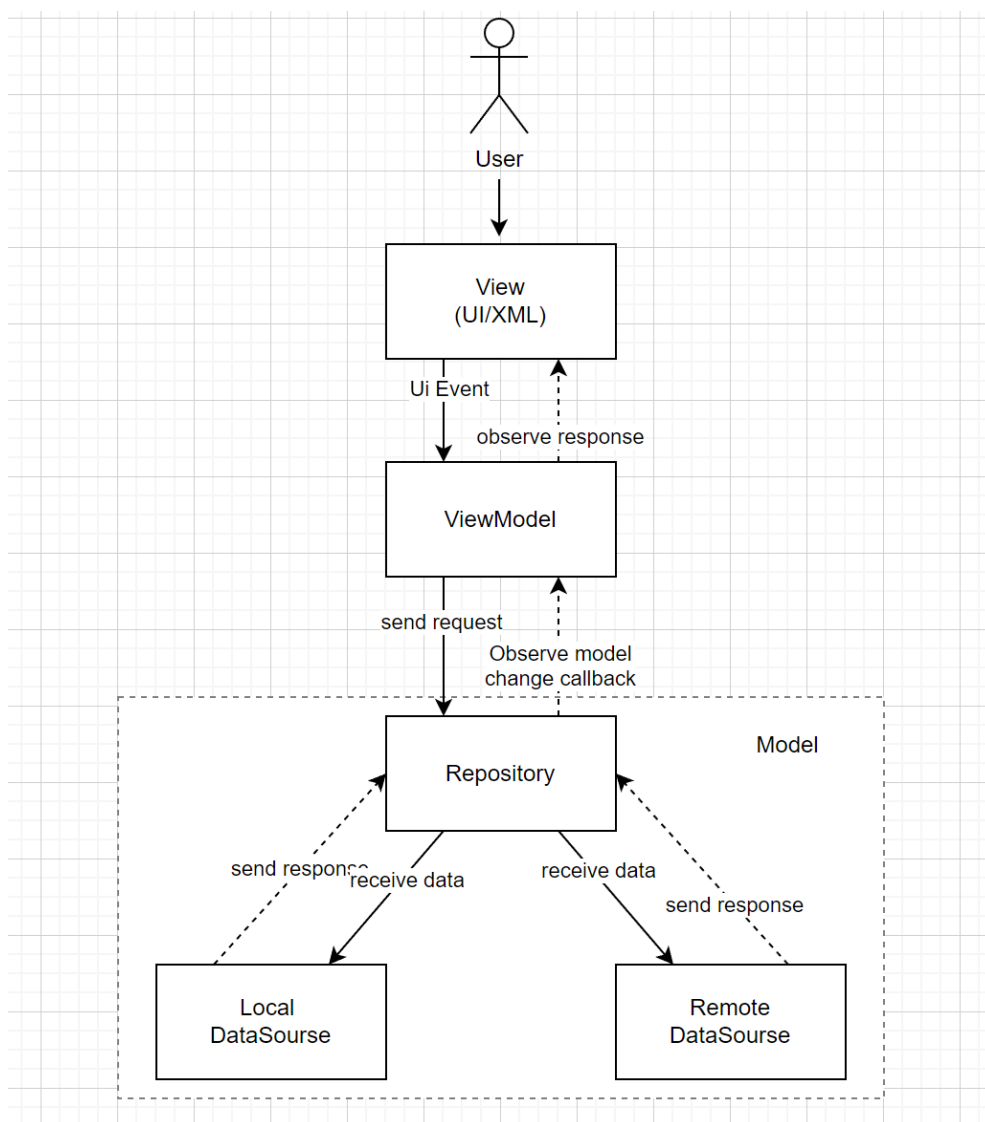


Рисунок 2.3 - Логічне представлення (Logical View)

Користувач використовує графічний інтерфейс (UI) для взаємодії із додатком. Усі запити користувача обробляються додатком на рівні ViewModel.

Оскільки використовується Firebase Realtime database, то є можливість автономного режиму роботи додатка. Cloud Firestore підтримує збереження даних в автономному режимі. Ця функція кешує копію даних Cloud Firestore, які програма активно використовує, тож програма може отримати доступ до даних, коли пристрій у автономному режимі. Можна записувати, переглядати та запитувати кешовані дані. Коли пристрій повертається в режим онлайн, Cloud Firestore синхронізує будь-які локальні зміни, внесені програмою, із серверною частиною Cloud Firestore. Щоб використовувати збереження в автономному режимі, не потрібно вносити жодних змін до коду, який використовується для доступу до даних Cloud Firestore. Якщо ввімкнено автономне збереження, клієнтська бібліотека Cloud Firestore автоматично керує доступом до даних в режимі онлайн та офлайн і синхронізує локальні дані, коли пристрій знову в мережі.

2.5 Уявлення процесів (Process View)

Процесний погляд на роботу визначається як розуміння того, що роботу можна розглядати як «процес», який має входи, кроки та вихід(и) та взаємодіє з іншими процесами в організації. Це загальне усвідомлення тактик і методологій, які використовуються «організованою групою суміжних видів діяльності, які працюють разом, щоб перетворити один або кілька видів вхідних даних у результати, які є цінними для клієнта».

Це визначення передає кілька ключових ідей:

Процес – це група дій.

Діяльність, що становить процес, не є випадковою чи випадковою; вони пов'язані та організовані.

Всі дії в процесі повинні працювати разом для досягнення спільної мети.

Існують процеси для досягнення результатів, які цінують внутрішні та зовнішні клієнти.

2.5.1 Діаграма діяльності

Діаграма діяльності є важливою діаграмою в UML для опису динамічних аспектів системи. Діаграма діяльності - це в основному блок-схема, яка представляє перехід від однієї діяльності до іншої. Діяльність можна описати як операцію системи.

Основні цілі діаграм діяльності подібні до інших чотирьох діаграм. Він фіксує динамічну поведінку системи.

На рисунку 2.4 представлено діаграму діяльності для авторизації користувача. Порядок виконання кроків з боку користувача та самої системи можна описати наступним чином:

Користувач заходить в додаток. Йому виводиться відповідна форма авторизації, поля якої користувач заповнює. Далі введені дані перевіряються на відповідність - відбувається процес валідації користувача:

Якщо користувач з введеними даними існує (вже зареєстрований), то йому надається доступ.

Якщо дані, введені користувачем у формі авторизації, не відповідають вимогам, то система повідомляє користувача про те, що користувача з такими параметрами не існує.

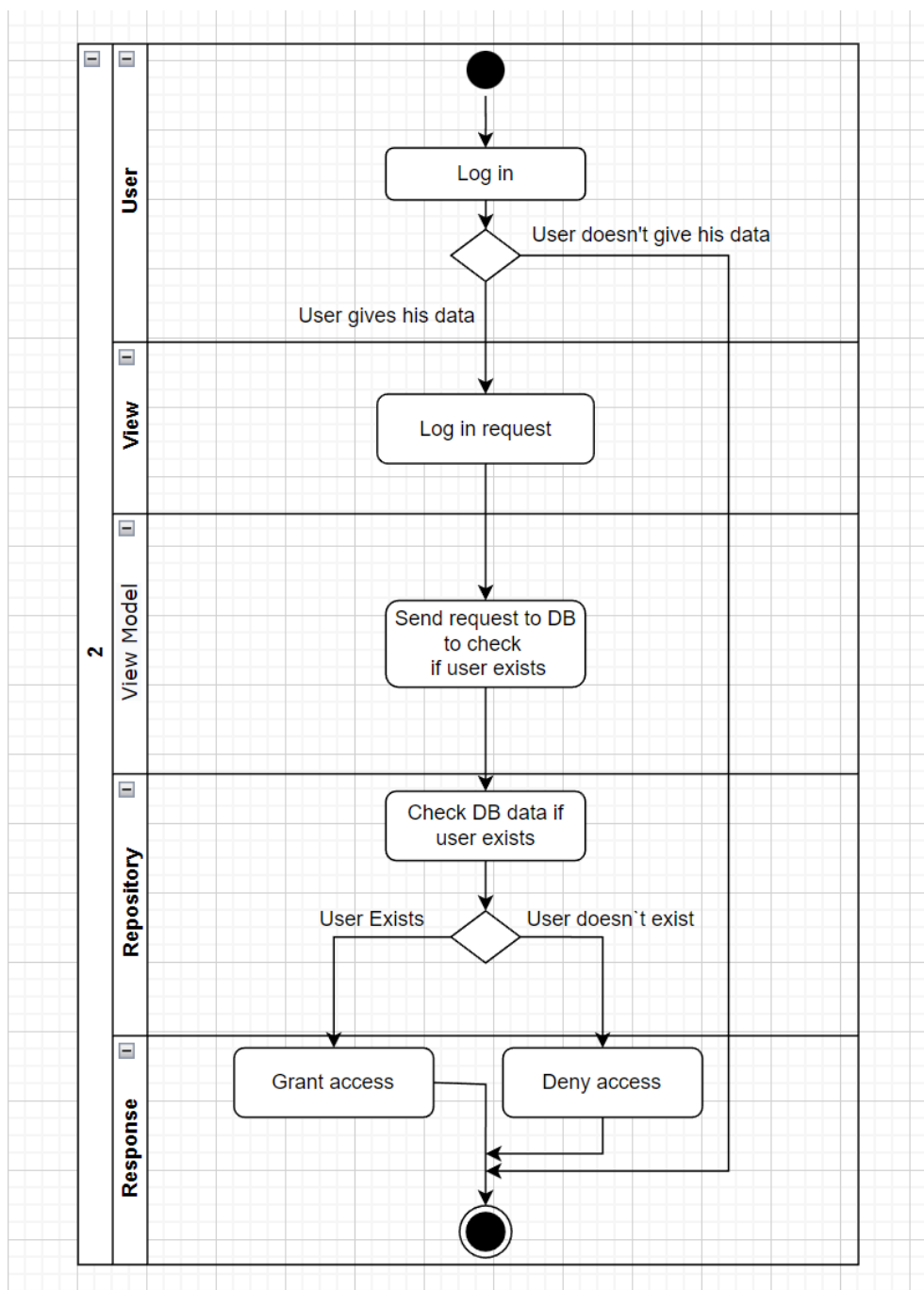


Рисунок 2.4 - Діаграма діяльності для входу зареєстрованого користувача

На рисунку 2.5 представлено діаграму діяльності для створення нового оголошення про зниклу тваринку. Порядок виконання кроків з боку користувача та самої системи можна описати наступним чином:

Користувач натискає відповідну іконку на головному екрані додатку. Користувачеві відкривається форма для заповнення: він має обов'язково вказати ім'я зниклого домашнього улюбленця, надати короткий опис тваринки, на карті

обрати локацію, де тваринка загубилася, натиснути на кнопку “Upload Image” для додавання фотографії тваринки. Після цього натискається кнопка “ Create Note”. Усі введені користувачем у формі дані перевіряються на відповідність вимогам, зокрема: поле з іменем, описом не має бути порожнім, має бути додана локація та фото.

