Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система для відстеження домашніх тварин.*

Виконав: студент *IV* курсу, групи *CIc-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

	(підпис)	*Степанко О.І.* (прізвище та ініціали)
Керівник	(підпис)	*Луцик Н.С.* (прізвище та ініціали)
Нормоконтроль	(підпис)	*Тиш Є.В.* (прізвище та ініціали)
Завідувач кафедри	(підпис)	*Осухівська Г.М.* (прізвище та ініціали)
Рецензент	(підпис)	*Гащин Н.Б.* (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студента *Степанко Оксани Ігорівни*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система для відстеження домашніх тварин.*

Керівник роботи *Луцик Надія Степанівна, к.т.н., доцент кафедри КС*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28* « *лютого* 2023 року № *4/7-237*
2. Термін подання студентом завершеної роботи *19.06.2023 р.*
3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання
2. Проєктна частина
3. Практична частина
4. Безпека життєдіяльності, основи охорони праці.
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. *Функціональна схема системи.*
2. *Діаграма варіантів використання системи.*
3. *Структурна схема Android -додатку.*
4. *Результат роботи системи.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,			
основи охорони праці			

7. Дата видачі завдання 02.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка технічного завдання*	*02.03-25.03.2023*	
2	*Аналіз технічного завдання*	*26.03-01.04.2023*	
3	*Аналіз вимог до системи відстеження тварин*	*02.04-16.04.2023*	
4	*Проектування структури системи пошуку тварин*	*17.04-04.05.2023*	
5	*Проектування програмної частини*	*05.05-10.05.2023*	
6	*Проектування апаратної частини*	*11.05-29.05.2023*	
7	*Налаштування і тестування системи відстеження тварин*	*30.05-05.06.2023*	
8	*Безпека життєдіяльності, основи охорони праці*	*06.06-10.06.2023*	
9	*Оформлення кваліфікаційної роботи*	*11.06-15.06.2023*	
10	*Попередній захист кваліфікаційної роботи*	*16.06-20.06.2023*	
11	*Захист кваліфікаційної роботи*	*19.06-24.06.2023*	

Студент _____ *Степанко Оксана Ігорівна*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцик Надія Степанівна.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система для відстеження домашніх тварин // Кваліфікаційна робота на здобуття освітнього ступеня бакалавр // Степанко Оксана Ігорівна // ТНТУ, спеціальність 123 «Комп’ютерна інженерія» // Тернопіль, 2023 // с.– 60 , рис. – 22 , табл. – 16, аркушів А1 – 4, бібліогр. – 17.

Ключові слова: відстеження, трекер, GPS, домашні тварини.

У даній кваліфікаційній роботі виконано розробку програмно-апаратного комплексу для відстеження розташування домашніх тварин.

Пояснювальна записка містить 4 розділи.

В першому розділі здійснюється аналіз предметної області. Проведено огляд вимог до системи відстеження домашніх тварин, розглянуті технології відстежування, а також визначені задачі кваліфікаційної роботи.

В другому розділі описані компоненти і особливості проектування системи відстежування домашніх тварин.

В третьому розділі процедура налаштування і тестування системи відстеження домашніх тварин.

В четвертому розділі розглянуті питання охорони праці та вимоги з техніки безпеки.

ABSTRACT

Computerized system for tracking domestic animals // Qualification work for the bachelor's degree // Stepanko Oksana Igorivna // TNTU, specialty 123 "Computer engineering" // Ternopil, 2023 // p.– 60, fig. - 22, table – 16, sheets A1 – 4, bibliography. - 17.

Keywords: tracking, tracker, GPS, domestic animals.

In this qualification work, the software and hardware complex for tracking the location of domestic animals was developed.

The explanatory note contains 4 sections.

In the first section, the analysis of the subject area is carried out. An overview of the requirements for the pet tracking system was conducted, tracking technologies were considered, and the tasks of the qualification work were defined.

The second section describes the components and design features of the pet tracking system.

In the third section, the procedure for setting up and testing the pet tracking system.

The fourth chapter deals with occupational health and safety requirements.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

GPS	Global Positioning System
GSM	Global System for Mobile Communications
PHP	Hypertext Preprocessor
UML	Unified Modeling Language
БД	база даних
ОС	операційна система
ТЗ	технічне завдання

ВСТУП

Останнім часом на вулицях міст можна побачити сотні бездомних тварин, які колись були домашніми. Частина із них загубилися або втекли від своїх господарів.

Найчастіше тварина, яка загубилася не може самостійно знайти дорогу додому. Також відомі випадки крадіжки тварин, особливо дорогих та породистих. У таких випадках ймовірність сприятливого, самостійного повернення тварини додому ще менша. Тоді людині доводиться самій шукати свою домашню тварину. Зазвичай пошук вихованців ґрунтується на інформації про особливості його поведінки та характеристики навколишньої території. Пошук, заснований на інтелектуальному аналізі поведінки тварини та характеристик місцевості, займає багато часу і не завжди дає позитивний результат. На вулиці домашніх тварин чекає велика кількість небезпек, тому власник має знайти свого друга якнайшвидше.

Необхідно створити трекер для домашніх вихованців. Звичайно, існує безліч таких пристроїв. Але моделі, які зараз представлені на ринку, відрізняються високою вартістю або недостатньо великим функціоналом. У зв'язку з цим, було прийнято рішення розробити програмно-апаратний комплекс для відстеження розташування домашніх вихованців, який матиме переваги перед своїми аналогами та відповідатиме основним вимогам споживачів.

Створення такого пристрою допоможе не тільки знаходити людям своїх вихованців, а й скоротить кількість безпритульних тварин та зробить вулиці міст безпечнішими для знаходження людей.

Метою кваліфікаційної роботи є розробка комп’ютеризованої системи, яка використовує GPS для визначення місцезнаходження, а також створення програми на Android, яка буде виводити дані про місцезнаходження власника та місцезнаходження тварини на карту.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Огляд існуючих рішень

Існує досить велика різноманітність пристроїв з можливістю відстеження розташування. Вони багато в чому полегшують життя людей, допомагаючи не тільки визначати своє місцезнаходження чи шлях до необхідного місця.

Для виявлення переваг та недоліків були обрані для розгляду 3 найбільш популярні пристрої для відстеження розташування домашніх тварин: Garmin Astro 320 з нашийником DC50, TK STAR Pet tracker (TK 909), Garmin Delta Smart.

Навігатор Garmin Astro 320 з нашийником DC50 - пристрій для відстеження розташування собак на полюванні на відстані до 14,5 км. Нашийник отримує сигнал GPS далі передає інформацію про місцезнаходження та траєкторію руху собак на пристрій Garmin Astro 320. Сигнал від нашийника передається кожні 5, 10, 15, 30 або 120 секунд залежно від налаштувань і заряду батареї.

Навігатор працює з готовими картами з Інтернету, а також з картами, які може створювати користувач самостійно із растрових зображень карт або знімків із супутника. На карті відображається розташування собаки та власника тварини. Також пристрій може відображати стан собаки (на карті буде видно сидячий вихованець, стоїть або біжить). Крім того, від нашийника надходить сигнал, якщо собака гавкає.

Пристрій може відображати інформацію про кількох вихованців одночасно, траєкторію руху та пройдену собакою відстань [1].

На рисунку 1.1 представлений зовнішній вигляд пристрою:

Рисунок 1.1 - Навігатор Garmin Astro 320 з нашийником DC50

Пристрій має ряд переваг таких, як:

- час автономної роботи пристрою до 54 годин у режимі економії енергії (дані пересилаються лише раз на дві хвилини) та до 24 годин у стандартному (радіосигнал з координатами надсилається кожні 5 секунд);
- високий рівень захисту від води та пилу, клас захисту IP- 68, що дозволяє занурювати нашийник під воду на глибину до 10 метрів;
- великий функціонал трекера;
- корпус підвищеної міцності, а також надійне кріплення пристрою до нашийника захищає пристрій від пошкоджень та знижує ймовірність втрати пристрою.

З недоліків можна виділити:

- високу вартість;
- вага нашийника 280 грам;
- розміри трекера 61х169х36 мм, що не дозволяє використовувати трекер для маленьких собак, котів та інших тварин.

TK STAR Pet tracker (TK 909) - недорогий GPS трекер для тварин. Для

відображення розташування тварини та вибору режиму роботи існує програма для Android та IOS. Також можна отримати посилання на Google maps у SMS повідомленнях, після виклику пристрою за телефонним номером SIM -картки з мобільного телефону. Реалізовано можливість надсилання SMS команди на трекер для налаштувань режимів його роботи та отримання необхідних даних (координати, заряд акумулятора та ін.). Пристрій оснащений мінімальними функціями для відстеження розташування тварин. Має невеликі розміри та вагу, отже, підходить для котів та собак дрібних порід. У комплекті йде нашийник, на який кріпиться пристрій. [2]

На рисунку 1.2 представлений зовнішній вигляд пристрою:

Рисунок 1.2 - TK STAR Pet tracker (TK 909)

З переваг можна виділити:

- низьку вартість;
- невеликі розміри трекера 70x37x20 мм;
- наявність кросплагформенного додатка;
- вага 46 грам.

З недоліків можна виділити:

- недостатній захист від води та пилу, рівень якого IP-54, що не дозволяє занурюватися вихованцю під воду, а витримує лише невелике попадання бризок на корпус;

- пластмасове кріплення, яке є недостатньо міцним і не може гарантувати гарне кріплення приладу до нашийника, що підвищує ймовірність втрати пристрою;
- в даному трекері використовуються застарілі модулі GSM та GPS з високим енергоспоживанням, що не може забезпечити тривалу роботу пристрою, а також недостатню точність позиціонування;
- мобільний додаток універсальний для всіх навігаційних пристроїв фірм TKSTAR. З недоліків мобільного додатка можна виділити: неповну русифікацію програми, відсутність можливості відображення розташування власника пристрою, а також не можливість додавання кількох вихованців в одному профілі. Рейтинг програми не високий у Play Market 3,4 у App Store 3,8.

Garmin Delta Smart - пристрій для дресирування собак, а також для відстеження розташування та стану тварини. За допомогою мобільного додатка Garmin Canine можна використовувати смартфон як віддалений портативний тренажер, а також переглядати місце розташування тварини і його переміщення. Також існує пульт для віддаленого керування пристроєм. Комплект з пультом коштує дорожче. Пристрій кріпиться до нашийника тварини [3].

На рисунку 1.3 представлений зовнішній вигляд пристрою:

Рисунок 1.3 - Навігатор Garmin Delta Smart

Переваги:

- наявність додаткового функціоналу, це не тільки трекер для відстеження розташування, а й пристрій для дресирування;
- тривалий час роботи;
- міцне кріплення пристрою до нашийника;
- габарити пристрою: 63 х 35 х 34 мм ;
- вага : 41 г;
- водонепроникність на глибині до 10 метрів.

Недоліки:

- відсутність у додатку українського інтерфейсу;
- Висока ціна.

Винесемо основні характеристики трекерів окрему таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика трекерів

	Garmin Astro 320 з нашийником DC50	TK STAR Pet tracker (TK 909)	Garmin Delta Smart
Ціна	14340	2300	6000
Тривалість автономної роботи режимі економії енергії)	До 56 годин	До 14 годин	До 20 годин
Вага (г)	280	46	41
Габарити (мм)	61 х 160х45	70 х 37 х 20	63 х 35 х 34
Клас зашиті від води та пилу	IP-68	IP-54	IP-57
Наявність програмного забезпечення	+	+	+
Визначення розташування власника пристрою	+		+

Існуючі пристрої мають деякі недоліки і не можуть повною мірою задовольнити всі вимоги власників домашніх вихованців. Тому необхідно

розробити власний пристрій, який матиме перевагу перед існуючими, традиційними способами пошуку тварин, а також мати ряд переваг перед своїми аналогами.

1.2 Вимоги до системи відстеження домашніх тварин

1.2.1 Нефункціональні вимоги

Визначено такі нефункціональні вимоги:

- Програма не повинна залежати від апаратної частини;
- Додаток повинен мати можливість масштабуватись і не зациклюватися на одному пристрої;
- Дані мають передаватися у зашифрованому вигляді. Так як система зберігатиме e-mail користувачів та їх паролі, повинна забезпечуватися захищена передача цих даних;
- Неможливість проходження процедури реєстрації без заповнення всіх полів, також інформація, що вводиться, повинна відповідати вимогам поля;
- Наявність ідентифікатора або QR коду для забезпечення унікальності пристрою;
- Доступ до системи можливий лише для зареєстрованих користувачів після проходження процедури реєстрації;
- Час роботи пристрою визначається характеристиками пристрою живлення;
- Пристрій повинен перебувати в сплячому режимі та включатися по натисканню кнопки;
- Наявність акумулятора, що знімається;
- Розміри пристрою не повинні перевищувати наступних параметрів: 15*50*50 мм;
- Вага пристрою не повинна перевищувати 60 грам.

Вимоги до системи:

- Система повинна працювати з будь-яким Android – пристроєм;
- Версія Android має бути не менше 4.4.

1.2.2 Функціональні вимоги

Для того щоб виділити функціональні вимоги до системи побудуємо діаграму прецедентів, вона відображає події, що виникають у системі з точки зору користувача, зображені за допомогою діаграми прецедентів у нотації UML [4]. На рисунку 1.4 зображено діаграму прецедентів. Дана діаграма використовується для відображення прагматики системи, що розробляється, описи структури об'єктів, з яких вона складається, їх взаємозв'язку та атрибутів. Ролі розподілені між Android -додатком, користувачем, трекером, базою даних та картами. В овали, у свою чергу, вписані прецеденти використання системи, що розробляється.

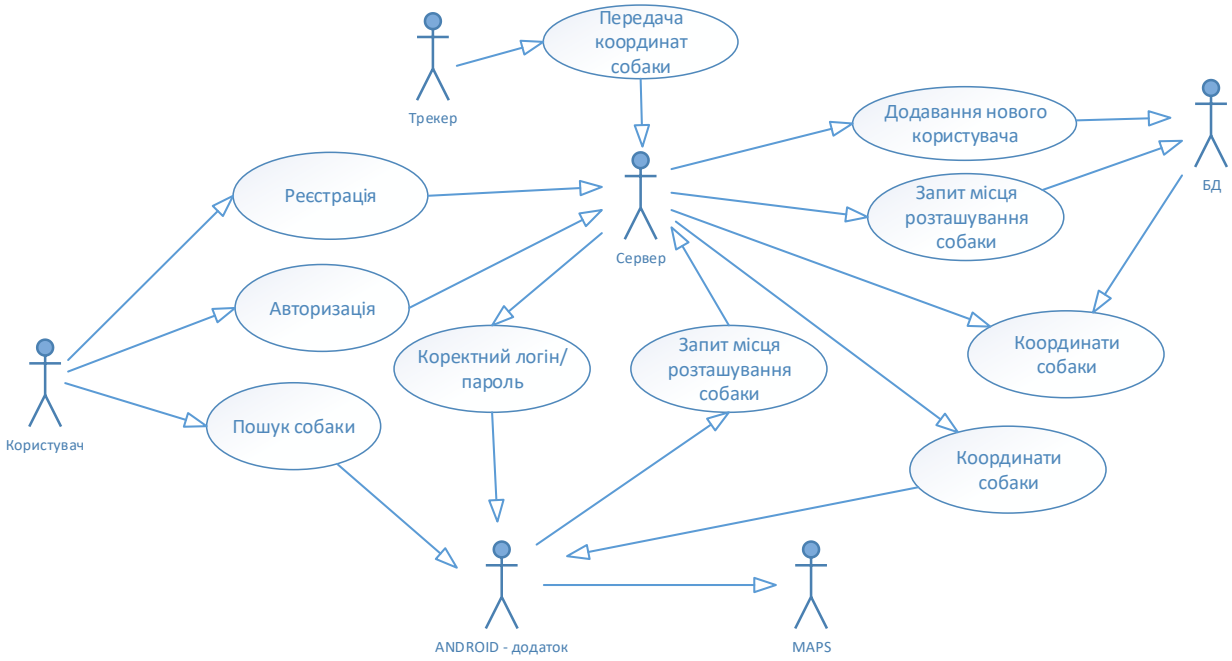

Рисунок 1.4 – Діаграма варіантів використання системи

Якщо користувач не був раніше зареєстрований у системі, то перед

початком роботи йому необхідно пройти процедуру реєстрації для цього, необхідно вказати у відповідних полях ім'я облікового запису, пароль, прізвище, ім'я, дату народження, кличку тварини та ідентифікатор нашийника. Користувачам, які раніше проходили цю процедуру, необхідно авторизуватися, щоб мати можливість працювати в системі: для цього їм необхідно ввести у відповідному інтерфейсі системі своє ім'я облікового запису та пароль.

Після цього користувач може перейти на карту і подивитися місце знаходження тварини або своє власне. Трекер у режимі реального часу отримує із супутників свої координати. Після цього вони можуть бути відображені у додатку на карті.

1.3 Постановка задач кваліфікаційної роботи

Сучасна система відстежування домашніх тварин повинна забезпечувати точність позиціювання, мати зручний інтерфейс і довгий час автономної роботи.

З проведеного вище огляду і аналізу можливих рішень, а також враховуючи вищенаведені вимоги, мета кваліфікаційної роботи буде досягнута шляхом вирішення наступних завдань:

– Розробка структури проектованої системи.

– Обґрунтування і розробка програмної частини.

– Обґрунтування модулів і розробка апаратної частини.

– Виготовлення, складання та налагодження макетного зразка трекера.

– Тестування трекера.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

Для реалізації цієї системи необхідний наступний набір підсистем:

– Клієнтська програма на Android. Програма забезпечує доступ користувача до системи, а також можливість виведення результату до програми на карту.

– Графічний інтерфейс клієнтської програми. Візуальне відображення реєстрації та авторизації в системі, відображення карти та позначок місцезнаходження.

– Серверний додаток. Забезпечує взаємодію апаратної реалізації системи та клієнтського застосування.

– База даних SQLite. База даних забезпечує зберігання даних про користувачів та вихованців, а також про їх координати.

Апаратна реалізація системи. Пристрій, що дозволяє визначати місце розташування тварини та відправляти отримані координати користувачеві.

2.1 Розробка структури системи пошуку домашніх тварин

Для опису загальної структури системи пошуку домашніх тварин можна подати функціональну схему. Ця схема пояснює окремі види процесів, які у проходять у функціональних блоках системи. На рисунку 2.1 представлено функціональну схему системи.

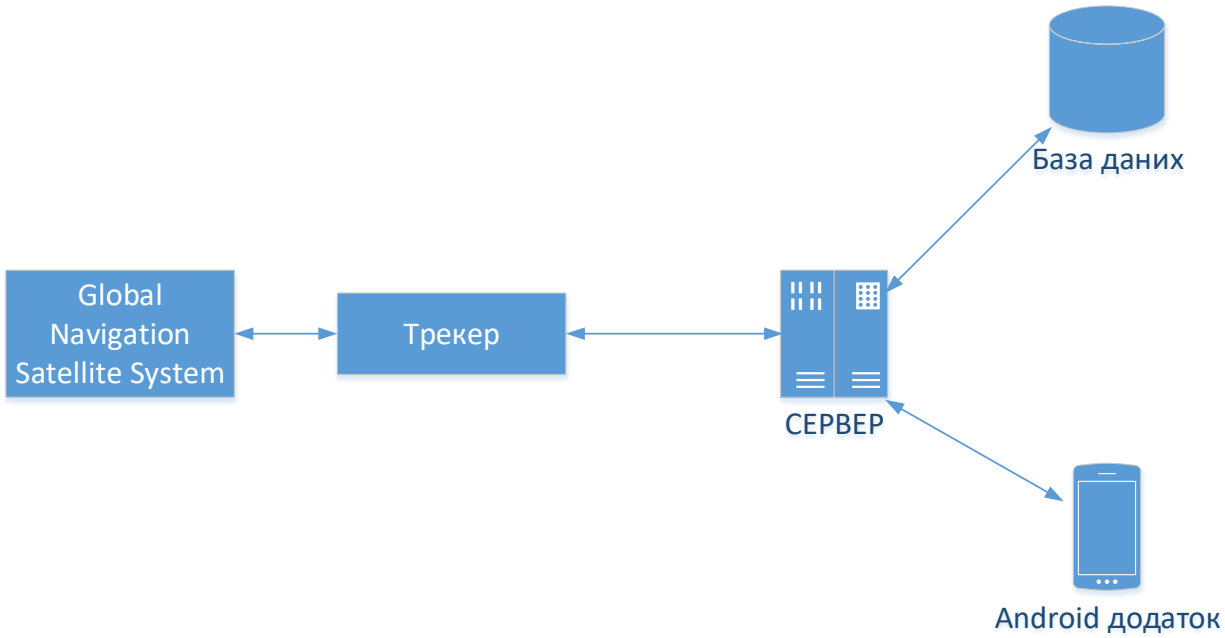

Рисунок 2.1 – Функціональна схема системи

Вся система взаємодіятиме через сервер. Трекер отримуватиме інформацію про своє місцезнаходження з супутників і передаватиме їх на сервер, який оброблятиме отримані координати та зберігатиме їх. У додатку за допомогою запитів буде реалізована можливість отримання необхідної інформації з БД, а також надсилання даних до бази при реєстрації нових користувачів. У БД буде міститися інформація про користувачів, тварин та їх координати. Також у додатку буде відображатися карта з отриманими координатами.

