ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO PARA LOCALIZACIÓN DE MASCOTAS BASADO EN TECNOLOGÍA NFC

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

MARCELO ALEXANDER CARVAJAL TAMAYO

DIRECTOR: ING. RAÚL DAVID MEJÍA NAVARRETE, M.Sc.

Quito, Agosto 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Marcelo Alexander Carvajal Tamayo, bajo mi supervisión.

Ing. Raúl David Mejía Navarrete, M.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Marcelo Alexander Carvajal Tamayo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Marcelo Alexander Carvajal Tamayo

DEDICATORIA

Con amor a mis padres y hermana.

AGRADECIMIENTO

A Dios por ser mi fortaleza y guiarme por el camino correcto.

A mis padres, Marcelo e Irma, quienes con su amor, paciencia y esfuerzo me han permitido concluir esta etapa de mis estudios.

A mis compañeros y amigos quienes me acompañaron en este proceso de aprendizaje.

A mis profesores de la Escuela Politécnica Nacional por sus conocimientos que me impartieron a lo largo de mi formación profesional, en especial al Ingeniero David Mejía, M.Sc. por la guía brindada en este trabajo de titulación y al Ingeniero Pablo Hidalgo, M.Sc. por su apoyo y los consejos otorgados durante mi carrera universitaria.

ÍNDICE DE CONTENIDO

RESUMEN

Este Trabajo de Titulación presenta el desarrollo de un prototipo que permite localizar mascotas, el cual dispone de una aplicación Android y usa la tecnología *Near Field Communication* (NFC). La arquitectura del prototipo es del tipo cliente-servidor y está compuesta por: aplicación cliente Android, base de datos y servicio web. Adicionalmente, el prototipo emplea etiquetas NFC, que se utilizan para identificar a la mascota, así como para que la etiqueta pueda ser leída en caso de que una mascota llegue a perderse y de esa manera se conozcan los detalles de la mascota y se la pueda localizar mediante el prototipo.

Este documento se encuentra organizado de la siguiente manera:

En el capítulo 1 se presenta una descripción general de los conceptos sobre las herramientas y tecnologías empleadas para el desarrollo del prototipo.

En el capítulo 2 se indica la metodología aplicada para el desarrollo del prototipo, en particular se presenta un análisis de las aplicaciones disponibles en el mercado para búsqueda de mascotas, el diseño levantado para el prototipo y la implementación del prototipo.

En el capítulo 3 se presentan los resultados de las pruebas realizadas a cada funcionalidad del prototipo, así como la corrección de errores encontrados.

Finalmente, en el capítulo 4 se encuentran las conclusiones y recomendaciones obtenidas al culminar el desarrollo de este trabajo.

Adicionalmente en los anexos se incluyen: el modelo de entrevista aplicado a propietarios de mascotas, los *sketches* realizados para las interfaces, el modelo de entrevista para validar el funcionamiento del prototipo y sus resultados, el código fuente del prototipo desarrollado, el manual de usuario y el manual instalación del prototipo.

PALABRAS CLAVE: Android, NFC, etiquetas NFC, servicio web, base de datos, mascotas, localización, GPS, Firebase Cloud Messaging, Google APIs

ABSTRACT

This final Degree Project presents the development of a prototype that allows users to find lost pets. The prototype is an Android application and uses Near Field Communication (NFC) technology. The architecture of the prototype is client-server type and is composed of an Android client application, database and web service. In addition, the prototype uses NFC tags, which helps to identify the pet while reading all the details on the tag in case a pet gets lost.

This document is organized as follows:

Chapter 1: General description of the concepts, everything about tools and technologies used for the prototype´s development.

Chapter 2: Methodology applied for the prototype development, a particular analysis of the applications available on the market for searching for pets, prototype design and prototype implementation.

Chapter 3: Test results for each functionality of the prototype, as well as the modifications of all errors found.

Finally, chapter 4 brings together all the conclusions and recommendations for the development of this project.

Additionally, annexes include: the interview model applied to pet owners, the sketches made for the interfaces, the interview model to validate the operation and its results, the source code of the developed prototype, the user manual and the installation manual.

KEYWORDS: Android, NFC, NFC tags, web service, database, pets, localization, GPS, Firebase Cloud Messaging, Google APIs

1. INTRODUCCIÓN

El extravío de mascotas representa una situación de angustia y desesperación que experimentan sus propietarios. Actualmente cuando las mascotas se pierden, los dueños empiezan a buscarlas mediante algunas alternativas, por ejemplo, anuncios en redes sociales, carteles de recompensa, “peinar” el vecindario, etc. Sin embargo, estas alternativas no siempre son viables.

Una gran parte de las mascotas no logran ser rescatadas y por ende con el pasar de los años, genera como consecuencia sobrepoblación de perros y gatos callejeros, así como, la contaminación e inclusive los accidentes de tránsito que estos pueden llegar a provocar, con lo cual se requiere que la sociedad tome conciencia y responsabilidad con su mascota [1].

Para ayudar en la búsqueda de las mascotas, se desarrolla un prototipo que determina la ubicación de estas, mediante el uso de una aplicación basada en Android y etiquetas compatibles con *Near Field Communication* (NFC).

En este capítulo se presentan los objetivos del Trabajo de Titulación y su alcance; además, se describen los conceptos principales necesarios para el desarrollo e implementación del prototipo.

1.1 OBJETIVOS

El objetivo general de este Trabajo de Titulación es:

- Desarrollar un prototipo para la localización de mascotas basado en tecnología NFC.

Los objetivos específicos de este Trabajo de Titulación son:

- Describir los conceptos teóricos referentes al desarrollo del prototipo.
- Diseñar los componentes del prototipo para la localización de mascotas: aplicación Android, base de datos, servicio web y etiqueta compatible con NFC.
- Implementar el prototipo mediante los componentes desarrollados.
- Analizar los resultados de las pruebas realizadas como parte del desarrollo del prototipo.

1.2 ALCANCE

En el presente Trabajo de Titulación se describen las características y funcionamiento del modo de operación lector/escritor de la tecnología NFC, la cual se utiliza en el desarrollo del prototipo para el sistema Android. El prototipo se implementa utilizando el entorno oficial de desarrollo integrado Android Studio.

El prototipo emplea las funcionalidades de NFC mediante el modo de operación lector/escritor entre un teléfono inteligente habilitado para NFC y una etiqueta compatible con esta tecnología, la misma que se encontrará ubicada en el collar de la mascota. El prototipo esta conformado por los siguientes componentes: aplicación Android, base de datos, servicio web y etiqueta compatible con NFC. En la Figura 1.1 se muestra un esquema del prototipo.

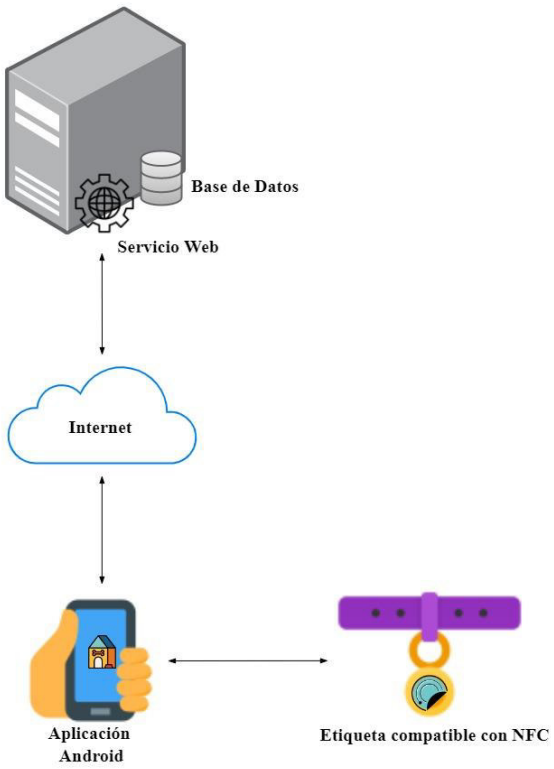

Figura 1.1. Diagrama esquemático del prototipo

El usuario por medio de la aplicación Android podrá registrar a sus mascotas mediante el servicio web en la base de datos, información que también debe ser registrada en una etiqueta compatible con NFC. En dicha etiqueta se almacena el identificador único de la mascota contenido en la base de datos, entre otros atributos que se determinen en el desarrollo del prototipo.

Como parte del análisis de requerimientos se realizarán entrevistas al menos a 5 personas propietarias de mascotas, así también se analizan al menos dos aplicaciones existentes en el mercado para búsqueda de mascotas.

Adicionalmente en el prototipo se definirán dos tipos de roles de usuario: el rol propietario de la mascota y el usuario del sistema. El rol propietario de la mascota podrá realizar un CRUD (crear, leer, actualizar y eliminar) de la información de sus mascotas, autenticarse, almacenar información en las etiquetas; además, tendrá la opción de reportar que su mascota se extravió. El rol de usuario del sistema podrá registrarse en el prototipo y notificar que ha encontrado una mascota extraviada.

Para notificar que ha encontrado una mascota, el usuario deberá leer la información de la etiqueta compatible con NFC, que lleva la mascota consigo, y posteriormente enviará una notificación a través de Internet con los datos de la posición geográfica capturados por el GPS del teléfono inteligente hacia la aplicación Android del propietario de la mascota. También podrá conocer los datos del propietario, por ejemplo, número de celular en caso de que no disponga de conexión a Internet.

1.3 MARCO TEÓRICO

En esta sección se describen los conceptos teóricos relacionados a la tecnología NFC, el sistema operativo Android, los servicios web, la base de datos, entre otras herramientas necesarias para el desarrollo e implementación del prototipo.

1.3.1 NEAR FIELD COMMUNICATION

Near Field Communication (NFC) [2] es una tecnología de comunicación inalámbrica de corto alcance que opera a una frecuencia de 13,56 MHz con una velocidad de transmisión de hasta 424 Kbps, y emplea acoplamiento inductivo[1]. Esta tecnología es una rama de la tecnología *Radio Frequency Identification* (RFID)[2] de alta frecuencia[3] [2], [3].

[1] Acoplamiento inductivo: se denomina al efecto que se produce al inducir una corriente eléctrica cuando el campo magnético de un dispositivo entra en contacto con un material conductor.

[2] *Radio Frequency Identification* (RFID): se refiere a la banda del espectro electromagnético que abarca un rango de frecuencias que va desde los 3 MHz hasta los 30 MHz.

[3] Alta frecuencia: se refiere a la banda del espectro electromagnético que abarca un rango de frecuencias que va desde los 3 MHz hasta los 30 MHz.

La comunicación NFC se efectúa cuando el usuario interactúa con un objeto inteligente, como una etiqueta NFC, un lector NFC u otro dispositivo habilitado para NFC a través de un teléfono inteligente, hasta una distancia de pocos centímetros (ver Figura 1.2).

El intercambio de información se realiza cuando dos dispositivos se acercan lo suficiente o por contacto físico [2].

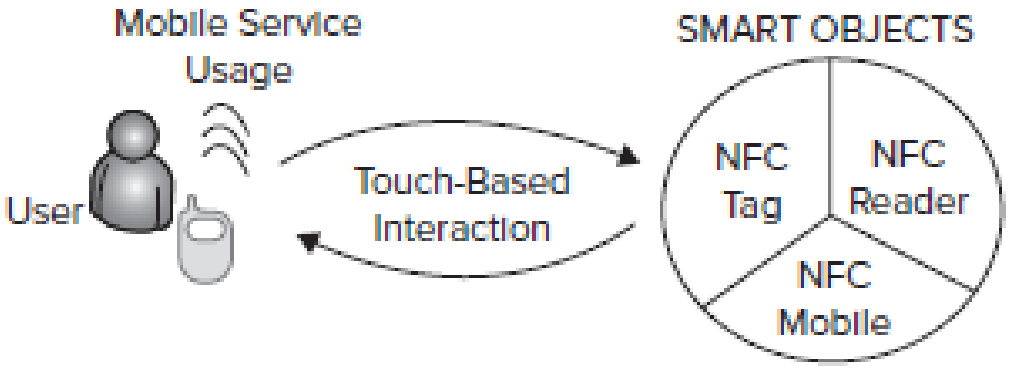

Figura 1.2. Interacción del usuario en la comunicación NFC [2]

1.3.1.1 Dispositivos NFC

Los dispositivos que emplea la tecnología NFC son los siguientes [5]:

- **Teléfono inteligente habilitado para NFC**: son los dispositivos NFC más importantes debido a su amplia gama de aplicaciones en salud, educación, servicios basados en ubicación, etc.
- **Lector NFC:** un lector NFC se encuentra en la capacidad de transferir datos a otro componente de un sistema. Un ejemplo de lector NFC es una terminal de pago[4].
- **Etiqueta NFC:** es cualquier tipo de etiqueta inteligente que cumple con lo establecido por el Fórum NFC[5].

Los dispositivos NFC se pueden clasificar en dos tipos dependiendo de la forma en que proporcionan energía a sus componentes, estos son: dispositivos activos y dispositivos pasivos.

[4] Terminales de pago: son dispositivos electrónicos que se encargan de procesar pagos a través de tarjetas de crédito, débito u otra forma de pago.

[5] Fórum NFC: es una asociación de la industria sin fines de lucro encargada de desarrollar soluciones NFC y así contribuir con el avance de la tecnología NFC.

1.3.1.1.1 Dispositivos Activos

Un dispositivo activo se caracteriza por tener una fuente de alimentación integrada, como una batería, y por lo tanto pueden generar su propio campo de radio frecuencia[6] e iniciar la comunicación [2].

Existen dos tipos de roles de comunicación para cada dispositivo NFC acorde a la función que este realizando en ese momento, ya sea el enviar o recibir información, estos roles son los siguientes [2]:

- **Iniciador:** Se encarga de iniciar la comunicación NFC. Este rol está representado por los dispositivos activos ya que requieren de una fuente de alimentación para iniciar la comunicación.
- **Destino:** Este rol responde a las peticiones del iniciador.

En la Tabla 1.1 se presentan las posibles interacciones de los dispositivos NFC.

Tabla 1.1. Posibles interacciones de los dispositivos NFC [2]

Iniciador	Destino
Teléfono inteligente habilitado para NFC	Etiqueta NFC
Teléfono inteligente habilitado para NFC	Teléfono inteligente habilitado para NFC
Lector NFC	Teléfono inteligente habilitado para NFC

1.3.1.1.2 Dispositivos Pasivos

Un dispositivo pasivo responde únicamente a los requerimientos de un dispositivo activo, no tienen una fuente de alimentación. Las etiquetas NFC son los únicos dispositivos pasivos en la tecnología NFC [2].

1.3.1.2 Etiquetas NFC

Las etiquetas NFC (ver Figura 1.3), también conocidas como etiquetas inteligentes o etiquetas de información, son pequeños circuitos impresos que actúan como un almacén de datos de poca memoria que se encuentran conectados a una antena; usualmente este

[6] Campo de radio frecuencia: se genera cuando una corriente alterna pasa a través de un material que posibilita la transmisión de electricidad.

circuito está sostenido por un papel adhesivo, el cual se puede adherir a cualquier tipo de superficie plana [4].

El Fórum NFC, con el fin de garantizar la interoperabilidad entre las etiquetas que ofrecen los diferentes proveedores y los fabricantes de dispositivos, ha establecido una clasificación para las etiquetas NFC. Actualmente, existen cinco tipos de etiquetas NFC [5], que se resumen en la Tabla 1.2.

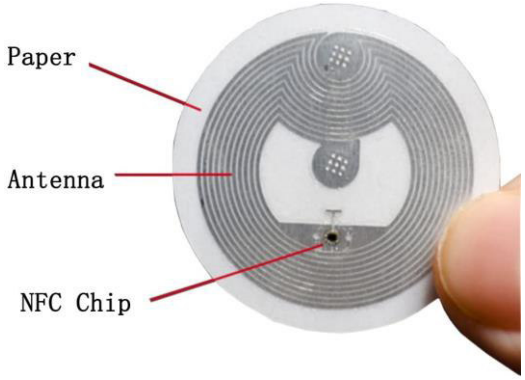

Figura 1.3. Etiqueta NFC [6]

Tabla 1.2. Tipos de etiquetas NFC y sus características [2], [4]

Tipo de etiqueta NFC	Estándar	Memoria	Velocidad de transmisión (Kbps)	Productos disponibles en el mercado
Tipo 1	ISO 14443 A	Hasta 2 KB	106	Innovision Topaz, Broadcom
Tipo 2	ISO 14443 A	Hasta 2 KB	106	NXP MIFARE Ultralight, NTAG 203, NTAG 210, NTAG 212, NTAG 213/215/216, NTAG I2C
Tipo 3	JIS X 6319-4	Hasta 1 MB	212, 424	Sony FeliCa
Tipo 4	ISO 14443 A/B	Hasta 32 KB	106 a 424	NXP DESFire, NXP SmartMX-JCPOP
Tipo 5	ISO 15693	Hasta 64 KB	26.48	NXP ICODE Series

1.3.1.3 Modos de Operación NFC

Existen los siguientes modos de operación NFC: lector/escritor, *peer-to-peer* y emulación de tarjeta. Estos modos de operación tienen su propia interfaz de comunicación, la cual está especificada en el estándar ISO/IEC 14443[7].

1.3.1.3.1 Modo Lector/Escritor

Este modo de operación permite leer o escribir información en las etiquetas NFC a través de la interacción de un teléfono inteligente habilitado para NFC. En el modo lector, el iniciador lee la información del dispositivo destino, el cual contiene los datos solicitados. En el modo escritura, el teléfono inteligente actúa como iniciador y escribe datos en el dispositivo destino [7].

En la Figura 1.4 se presenta una representación gráfica del modo lector/escritor.

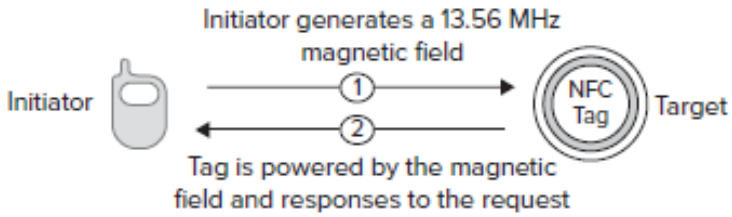

Figura 1.4. Diagrama esquemático del modo de operación lector/escritor [2]

Los pasos (ver Figura 1.5) del modo de operación lector/escritor son los siguientes [2]:

1. **Solicitud de lectura:** El usuario solicita la información requerida acercando un teléfono inteligente a una etiqueta NFC, la cual se puede encontrar adherida a un producto, como un poster, una caja, etc.
2. **Transferencia de datos:** La información almacenada en la etiqueta NFC se transmite al teléfono inteligente.
3. **Procesamiento dentro del dispositivo:** Una vez transmitidos los datos al teléfono inteligente, se los pueden usar para varios propósitos tales como: abrir una aplicación, mostrar información al usuario o utilizar la información para propósitos adicionales.

[7] ISO/IEC 14443: es un estándar relacionado con las tarjetas de proximidad o tarjetas con circuitos integrados.

4. **Uso de servicios adicionales:** Este es un paso opcional, mediante el cual, el teléfono inteligente emplea sus funciones avanzadas como la conexión a un proveedor de servicios a través de Internet.
5. **Solicitud de escritura:** El usuario solicita escribir información en una etiqueta NFC acercando el teléfono inteligente a esta; si la etiqueta NFC ya contiene datos, estos son eliminados y reemplazados por la nueva información.
6. **Reconocimiento:** La etiqueta NFC envía al usuario los datos de reconocimiento para informar si la operación fue exitosa.

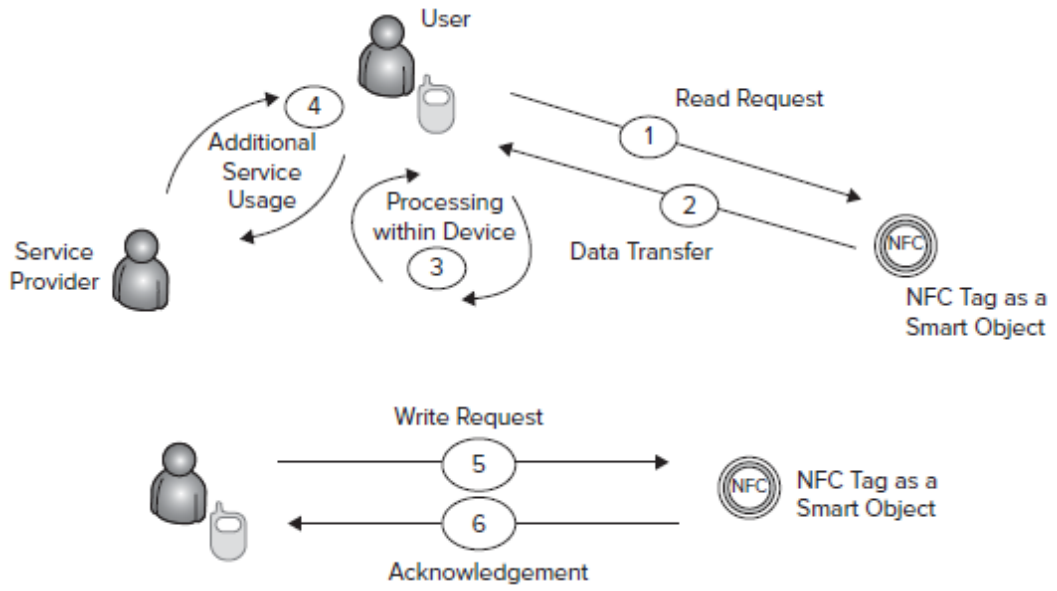

Figura 1.5. Modelo de uso genérico del modo de operación lector/escritor [2]

1.3.1.3.2 Modo Peer-to-Peer

En el modo *peer-to-peer*, el intercambio de información se realiza entre dos teléfonos inteligentes habilitados para NFC. Este modo se basa en el estándar ISO/IEC 18092[8]. Ambos dispositivos funcionan en modo activo durante la comunicación y los datos se envían a través de un canal *half* dúplex[9]; es decir, el segundo dispositivo únicamente puede transmitir cuando el primer dispositivo NFC haya finalizado su transmisión [4] .

[8] ISO/IEC 18092: es un estándar que define los modos de comunicación para la interfaz y el protocolo de comunicación de campo cercano (NFCIP 1).

[9] Canal *half* dúplex: se refiere a un canal de comunicaciones que permite intercambiar la transmisión en dos direcciones, pero no en ambas direcciones de manera simultánea.

En la Figura 1.6 se presenta una representación gráfica del modo *peer-to-peer*.

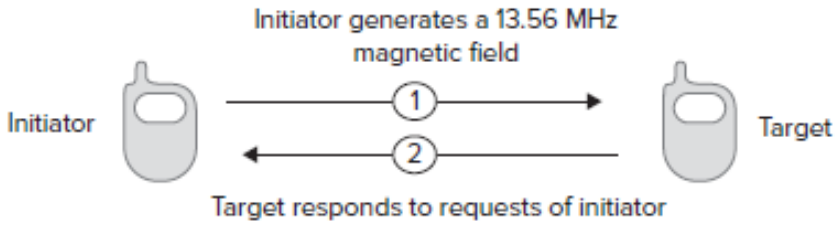

Figura 1.6. Diagrama esquemático del modo de operación peer-to-peer [2]

1.3.1.3.3 Modo Emulación de Tarjeta

El modo de emulación de tarjeta permite que un teléfono inteligente habilitado para NFC funcione como una tarjeta NFC. Ejemplos de tarjetas NFC emuladas por teléfono inteligentes son las tarjetas de crédito, débito y de fidelización [2].

En este modo de operación, el teléfono inteligente con NFC no genera su propio campo de radiofrecuencia; el lector NFC crea este campo en su lugar. Este modo es compatible con los estándares ISO/IEC 14443 y FeliCa[10] [2].

En la Figura 1.7 se presenta una representación gráfica del modo de emulación de tarjeta.

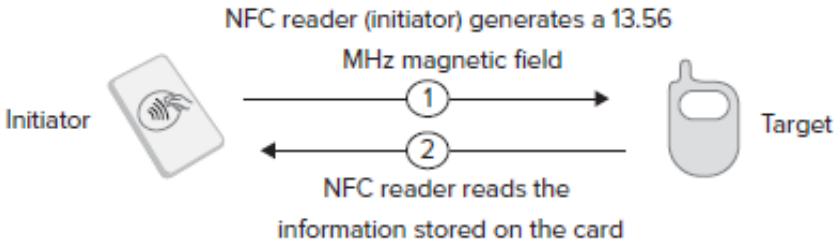

Figura 1.7. Esquema del modo de operación emulación de tarjeta [2]

1.3.1.4 NFC Data Exchange Format

NDEF (NFC *Data Exchange Format*) es una especificación de formato de datos binario estandarizado por el Fórum NFC; este formato se utiliza para describir como encapsular una o más cargas útiles definidas por la aplicación en un único mensaje. Un mensaje NDEF se intercambia entre dispositivos compatibles con NFC [8].