Якщо дані відповідають вимогам, то оголошення створюється, користувачеві видається відповідне повідомлення: “Note created”.

Якщо дані не відповідають вимогам, то додаток повідомляє про це користувача шляхом підсвічення відповідних полей червоним кольором з відповідною вказівкою заповнити їх або виведенням вказівки про додавання фото чи локації.

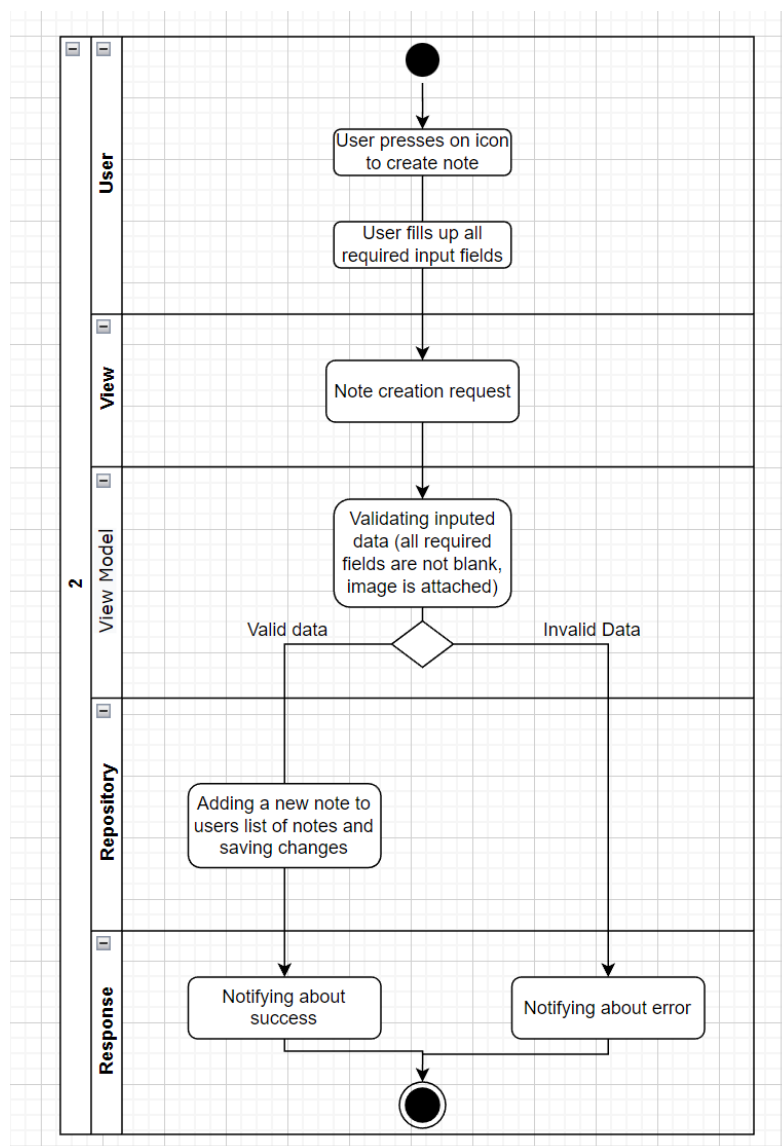


Рисунок 2.5 - Діаграма діяльності для створення нового оголошення

На рисунку 2.6 представлено діаграму діяльності для видалення існуючого оголошення про зниклу тваринку. Порядок виконання кроків з боку користувача та самої системи можна описати наступним чином:

користувач заходить у власний профіль, де знаходиться перелік “Your missing pets” і натискає на оголошення, яке він хоче видалити. Всередині самого оголошення, що відкривається, користувач натискає на кнопку “Delete Note”. Після цього це оголошення видаляється і зникає зі списку всіх оголошень. Якщо це було єдине існуюче оголошення, то список стає порожнім.

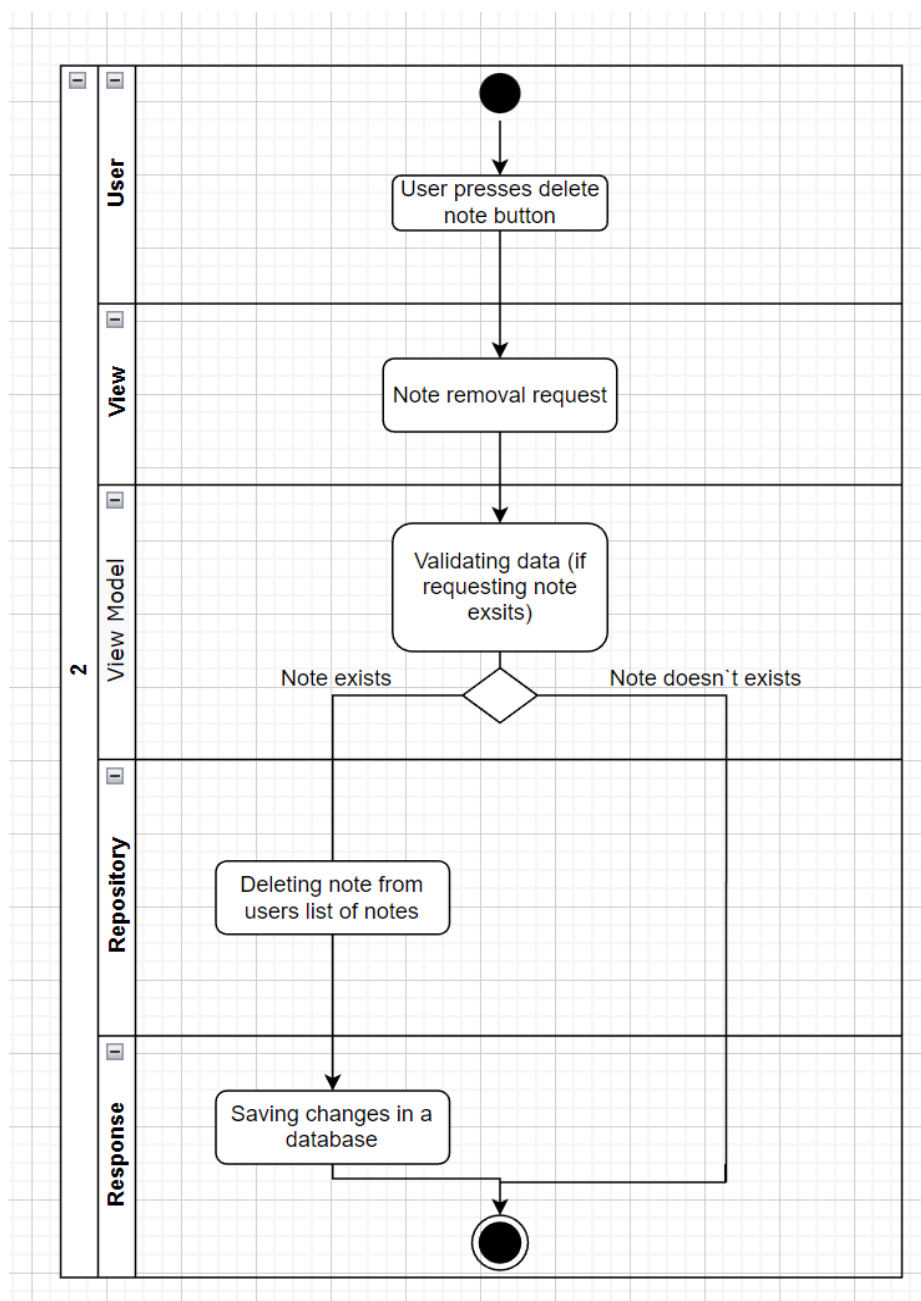


Рисунок 2.6 - Діаграма діяльності для видалення існуючого оголошення

На рисунку 2.7 представлено діаграму діяльності для видалення існуючого оголошення про зниклу тваринку. Порядок виконання кроків з боку користувача та самої системи можна описати наступним чином:

після авторизації користувач потрапляє на головний екран додатку, що містить список всіх опублікованих оголошень. Цей перелік відображується за двома сценаріями:

якщо на пристрої користувача увімкнена геолокація, то список сортується відповідно до його місця знаходження - від найближчого до найвіддаленішого.

якщо на пристрої користувача вимкнута геолокація, то список сортується відповідно до часу створення оголошення - від найновішого до найстарішого.

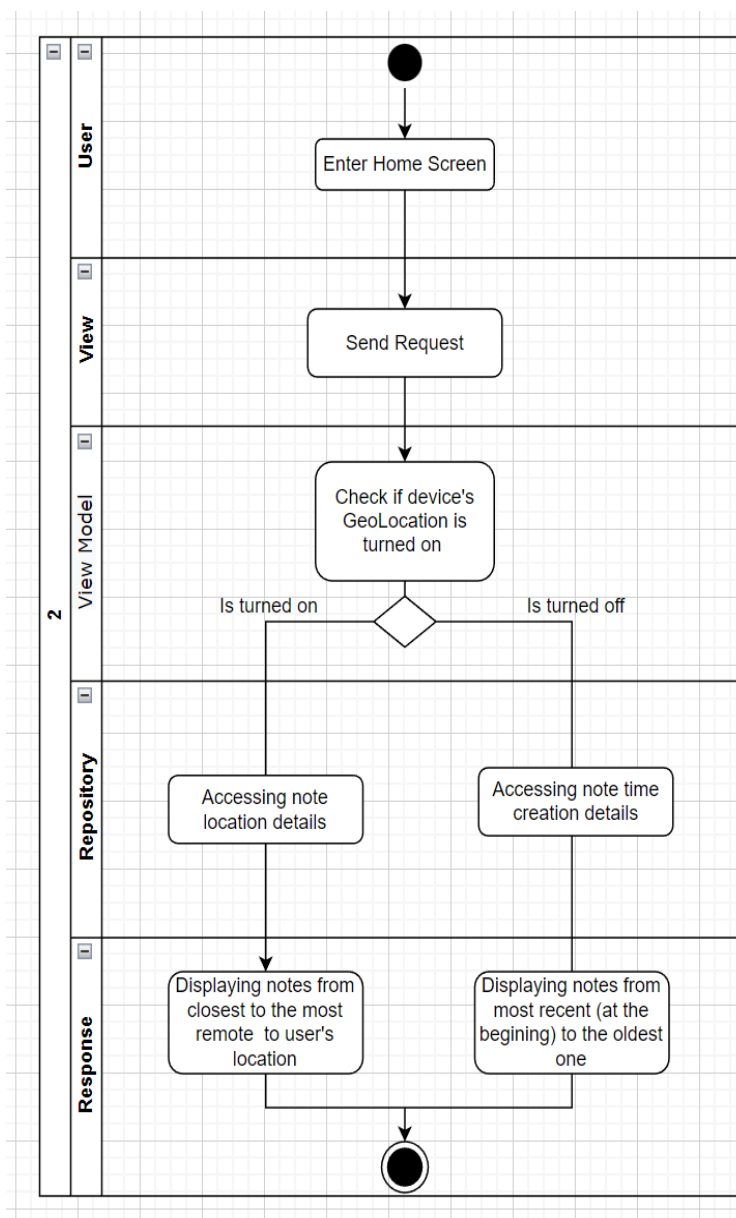


Рисунок 2.7 - Діаграма діяльності для перегляду всіх існуючих оголошень

2.5.2 Діаграма послідовності

Діаграми послідовностей є популярним рішенням для динамічного моделювання в UML, оскільки вони спеціально зосереджені на лініях життя, або процесах та об'єктах, які живуть одночасно, і повідомленнях, якими вони обмінюються для виконання функції до завершення лінії життя.