Для опису структурного складу програмної частини системи необхідно побудувати її структурну схему. Внутрішні компоненти відповідають за логіку функціонування та зберігання даних, а зовнішні за візуальну взаємодію із системою.

Структурна схема системи представлена рисунку 2.2. Структурна схема Android - додатку включає в себе внутрішні та зовнішні компоненти. Склад зовнішніх компонентів: інтерфейс програми та графічні елементи на карті. У внутрішні компоненти входять: база даних, функції введення/виведення даних та сервер.

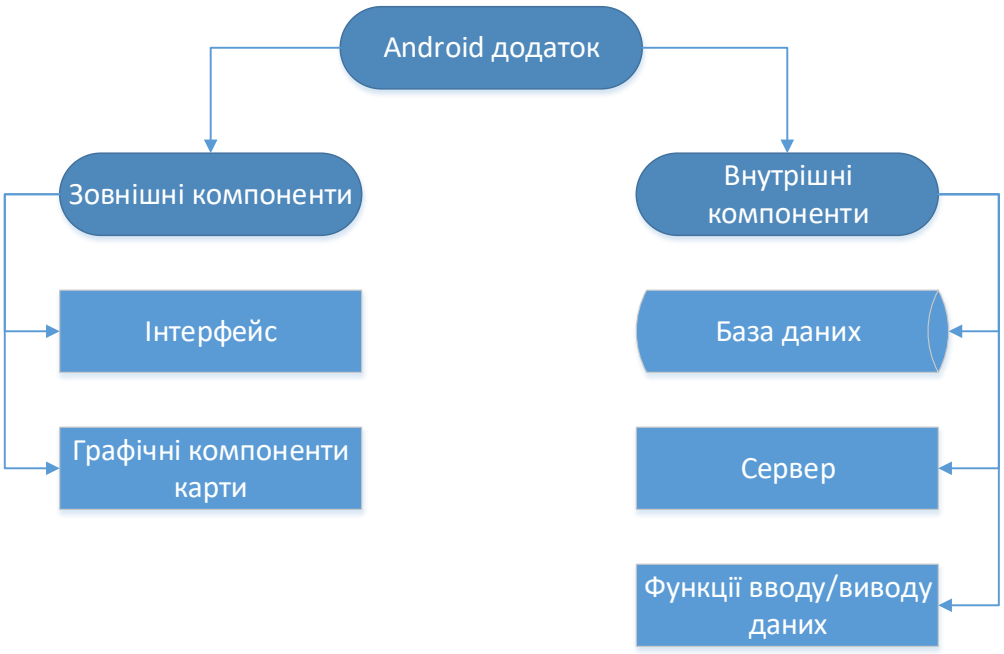

Рисунок 2.2 - Структурна схема Android -додатку

Основним призначенням нашого пристрою буде отримання координат розташування трекера та передача їх у додаток на Android, де вони відображатимуться на карті.

Для розробки пристрою необхідно визначитися з технологіями, що використовуються для визначення координат і передачі даних.

2.2 Обґрунтування вибору апаратного забезпечення

На першому етапі проектування необхідно визначити параметри основних модулів, які будуть використовуватись для системи відстеження домашніх тварин. Відповідно до проведеного огляду і вимог необхідно визначити які модулі стеження будуть використані і визначитись з елементами живлення системи відстеження домашніх тварин.

2.2.1 Вибір технології відстежування розташування домашніх тварин

Для відстежування розташування використовуються різні технології, такі як GPS, GSM та мережа Wi-Fi. Розглянемо кожну з них докладніше:

Найбільшою і розвиненою інфраструктурою позиціонування є GPS. Ця система визначає відстань, отже, і розташування шляхом розрахунку часу, протягом якого радіосигнал доходить від супутникового передавача до наземного приймача. Сигналів від чотирьох супутників достатньо, щоб визначити місце розташування приймача в тривимірному просторі [5]. Координати визначаються з похибкою 2-4 метри. З недоліків GPS можна виділити неможливість визначення розташування у приміщенні. Тому у сучасних модулях використовується технологія A-GPS. Перевага цієї технології полягає в тому, що A-GPS залежить не тільки від супутників, а й від мережевих ресурсів. Наявність цієї технології є важливим параметром під час вибору модуля.

Визначення розташування на основі сигналів GSM також широко поширене. Суть технології в тому, щоб аналізувати інформацію про ті базові станції стільникового зв'язку, які знаходяться поблизу. Однак точність помітно менша, ніж у GPS, похибка збільшується до 50-100 метрів, а якщо визначення розташування виробляються в сільській місцевості або в невеликих містах, де в будь-якій точці є сигнал тільки від однієї, максимум - двох базових станцій одного оператора, то можлива похибка зростає до 10-15 км [5].

Також є метод позиціонування за допомогою мереж Wi-Fi. Цей метод використовується компанією Cisco і полягає в тому, щоб здійснити тріангуляцію рівня сигналу від точок доступу Wi-Fi і таким чином відпозиціонувати пристрій [6]. Цей метод є досить інформативним. Точність даного методу 5-7 м.

Таким чином, з представлених технологій визначення розташування за допомогою GPS є найбільш оптимальним. Також на ринку є модулі, які поєднують у собі GPS для визначення місцезнаходження та GSM/GPRS для передачі даних. Один із таких модулів це Quectel MC60. У таблиці 2.1 порівняємо його характеристики з характеристиками модулів які також

можуть бути використані для реалізації даної системи.

Таблиця 2.1 – Порівняння модулів

	Quectel L76-L + Quectel M66	Quectel MC60
	GNSS частина	
Чіпсет	MTK3333	MTK3333
Навігаційні системи	GPS	GPS
Чутливість при стеженні (дБ)	-165	-167
Чутливість при виявленні (дБ)	-148	-152
Енергоспоживання при стеженні (мАг)	22	22
Енергоспоживання при виявленні (мАч)	29	29
Підтримка технології A-GPS	+	+
Час холодного старту (с)	35	35
Час гарячого старту <9	1	1
Напруга живлення V	2,8 ~ 4,5	2,8~4,3
Тип антени	Зовнішня	Зовнішня
	GPRS частина	
Технологія	GSM/GPRS	GSM/GPRS
Частоти (МГц)	850/900/1800/1900	850/900/1800/1900
Енергоспоживання в режимі очікування (мАг)	1,2	1,2
Напруга живлення (В)	3,3 ~ 4,6	3,3~4,6
	Загальне	
Габарити (мм)	10,1 x 9,7 x 2,5 17,7 x 15,8 x 2,3	18,7 x 16 x 2,1

З таблиці видно, що використання модуля, який поєднує в собі дві технології є більш оптимальним, оскільки це рішення зменшує розміри пристрою. Крім того вибір модуля Quectel MC60 полегшує роботу над передачею даних. Немає потреби взаємодії між GSM та GPS модулями. Також цей модуль підтримує технологію OpenCPU. Ця технологія дозволяє програмувати модуль без підключення зовнішнього мікроконтролера.

2.2.2 Підбір акумулятора

В даний час найбільш поширені Li-Ion та Li-Pol акумулятори [8].

При порівнянні двох типів акумуляторів можна виділити такі переваги літій-полімерних акумуляторів:

– Менша вага та розміри при ідентичних характеристиках.

– Найменший час, необхідний для зарядки.

– Вища (майже вдвічі) ємність при однаковому загальному обсязі батареї.

– Найменша вибухонебезпечність батареї.

Тому найкращим рішенням буде використання літій-полімерного акумулятора.

Порівняємо представлені над ринком моделі акумуляторів у таблиці 2.2. Основними параметрами при виборі акумулятора є невеликі розміри та вага:

Таблиця 2.2 – Порівняння акумуляторів

	Li-Pol 018265	Li-Pol 018251	Li-Pol 018251	PL803050
Тип акумулятора	Li-Pol	Li-Pol	Li-Pol	Li-Pol
Місткість (мАг)	1500	900	550	1200
Габарити (мм)	2 x 55 x 85	3 x 50 x 57	4 x 40 x 45	8x 30x50

Висновок: Виходячи з інформації, представленої в таблиці, було обрано літій-полімерний акумулятор PL803050 (рисунок 2.3).

Рисунок 2.3 - Акумулятор PL803050

Даний акумулятор призначений для Bluetooth-пристроїв, телефонів відеореєстраторів, плеєрів, GPS-навігаторів та іншої дрібної електроніки.

2.3 Обґрунтування вибору програмного забезпечення

У нашій системі немає необхідності зберігати та записувати великий обсяг даних. А як одна з головних вимог до БД можна виділити мобільність бази даних, тому що важлива можливість створення в майбутньому кросплатформового застосування. Тому оптимальним буде використання БД SQLite [9].

Програмна частина у цій роботі і складається з двох частин:

– Серверна частина.

– Клієнтська частина.

Для розробки серверної частини програми необхідно вибрати технологію. На даний момент найпопулярнішими вважаються технології PHP і ASP.NET.

При їх порівнянні можна помітити, що ASP.NET простіше PHP у плані реалізації тих самих завдань. Ще одним плюсом ASP.NET [9] є те, що програма пишеться на типізованих компілюваних .NET мовах і як наслідок спрощення налагодження. Виходячи з цього для розробки нашої програми ми будемо використовувати технологію ASP.NET, а зокрема мову C#. Також вибір ASP.NET можна обґрунтувати необхідністю масштабувати систему, оскільки кількість зареєстрованих у системі користувачів постійно збільшуватиметься.

Найбільш популярні мови для розробки додатків на Android це Java та Kotlin:

Java - це найпоширеніша мова для розробки мобільних програм на Android. Є сильно типізованою об'єктно орієнтованою мовою програмування.

Kotlin - це сучасна об'єктно орієнтована мова програмування, що компілюється для платформ Java та JavaScript.

Для розробки клієнтської частини програми було обрано мову Java , оскільки мова має безліч готових бібліотек і SDK, що прискорює процес розробки.

2.3.1 Вибір середовища розробки

Необхідно вибрати середовище розробки для серверної та клієнтської частини.

Для розробки серверної частини буде використано Microsoft Visual Studio - лінійку продуктів компанії Microsoft, що включають інтегроване середовище і інструментальні засоби для розробки програмного забезпечення.

Для розробки клієнтської частини буде використано Android Studio. Знаходиться у вільному доступі, підтримується такими ОС як Windows, OS X та Linux. Є найбільш поширеним середовищем розробки для розробки програм на Android. Має ряд переваг таких як: гнучкість середовища розробки, великий набір функцій, можливість тестування коректної роботи додатків на різних системах можна виробляти в емуляторі, можливість розробки під різні версії Android, а також доступ до бібліотек шаблонів і компонент що спрощує роботу над доатком.

2.3.2 Вибір мап

На ринку картографічних та довідкових сервісів можна виділити найбільш повну і функціональну картографічну систему Google Maps. Саме тому для даної системи було обрано ці карти.

2.3.3 Опис даних

Складемо схему БД для нашого проекту. Схема бази даних- її структура, описана формальною мовою, що підтримується СУБД. Структурна схема бази даних системи представлена рисунку 2.4.

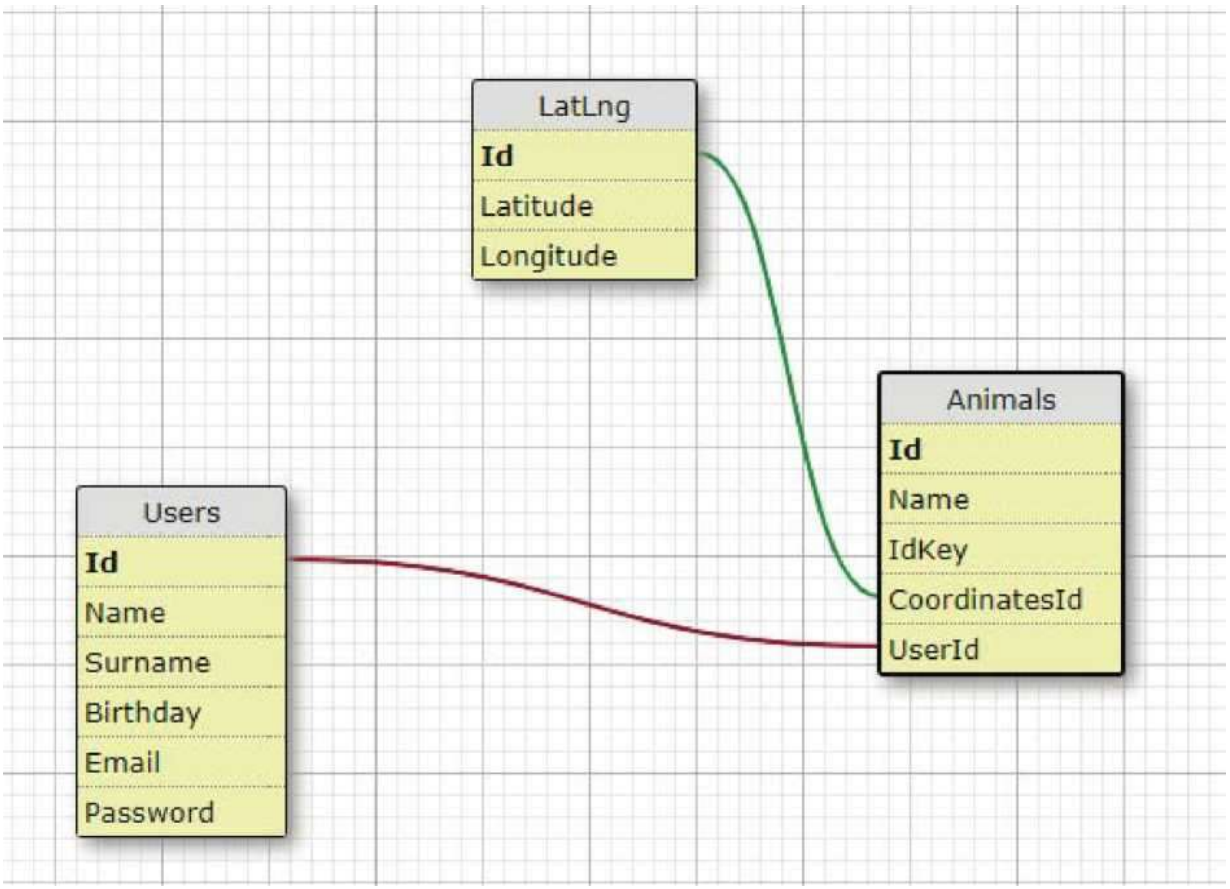

Рисунок 2.4 – Схема бази даних

Оскільки в якості СУБД була використана SQLite, ця система підтримує не всі типи даних, тому нам необхідно вказати, які типи даних ми використовуємо:

- INTEGER - значення є ціле число зі знаком;
- TEXT - значення є текстовий рядок, що зберігається з використанням кодування бази даних. Оскільки в SQLite немає спеціального типу, для дати й часу відповідних полів можна використовувати тип даних text;
- REAL - значення являє собою значення з плаваючою комою.

База даних складається із трьох таблиць. У таблиці 2.3 подано їх короткий опис.

Таблиця 2.3 – Опис таблиць бази даних

Таблиця	Опис
User	містить інформацію про користувача
Animal	містить інформацію про тварину
LatLong	містить координати тварини

У таблицях 2.4 - 2.6 подано опис полів таблиці User, Animal, LatLong.

Таблиця 2.4 - Опис полів таблиці User

Поле	Тип даних	Опис
Id (первинний ключ)	INTEGER	унікальний ідентифікатор користувача
Name	TEXT	ім'я
Surname	TEXT	прізвище
Birthday	TEXT	дата народження
E-mail	TEXT	e-mail користувача, вказаний при реєстрації
Password	TEXT	пароль, вказаний під час реєстрації

Таблиця 2.5 – Опис полів таблиці Animal

Поле	Тип даних	Опис
Id (первинний ключ)	INTEGER	Унікальний ідентифікатор тварини
Name	TEXT	кличка тварини
IdKey	INTEGER	ідентифікатор нашийника, який буде вказано в документації до нашийника
CoordinatesID	INTEGER	(Зовнішній ключ, посилається на таблицю LatLong) ідентифікатор координат тварини.
Userid	INTEGER	(Зовнішній ключ, посилається на таблицю User) ідентифікатор господаря тварини.

Таблиця 2.6 – Опис полів таблиці LatLong

Поле	Тип даних	Опис
Id (первинний ключ)	INTEGER	унікальний ідентифікатор координат тварини
Lat	REAL	широта
Long	REAL	довгота

2.4 Функціональна схема апаратної частини системи

Розглянемо характеристики та можливості датчиків та модулів, які використовуються для проектування апаратної частини комплексу для відстеження розташування тварини.

Функціональна схема взаємодії елементів системи представлена рисунку 2.5.

Трекер складатиметься з:

- мікроконтролера;
- GNSS/GSM модуля;
- джерела живлення.

На контролер надходять дані з модуля GPS у яких міститься інформація про місцезнаходження тварини. За допомогою GSM модуля контролер взаємодіє з сервером, як на відправлення даних, так і на прийом. Для передачі даних необхідно підключити SIM- карту та антени. За допомогою джерела живлення (акумулятора) пристрій отримує необхідну напругу.

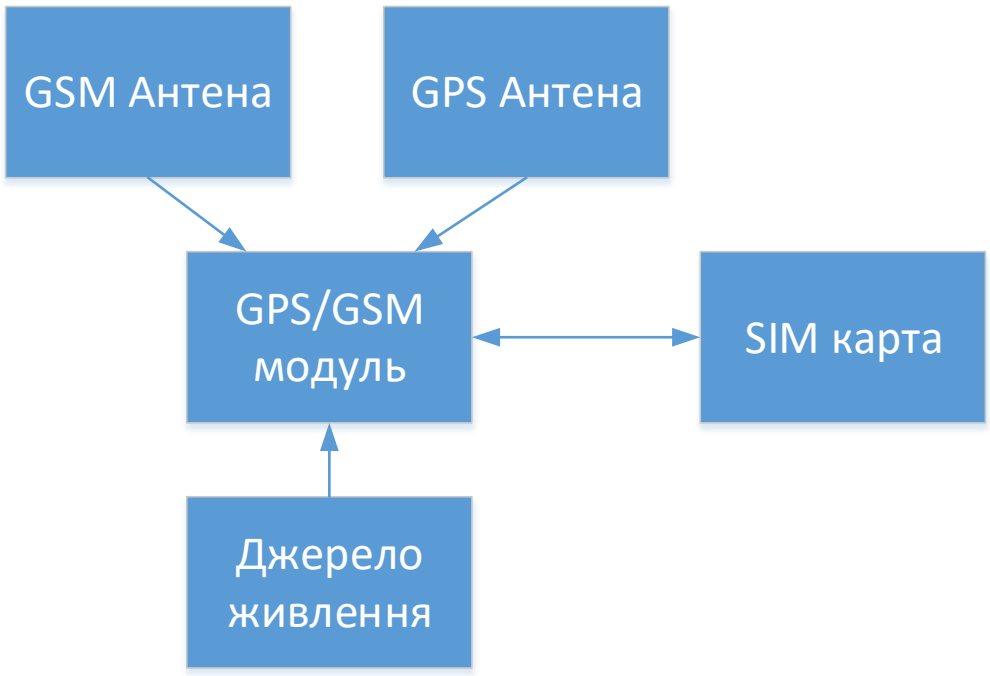

Рисунок 2.5 - Функціональна схема трекера

2.5 Програмна реалізація серверної частини

В основі серверної частини нашої клієнт-серверної програми лежить фреймворк ASP.NET MVC.

Суть цієї архітектури зводиться до поділу моделі даних, інтерфейсу користувача і керуючої логіки контролера. Програма серверної частини складається з:

- Model (модель) - це частина, яка містить у собі функціональну бізнес-логіку програми. Містить або подає дані, з якими працюють користувачі;
- Controller (контролер) - обробляє запити, що надходять, виконує операції з моделлю і вибирає подання для візуалізації користувачеві.

Так як на сервері немає необхідності у візуальному представленні на даний момент View не було реалізовано.

Взаємодія між ними відбувається за допомогою запитів HTTP.

У корені програми знаходяться чотири основні папки:

1) Controllers - дана папка містить класи, які використовуються для обробки запитів, що надходять. Контролер є центральним компонентом архітектури MVC. Вона використовуються для організації логіки запитів, що надходять. За угодами про назву назви контролерів повинні закінчуватися на суфікс "Controller", решта ж до цього суфікса вважається ім'ям контролера. Ця папка містить 2 класи: Accountcontroller, AnimalLocationController. Контролери в цих папках реагують на вхідні запити HTTP і обробляють їх. Слід зазначити, кожен метод контролера відповідає за один HTTP запит. Адреса запиту, який має реагувати кожен метод прописується в атрибуті “Route”. За допомогою атрибутів [HttpGet], [HttpPost], [HttpDelete] або [HttpPut] визначається тип запиту, що обробляється.