[10] FeliCa: es un sistema de tarjetas inteligentes RFID desarrollado por Sony.

Un mensaje NDEF contiene uno o más registros NDEF. Cada registro NDEF está compuesto por la carga útil (*payload*), la cual puede almacenar hasta $2^{32} - 1$ octetos, y un encabezado (*header*). El número máximo de registros NDEF que se pueden almacenar no se encuentra definido [2].

En la Figura 1.8 se presenta un ejemplo de un mensaje NDEF compuesto por tres registros, uno para el nombre de contacto telefónico, otro para un número de teléfono y el último para una dirección.

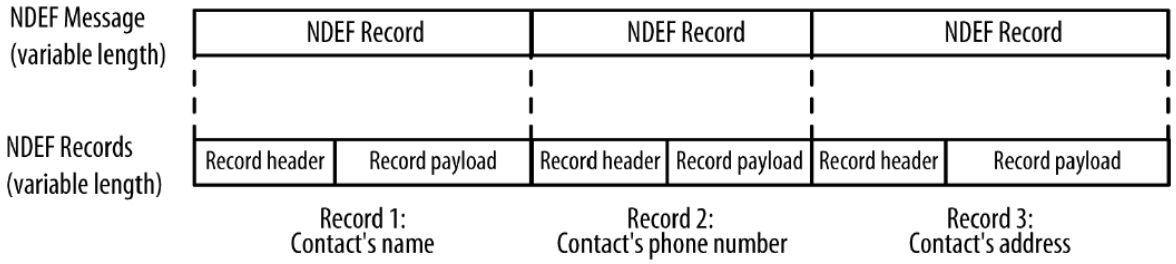

Figura 1.8. Estructura de un mensaje NDEF compuesto por tres registros [9]

Como se observa en la Figura 1.9, el tamaño de los registros NDEF es de longitud variable; cada uno compuesto por diferentes campos en su encabezado, los mismos que se usan para describir al registro e indicar su ubicación en el mensaje.

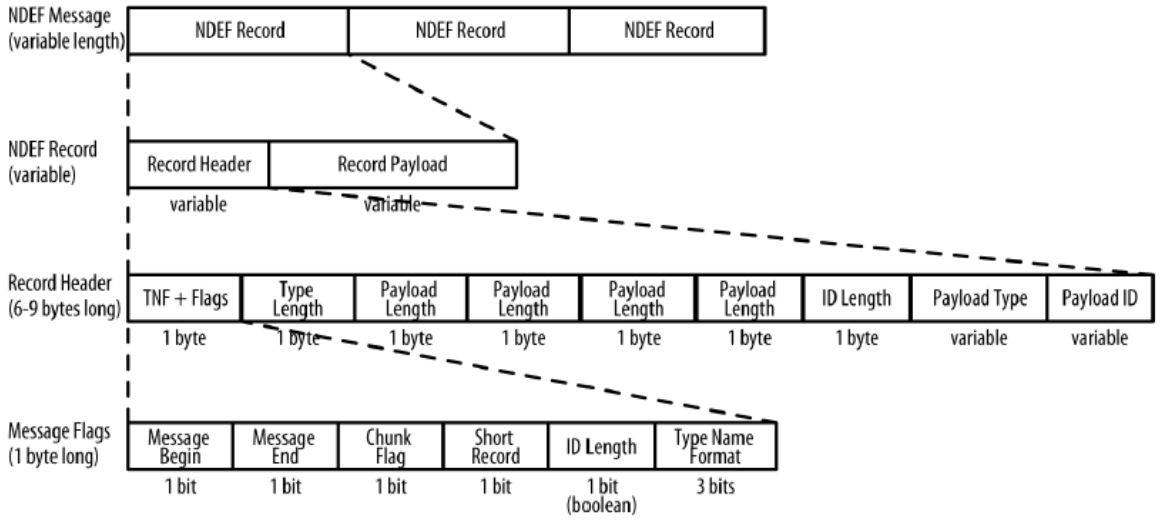

Figura 1.9. Estructura completa de un mensaje NDEF [9]

El encabezado (ver Figura 1.10) de un registro NDEF se compone de los siguientes campos [2]:

- ***Message Begin* (MB):** La bandera MB es un campo de 1 bit que indica el inicio de un mensaje NDEF. En el primer registro del mensaje NDEF; la bandera MB se establece en 1L.
- ***Message End* (ME):** La bandera ME es un campo de 1 bit que indica el final de un mensaje NDEF. Esta bandera se establece en 1L para indicar que es el último registro en un mensaje NDEF.
- ***Chunk Flag* (CF):** La bandera CF es un campo de 1 bit que indica si la carga útil está fragmentada.
- ***Short Record* (SR):** La bandera SR es un campo de 1 bit que indica si la longitud del campo *PAYLOAD* es un único octeto o 4 octetos.
- **IL:** La bandera IL indica que el campo *ID_LENGTH* está presente en el encabezado como un único octeto.
- **TNF (*Type Name Format*):** Es un campo de 3 bits que representa la estructura del valor del campo *TYPE*. En la Tabla 1.3 se presentan los valores del campo TNF en formato hexadecimal junto con el nombre del tipo de formato al que representan.

Tabla 1.3. Valores del campo TNF [2]

Nombre del tipo de formato	Valor
Empty	0x00
NFC Forum well-known-type	0x01
Media-type	0x02
Absolute URI	0x03
NFC Forum external-type	0x04
Unknown	0x05
Unchanged	0x06
Reserved	0x07

- **TYPE_LENGTH:** Es un campo de 8 bits que especifica la longitud en octetos del campo *TYPE*.
- **ID_LENGTH:** Es un campo de 8 bits que especifica la longitud en octetos del campo ID.
- **PAYLOAD_LENGTH:** Es un campo que indica la longitud en octetos del campo *PAYLOAD*. El tamaño del campo está establecido por el valor de la bandera SR.

- **TYPE:** Este campo describe el tipo de dato que almacena el campo *PAYLOAD*.
- **ID:** El valor de este campo es un identificador en forma de referencia URI. Los registros de fragmentos intermedios o finales no tienen un campo ID [10].
- **PAYLOAD:** Este campo contiene los datos de la aplicación.

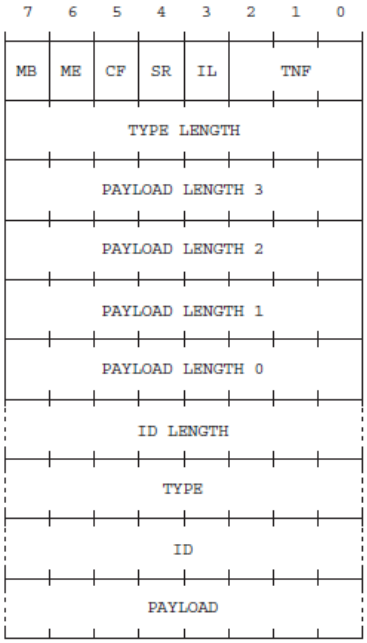

Figura 1.10. Estructura de un registro NDEF [2]

1.3.2 SISTEMA OPERATIVO ANDROID

Android es un sistema operativo de código abierto desarrollado inicialmente por Android Inc., compañía de software con sede en Silicon Valley antes de ser adquirida por Google en 2005. Desde entonces, Google y la Open Handset Alliance (OHA)[11] han seguido desarrollando nuevas actualizaciones para Android con el propósito de mejorar la experiencia del usuario y corregir errores [11].

El sistema operativo Android se basa en el kernel de Linux y su primera versión comercial llegó al mercado en 2008 [11]. El lenguaje de programación Java se usa principalmente para desarrollar aplicaciones para Android, sin embargo, se pueden usar otros lenguajes. Este sistema está diseñado para dispositivos, como teléfonos inteligentes, relojes inteligentes, automóviles con el sistema Android Auto, entre otros [12].

[11] Open Handset Alliance (OHA): es una alianza comercial formada por 84 empresas de tecnología y telefonía móvil dedicada a la innovación y mejora de la telefonía móvil y experiencia del usuario.

La arquitectura del sistema operativo Android versión 11 (ver Figura 1.11) está conformada por varias capas, las cuales se describen a continuación [13]:

- ***Kernel* de Linux:** esta capa maneja los servicios básicos, incluidos los controladores de hardware y la gestión de energía.
- **Capa de abstracción de hardware:** se refiere a una capa de abstracción entre los servicios del sistema y sus controladores de hardware.
- **Librerías nativas C/C++:** Android incluye varias bibliotecas C/C++ que proporcionan soporte para el desarrollo de Android.
- **Android *Runtime*:** es el motor que ejecuta las aplicaciones para Android.
- ***Framework* de la API de Java:** esta capa administra la interfaz de usuario y los recursos de la aplicación; proporciona las clases o plantillas para desarrollar aplicaciones Android.
- **Aplicaciones del sistema:** esta capa aloja las aplicaciones del sistema y las desarrolladas por terceros; además, utiliza las clases y los servicios disponibles del *framework* de la API de Java.

Las aplicaciones de Android constan de componentes vinculados entre sí por un manifiesto de aplicación que describe cada componente y su forma de interacción. Los siguientes componentes incluyen los componentes básicos para todas las aplicaciones de Android [13]:

- **Actividades:** representan a la interfaz del usuario de una aplicación. Las actividades usan fragmentos y vistas para desplegar información y responder a las interacciones del usuario.
- **Servicios:** se utilizan para realizar tareas de ejecución prolongada o aquellas que no requieren interacción del usuario.
- ***Intents*:** se utilizan para iniciar y detener Actividades y Servicios, como también para transmitir datos en todo el sistema.
- ***Broadcast Receivers*:** se utilizan para permitir que la aplicación escuche y reciba *Intents* que coincidan con criterios previamente especificados.
- ***Content Providers*:** son utilizados para compartir datos entre aplicaciones.
- **Notificaciones:** permiten alertar al usuario sobre eventos de una aplicación sin necesidad de interrumpir la actividad actual.

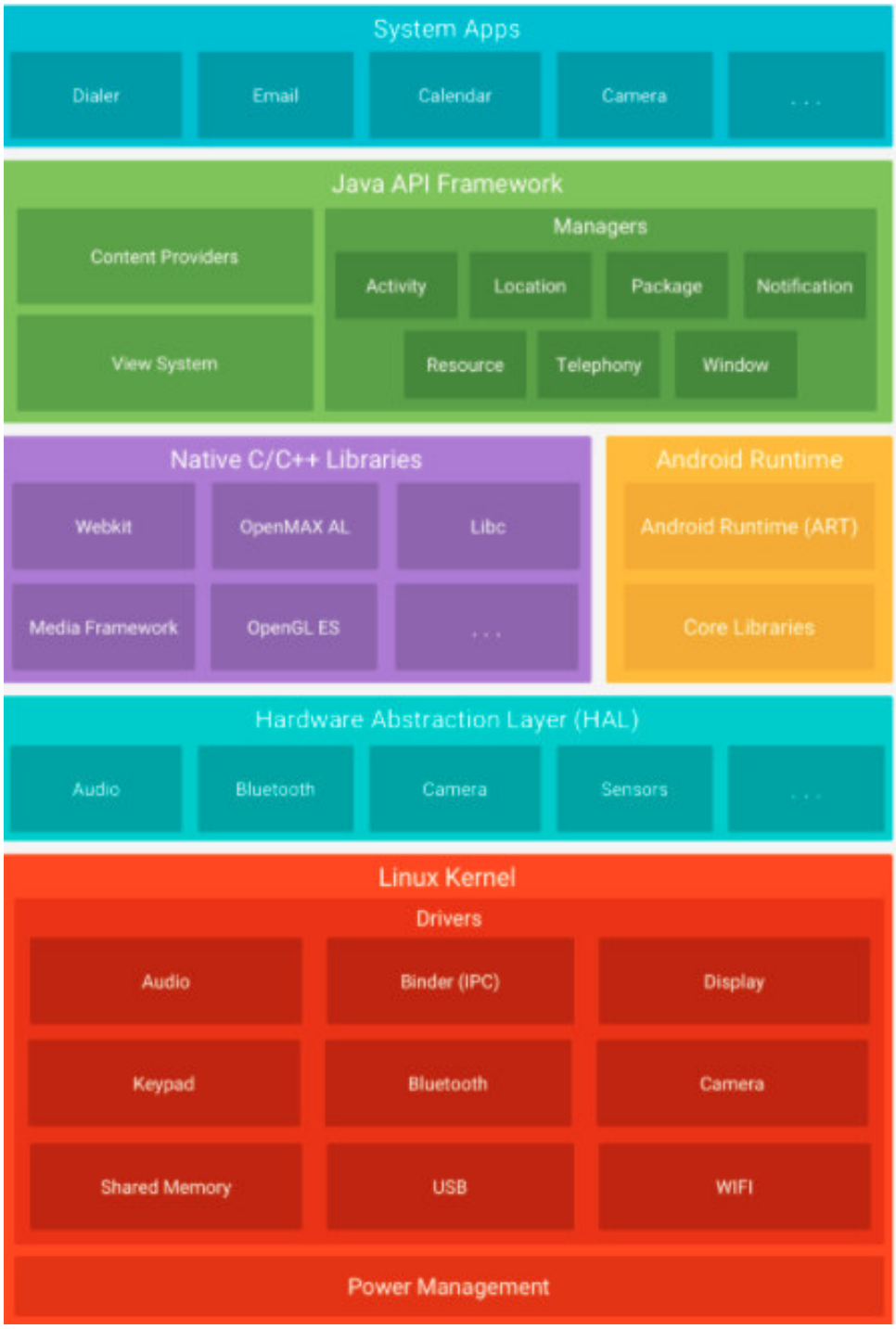

Figura 1.11. Arquitectura del sistema operativo Android [14]

1.3.3 APPLICATION PROGRAMMING INTERFACE

Una API (*Application Programming Interface*) es un conjunto de funciones que dan acceso a cierta funcionalidad de un software en específico. En la actualidad, las API, en su mayoría, utilizan el protocolo HTTP para la transferencia de recursos o archivos entre equipos

interconectados a una red. Este protocolo utiliza métodos y códigos de respuesta para comunicar al servidor o cliente de una determinada acción [15].

Los métodos y códigos de estado HTTP comúnmente utilizados se presentan en la Tabla 1.4 y en la Tabla 1.5, respectivamente.

Tabla 1.4. Ejemplos de métodos HTTP [16]

Método	Descripción
GET	Obtiene un recurso o una colección
POST	Crea un nuevo recurso
PUT	Actualiza un recurso existente
DELETE	Elimina un recurso

Tabla 1.5. Ejemplos de códigos HTTP [17]

Código	Descripción
200	Indica que un recurso o colección existe
201	Indica que se creó un recurso exitosamente
204	Representa un resultado exitoso sin devolver ningún valor.
304	Indica que un recurso o colección ha cambiado
401	Indica que el cliente debe estar autorizado para acceder a un recurso
404	Informa que un recurso solicitado no existe
500	Informa un error interno en el servidor
503	Indica que el servidor se encuentra fuera de servicio

1.3.3.1 Servicio Web

Un servicio web es un tipo de API que permite la interoperabilidad entre varios sistemas de manera independiente al lenguaje de programación o plataforma con el que fue desarrollado. El World Wide Web Consortium (WC3)[12] define un servicio web como un sistema de software diseñado para soportar interacciones entre una máquina y otra

[12] World Wide Web Consortium (WC3): es una comunidad internacional dedicada al desarrollo de estándares para afianzar el crecimiento de la web.

máquina a través de la red; es decir, los servicios web proporcionan una forma estándar de interacción entre aplicaciones de software que se ejecutan en diferentes plataformas [18].

Por lo general, los servicios web se encuentran expuestos a través de Internet a la espera de peticiones por parte de cualquier dispositivo. Cuando recibe una petición de un dispositivo, a través de una red, el servicio web devuelve los recursos solicitados. Estos recursos pueden ser texto en formato JSON o XML, o imágenes, archivos de audio, etc. [19].

1.3.3.2 Google Places API

Google *Places* API es un servicio que devuelve información sobre lugares, como establecimientos, ubicaciones geográficas o puntos de referencia destacados, a través de solicitudes HTTP [20].

Esta API ofrece las siguientes opciones [20]:

- Lista de lugares cercanos según la ubicación de un usuario.
- Información detallada acerca de un lugar en específico, como las reseñas de usuarios.
- Acceso a fotos relacionadas a un lugar.
- Completar el nombre o la dirección de un lugar conforme el usuario escribe.

1.3.3.3 Google Maps API

La API de Google *Maps* permite a los usuarios visualizar el mundo real, por medio de mapas estáticos o interactivos, los cuales pueden personalizarse e incorporarse en sitios web o aplicaciones móviles [21].

Algunos de los servicios que ofrece Google *Maps* API son los siguientes [21]:

- **SDK[13] *Maps* para Android**: permite agregar mapas a una aplicación para Android.
- **SDK *Maps* para iOS:** permite agregar mapas a una aplicación para iOS.
- ***Maps* JavaScript API:** permite agregar mapas interactivos y personalizarlos.

[13] SDK: es una colección de herramientas de desarrollo de software y bibliotecas necesarias para desarrollar aplicaciones informáticas.

- **API *Street View*:** permite incorporar una vista panorámica de algún sitio a una página web.

1.3.3.4 Firebase Cloud Messaging

Firebase *Cloud Messaging* (FCM) es una solución de mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita. Esto lo convierte en una herramienta adecuada para impulsar campañas de *marketing*, participación de la audiencia o programas de fidelización [22].

Este servicio proporciona diferentes funcionalidades, entre ellas [23]:

- Enviar mensajes a dispositivos específicos.
- Enviar mensajes a un grupo de dispositivos.
- Crear y eliminar una suscripción a un tema.

1.3.4 BASE DE DATOS

Una base de datos es una colección organizada de información o datos. Los datos dentro de los tipos más comunes de bases de datos se modelan en tablas, compuestas por filas y columnas, para una consulta y procesamiento de datos eficientes.

Además, permiten realizar un CRUD (crear, leer, actualizar, eliminar) de la información contenida. La mayoría de las bases de datos utilizan el lenguaje SQL (*Structured Query Language*) para escribir y consultar datos [24].

1.3.4.1 Gestor de Base de Datos

Una base de datos generalmente está controlada por un programa de software de base de datos conocido como sistema de administración de base de datos (DBMS).

Un DBMS sirve como una interfaz entre la base de datos y sus consumidores, permitiendo recuperar y actualizar la información, así como organizar y optimizar la información.

Un DBMS también facilita la supervisión y el control de las bases de datos, lo que permite realizar diversas actividades administrativas, como la supervisión del rendimiento, la copia de seguridad y la recuperación [24].

Algunos DBMS populares son MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro y Oracle Database [24].

1.3.4.2 Base de Datos Relacionales

Una base de datos relacional es un tipo de base de datos que almacena y brinda acceso a conjuntos de datos que se encuentran relacionados entre sí y que se representan mediante tablas.

En una base de datos relacional, cada fila de una tabla es un registro con un identificador único, el cual permite establecer relaciones entre las tablas. Las columnas de una tabla contienen atributos de sus datos y, por lo general, cada registro tiene un valor para cada atributo [25].

1.3.4.3 Structured Query Language

SQL (*Structured Query Language*) es un lenguaje utilizado por la mayoría de las bases de datos relacionales para consultar, recuperar, definir datos y proporcionar control de acceso. SQL fue desarrolló inicialmente por IBM en el año 1970 para la base de datos Oracle [24].

1.3.5 HERRAMIENTAS DE DESARROLLO DE SOFTWARE

1.3.5.1 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA[14].

Algunas de las funcionalidades que ofrece Android Studio son las siguientes [13]:

- Integración con Github y su sistema de control de versiones.
- Simplificación en la creación de nuevos proyectos y plantillas de las aplicaciones a través del asistente de proyectos de Android.
- Creación y administración de dispositivos virtuales bajo el sistema operativo Android con restricciones de hardware y memoria establecidas.
- Depuración de código en tiempo de ejecución.

[14] IntelliJ IDEA: es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos.

1.3.5.2 XAMPP

XAMPP es un conjunto de aplicaciones que permite levantar un servidor web multiplataforma, que permite a los desarrolladores probar sus programas de forma local antes de publicarlos en un servidor en Internet.

XAMPP brinda un entorno de desarrollo adecuado para probar y verificar el funcionamiento de proyectos basados en Apache[15], Perl[16], MySQL[17] y PHP[18] [26].

1.3.5.3 GIT

Git es un sistema de control de versiones gratuito y de código abierto. Este sistema hace un seguimiento de los cambios realizados en un código conforme se lo modifica. Si durante la codificación se produce un error fatal, GIT permite al desarrollador volver a un estado estable del código. También permite mostrar los cambios realizados en el código a lo largo de su desarrollo [27].

1.3.5.4 GITHUB

Github es un repositorio en línea que permite administrar y controlar las versiones de un proyecto. Es ampliamente utilizado por desarrolladores para almacenar código fuente y compartirlo con los miembros de su empresa e inclusive habilitar un acceso público en el código, lo que brinda a personas de todo el mundo la oportunidad de colaborar y mejorar un software [28].

1.3.6 METODOLOGÍA KANBAN

Kanban es un sistema visual para administrar el trabajo a medida que se avanza en un proceso. El primer sistema Kanban fue desarrollado para el fabricante de automóviles japonés Toyota. El objetivo de Kanban es identificar posibles “cuellos de botella” en el proceso y solucionarlos para que el desarrollo de un trabajo pueda fluir de manera rentable y a una velocidad o rendimiento óptimos [29] .

---

[15] Apache: es un servidor web de código abierto, multiplataforma y gratuito.

[16] Perl: es un lenguaje de programación desarrollado inicialmente para la manipulación de texto.

[17] MySQL: es un sistema de gestión de base de datos relacionales.

[18] PHP: es un lenguaje de programación, del lado del servidor, adecuado para el desarrollo web.

Para visualizar el flujo de trabajo se establece un tablero denominado tablero Kanban, el cual consta de notas adhesivas o tarjetas. En un modelo tradicional de tablero Kanban, existen tres columnas, como se muestra en la Figura 1.12. Cada tarea se representa como una tarjeta, y estas se desplazan por las columnas *To Do*, *Doing* y *Done* a medida que se ponen en cola, se desarrollan y se completan, respectivamente. La columna *To Do* suele denominarse *backlog*, ya que presenta todas las tareas pendientes por cumplir con el desarrollo de un proyecto.

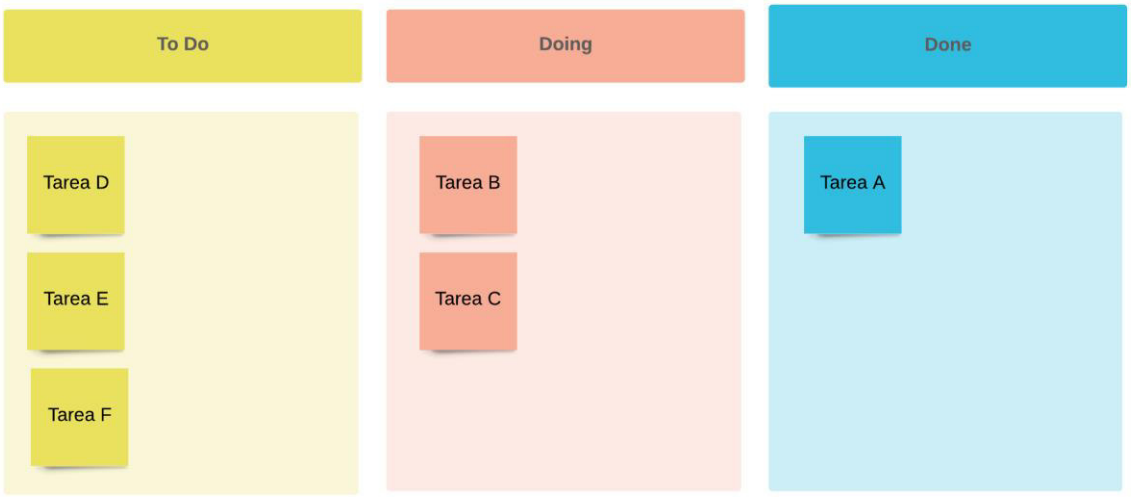

Figura 1.12. Ejemplo de tablero Kanban

1.3.7 RELACIÓN CON TRABAJOS ANTERIORES DEL ÁREA

El presente Trabajo de Titulación se encuentra relacionado con los trabajos de titulación titulados “IMPLEMENTACIÓN DE UN SISTEMA PROTOTIPO DE ADMINISTRACIÓN DE INVENTARIO DE BIENES UTILIZANDO UNA APLICACIÓN MOVIL CON SISTEMA OPERATIVO ANDROID Y TECNOLOGÍA NFC PARA LA ESCUELA POLITÉCNICA NACIONAL” y “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PROTOTIPO DE ETIQUETAS INFORMATIVAS DE USO ACADÉMICO UTILIZANDO UNA APLICACIÓN ANDROID MEDIANTE COMUNICACIÓN NFC”.