На рисунку 2.8 представлено діаграму послідовності для авторизації користувача у системі.

Першим кроком від клієнта відправляється заповнена користувачем форма авторизації. Запит з даними форми надсилається далі для валідації користувача.

Після цього відбувається валідація отриманих даних (перевіряються, чи існує користувач з введеним логіном (адресою ел. пошти) та паролем у системі).

Якщо особа користувача підтверджується, то відповідне повідомлення повертається системі, яка надає користувачеві доступ.

Якщо особа користувача не підтверджується, то відповідне повідомлення про помилку авторизації повертається системі, яка в свою чергу не надає користувачеві доступ.

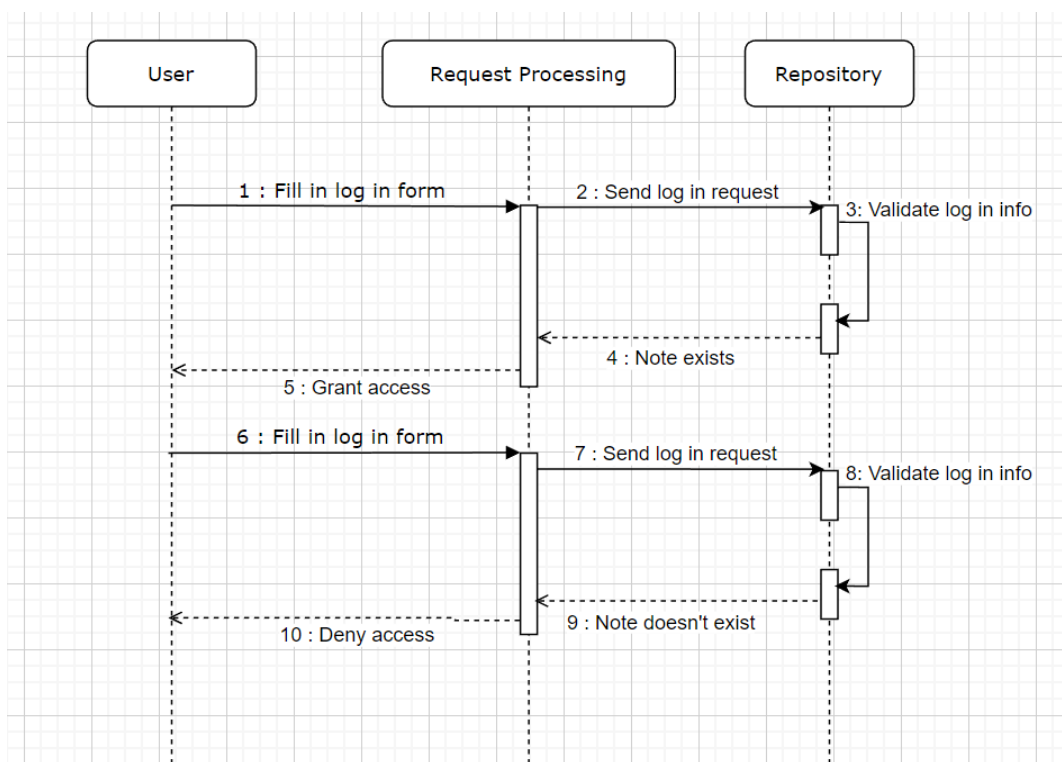


Рисунок 2.8 - Діаграма послідовності для авторизації користувача у системі

На рисунку 2.9 представлено діаграму послідовності для виведення списку всіх оголошень.

Користувач заходить на головний екран додатку. Відповідний запит на формування списку усіх оголошень про зниклих тварин надсилається системі.

Якщо на пристрої користувача була увімкнена геолокація, то з бази даних отримується інформація про координати (довготу та широту) локацій, де тваринка була загублена, з оголошень. Далі дані повертаються системі, яка на основі поточного місцезнаходження користувача створює список зниклих тварин, сортуючи його від найближчої локації з оголошення до найвіддаленішої.

Якщо на пристрої користувача не була увімкнена геолокація, то з бази даних отримується інформація про час та дату створення оголошень. Далі дані повертаються системі, яка на основі поточної дати та часу створює список зниклих тварин, сортуючи його від найновіше створеного оголошення до найстарішого.

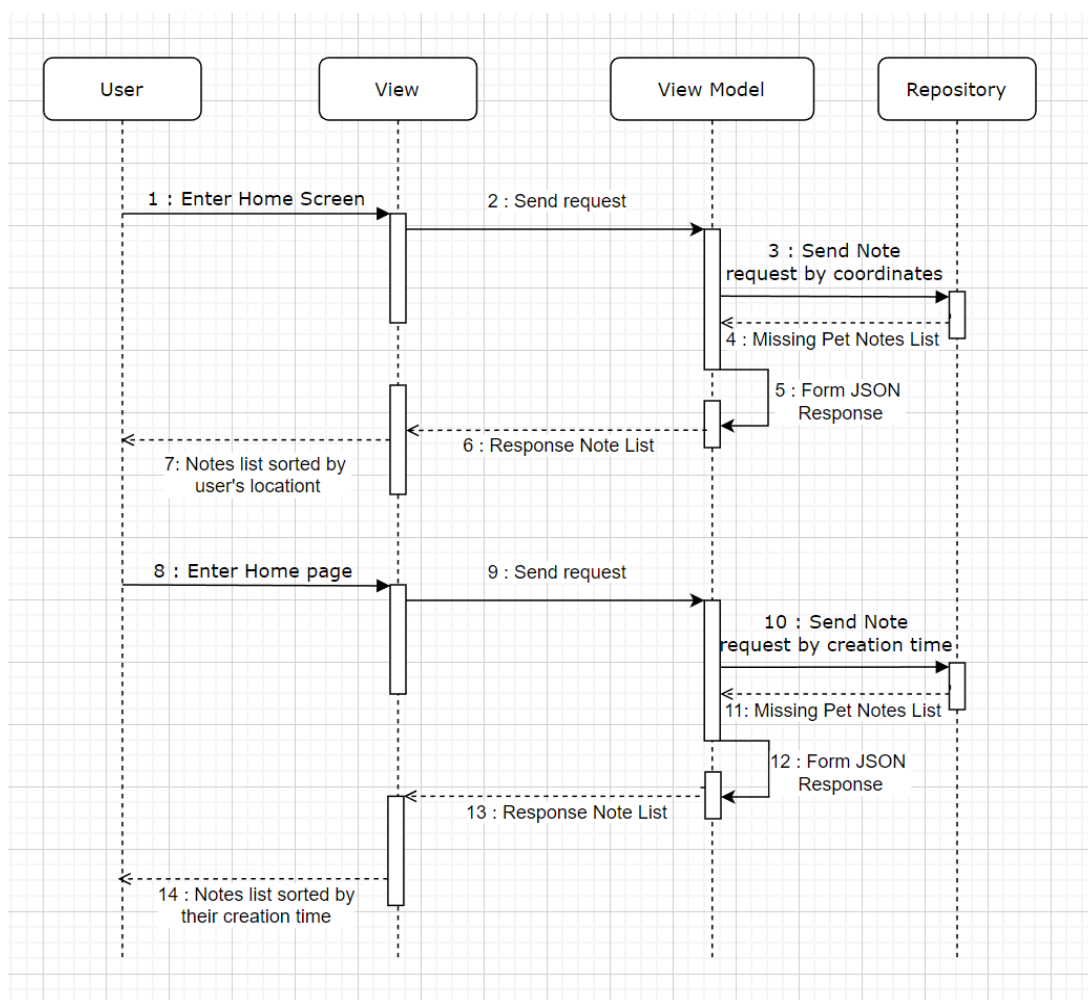


Рисунок 2.9 - Діаграма послідовності для виведення списку всіх оголошень

Діаграма послідовності видалення персонального коктейлю представлена на рисунку 2.10.

Першим кроком надсилається запит на видалення користувацького оголошення. Далі надсилається запит на рівень бази даних для перевірки чи існує оголошення, щодо якого виконується запит.

У разі існування оголошення повертається відповідне повідомлення. Після отримання підтвердження існування оголошення, на рівень бази даних надсилається запит видалення. У якості відповіді повертається повідомлення про успішність видалення, і відбуваються відповідні зміни на рівні клієнту.

Після перевірки на рівні баз даних, у разі відсутності шуканого оголошення в базі даних відправляється відповідне повідомлення, і воно вже повертає повідомлення про відсутність оголошення на рівень клієнта.