Опис розроблених контролерів:

- Accountcontroller - цей контролер визначає реакцію сервера на вхідні

запити під час роботи з обліковими записами користувача. Вихідний код цього контролера представлений у лістингу Б.1.

- AnimalLocationController - даний контролер визначає реакцію сервера на вхідні запити під час роботи з координатами тварини Вихідний код даного контролера представлений листингу Б.2.

У таблиці 2.7 і 2.8 наведено опис методів класу Accountcontroller і AnimalLocationController.

Таблиця 2.7 – Опис методів класу Accountcontroller»

Метод	Опис
Login	цей метод використовується, для обробки інформації користувачеві під час авторизації.
Registration	цей метод використовується, для обробки інформації користувачеві під час реєстрації.
GetIdentity	цей метод викликається якщо користувач був знайдений, то створюється об'єкт ClaimsIdentity

Таблиця 2.8 - Опис методів класу AnimalLocationController

Метод	Опис
Put	цей метод використовується для надсилання запиту на отримання координат тварини
Get	цей метод використовується для отримання координат тварини
FindAnimal	цей метод використовується при визначенні розташування тварини

2) Models – містяться класи опису даних, що використовуються у додатку. Моделі зберігають лише стан як властивостей. Ця папка містить наступні папки: AnimalLocation, Database

AnimalLocation - ця папка використовується для роботи з координатами тварини. Містить два класи:

- ModelForGettingAnimalLocation - цей клас визначає ряд властивостей, які описують інформацію для отримання координат тварини. Містить поля, що містять email користувача та ідентифікатор пристрою. Вихідний код цього

класу представлений у лістингу Б.3.

- ModelForPutAnimalLocation - цей клас визначає ряд властивостей, які описують інформацію для відправки запиту на отримання координат тварини. Містить поля, що містять email користувача, ідентифікатор пристрою та координати тварини. Вихідний код цього класу представлений у лістингу Б.4.

Database - ця папка містить інформацію про те, що зберігається в базі даних, а також функціональну бізнес-логіку програми. Містить такі класи:

- Animal - даний клас визначає ряд властивостей, що описують інформацію про тварини: унікальний ідентифікатор, ім'я, ідентифікатор нашийника та ідентифікатор координат тварини.

- DatabaseContext - використовується як проміжний шар між додатком і базою даних. Клас DatabaseContext успадковується від абстрактного класу DbContext, який визначено всередині Entity Framework та надає широкі можливості для роботи з базою даних. Також у класі DataContext є два поля: Users, Animals, які інтерпретуються до бази даних як таблиці, і навіть Entity Framework перебирає роботу зі створенням зв'язків між таблицями. Вихідний код цього класу представлений у лістингу Б.5.

- LatLng - цей клас визначає ряд властивостей, які описують координати: широта та довгота. Також цей клас містить метод Update для оновлення координат тварини.

- User - даний файл визначає ряд властивостей, які описують інформацію про користувача: унікальний ідентифікатор, ім'я, прізвище, день народження, логін та пароль та унікальний ідентифікатор тварини. Також цей клас містить клас UserComparer, який застосовується для порівняння користувачів. Вихідний код цього класу представлений у лістингу Б.6.

Також існує 3 класи:

- Accountinfo - цей клас визначає ряд властивостей, які описують інформацію про обліковий запис користувача: ім'я, прізвище, маркер доступу, інформацію про тварини. Вихідний код цього класу представлений у лістингу

Б.7.

- RegistrationModel - цей клас визначає ряд властивостей, які описують інформацію необхідну під час реєстрації користувача: унікальний ідентифікатор, ім'я, прізвище, день народження, логін та пароль, а також інформацію про тварину. Ця модель зберігає лише стан як властивостей. Також існує метод ToUser(), який використовується для додавання користувача до бази даних. Вихідний код цього класу представлений у лістингу Б.8.

- LoginModel - цей клас визначає ряд властивостей, які описують інформацію необхідну при авторизації користувача: логін та пароль. Ця модель зберігає лише стан як властивостей. Вихідний код цього класу представлений у лістингу Б.9.

3) Authenticate – містяться класи, які описують методи для авторизації. Механізм авторизації використовує JWT -токени. Таким чином здійснюється передача даних про користувача у форматі JSON у зашифрованому вигляді. Ця папка містить 2 класи:

- AuthOptions – клас що визначає характеристики токена для авторизації. Містить 4 константи. Константа Issuer видавець токена. Назву можна визначити любу. Audience представляє користувача токена - вказана адреса поточної програми. Константа Lifetime визначає час життя токена. Для створення токена використано константу Key, яка зберігає ключ. Вихідний код даного класу представлений у лістингу Б.10.

- TokenGeneration – клас, який генерує спеціальний токен для авторизації. Містить метод GetToken, що генерує токен. Тут створюється JWT – токен і відбувається ініціалізація констант із класу AuthOptions. Вихідний код цього класу представлений у лістингу Б.11.

4) Migration – дозволяє вносити зміни до бази даних при змінах моделей та контексту даних. Папка містить два файли: Configration.cs та файл початкової міграції. Перший файл містить оголошення однойменного класу

Configuration, який встановлює конфігураційні налаштування Файл початкової міграції встановлює, як база даних визначається на даний момент. Вихідні коди даних класів представлені у лістингу Б.12 та Б.13.

Також реалізовано 2 класи Startup та Program.

- Startup – клас, який визначає налаштування веб сервісу. Є вхідною точкою до ASP.NET Core. У даному класі є конструктор і два методи: ConfigureServices та Configure. Метод ConfigureServices реєструє сервери, які використовуються програмою та відповідає за отримання рядка підключення до бази даних із конфігураційного файлу. У методі Configure встановлюються параметри видачі вікна помилок, також у цьому методі налаштовується фільтр запитів, на які веб-сервер повинен відповідати. Всі запити, які не потрапляють до цього фільтра, будуть відхилятися сервером. Вихідний код цього класу представлений у лістингу Б.14.

- Program - в даному класі створюється об'єкт IWebHost в рамках якого розгортається веб-додаток. Для створення IWebHost використовується об'єкт IWebHostBuilder. Цей об'єкт створюється за допомогою методу CreateWebHostBuilder(args). Цей метод виконує ряд завдань, таких як налаштування вузла, налаштування веб-сервісу Kestrel, завдання імені хоста та визначення головного файлу програми. Після цього в методі Main відбувається створення хоста IWebHost за допомогою методу Build(), а потім відбувається запуск за допомогою методу Run. Вихідний код цього класу представлений у лістингу Б.15.

2.6 Програмна реалізація клієнтської частини

В основі клієнтської частини лежить уявлення MVP. На рисунку 2.6 показана схема взаємодії елементів MVP.

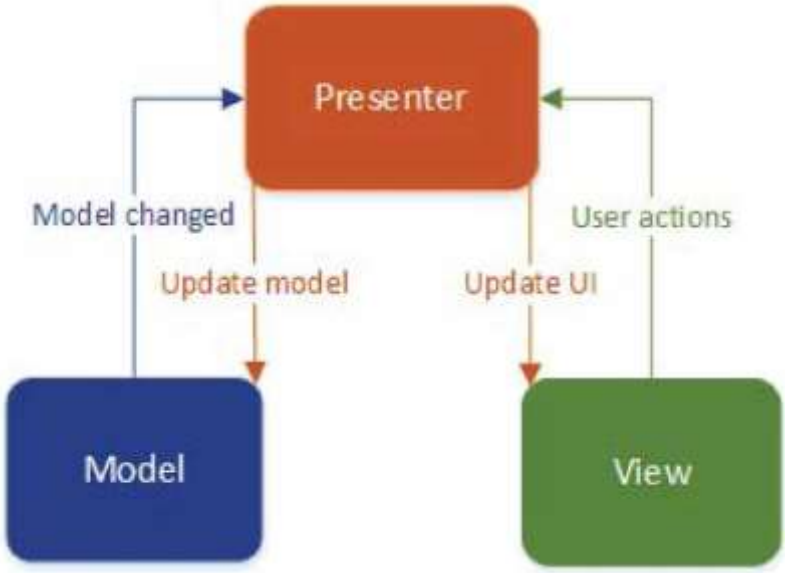

Рисунок 2.6 - Подання MVP

MVP розшифровується як Model-View-Presenter.

– Модель (Model) - зберігає у собі всю бізнес-логіку, за необхідності отримує дані зі сховища.

– Вигляд (View) – реалізує відображення даних (з Моделі) та звертається до Presenter за оновленнями. Як View може бути Activity.

– Представник (Presenter) – реалізує взаємодію між Моделлю та Видом.

Важливо пам'ятати, що безпосередньо View і Model не взаємодіють лише через Presenter.

У корені програми містяться такі папки:

1) Activities – містить класи, що використовуються для інформації. Містить 2 класи:

-MapActivity – клас для роботи з картами. Містить методи, які використовуються для роботи з картами. Вихідний код цього класу представлений у лістингу В.1.

-RootActivity – цей клас містить загальні методи для всіх активностей. Вихідний код цього класу представлений у лістингу В.2.

У таблицях 2.9-2.10 наведено опис методів класів MapActivity і RootActivity.

Таблиця 2.9 – Опис методів класу MapActivity

Метод	Опис
OnCreate	даний метод задає початкову установку параметрів при ініціалізації активності
OnStart	у цьому методі відбувається ініціалізація дій видимих користувачеві
OnDestroy	у цьому методі відбувається очищення всіх ресурсів
OnMapReady	цей метод запускається, коли карта готова до роботи
onRequestPermissionsResult	цей метод використовується для обробки результатів дозволів програми
checkPermission	даний метод використовується для отримання дозволів програми
initialMap	у цьому методі відбувається ініціалізація картки
setInitialFocusOnMap	даний метод використовується для фокусування карт певному місці
startUserLocationUpdate	даний метод використовується для визначення розташування користувача
getAnimalCoordinates	метод, який використовується для отримання координат тварини
showMessage	метод для виведення повідомлень

Таблиця 2.10 – Опис методів класу RootActivity

Метод	Опис
onCreate	даний метод задає початкову установку параметрів при ініціалізації активності
showProgress	даний метод відображає індикатор виконання
hideProgress	цей метод приховує індикатор виконання

2) Fragment – використовується для обробки інформації. Містить папку Presenters - ця папка містить класи, які обробляють інформацію, введену користувачем. Містить класи:

- LoginFragmentPresenter - даний клас містить методи для обробки інформації, що вводиться про користувача при авторизації. Вихідний код

цього класу представлений у лістингу В.3.

- RegistrationFragmentPresenter - даний клас містить методи для обробки інформації, що вводиться про користувача при реєстрації. Вихідний код цього класу представлений у лістингу В.4.

- Presenter – цей клас використовується для обробки загальних методів. Вихідний код цього класу представлений у лістингу В.5.

У таблицях 2.11 - 2.13 наведено опис методів класу LoginFragmentPresenter, RegistrationFragmentPresenter і Presenter

Таблиця 2.11 - Опис методів класу LoginFragmentPresenter

Метод	Опис методу
CheckAuthentication	цей метод використовується для автентифікації користувача
redirectT oRegistration	цей метод використовується для перенаправлення користувача на сторінку реєстрації, якщо користувач не знайдено
cancelAuth	цей метод викликається, якщо авторизація не була пройдена
Login	цей метод використовується для авторизації
onComplete	цей метод викликається при успішному завершенні авторизації
onFail	цей метод викликається, якщо авторизація завершена з помилкою
onFailRequest	цей метод викликається при помилці підключення до сервера

Таблиця 2.12 - Опис методів класу RegistrationFragmentPresenter

Метод	Опис методу
Registration	цей метод використовується для реєстрації
validateF ields	цей метод використовується для перевірки введення даних та їх коректності
onComplete	цей метод викликається при успішному завершенні реєстрації
onFail	цей метод викликається, якщо реєстрація завершена з помилкою
onFailRequest	цей метод викликається при помилці підключення до сервера

Таблиця 2.13 – Опис методів класу Presenter

Метод	Опис методу
startUpdateMap	у цьому методі задаються умови для оновлення картки
cancelUpdate	цей метод використовується при скасуванні оновлення картки
getInstanceT imerT ask	метод для отримання екземпляра класу TimerT ask
updateMap	метод, який використовується для оновлення картки

Також містить 3 класи:

- LoginFragment - цей клас використовується для обробки інформації при авторизації користувача. Вихідний код цього класу представлений у лістингу В.6.

- RegistrationFragment - цей клас використовується для обробки інформації під час реєстрації користувача. Даний клас містить метод initialViews, який використовується для опису реакції на натискання на кнопку, методи для отримання інформації про користувача та тварину, а також обробники помилок. Вихідний код цього класу представлений у лістингу В.7;

- RootFragment – цей клас використовується для обробки загальних методів. Містить метод переходу на карту. Вихідний код цього класу представлений у лістингу В.8.

У таблиці 2.14 наведено опис методів класу LoginFragment.

Таблиця 2.14 - Опис методів класу LoginFragment

Метод	Опис методу
OnCreate	даний метод задає початкову установку параметрів при ініціалізації активності
getEmail	метод для отримання email
getPassword	метод для отримання пароля
initialViews	цей метод використовується для опису реакції на натискання на кнопку
showProgress	даний метод відображає індикатор виконання
hideProgress	цей метод приховує індикатор виконання

3) Models – папка містить класи для роботи з даними. Містить папку

Apiclient – використовується для обробки запитів. Містить 4 папки:

- animal.location - містить класи UpdateAnimalLocation і UpdateAnimalLocationModel, які використовуються для отримання логіну та ідентифікатора нашийника, що використовуються під час оновлення розташування. А також клас UpdateAnimalLocationTask, який використовується для надсилання запитів. Код класу UpdateAnimalLocationTask представлений у лістингу В.9;
- authentication - містять містить класи Authenticator і AuthResponse, які використовуються для отримання даних, що використовуються для авторизації. А також клас LoginTask, який використовується для надсилання запитів. Код класу LoginTask представлений у лістингу В.10;
- Enities – містить існуючі сутності. Є 2 класи User та Animal. Дані класи містять методи, які використовуються для отримання інформації про користувача та тварини;
- Registration - містять містить клас ApiRegistration, який використовується для отримання даних,використовуються для реєстрації. А також клас RegistrationTask, який використовується для надсилання запитів. Код класу RegistrationTask представлений у лістингу В.11;

Також ця папка містить 3 класи:

- AccountInfo - цей клас містить методи, які використовуються для отримання інформації про користувача;
- ApiClient - цей клас створений для зв'язку клієнтської програми з сервером, тут відбувається налаштування параметрів для підключення до хостингу та реалізовані методи, які необхідні для надсилання HTTP запитів до веб-сервера. Код цього класу представлений у лістингу В.12;
- AuthResult - цей клас використовується для отримання результатів авторизації користувача. Код цього класу представлений у В.13.

Також містить 2 класи:

1) Setting - цей клас містять методи, які визначають методи для

налаштувань. Код цього класу представлений у лістингу В.14.

2) Validator - цей клас містить методи для валідації даних, що вводяться користувачем (логін, пароль та дата народження). Код цього класу представлений у лістингу В.15.

У таблиці 2.15 наведено опис методів класу Setting.

Таблиця 2.15 – Опис методів класу Setting

Метод	Опис методу
saveAccountlnfo	цей метод застосовується для збереження інформації про користувача
getAccessToken	цей метод використовується для отримання стану підключення
isLogin	метод, який викликається при успішній авторизації користувача

За допомогою XML розмітки реалізовано інтерфейс сторінок авторизації, реєстрації та картки клієнтської програми. Даний спосіб допомагає створити зручний інтерфейс з кодом, що добре обслуговується. Коди XML представлені у додатку Г.

2.7 Висновок до розділу

Для розробки нашого проекту було обрано такі програмні компоненти:

- СУБД: SQLite.
- Мови програмування: C# та Java.
- Середовище розробки серверної частини Visual Studio.
- Середовище розробки клієнтської частини: Android Studio.
- Карти - Google maps.

В апаратній частині було обрано модуль

- GNSS/GPRS модуль Quectel MC60.
- Li-Pol 803050.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Апаратна частина

В першу чергу необхідно підключити батарею до 1 та 2 ніжки модуля, а PWKEY підключити до землі. GSM антена була підключена в гніздо J102, а GNSS антена в гніздо J101. Перемикачі S101 і S102 , які використовуються для включення GSM і GPS, повинні бути переведені в положення ON. Для зручності була припаяна кнопка, яка використовуватиметься для увімкнення та вимикання пристрою.

На рисунку 3.1 показано макетний зразок трекера. На рисунку К.1 представлена електрична схема обв'язування модуля.

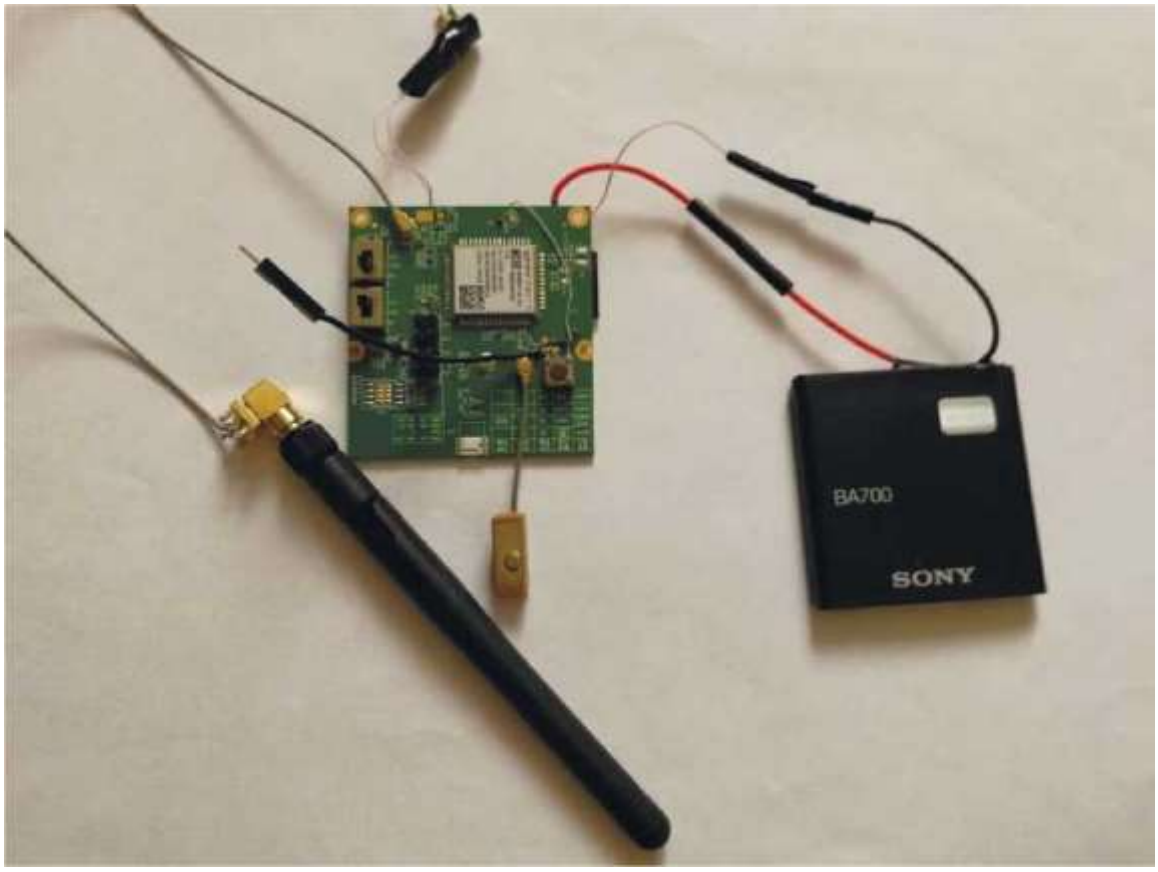

Рисунок 3.1 – Макетний зразок трекера

Також для функціонування GSM частини необхідно вставити SIM - картку. На рисунку 3.2 показано зворотний бік модуля, на якому видно карту, а також видно до яких ніжок під'єднані проводи, що використовуються.

Рисунок 3.2 - Зворотний бік модуля

Для керування модулем та налаштування режиму роботи будуть використовуватись AT-команди. Для введення команд AT використовується термінальна програма.

Необхідно з'єднати модуль із UART використовуючи TX та RX. На рисунку 3.3 зображено схему підключення модуля до UART.

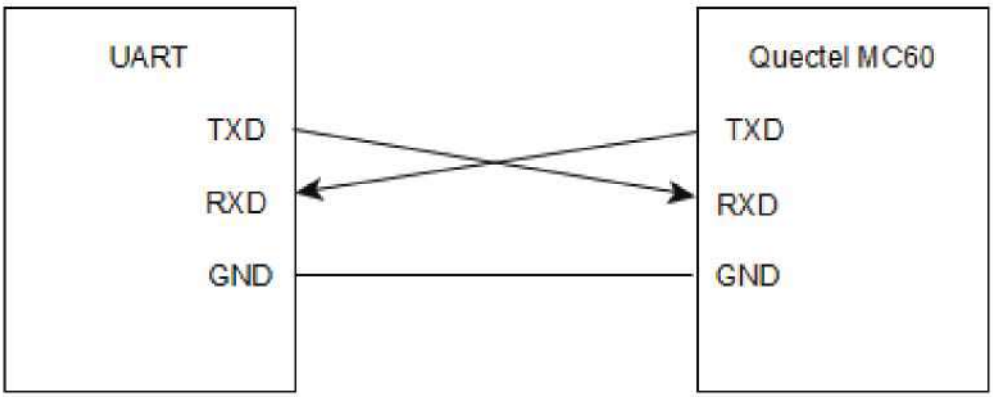

Рисунок 3.3 – Схема підключення модуля

На рисунку 3.4 показано підключення модуля Quectel MC60 до UART.