En el primer trabajo se realizó la implementación de un prototipo para la administración de inventario de bienes utilizando una aplicación móvil, una aplicación web y tecnología NFC. El sistema está compuesto por: servidor web, base de datos, aplicación desarrollada para Android y aplicación web. Para la identificación de cada bien se emplean etiquetas NFC, las cuales se encuentran asociadas a un código contenido en una base de datos juntamente con las características del bien.

En el segundo trabajo se desarrolló un prototipo que brinda información sobre horarios, cursos, prácticas de laboratorio, profesores y horarios de consulta mediante comunicación NFC. El sistema está conformado por: una base de datos, una aplicación móvil, una aplicación web y un servidor web. La aplicación móvil está desarrollada en el sistema operativo Android mediante las funcionalidades de la tecnología NFC, permite leer las etiquetas y obtener su información. Adicionalmente, la aplicación web permite administrar la información contenida en la base de datos.

La principal diferencia que existe con los trabajos anteriormente mencionados radica en la problemática planteada: el primero implementó un prototipo para la administración de inventario de bienes mediante el uso de la tecnología NFC, el segundo desarrolló un prototipo con el fin de brindar información académica a los estudiantes mediante comunicación NFC; mientras que para este trabajo de titulación se requiere utilizar la tecnología NFC como una alternativa que permita localizar mascotas extraviadas, además para este trabajo de titulación se utilizará geolocalización.

2. METODOLOGÍA

En el presente Trabajo de Titulación se empleó la metodología Kanban. Con base en esta metodología se definieron las tareas pendientes a realizar, estas tareas fueron colocadas en el Tablero Kanban.

En este capítulo se presenta un análisis comparativo de las aplicaciones WizaPet, Tractive GPS Pet Finder y Welppy, las mismas que serán la base para establecer las funcionalidades del prototipo propuesto.

Como parte del diseño del prototipo, se realizaron entrevistas a propietarios de mascotas, a través de estas entrevistas se establecieron las historias de usuario y los requerimientos del prototipo; adicionalmente con base en la información levantada se generaron los casos de usos del prototipo y se incluyeron las tareas a desarrollar en el tablero Kanban. También, se estableció la arquitectura del prototipo, se generaron el diagrama de base de datos y el diagrama de clases, así como los *sketches* para las interfaces de usuario.

Finalmente, se realizó la implementación del prototipo, para lo cual se generó un repositorio remoto en GitHub de forma de poder hacer el seguimiento de la evolución del código, así como de disponer de un mecanismo que permita, en caso de existir algún inconveniente, recuperar el código. Además, se levantó un ambiente de desarrollo que incluyó la instalación del paquete de software XAMPP, el IDE Android Studio y GitHub Desktop para gestionar la base de datos y los archivos del servicio web, desarrollar la aplicación Android con cada una de sus funcionalidades y administrar el código generado, respectivamente, así mismo se utilizaron etiquetas NFC para almacenar información dentro de ellas.

2.1 ANÁLISIS DE APLICACIONES MÓVILES DISPONIBLES EN EL MERCADO PARA LA BÚSQUEDA DE MASCOTAS

Entre tan amplia gama de herramientas técnicas, las más utilizadas y las mejores formas de buscar mascotas en aplicaciones móviles son las que ofrecen localización de animales vía GPS, como Tractive GPS Pet Finder, WizaPet y Welppy; a continuación, se detallan las características de estas aplicaciones:

- **WizaPet:** es una aplicación para ayudar a encontrar gatos y perros perdidos y llevarlos a casa lo antes posible. La aplicación puede crear una gran red de usuarios que se conectan entre sí, cuando se extravía alguna de sus mascotas, siendo un vínculo social para poder encontrarlas.

El primer paso para usar Wizapet es descargar la aplicación desde la tienda de Google Play o mediante su página web, subsiguiente cuando una mascota se pierde, el usuario debe cargar una imagen del mismo en un mapa, indicando la ubicación en la que se vio a la mascota por última vez.

La aplicación generará una alerta, la cual cubrirá a todos los usuarios de la zona que se encuentren registrados y estén a un radio de alcance cercano al sitio en el que se generó la alerta. Además, la aplicación dispone de un sistema de mensajería vía chat instantáneo que permite al usuario conversar con personas que actualmente están involucradas en el rescate de perros o gatos [30].

- **Tractive GPS Pet Finder:** esta aplicación tiene una gran confiabilidad por parte de los usuarios que la utilizan, porque usa posicionamiento GPS para ubicar con precisión a las mascotas extraviadas.

  Una de las principales características de esta aplicación es que no está enfocada a ayudar a otras personas, sino que requiere colocar un dispositivo de posicionamiento en la mascota, este dispositivo rastreo GPS es de la marca Tractive. Es decir que, los usuarios de esta aplicación requieren comprar un dispositivo de rastreo, con el cual podrán encontrar a sus mascotas sin la necesidad de terceros [31].

- **Welppy:** esta aplicación hace posible la colaboración entre usuarios cuando más se necesita. En este sentido, esta aplicación permite recibir o emitir alertas tempranas en tiempo real y geolocalizarlas, facilitando así la colaboración de la comunidad que tiene la aplicación. Además, consta con funciones útiles para hacer sonar una alarma en situaciones como una ola de saqueos, alertando a otros usuarios cercanos para evitar un riesgo mayor.

  La aplicación permite recibir notificaciones sobre artículos extraviados, robados o perdidos; informar si se ha encontrado una billetera o llaves, y buscar ayuda si un artículo se pierde o es robado. Por otro lado, gracias al sistema de alerta temprana, se pueden reportar situaciones de riesgo como incendios, inundaciones, fugas de gas, situaciones puntuales de contaminación, plagas o animales perdidos.

  Como característica esencial de la aplicación, los usuarios podrán recibir alertas verificadas de las autoridades de la zona: policía, bomberos, etc. [32].

En la Tabla 2.1 se presenta una comparación entre las aplicaciones descritas.

Tabla 2.1. Comparación aplicaciones móviles para búsqueda de mascotas

	WizaPet	**Tractive GPS Pet Finder**	**Welppy**
Sistemas soportados	Android / IOS	Android / IOS	Android / IOS
Precio	Gratuito	Pago mensual ($9.99)	Gratuito con precio de suscripción plus
Características	Mapas y notificaciones nativos, pago de seguro y soporte dedicado	Compra de localizador independiente	Alerta en tiempo real, objetos perdidos
Desarrolladora	Wizapet	Tractive	Welppy

2.2 DISEÑO

2.2.1 HISTORIAS DE USUARIO

Con base a las entrevistas realizadas a propietarios de mascotas (ver ANEXO A), se procedió a levantar las historias de usuario, de acuerdo con el formato y guía para la elaboración de historias de usuario definidas en [33], donde se explica que cada historia de usuario dispone de un identificador (`ID`) con el formato `HU_<número de historia de usuario>_<tipo de usuario>`, al igual que un título y su descripción.

Se han identificado dos tipos de usuario: propietario de mascota y usuario. Las historias de usuario para el propietario de mascota se listan en la Tabla 2.2, mientras que en la Tabla 2.3 se listan las del usuario.

Tabla 2.2. Historia de usuario para el propietario de mascota (Parte 1 de 2)

ID	**Título**	**Descripción**
HU_01_PR	Autenticación del usuario	El propietario de la mascota podrá ingresar con sus credenciales: correo electrónico y contraseña, para hacer uso de las funcionalidades del sistema.
HU_02_PR	Registro de una nueva mascota	El propietario de la mascota podrá guardar la información de sus mascotas, como nombre de la mascota, raza, género y alguna descripción adicional.
HU_03_PR	Gestión de la información de una mascota	El propietario de la mascota podrá eliminar o actualizar la información de sus mascotas.
HU_04_PR	Reportar una mascota como encontrada	El propietario de la mascota podrá cambiar el estado de mascota desaparecida a mascota encontrada de tal manera que se remueva la información de mascota desaparecida.

Tabla 2.2. Historia de usuario para el propietario de mascota (Parte 2 de 2)

ID	Título	Descripción
HU_05_PR	Reportar una mascota como desaparecida	El propietario de la mascota podrá informar a los usuarios del sistema que una de sus mascotas desapareció. Para ello, el propietario solicitará ubicar en un mapa el lugar donde vio a su mascota por última vez y enviar esta información a los usuarios del sistema junto con los datos de su mascota.
HU_06_PR	Almacenar información de una mascota en una etiqueta NFC	El propietario de la mascota podrá guardar la información de una mascota en una etiqueta NFC. La información a ser almacenada estará compuesta por un id de la mascota, número de teléfono celular del propietario y las coordenadas geográficas del hogar de la mascota.
HU_07_PR	Recibir la notificación que la mascota del propietario fue encontrada	El propietario de la mascota podrá recibir una notificación cuando su mascota haya sido encontrada. Además, el propietario requiere que la notificación a recibir este compuesta de la ubicación en la cual se encuentra la mascota en ese momento y los datos del usuario quien notificó dicho evento, entre ellos su número de celular para comunicarse con él inmediatamente.

Tabla 2.3. Historia de usuario del usuario del sistema (Parte 1 de 2)

ID	Título	Descripción
HU_01_SI	Registrarse en el sistema	El usuario del sistema podrá ingresar sus datos, tales como nombre, correo electrónico, número de teléfono celular y contraseña para que pueda registrarse en la aplicación.
HU_02_SI	Autenticación del usuario	El usuario del sistema podrá ingresar con sus credenciales: correo electrónico y contraseña, para hacer uso de las funcionalidades del sistema.
HU_03_SI	Mostrar un listado de las mascotas desaparecidas	El usuario del sistema podrá visualizar todas las mascotas del sistema que fueron reportadas como desaparecidas
HU_04_SI	Gestión de la información del usuario	El usuario del sistema podrá eliminar o editar su información personal, ya sea para actualizar o corregir algún error en sus datos.
HU_05_SI	Mostrar información de una mascota almacenada en una etiqueta NFC	El usuario del sistema podrá recuperar la información de una mascota, almacenada en una etiqueta NFC. El usuario también solicita mostrar en un mapa el hogar de la mascota, con base en las coordenadas geográficas almacenadas en la etiqueta.
HU_07_SI	Recibir notificaciones de las mascotas reportadas como desaparecidas	El usuario del sistema podrá recibir notificaciones cuando una mascota es reportada como desaparecida y pueda conocer la ubicación donde la mascota fue vista por última vez.

Tabla 2.3. Historia de usuario del usuario del sistema (Parte 2 de 2)

ID	Título	Descripción
HU_06_SI	Notificar al propietario de una mascota que su mascota fue encontrada	El usuario del sistema podrá notificar que encontró una mascota a su propietario, para ello, deberá leer la información de la etiqueta NFC, ubicada en el collar de la mascota. En caso de que el usuario del sistema no disponga de conexión a Internet para notificar al propietario, podrá hacer uso del número de teléfono celular del propietario de la mascota almacenado en la etiqueta NFC para poder comunicarse con él directamente.

2.2.2 REQUERIMIENTOS DEL PROTOTIPO

Una vez que se realizaron las historias de usuarios, se procedió a establecer los siguientes requerimientos funcionales del prototipo:

- El prototipo implementará un mecanismo para el registro de usuarios mediante el uso de un formulario.
- El prototipo contará con un mecanismo de autenticación de usuarios utilizando como credenciales un correo electrónico y contraseña.
- En el prototipo se definirán dos roles de usuario: propietario de la mascota y usuario del sistema.
- El prototipo permitirá al usuario del sistema realizar un CRUD de su información personal.
- El prototipo tendrá un mecanismo de notificación que permita a los propietarios de mascotas conocer la ubicación de su mascota cuando haya sido encontrada por un usuario del sistema.
- El prototipo permitirá al propietario de la mascota realizar un CRUD de sus mascotas.
- El prototipo contará con un mecanismo que permita a los propietarios de mascotas reportar cuando su mascota haya sido desaparecida o encontrada.
- El prototipo tendrá un mecanismo que permita a los propietarios de mascotas guardar la información de su mascota en una etiqueta NFC.

- El prototipo dispondrá de un mecanismo que permita al usuario del sistema visualizar la información de las mascotas que fueron reportadas como desaparecidas.
- El prototipo tendrá un mecanismo de notificación que permita al usuario del sistema conocer la información de una mascota reportada como desaparecida.
- El prototipo dispondrá de un mecanismo que permita a los usuarios del sistema visualizar la información de una mascota contenida en una etiqueta NFC.

Los requerimientos no funcionales del prototipo son:

- El prototipo dispondrá de un servicio web codificado con el lenguaje de programación PHP mediante el editor de código Visual Studio Code.
- El prototipo tendrá una base de datos MySQL.
- El prototipo contará con una aplicación Android codificada con el lenguaje de programación Java mediante el IDE Android Studio.
- Las interfaces graficas deben ser amigables e intuitivas para los usuarios.
- El prototipo usará etiquetas NFC para almacenar la información de las mascotas.

2.2.3 DIAGRAMA DE CASOS DE USO

Los actores del prototipo son: propietario de la mascota y usuario del sistema, mismos que cuentan con funciones que pueden ser intercaladas dependiendo de los requerimientos de los mismos, teniendo la disponibilidad de cambiar de rol sin la necesidad de crear un nuevo usuario.

En la Figura 2.1 se presenta cada uno de los procesos y funciones que puede realizar el actor propietario de la mascota, esto mediante un diagrama de casos de uso con las posibles interacciones que realiza el actor.

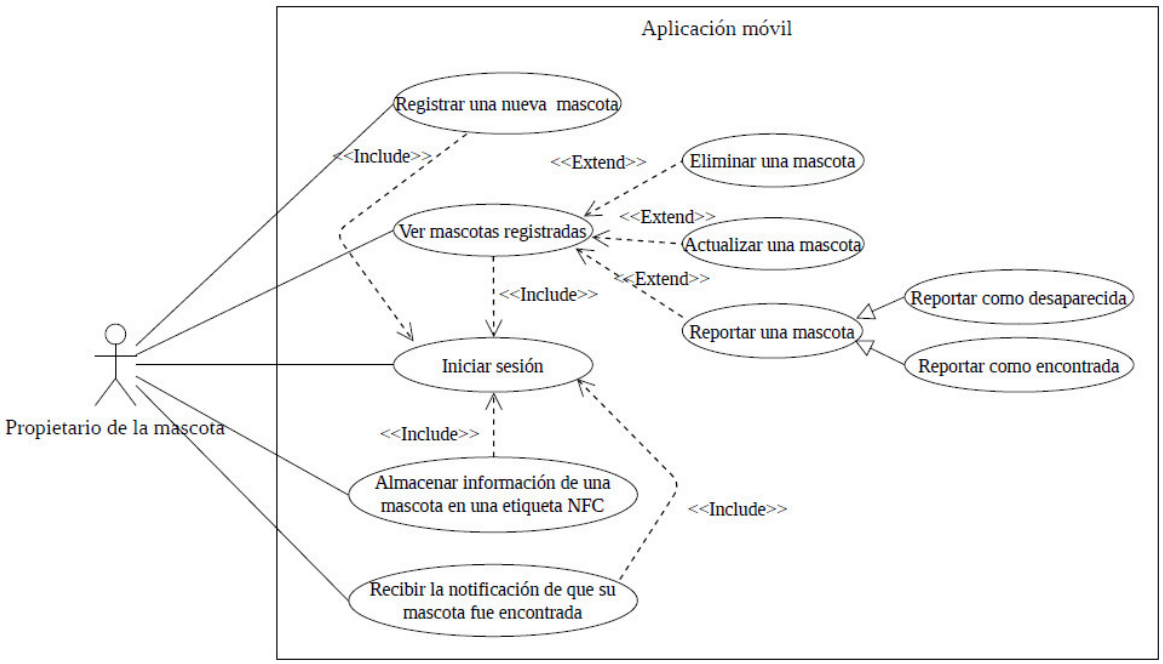

Figura 2.1. Diagrama de casos de uso para el propietario de la mascota

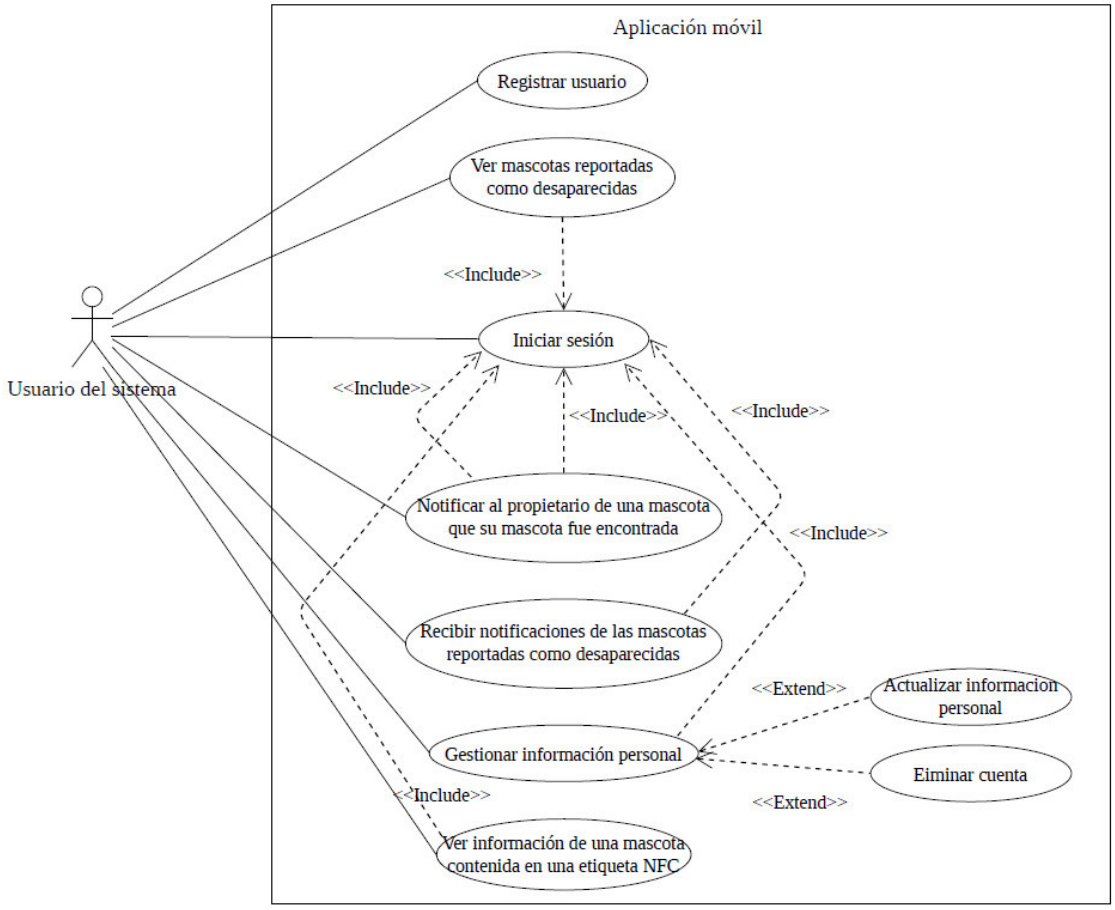

Figura 2.2. Diagrama de casos de uso para el usuario del sistema

2.2.4 TABLERO KANBAN

La lista de tareas que deben ser realizadas para cumplir con los requerimientos establecidos según cada rol de usuario definidas en las historias de usuario y en los casos de uso se presenta en la Tabla 2.4.

Tabla 2.4. Tareas a realizar (Parte 1 de 5)

Requerimiento	Tarea	To Do	Doing	Done
Registrar usuarios	Crear la vista para el registro de usuarios			
	Añadir las cajas de texto para ingresar el nombre, correo electrónico y número de celular del usuario			
	Incorporar un botón para iniciar el proceso de registro			
	Mostrar un mensaje para indicar si el registro de usuario se cumplió con éxito o presentó algún inconveniente			
Autenticar usuarios	Crear la vista para la autenticación de usuarios			
	Añadir los campos de texto para ingresar el correo electrónico y número de celular del usuario			
	Añadir un botón para iniciar el proceso de autenticación			
Gestionar la información del usuario	Crear la vista para actualizar la información del usuario			
	Añadir los campos de texto para editar el nombre, correo electrónico y número de celular del usuario			
	Añadir un botón para iniciar el proceso de actualización			
	Mostrar un mensaje para indicar si la actualización de la información del usuario fue exitosa o presentó algún inconveniente			
	Crear la vista para actualizar la contraseña del usuario			
	Añadir los campos de texto para ingresar la contraseña actual, nueva contraseña y la confirmación de la nueva contraseña			
	Añadir un botón para iniciar el proceso de actualización de la contraseña			
	Mostrar un mensaje para indicar si la actualización de la contraseña fue exitosa o presentó algún inconveniente			

Tabla 2.4. Tareas a realizar (Parte 2 de 5)

Requerimiento	Tarea	To Do	Doing	Done
Registrar mascotas	Crear la vista para el registro de mascotas			
	Añadir los campos de texto para ingresar el nombre, raza y una descripción de la mascota			
	Añadir los botones de opción para elegir el género de la mascota			
	Añadir un botón para iniciar el registro de la mascota			
	Crear la vista para mostrar la lista de mascotas registradas por el usuario			
Gestionar la información de mascotas	Crear la vista para actualizar la información de una mascota			
	Añadir los campos de texto para editar el nombre, raza y la descripción de una mascota			
	Añadir los botones de opción para editar el género de una mascota			
	Añadir un botón para iniciar el proceso de actualización de una mascota			
	Mostrar un mensaje para indicar si la actualización de la información de la mascota fue exitosa o presentó algún inconveniente			
	Crear la vista para eliminar la información de una mascota			
	Añadir un botón para eliminar una mascota			
	Incorporar un cuadro de diálogo para confirmar la eliminación de una mascota			
Almacenar información de una mascota en una etiqueta NFC	Crear la vista para seleccionar una mascota registrada y la ubicación de su hogar			
	Incorporar un mapa para seleccionar la ubicación del hogar de la mascota			
	Añadir una lista desplegable para seleccionar una de las mascotas registradas por su propietario			
	Añadir un campo para ingresar la dirección del hogar de la mascota			
	Añadir un botón para confirmar la mascota y su ubicación seleccionada			

Tabla 2.4. Tareas a realizar (Parte 3 de 5)

Requerimiento	Tarea	To Do	Doing	Done
Almacenar información de una mascota en una etiqueta NFC	Crear la vista para mostrar la información de la mascota, la ubicación de su hogar, anteriormente seleccionados			
	Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario			
	Añadir un botón para iniciar el proceso de transferencia de datos hacia la etiqueta NFC			
	Crear la vista para iniciar la transferencia de datos hacia la etiqueta NFC			
	Añadir una vista de texto para indicar al usuario si su teléfono inteligente es capaz de realizar la operación de escritura NFC			
	Mostrar un cuadro de diálogo para informar si la escritura NFC fue exitosa o presentó algún inconveniente			
Mostrar información obtenida desde una etiqueta NFC	Crear la vista para iniciar la transferencia de datos de la etiqueta NFC a la aplicación Android			
	Añadir una vista de texto para indicar al usuario si su teléfono inteligente es capaz de realizar la operación de lectura NFC			
	Crear la vista para mostrar la información de la mascota contenida en la etiqueta NFC			
	Incorporar un mapa para mostrar al usuario la ubicación del hogar de la mascota			
	Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario			
Listar mascotas desaparecidas	Crear la vista para mostrar todas las mascotas reportadas como desaparecidas del sistema			
	Añadir las vistas de texto para mostrar al usuario el nombre y raza de las mascotas desaparecidas			

Tabla 2.4. Tareas a realizar (Parte 4 de 5)

Requerimiento	Tarea	To Do	Doing	Done
Reportar una mascota encontrada	Añadir un botón para reportar una mascota como encontrada y quitarla de la lista de mascotas desaparecidas			
Notificar una mascota encontrada	Crear la vista para iniciar la transferencia de datos de la etiqueta NFC a la aplicación Android			
	Crear la vista para mostrar la información de la mascota encontrada proveída por la etiqueta NFC			
	Incorporar un mapa para mostrar al usuario la ubicación del hogar de la mascota que fue encontrada y su ubicación actual			
	Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario			
	Añadir un botón para notificar al propietario que su mascota fue encontrada			
	Añadir un botón para comunicarse directamente con el propietario de la mascota mediante llamada telefónica			
	Crear la vista para mostrar al propietario de la mascota la información del usuario quien encontró a su mascota y la ubicación de la misma			
	Incorporar un mapa para mostrar al propietario la ubicación donde fue encontrada su mascota			
	Añadir las vistas de texto para mostrar al propietario el nombre, correo electrónico y numero de celular del usuario quien encontró la mascota			
	Añadir un botón para que el propietario de la mascota encontrada se comunique mediante llamada telefónica con el usuario quien encontró su mascota			
Notificar una mascota desaparecida	Crear la vista para notificar una mascota como desaparecida			
	Incorporar un mapa para seleccionar la ubicación en donde fue vista por última vez una mascota seleccionada			

Tabla 2.4. Tareas a realizar (Parte 5 de 5)

Requerimiento	Tarea	To Do	Doing	Done
Notificar una mascota desaparecida	Añadir un botón para enviar la notificación de mascota desaparecida a todos los usuarios del sistema que se encuentran en línea			
	Crear la vista para mostrar la información de la mascota desaparecida			
	Incorporar un mapa para mostrar al usuario la ubicación donde fue vista por última vez la mascota desaparecida			
	Añadir las vistas de texto para mostrar al usuario el nombre, raza, género y descripción de la mascota desaparecida			

2.2.5 ARQUITECTURA DEL PROTOTIPO

El prototipo se implementó con base en la arquitectura cliente-servidor. En el lado del servidor, se empleó el paquete de software XAMPP, el cual incluye un servidor web que se encarga de recibir y enviar los recursos solicitados en formato JSON mediante el protocolo HTTP y que a su vez se comunica con un servicio web el cual se encarga de gestionar los datos que son almacenados en una base de datos. El cliente es una aplicación Android encargada de procesar las peticiones y respuestas que se envían al servicio web a recursos que se identifican a través de una URI. Además, la aplicación Android usa el modo de operación lector/escritor para recuperar o guardar información desde o hacia la etiqueta NFC, la cual contiene la información de una mascota.