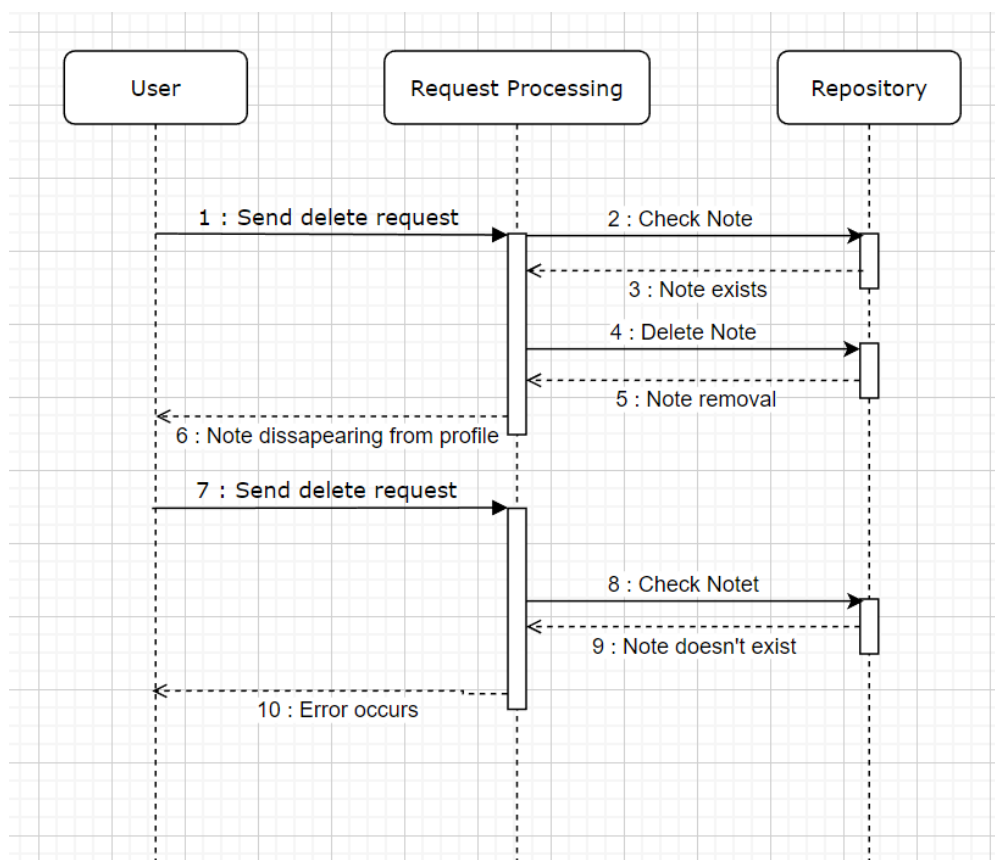


Рисунок 2.10 - Діаграма послідовності для видалення оголошення користувача

Діаграми активностей та діяльностей для інших сценаріїв роботи додатку будуються аналогічним способом.

2.6 Подання даних (Data View)

Під час розробки додатку важливим етапом є проектування бази даних, яка буде забезпечувати зберігання даних і можливість здійснювати операції над ними. Для реалізації бази даних було обрано Realtime Database та Storage. Як вже згадувалось вище, Firebase є платформою, яка забезпечує рішеннями для швидкої розробки програмного продукту. База даних Firebase Realtime — це хмарна база даних. Дані зберігаються у форматі JSON і синхронізуються в режимі реального часу з кожним підключеним клієнтом. Замість типових запитів HTTP база даних Firebase Realtime використовує синхронізацію даних — щоразу, коли дані змінюються, будь-який підключений пристрій отримує це оновлення протягом мілісекунд. Усі дані бази даних Firebase Realtime Database зберігаються як об'єкти JSON. Ви можете уявити базу даних як хмарне дерево JSON. На відміну від бази даних SQL, тут немає таблиць або записів. Коли дані додаються до дерева JSON, воно стає вузлом в існуючій структурі JSON із пов'язаним ключем.

Структура зберігання даних у дереві JSON наведена на рисунках 2.11 та 2.12. На рис. 2.11 зображено дерево оголошень: кожне оголошення додається до основного вузла `missing_pets` шляхом його приєднання спочатку до `id` користувача, що його створив. Кожне оголошення має власне унікальне `id`, яке є вузлом для інформації, що знаходиться всередині оголошення. Як показано на рис. 2.10, вміст оголошення складається з: дати створення, опису, унікального номеру зображення, що було завантажено до оголошення, набору географічних координат позиції втрати тваринки та ім'я тваринки (типи даних вказані поруч).

На рис. 2.12 зображено дерево коментарів до оголошень: кожний коментар(репорт) додається до основного вузла `reports` шляхом його приєднання спочатку до `id` оголошення, під яким він був доданий. Кожний коментар має також власне унікальне `id`, яке є вузлом для інформації, що знаходиться всередині коментаря.

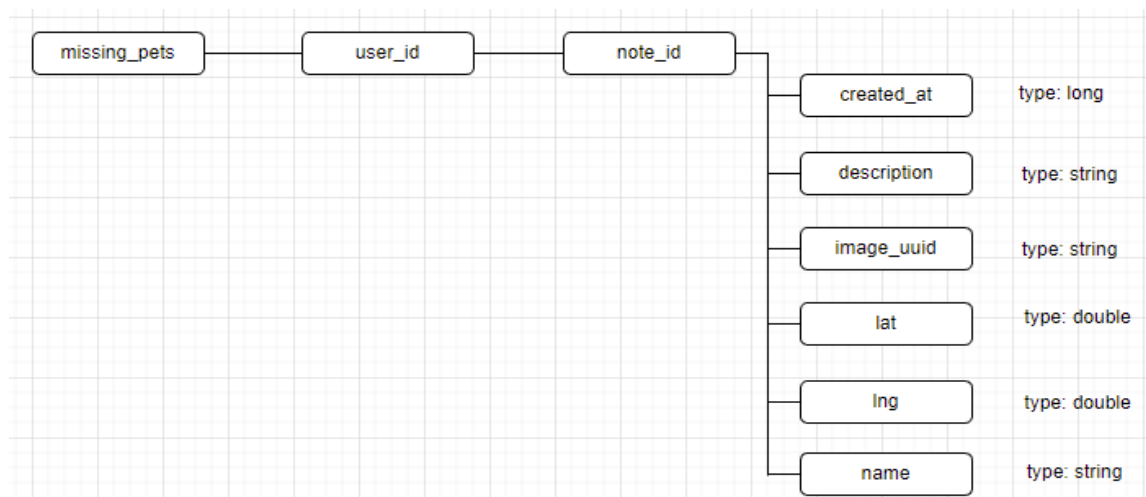


Рисунок 2.11 - Структура дерева оголошень

Як показано на рис. 2.12, вміст коментаря складається з: дати створення, опису, зображень, що були додані до коментаря (типи даних вказані поруч, optional біля `attached_image` вказує на те, що додавання зображення для створення коментаря не є обов'язковим).

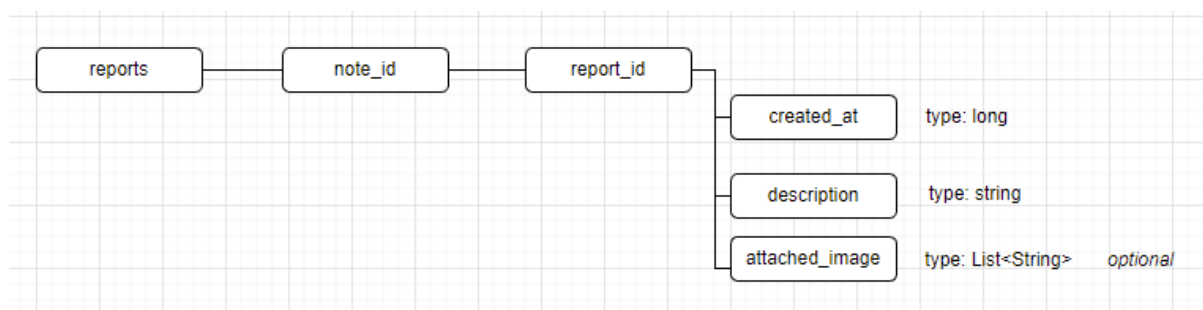


Рисунок 2.12 - Структура дерева коментарів до оголошень

Також окрім Firebase Realtime Database буде використовуватись Storage. Це вбудований компонент, який забезпечує зберігання користувацького контенту - в нашому випадку це фото тваринок. Використання цього компоненту є обов'язковим та одним з ключових для розроблюваного додатку, адже база даних повинна мати в собі інформацію про вигляд зниклих тваринок.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Структура проекту

На рисунку 3.1 наведена діаграма, що демонструє структуру розміщення файлів у папках проекту. Ліва частина діаграми демонструє структуру головної частини, що містить основні папки missing, notedetails, login, navigation, profile. Права частина демонструє структуру частини інтерфейсу користувача. Логіка розміщення файлів у папках наведена у таблиці 3.1

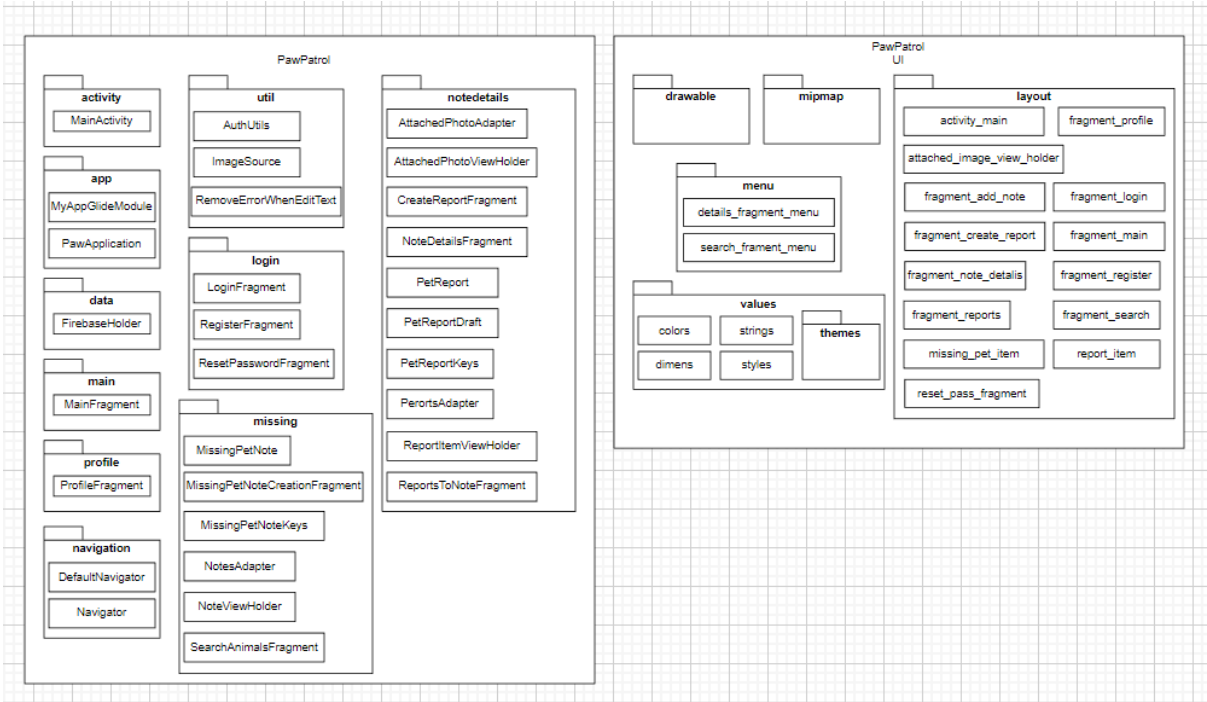


Рисунок 3.1 - Структура проекту

Таблиця 3.1 – Опис структури пректу

Назва папки	Призначення файлів
PawPatrol	
activity	Main activity містить клас MainActivity з якого починається виконання додатку
app	Містить файли з налаштуванням запуску та налаштуванням глобальної конфігурації параметрів Glide
data	Містить файл з екземпляром Firebase Database та посилання на Realtime Database та Storage