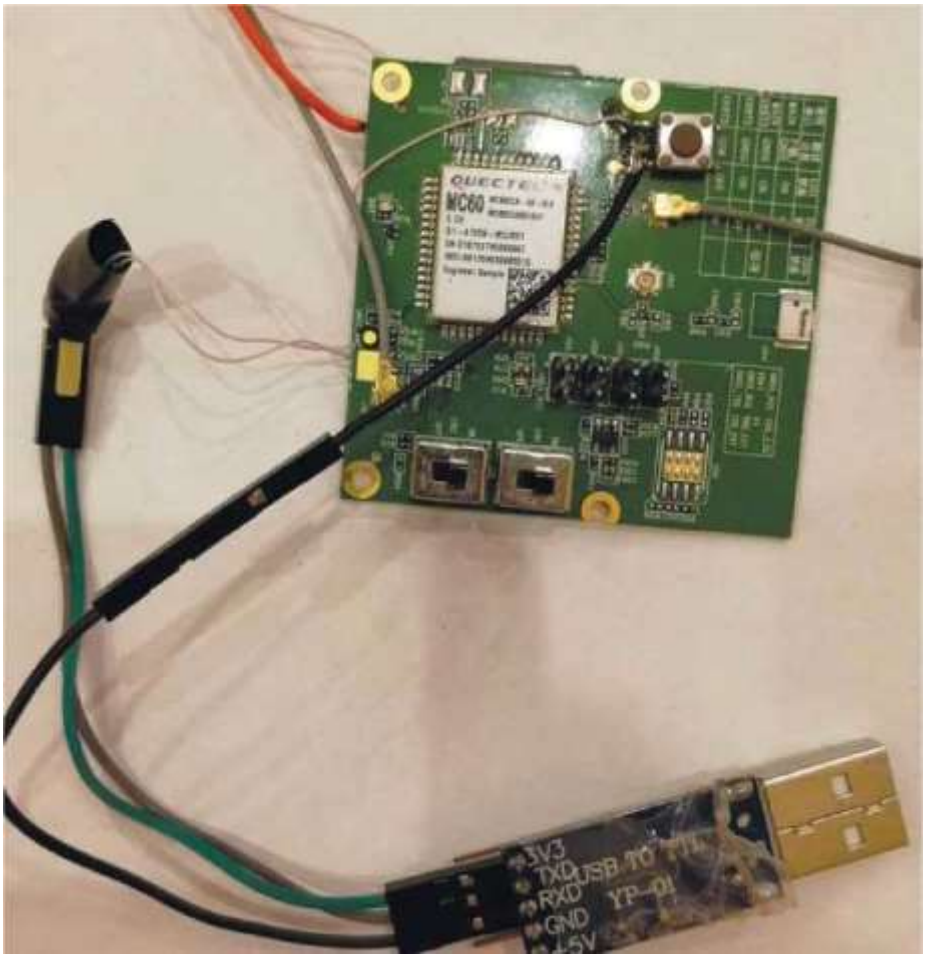

Рисунок 3.4 – Підключення модуля

Для початку необхідно налаштувати швидкість роботи модуля, вона дорівнює 115 200. Далі потрібно відправити тестову команду AT, якщо модуль відповів її у ОК, він готовий до роботи. Нижче розглянемо докладніше основні команди, які були використані під час роботи з даним модулем.

1) Загальні команди:

- AT+GMI – ця команда поверне три рядки. Перший рядок – це інформація про компанію-виробника, другий – це ідентифікатор модуля, а третій містить версію прошивки.
- AT+GSN - ця команда повертає IMEI модуля.
- AT+GCAP – ця команда повертає можливості модуля.
- AT+CCLK - ця команда повертає поточну дату та час пристрою.
- AT+CBC – ця команда використовується для моніторингу напруги живлення модуля. Повертає три параметри. Перший показує чи заряджається модуль, другий параметр - це рівень заряду, а третій напруга живлення модуля. Повний перелік команд з отриманими результатами наведено в додатку Д.

2) GSM команди модуля:

- AT+QSIMDET - ця команда використовується для увімкнення режиму детектування наявності SIM -картки. Повертає три параметри. Перший означає, що режим детектування карт вимкнено.
- AT+CPIN=XXXX - за необхідності за допомогою цієї команди необхідно ввести PIN- код при включенні модуля.
- AT+ COPS – ця команда, яка виводить інформацію про оператора.
- AT+GSN – ця команда використовується для отримання IMEI.
- AT+CPAS – ця команда містить інформацію про стан модуля.
- AT+CREG - ця команда містить інформацію про тип реєстрації у мережі. Повертає 2 параметри. Показують наявність коду реєстрації та тип реєстрації.
- AT+CSQ – це команда містить інформацію про рівень сигналу.

Також за допомогою AT -команд можна дзвонити та надсилати СМС повідомлення, розглянемо основні команди для цього:

- ATD +80xxxxxxx; - команда використовується для дзвінка. Номер вводиться без пробілів та точок.

- ATA – ця команда використовується для відповіді на вхідний дзвінок.

- ATH – ця команда використовується для відключення виклику.

- AT+CMGS - ця команда використовується для надсилання СМС повідомлень. Необхідно вказати номер у лапках, а потім ввести тест повідомлення. Після закінчення введення необхідно ввести ctrl+z для надсилання повідомлення або esc для скасування.

- AT+CMGR - ця команда використовується для читання вхідних SMS повідомлень.

У додатку Е представлений повний перелік команд і відповідей модуля, що використовуються.

3) GNSS команди модуля:

- AT+QGNSSC - ця команда використовується для керування живленням GNSS.

- AT+QGNSSRD – ця команда використовується для отримання навігаційної інформації GNSS.

- AT+QGNSSTS – ця команда використовується для отримання статусу синхронізації часу для модуля GNSS.

- AT+QGNSSEPO - ця команда використовується для увімкнення або вимкнення функції EPO (A-GPS).

- AT+QGEPOAID – ця команда використовується для запуску функції EPO.

Повний перелік команд із отриманими результатами представлений у додатку Ж.

4) GPRS команди модуля:

- AT+CGATT – ця команда використовується для підключення модуля до GPRS.

- AT+CGPADDR – ця команда повертає IP адресу контексту PDP

(PDP – це протокол пакетних даних).

- AT+QICSGP - ця команда використовується для налаштування APN (APN – ідентифікатор мережі пакетної передачі даних).
- AT+QIFGCNT – ця команда використовується для активації GPRS.
- AT+QGSMLOC - ця команда повертає 2 параметри. Перший містить інформацію про розташування пристрою, отриману за допомогою GSM, а другий - це точні дата та час мережі.

Повний перелік команд з отриманими результатами наведено в додатку З.

5) Команди передачі даних:

- AT+QIOPEN - ця команда використовується для відкриття TCP/UDP підключення. На вхід подається 3 параметри: тип підключення, IP-адреса і порт. У відповідь повертаються інформація про успішність підключення.
- AT+QISEND - дана команда використовується передачі повідомлень. Після закінчення введення необхідно ввести ctrl+z для надсилання повідомлення або esc для скасування.
- AT+QICLOSE - ця команда використовується для закриття TCP/UDP підключення. Повний перелік команд з отриманими результатами наведено у додатку И.

3.2 Візуальне подання реалізації

Форма реєстрації користувача.

У формі реєстрації користувача необхідно заповнити усі поля. Необхідно вказати ім'я, прізвище, дату народження, кличка тварини та ідентифікатор нашийника тварини, а також придумати логін та пароль. Дата народження та ідентифікатор пристрою мають бути цифровими значеннями. Також необхідно забезпечити підключення телефону до Інтернету для надсилання координат на сервер. Скріншот форми реєстрації користувача

представлений рисунку 3.5.

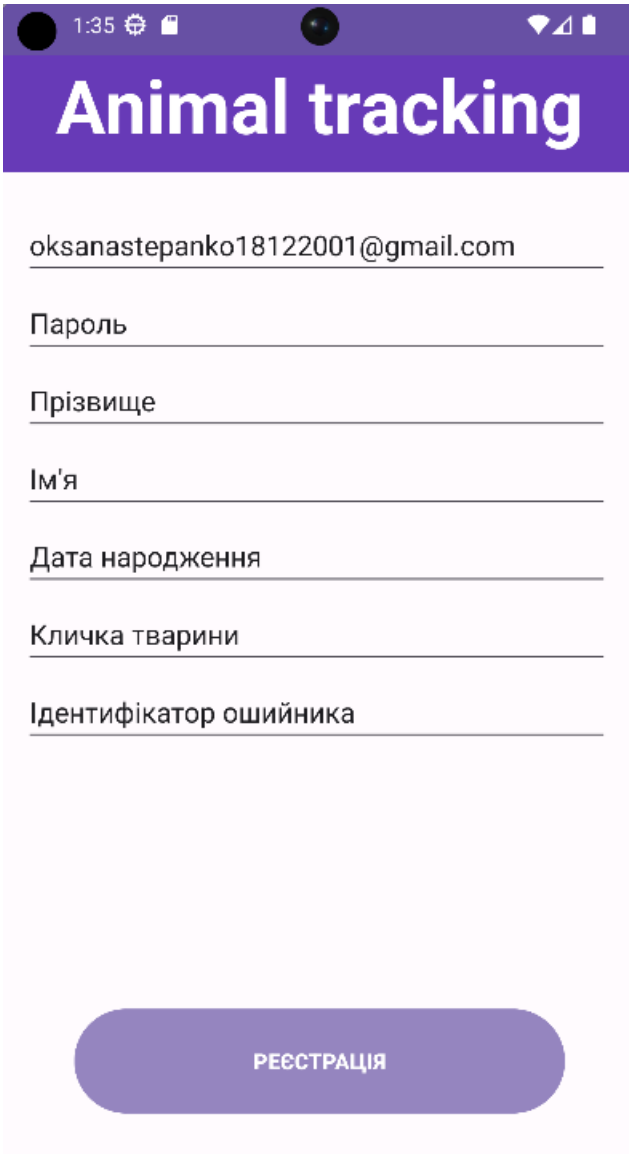

Рисунок 3.5 - Форма реєстрації користувача

2. Форма авторизації користувача

У формі авторизації користувача необхідно заповнити усі поля. Вказати ім'я користувача та пароль, які були використані раніше під час реєстрації. Якщо користувач не був зареєстрований, необхідно повернутися на сторінку реєстрації. Також необхідно забезпечити підключення телефону до Інтернету для звіряння даних. Скріншот форми авторизації користувача представлений рисунку 3.6.

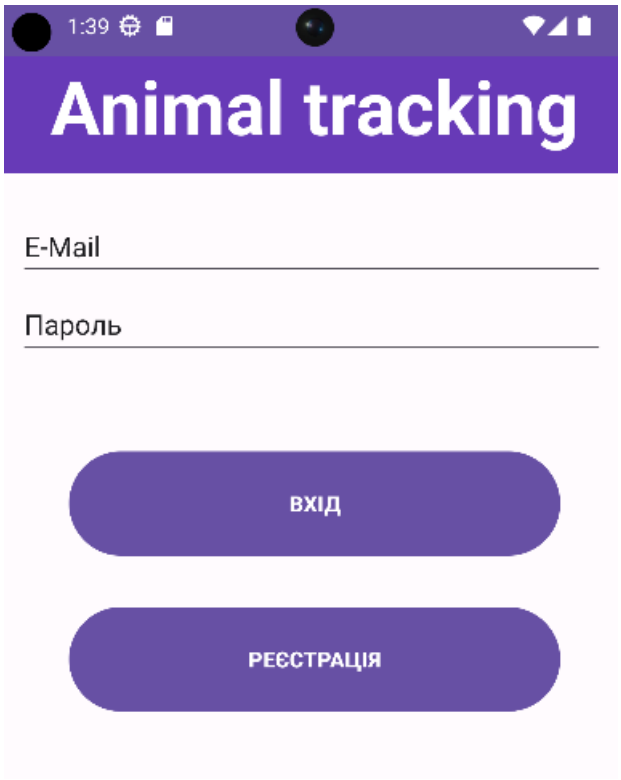

Рисунок 3.6 - Форма авторизації користувача

3. Карта

Після переходу на карту запитаються про дозвіл на доступ програми до даних про місцезнаходження пристрою і на карті відображається місцезнаходження користувача та тварини.

3.3 Тестування системи

При введенні даних необхідно переконатися заповнення всіх полів. Якщо якісь поля не були заповнені, користувач бачить попередження про необхідність заповнення поля і не може пройти реєстрацію.

Ідентифікатор пристрою та дата народження мають бути числовими значеннями. Під час введення відображається лише клавіатура. На рисунку 3.7 відображено поле введення ідентифікатора.

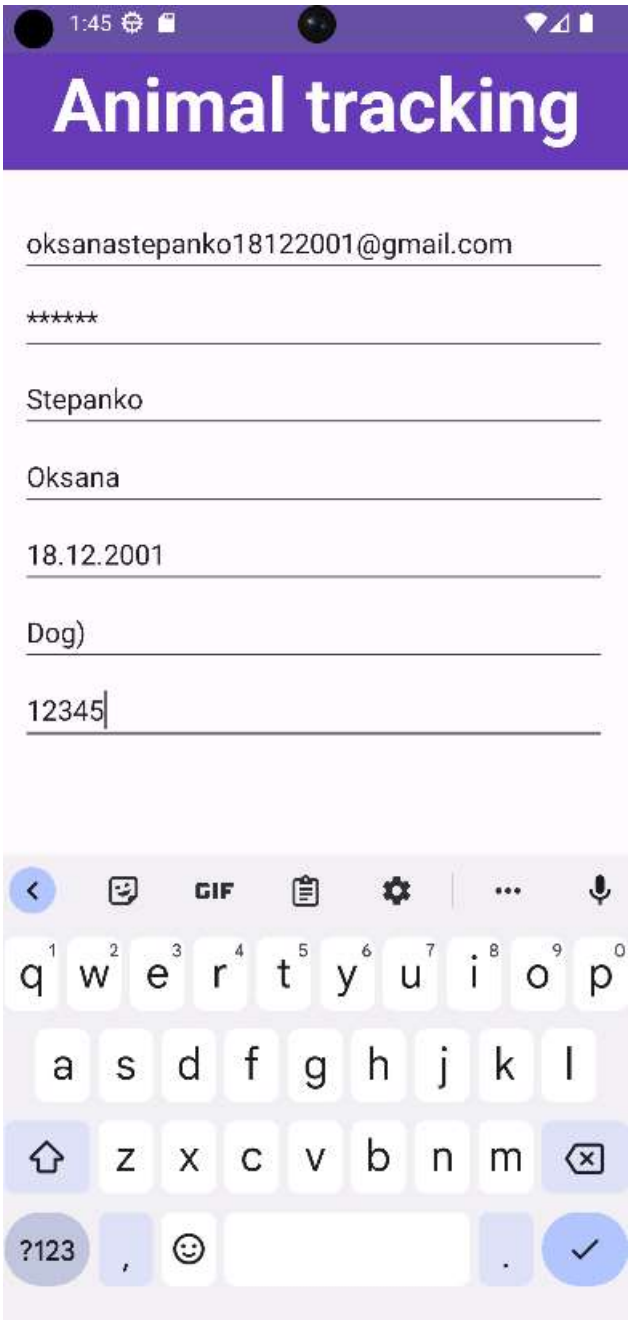

Рисунок 3.7 – Поле введення ідентифікатора

Після завершення введення даних необхідно переконатися в підключенні телефону до Інтернету, щоб надіслати програму даних про реєстрацію користувача на сервер.

Для визначення координат пристрою використовуються AT-команди QGREFLOC та QGNSSRD. На рисунках 3.8, 3.9 подано результат виконання цієї команди.

```
AT+QGSMLOC=1
+QGSMLOC: 49.551472, 25.632476,2023/05/30,15:39:33
```

Рисунок 3.8 – Лістинг визначення розташування за GSM

```
AT+QGNSSRD?
+QGNSSRD:
$GNRMC,144641.000,A,5507.2813,N,06122.3096,E,0.77,221.28,17061
9,,, A*75
$GNVTG,221.28,T,,M,0.77,N,1.43,K,A*2E
$GNGGA,144641.000,5507.2813,N,06122.3096,E,1,4,3.16,240.1,M,-
11.6,M,,*68
$GPGSA,A,3,15,13,27,,,,,,,,,3.31,3.16,0.98*05
$GLGSA,A,3,68,,,,,,,,,,,,3.31,3.16,0.98*14
$GPGSV,3,1,09,10,79,219,,21,66,128,,10,50,237,,27,47,291,17*74
$GPGSV,3,2,09,15,43,071,30,13,21,041,20,08,19,319,,
16,14,253,*7C
$GPGSV,3,3,09,20,01,356,*42
$GLGSV,3,1,10,83,86,211,,69,67,199,,68,55,049,17,85,31,317,*66
$GLGSV,3,2,10,83,27,143,,76,14,339,, 77,10,037,,70,10,216,*66
$GLGSV,3,3,10,67,04,039,,75,03,300,*69
$GNGLL, $GNGLL,5500.2800,N,06100.3000,E,14400.,A,A*41
```

Рисунок 3.9 – Лістинг визначення розташування GPS

Крім координат були визначені точні дата і час мережі. Після отримання даних про місцезнаходження необхідно надіслати ці дані на сервер, використовуючи 2 команди: QIOPEN для підключення модуля до сервера і QISEND для надсилання запиту на сервер. На рисунку 3.10 показано результат виконання даних команд.

```
AT+QIOPEN="TCP","80.78.248.203",80
OK
CONNECT OK
AT+QISEND
> POST /api/animalloc/put HTTP/1.1
HOST: 80.78.248.203
Accept: application/json
Content-Type: application/json
Content-Length: 104
{"email":"oksanastepanko18122001@gmail.com","idKey":"11111",
"Coordinates":{"latitude": 49.551472,"Long": 25.632476}}
```

Рисунок 3.10 – Лістинг надсилання повідомлення на сервер

Необхідно переконатися в додаванні координат таблицю, що містить координати пристрою.

Після цього необхідно переконатися у правильності відображення розташування на карті. Під час переходу на карту потрібно дати дозвіл на доступ програми до даних про місцезнаходження пристрою.

На рисунку 3.11 показано відображення розташування тварини (червона мітка) та користувача (синя мітка).

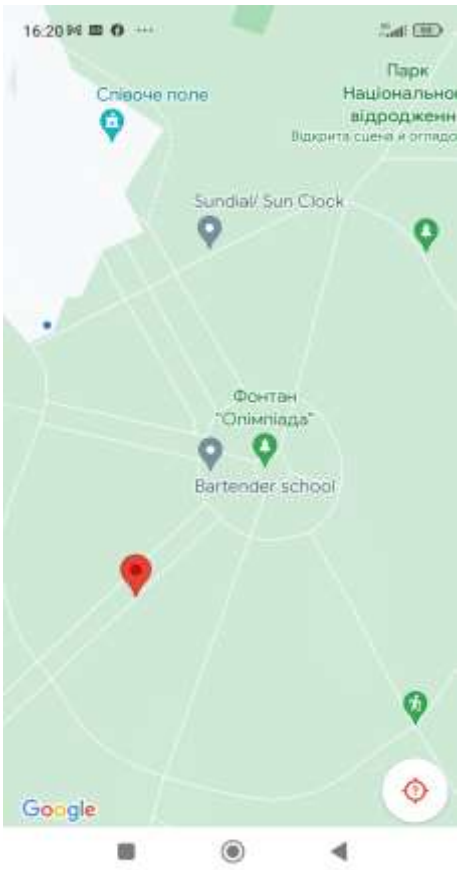

Рисунок 3.11 - Відображення міток

При натисканні на червону мітку Google карти дають можливість побудувати шлях до точки призначення. На рисунку 3.12 показано прокладені можливі шляхи до точки призначення.

Рисунок 3.12 - Визначення мінімального шляху до тварини

Таким чином можна швидко знайти домашню тварину. Єдиний недолік що маршрут будується до останньої точки де були зафіксовані координати. Для оновлення координати на мапі необхідно оновити.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні чинники небезпеки

Так як дана робота присвячена розробці системи відстеження домашніх тварин тому в цьому розділі доцільно проаналізувати психологічні чинники небезпеки під високим рівнем стресу та психофізичного навантаження.

На основі аналізу статистичних даних та висновків експертів у галузі безпеки життєдіяльності можна стверджувати, що від 60 до 90% травм на виробництві і в побуті відбувається з вини потерпілих [15].