2.2.6 BASE DE DATOS

A partir de la información obtenida en las historias de usuario, se procedió a identificar las tablas que se requieren, así como los atributos o columnas de las mismas y sus relaciones. El diagrama relacional de la base de datos se presenta en la Figura 2.3.

La tabla `tblCliente` almacena la información de los usuarios registrados en el sistema, como nombre, correo electrónico, contraseña y número de teléfono celular. Esta tabla tiene una relación de uno a varios con las tablas `tblMascota` y `tblToken`.

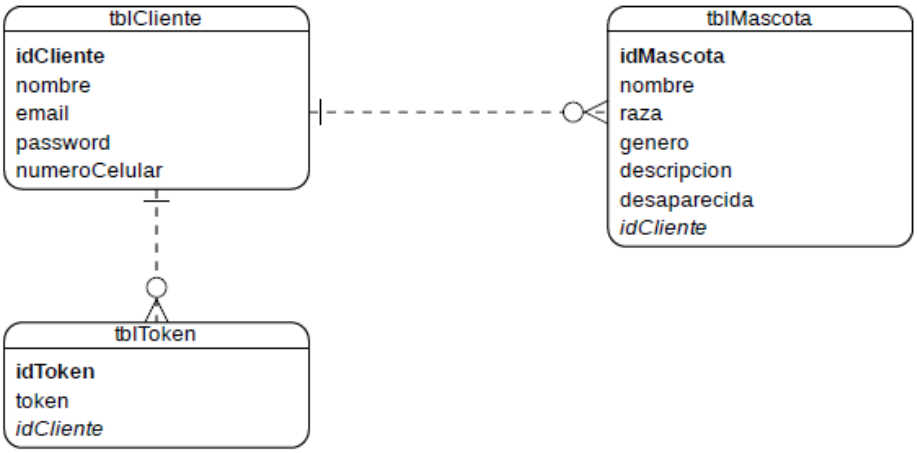

Figura 2.3. Diagrama relacional de la base de datos

La tabla `tblMascota` almacena la información de las mascotas de los usuarios como: nombre, raza, género y descripción; además dispone de un atributo denominado `desaparecida`, el cual puede tomar un valor de 0 si la mascota no se encuentra desaparecida, y un valor mayor o igual a 1 para indicar que la mascota se encuentra desaparecida.

La relación entre las tablas `tblCliente` y `tblMascota` es de uno a muchos, es decir un usuario puede tener muchas mascotas, sin embargo, una mascota es propiedad de un solo usuario por lo cual esta tabla contiene el atributo `idCliente` para identificar al usuario al que le pertenece cada mascota.

2.2.7 DIAGRAMA DE CLASES

La Figura 2.4 presenta el diagrama de clases para gestionar la información de la base de datos del prototipo. Las clases empleadas son: `Cliente`, `Token` y `Mascota`.

La relación que forman las clases es del tipo asociación con una cardinalidad de 1 a 1 ó más, lo cual representa el número de objetos que participan en la relación. Por otra parte, la clase `Mascota` implementa la interfaz `Serializable` de Java para convertir un objeto a un *stream* de bytes y poder enviarlo de una actividad a otra.

Cada clase está definida por su identificador, atributos y métodos. Estos dos últimos representan las características y comportamientos de un objeto instanciado de una clase; además, cuentan con modificadores de acceso para indicar que los atributos o métodos serán públicos (+) o privados (-).

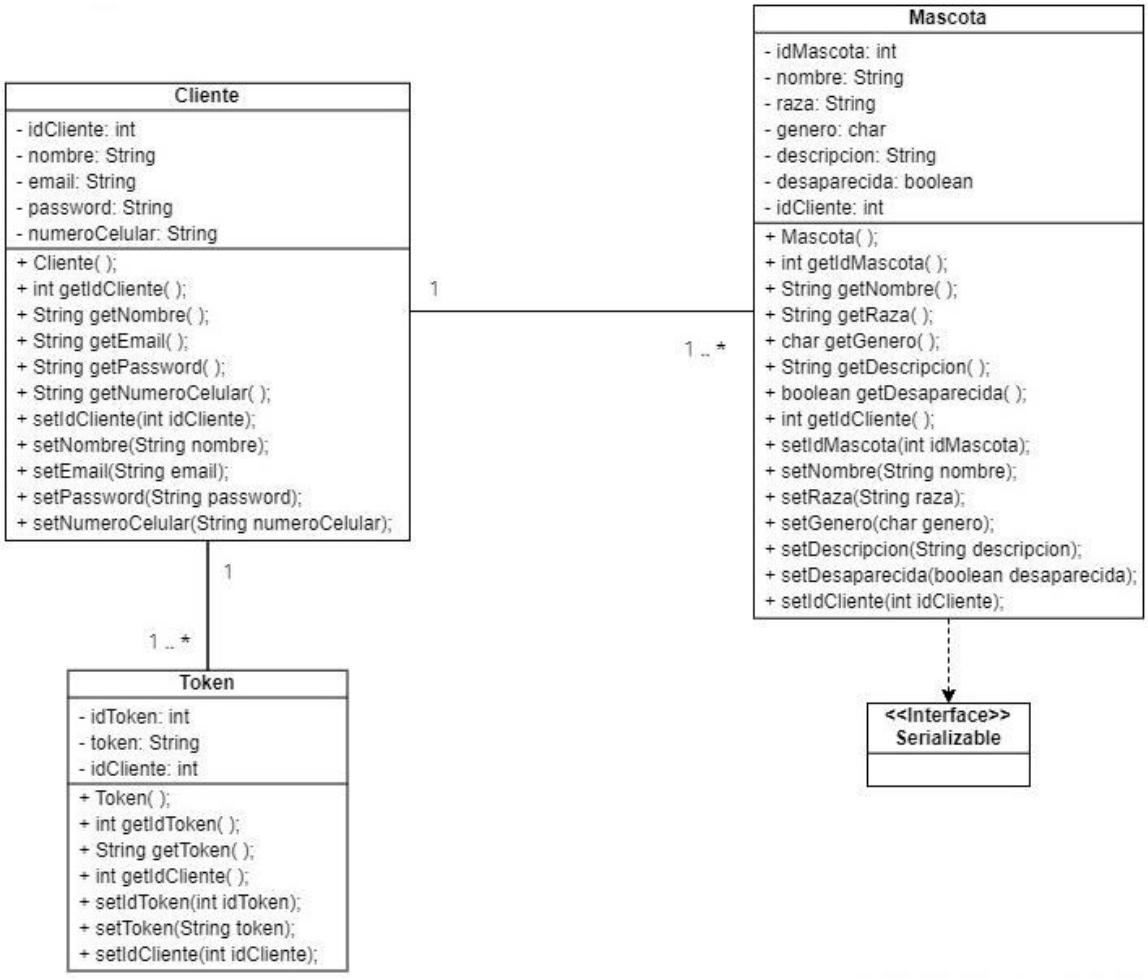

Figura 2.4. Diagrama de clases

Para manipular la información de los atributos privados desde otra clase, se crearon los métodos *get* y *set* por cada atributo, así mismo se estableció en cada clase su método constructor, el cual es invocado al momento de crear un objeto.

2.2.8 SKETCHES PARA LAS INTERFACES

En esta sección se presentan algunos de los bocetos (*sketches*), los cuales han sido creados empleando la herramienta *online* de Visual Paradigm y que servirán de base para el diseño de la interfaz gráfica de usuario (GUI) de la aplicación Android.

El objetivo de los *sketches* es brindar una idea general sobre la distribución de los iconos, cajas de texto, imágenes, botones, entre otros, con los cuales el usuario podrá interactuar con la aplicación. En el ANEXO B se encuentran los *sketches* desarrollados.

El nombre asignado a la aplicación móvil que se desarrolla en este Trabajo de Titulación es "`ReturnHome`", cuya función principal es ofrecer a los propietarios de mascotas una herramienta que permitir dar con el paradero de su mascota a través de etiquetas NFC y conexión a Internet.

El ícono de la aplicación móvil se presenta en la Figura 2.5, el cual ha sido descargado del sitio web Flaticon bajo licencia gratuita.

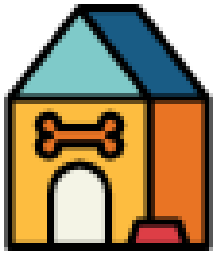

Figura 2.5. Ícono de la aplicación Android ReturnHome

La pantalla que el usuario puede ver al iniciar la aplicación se muestra en la Figura 2.6 (a), este *sketch* contiene dos entradas de texto que permitan al usuario ingresar su correo electrónico y su contraseña para iniciar sesión y autenticarse, luego de lo cual podrá acceder a las demás funcionalidades de la aplicación Android, así mismo le permite al usuario dirigirse a otra pantalla para crear una nueva cuenta o registrarse.

En la Figura 2.6 (b) se presenta el *sketch* de la pantalla de registro de un nuevo usuario, mediante la cual una persona podrá registrarse en el prototipo ingresando su nombre completo, correo electrónico, contraseña y número de teléfono celular.

La Figura 2.7 (a) presenta el *sketch* de la pantalla `Home` que se mostrará al usuario cuando haya iniciado sesión en la aplicación Android. Este *sketch* presenta dos pestañas, las cuales contienen la información de las mascotas registradas por su propietario y las mascotas reportadas como desaparecidas del sistema; además en este *sketch* el propietario de mascotas podrá utilizar para cada una de sus mascotas las siguientes opciones:

- **Editar:** opción que permite abrir una vista emergente con la información actual de la mascota y posteriormente actualizarla (ver Figura 2.7 (b)).
- **Eliminar:** opción que permite eliminar la información de la mascota seleccionada.
- **Reportar como:** submenú que muestra una ventana emergente con las opciones para reportar una mascota como desaparecida o encontrada.

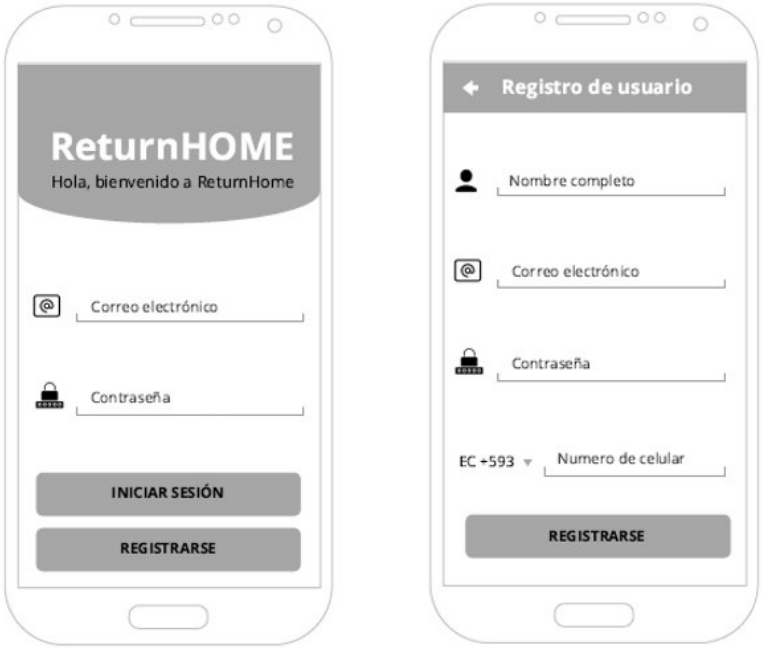

Figura 2.6. (a) *Sketch* de la pantalla de inicio de la aplicación. (b) *Sketch* de la pantalla de registro de la información del usuario.

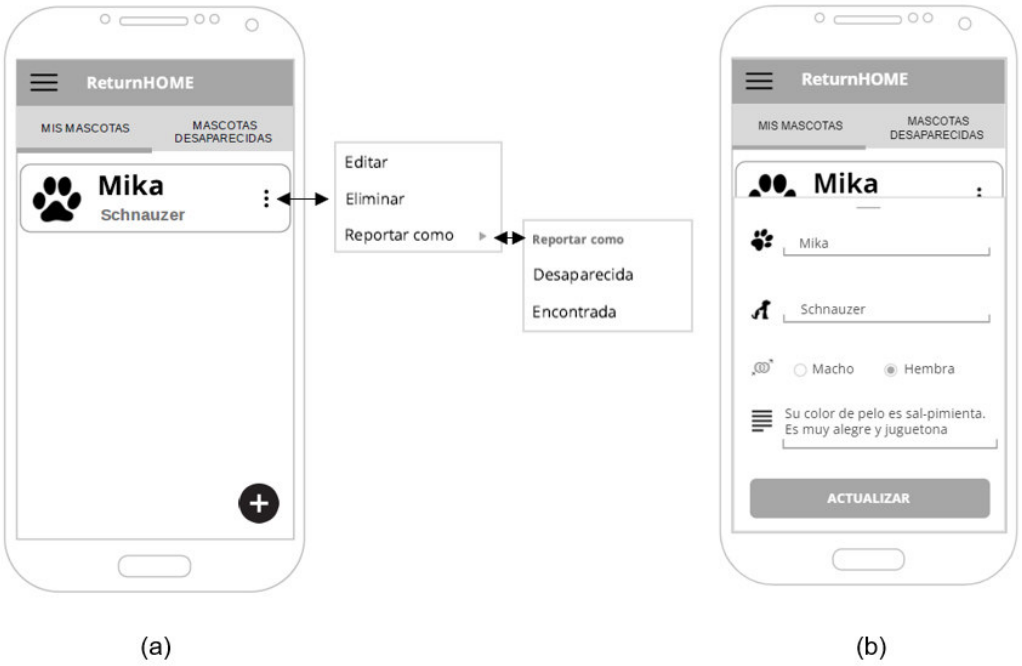

Figura 2.7. (a) *Sketch* de la pantalla Home para la visualización de las mascotas del usuario y las mascotas desaparecidas del sistema. (b) *Sketch* de la ventana emergente para la actualizar la información de la mascota.

La Figura 2.8 (a) presenta el *sketch* de la ventana emergente que permite añadir una nueva mascota cuando el usuario oprima en el botón "+" de la parte inferior derecha de la pantalla Home. Para ingresar información de una nueva mascota, el usuario deberá ingresar el nombre de la mascota, raza, género, estos campos serán obligatorios para registrar una mascota, así mismo el usuario podrá ingresar una descripción de su mascota de manera opcional

La Figura 2.8 (b) muestra el *sketch* del menú principal de la aplicación, el cual contiene la información del usuario que inició sesión, como su nombre completo y su número de teléfono celular.

El menú principal del *sketch* de la Figura 2.8 (b) también incluye las siguientes opciones:

- **Escribir etiqueta:** opción que consta de tres pantallas, las cuales se presentan en la Figura 2.9. La pantalla que inicialmente se mostrará al usuario cuando oprima la opción `Escribir etiqueta` se indica en la Figura 2.9 (a), en esta pantalla el usuario podrá seleccionar una de sus mascotas y a su vez la ubicación del hogar de la mascota seleccionada. Una vez seleccionada la mascota y la ubicación de su hogar, el usuario deberá oprimir el botón `Seleccionar ubicación` lo cual provocará que se presente la pantalla indicada en la Figura 2.9 (b), en la cual se incluirá la información de la mascota seleccionada, el número de contacto del propietario y la ubicación del hogar de la mascota. Para abrir la siguiente y ultima pantalla, el usuario deberá presionar en el botón `Escribir etiqueta` de la pantalla que se indica en la Figura 2.9 (b).

  La Figura 2.9 (c) presenta el *sketch* de la pantalla para iniciar la transferencia de datos hacia la etiqueta NFC; indicado al usuario si su teléfono inteligente es capaz de realizar esta acción.

  Si el teléfono inteligente es compatible con la tecnología NFC, el usuario podrá acercar su teléfono a la etiqueta NFC e iniciar el envío de datos. Cuando el proceso de transferencia de datos haya culminado, la aplicación le mostrará al usuario un cuadro de diálogo con un mensaje informando el resultado del proceso.

- **Leer etiqueta:** opción que consta de dos pantallas que se presentan en la Figura 2.10. La Figura 2.10 (a) presenta el *sketch* de la pantalla para iniciar la transferencia de datos *desde la etiqueta NFC hacia la aplicación Android, estos datos serán presentados al usuario en la pantalla de la Figura 2.10 (b).*

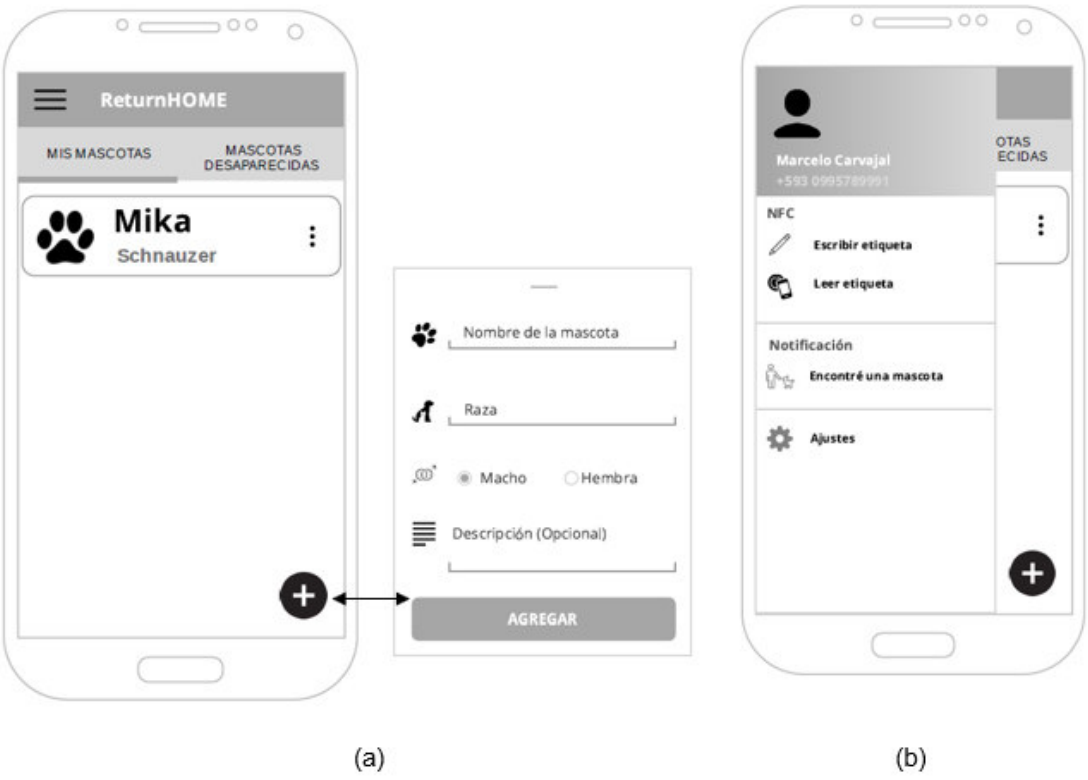

Figura 2.8. (a) *Sketch* de la ventana emergente para añadir una nueva mascota. (b) *Sketch* del menú lateral de la pantalla Home

- **Encontré una mascota:** opción que consta de dos pantallas, las cuales se ilustran en la Figura 2.10 (a) y en la Figura 2.11.

  El *sketch* de la Figura 2.11 indica los datos de la mascota que fue encontrada, la ubicación del hogar de la mascota y la ubicación donde fue encontrada al momento de leer la etiqueta NFC, así mismo el usuario puede enviar una notificación al propietario de la mascota o comunicarse directamente con el propietario mediante llamada telefónica.

- **Ajustes:** opción que consta de cuatro opciones, las cuales se presentan en la Figura 2.12.

  Este *sketch* representa todas las opciones posibles de configuración de la cuenta del usuario, como cambiar la contraseña, editar la información de la cuenta, eliminar la cuenta y cerrar sesión, al seleccionar estas dos últimas opciones le muestran al usuario un cuadro de diálogo para confirmar si desea continuar o no con el proceso.

Figura 2.9. Sketch de la pantalla (a) selección de mascota y ubicación de su hogar. (b) información de la mascota y la ubicación de su hogar. (c) inicio de la transferencia de datos desde la aplicación Android hacia la etiqueta NFC

Figura 2.10. Sketch de la pantalla (a) inicio de la transferencia de datos desde la etiqueta NFC hacia la aplicación Android. (b) información de la mascota

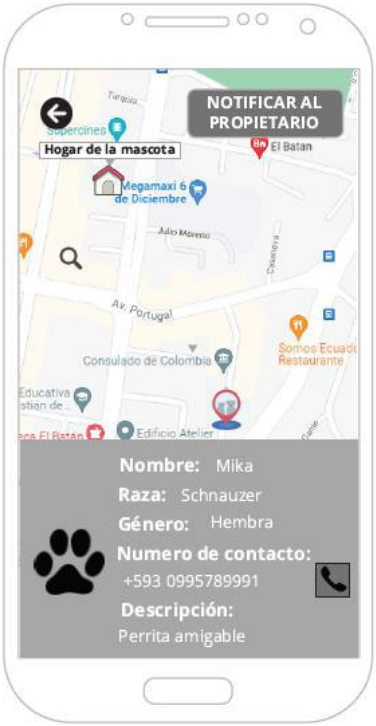

Figura 2.11. *Sketch* de la pantalla para mostrar la información de la mascota que fue encontrada y enviar una notificación al propietario.