Назва папки		Призначення файлів
profile		містить файл з класом фрагменту профілю користувача
navigation		кореневий каталог
	DefaultNavigator	Клас, що реалізує інтерфейс Navigator з реалізацією його методів
	Navigator	Інтерфейс з методами для навігації
util		Містить класи та об'єкти з допоміжними функціями для авторизації та аутентифікації, роботи з джерелом зображення та редагування тексту
login		основні компоненти React
	RegisterFragment	Відповідає за реєстрацію
	LoginFragment	Відповідає за авторизацію
	ResetPasswordFragment	Відповідає за скидання паролю
missing		Містить файли, що відповідають за створення оголошення, відображення загального переліку оголошень та його сортування
notedetails		Містить файли для створення, додавання коментарів, їх прикріплення до оголошення та відображення для користувача
PatPatrol UI		
drawable		Містить усі іконки, що використовуються в додатку
layout		Містить .xml-файли з розміткою макетів екранів додатку
mipmap		Графічні файли, які використовуються для іконок програми під різні роздільні здатності екранів
values		Містить усі кольори, стилі, теми та розміри відображення картинок для додатку

3.2 Опис класів додатку

На рисунку 3.2 зображена UML-діаграма класів проекту (у додатку Б наведено рисунок у більшому розмірі). Клас DefaultNavigator реалізує інтерфейс Navigator з реалізацією його методів. У даному класі прописано уся логіка навігації додатком зі зміною візуальних складників, назви методів класу, наприклад, як

`navigateToLogin()`, `navigateToCreateAccount()`, `navigateToNoteDetails(authorId: String, noteId: String)` прямо відповідають за кінцеву точку навігації.

У `MainActivity` у методі `onAuthStateChanged()` в змінну `isLoggedIn` повертається поточний авторизований користувач, якщо він існує, то навігатор перенаправляє одразу на головний екран додатку, якщо він не існує, то навігатор відправляє до екрану авторизації.

`LoginFragment` відповідає за авторизацію користувача, саме тут відбувається перевірка на відповідність паролю та адреси ел. пошти вимогам (не порожність, кількість символів, відповідність шаблонам) за допомогою методів допоміжного об'єкта `AuthUtils`. Потім вже перевірка для авторизації відбувається на стороні `Firebase`. З цього класу можлива навігація до класу `ResetPasswordFragment`, де відбувається перевірка введеної адреси ел. пошти для скидання паролю знов за допомогою методу `isValidEmail()` з `AuthUtils`. У фрагменті для реєстрації користувача `RegisterFragment` відбуваються такі самі перевірки.

Оскільки для завантаження та кешування зображень була використана бібліотека `Glide`, то було створено клас `MyAppGlideModule` для успадкування `AppGlideModule` та глобальної конфігурації параметрів `Glide`.

У `FirebaseHolder` `getInstance()` отримує екземпляр бази даних. Оскільки БД організована навколо вузлів `JSON`, то потрібно ссилатися на дочірнього елемента батьківського проекту. Тому у `FirebaseHolder` визначаються назви вузлів `JSON`, а також використання `Storage` дозволяє отримувати доступ до даних через систему файлів/папок – для зберігання зображень було визначено назву для `imageStorageRef` як папку `images`.

У `MainFragment` відбувається навігація до `SearchAnimalsFragment`, що відповідає за відображення повного списку оголошень. Головна логіка відображення списку тваринок у `SearchAnimalsFragment` прописана у методі `onDataChanged()`, в якому в залежності від того, чи увімкнена геолокація, відбувається сортування оголошень відповідно до неї, або часу їх створення.

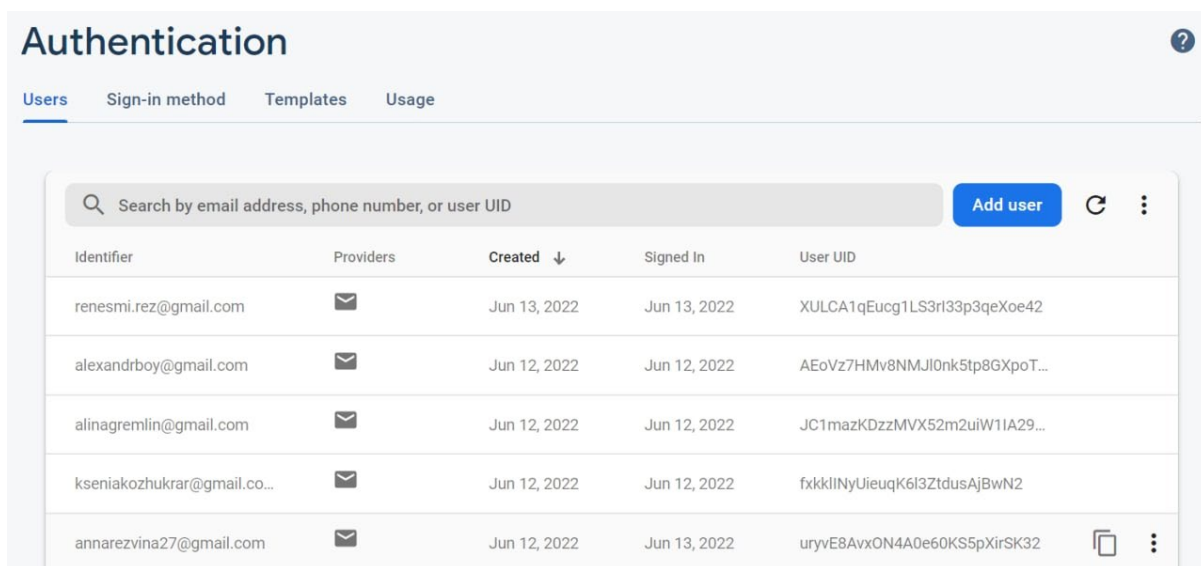
Клас даних `MissingPetNote` містить метод `fromJson()`, що відповідає за перетворення рядку `JSON` на об'єкт у `Kotlin` за допомогою інтерфейсу `Parcelable`.

для його видалення та перегляду коментарів; якщо користувач не є автором, то видимою є лише кнопка для додавання коментаря.

3.3 Збереження даних у Firebase

Firebase надає бекенд, простий у використанні SDK і готові бібліотеки інтерфейсу користувача для реалізації аутентифікації користувачів у додатку. Було використано автентифікацію за допомогою email та пароля (рис. 3.3).

Під час реєстрації користувачів у програмі інформація про них буде відображатися на вкладці «Authentication», де можна керувати користувачами, наприклад, додати користувача до бази програми, а також вимкнути або видалити користувача.



The screenshot shows the 'Authentication' tab in the Firebase console, specifically the 'Users' sub-tab. It displays a table of registered users with columns for Identifier, Providers, Created, Signed In, and User UID. There are five users listed, all using email as a provider. The table includes search and action buttons like 'Add user' and 'Refresh'.

Identifier	Providers	Created ↓	Signed In	User UID
renesmi.rez@gmail.com	✉	Jun 13, 2022	Jun 13, 2022	XULCA1qEucg1LS3rI33p3qeXoe42
alexandrboy@gmail.com	✉	Jun 12, 2022	Jun 12, 2022	AEoVz7HMv8NMJI0nk5tp8GXpoT...
alinagremlin@gmail.com	✉	Jun 12, 2022	Jun 12, 2022	JC1mazKDzzMVX52m2uiW1IA29...
kseniakozhukrar@gmail.co...	✉	Jun 12, 2022	Jun 12, 2022	fxkkllNyUieuqK6i3ZtdusAjBwN2
annarezvina27@gmail.com	✉	Jun 12, 2022	Jun 13, 2022	uryvE8AvxON4A0e60KS5pXirSK32

Рисунок 3.3 - Користувачі системи з адресами їх ел.пошти та id

На рисунку 3.4 зображено структуру дерева JSON як приклад збереження даних в Firebase Realtime Database.



Рисунок 3.4. - Приклад збереження даних в Firebase Realtime Database

На рисунку 3.5 зображено структуру вузлів `missing_pets` та `reports` з відповідними даними, введенними користувачами.