Основні причини цього:

– низький рівень професійної підготовки з питань безпеки;

– недостатнє виховання;

– слабка установка людини на дотримання вимог безпеки;

– допуск до небезпечних робіт осіб з підвищеним ризиком травматизму;

– перебування людей у стані втоми чи інших психічних станах, які знижують безпеку діяльності.

Виділяють комплекс чинників, що збільшують індивідуальну схильність людини до небезпеки, це:

– особливості темпераменту;

– функціональні зміни в організмі;

– дефекти органів відчуття;

– незадоволення даним видом діяльності.

Водночас схильність до небезпеки через несприятливий характер діяльності:

– значні фізичні та розумові зусилля;

– незручна робоча поза; – високий темп праці;

– нервово-емоційні перевантаження;

– перенапруга слухових та зорових аналізаторів;

– несумісність робочого місця, засобів праці та антропометричних даних людини.

Зазначені фактори призводять до підвищеної фізичної та нервової втоми, яка послаблює психіку, знижує швидкість та точність орієнтації, притупляє пильність та увагу, порушує сприйняття навколишньої ситуації. Це також спричинює травматизм.

Психологи виділяють спеціальний розділ, психологію безпеки, в якому розглядають психічні властивості та різноманітні форми психічних станів, що спостерігаються у процесі трудової діяльності [15]. Психічні процеси становлять основу психічної діяльності. Без них неможливе формування знань та надбання життєвого досвіду. Розрізняють пізнавальні, емоційні та вольові психічні процеси.

Психічні властивості – це стійкі особливості особи: інтелектуальні, емоційні, вольові, трудові та ін. Психічні стани зумовлюють особливості психічної діяльності у конкретний період часу та можуть позитивно чи негативно впливати на всі психічні процеси. На думку багатьох психологів, ефективність діяльності, працездатність, людини залежить від рівня психічного напруження.

Підвищення рівня психічного напруження істотно збільшує ефективність праці. Але існує критична межа активації, після якої результати праці знижуються аж до повної втрати працездатності.

Існують два типи позамежевого психологічного напруження – гальмівний та збудливий. Гальмівний тип характеризується скутістю та сповільненістю рухів. Людина не здатна з колишньою спритністю виконувати професій ні дії. Знижується швидкість реакцій, сповільнюється процес мислення, погіршується згадування, розпорошується увага та виникають інші негативні прояви, не властиві даній людині у спокійному стані. Збудливий тип

проявляє себе гіперактивністю, багатомовністю, тремтінням рук та голосу. Оператори здійснюють численні, не продиктовані конкретною потребою, дії.

Психогенні зміни настрою та афектні стани виникають під впливом психічних дій. Зниження настрою та апатія можуть бути наявні від кількох хвилин до одного – двох місяців, Погіршення настрою спостерігається внаслідок конфліктних ситуацій, після загибелі близьких та в інших випадках. При цьому з’являються байдужість, млявість, загальна скутість, загальмованість, сповільнення темпу мислення . Погіршення настрою супроводжується погіршенням самоконтролю, що може стати причиною травматизму та збільшує ризик виникнення небезпечних ситуацій.

Афектні стани (афект – вибух емоцій) можуть виникнути внаслідок виробничих невдач, під впливом образи. У стані афекту у людини розвивається емоційне звуження обсягу свідомості. Можуть спостерігатися різкі рухи, агресивні та руйнівні дії. Особи, схильні до афектних станів, належать до категорії з підвищеним ризиком травматизму та не повинні призначатися на посади з високою відповідальністю.

Тому під час стресів, чи нервових напружень необхідно звертатись до відповідних фахівців, які допоможуть вийти з складної систуації.

4.2 Проведення інструктажів з охорони праці

Інструктажі з питань охорони праці проводяться на всіх підприємствах, установах і організаціях незалежно від характеру їх трудової діяльності, підлеглості і форми власності. Мета інструктажу - навчити працівника правильно і безпечно для себе і навколишнього середовища виконувати свої трудові обов'язки [16].

Інструктажі за часом і характером проведення поділяють на:

1) Вступний інструктаж проводиться з усіма працівниками, які щойно прийняті на роботу (постійну або тимчасову), незалежно від їх освіти, стажу роботи за цією професією або посади; працівниками, які знаходяться у

відрядженні на підприємстві й беруть безпосередню участь у виробничому процесі; з водіями транспортних засобів, які вперше в'їжджають на територію підприємства; учнями, вихованцями та студентами навчально-виховних закладів перед початком трудового й професійного навчання в лабораторіях, майстернях на полігонах тощо.

Вступний інструктаж проводить спеціаліст відділу охорони праці або особа, що призначена наказом для проведення цієї роботи.

Запис про проведення вступного інструктажу робиться в спеціальному журналі, а також у документі про прийняття працівника на роботу, де розписуються інструктуючий та проінструктований працівники.

2) Первинний інструктаж проводиться на робочому місці до початку роботи з новоприйнятим працівником або працівником, який буде виконувати нову для нього роботу, студентом, учнем та вихованцем перед роботою в майстернях, лабораторіях, дільницях тощо. Первинний інструктаж проводиться індивідуально або для групи осіб спільного фаху за програмою, складеною з урахуванням вимог відповідних інструкцій з охорони праці та інших нормативних актів про охорону праці, технічної документації і орієнтованого переліку питань первинного інструктажу, викладених в додатку до Типового положення про навчання, інструктаж та перевірку знань з питань охорони праці. Програма первинного інструктажу розробляється керівником цеху чи дільниці, узгоджується зі службою охорони праці і затверджується роботодавцем, керівником навчального закладу або відповідного структурного підрозділу.

Усі робітники і випускники професійних навчальних закладів після первинного інструктажу на робочому місці повинні пройти стажування протягом 2-15 змін під керівництвом досвідчених кваліфікованих робітників або спеціалістів, що призначаються наказом (розпорядженням) по підприємству, цеху, дільниці, виробництву. В окремих випадках стажування може не призначатися, якщо робітник має стаж роботи за своєю професією не

менше трьох років, а робота, яку він виконуватиме, для нього знайома з попереднього місця праці.

3) Повторний інструктаж проводиться на робочому місці з усіма працівниками: на роботах із підвищеною небезпекою - один раз на квартал; на інших роботах - один раз у півріччя. Мета інструктажу - поновити знання та уміння виконувати працівником роботу правильно і безпечно. Проводиться інструктаж індивідуально або для групи працівників, що виконують однотипні роботи, за програмою первинного інструктажу в повному обсязі.

4) Позаплановий інструктаж проводиться з працівниками на робочому місці або в кабінеті охорони праці у таких випадках:

- при введенні в дію нових або змінених нормативних актів про охорону праці;

- при зміні технологічного процесу, заміні або модернізації устаткування, приладів та інструментів, вихідної сировини, матеріалів та інших факторів, що впливають на охорону праці;

- при порушенні працівником нормативних актів, що може призвести до травми, отруєння або аварії;

- на вимогу працівника органу державного нагляду або вищої за ієрархією державної чи господарської організації при виявленні недостатнього знання працівником безпечних прийомів праці і нормативних актів про охорону праці;

- при перерві в роботі виконавця робіт більше, ніж ЗО календарних днів (для робіт із підвищеною небезпекою), а для решти робіт - більше 60 днів.

Позаплановий інструктаж проводиться індивідуально або для групи працівників спільного фаху. Обсяг і зміст інструктажу визначається для кожного окремого випадку залежно від причин і обставин, що викликали необхідність його проведення.

Цільовий інструктаж проводиться у таких випадках:

- при виконанні разових робіт, що не пов'язані безпосередньо з основними роботами працівника;

- при ліквідації наслідків аварії і стихійного лиха;

- при виконанні робіт, що оформляються нарядом-допуском, письмовим дозволом та іншими документами;

- в разі проведення екскурсій або організації масових заходів з учнями та вихованцями (екскурсії, походи, спортивні заходи тощо).

Цільовий інструктаж фіксується нарядом-допуском або іншим документом, що дозволяє проведення робіт.

Первинний, повторний, позаплановий та цільовий інструктажі проводить безпосередньо керівник робіт (начальник виробництва, цеху, дільниці, майстер, інструктор виробничого навчання, викладач тощо). Перевірка знань здійснюється усним опитуванням або за допомогою технічних засобів навчання, а також перевіркою навичок виконання робіт відповідно до вимог безпеки.

Первинний, повторний та позаплановий інструктажі, стажування та допуск до роботи реєструються в спеціальних журналах. При цьому обов'язкові підписи як інструктованого, так і інструктуючого. Журнали інструктажів повинні бути пронумеровані, прошнуровані і скріплені печаткою.

Працівники, що не пов'язані з обслуговуванням обладнання, використанням інструменту, збереженням сировини, матеріалів тощо, можуть бути звільнені від первинного, повторного та позапланового інструктажу за наказом (розпорядженням) керівника підприємства по узгодженню з державним інспектором Держнаглядохоронпраці.

Роботодавець або керівник структурного підрозділу зобов'язаний видати працівнику примірник інструкції з охорони праці за його професією або вивісити її на робочому місці.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було виконано наступне:

– проведено аналіз існуючих пристроїв для відстеження розташування домашніх тварин, виявлено їх переваги та недоліки;

– визначено набір основних функцій та вимог до системи, що розробляється;

– розроблено архітектуру та функціональний склад системи;

– обрані засоби реалізації та платформа для функціонування системи;

– спроектовано структуру бази даних;

– розроблено серверну частину системи;

– розроблено клієнтську частину системи;

– здійснено складання макетного зразка трекера;

– проведено тестування роботи кожної частини окремо та їх взаємодії.

Результатом виконаної роботи є спроектована система для відстеження розташування домашніх тварин.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Garmin Astro 320 з GPS нашийником DC50 - надійний помічник для полювання з собакою URL: https://garmin.lviv.ua/articles/605 (дата звернення: 15.05.2023).
2. ТК-909 Нашийник з GPS Трекером для Великих Собак, Трекер для Полювання, Tracker, TKSTAR URL: https://700.com.ua/ua/p1149688900-909-oshejnik-gps.html (дата звернення: 15.05.2023).
3. Delta Smart™ Dog Training System URL: https://www.garmin.com/en-US/p/527284#specs (дата звернення: 15.05.2023).
4. UML для бізнес-моделювання URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 15.05.2023).
5. GPSM - кращий моніторинг автомобілів URL: https://intelli.com.ua/ua/statti/blog.html (дата звернення: 15.05.2023).
6. Wi-Fi location. Позиціонування та обробка даних мобільних клієнтів URL: https://netwave.ua/product/wi-fi-location/ (дата звернення: 15.05.2023).
7. GSM/GPRS M66 URL: https://www.quectel.com/product/gsm-gprs-m66 (дата звернення: 15.05.2023).
8. Іон літію проти Літій -полімерні батареї - що краще? URL: https://ua.torphanbattery.com/news/lithium-ion-vs-lithium-polymer-batteries-wh-49324646.html (дата звернення: 24.05.2023).
9. SQLite Tutorial URL: https://www.sqlitetutorial.net (дата звернення: 15.05.2023).
10. ASP.NET Free. Cross-platform. Open source. A framework for building web apps and services with .NET and C# URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата звернення: 24.05.2023).
11. GPS для контролю тварин. URL: https://totalcontrol.ua/blog/article/58 (дата звернення: 15.05.2023).

12. Нашийник із GPS для котів і собак-маячок. URL: https://fixon.com.ua/language/uk/mayachok-nashyjnyk-z-gps-dlya-kotiv-i-sobak/ (дата звернення: 15.05.2023).

13. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU.Tern.: TNTU. 2023. Vol 109. No 1. P. 54–65.

14. Yatsyshyn V., Pastukh O., Zharovskyi R., Shabliy N.Software tool for productivity metrics measure of relational database management system. Mathematical Modeling. No 1 (48). 2023. p. 7-17.

15. Психологія безпеки. URL: https://pidru4niki.com/70727/bzhd/psihologiya_bezpeki (дата звернення: 13.06.2023).

16. Дуднікова І.І. Безпека життєдіяльності. Навч. посібник. – 2-ге вид.,доп. – К.: Вид-во Європ. ун-ту, 2013. 268 с.

17. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

Додаток А.

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“___” _____2023 р

КОМП’ЮТЕРИЗОВАНА СИСТЕМА ДЛЯ ВІДСТЕЖЕННЯ ДОМАШНІХ ТВАРИН

ТЕХНІЧНЕ ЗАВДАННЯ

на _9_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІс-41
_____ к.т.н., доц. Луцик Н.С.	_____ Степанко О.І.
«____» _____ 2023 р.	«____» _____ 2023 р.

Тернопіль 2023

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система для відстеження домашніх тварин».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.360.00.00

1.2 Виконавець

Студентка групи СІс-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Степанко Оксана Ігорівна.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-237 від 28.02.2023 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 01.03.2023 р.

Плановий термін завершення виконання кваліфікаційної роботи – 19.06.2023 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Система що розробляється призначена для:

– Визначення місця розташування домашніх тварин.
– Відображення на карті і побудови короткого маршруту до тварини.
– Зберігання даних про домашніх тварин.

2.2 Мета створення системи

Метою кваліфікаційної роботи є розробка комп’ютеризованої системи, яка використовує GPS для визначення місцезнаходження, а також створення програми на Android, яка буде виводити дані про місцезнаходження власника та місцезнаходження тварини на карту.

2.3 Характеристика об'єкту

Об’єкт розробки є система яка буде мати можливість здійснювати відстеження великих домашніх тварин і надсилати дані власнику.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Комп’ютерна система відеонгляду повинна складатись з:

- GPS трекера;
- бази даних;
- додатку з графічним інтерфейсом.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Основна вимога, яка ставиться до способів та засобів інформаційного обміну – це їх узгодженість.

3.1.3 Вимоги до режимів функціонування системи

Для системи визначено два режими функціонування:

– нормальний режим функціонування;

– аварійний режим функціонування.

Для забезпечення нормального режиму функціонування системи необхідно виконувати вимоги і дотримуватись умов експлуатації програмного забезпечення і комплексу технічних засобів системи, вказані у відповідних технічних документах (технічна документація, інструкції з експлуатації і т. д.).

Аварійний режим функціонування системи характеризується відмовою одного або декількох компонент програмного і (або) технічного забезпечення. При цьому функції роботи системи продовжують підтримувати роботу системи відстеження в межах базових налаштувань.

3.1.4 Вимоги по діагностуванню системи

Для діагностування системи відстеження використовуються інструменти діагностування основних процесів системи, які дозволяють перевірити основні функціональні елементи системи.

Інструменти повинні забезпечувати зручний інтерфейс для можливості перегляду діагностичних подій, моніторингу процесу виконання програм.

3.1.5 Перспективи розвитку, проектування системи

Дана система може бути розширена завдяки додаванню нових функцій і модулів.

3.2 Показники призначення

Система повинна передбачати можливість масштабування. Можливості масштабування повинні забезпечуватися засобами використовуваного базового програмного і технічного забезпечення.

3.2.1 Вимоги до надійності

Система повинна забезпечувати працездатність та відновлення своїх функцій при виникненні наступних ситуацій:

– при збоях в системі електропостачання апаратної частини;

– при помилках в роботі апаратних засобів;

– при помилках, пов'язаних з програмним забезпеченням (ОС і драйвери пристроїв).

Для захисту апаратури від стрибків напруги і комутаційних завад повинні застосовуватися мережні фільтри.

3.3 Вимоги до безпеки

Зовнішні елементи технічних засобів системи, що перебувають під напругою, повинні мати захист від випадкового дотику, а самі технічні засоби мати занулення або захисне заземлення.

Система електроживлення повинна забезпечувати захисне вимикання при перевантаженнях і коротких замиканнях в колах навантаження, а також аварійне ручне вимикання.

Загальні вимоги пожежної безпеки повинні відповідати нормам на побутове електрообладнання. У разі пожежі не має виділятися отруйних газів і димів. Після зняття електроживлення має бути доступне застосування будь-яких засобів пожежогасіння.

3.3.1 Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Мікроклімат в приміщеннях повинен відповідати нормам виробничого мікроклімату по ДСН 3.3.6.042-99:

– температуру повітря в межах від +10°С до +35°С;

– відносну вологість повітря при 25°С в межах від 30% до 80%;

– атмосферний тиск 760±25 мм рт. ст.

Періодичне технічне обслуговування використовуваних технічних засобів має проводитися відповідно до вимог технічної документації, але не рідше ніж один раз на рік.

Періодичне технічне обслуговування і тестування технічних засобів повинні включати обслуговування і тестування всіх використовуваних засобів.

На підставі результатів тестування технічних засобів повинні проводитися аналіз причин виникнення виявлених дефектів і прийматися заходи по їх ліквідації.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Система повинна забезпечувати захист від несанкціонованого доступу на рівні не нижче встановленого вимогами, що пред'являються до категорії 1Д по класифікації документа, що діє, “Автоматизовані системи. Захист від

несанкціонованого доступу до інформації. Класифікація автоматизованих систем".

Компоненти підсистеми захисту від НСД повинні забезпечувати:

– ідентифікацію користувача;

– перевірку повноважень користувача при роботі з системою;

– розмежування доступу користувачів.

Рівень захищеності від несанкціонованого доступу засобів обчислювальної техніки, що здійснюють обробку конфіденційної інформації, повинен відповідати вимогам класу захищеності згідно вимогам документу "Засоби обчислювальної техніки. Захист від несанкціонованого доступу до інформації. Показники захищеності від несанкціонованого доступу до інформації".

3.4.1 Вимоги по збереженню інформації при аваріях

Інформація, при виникненні аварійних ситуацій повинна бути збережена на резервних носіях.

3.4.2 Вимоги по стандартизації і уніфікації

Система повинна відповідати вимогам ергономіки і зручності користування за умови комплектування високоякісним обладнанням (ЕОМ, монітор і інше обладнання), що має необхідні сертифікати відповідності і безпеки.

3.4.3 Вимоги до функцій (завдань), що виконуються системою:

– забезпечення якісного зображення;

– забезпечення умов зберігання даних користувачів;

– забезпечення необхідного рівня сигналу;

– забезпечення високої швидкодії.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

Функціональна схема системи.

Діаграма варіантів використання системи.

Структурна схема Android -додатку.

Результат роботи системи.

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка технічного завдання	01.03-25.03.2023
2	Аналіз технічного завдання	26.03-01.04.2023
3	Аналіз вимог до системи відстеження тварин	02.04-16.04.2023
4	Проектування структури системи пошуку тварин	17.04-04.05.2023
5	Проектування програмної частини	05.05-10.05.2023
6	Проектування апаратної частини	11.05-29.05.2023
7	Налаштування і тестування системи відстеження тварин	30.05-05.06.2023
8	Безпека життєдіяльності, основи охорони праці	06.06-10.06.2023
9	Оформлення кваліфікаційної роботи	11.06-15.06.2023
10	Попередній захист кваліфікаційної роботи	16.06-20.06.2023
11	Захист кваліфікаційної роботи	19.06-24.06.2023

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б.