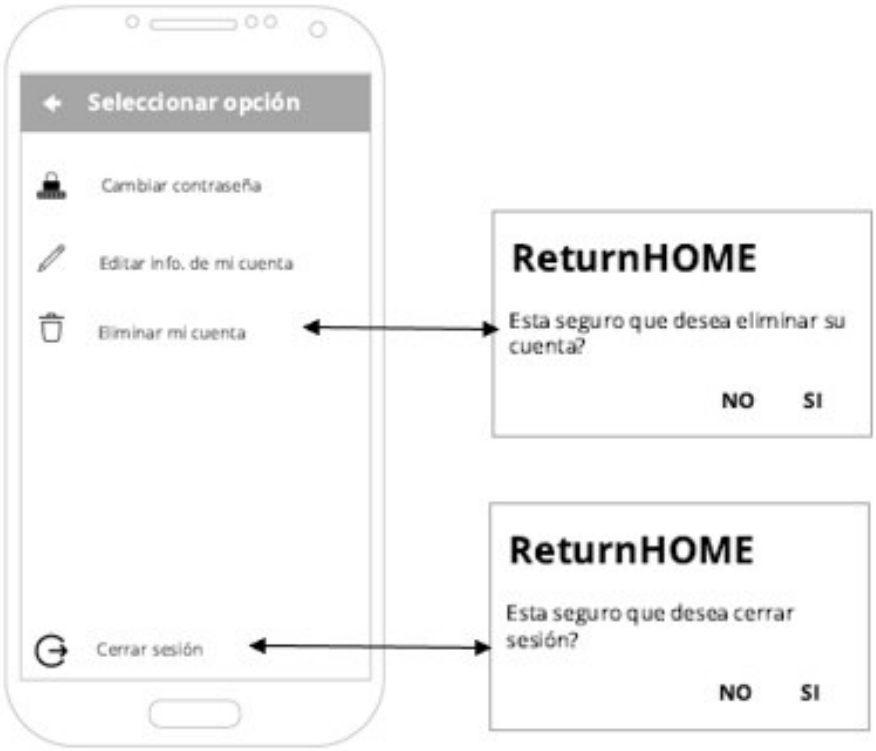

Figura 2.12. Sketch de la pantalla ajustes de la cuenta

2.3 IMPLEMENTACIÓN

2.3.1 CONTROL DE VERSIONES

Para realizar el seguimiento de los cambios efectuados a lo largo del desarrollo del código del prototipo, se utiliza la herramienta GitHub Desktop la cual incluye las funciones de Git sin necesidad de utilizar una línea de comandos.

La Figura 2.13 ilustra la creación del repositorio en GitHub. El repositorio remoto se le ha dado por nombre `ReturnHome`, el cual incluirá el código desarrollado tanto para la aplicación Android como para el servicio web. Además, se incluye una descripción y la privacidad se selecciona en modo privado.

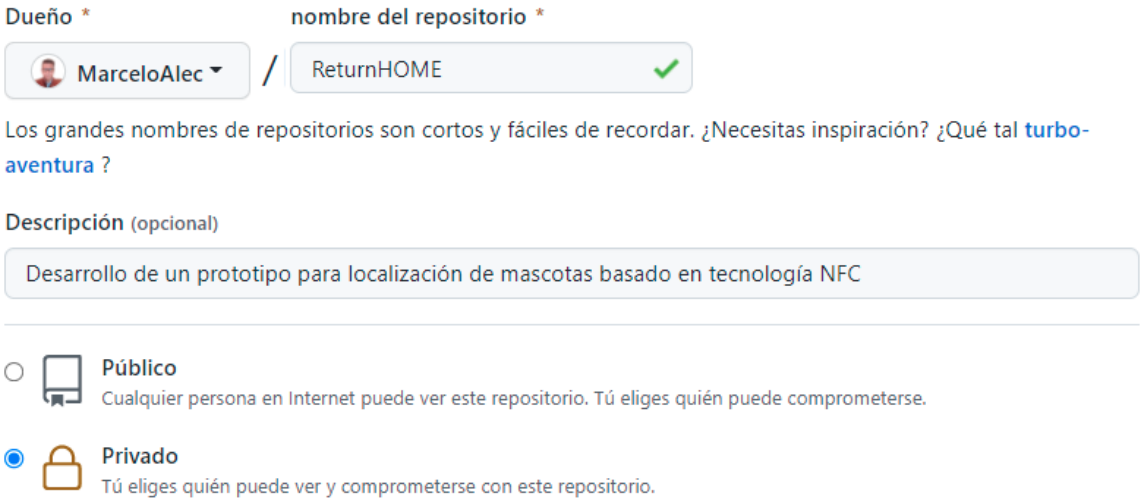

Figura 2.13. Creación del repositorio en GitHub

Para realizar una copia local del repositorio remoto se emplea la opción `Clone a repository` de GitHub Desktop, como se presenta en la Figura 2.14.

En la Figura 2.15 se ilustra el historial de algunos de los cambios efectuados a lo largo del desarrollo del prototipo. Se puede observar que se presenta el nombre, así como una descripción opcional de la función que realizan. También se observa los archivos en donde fueron realizados los cambios (recuadro rojo) y las líneas de código agregadas o eliminadas (recuadro verde).

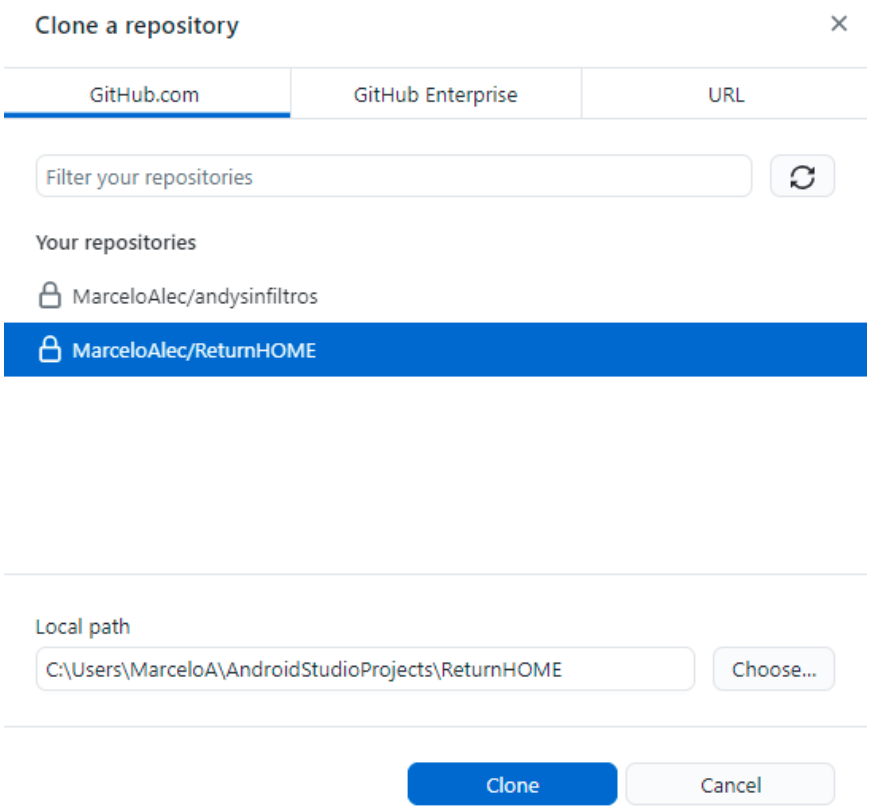

Figura 2.14. Creación de una copia local del repositorio remoto

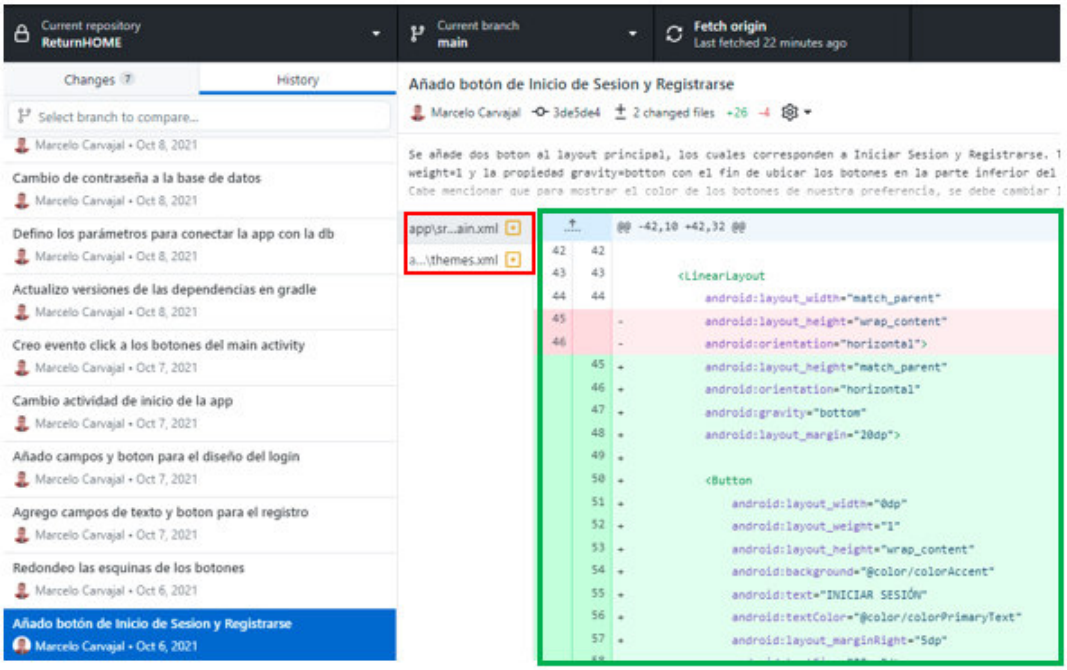

Figura 2.15. Interfaz gráfica de GitHub Desktop con el repositorio del prototipo seleccionado

2.3.2 BASE DE DATOS

Para la implementación de la base de datos se utilizó el paquete de software XAMPP, que contiene varias herramientas como: el gestor de base de datos MySQL, el servidor web Apache y el intérprete para el lenguaje de programación PHP, entre otros. La base de datos se implementa a través de la interfaz gráfica de base de datos denominada phpMyAdmin.

Para la creación de la base de datos, se abre un navegador web y se accede a la página de administración de base de datos a través de la URL `http://localhost:82/phpmyadmin/`; el número de puerto dependerá del valor configurado previamente en el equipo local. Cabe mencionar que para acceder a la página de administración de base de datos se debe iniciar Apache y MySQL del panel de control de XAMPP. Una captura del panel de control se indica en la Figura 2.16.

Figura 2.16. Panel de control de XAMPP que presenta que Apache y MySQL han iniciado

En la Figura 2.17 se presenta la base de datos `dbreturnhome` con sus respectivas tablas y relaciones.

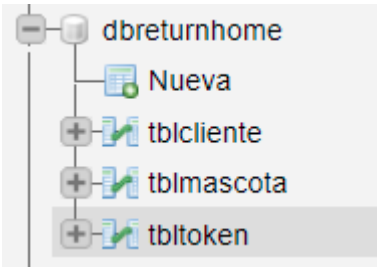

Figura 2.17. Base de datos del prototipo

A través de phpMyAdmin se implementaron las tablas de la base de datos, según el diagrama relacional. La Figura 2.18 presenta la tabla `Cliente`, donde se almacenará la información de los usuarios; esta tabla contiene un identificador, el cual es la clave primaria de la tabla y a su vez un valor entero auto numerado.

#	Nombre	Tipo	Extra
1	idCliente	int(11)	AUTO_INCREMENT
2	nombre	varchar(150)	
3	email	varchar(150)	
4	password	varchar(255)	
5	numeroCelular	varchar(255)	

Figura 2.18. Tabla `Cliente`

La Figura 2.19 muestra la tabla `Mascota`, en la cual se ha definido un identificador auto enumerado, el cual representa la clave primaria de la tabla, así mismo se incluye una clave foránea denominada `idCliente` que permite que se relacionen los datos de esta tabla con la tabla `cliente`. Esta tabla también contiene los campos `nombre`, `raza`, `género`, `descripción` que permiten almacenar la información de la mascota y un campo denominado `desaparecida` que indica si la mascota se encuentra o no desaparecida.

En la Figura 2.20 se muestra la tabla `tbltoken`, la cual contiene un identificador que representa la clave primaria auto numerada; además de un campo `token`, el cual almacena un identificador cuando el usuario inicia sesión en la aplicación Android, así mismo esta tabla presenta un `idCliente` que se usa para la relación con la tabla `Cliente`.

#	Nombre	Tipo	Extra
1	idMascota	int(11)	AUTO_INCREMENT
2	nombre	varchar(150)	
3	raza	varchar(150)	
4	genero	char(1)	
5	descripcion	varchar(255)	
6	desaparecida	tinyint(1)	
7	idCliente	int(11)	

Figura 2.19. Tabla `Mascota`

#	Nombre	Tipo	Extra
1	idToken	int(11)	AUTO_INCREMENT
2	token	varchar(255)	
3	idCliente	int(11)	

Figura 2.20. Tabla `Token`

2.3.3 SERVICIO WEB

Dentro de la ubicación en la cual fue instalado XAMPP se encuentra un directorio denominado `htdocs`, en el cual se almacenan todos los archivos o directorios que serán expuestos por el servidor web Apache, así como los archivos que conformarán el servicio web.

Cada archivo o directorio que se almacene en `htdocs` dispondrá de una URL. En la Figura 2.21 se presentan todos los archivos que fueron generados para el prototipo y que forman parte del servicio web, estos archivos se encuentran almacenados en el directorio `v1` que hace referencia a la versión 1 del servicio web y este a su vez se encuentra en el directorio `api.returnhome.com` dentro del directorio `htdocs`.

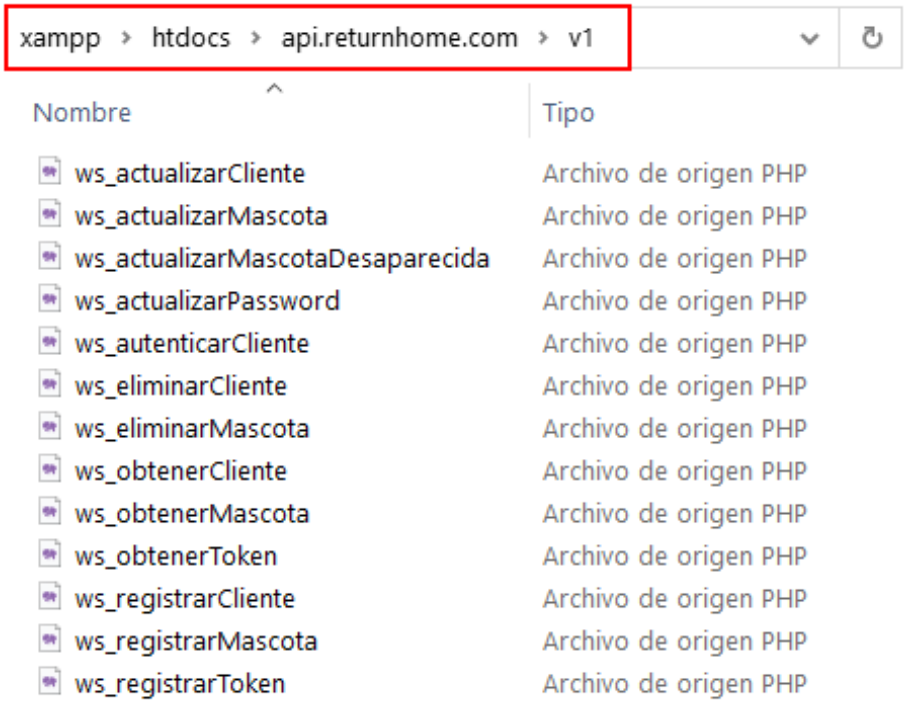

Figura 2.21. Archivos del servicio web con extensión PHP

Cada archivo tiene una funcionalidad en particular. A manera de ejemplo, se explicará uno de ellos. El código completo del servicio web se encuentra en el ANEXO E.

El archivo que define el método para recuperar la información de un cliente del servicio web se denomina `ws_obtenerCliente.php` y se encarga de obtener la información de un cliente con base en su identificador único en formato JSON, como se ilustra en la Figura 2.22. Las líneas 1 y 38 indican la apertura y cierre, respectivamente, de un bloque de código escrito en el lenguaje de programación PHP. La línea 2 indica el nombre del equipo en el cual se encuentra alojada la base de datos, en este caso corresponde a `localhost`.

```
1  <?php
2  $hostname = "localhost";
3  $username = "root";
4  $password = "1998*";
5  $database_name = "dbreturnhome";
6
7  if(isset($_GET['idCliente'])){
8
9      $idCliente = $_GET['idCliente'];
10
11     //CONEXION A LA BASE DE DATOS
12     $conexion=new PDO("mysql:host=".$hostname.";dbname=".$database_name."",
13                                    $username,
14                                    $password,
15                                    array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
16
17     $stmt = $conexion->prepare("SELECT * FROM tblcliente WHERE idCliente = ?");
18     $stmt->bindValue(1,$idCliente);
19
20     if($stmt->execute()){
21
22         $respuesta = $stmt->fetch(PDO::FETCH_ASSOC);
23
24         $stmt=null;
25
26         http_response_code(200);
27
28         echo json_encode(array("cliente" => $respuesta));
29
30     }
31     else{
32         http_response_code(404);
33     }
34 }
35 else{
36     http_response_code(404);
37 }
38 ?>
```

Figura 2.22. Código para consultar un cliente por su identificador único

Las líneas 3 y 4 incluyen las credenciales para el acceso a la base de datos. La línea 5 especifica el nombre de la base de datos (`dbreturnhome`). En la línea 7 se verifica si se ha incluido el `idCliente` en el pedido de HTTP y si posee un valor diferente de vacío. La

conexión a la base de datos se la establece a través de la clase `PDO` con los parámetros establecidos previamente (líneas 12 a 15).

En las líneas 17 y 18 se establece la consulta SQL para seleccionar toda la información del usuario al que corresponde el id remitido en el pedido. Posteriormente, en la línea 20 se procede a ejecutar la consulta a la base de datos.

Una vez obtenida la respuesta desde la base de datos, se invoca al método `fetch()` con el parámetro `PDO::FETCH_ASSOC` para especificar que el resultado devuelto sea en forma de *array* asociativo donde la clave es el nombre de cada columna y el valor el dato que pertenece a esa columna (línea 22).

Después, se establece la variable `$stmt` en nulo para cerrar la conexión con la base de datos y posterior a esto, se envía la respuesta HTTP con la información del cliente en formato JSON y el código de respuesta 200, indicando que la solicitud ha tenido éxito (líneas 26 a 28). Finalmente, la última parte del código (líneas 31 a 37) permite generar una notificación 404 del protocolo HTTP en caso de errores.

2.3.4 APLICACIÓN ANDROID

Para empezar a desarrollar la aplicación se debe crear un nuevo proyecto dentro del IDE Android Studio, para lo cual se deben configurar algunos parámetros como se ilustra en la Figura 2.23. Al crear un nuevo proyecto, este IDE permite elegir el nivel de API mínimo para la aplicación, en este caso se eligió el nivel de API 21, con el cual la aplicación podrá ejecutarse en el 98% de dispositivos Android (ver recuadro rojo).

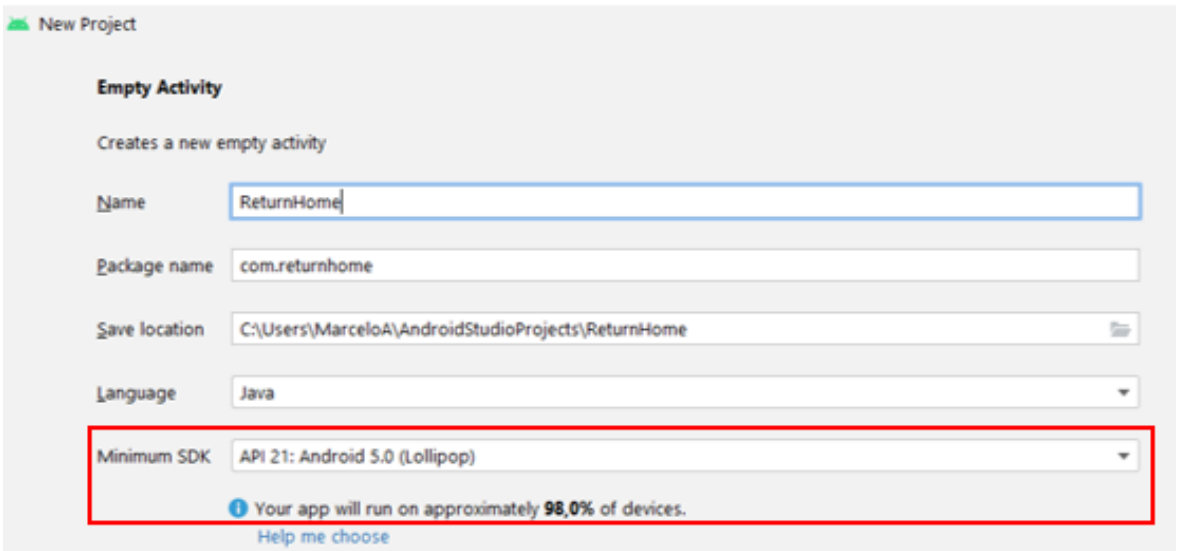

Figura 2.23. Creación de un nuevo proyecto en Android Studio

Posteriormente, se agregan las dependencias. En particular para este proyecto se empleará la librería de Retrofit para realizar las peticiones HTTP al servicio web. En la Figura 2.24 se presenta el código requerido para incluir la librería Retrofit. Además, se requiere la librería Gson para convertir los objetos Java a JSON y viceversa; el código requerido se presenta en la Figura 2.25.

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
```

Figura 2.24. Librería de Retrofit

```
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

Figura 2.25. Librería de Gson

Para utilizar el servicio de mensajería de Firebase, se debe registrar la aplicación en Firebase; esto se lo puede realizar con la ayuda del asistente de Android Studio como se indica en la Figura 2.26; el asistente indicará los pasos a realizar para hacer uso de las funcionalidades de Firebase *Cloud Messaging*.

Para habilitar la tecnología NFC en una aplicación, se deben otorgar los permisos necesarios a la aplicación de tal forma de que esta pueda usar el hardware NFC. Otorgar este permiso (ver Figura 2.27) permite que la aplicación maneje los *intents* y use hardware NFC. Se manejará el *intent*: `ACTION_NDEF_DISCOVERED`, para indicar que el contenido de una etiqueta NFC al momento de escanearla debe estar en formato NDEF.

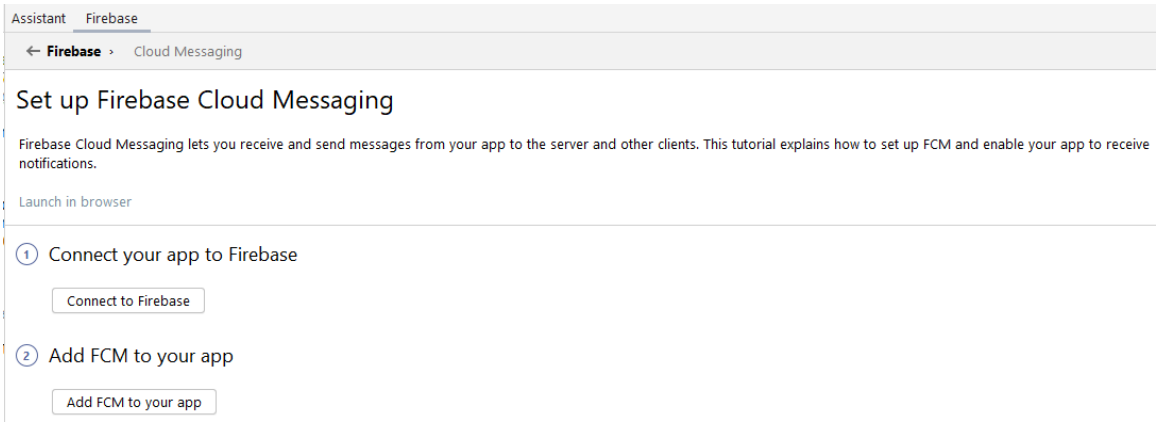

Figura 2.26. Asistente de Android Studio para el servicio de mensajería de Firebase

```
<uses-permission android:name="android.permission.NFC" />
```

Figura 2.27. Permiso NFC para Android

El código completo de la aplicación Android se encuentra en el ANEXO E. A manera de ejemplo se explicarán algunos detalles.

La clase `NFCAdapter` representa el adaptador NFC para el teléfono inteligente. Lo primero que se debe hacer es verificar si `NFCAdapter` está disponible, como se muestra en la Figura 2.28. En la línea 72, se crea un objeto de tipo `NFCAdapter` denominado `myNfcAdapter`.

El método `getDefaultAdapter` se usa para obtener el adaptador NFC para el dispositivo, este método devuelve el adaptador NFC o devuelve un valor nulo si el adaptador NFC no existe. Además, se muestra un texto informativo en la pantalla para indicar al usuario que puede acercar su teléfono inteligente a la etiqueta para iniciar la transferencia de datos (línea 75) siempre y cuando se compruebe que el adaptador NFC sea diferente de nulo, de lo contrario le indica al usuario que su dispositivo no es compatible con NFC (línea 86).