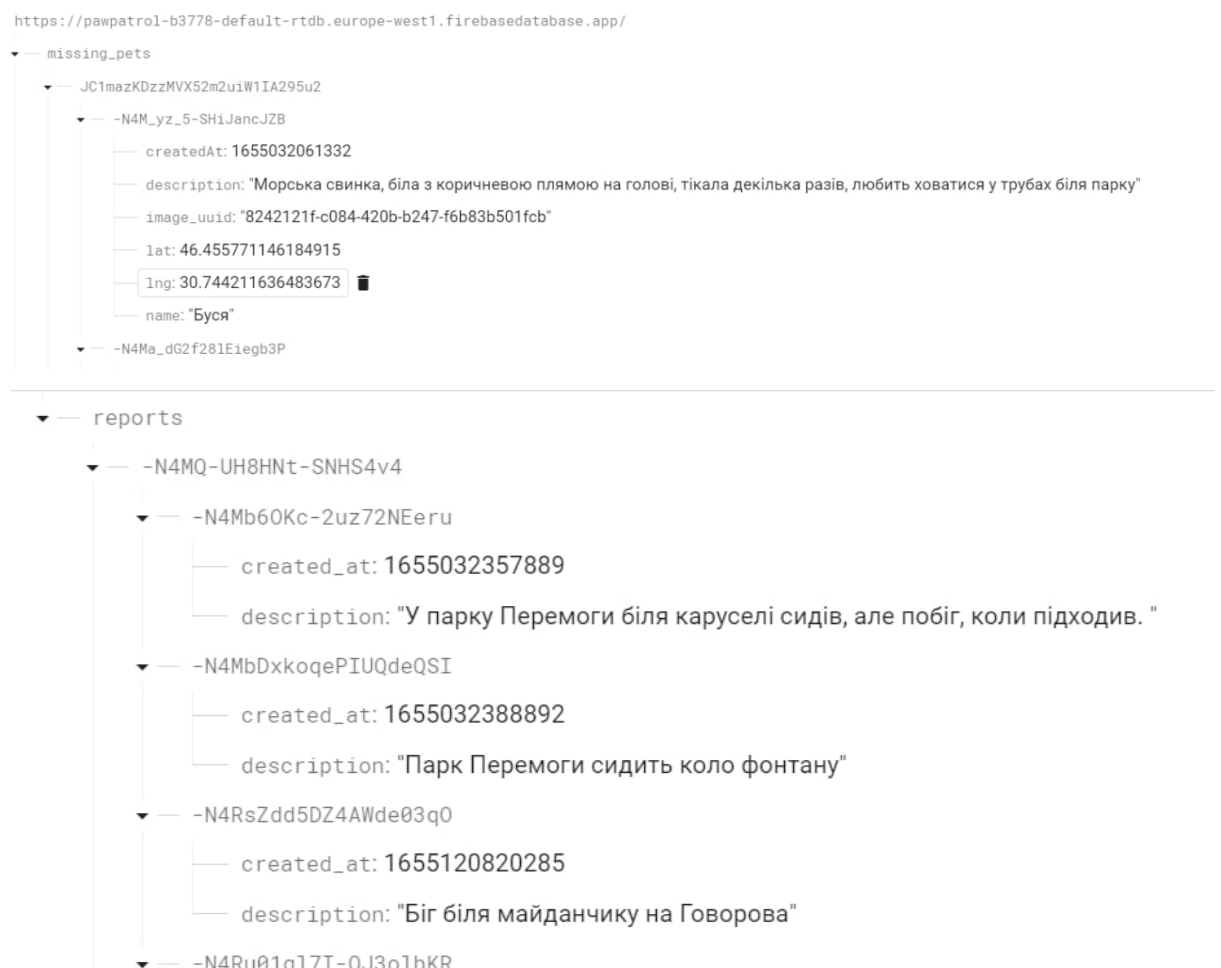


Рисунок 3.5 - Вигляд відповідних вузлів `missing_pets` та `reports`, з їх нащадками

Як зазначалося у пункті 3.2 у Storage міститься одна папка з назвою images (рис. 3.6) для зберігання зображень, що завантажують користувачі (рис. 3.7).

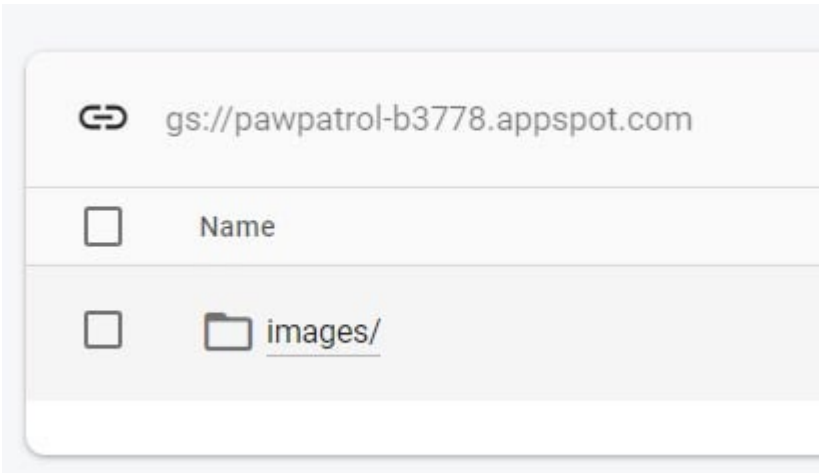














Рисунок 3.6 - Вміст Storage

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 03e576a6-6825-4f63-98c1-a6515f89c022	82.52 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 1fe37234-6564-4766-9388-d16d2f7a5634	174.08 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 2d0d1d4b-0eee-496f-9fa6-2065baf6e192	92.45 KB	image/jpeg	Jun 13, 2022
<input type="checkbox"/>	 33980bbc-9021-4c79-9e39-0bb0f6576aea	171.57 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 492ea367-9a56-4ba1-b823-017b4ad2dc84	60.89 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 649f993f-a33f-48c4-866f-529f7e628ac0	64.88 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 8242121f-c084-420b-b247-f6b83b501fcb	126.06 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 872af8c0-bd3c-4ca6-a3b5-8dd9bc2a86ba	102.76 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 89ed6bca-b35b-429d-94cd-aaa3380285e0	60.89 KB	image/jpeg	Jun 12, 2022
<input type="checkbox"/>	 925e5efc-a1ed-4823-a056-af0b90ae2f58	135.4 KB	image/jpeg	Jun 12, 2022

 872af8c0-bd3c-4ca6-a3b5-8dd9bc2a86ba X



Name

872af8c0-bd3c-4ca6-a3b5-8dd9bc2a86ba...

Size

105,227 bytes

Type

image/jpeg

Created

Jun 12, 2022, 2:00:01 PM

Updated

Jun 12, 2022, 2:00:01 PM

File location

▼

Other metadata

▼

Рисунок 3.7 - Вміст папки Images

3.4 Результати виконання у вигляді скріншотів додатку

Як було зазначено у функціональних вимогах до програмного продукту, користувач має мати змогу відновити пароль від свого акаунту, якщо було забуто поточний. Сценарій виконання даного запиту представлено на рисунку 3.8.

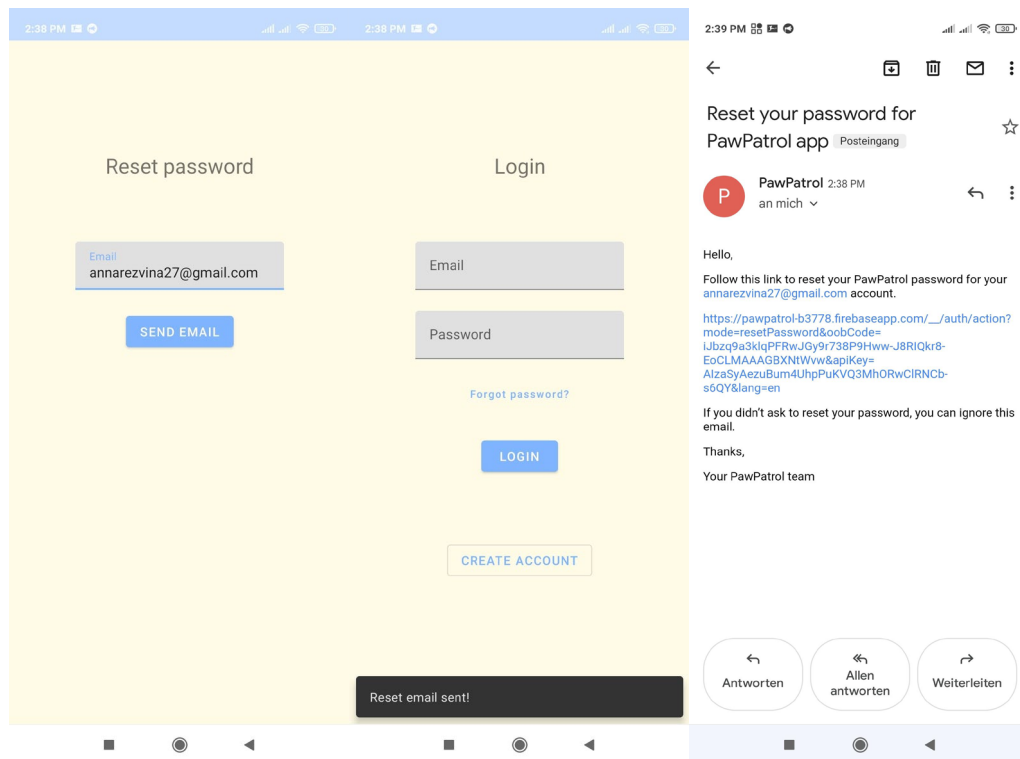


Рисунок 3.8 - Відновлення паролю акаунту

Наступний сценарій перевіряє коректне введення персональних даних(поля не повинні порожніми). Якщо користувача з введеним імейлом не знайдено, то виводиться відповідне повідомлення. Результат виконання на рисунку 3.9.

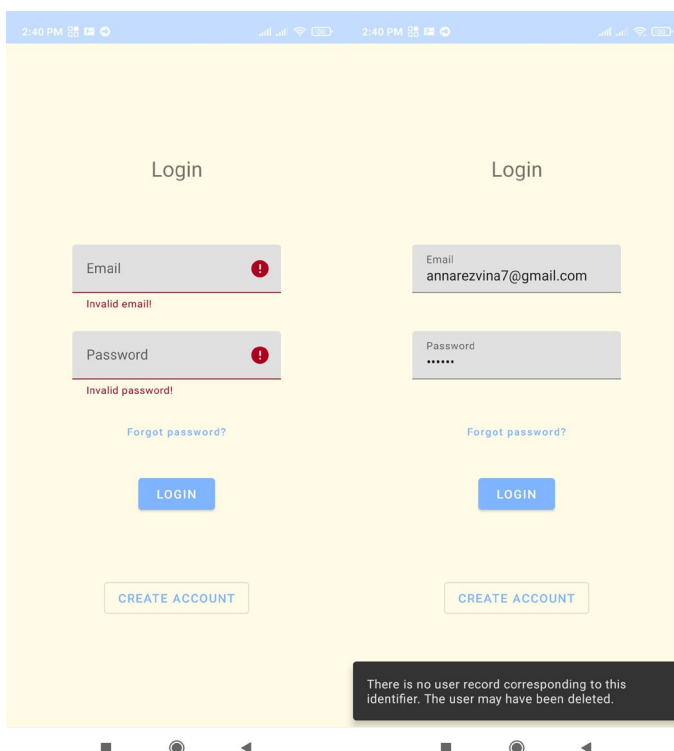


Рисунок 3.9 - Виконання сценарію Log in

Рисунок 3.10 демонструє процес реєстрації. Додаток перевіряє введений пароль та імейл на відповідність вимогам(імейл має відповідати стандартному шаблону, а пароль повинен містити не менше 6 символів).