Лістинги серверної частини

Лістинг Б.1 - Вихідний код "AccountController"

```
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ServerAPI.Authenticate;
using ServerAPI.Models;
using ServerAPI.Models.Database;
using static ServerAPI.Models.Database.User;
namespace ServerAPI.Controllers {
[Route("api/account")]
[ApiController]
public class AccountController : Controller {
private readonly DatabaseContext database;
public AccountController(DatabaseContext database) {
this.database = database;
}
[HttpPost]
[Route("login")]
public async Task<IActionResult> Login([FromBody] LoginModel
model) // метод використовується для обробки інформації про
користувача при авторизації
{
User user = await database.Users
.Include(u => u.Animals)
                .ThenInclude(u => u.Coordinates)
.FirstOrDefaultAsync(u => u.Email.Equals(model.Email) &&
u.Password.Equals(model.Password));
if (user is null) return StatusCode (400, "incorrect email
or password");
ClaimsIdentity identity = GetIdentity(model.Email,
model.Password);
string accessToken = TokenGenerator.GetToken(identity);
AccountInfo accountInfo = new AccountInfo {
Name = user.Name,
Surname = user.Surname,
AccessToken = accessToken, Animals = user.Animals
};
return Ok(accountInfo); }
[HttpPost]
[Route("registration")] // Цей метод використовується для
обробки інформації про користувача при реєстрації
Public IActionResult Registration([FromBody]
RegistrationModel model) {
if (model is null) return StatusCode (204, "data is empty");
```

```
     User user = model.ToUser();
     if (database.Users.ToList().Contains(user, new
UserComparer())) return StatusCode(400, new { Message = "user
contains in database" });
     database.Users.Add(user);
     database.SaveChanges();
     ClaimsIdentity identity = GetIdentity(model.Email,
model.Password);
     string accessToken = TokenGenerator.GetToken(identity);
     AccountInfo accountInfo = new AccountInfo
     {
     Name = model.Name,
     Surname = model.Surname,
     AccessToken = accessToken, Animals = model.Animals };
     return Ok(accountInfo);
     }
     private ClaimsIdentity GetIdentity(string email, string
password) // метод викликається, якщо користувач був знайдений,
то створюється об'єкт ClaimsIdentity
     {
     List<Claim> claims = new List<Claim>
     {
     new Claim(ClaimsIdentity.DefaultNameClaimType, email) };
     return new ClaimsIdentity(claims, "Token",
ClaimsIdentity.DefaultNameClaimType,
     ClaimsIdentity. DefaultRoleClaimType);
     }
     }
     }
```

Лістинг Б.2 - Вихідний код "AnimalLocationController"

```
     using System;
     using System.Threading.Tasks;
     using Microsoft. AspNetCore.Authentication.JwtBearer;
     using Microsoft.AspNetCore.Authorization;
     using Microsoft. AspNetCore.Mvc;
     using Microsoft.EntityFrameworkCore;
     using ServerAPI.Models.AnimalLocation;
     using ServerAPI.Models.Database;
     namespace ServerAPI.Controllers {
     [Route("api/animallocation")]
     [ApiController]
     Public class AnimalLocationController : ControllerBase
     {
     private readonly Databasecontext database;
     public AnimalLocationController(DatabaseContext database) =>
this.database = database;
     [HttpPost]
     [Route("put")]
     public async Task<IActionResult> Put([FromBody]
ModelForPutAnimalLocation model) //відправка запиту отримання
координат тварини
```

```
     if (model is null) return BadRequest("data is empty");
     Animal animal = await FindAnimal(model.Email, model.IdKey);
     if (animal is null) return BadRequest("animal not found");
     if (animal .Coordinates is null)
     animal.Coordinates = model.Coordinates;
     else
     animal.Coordinates.Update(model.Coordinates);
     database
     .An ima ls
     . Update (animal);
     await database
     .SaveChangesAsync();
     retu rn Ok();
     }
     [HttpPost]
     [Route("get")]
     public async Task<IActionResult> Get([FromBody]
ModelForGettingAnimalLocation model) // отримання координат
тварини
     {
     Animal animal = await FindAnimal(model.Email, model.IdKey);
     if (animal is null) return BadRequest("animal not found");
     if (animal.Coordinates is null) BadRequest("location of
animal is null");
     return Ok(animal.Coordinates);
     }
     private async Task<Animal> FindAnimal(string email, string
idKey) //Визначення розташування тварини
     {
     User user = await database
     .Users
     .Include(u => u.Animals)
     .ThenInclude(a => a.Coordinates)
     .FirstOrDefaultAsync(u => u.Email.Equals(email));
     if (user is null) return null;
     return user .Animals .Find(a => a.IdKey.Equals(idKey));
     }
```

Лістинг Б.3. - Вихідний код "ModelForGettingAnimalLocation"

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace ServerAPI.Models.AnimalLocation {
Public class ModelForGettingAnimalLocation {
public string Email { get; set; }
public string IdKey {get; set; } //ідентифікатор нашийника }
}
```

Лістинг Б.4. - Вихідний код "ModelForPutAnimalLocation"

```
using ServerAPI.Models.Database;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace ServerAPI.Models.AnimalLocation {
Public class ModelForPutAnimalLocation
{
public string Email { get; set; }
public string IdKey {get; set; } // Ідентифікатор нашийника
public LatLng Coordinates {get; set; } //координати
}
}
```

Лістинг Б.5. - Вихідний код "DatabaseContext"

```
using Microsoft.EntityFrameworkCore;
namespace ServerAPI.Models.Database {
public class DatabaseContext:DbContext // визначає контекст
даних, для взаємодії з базою даних
{
public DatabaseContext(DbContextOptions<DatabaseContext>
options) : base(options) { }
public DbSet<User> Users { get; set; } public DbSet<Animal>
Animals { get; set; } }
}
```

Лістинг Б.6 - Вихідний код “User”

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace ServerAPI.Models.Database {
public class User {
//дані поля описують інформацію про користувача
public int Id { get; set; }
public string Name { get; set; }
public string Surname {get; set; }
public string Birthday { get; set; }
public string Email { get; set; }
public string Password { get; set; }
public List<Animal> Animals { get; set; }
public User() {
Id = 0;
Name = Surname = Email = Password = Birthday = string.Empty;
Animals = new List<Animal>();
}
public class UserComparer : IEqualityComparer<User>
//застосовується для порівняння користувачів
{ public bool Equals(User x, User y) {
if (ReferenceEquals(x, y)) return true;
if (ReferenceEquals(x, null) | | ReferenceEquals(y, null))
return false;
try {
```

```
     return (x.Id == y.Id && x.Name == y.Name && x.Surname ==
y.Surname &&
     x.Birthday == y.Birthday && x.Email == y.Email && x.Password
== y.Password &&
     x.Animals.SequenceEqual(y.Animals));
     }
     catch {return false; } }
     public int GetHashCode(User obj) {
     throw new NotImplementedException();
     } }
     }
     }
```

Лістинг Б.7 - Вихідний код "Accountlnfo"

```
using ServerAPI.Models.Database;
using System.Collections.Generic;
namespace ServerAPI.Models {
public class AccountInfo // Обліковий запис користувача {
public string Name { get; set; }
public string Surname {get; set; }
public string AccessToken { get; set; }
public List<Animal> Animals { get; set; } }
}
```

Лістинг Б.8. - Вихідний код " RegistrationModel"

```
using ServerAPI.Models.Database;
using System.Collections.Generic;
namespace ServerAPI.Models
{
public class RegistrationModel
{
//дані необхідні під час реєстрації
public string Name { get; set; }
public string Surname {get; set; }
public string Birthday { get; set; }
public string Email { get; set; }
public string Password { get; set; }
public List<Animal> Animals { get; set; }
public User ToUser() => //додавання користувача до бази
даних
new User
{
Name = Name,
Surname = Surname,
Birthday = Birthday,
Email = Email,
Password = Password,
Animals = Animals };
} }
```

Лістинг Б.9. - Вихідний код “ LoginModel”

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace ServerAPI.Models
{
public class LoginModel
{
public string Email { get ; set ; }
public string Password { get ; set ; } } }
```

Лістинг Б.10 - Вихідний код "AuthOptions"

```
using Microsoft. IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ServerAPI.Authenticate
{
public class AuthOptions
{
public const string ISSUER = "ServerAPI"; // видавець токена
public const string AUDIENCE = "ServerAPI"; //споживач
токена
public const int LIFETIME = 3; //час життя токена
private const string PRIVATE_KEY =
"6oqFQOL5y5CA8pPnVk2NdBH4e0I0sA"; //ключ public static
SymmetricSecurityKey SecurityKey =>
New SymmetricSecurityKey(Encoding. ASCII.
GetBytes(PRIVATE_KEY));
}
```

Лістинг Б.11 - Вихідний код "TokenGenerator"

```
using Microsoft. IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
namespace ServerAPI.Authenticate
{
public static class TokenGenerator
{
public static string GetToken(ClaimsIdentity claims) //Цей
метод генерує токен
{
DateTime dateNow = DateTime. Now;
JwtSecurityToken token = новий JwtSecurityToken(
issuer: AuthOptions.ISSUER,
audience: AuthOptions.AUDIENCE,
notBefore: dateNow,
expires: dateNow.Add(TimeSpan.
```

```
FromHours(AuthOptions.LIFETIME)),
     claims: claims.Claims,
     signingCredentials: new
SigningCredentials(AuthOptions.SecurityKey, SecurityAlgorithms.
HmacSha256));
     return new JwtSecurityTokenHandler(). WriteToken(token);
     }
     }
     }
```

Лістинг Б.12 - Початковий код Migration

```
     using Microsoft.EntityFrameworkCore.Migrations;
     namespace ServerAPI.Migrations
     {
     public partial class Initial : Migration
     {
     protected override void Up(MigrationBuilder
migrationBuilder)
     {
     migrationBuilder.CreateTable( name: "LatLng", columns: table
=> new
     {
     Id = table. Column<int>(nullable: false)
     .Annotation("Sqlite:Autoincrement",  true),  Lat  =  table.
Column<float>(nullable:        false),        Longitude        =
table.Column<float>(nullable: false)
     },
     constraints: table =>
     {
     table.PrimaryKey("PK_LatLng", x => x.Id);
     });
     migrationBuilder.CreateTable( name: "Users ", columns: table
=> new
     {
     Id = table. Column<int>(nullable: false)
     .Annotation("Sqlite:Autoincrement",     true),     Name     =
table.Column<string>(nullable:       true),       Surname       =
table.Column<string>(nullable:    true),    Birthday    =    table.
Column<string>(nullable:      true),      Email      =      table.
Column<string>(nullable:     true),     Password     =     table.
Column<string>(nullable: true) }, constraints: table => {
     table.PrimaryKey("PK_Users", x => x.Id);
     });
     migrationBuilder.CreateTable( name: "Animals", columns:
table => new {
     Id = table. Column<int>(nullable: false)
     .Annotation("Sqlite:Autoincrement",     true),     Name     =
table.Column<string>(nullable:    true),    IdKey    =    table.
Column<string>(nullable:       true),       CoordinatesId       =
table.Column<int>(nullable:        true),        UserId        =
table.Column<int>(nullable: true) }, constraints: table => ï
     {
```

```
     table.PrimaryKey("PK_Animals", x => x.Id); table.ForeignKey(
     name: "FK_Animals_LatLng_CoordinatesId", column: x =>
x.CoordinatesId, principalTable: "LatLng",
     principalColumn: "Id",
     onDelete: ReferentialAction.Restrict);
     table.ForeignKey(
     name: "FK_Animals_Users_UserId",
     column: x => x.UserId, principalTable: "Users ",
principalColumn: "Id", onDelete: ReferentialAction. Restrict);
     });
     migrationBuilder.CreateIndex(
     name: "IX_Animals_CoordinatesId",
     table: "Animals ", column: "CoordinatesId");
     migrationBuilder.CreateIndex( name: "IX_Animals_UserId ",
table: "Animals ", column: "UserId ");
     }
     protected override void Down(MigrationBuilder
migrationBuilder) {
     migrationBuilder.DropTable( name: "Animals");
     migrationBuilder.DropTable( name: "LatLng");
     migrationBuilder.DropTable( name: "Users ");
     }
     } }
```

Лістинг Б.13 - Початковий код Migration

```
     using System;
     using Microsoft.EntityFrameworkCore;
     using Microsoft.EntityFrameworkCore.Infrastructure;
     using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
     using ServerAPI.Models.Database;
     namespace ServerAPI.Migrations {
     [DbContext(typeof(DatabaseContext))]
     partial class DatabaseContextModelSnapshot : ModelSnapshot {
     protected override void BuildModel(ModelBuilder
modelBuilder) {
     #pragma warning disable 612, 618
     modelBuilder
     .HasAnnotation("ProductVersion", "2.1.4-rtm-31024");
     ModelBuilder. Entity("ServerAPI. Models.Database.Animal", b
=> {
     b.Property<int>("Id")
     .ValueGeneratedOnAdd();
     b. Property<int?>("CoordinatesId");
     b. Property<string>("IdKey");
     b.Property<string>("Name");
     b.Property<int?>("UserId");
     b.HasKey("Id");
     b. HasIndex("CoordinatesId");
     b.HasIndex("UserId");
     b.ToTable("Animals");
     });
     ModelBuilder. Entity("ServerAPI. Models.Database.LatLng", b
```

```
=>
     {
     b.Property<int>("Id")
     .ValueGeneratedOnAdd();
     b.Property<float>("Lat");
     b. Property<float>("Longitude");
     b.HasKey("Id");
     b. ToTable("LatLng ");
     });
     modelBuilder.Entity("ServerAPI.Models.Database.User", b =>
     {
     b.Property<int>("Id")
     .ValueGeneratedOnAdd();
     b. Property<string>("Birthday");
     b. Property<string>("Email");
     b.Property<string>("Name");
     b. Property<string>("Password");
     b.Property<string>("Surname");
     b.HasKey("Id");
     b. ToTable("Users");
     });
     ModelBuilder. Entity("ServerAPI. Models.Database.Animal", b
=>
     {
     b. HasOne("ServerAPI. Models.Database.LatLng",
"Coordinates")
     . WithMany()
     . HasForeignKey("CoordinatesId") ;
              b.HasOne("ServerAPI.Models.Database.User")
     .WithMany("Animals") .HasForeignKey("UserId");
     });
     #pragma warning restore 612, 618
     }
     }
     }
```

Лістинг Б.14 - Вихідний код "Startup"

```
using Microsoft. AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft. AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using ServerAPI.Authenticate;
using ServerAPI.Models.Database;
namespace ServerAPI {
public class Startup
{
public Startup(IConfiguration configuration)
{
Configuration = configuration;
```

```
    }
    public IConfiguration Configuration { get; }
    public void ConfigureServices(IServiceCollection
services)//perMCTpaukin служб програми
    {
    services.AddAuthentication(JwtBearerDefaults.AuthenticationS
cheme) .AddJwtBearer(authOptions =>
    {
    authOptions.RequireHttpsMetadata = false;
    authOptions.TokenValidationParameters = new
    Microsoft. IdentityModel.Tokens. TokenValidationParameters {
    ValidateIssuer = true,
    ValidateAudience = true,
    ValidateLifetime = true,
    ValidateIssuerSigningKey = true,
    ValidIssuer = AuthOptions.ISSUER, ValidAudience =
AuthOptions. AUDIENCE, IssuerSigningKey = AuthOptions.
SecurityKey
    };
    });
    string databaseConnetion =
    Configuration.GetConnectionString("DatabaseConnection");
    services.AddDbContext<DatabaseContext>(options =>
options.UseSqlite(databaseConnetion));
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersi
on.Version_2_1) .AddJsonOptions(jsonOptions =>
    jsonOptions.SerializerSettings. ReferenceLoopHandling =
Newtonsoft.Json. ReferenceLoopHandling.Ignore);
    }
    public void Configure(IApplicationBuilder app,
IHostingEnvironment envV //встановлює , як додаток буде
обробляти запит
    {
    if (env.IsDevelopment())
    {
    app.UseDeveloperExceptionPage();
    }
    else
    {
    app.UseHsts();
    }
    app.UseAuthentication();
    app.UseCors(corsOptions =>
    {
    corsOptions.AllowAnyHeader();
    corsOptions.AllowAnyMethod();
    corsOptions.AllowAnyOrigin();
    corsOptions.AllowCredentials();
    });
    app.UseMvc();
```

Лістинг Б.15 - Вихідний код " Program"

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
namespace ServerAPI {
public class Program
{
public static void Main(string[] args)
{
CreateWebHostBuilder(args).Build().Run(); }
public static IWebHostBuilder CreateWebHostBuilder(string[]
args) => WebHost.CreateDefaultBuilder(args)
.UseKestrel() // налаштовуємо веб-сервер Kestrel
.UseUrls(" http://192.168.0.104 ")
.UseStartup<Startup>(); // встановлюємо головний файл
програми
}
}
```

Додаток В.

Лістинги клієнтської частини

Лістинг В.1 - Вихідний код " MapActivity"

```
package susu.com.animallocation.activities;
import android.Manifest;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.Serviceconnection;
import android.content.pm.PackageManager;
import android.os.IBinder;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import
com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import susu.com.animallocation.R;
import susu.com.animallocation.models.Settings;
import
susu.com.animallocation.fragment.presenters.MapPresenter.Present
er;
import
susu.com.animallocation.services.UpdateAnimalLocation.Update
AnimalLocationService;
public class MapActivity extends FragmentActivity implements
OnMapReadyCallback {
private UpdateAnimalLocationService.ServiceBinder
serviceBinder ;
private boolean isBindService ;
private GoogleMap mMap ;
private Presenter presenter ;
private Settings settings ;
private static final int LOCATION_PERMISSION_CODE = 1001;
@Override
protected void onCreate(Bundle savedInstanceState)
{ //Початкова установка параметрів
```

```
     super .onCreate(savedInstanceState);
     setContentView(R.layout. map_activity );
     presenter = new Presenter( this );
     settings = new Settings( this );
     if (checkPermissions())
     initialMap();
     }
     @Override
     protected void onStart()
     { // ініціалізація дій
     super .onStart();
     Intent serviceintent = new Intent( this ,
UpdateAnimalLocationService. class );
     String accessToken = settings .getAccessToken();
     serviceintent
     .putExtra( "EMAIL" , email)
     .putExtra( "ID_KEY" , idKey)
     .putExtra( "TOKEN" , accessToken);
     bindService(serviceintent, serviceConnection , Context.
BIND_AUTO_CREATE );
     }
     @Override
     protected void onDestroy()
     { // очищення всіх ресурсів
     super .onDestroy();
     if ( isBindService ) {
     unbindService( serviceConnection );
     isBindService = false ;
     }
     }
     @Override
     public void onMapReady(GoogleMap googleMap) { // готовність
карти
     mMap = googleMap;
     setinitialFocusOnMap();
     startUserLocationUpdate();
     }
     @Override
     public void onRequestPermissionsResult( int requestcode,
//запит дозволів
     @NonNull String[] permissions, @NonNull int [] grantResults)
{
     switch (requestcode) {
     case LOCATION_PERMISSION_CODE :
     if (grantResults[0] == PackageManager. PERMISSION_GRANTED )
{ initialMap();
     }
     else checkPermissions();
     break ;
     }
     }
     private boolean checkPermissions() {
     if (ContextCompat. checkSelfPermission ( this ,
```

```
     Manifest.permission. ACCESS_ FINE_LOCATION ) !=
PackageManager. PERMISSION_GRANTED ) {
     ActivityCompat. requestpermissions ( this , new
String[]{Manifest.permission. ACCESS_FINE_LOCATION },
LOCATION_PERMISSION_CODE );
     return false ;
     }
     return true ;
     }
     private void initialMap()
     { //ініціалізація карт
     SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
     .findFragmentById(R.id. map );
     mapFragment.getMapAsync( this );
     }
     private void setInitialFocusOnMap()
     { //фокусування карт
     LatLng Chelyabinsk = new LatLng(55.0914, 61.2544);
     mMap .moveCamera(CameraUpdateFactory. newLatLngZoom
(Chelyabinsk, 12));
     }
     @SuppressWarnings( "MissingPermission" )
     private void startUserLocationUpdate()
     { //Визначення розташування користувача
     Fused LocationProviderClient locationProviderClient
     = New FusedLocationProviderClient( this );
     LocationRequest locationRequest = новий LocationRequest();
     locat ion Req uest.setInte rval(2000);
     locationRequest.setFastestInterval(1000);
     locationRequest.setPriority(LocationRequest.
PRIORITY_HIGH_ACCURACY );
     LocationCallback locationCallback = новий LocationCallback()
{ @Override public void onLocationResult(LocationResult
locationResult) { if (locationResult == null ) {
     return ;
     }
     }
     };
     locationProviderClient.get LastLocation();
     mMap .setMyLocationEnabled( true );
     locationProviderClient.requestLocationUpdates(locationReques
t, locationCallback,
     null );
     }
     private ServiceConnection serviceConnection = new
ServiceConnection()
     { @Override
     public void onServiceConnected(ComponentName componentName,
IBinder serviceBinder) {
     MapActivity. this . serviceBinder = (
     UpdateAnimalLocationService.ServiceBinder) serviceBinder;
isBindService = true ;
```

```
     presenter .startUpdateMap();
     }
     @Override
     public void onServiceDisconnected(ComponentName
componentName) { isBindService = false ;
     }
     };
     private Marker mapMarker ;
     // робота з маркерами
     public void setMarker(LatLng coordinates) {
     if ( mMap == null ) return ;
     mapMarker = mMap .addMarker( new
MarkerOptions().position(coordinates)); }
     public void deleteMarker() {
     if ( mMap == null ) return ;
     ma pMa rke r .remove();
     }
     public Marker getMapMarker() {
     return mapMarker ;
     }
     public LatLng getAnimalCoordinates() {
     if ( serviceBinder == null &&
       ! serviceBinder .isSuccessful() && isBindService ) return
                              null ;
     return serviceBinder .getAnimalCoordinates();
     }
     public void showMessage(String message) {
     Snackbar. make (getCurrentFocus(), message, Snackbar.
LENGTH_SHORT );
     }
     }
```

Лістинг В.2 - Вихідний код " RootActivity"

```
     package susu.com.animallocation.activities;
     import android.os.Bundle;
     import android.support.annotation.Nullable;
     import android.support.v7.app.AppCompatActivity;
     import android.widget.ProgressBar;
     import susu.com.animallocation.R;
     import susu.com.animallocation.fragment.LoginF ragment;
     import static android.view.View.*;
     public class RootActivity extends AppCompatActivity {
     ProgressBar progressBar ;
     @Override
     protected void onCreate(@Nullable Bundle savedInstanceState)
     { // встановлення параметрів
     super .onCreate(savedInstanceState);
setContentView(R.layout. root_activity );
     progressBar = findViewById(R.id. progress );
     getSupportF ragmentManager()
     .beginTransaction()
     .add(R.id. fragment_container , new LoginFragment())
```

```
.commit();
     }
     public void showProgress()
     { //Відображення індикатора виконання
     if ( progressBar .getVisibility() != VISIBLE )
     progressBar .setVisibility( VISIBLE );
     }
     public void hideProgress()
     { //приховування індикатора виконання
     if ( progressBar .getVisibility() != INVISIBLE ) {
progressBar .setVisibility( INVISIBLE );
     }
     }
     }
```