Si el teléfono inteligente es compatible con NFC pero no se encuentra habilitado, se ejecutará el método `mostrarDialogoActivarNFC()`, caso contrario la aplicación iniciará una animación relacionada a NFC (línea 81).

```
71          //OBTIENE EL ADAPTADOR NFC DEL TELEFONO INTELIGENTE
72          mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
73          if(mNfcAdapter != null){
74              mTextViewTelefonoHabilitadoNFCInfo
75                  .setText("MANTENGA LA ETIQUETA NFC CONTRA LA PARTE POSTERIOR " +
76                   "DE SU DISPOSITIVO MOVIL PARA ESCRIBIR EN ELLA");
77              if(!mNfcAdapter.isEnabled()){
78                  mostrarDialogoActivarNFC();
79              }
80              else{
81                  mAnimacionNfc.playAnimation();
82              }
83          }
84          else{
85              mTextViewTelefonoHabilitadoNFCInfo
86                      .setText("SU DISPOSITIVO MOVIL NO ES " +
87                      "COMPATIBLE CON LA TECNOLOGIA NFC");
88          }
```

Figura 2.28. Código para obtener el adaptador NFC del teléfono inteligente

Para emplear las funcionalidades de NFC en la aplicación implementada, se utilizará el sistema de despacho en primer plano, el cual está diseñado para manejar etiquetas cuando

la aplicación se está ejecutando. Para implementar el sistema de despacho en primer plano, primero se debe crear un *intent* `PendingIntent` para que Android pueda obtener los detalles de la etiqueta, como se muestra en la línea 92 de la Figura 2.29. Luego, se debe declarar los filtros de intención para manejar las etiquetas (línea 97 y 98).

Si los filtros registrados en el sistema de despacho en primer plano coinciden con la etiqueta, entonces la aplicación está en capacidad de manejar el *intent*. En la línea 101 se agrega la tecnología de etiqueta que se admitirá, en este caso `Ndef`.

```
92      mPendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0,
93              new Intent( packageContext: this, getClass())
94                      .addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP),  flags: 0);
95
96      //SE AÑADE LOS FILTROS DE INTENTS QUE MANEJARÁ LA ETIQUETA, SI ESTE
97      //FILTRO COINCIDE CON LA ETIQUETA ENTONCES LA APLICACION MANEJARÁ LA INTENCION
98      IntentFilter ndef = new IntentFilter(NfcAdapter.ACTION_NDEF_DISCOVERED);
99      mFilters = new IntentFilter[]{ndef};
100     //SE AÑADE LA TECNOLOGIA DE ETIQUETA QUE LA APLICACION PUEDE MANEJAR
101     mListaTech = new String[][] { new String[] { Ndef.class.getName()}};
```

Figura 2.29. Implementación del sistema de envío de etiquetas en primer plano

Para habilitar y deshabilitar el despacho en primer plano en los métodos `onPause()` y `onResume()`, se implementan dos métodos.

El método `disabledForegroundDispatch()` se usa en el manipulador de evento `onPause()` para deshabilitar el sistema de envío en primer plano temporalmente. El método `enableForegroundDispatch()` habilita el sistema de envío en primer plano con los parámetros obtenidos de `pendingIntent`, filtros de *intent* y el tipo de tecnología de etiquetas como se indica en la Figura 2.30.

```
151     @Override
152     protected void onPause() {
153         super.onPause();
154         if(mNfcAdapter != null){
155             mNfcAdapter.disableForegroundDispatch( activity: this);
156         }
157     }
158
159     @Override
160     protected void onResume() {
161         super.onResume();
162
163         if (mNfcAdapter != null) {
164             mNfcAdapter.enableForegroundDispatch( activity: this,
165                     mPendingIntent, mFilters, mListaTech);
166         }
167     }
```

Figura 2.30. Código para para habilitar y deshabilitar el sistema de envío de etiquetas en primer plano

2.3.4.1 REGISTRO DE USUARIOS

El método para el registro de un nuevo usuario tiene por nombre `registrar`, tal como se muestra en la línea 77 de la Figura 2.31. Este método no acepta ningún parámetro de entrada. Desde la actividad `Registro` se recuperará información que será almacenada en las variables `nombre`, `email`, `password`, `codigoPais` y `numeroCelular` (línea 78 a 82). En la línea 84 se verifica que no haya información nula.

En la línea 89 se llama al método `registrar` de la clase `ClienteController`, este método recibe como parámetro la instancia del nuevo usuario que se enviará al servicio web para su registro en la base de datos. Posteriormente, se llama al método `enqueue`, el cual requiere por parámetro un `Callback` de Retrofit; este método de forma asíncrona enviará la petición HTTP y devolverá la respuesta obtenida desde el servicio web la cual será del tipo `RHRespuesta` (línea 91).

```
77      private void registrar() {
78          String nombre = mTextInputNombre.getText().toString();
79          String email = mTextInputEmail.getText().toString();
80          String password = mTextInputPassword.getText().toString();
81          String codigoPais = mCountryCodePicker.getSelectedCountryCodeWithPlus();
82          String numeroCelular = mTextInputNumeroCelular.getText().toString();
83
84          if(!nombre.isEmpty() &&
85                  !email.isEmpty() &&
86                  !password.isEmpty() &&
87                  !numeroCelular.isEmpty()){
88
89              ClienteController.registrar(new Cliente(nombre,email,password,
90                              phoneNumber: codigoPais+" "+numeroCelular))
91                              .enqueue(new Callback<RHRespuesta>() {
92
93                  @Override
94                  //METODO QUE SE EJECUTA CUANDO LA PETICION TRAE DATOS
95                  public void onResponse(Call<RHRespuesta> call, Response<RHRespuesta> response) {
96                      if(response.code() == 201){
97                          Toast.makeText( context: RegistroActivity.this,
98                                  text: "La cuenta fue creada con exito",
99                                  Toast.LENGTH_LONG).show();
100                         finish();
101                     }
102                     else{
103                         Toast.makeText( context: RegistroActivity.this,
104                                 text: "Ocurrió un problema al registrar el usuario",
105                                 Toast.LENGTH_LONG).show();
106                     }
107                 }
108                 //METODO QUE SE EJECUTA CUANDO LA PETICIÓN FALLA
109                 @Override
110                 public void onFailure(Call<RHRespuesta> call, Throwable t) {
111                     Toast.makeText( context: RegistroActivity.this,
112                             text: "Registro fallido", Toast.LENGTH_LONG).show();
113                 }
114             });
115         }
116         else{
117             Toast.makeText( context: this,  text: "Ingrese todos los campos",
118                     Toast.LENGTH_LONG).show();
119         }
120     }
121 }
```

Figura 2.31. Función para registrar un nuevo usuario

Adicionalmente, se requiere que se defina el método que se ejecutará al obtener la respuesta, el mismo que se denomina `onResponse` (línea 95), y mediante el objeto `response` de dicho método, se verificará si la respuesta por parte del servicio web es exitosa (línea 96); es decir, si se ha registrado de forma correcta un nuevo usuario en la base de datos, además, mediante un mensaje se indicará al usuario que su cuenta fue creada (línea 98).

En la línea 103 al recibir un código HTTP diferente de 201 se mostrará un mensaje indicando que ocurrió un problema con el registro de usuario.

El método `onFailure` (línea 110) se ejecutará al presentar un problema de conexión a Internet o errores de conversión de formato JSON.

2.3.4.2 INICIAR SESIÓN

La Figura 2.32 presenta una parte del código del proceso de inicio de sesión en la aplicación. Una vez que el servicio web retorne a la aplicación una respuesta de que el usuario se ha autenticado y registrado su nuevo *token* en la base de datos de forma exitosa, la aplicación utilizará las funcionalidades de Firebase para suscribir al usuario al tópico `mascotas-desaparecidas`, proceso que se realizará al llamar al método `suscribrirMascotaDesaparecida` de la clase `NotificacionController` (línea 164). Posterior a esto, se usa la librería `SharedPreferences` para almacenar de manera local los datos del usuario autenticado, entregados por el servicio web, estos son: id del usuario, su nombre y número de celular, además se almacena un valor booleano que indicará si el usuario inició sesión. Este valor será de utilidad para que el usuario no tenga que volver a ingresar sus credenciales de inicio de sesión cuando el usuario cierre y abra nuevamente la aplicación (líneas 171 a 174).

Finalmente, se crea un objeto del tipo *Intent* que permite abrir una nueva actividad, en este caso la pantalla Home de la aplicación. Además, en el objeto `intent` se establecen dos banderas: `FLAG_ACTIVITY_NEW_TASK` y `FLAG_ACIVITY_CLEAR_TASK` para indicar que la actividad a iniciar formará parte de una nueva pila de actividades y que todas las actividades anteriores a la iniciada se eliminarán, respectivamente (líneas 176 a 179).

La línea 180 inicia la nueva actividad cuando se invoca al método `startActivity`, el cual recibe como parámetro el objeto `intent`, instanciado anteriormente.

```
159     /*
160     EL USUARIO AUTENTICADO SE SUSBRIBE AL TEMA MASCOTAS DESAPARECIDAS PARA RECIBIR
161     NOTIFICACIONES DE LAS MASCOTAS REPORTADAS COMO DESAPARECIDAS
162      */
163
164     NotificacionController.suscribirMascotaDesaparecida();
165
166     /*
167     SE ALMACENAN DE MANERA LOCAL LOS DATOS DEL USUARIO AUTENTICADO ATRAVES DE LA
168     CLASE SHAREDPREFERENCES
169      */
170
171     mAppSharedPreferences.actualizarEstadoAuth(true);
172     mAppSharedPreferences.guardarNombreCliente(nombreCliente);
173     mAppSharedPreferences.guardarIdCliente(idCliente);
174     mAppSharedPreferences.guardarNumeroCelular(numeroCelular);
175
176     Intent intent = new Intent( packageContext: PrincipalActivity.this,
177                                 HomeActivity.class);
178     //SI EL USUARIO INGRESA AL HOME ACTIVITY NO PODRA REGRESAR AL PRINCIPAL ACTIVITY
179     intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
180     startActivity(intent);
```

Figura 2.32. Parte del código del proceso de inicio de sesión

2.3.4.3 ESCRITURA DE DATOS EN LA ETIQUETA NFC

Cuando el teléfono inteligente detecta que la tarjeta NFC esta próxima al dispositivo, el sistema de envío en primer plano verificará que el filtro de *intent* establecido coincida con el *intent* que se recibe cuando el teléfono inteligente escanea una etiqueta [34]. Si coinciden los *intents*, la aplicación ejecutará el método `onNewIntent()`, como se indica en la línea 171 de la Figura 2.33.

En la línea 175 se crea el objeto de tipo `Tag` que representa la instancia de la etiqueta escaneada. El método `getParcelableExtra` se usa para recuperar los datos del *intent* recibido al escanear la etiqueta, además, el parámetro `NfcAdapter.EXTRA_TAG` se utiliza para obtener la información de dicha etiqueta (línea 175).

Después de obtener la instancia de la etiqueta escaneada, se prepara el mensaje NDEF con la información a escribir en la etiqueta. Estas operaciones la realizan los métodos `crearMensajeNdef()` y `escrirMensajeNdef()` en las líneas 178 y 182, respectivamente.

El método `crearMensajeNdef()` recibe como parámetro: el identificador único de la mascota, el número de celular del propietario y las coordenadas del hogar de la mascota, con estos valores se formará un objeto JSON con propiedades tal como se muestra en las líneas 53 a 55 de la Figura 2.34.

```
170     @Override
171     public void onNewIntent(Intent intent) {
172         super.onNewIntent(intent);
173         try{
174
175             Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
176
177             NdefMessage newMessage = ClienteController
178                     .crearMensajeNdef(mExtraIdPet,
179                                       mAppSharedPreferences.obtenerNumeroCelular(),
180                                       mPetHomeLatLng);
181
182             mostrarResultadoEscritura(ClienteController.escribirMensajeNdef(newMessage, tag));
183         }
184         catch(Exception e) {
185             Toast.makeText( context: this,  text: "Ocurrio un error", Toast.LENGTH_LONG).show();
186         }
187
188     }
```

Figura 2.33. Método `onNewIntent` – Actividad `EscribirEtiqueta`

En la línea 58 se crea el registro NDEF del tipo MIME `application/json` y se agrega el objeto JSON en formato de arreglo de bytes, el cual representa la carga útil del registro NDEF. Por último, este método devuelve el mensaje NDEF creado y conformado por un solo registro (línea 61).

```
50  @   public static NdefMessage crearMensajeNdef(int idPet, String phoneNumber, LatLng petHomeCoordinates){
51
52          JsonObject petInfo = new JsonObject();
53          petInfo.addProperty( property: "id",idPet);
54          petInfo.addProperty( property: "tel",phoneNumber);
55          petInfo.addProperty( property: "geo", value: petHomeCoordinates.latitude + "," +
56                                                  petHomeCoordinates.longitude);
57
58          NdefRecord recordPetInfo = NdefRecord.createMime( mimeType: "application/json",
59                                                            petInfo.toString().getBytes());
60
61          return new NdefMessage(new NdefRecord[]{recordPetInfo});
62
63      }
```

Figura 2.34. Método para crear un mensaje NDEF

El método `escribirMensajeNdef()` recibe como parámetro: el mensaje NDEF y la instancia de la etiqueta escaneada, como se muestra en la línea 65 de la Figura 2.35. En la línea 67 se crea el objeto del tipo `Map` que almacenará el resultado de la operación de escritura de la etiqueta. En la línea 69 se obtiene la longitud del mensaje NDEF y se lo almacena.

En la línea 72 se obtiene una instancia del tipo `Ndef` para la etiqueta escaneada. Posteriormente, se verifica que el objeto `ndef` sea diferente de nulo; es decir; que la etiqueta sea compatible con el formato NDEF. Si es compatible, se utilizarán los métodos de la clase `Ndef` para habilitar las operaciones de E/S en la etiqueta; de lo contrario, se

devolverá la variable `infoEscritura`; indicando en uno de sus parámetros que no se pudo guardar datos en la etiqueta NFC tal como se muestra en la línea 94.

El método `connect()` de la clase NDEF (línea 74) permite abrir una comunicación con la etiqueta NFC. Luego, se utilizan los métodos: `isWritable()` y `getMaxSize()` para verificar que la etiqueta permita almacenar datos y que su capacidad de memoria sea lo suficientemente grande para almacenar el mensaje NDEF, respectivamente.

Al estar la etiqueta NFC protegida contra la escritura o si el mensaje NDEF a almacenar ocuparía un mayor espacio en la memoria de la etiqueta, la operación de escritura de datos se cancela; devolviendo la variable `infoEscritura` con información del motivo por el cual el proceso de escritura de datos no resultó exitoso (línea 77 y 83).

Para escribir datos con formato NDEF en la etiqueta NFC, se emplea el método `writeNdefMessage()` y para completar el proceso de escritura se cierra la comunicación con la etiqueta NFC mediante el método `close()`; devolviendo la variable `infoEscritura` (líneas 87 a 96).

```
65 @    public static Map<String, String> escribirMensajeNdef(NdefMessage mensaje, Tag tag) {
66
67          Map<String, String> infoEscritura = new HashMap<>();
68
69          int tamano = mensaje.toByteArray().length;
70
71          try {
72              Ndef ndef = Ndef.get(tag);
73              if (ndef != null) {
74                  ndef.connect();
75                  if (!ndef.isWritable()) {
76                      infoEscritura.put("estado","NOK");
77                      infoEscritura.put("mensaje","La etiqueta es de solo lectura");
78                      return infoEscritura;
79                  }
80                  if (ndef.getMaxSize() < tamano) {
81                      infoEscritura.put("estado","NOK");
82                      infoEscritura
83                          .put("mensaje","Los datos a ser escritos ("+ tamano +" bytes) " +
84                                  "superan el tamaño de la etiqueta ("+ ndef.getMaxSize() +" bytes)");
85                      return infoEscritura;
86                  }
87                  ndef.writeNdefMessage(mensaje);
88                  ndef.close();
89                  infoEscritura.put("estado","OK");
90                  infoEscritura.put("mensaje", "Los datos se han escrito exitosamente");
91                  return infoEscritura;
92              }
93              else{
94                  infoEscritura.put("estado","NOK");
95                  infoEscritura.put("mensaje", "No se pudo escribir los datos a la etiqueta");
96                  return infoEscritura;
97              }
98
99          } catch (Exception e) {
100             infoEscritura.put("estado","NOK");
101             infoEscritura.put("mensaje", "No se pudo escribir los datos a la etiqueta");
102             return infoEscritura;
103         }
104     }
```

Figura 2.35. Método para escribir un mensaje NDEF

2.3.4.4 LECTURA DE DATOS DE LA ETIQUETA NFC

De manera similar al proceso de escritura de datos, el usuario debe acercar la etiqueta NFC contra la parte posterior del teléfono inteligente, en ese instante, el teléfono detectará la etiqueta y lanzará el método `onNewIntent()`, como se indica en la Figura 2.36.

En la línea 162 se obtienen los datos del *intent* generado ante la presencia de una etiqueta NFC, usando el método `getParcelableArrayExtra()`. Este método obtendrá la información contenida en la etiqueta al recibir como parámetro `NfcAdapter.EXTRA_NDEF_MESSAGES`, dicha información se almacenará en la variable `rawMensajes`. Posteriormente, al llamar al método `obtenerMensajeNdef()` de la clase `ClienteController` se obtiene el mensaje NDEF contenido en la etiqueta NFC (línea 163). El objeto `message` de tipo `NdefMessage` llama al método `getRecords`() para obtener el registro del mensaje NDEF (línea 164). Después, se comprueba si el tipo de registro corresponde al tipo MIME `application/json`. Si el tipo de registro corresponde a `application/json`, se procede a obtener la carga útil del registro y se la almacena en la variable `payload` (línea 168), de lo contrario se mostrará al usuario que los datos de la etiqueta no se encuentran en formato JSON (línea 187).

```
155     //METODO QUE SE EJECUTA AUTOMATICAMENTE CUANDO LA ETIQUETA ES ESCANEADA
156     @Override
157     public void onNewIntent(Intent intent) {
158         super.onNewIntent(intent);
159
160         //SE OBTIENE LOS DATOS CONTENIDOS EN LA INTENCION
161         try{
162             Parcelable[] rawMensajes= intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
163             NdefMessage message = ClienteController.obtenerMensajeNdef(rawMensajes);
164             NdefRecord record = message.getRecords()[0];
165             String tipo = new String(record.getType());
166
167             if(tipo.equals("application/json")){
168                 String payload = new String(record.getPayload());
169                 JsonParser parser = new JsonParser();
170                 JsonObject mascotaInfoJSON = (JsonObject) parser.parse(payload);
171
172                 idMascota = Integer.parseInt(mascotaInfoJSON.get("id")
173                                 .toString().replace( oldChar: '"', newChar: ' ').trim());
174                 numeroCelular = mascotaInfoJSON.get("tel")
175                                 .toString().replace( oldChar: '"', newChar: ' ').trim();
176                 String[] coordenadas = mascotaInfoJSON.get("geo")
177                                 .toString().split( regex: ",");
178                 double latitud = Double.parseDouble(coordenadas[0]
179                                 .replace( oldChar: '"', newChar: ' ').trim());
180                 double longitud = Double.parseDouble(coordenadas[1]
181                                 .replace( oldChar: '"', newChar: ' ').trim());
182                 hogarMascotaLatLng = new LatLng(latitud,longitud);
183
184                 irADetalleLecturaONotificacionMascotaEncontradaActivity();
185             }
186             else{
187                 Toast.makeText( context: this,
188                         text: "Los datos en la etiqueta no se encuentra en formato Json",
189                             Toast.LENGTH_LONG).show();
190             }
191         }
192         catch(Exception e){
193             Toast.makeText( context: this,  text: "Error"+ e.toString(), Toast.LENGTH_LONG).show();
194         }
195
196     }
```

Figura 2.36. Método `onNewIntent` - Actividad `LecturaEtiqueta`

Una vez obtenida la carga útil del registro NDEF y almacenada en la variable `payload` del tipo `String`, se procede a realizar un *casting* para convertir la secuencia de caracteres que conforman la sintaxis de un JSON a un objeto del tipo `JsonObject` (línea 170). Posterior a esto, se obtienen los valores del objeto `mascotaInfoJSON` al especificar el nombre de sus propiedades: `id`, `tel` y `geo`, esta última está compuesta de la latitud y longitud de la ubicación del hogar de la mascota, las cuales conformarán los parámetros de entrada del constructor de la clase `LatLng` para instanciar el objeto `hogarMascotaLatLng` (líneas 172 a 182).

Después de la lectura de datos de la etiqueta, se invoca al método de la línea 184 para abrir una actividad que mostrará una interfaz de usuario con sus respectivos componentes visuales en los cuales se situarán los datos de la mascota y la ubicación de su hogar.

2.3.4.5 NOTIFICACIÓN GENERADA AL ENCONTRAR UNA MASCOTA

Cuando un usuario del sistema encuentra una mascota, el usuario debe ejecutar una acción en la aplicación Android que permite enviar una petición HTTP al servicio de Firebase *Cloud Messaging* (FCM) con la información de la mascota encontrada y el *token* del propietario de la mascota. Al recibir la petición HTTP, el servicio de FCM enruta el mensaje al dispositivo que pertenezca el *token* recibido, en este caso al teléfono inteligente del propietario de la mascota.

El *token*, el cual identifica de forma única la instancia de la aplicación en un teléfono inteligente, se obtendrá del servicio de FCM cada vez que un usuario inicie sesión en la aplicación Android y se almacenará en la base de datos del prototipo. Cabe mencionar que de manera predeterminada el SDK de FCM genera un *token* automáticamente para la aplicación Android cuando esta es iniciada, sin embargo, se deshabilitó esta configuración al definir en el archivo `AndroidManifest.xml` el metadato de la Figura 2.37, para que los *token* puedan ser generados al invocar al método `getToken()` de la clase `FirebaseMessaging` cuando un usuario inicie sesión.

```
<meta-data
    android:name="firebase_messaging_auto_init_enabled"
    android:value="false" />
```

Figura 2.37 Metadato para deshabilitar la inicialización automática de FCM

La Figura 2.38 ilustra lo anteriormente mencionado en lo siguiente:

1. Al autenticarse el usuario en la aplicación Android de manera exitosa, se invoca al método `getToken()` para obtener un identificador de la aplicación y con el cual poder enviar y recibir una notificación.
2. El servicio de FCM envía el *token* generado a la aplicación Android, la cual a través del servicio web del prototipo almacena el *token* recibido en la base de datos del prototipo.
3. Un usuario que encontró una mascota ejecuta una acción en la aplicación Android la cual envía la información de la mascota incluido el *token* de su propietario al servicio de FCM.
4. El servicio de FCM se encarga de enrutar el mensaje a la aplicación Android que corresponda el *token* recibido.