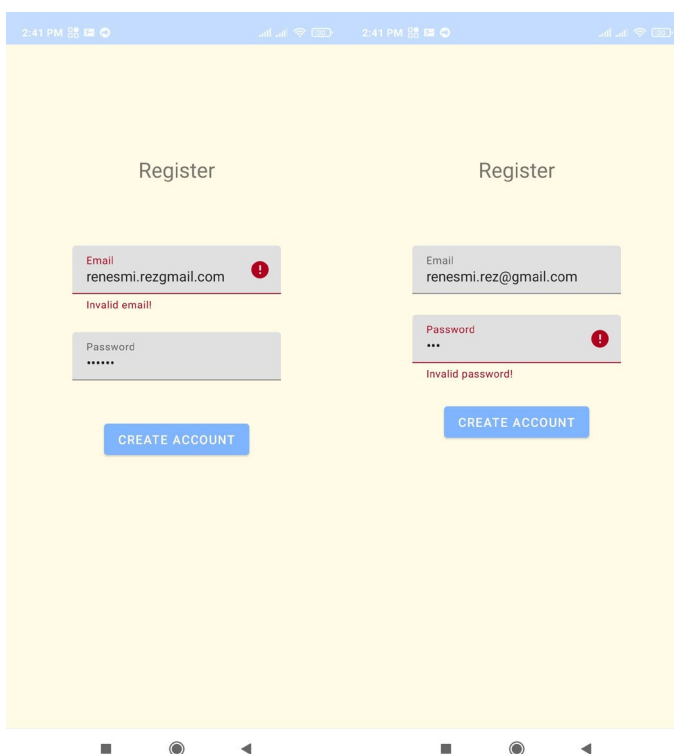


Рисунок 3.10 - Процес перевірки даних для реєстрації

На рисунку 3.11(а, б) виведено списки оголошень відповідно до двох варіантів сценаріїв. Перший сценарій демонструє список оголошень до того, як користувач увімкнув сервіси геолокації, тому оголошення відсортовані за параметром час(від найновішого до найдавнішого). Другий сценарій демонструє список оголошень після того, як користувач увімкнув сервіси геолокації, тому оголошення відсортовані за параметром дальність відстані(від найближчого до найвіддаленішого).

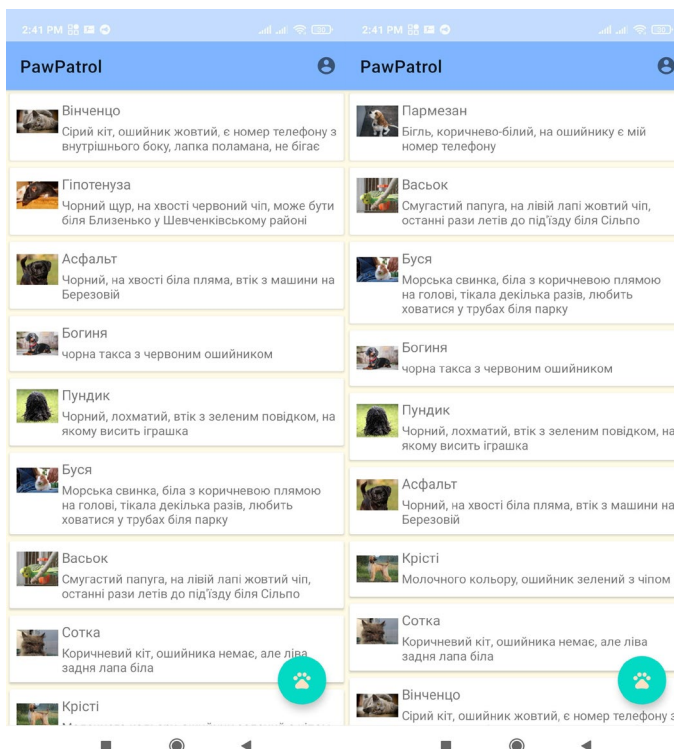


Рисунок 3.11: Загальний список оголошень

а) - сортування списку оголошень за параметром час;

б) - сортування списку оголошень за параметром відстань;

У профілі користувача знаходиться перелік персонально створених оголошень. На рисунку 3.12 зображено профіль користувача, що не має створених оголошень.

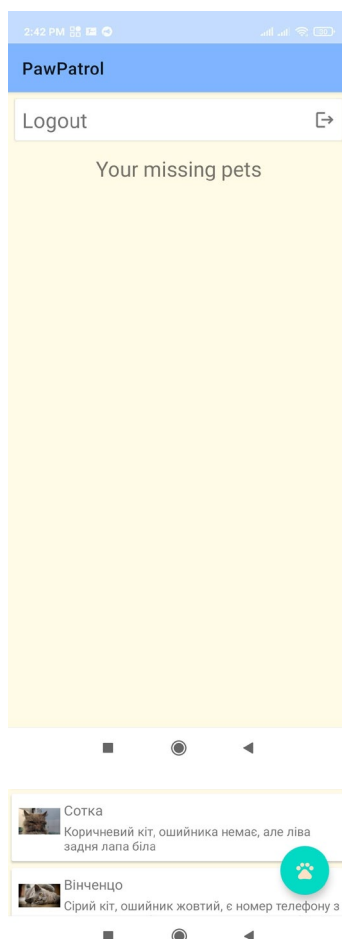


Рисунок 3.12 - Порожній список персональних оголошень користувача

Користувач може створити нове оголошення шляхом натискання на кнопку з іконкою “Лапка” зеленого кольору, що міститься в правому нижньому куті екрану.

Після натискання з’являється порожня форма створення оголошення, в яку потрібно вводити ім’я тваринки, короткий опис, вказувати геолокацію та додавати фотографію домашнього улюбленця. Текстові поля введення також перевіряють коректність заповненої інформації, а також перевіряється наявність картинки, адже вона є обов’язковим атрибутом створення оголошення.

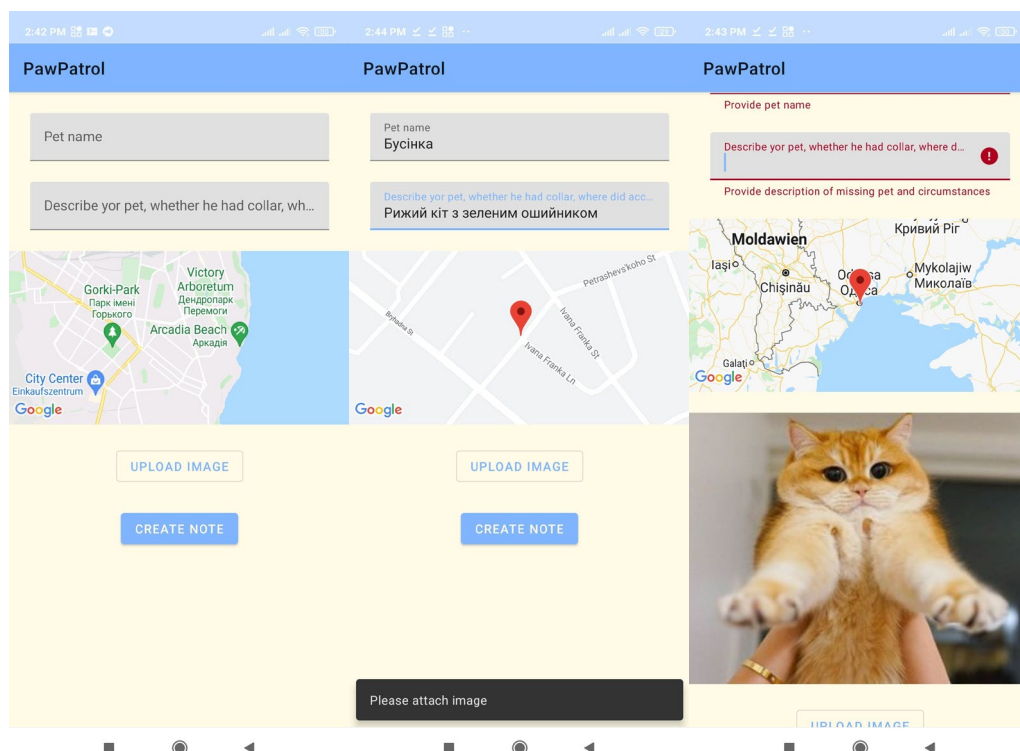


Рисунок 3.13 - Процес створення оголошення

Тепер у персональному списку оголошень користувача з'явилося створене оголошення. Це зображено на рисунку 3.14.

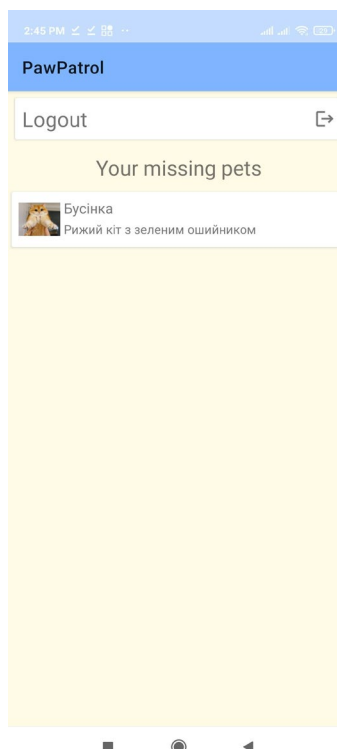


Рисунок 3.14 - Список користувача після створення оголошення

Якщо раптом користувач хоче змінити інформацію про тваринку, йому потрібно зайти на це оголошення через персональний список у своєму профілі. Тоді

у верхньому правому куті користувачу буде доступна іконка “Олівець”, що переведе оголошення у режим редагування. Під час редагування оголошення у користувача є можливість змінити будь-яку інформацію, яка є в оголошенні. Процес показано на рисунку 3.15

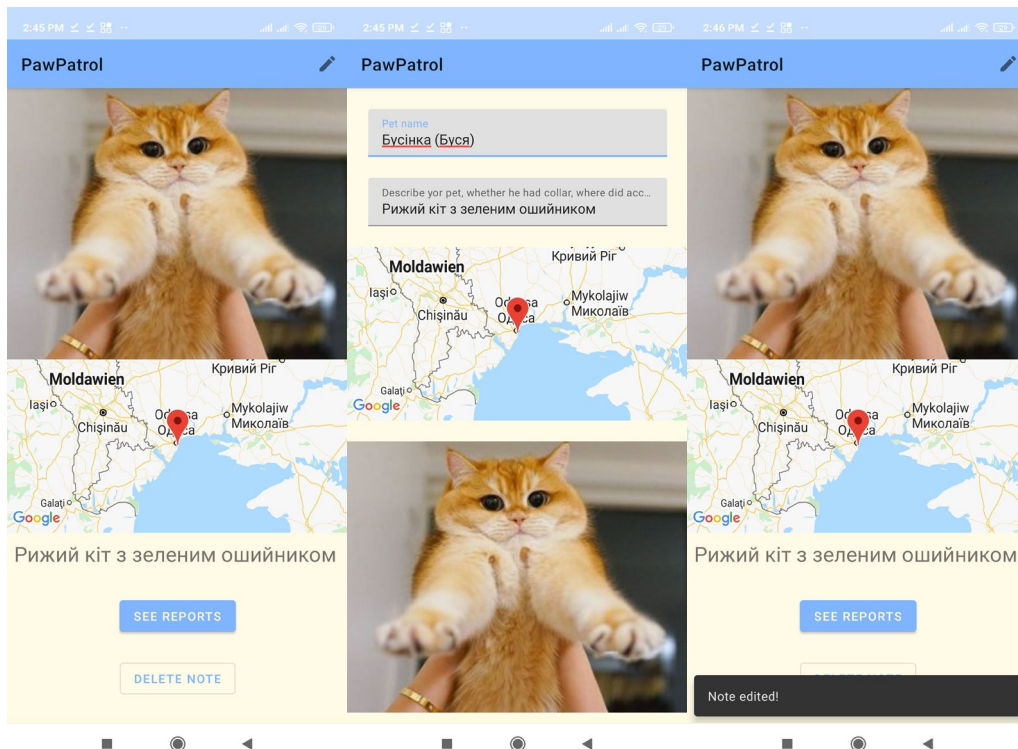


Рисунок 3.15 - Процес редагування оголошення

Після збереження зміненого оголошення воно також змінюється у списку користувацьких оголошень. Якщо зайти на сторінку цього оголошення і натиснути на кнопку “See Reports”, то можна побачити, що поки що на даному оголошенні немає коментарів.