Лістинг В.3 - Вихідний код “LoginFragmentPresenter”

```
     package susu.com.animallocation.fragment.presenters;
     import android.content.Intent;
     import android.support.transition.TransitionManager;
     import android.support.v4.app.Fragment;
     import android.support.v4.app.FragmentTransaction;
     import android.transition.Transition;
     import susu.com.animallocation.R;
     import susu.com.animallocation.fragment.LoginF ragment;
     import
susu.com.animallocation.fragment.RegistrationFragment;
     import susu.com.animallocation.fragment.RootFragment;
     import susu.com.animallocation.models.Settings;
     import susu.com.animallocation.models.apiclient.AccountInfo;
     import
susu.com.animallocation.models.apiclient.authentication.Authenti
cator;
     import
susu.com.animallocation.models.apiclient.authentication.LoginCal
lback;
     public class LoginFragmentPresenter {
     private LoginFragment fragment ;
     private Settings settings ;
     private String email ;
     private String password ;
     public LoginFragmentPresenter(LoginFragment fragment) {
     this . fragment = fragment;
     settings = new Settings(fragment.getActivity());
     }
     public void checkAuthentication()
     { //аутенфікація користувача
     if ( settings .is Login()) fragment .redirectT oMap();
     }
     public void redirectToRegistration()
     { //перенаправлення на сторінку реєстрації
     fragment .getActivity()
     .getSupportF ragmentManager()
```

```
     .beginTransaction()
     .setTransition(FragmentT ransaction. TRANSIT_FRAGMENT_OPEN )
     .replace(R.id. fragment_container , new
RegistrationFragment())
     .addToBackStack( null ) .commit();
     }
     Authenticator authenticator ;
     public void cancelAuth()
     { //скасування авторизації
     if ( authenticator != null ) { authenticator .cancel();
     }
     fragment .hideProgress();
     }
     public void login() { // авторизація fragment
.showProgress();
     email = fragment .getEmail ();
     password = fragment .getPassword();
     authenticator = новий Authenticator();
     authenticator .Authenticate( email , password , new
LoginCallback() {
     ^Override
     public void onComplete(AccountInfo accountInfo)
     { // успішне завершення авторизації
     settings .saveAccountInfo(accountInfo);
     fragment .redirectT oMap();
     fragment .hideProgress();
     }
     @Override
     public void onFail(String message)
     { //помилка
     fragment .showMessage(message);
     fragment .hideProgress();
     }
     @Override
     public void onFailRequest() { //немає підключення
     fragment .showMessage( fragment .getString(R.string.
error_connection_server ));
     fragment .hideProgress();
     }
     });
     }
     }
```

Лістинг В.4 - Вихідний код "RegistrationFragmentPresenter"

```
     package susu.com.animallocation.fragment.presenters;
     import android.text.TextUtils;
     import java.util.ArrayList;
     import susu.com.animallocation.R;
     import
susu.com.animallocation.fragment.RegistrationFragment;
     import susu.com.animallocation.models.apiclient.AccountInfo;
     import
```

```
susu.com.animallocation.models.apiclient.entities.Animal;
     import susu.com.animallocation.models.Settings;
     import susu.com.animallocation.models.Validator;
     import
susu.com.animallocation.models.apiclient.registration.ApiRegistr
ation;
     import
susu.com.animallocation.models.apiclient.registration.Registrati
onCallback;
     public class RegistrationFragmentPresenter {
     private RegistrationFragment fragment ;
     private Settings settings ;
     private String email ;
     private String password ;
     Public RegistrationFragmentPresenter(RegistrationFragment
fragment) {
     this . fragment = fragment;
     settings = new Settings(fragment.getActivity());
     }
     ApiRegistration registration ;
     public void registration^ { //реєстрація
     email = fragment .getEmail();
     password = fragment .getPassword();
     String firstname= fragment .getFirstname();
     String surname= fragment .getSurname();
     String birthday = fragment . getBirthday ();
     String animalName = fragment .getAnimalName();
     String animalId = fragment .getAnimalId();
     if (!validateFields( email , password , firstname, surname,
birthday, animalName, animalId))
     return ;
     Animal animal = new Animal(animalName, animalId);
ArrayList<Animal> animals = new ArrayList<>();
animals.add(animal);
     registration = new ApiRegistration();
     registration . _ _ _ _
     registrationCallback );
     }
     public void cancelRegistration() { if ( registration != null
) { registration .cancel();
     }
     }
     private boolean validateFields(String email, String
password, String firstname, String surname,
     String birthday, String animalName, String animalld) {
//перевірка правильності введення
     if (TextUtils. isEmpty (email))
     {
     fragment .setErrorLogin( fragment .getString(R.string.
error_field_required )); return false ;
     }
     if (TextUtils. isEmpty (password))
     {
```

```
     fragment .setErrorPassword( fragment .getString(R.string.
error_field_required )); return false ;
     }
     if (!Validator. validateEmail (email))
     {
     fragment .setErrorLogin( fragment .getString(R.string.
error_incorrect_email )); return false ;
     }
     if (!Validator. validatePassword (password)) {
     fragment .setErrorPassword( fragment .getString(R.string.
error_incorrect_password )); return false ;
     }
     if (TextUtils. isEmpty (firstname)) {
     fragment .setErrorFirstname( fragment .getString(R.string.
error_field_required )); return false ;
     }
     if (TextUtils. isEmpty (surname)) {
     fragment .setErrorSurname( fragment .getString(R.string.
error_field_required )); return false ;
     }
     if (TextUtils. isEmpty (birthday)) {
     fragment .setErrorBirthday( fragment .getString(R.string.
error_field_required )); return false ;
     }
     if (!Validator. validateBirthday (birthday)) {
     fragment .setErrorBirthday( fragment .getString(R.string.
error_incorrect_birthday )); return false ;
     }
     if (TextUtils. isEmpty (animalName)) {
     fragment .setErrorAnimalName( fragment .getString(R.string.
error_fieLd_required )); return false ;
     }
     if (TextUtils. isEmpty (animalld)) {
     fragment .setErrorAnimalld( fragment .getString(R.string.
error_fieLd_required )); return false ;
     }
     return true ;
     }
     private RegistrationCallback registrationcallback = new
RegistrationCallback()
     {
     @Override
     public void onComplete(AccountInfo accountInfo)
     { //Успішне завершення
     settings .saveAccountInfo(accountInfo);
     fragment .redi rectToMap();
     }
     @Override
     public void onFail(String message) { // Завершення з
помилкою fragment .showMessage(message);
     }
     @Override
     public void onFailRequest() { // помилка підключення
```

```
     fragment .showMessage( fragment .getString(R.string.
error_connection_server ));
     }
     };
     }
```

Лістинг В.5 - Вихідний код "Presenter"

```
     package
susu.com.animallocation.fragment.presenters.MapPresenter;
     import android.os.Handler;
     import com.google.android.gms.maps.model.LatLng;
     import java.util.Timer;
     import java.util.TimerTask;
     import susu.com.animallocation.activities.MapActivity;
     public class Presenter {
     private MapActivity activity ;
     private Timer timer ;
     private static final long DELAY = 0;
     private static final long PERIOD = 5;
     public Presenter(MapActivity activity) { this . activity =
activity;
     }
     public void startUpdateMap()
     { // умови оновлення карт
      timer = new Timer();
     timer .schedule(
     getInstanceT imerTask(),
     0,
     10000
     );
     }
     private void cancelUpdate()
     { //скасування оновлення
     timer .cancel();
     }
     private TimerTask getInstanceTimerTask() { return new
TimerTask() {
     @Override
     public void run() {
     handler .post( runnable );
     }
     };
     }
     private final Handler handler = new Handler();
     private final Runnable runnable = () -> Presenter. this
.updateMap(); private void updateMap()
     { //оновлення карт
     LatLng animalCoordinates = activity .getAnimalCoordinates();
     if (animalCoordinates == null ) { return ;
     }
     if ( activity .getMapMarker() != null ) { activity
.deleteMarker();
```

```
}
activity .setMarker(animalCoordinates);
}
}
```

Лістинг В.6 - Вихідний код "LoginFragment"

```
package susu.com.animallocation.fragment;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.view.inputmethod.Editorinfo;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import susu.com.animallocation.R;
import susu.com.animallocation.activities.RootActivity;
import
susu.com.animallocation.fragment.presenters.LoginFragmentPresent
er;
public class LoginFragment extends RootFragment {
private AutoCompleteTextView emailview ;
private EditText passwordView ;
LoginF ragmentPresenter presentar ;
@Override
public void onCreate(@Nullable Bundle savedinstanceState)
{ //Початкова установка параметрів
super .onCreate(savedinstanceState);
presenter = new LoginFragmentPresenter( this );
presenter .checkAuthentication();
}
@Nullable
@Override
Public View onCreateView(Layoutinflater inflater,
                @Nullable ViewGroup container,
                 Bundle savedInstanceState) {
return inflater.inflate(R.layout. login_fragment ,
container, false );
}
@Override
public void onViewCreated(View view, @Nullable Bundle
savedinstanceState)
{
super .onViewCreated(view, savedinstanceState);
initialViews(view);
getView().requestFocus();
}
@Override
```

```
    public void onPause()
    {
    super .onPause();
    presenter .cancelAuth();
    }
    @Override
    public void onStop()
    {
    super .onStop();
    presenter .cancelAuth();
    }
    @Override
    public void onDestroy()
    {
    super .onDestroy();
    presenter .cancelAuth();
    }
    public String getEmail()
    { //отримання логіну
    return emailView .getText().toString();
    }
    public String getPassword()
    { //отримання пароля
    return passwordView .getText().toString();
    }
    private void initialViews(View view) {
     // реакція на натискання на екран
    emailView = view.findViewById(R.id. email );
    emailView .setOnEditorActionListener( editViewListener );
    passwordview = view.findViewById(R.id. password );
    passwordView .setOnEditorActionListener( editViewListener );
    Button buttonSignIn = view.findViewById(R.id. signInButton
);
    buttonSignIn.setOnClickListener( buttonSignInListener );
    TextView registrationButton = view.findViewById(R.id.
registrationButton );
    registrationButton.setOnClickListener(
registrationButtonClickListener );
    }
    TextView.OnEditorActionListener editViewListener =
(textView, id, keyEvent) -> { if (id == EditorInfo.
IME_ACTION_DONE | | id == EditorInfo. IME_NULL ) { return true ;
    }
    return false ;
    };
    OnClickListener buttonSignInListener = v -> presenter
.login();
    OnClickListener registrationButtonClickListener = v ->
presenter .redirectToRegistration();
    public void showProgress()
    { ((RootActivity) getActivity()).showProgress();
    }
    public void hideProgress()
```

```
    { ((RootActivity) getActivity()).hideProgress();
    }
}
```

Лістинг В.7 - Вихідний код "RegistrationFragment"

```
    package susu.com.animallocation.fragment;
    import android.os.Bundle;
    import android.support.annotation.NonNull;
    import android.support.annotation.Nullable;
    import android.view.LayoutInflater;
    import android.view.View;
    import android.view.View.OnClickListener;
    import android.view.ViewGroup;
    import android.widget.AutoCompleteTextView;
    import android.widget.Button;
    import android.widget.EditText;
    import android.widget.TextView;
    import susu.com.animallocation.R;
    import
susu.com.animallocation.fragment.presenters.RegistrationF
ragmentPresenter;
    public class RegistrationFragment extends RootFragment {
    private AutoCompleteTextView emailview ;
    private EditText passwordView ;
    private AutoCompleteTextView userFirstnameView ;
    private AutoCompleteTextView userSurnameView ;
    private TextView userBirthdayView ;
    private AutoCompleteTextView animalNameView ;
    private AutoCompleteTextView animalIdView ;
    private RegistrationFragmentPresenter presenter ;
    @Nullable
    @Override
    Public View onCreateView(@NonNull LayoutInflater inflater,
                  @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
    return inflater.inflate(R.layout. registration_fragment ,
container, false ); }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable
Bundle savedInstanceState) {
    super .onViewCreated(view, savedInstanceState);
    presenter = new RegistrationFragmentPresenter( this );
    getView().requestFocus();
    initialViews(view);
    }
    @Override
    public void onPause() {
    super .onPause();
    presenter .cancelRegistration();
    }
    @Override
    public void onStop() {
```

```
     super .onStop();
     presenter .cancelRegistration(); }
     @Override
     public void onDestroy() {
     super .onDestroy();
     presenter .cancelRegistration(); }
                private void initialViews(View view)
                     { // реакція на натискання
          emailView = view.findViewById(R.id. email );
     passwordview = view.findViewById(R.id. password );
     userFirstnameView = view.findViewById(R.id. userFirstname );
     userSurnameView = view.findViewById(R.id. userSurname );
     userBirthdayView = view.findViewById(R.id. userBirthday );
     animalNameView = view.findViewById(R.id. animalName );
     animalIdView = view.findViewById(R.id. idKey );
     Button registrationButton = view.findViewById(R.id.
signInButton ); registrationButton.setOnClickListener(
buttonRegistrationListener );
     }
     //ввід данних
     public String getEmail(){ return emailView
.getText().toString(); }
     public String getPassword(){ return passwordView
.getText().toString(); }
     public String getFirstname() { return userFirstnameView
.getText().toString(); } public String getSurname(){ return
userSurnameView .getText().toString(); }
     public String getBirthday(){ return userBirthdayView
.getText().toString(); }
     public String getAnimalName(){ return animalNameView
.getText().toString(); }
     public String getAnimalId(){ return animalIdView
.getText().toString(); } //обробка помилок
     public void setErrorFirstname(String error) {
userFirstnameView.setError (error);
     userF irstnameView .requestFocus();
     }
     public void setErrorSurname(String error) { userSurnameView
.setError(error);
     userSurnameView .requestFocus();
     }
     public void setErrorBirthday(String error) {
userBirthdayView.setError (error);
     userBirthdayView .requestFocus();
     }
     public void setErrorAnimalName(String error) {
animalNameView .setError(error);
     animalNameView .requestFocus();
     }
     public void setErrorAnimalId(String error) { animalIdView
.setError(error);
     animalIdView .requestFocus();
     }
```

```
    public void setErrorLogin(String error) {
    emailView .setError(error);
    emailView .requestFocus();
    }
    public void setErrorPassword(String error) { passwordView
.setError(error);
    passwordView .requestFocus();
    }
    private OnClickListener buttonRegistrationListener = v -> {
presenter .registration();
    };
    }
```

Лістинг В.8. - Вихідний код "RootFragment"

```
    package susu.com.animallocation.fragment;
    import android.content.Intent;
    import android.support.design.widget.Snackbar;
    import android.support.v4.app.Fragment;
    import susu.com.animallocation.R;
    import susu.com.animallocation.activities.MapActivity;
    public class RootFragment extends Fragment {
    public void redirectToFragment(Fragment fragment) {
    getActivity()
    .getSupportFragmentManager()
    .beginTransaction()
    .replace(R.id. fragment_container , fr agment) .commit();
    }
    public void redirectToMap() { //перехід на карту
    Intent intent = new Intent(getActivity(), MapActivity. class
); startActivity(intent);
    getActivity().finish();
    }
    public void showMessage(String message) {
    Snackbar. make (getActivity().getCurrentFocus(), message,
Snackbar. LENGTH_LONG ). show();
    }
    }
```

Лістинг В.9. - Вихідний код “UpdateAnimalLocationTask ”

```
    package
susu.com.animallocation.models.apiclient.animal.location;
    import android.os.AsyncTask;
    import android.support.annotation.NonNull;
    import com.google.android.gms.maps.model.LatLng;
    import com.google.gson.Gson;
    import java.io.IOException;
    import okhttp3.Response;
    import susu.com.animallocation.models.apiclient.ApiClient;
    class UpdateAnimalLocationTask extends AsyncTask<Void, Void,
LatLng> {
    private static final String ACTION_ROUTE =
"/api/animallocation/get" ;
```

```
     private String email ;
     private String idKey ;
     private String accessToken ;
     private Gson gson ;
     private UpdateAnimalLocationCallback callback ;
     public UpdateAnimalLocationTask(@NonNull String email,
     @NonNull String idKey,
     @NonNull String accessToken,
     @NonNull UpdateAnimalLocationCallback callback) {
     this . email = email;
     this . idKey = idKey;
     this . accessToken = accessToken;
     this . gson = новий Gson ();
     this . callback = callback;
     }
     @Override
     protected LatLng doInBackground(Void... voids) { //
надсилання запиту
     try {
     ApiClient client = new ApiClient();
     String jsonData = gson .toJson( new
UpdateAnimalLocationModel( email , idKey ));
     String url = ApiClient. HOST + ACTION_ROUTE ;
     Response response = client.postRequest(url, accessToken ,
jsonData);
     if (response.isSuccessful() && response.code() == ApiClient.
SUCCESS_CODE ) {
     String jsonResponse = response.body().string();
     LatLng animalLocation = gson .fromJson(jsonResponse, LatLng.
class );
     return animalLocation;
     }
     else if (response.code() == ApiClient.
NOT_AUTHENTICATION_CODE ) {
     return null ;
     } else return null ;
     } catch (lOException e) {
     return null ;
     }
     }
     @Override
     protected void onPostExecute(LatLng latLng) { if
(isCancelled()) return ;
     if (latLng! = null )
     callback .OnComplete(latLng);
     else
     callback .OnFail();
     }
     }
```

Лістинг В.10 - Вихідний код "LoginTask"

```
     package
```

```
susu.com.animallocation.models.apiclient.authentication;
     import android.os.AsyncTask;
     import com.google.android.gms.common.api.Api;
     import com.google.gson.Gson;
     import okhttp3.Response;
     import susu.com.animallocation.models.apiclient.AccountInfo;
     import susu.com.animallocation.models.apiclient.ApiClient;
     import susu.com.animallocation.models.apiclient.AuthResult;
     class LoginTask extends AsyncTask <Void, Void, AuthResult> {
     private LoginModel model ;
     private ApiClient client ;
     private LoginCallback callback ;
     private static final String ACTION_LOGIN =
"/api/account/login" ;
     LoginTask(String email, String password, LoginCallback
callback) {
     model = new LoginModel(email, password);
     client = new ApiClient();
     this . callback = callback;
     }
     @Override
     protected AuthResult doInBackground(Void... voids) { //
надсилання запиту
     Gson gson = новий Gson();
     cancel( false );
     String jsonData = gson.toJson( model );
     try {
     String url = ApiClient. HOST + ACTION_LOGIN ;
     Response response = client .postRequest(url, jsonData);
     if (response.code() == ApiClient. SUCCESS_CODE ) {
     String jsonResponse = response.body().string();
     Accountinfo accountinfo = gson.fromJson(jsonResponse,
Accountinfo. class );
     return new AuthResult(accountinfo);
     } else {
     return new AuthResult(response.code(), response.message());
     }
     } catch (Exception e) {
     return null ;
     }
     }
     @Override
     protected void onPostExecute(AuthResult result) { if
(isCancelled()) return ;
     if (result == null ) {
     callback .onFailRequest();
     return ;
     }
     if (result.getCode() == ApiClient. SUCCESS_CODE ) { callback
.onComplete(result.getAccountinfo());
     } else {
     callback .onFail(result.getMessage());
     }
```

```
    }
    private class LoginModel {
    private String email ;
    private String password ;
    Public LoginModel(String email, String password) { this .
email = email;
    this . password = password;
    }
    }
    }
```

Лістинг В.11 - Вихідний код "RegistrationTask"

```
    package
susu.com.animallocation.models.apiclient.registration;
    import android.os.AsyncTask;
    import android.text.TextUtils;
    import com.google.android.gms.common. api.Api;
    import com.google.gson.Gson;
    import java.io.iOException;
    import java.util.List;
    import okhttp3.Response;
    import susu.com. animallocation.
models.apiclient.Accountinfo;
    import susu.com. animallocation.
models.apiclient.AuthResult;
    import susu.com. animallocation.
models.apiclient.entities.Animal;
    import susu.com. animallocation. models.apiclient.ApiClient;
    class RegistrationTask extends AsyncTask<Void, Void,
AuthResult> {
    private final String ACTION_REGISTRATION =
"/api/account/registration " ;
    private RegistrationCallback callback ;
    private RegistrationModel model ;
    private Gson gson ;
    private ApiClient apiClient ;
    RegistrationTask(String email, String password,
    String name, String surname, String birthday, List<Animal>
animals, RegistrationCallback callback) {
    model = new RegistrationModel(name, surname, birthday,
email, password, animals);
    this . callback = callback;
    gson = новий Gson ();
    apiClient = new ApiClient();
    }
    @Override
    protected AuthResult doInBackground(Void ... voids) {
    String url = ApiClient. HOST + ACTION_REGISTRATION ;
    String jsonData = gson . доJson( model );
    try {
    Response response = apiClient .postRequest(url, jsonData);
    if (response.isSuccessful() && response.code() == ApiClient.
```

```
SUCCESS_CODE ) {
     String jsonResponse = response.body().string();
     AccountInfo accountInfo = gson .fromJson(jsonResponse,
AccountInfo. class );
     return new AuthResult(accountInfo);
     } else {
     return new AuthResult(response.code(), response. message());
     }
     } catch (IOException e) { return null ;
     }
     }
     @Override
     protected void onPostExecute(AuthResult result) { if
(isCancelled()) return ;
     if (result == null ) {
     callback .onFail( "Connection failed " ); return ;
     }
     if (result.getCode() == ApiClient. SU CCE SS_ COD E ) {
     callback .onComplete(result.getAccountInfo());
     } else {
     callback .onFail(result.getMessage());
     }
     }
     private class RegistrationModel {
     private String name ;
     private String surname ;
     private String birthday ;
     private String email ;
     private String password ;
     private List<Animal> animals ;
     Public RegistrationModel(String name, String surname, String
birthday,
     String email, String password,List<Animal> animals ) {
     this . name = name;
     this . surname = surname;
     this . birthday = birthday;
     this . email = email;
     this . password = password;
     this . animals = animals;
     }
     }
     }
```