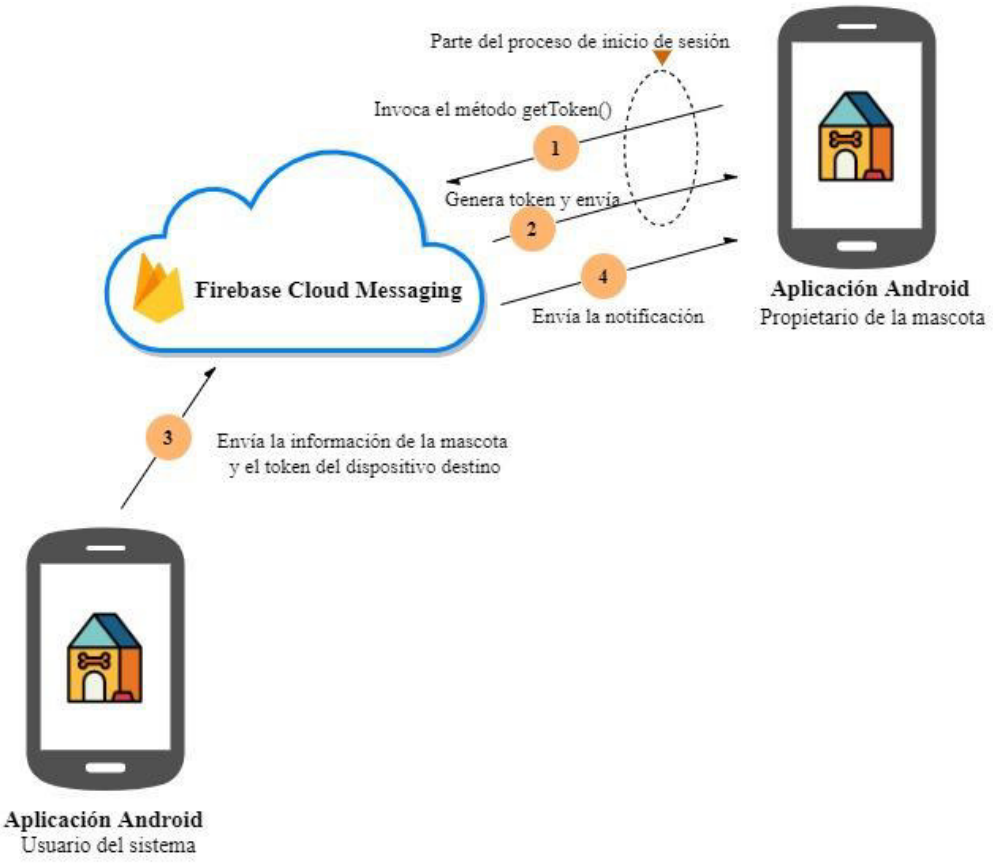

Figura 2.38. Proceso de envío de notificación usando Firebase *Cloud Messaging*

Dentro del archivo `AndroidManifest.xml` como se indica en la Figura 2.39, se debe declarar un servicio que extienda de `FirebaseMessagingService` e incorporar un filtro de *intent* para recibir las notificaciones de FCM y administrar su contenido.

```xml
<service
    android:name=".services.MyFirebaseMessagingService">
    <!-- FILTRO DE INTENTS PARA RECIBIR NOTIFICACIONES CUANDO LA APP
         SE ENCUENTRE EN SEGUNDO PLANO  -->
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Figura 2.39. Servicio para recibir notificaciones de NFC

Para gestionar los notificaciones recibidas se debe sobrescribir el método `onMessageReceived()` de la clase `FirebaseMessagingService` como se indica en la Figura 2.40. En las primeras líneas del método se extrae la carga útil del mensaje y el contenido del parámetro `title` que contiene la información del título de la notificación y el parámetro `body` que contiene el cuerpo de la notificación a ser generada (líneas 33 a 35).