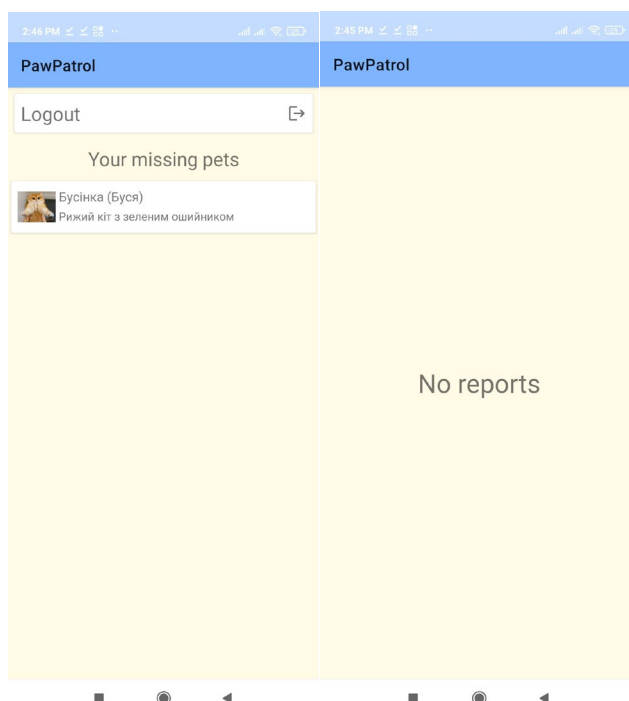


Рисунок 3.16 - Перегляд коментарів до зміненого оголошення

Розглянемо процес створення коментаря. Для того, щоб додати коментар треба відкрити оголошення і натиснути на кнопку “Report Pet To Owner”. Відкриється форма, де треба вказати поточне місцезнаходження зазначеної тваринки та по можливості додати фотографію. Процес створення коментаря продемонстровано на рисунку 3.17.

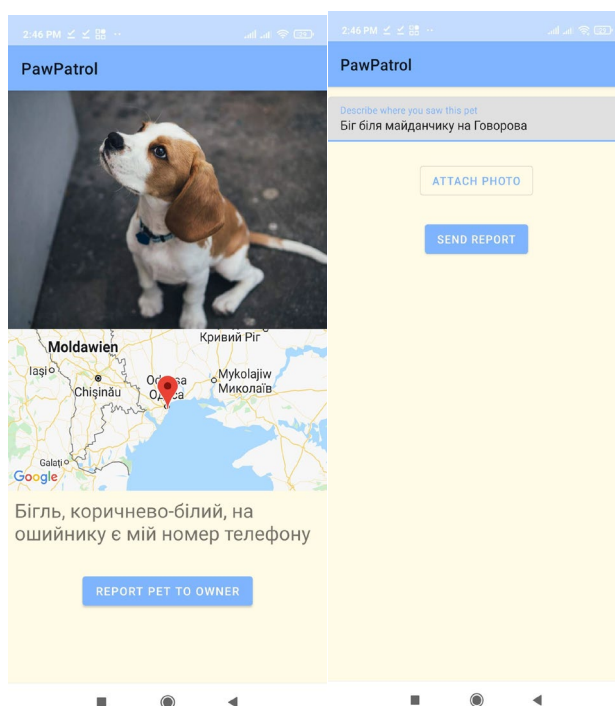


Рисунок 3.17 - Процес створення коментаря

Тепер подивимося на профіль іншого користувача: у цього користувача створені два оголошення про Крісті та Пармезана. Переглянемо коментарі під оголошенням про собаку Пармезан. Екран, що містить коментарі різного вигляду продемонстровано на рисунку 3.18

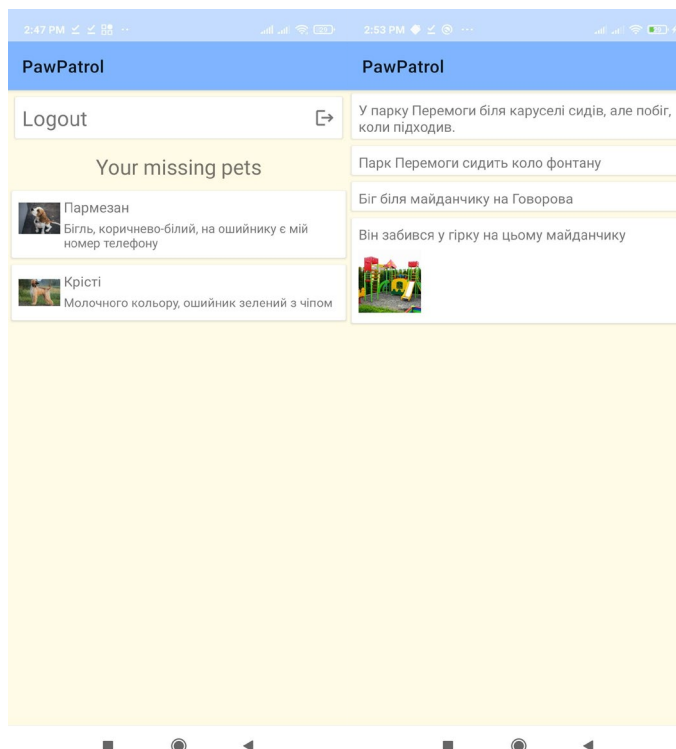


Рисунок 3.18 - Перегляд коментарів під оголошенням

На рисунку 3.19 показано, як виглядає іконка додатку на екрані додатків у телефоні. Вона має такий самий стиль та колір, як кнопка основної дії у додатку, та відразу викликає чітку асоціацію з темою додатку.

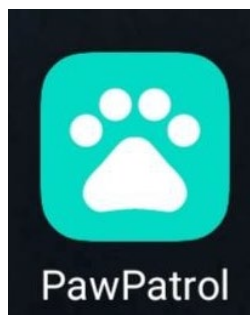


Рисунок 3.19 - Іконка додатку

ВИСНОВКИ

Під час виконання курсової роботи було проведено аналіз існуючих аналогів мобільного додатку, що розробляється, та виявлено їх недоліки. Під час виконання роботи було проведено аналіз та обрано інструменти розробки, які максимально відповідають вимогам до сучасного мобільного додатку на операційній системі Android. У результаті було обрано Kotlin, як основну мову програмування, та Firebase Realtime Database разом з Storage у якості бази даних. Також було використано програмований, картографічний сервіс Google Maps API . Розробка велась у інтегрованому середовищі розробки Android Studio, що який розроблено спеціально для технологій Android, тому він включає тільки потрібні для розробки інструменти.

Також було проведено ретельне ручне тестування графічного інтерфейсу користувача, і представлено максимальну кількість можливих сценаріїв взаємодії користувача з додатком.

У результаті виконання курсової роботи було створено додаток типу Mobile App – “PawPatrol”, що дозволяє своїм користувачам переглядати та створювати численні оголошення загублених домашніх тваринок та отримувати актуальну інформацію про їхнє місцезнаходження від інших користувачів у коментарях. Для подальшого розвитку додатка можливим є розширення соціальних функцій програмного продукту, наприклад, усиновлення домашніх тварин, чиїх господарів не було знайдено або створення функціоналу “Чат з власником” окрім доступу до коментарів. Але в цьому випадку окрему увагу варто приділити безпеці користувачів шляхом залучення модераторів та обмеження доступу до певної персональної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kotlin vs Java: Important Differences That You Must Know – [Електронний ресурс]. URL: <https://hackr.io/blog/kotlin-vs-java>
2. Visual Studio Code vs Android Studio – Functionality, Search and Source Control – [Електронний ресурс]. URL: <https://blog.codemagic.io/android-studio-vs-visual-studio-code/>
3. Compare Android Studio and Visual Studio – [Електронний ресурс]. URL: <https://www.g2.com/compare/android-studio-vs-visual-studio>
4. Maps SDK for Android – [Електронний ресурс]. URL: <https://developers.google.com/maps/documentation/android-sdk/config>
5. Firebase Realtime Database – [Електронний ресурс]. URL: https://www.techotopia.com/index.php?title=Firebase_Realtime_Database&mobileaction=toggle_view_mobile
6. Structure Your Database – [Електронний ресурс]. URL: <https://firebase.google.com/docs/database/web/structure-data>
7. Firebase Documentation – [Електронний ресурс]. URL: <https://firebase.google.com/docs/database>
8. Firebase: Realtime Database Reading and Writing – [Електронний ресурс]. URL: <https://w3tpoint.com/firebase/firebase-realtime-database-reading-and-writing>
9. MYSQL ADVANTAGES AND DISADVANTAGES – [Електронний ресурс]. URL: <https://blueclawdb.com/mysql/advantages-disadvantages-mysql/>
10. Kotlin docs – [Електронний ресурс]. URL: <https://kotlinlang.org/>
11. Develop Android apps with Kotlin – [Електронний ресурс]. URL: <https://developer.android.com/kotlin>

ДОДАТКИ

Додаток А

(посилання на репозиторій проекту)

https://github.com/anng34/_PetFinder_