Лістинг В.12 - Вихідний код "LoginTask"

```
package susu.com.animallocation.models.apiclient;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import okhttp3.OkHttpClient.Builder;
```

```
public class ApiClient {
//Параметри підключення
public staticfinal String HOST = " http://80.78.248.202 " ;
public staticfinalint SUCCESS_CODE =200;
public staticfinalint BAD_REQUEST = 400;
public staticfinalint NOT_AUTHENTICATION_CODE =401;
private static final MediaType JSON = MediaType. parse (
"application/json; charset=utf-8" );
private static final String AUTHORIZATION = "Authorization"
;
private static final int TIMEOUT = 30;
private OkHttpClient client ;
public ApiClient() {
client = new Builder()
.writeTimeout( TIMEOUT , TimeUnit. SECONDS )
.readTimeout( TIMEOUT , TimeUnit. SECONDS ) .build();
}
public Response postRequest(String url, String data) throws
IOException
{ //формування запиту
RequestBody body = RequestBody. create ( JSON , data);
Request request = new Request.Builder()
.post(body)
.url(url)
.build();
Response response = client .newCall(request).execute();
return response;
}
public Response postRequest(String url, String token, String
jsonData) throws IOException {
RequestBody body = RequestBody. create ( JSON , jsonData);
String authToken = String. format ( "Bearer %s" , token);
Request request = new Request.Builder()
post(body)
.header( AUTHORIZATION , authToken)
.url(url)
.build();
return client .newCall(request).execute();
}
}
```

Лістинг В.13 - Вихідний код "AuthResult"

```
package susu.com.animallocation.models.apiclient;
public class AuthResult {
private Accountinfo accountinfo ;
private String message ;
private int code ;
public AuthResult( int code, String message) {
this . code = code;
this . message = message;
}
public AuthResult(Accountinfo accountinfo)
```

```
{ //порівняння користувачів
this . accountinfo = accountinfo;
this . code = ApiClient. SUCCESS_CODE ;
this . message = null ;
}
public Accountinfo getAccountinfo() {
return accountinfo ;
}
public String getMessage() {
return message ;
}
public int getCode() {
return code ;
}
}
```

Лістинг В.14 - Вихідний код "Setting"

```
package susu.com.animallocation.models;
import android.content.Context;
import android.content.SharedPreferences;
import susu.com.animallocation.models.apiclient.AccountInfo;
import
susu.com.animallocation.models.apiclient.entities.Animal;
public class Settings {
private static final String ACCOUNT_SETTINGS =
"ACCOUNT_SETTINGS" ;
private static final String NAME = "NAME" ;
private static final String SURNAME = "SURNAME" ;
private static final String ACCESS_TOKEN = "ACCESS_TOKEN" ;
private static final String AMOUNT_ANIMALS =
"AMOUNT_ANIMALS" ;
private static final String IS_LOGIN = "IS_LOGIN" ;
private SharedPreferences connectionSettings ;
private SharedPreferences accountSettings ;
public Settings(Context context) {
accountSettings = context.getSharedPreferences(
ACCOUNT_SETTINGS , Context. MODE_PRIVATE );
}
public void saveAccountInfo(AccountInfo accountInfo) {
if (accountInfo == null ) throw new NullPointerException(
"account info is empty" );
SharedPreferences.Editor editor = accountSettings .edit()
.putString( NAME , accountInfo.getName())
.putString( SURNAME , accountInfo.getSurname())
.putString( ACCESS_TOKEN , accountInfo.getAccessToken())
.putInt( AMOUNT_ANIMALS , accountInfo.getAnimals().size())
.putBoolean( IS_LOGIN , true );
for (Animal animal :
accountInfo.getAnimals()) {
   editor.putString(animal.getIdKey(), animal.getName());
}
editor.apply();
```

```
}
public String getAccessToken() {
return accountSettings .getString( ACCESS_TOKEN , null );
}
public boolean isLogin() {
return accountSettings .getBoolean( IS_LOGIN , false );
}
}
```

Лістинг В.15 - Вихідний код "LoginTask"

```
package susu.com.animallocation.models;
import android.text.TextUtils;
public class Validator {
public static boolean validateEmail(String email) { return
true ;
}
public static boolean validatePassword(String password) {
return true ;
}
private final String pattern = "[1-9]{2}-[1-9]{2}-[1-9]{2}"
;
public static boolean validateBirthday(String birthday) {
return true ;
}
public static boolean isEmptyField(String field) {
return TextUtils. isEmpty(field);
}
```

Лістинг В.15 - Вихідний код " SuperActivity"

```
package susu.com.animallocation.activities;
import android.app.ProgressDialog;
import android.content.Context;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.view.inputmethod.InputMethodManager;
public class SuperActivity extends AppCompatActivity {
public final String EMAIL_KEY = "EMAIL" ;
public final String PASSWORD_KEY = "PASSWORD" ;
protected ProgressDialog progressDialog ;
public void showMessage(String message)
{ //виведення повідомлення
Snackbar. make (getCurrentFocus(), message, Snackbar.
LENGTH_LONG ).show();
}
public void showProgress(String message)
 { // відображення індикатора виконання
if ( progressDialog == null )
progressDialog = new ProgressDialog( this );
progressDialog .setMessage(message);
progressDialog .show();
}
public void hideProgress()
```

```
{ //приховування індикатора
if ( progressDialog != null ) {
progressDialog .dismiss();
}
}
}
```

Лістинг В.16 - Вихідний код “ UpdateAnimalLocationService ”

```
package
susu.com.animallocation.services.UpdateAnimalLocation;
import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.support.annotation.Nullable;
import com.google.android.gms.maps.model.LatLng;
import java.util.Timer;
import java.util.TimerTask;
impo rt
susu.com.animallocation.models.apiclient.animal.location.Upd
ateAnimalLocationCallback ;
import
susu.com.animallocation.models.apiclient.animal.location.Updater
Anima Location;
public class UpdateAnimalLocationService extends Service {
private static final String TAG =
"UpdateAnimalLocationService" ;
private final IBinder binder = новий ServiceBinder();
// coordinates of animal
private LatLng animalLocation ;
// status result getting coordinates of animal
private boolean isSuccessful ;
private String email ;
private String idKey ;
private String accessToken ;
@Override
public void onCreate() { super .onCreate();
}
@Override
public int onStartCommand(Intent intent, int flags, int
startId) { return super .onStartCommand(intent, flags, startId);
}
@Override
public void onDestroy() { super .onDestroy();
}
@Nullable
@Override
public IBinder onBind(Intent intent) {
email = intent.getStringExtra( "EMAIL" );
idKey = intent.getStringExtra( "ID_KEY" );
accessToken = intent.getStringExtra( "TOKEN" );
updateLocation ();
```

```
     return binder ;
     }
     public void updateLocation() { Timer timer = new Timer();
timer .schedule(getInstanceTimerTask(), 0, 10000);
     }
     private TimerTask getInstanceTimerTask() { return new
TimerTask() {
     @Override
     public void run() {
     UpdaterAnima Location. UpdateAnimalLocation ( email , idKey
, accessToken , new UpdateAnimalLocationCallback() { @Override
public void OnComplete(LatLng animalLocation) {
UpdateAnimalLocationService. this . animalLocation =
animalLocation;
     UpdateAnimalLocationService. this . isSuccessful = true ; }
     @Override
     public void OnFail() {
     UpdateAnimalLocationService. this . isSuccessful = false ;
     } });
     } };
     }
     public LatLng getLatLng() { return animalLocation ;
     }
     public boolean isSuccessful() { return isSuccessful ;
     }
     public class ServiceBinder extends Binder { public LatLng
getAnimalCoordinates() {
     return animalLocation ;
     }
     public boolean isSuccessful() { return isSuccessful ;
     }
     }
     }
```

Додаток Г.

Лістинги коду сторінок інтерфейс сторінок авторизації, реєстрації та картки клієнтської програми

Лістинг Г.1 - Початковий код "Login_fragment.xml"

```
<? xml version="1.0" encoding="utf-8" ?>
< android.support.constraint.ConstraintLayout
xmlns:android=" http://schemas.android.com/apk/res/android "
xmlns:tools=" http://schemas.android.com/tools "
and roid:layout_width="match_pa rent"
android:layout_height="match_parent" xmlns:app="
http://schemas.android.com/apk/res-auto "
android:gravity="center_horizontal"
android:orientation="vertical "
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"" >
< ScrollView
android:id="@+id/login_form"
android:layout_width="match_parent"
android:layout_he ight = "wra p_c ont ent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent" >
< LinearLayout
a ndroid:id = "@+id/email_login_fo rm"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical" >
< android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
< AutoCompleteTextV iew
android:id="@+id/email"
android:hint="@string/prompt_email"
android:inputType="textEmailAddress"
style="@style/EditTextStyle" />
</ android.support.design.widget.TextInputLayout >
< android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
< EditText
android:id="@+id/password"
android:hint="@string/prompt_password"
android:inputType="textPassword" style="@style/EditTextStyle" />
</ android.support.design.widget.TextInputLayout >
< Button
android:id="@+id/signInButton"
android:layout_marginTop="16dp"
```

```
     android:t ext = "@string/actio n_s ign_in"
style="@style/ButtonStyle"
android:background="@drawable/btn_anim" / >
     < TextView
     android:id="@+id/registrationButton" and
roid:layout_width="wrap_content" and
roid:layout_height="wrap_content" and
roid:text="@string/registration" android:textSize="20dp"
     android:textColor="@color/colorPrimary"
android:fontFamily="sans-serif-light" and
roid:layout_gravity="center_horizontal"
android:layout_marginTop="@dimen/activity_vertical_margin" / >
     </ LinearLayout >
     </ ScrollView >
     </ android.support.constraint.ConstraintLayout >
```

Лістинг Г.2 - Початковий код "Map_Activity.xml"

```
     <? xml version="1.0" encoding="utf-8" ?> fragment
     xmlns:android=" http://schemas.android.com/apk/res/android "
xmlns:tools=" http://schemas.android.com/tools "
     and roid:id="@+id/map"
     android:name="com.google.android.gms.maps.SupportMapFragment
" android:layout_width="match_parent"
     android:layout_height="match_parent"
     tools:context=".activities.MapActivity" />
```

Лістинг Г.3 - Вихідний код "Registration_fragment.xml"

```
     <? xml version="1.0" encoding="utf-8" ?>
     < LinearLayout
     xmlns:android=" http://schemas.android.com/apk/res/android "
xmlns:app=" http://schemas.android.com/apk/res-auto "
xmlns:tools=" http:/ /schemas.android.com/tools "
     and roid:layout_width="match_pa rent"
android:layout_height="match_parent"
tools:context=".fragment.RegistrationFragment"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity
"@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" >
     < ScrollView
     android:layout_width="match_parent"
android:layout_height="match_parent" >
     < LinearLayout
     android:orientation="vertical"
     android:layout_width="match_parent"
android:layout_height="match_parent" >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
     < AutoCompleteTextV iew
     android:id="@+id/email"
     android:hint="@string/prompt_email"
```

```
android:inputType="textEmailAddress"
style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
     < EditText
     android:id="@+id/password"
     android:hint="@string/prompt_password"
android:inputType="textPassword" style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
     < AutoCompleteTextV iew
     android:id="@+id/userFirstname"
     android:hint="@string/user_name"
style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and
roid:layout_height="wrap_content" >
     < AutoCompleteTextV iew
     android:id="@+id/userSurname"
     android:hint="@string/user_surname"
style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and
roid:layout_height="wrap_content" >
     < EditText
     android:id="@+id/userBirthday"
     android:inputType="date" android:ems="10"
     android:hint="@string/user_birthday"
style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
     < AutoCompleteTextV iew
     android:id="@+id/animalName"
     android:hint="@string/animal_name"
style="@style/EditTextStyle" />
     </ android.support.design.widget.TextInputLayout >
     < android.support.design.widget.TextInputLayout and
roid:layout_width="match_pa rent" and roid:layout_height =
"wrap_content" >
     < AutoCompleteTextV iew
     android:id="@+id/idKey" android:hint="@string/animal_id"
android:inputType="number" style="@style/EditTextStyle" / >
     </ android.support.design.widget.TextInputLayout >
     < Button
     android:id="@+id/signInButton"
```

```
android:layout_marginTop="16dp"
     and roid:text = "@string/registration"
     style="@style/ButtonStyle" />
     </ LinearLayout >
     </ ScrollView >
     </ LinearLayout >
```

Лістинг Г.3 - Початковий код "Root_Activity.xml"

```
     <? xml version="1.0" encoding="utf-8" ?>
     < LinearLayout
     android:orientation="vertical"
     xmlns:android=" http://schemas.android.com/apk/res/android "
and roid:layout_width="match_pa rent"
android:layout_height="match_parent" >
     < ProgressBar
     android:id="@+id/progress"
     android:layout_width="match_parent"
     android:layout_height="4dp"
     style="?android:attr/progressBarStyleHorizontal"
android:indetermi nat eBehavior = "repeat"
android:indeterminate="true"
     android:scaleY="4"
     android:visibility="invisible" />
     < LinearLayout
     android:orientation="vertical"
     android:id="@+id/fragment_container"
     android:layout_width="match_parent"
     android:layout_height="match_parent " />
     </ LinearLayout >
```

Додаток Д.

Загальні АТ -команди

```
AT
OK
AT+GMI
//інформація про модуль
Quectel_Ltd Quectel_MC60E Revision: MTK 0828
OK
AT+GSN
//IMEI модуля
XXXXXXXXXXXXXXX
OK
AT+GCAP
// можливості модуля
+GCAP: +CGSM,+FCLASS
OK
AT+CCLK?
//дата та час пристрою
+CCLK: "04/05/23,00:02:37+00"
OK
AT+CBC
//рівень заряду
+CBC: 0,72,3870
OK
```

Додаток Е.

AT-команди для GSM

```
AT+QSIMDET ?//режим детектування CIM картки
+QSIMDET: 0,0,0
OK
AT+QSIMDET = 1, 1
OK
AT+QSIMDET?
+QSIMDET: 1,1,0
OK
AT+COPS? //оператор
+COPS: 0,0, "Tel"
OK
AT+GSN
861359032495515
OK
AT+CPAS //стан модуля
+CPAS: 0 // готовий до роботи
OK
AT+CREG? //мережа
+CREG: 0,1
OK
AT+CSQ//сигнал
+CSQ: 16,0
OK
ATD +XXXXXXXXXXXX; //дзвінок
OK
ATH
OK
```

Додаток Ж.

AT-команди для GNSS

```
AT+QGNSSC=1 // увімкнути GNSS
OK
AT+QGNSSRD? //інформація про місцезнаходження
+QGNSSRD: $GNRMC,004141.669,V,, ,,,0.00,0.00,010104,,,N*5E
$GNVTG,0.00,T,,M,0.00,N,0.00,K,N*2C
$GNGGA,004141.669,,,,,,,,,,,M,,M,,*5F
$GPGSA,A,1,,,,,,,,,,,,,,,*1E
$GLGSA,A,1,,,,,,,,,,,,,,,*02
$GPGSV,1,1,00*79
$GLGSV,1,1,00*65
$GNGLL,,,,,,004141.669,V,N*6D
OK
AT+QGNSSTS? //час синхронізації
+QGNSSTS: 1
OK
AT+QGNSSEPO = 1 // вкл AGPS
OK
AT+QGEPOAID
OK
```

Додаток 3.

AT-команди для GPRS

```
AT+CGATT?
+CGATT: 1
OK
AT+CGPADDR
+CGPADDR: 1, ""
+CGPADDR: 2, ""
+CGPADDR: 3, ""
OK
AT+QIFGCNT?
+QIFGCNT: 0,0
OK
AT+QIFGCNT=1
OK
AT+QGSMLOC =1 //місце розташування, дата, час
+QGSMLOC: 0, 49.551472, 25.632476,2023/05/30,15:39:33
OK
AT+QICSGP=1,"internet.XXXXXXXXX"
OK
```

Додаток И.

АТ-команди для передачі даних

```
AT+QIOPEN="TCP","80.78.248.203","80" //підключення OK
CONNECT OK
AT+QISEND // Запит
> POST /api/animallocation/put HTTP/1.1
HOST: 80.78.248.202
Accept: application/json
Content-Type: application/json
Content-Length: 104
{"email":"oksanastepanko18122001@gmail.com","idKey":"11111",
"Coordinates":{"latitude": 49.551472,"Long": 25.632476}}
AT+QICLOSE //вимкнення CLOSE OK
```

Додаток К.

Електрична схема модуля

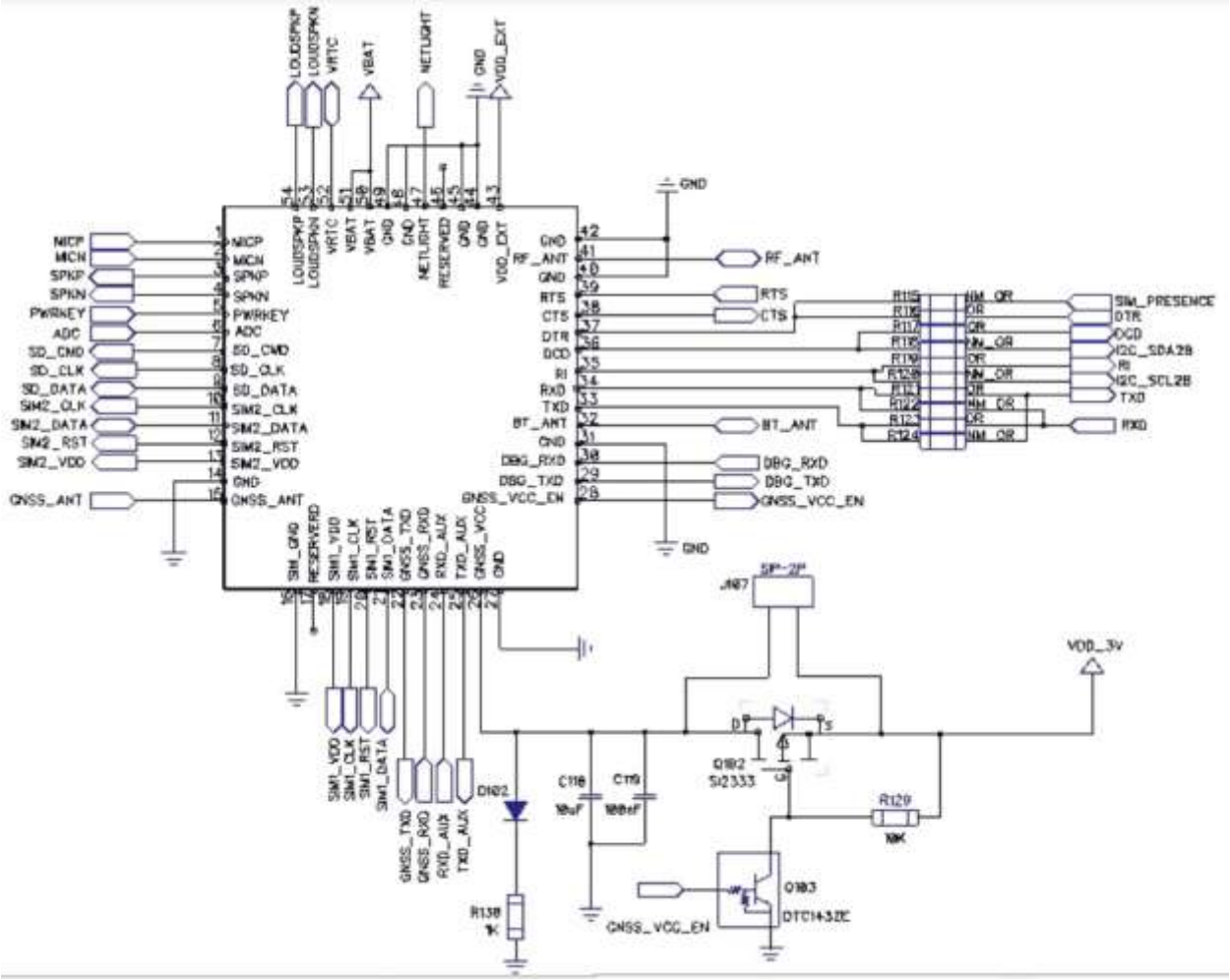

Рисунок К.1 - Електрична схема обв'язки модуля