```
29      @Override
30      public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
31          super.onMessageReceived(remoteMessage);
32
33          Map<String, String> data = remoteMessage.getData();
34          String title = data.get("title");
35          String body = data.get("body");
36
37          if (title != null) {
38
39              if(title.contains("Mascota encontrada")){
40
41                  int  idCliente = Integer.valueOf(data.get("idCliente"));
42                  String nombreMascota = data.get("nombreMascota");
43                  String numeroCelular = data.get("numeroCelular");
44                  double mascotaLat = Double.parseDouble(data.get("mascotaLat"));
45                  double mascotaLng  = Double.parseDouble(data.get("mascotaLng"));
46
47                  LatLng petLatLng = new LatLng(mascotaLat, mascotaLng);
48                  mostrarNotificacionMascotaEncontrada(title, body, idCliente,
49                                                       numeroCelular, nombreMascota, petLatLng);
50              }
51              else {
52
53                  int idMascota = Integer.valueOf(data.get("idMascota"));
54                  String nombreMascota = data.get("nombreMascota");
55                  double mascotaLat = Double.parseDouble(data.get("mascotaLat"));
56                  double mascotaLng = Double.parseDouble(data.get("mascotaLng"));
57
58                  LatLng mascotaLatLng = new LatLng(mascotaLat, mascotaLng);
59                  mostrarNotificacionMascotaDesaparecida(title, body, idMascota,
60                                                         nombreMascota, mascotaLatLng);
61              }
62          }
63      }
```

Figura 2.40. Código para gestionar el contenido de una notificación recibida

Posteriormente, se verifica que el título de la notificación no sea nulo y el contenido de la variable `title` coincida con el valor de los caracteres especificados (línea 39). Si estos dos condicionales se cumplen, se procede a obtener el contenido de los demás parámetros recibidos en la notificación estos son: `idCliente`, `nombreMascota`, `numeroCelular`,

`mascotaLat` y `mascotaLng`, así como a almacenar cada parámetro en diferente variable dependiendo de su tipo de dato (líneas 41 a 47).

En la línea 48 se llama al método que permite mostrar al usuario una ventana flotante, denominada notificación emergente, en la barra de estado de un teléfono inteligente. Por otra parte, las líneas 53 a 59 representa el código que se ejecuta al recibir una notificación cuando una mascota sea reportada como desaparecida.

En la Figura 2.41 se presentan algunas de las interfaces gráficas de usuario implementadas en Android Studio que corresponden a las funcionalidades de iniciar sesión, registrar un nuevo usuario, enviar la notificación de una mascota encontrada, así como, leer y escribir datos en una etiqueta NFC.

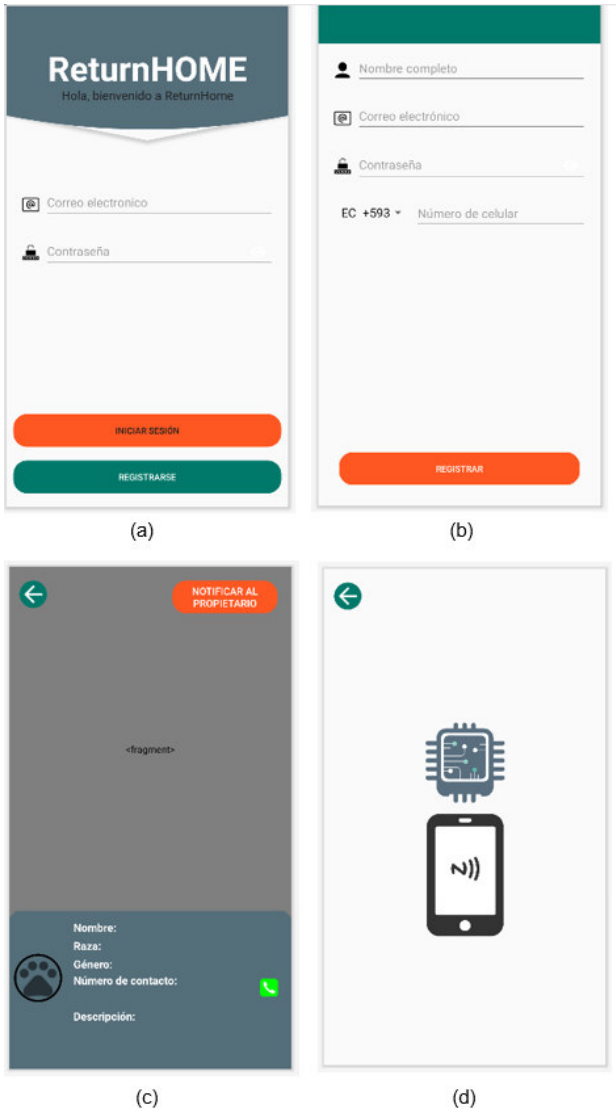

Figura 2.41. Interfaz gráfica de usuario para (a) iniciar sesión, (b) registrar un nuevo usuario, (c) notificar una mascota encontrada (d) leer y escribir datos de una etiqueta NFC

3. RESULTADOS Y DISCUSIÓN

Como evidencia del uso de la metodología Kanban, en la Tabla 3.1 se presentan los diferentes requerimientos que fueron cumplidos para el desarrollo del prototipo con sus respectivas tareas, las cuales se ubican en la columna *Done* del tablero Kanban.

Tabla 3.1 Tareas realizadas (Parte 1 de 4)

Requerimiento	Tarea	To Do	Doing	Done
Registrar usuarios				Crear la vista para el registro de usuarios
				Añadir las cajas de texto para ingresar el nombre, correo electrónico y número de celular del usuario
				Incorporar un botón para iniciar el proceso de registro
				Mostrar un mensaje para indicar si el registro de usuario se cumplió con éxito o presentó algún inconveniente
Autenticar usuarios				Crear la vista para la autenticación de usuarios
				Añadir los campos de texto para ingresar el correo electrónico y número de celular del usuario
				Añadir un botón para iniciar el proceso de autenticación
Gestionar la información del usuario				Crear la vista para actualizar la información del usuario
				Añadir los campos de texto para editar el nombre, correo electrónico y número de celular del usuario
				Añadir un botón para iniciar el proceso de actualización
				Mostrar un mensaje para indicar si la actualización de la información del usuario fue exitosa o presentó algún inconveniente
				Crear la vista para actualizar la contraseña del usuario
				Añadir los campos de texto para ingresar la contraseña actual, nueva contraseña y la confirmación de la nueva contraseña
				Añadir un botón para iniciar el proceso de actualización de la contraseña
				Mostrar un mensaje para indicar si la actualización de la contraseña fue exitosa o presentó algún inconveniente
Registrar mascotas				Crear la vista para el registro de mascotas
				Añadir los campos de texto para ingresar el nombre, raza y una descripción de la mascota
				Añadir los botones de opción para elegir el género de la mascota
				Añadir un botón para iniciar el registro de la mascota

Tabla 3.1. Tareas realizadas (Parte 2 de 4)

Requerimiento	Tarea	To Do	Doing	Done
Registrar mascotas				Crear la vista para mostrar la lista de mascotas registradas por el usuario
Gestionar la información de mascotas				Crear la vista para actualizar la información de una mascota
				Añadir los campos de texto para editar el nombre, raza y la descripción de una mascota
				Añadir los botones de opción para editar el género de una mascota
				Añadir un botón para iniciar el proceso de actualización de una mascota
				Mostrar un mensaje para indicar si la actualización de la información de la mascota fue exitosa o presentó algún inconveniente
				Crear la vista para eliminar la información de una mascota
				Añadir un botón para eliminar una mascota
				Incorporar un cuadro de diálogo para confirmar la eliminación de una mascota
Almacenar información de una mascota en una etiqueta NFC				Crear la vista para seleccionar una mascota registrada y la ubicación de su hogar
				Incorporar un mapa para seleccionar la ubicación del hogar de la mascota
				Añadir una lista desplegable para seleccionar una de las mascotas registradas por su propietario
				Añadir un campo para ingresar la dirección del hogar de la mascota
				Añadir un botón para confirmar la mascota y su ubicación seleccionada
				Crear la vista para mostrar la información de la mascota, la ubicación de su hogar, anteriormente seleccionados
				Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario
				Añadir un botón para iniciar el proceso de transferencia de datos hacia la etiqueta NFC
				Crear la vista para iniciar la transferencia de datos hacia la etiqueta NFC
				Añadir una vista de texto para indicar al usuario si su teléfono inteligente es capaz de realizar la operación de escritura NFC
				Mostrar un cuadro de diálogo para informar si la escritura NFC fue exitosa o presentó algún inconveniente

Tabla 3.1. Tareas realizadas (Parte 3 de 4)

Requerimiento	Tarea	To Do	Doing	Done
Mostrar información obtenida desde una etiqueta NFC				Crear la vista para iniciar la transferencia de datos de la etiqueta NFC a la aplicación Android
				Añadir una vista de texto para indicar al usuario si su teléfono inteligente es capaz de realizar la operación de lectura NFC
				Crear la vista para mostrar la información de la mascota contenida en la etiqueta NFC
				Incorporar un mapa para mostrar al usuario la ubicación del hogar de la mascota
				Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario
Listar mascotas desaparecidas				Crear la vista para mostrar todas las mascotas reportadas como desaparecidas del sistema
				Añadir las vistas de texto para mostrar al usuario el nombre y raza de las mascotas desaparecidas
Reportar una mascota encontrada				Añadir un botón para reportar una mascota como encontrada y quitarla de la lista de mascotas desaparecidas
Notificar una mascota encontrada				Crear la vista para iniciar la transferencia de datos de la etiqueta NFC a la aplicación Android
				Crear la vista para mostrar la información de la mascota encontrada proveída por la etiqueta NFC
				Incorporar un mapa para mostrar al usuario la ubicación del hogar de la mascota que fue encontrada y su ubicación actual
				Añadir las vistas de texto para mostrar el nombre, raza y género de la mascota, como también el número de celular del propietario
				Añadir un botón para notificar al propietario que su mascota fue encontrada
				Añadir un botón para comunicarse directamente con el propietario de la mascota mediante llamada telefónica
				Crear la vista para mostrar al propietario de la mascota la información del usuario quien encontró a su mascota y la ubicación de la misma
				Incorporar un mapa para mostrar al propietario la ubicación donde fue encontrada su mascota

Tabla 3.1. Tareas realizadas (Parte 4 de 4)

Requerimiento	Tarea	To Do	Doing	Done
Notificar una mascota encontrada				Añadir las vistas de texto para mostrar al propietario el nombre, correo electrónico y numero de celular del usuario quien encontró la mascota
				Añadir un botón para que el propietario de la mascota encontrada se comunique mediante llamada telefónica con el usuario quien encontró su mascota
Notificar una mascota desaparecida				Crear la vista para notificar una mascota como desaparecida
				Incorporar un mapa para seleccionar la ubicación en donde fue vista por última vez una mascota seleccionada
				Añadir un botón para enviar la notificación de mascota desaparecida a todos los usuarios del sistema
				Crear la vista para mostrar la información de la mascota desaparecida
				Incorporar un mapa para mostrar al usuario la ubicación donde fue vista por última vez la mascota desaparecida
				Añadir las vistas de texto para mostrar al usuario el nombre, raza, género y descripción de la mascota desaparecida

Para comprobar el correcto funcionamiento del prototipo se realizaron pruebas a cada una de las funcionalidades del servicio web, así como de la aplicación Android; además se ingresó información en cada uno de los campos de texto de las diferentes pantallas para validar que no existan campos vacíos cuando la información sea obligatoria.

Finalmente, se realizó un modelo de entrevista, y con base en este modelo se aplicaron entrevistas a los propietarios de mascotas para validar el funcionamiento del prototipo.

De las pruebas realizadas, así como de las entrevistas se detectaron errores y se corrigieron los mismos.

3.1 PRUEBAS DE FUNCIONAMIENTO

3.1.1 REGISTRO DE USUARIO

En la Figura 3.1 (a) se observa el mensaje (ver recuadro rojo) al oprimir el botón `Registrar` cuando no se ha completado uno o todos los campos requeridos para el registro de un nuevo usuario, mientras que en la Figura 3.1 (b) se presenta un mensaje (ver

recuadro verde) indicando que se intenta registrar los datos usando un correo previamente registrado.

Figura 3.1. Validación de datos al crear un nuevo usuario (a) campos vacíos y (b) email anteriormente registrado

Una vez completos todos los campos, como se muestra en la Figura 3.2, se oprimió el botón Registrar para enviar los datos de registro al servicio web.

Figura 3.2. Pantalla con la información de un usuario a ser registrado

La Figura 3.3 muestra la información del usuario registrado en la base de datos: identificador único del usuario, nombre, correo electrónico y número de celular; tal cual como se ingresó en la pantalla de registro del usuario, sin embargo, la contraseña se presenta cifrada con fines de seguridad.

idCliente	nombre	email	password	numeroCelular
70	Juan Gómez	juan_gomez@hotmail.com	$2y$10$eCIXqsUn7.dpmZ.Pv€	+593 0993602133

Figura 3.3. Usuario registrado en la base de datos

3.1.2 INICIAR SESIÓN

Par iniciar sesión en la aplicación, el usuario debe ingresar en la pantalla de ingreso sus credenciales Figura 3.4. Si los campos: correo electrónico y contraseña, se encuentran vacíos o si se ingresa de manera errónea una de las credenciales, cuando se oprima el botón `Iniciar Sesión`, la aplicación mostrará un mensaje de error, como se indica en la Figura 3.4.

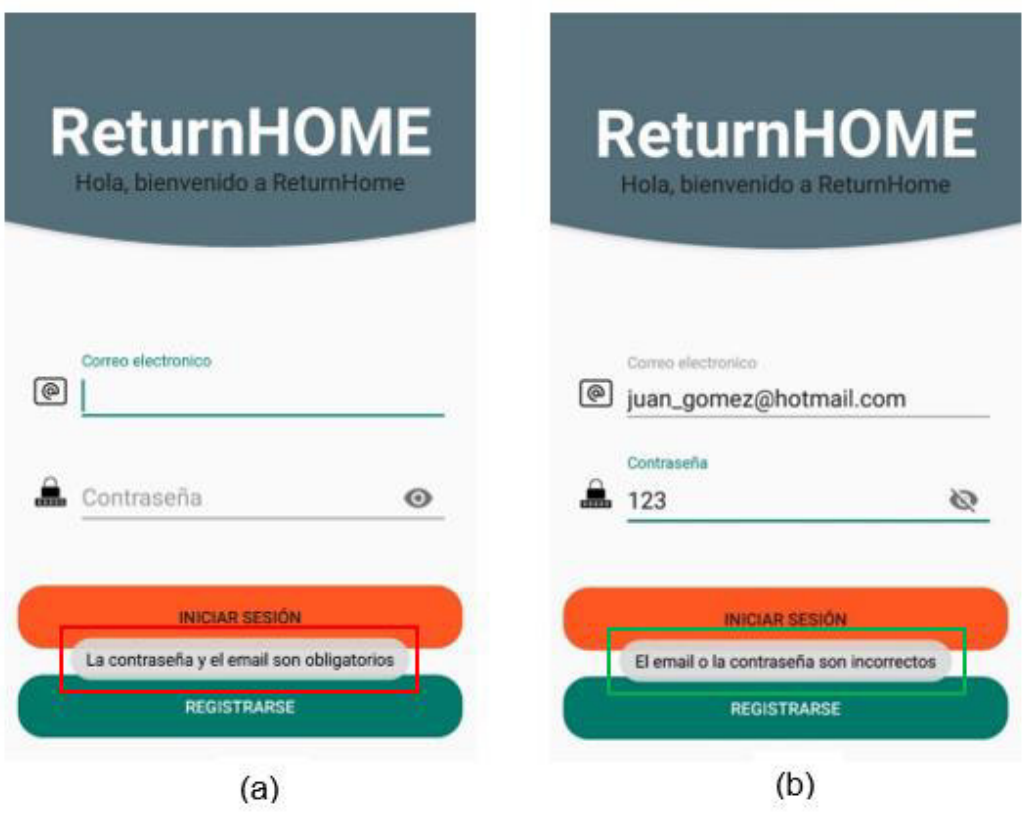

Figura 3.4. Validación de datos al iniciar sesión: (a) campos vacíos, (b) email o contraseña incorrectos

Al realizar el proceso con éxito, en la base de datos se incluye el *token* generado para este inicio de sesión. Como ejemplo se presenta en la Figura 3.5 el *token* asignado para esta prueba.

idToken ▼ 1	token	idCliente
45	dX28aLN2SemPhtQBx8GYAh:APA91bEiz52X2JC3XE0OFOAoiEr...	70

Figura 3.5. *Token* registrado en la base de datos

3.1.3 REGISTRO DE UNA MASCOTA

Para verificar la validación de los campos para el registro de una mascota, se oprimió el botón `Agregar` con todos los campos vacíos y la pantalla muestra un mensaje indicando que se debe ingresar todos los campos, como se ilustra en la Figura 3.6.

Una vez que se han completado todos los campos para el registro de una mascota, como se puede observar en la Figura 3.7 (a), se oprimió el botón `Agregar` y en la pestaña `Mis Mascotas` de la pantalla Home se muestra la mascota que acaba de ser agregada, como se ilustra en la Figura 3.7 (b).

Figura 3.6. Validación para el registro de una mascota

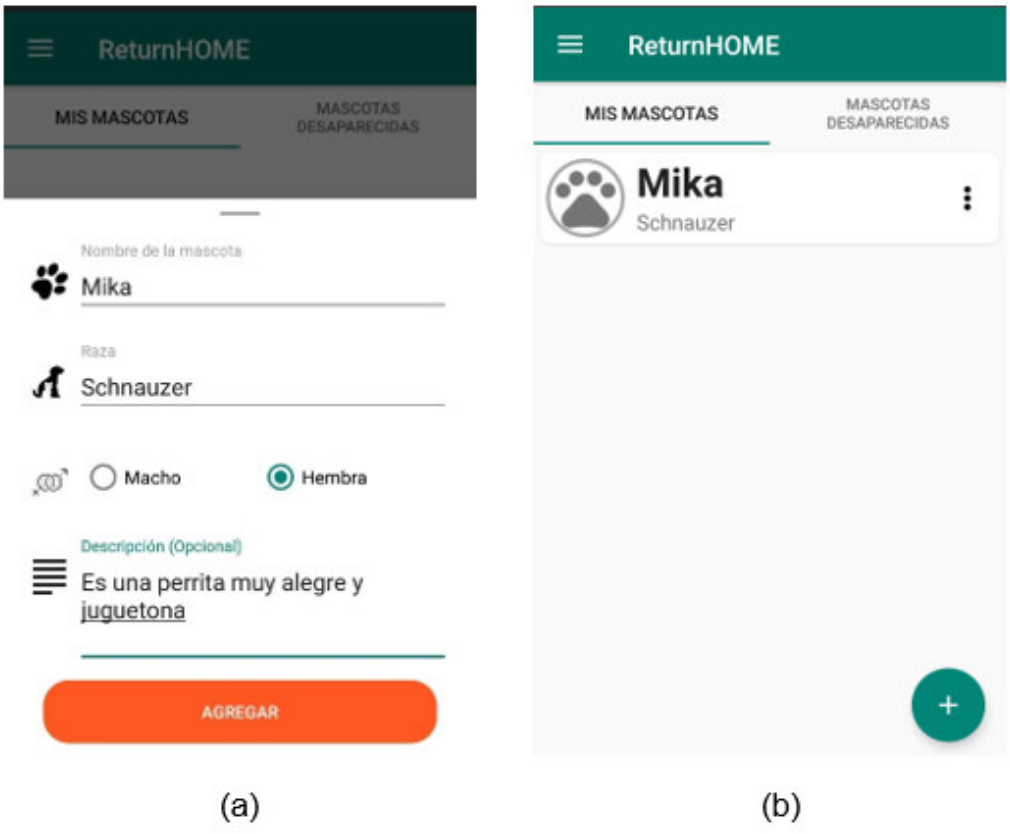

Figura 3.7. (a) Pantalla para el registro de una mascota (b) Mascota registrada

La información registrada en la base de datos correspondiente se presenta en la Figura 3.8.

idMascota ▼ 1	nombre	raza	genero	descripcion	desaparecida	idCliente
118	Mika	Schnauzer	F	Es una perrita muy alegre y juguetona	0	70

Figura 3.8. Registro de una mascota en la base de datos

3.1.4 EDITAR UNA MASCOTA

Para editar la información de una mascota, se presionó la opción `Editar` e inmediatamente se desplegó una ventana con la información de la mascota como se muestra en la Figura 3.9 (a). Posterior a esto, se eliminó la descripción de la mascota ya que es un campo opcional y para guardar se oprimió el botón `Actualizar`. Una vez hecho esto, la aplicación mostró un mensaje indicando que la actualización fue exitosa como se muestra en el recuadro rojo de la Figura 3.9 (b).

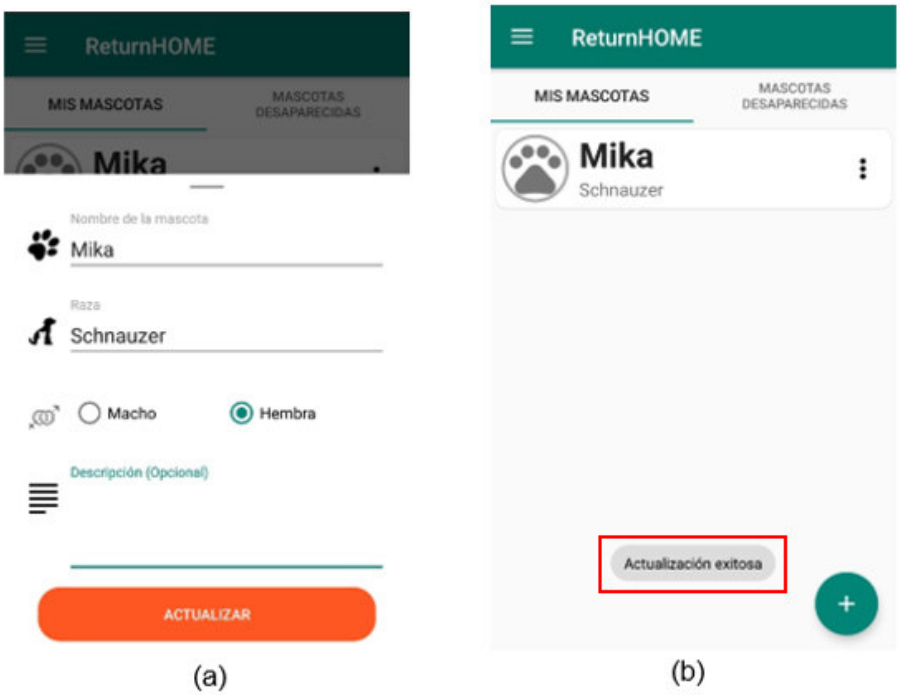

Figura 3.9. (a) Ventana emergente para la edición de una mascota (b) Mensaje de actualización exitosa de la mascota

3.1.5 ELIMINAR UNA MASCOTA

Al momento de intentar eliminar una mascota, la aplicación mostró un cuadro de advertencia para confirmar su eliminación, como se indica en la Figura 3.10.

Figura 3.10. Cuadro de diálogo para confirmar la eliminación de una mascota

3.1.6 ESCRIBIR INFORMACIÓN DE UNA MASCOTA EN UNA ETIQUETA NFC

Para guardar la información de una mascota en una etiqueta NFC, se seleccionó la mascota de interés y la ubicación de su hogar, como se indica en la Figura 3.11 (a); posterior a esto, la aplicación presenta los siguientes datos de la mascota seleccionada: nombre, raza, género y el número de celular de su propietario, al igual que la ubicación de su hogar (ver la Figura 3.11 (b)).

Posteriormente, la aplicación presenta un texto que indica que se puede acercar el teléfono inteligente a la etiqueta NFC para guardar la información. Una vez hecho esto, la aplicación presenta un cuadro indicando que la acción fue exitosa, como se observa en la Figura 3.11 (c).

Para visualizar los datos que se guardaron en la etiqueta, se empleó la aplicación NFC Writer, disponible en Google Play para dispositivos compatibles con NFC, se empleó la opción para leer etiquetas. El resultado se presenta en la Figura 3.12**.** Esta información se encuentra en formato JSON con claves `id`, `tel` y `geo` que almacenan el valor del id de la mascota, el número de celular de su propietario y las coordenadas geográficas de su hogar, respectivamente.

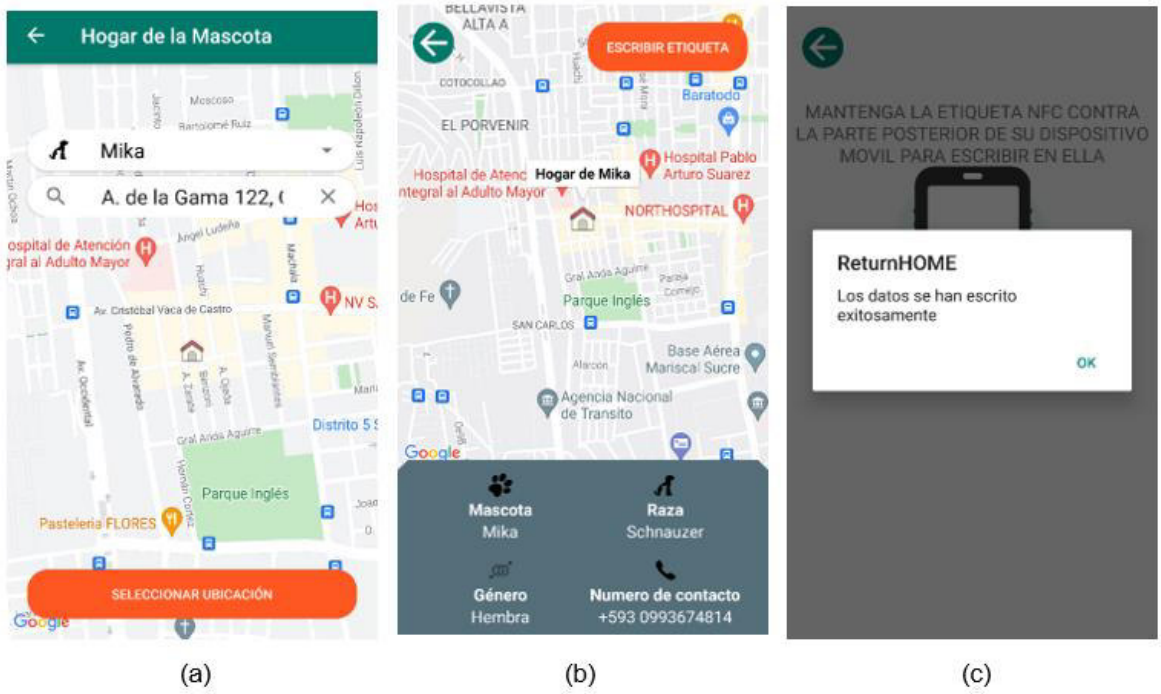

Figura 3.11. Pantallas mostradas al escribir una etiqueta NFC (a) selección de una mascota y su hogar, (b) información de la mascota seleccionada y (c) cuadro de diálogo del resultado de escritura NFC.

Figura 3.12. Información almacenada en la etiqueta NFC

3.1.7 LEER INFORMACIÓN DE UNA MASCOTA CONTENIDA EN UNA ETIQUETA NFC

Para leer la información almacenada en una etiqueta NFC, se acercó la etiqueta al teléfono inteligente al encontrarse en la pantalla de la aplicación que se muestra en la Figura 3.13 (a). Una vez leídos los datos, la aplicación cambió de pantalla, como se presenta en la Figura 3.13 (b), y presenta la información de la mascota en cuestión. Esta información incluye: nombre de la mascota, raza, género y el número de celular del propietario.

Figura 3.13. Pantallas asociadas al leer una etiqueta NFC. (a) inicio de la lectura NFC, (b) información de una mascota al leer una etiqueta NFC

3.1.8 NOTIFICACIÓN DE UNA MASCOTA ENCONTRADA

Para enviar la notificación al propietario de una mascota que fue encontrada, se oprime la opción `Encontré una mascota`. Posterior a esto, la aplicación muestra una pantalla, la cual se presenta en la Figura 3.15 (a), con indicaciones para proceder a leer los datos de la etiqueta NFC que porta la mascota en su collar (ver Figura 3.14). Después, la aplicación presenta una pantalla, como se observa en la Figura 3.15 (b), con la información de la

mascota encontrada, incluyendo el número de celular del propietario y un botón que inicia la aplicación denominada Teléfono del celular, y en la cual se incluye el número de teléfono del propietario.

La pantalla que se ilustra en la Figura 3.15 (b) también presenta un botón para enviar la notificación al propietario de la mascota, así como los marcadores de posición del hogar de la mascota y el de su ubicación actual. Una vez que se oprima en el botón `Notificar al propietario`, se envía la notificación al dueño de la mascota.

La notificación que recibirá el dueño de la mascota se presenta en la Figura 3.17. Esta notificación incluye el nombre de la mascota encontrada y su ubicación actual, así mismo incluye los siguientes botones:

- **Mostrar en Mapa:** Al oprimir este botón, la aplicación abre una pantalla, como se presenta en la Figura 3.16, con la información del usuario quien encontró la mascota y la ubicación actual de la misma, además, incluye un botón, el cual abre la aplicación Teléfono.
- **Contactar:** Esta opción muestra la pantalla de la aplicación Teléfono en la cual se coloca automáticamente el número de celular del usuario quien encontró la mascota.

Figura 3.14. Mascota con una etiqueta NFC

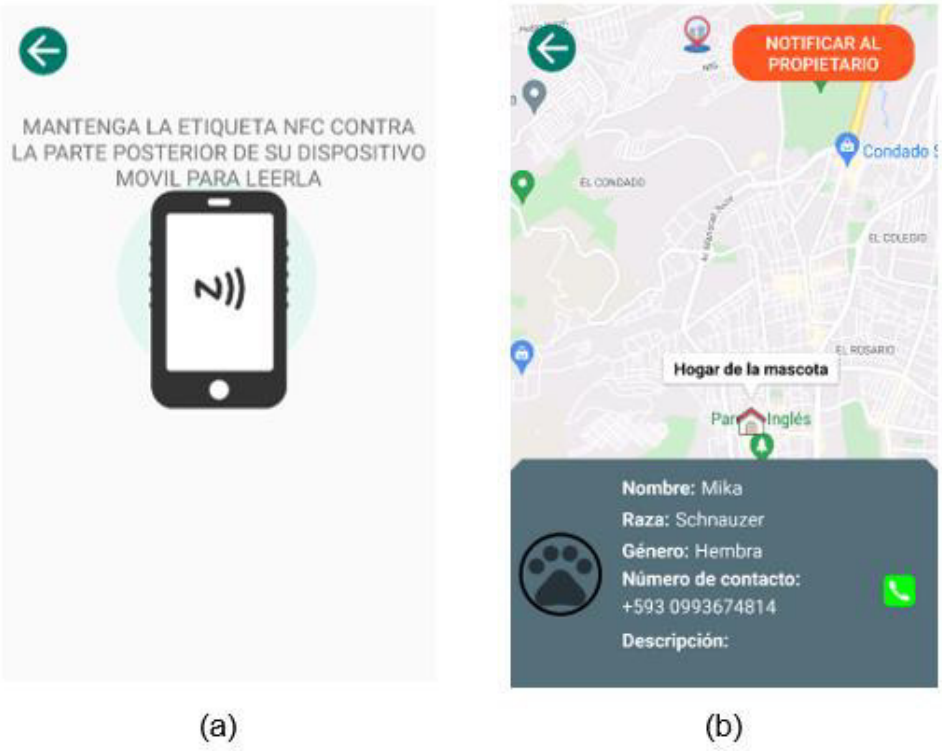

Figura 3.15. Pantalla asociada al notificar una mascota encontrada. (a) inicio de lectura NFC, (b) pantalla para enviar la notificación

Figura 3.16. Pantalla para mostrar la información del usuario quien encontró una mascota

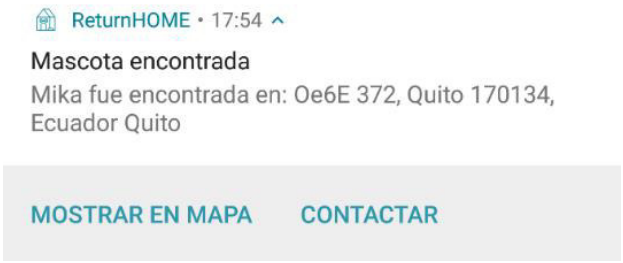

Figura 3.17. Notificación de una mascota encontrada

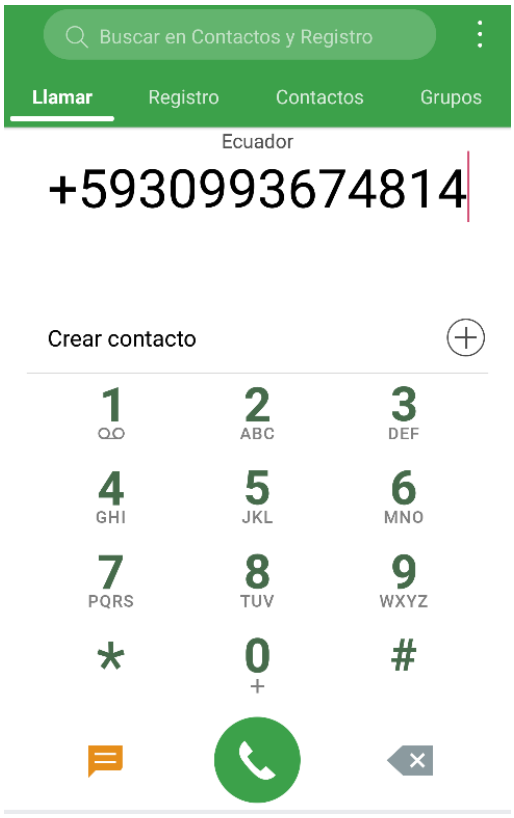

Figura 3.18. Pantalla de la aplicación de telefonía del teléfono inteligente

3.1.9 ACTUALIZAR LA CONTRASEÑA DEL USUARIO

La pantalla para actualizar la contraseña se presenta en la Figura 3.19, esta muestra un mensaje (ver recuadro rojo) al oprimir el botón `Actualizar` cuando no se ingresa ningún valor en los campos. Una vez ingresada la actual y nueva contraseña, al igual que su confirmación, el usuario debe oprimir el botón `Actualizar` y la aplicación presenta un mensaje indicando el éxito del proceso, como se ilustra en la Figura 3.20.

Figura 3.19. Validación de los campos para actualizar la contraseña

Figura 3.20. Ingreso de la contraseña del usuario

3.1.10 EDITAR LA INFORMACIÓN DEL USUARIO

La pantalla para editar la información del usuario se presenta en la Figura 3.21. En esta pantalla se presenta el nombre, correo electrónico y contraseña del usuario con la

posibilidad de editar el contenido de estos campos. Al oprimir el botón `Actualizar`, la aplicación regresa a la pantalla anterior y muestra un mensaje indicando el éxito del proceso como se ilustra en la Figura 3.22.

Figura 3.21. Pantalla para actualizar los datos del usuario

Figura 3.22. Pantalla de ajustes de la cuenta del usuario

3.1.11 ELIMINAR LA CUENTA

Al eliminar la cuenta de usuario, la aplicación muestra un cuadro de diálogo para confirmar el proceso de eliminación de cuenta. Las opciones posibles son `Si` o `No`, tal como se indica en la Figura 3.23.

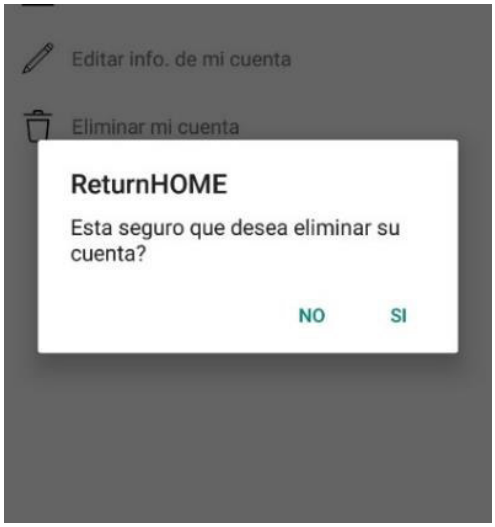

Figura 3.23. Advertencia de eliminación de cuenta

3.1.12 CERRAR SESIÓN

Al momento de cerrar sesión se presenta un cuadro de diálogo con el fin de confirmar al usuario si desea o no realizar este proceso, como se observa en la Figura 3.24.

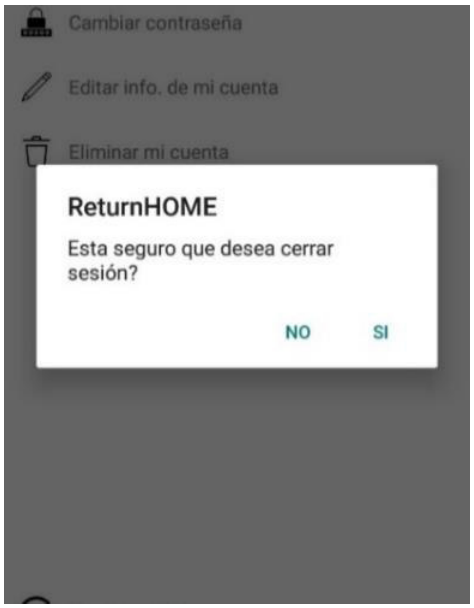

Figura 3.24. Advertencia al cerrar sesión

3.1.13 REPORTAR UNA MASCOTA COMO DESAPARECIDA

Para reportar una mascota desaparecida, el usuario debe seleccionar la ubicación donde vio por última vez a la mascota, como se indica en la Figura 3.25.

Al oprimir el botón `Seleccionar ubicación`, se envía la notificación de mascota desaparecida a todos los usuarios del sistema, como se indica en la Figura 3.26. Esta notificación incluye el lugar donde se vio por última vez a la mascota, además, de un botón denominado `Mostrar en Mapa`.

El botón `Mostrar en Mapa` abre la pantalla que se indica en la Figura 3.27, esta pantalla muestra la información de la mascota desaparecida y un marcador el cual fija la posición en donde se vio por última vez a la mascota.

Una vez reportada la mascota como desaparecida, esta se muestra en la pestaña de `Mascotas Desaparecidas`, como se indica en la Figura 3.28.

Figura 3.25. Pantalla para seleccionar la ubicación donde fue vista por última vez una mascota

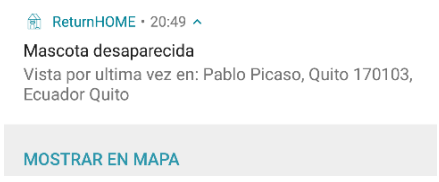

Figura 3.26. Notificación de una mascota desaparecida

Figura 3.27. Pantalla para mostrar información de una mascota reportada como desaparecida

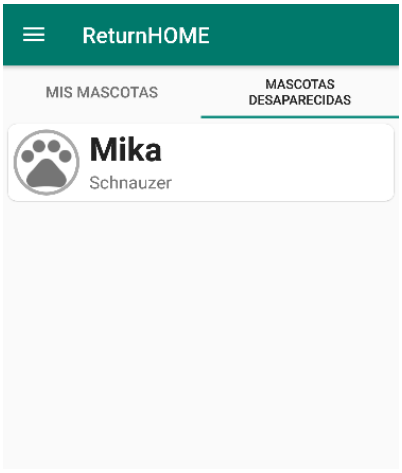

Figura 3.28. Pantalla con la lista de las mascotas desaparecidas

3.1.14 REPORTAR UNA MASCOTA COMO ENCONTRADA

Para reportar una mascota encontrada, el usuario debe oprimir la opción `Encontrada`, como se muestra en la Figura 3.29, posterior a esto, se presenta un mensaje indicando el éxito de la acción, como se ilustra en la Figura 3.30.

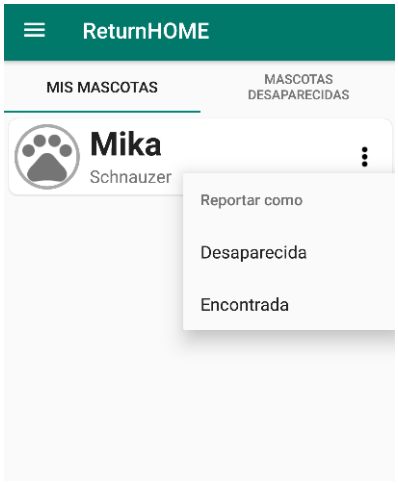

Figura 3.29. Opciones disponibles para reportar una mascota

Figura 3.30. Mensaje mostrado al reportar una mascota como encontrada

3.2 CORRECCIÓN DE ERRORES

Para validar el prototipo se generaron entrevistas para que usuarios realicen pruebas de cada una de las funcionalidades del prototipo. El modelo de entrevista se incluye en el ANEXO C

Una vez obtenidos los resultados de las entrevistas, los cuales se incluyen en el ANEXO D, se obtuvieron algunas sugerencias y errores con respecto a los mensajes informativos, notificaciones recibidas y la interfaz gráfica de la aplicación Android. Los errores y sugerencias que ocurrieron al momento de realizar las pruebas se presentan en la Tabla 3.2 y Tabla 3.3, respectivamente.

Tabla 3.2. Errores encontrados en las pruebas de funcionamiento

Error	Implementación
Al cerrar sesión, el propietario de la mascota seguía recibiendo la notificación cuando su mascota era encontrada.	Se agregó el método `deleteToken()` de la clase `FirebaseMessaging` en el proceso de cierre de sesión y eliminación de cuenta para eliminar el identificador de la instancia de la aplicación de los servicios de Firebase *Cloud Messaging.*
Al cerrar sesión, los usuarios seguían recibiendo la notificación de mascota desaparecida.	Se agrega el método `unsubscribeFromTopic()` de la clase `FirebaseMessaging` tanto en el proceso de cierre de sesión como al eliminar la cuenta para anular la suscripción del dispositivo al tópico o tema `mascotas-desaparecidas`.

Tabla 3.3. Sugerencias obtenidas en las pruebas de funcionamiento

Sugerencia	Implementación
Incrementar el tiempo con el que se muestran los mensajes flotantes al realizar una determinada acción	Se agregó un mayor tiempo al mensaje `Toast` al definir en su creación el parámetro `Toast.LENGHT_LONG`, el cual establece un tiempo de tres segundos para mostrar los mensajes.
Ajustar el diseño de las pantallas en modo horizontal puesto que la relación de aspecto de los componentes visuales no es el adecuado.	Dado que los diseños de las vistas se realizaron única y exclusivamente para modo vertical de pantalla, se procedió a declarar la propiedad `android:screenOrientation` con el valor `portrait` en cada una de las actividades del archivo `AndroidManisfest.xml` para establecer la orientación vertical en la que se mostrará las actividades en el dispositivo.
Sería conveniente agregar un indicador de progreso para los procesos de registro e inicio de sesión.	Se procede a instanciar un objeto de la clase `AlertDialog` e invocar los métodos `show()` y `dismiss()` para mostrar al usuario un mensaje de espera y ocultarlo cuando se envié y reciba una respuesta por parte del servicio web al registrar e iniciar sesión.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Al finalizar el presente Trabajo de Titulación se dispone de una herramienta que permite ayudar con la búsqueda de mascotas mediante el uso de etiquetas NFC.
- En la fase de implementación del prototipo se procedió a levantar un ambiente de desarrollo e instalar las herramientas necesarias para el desarrollo de software entre ellas: XAMMP, Visual Studio Code, GitHub Desktop y Android Studio para codificar los diferentes componentes de la arquitectura del prototipo los cuales son: base de datos, servicio web, aplicación Android y etiquetas NFC.
- Para la lectura y escritura de etiquetas NFC se empleó la librería `android.nfc`, con la cual se comprobó si el teléfono inteligente es compatible con la tecnología NFC, además, permitió construir el mensaje NDEF para su posterior almacenamiento en la etiqueta.
- El control de versiones y almacenamiento del código fuente generado se realizó a través de la herramienta GitHub Desktop, la cual mediante su interfaz gráfica permite emitir algunos comandos GIT como: *merge*, *clone*, *push*, *create*, *commit*, entre otros, sin la necesidad de acceder a una interfaz de línea de comandos.
- Las API habilitadas de los servicios de Google fueron: *Maps* SDK *for* Android, *Places* API y Firebase *Cloud Messaging*. Estas API se integraron como parte del desarrollo de la aplicación Android para mostrar mapas interactivos al usuario con marcadores de posición que representan el lugar en donde fue encontrada o desaparecida una mascota, autocompletar el nombre de un lugar a medida que usuario lo digita y enviar notificaciones dispositivo a dispositivo o mediante la suscripción por temas.
- La arquitectura empleada en el prototipo se basa en el modelo cliente – servidor por lo que para poder establecer una comunicación entre el servicio web y la aplicación Android a través del protocolo HTTP, se agregaron en el archivo `AndroidManifest.xml` los permisos de `INTERNET` y `ACCESS_NETWORK_STATE`, así mismo, se agregó el permiso `ACCESS_FINE_LOCATION` para permitir que el servicio de GPS del teléfono inteligente proporcione información acerca de la ubicación actual del dispositivo.

- El tipo de etiqueta NFC utilizada en las pruebas de funcionamiento del prototipo corresponde al tipo 2 del *Forum* NFC y usa el chip NTAG213. Además, en el collar de la mascota se incorporó un llavero de plástico debido a que la etiqueta NFC NTAG213 no es aplicable a superficies metálicas.

4.2 RECOMENDACIONES

- Se recomienda implementar una funcionalidad que permita al propietario de la mascota subir una imagen de su mascota a la aplicación.
- Se podría considerar continuar el desarrollo de la aplicación para el sistema operativo IOS.
- Se podría mejorar el aspecto visual de la aplicación incluyendo efectos y animaciones personalizadas a los diferentes componentes visuales de la interfaz gráfica proporcionadas por Material *Design*.
- Se recomienda añadir una opción en la lista de mascota desaparecidas que permita ver la ubicación en donde fue vista por última vez cada una de ellas.
- Se podría añadir una funcionalidad que guarde las notificaciones recibidas tanto de una mascota encontrada al igual que las reportadas como desaparecidas.
- Se recomienda incluir una validación que permita identificar si el número de celular y correo electrónico del usuario existe y se encuentra operativo.
- Se recomienda consultar en el *datasheet* de una etiqueta NFC, además de sus especificaciones técnicas, el tipo de superficie sea metal, madera, plástico, entre otras, sobre la cual la etiqueta NFC es aplicable.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] D. Rojas, «Repositorio Digital UCE,» Diciembre 2013. [En línea]. Available: http://www.dspace.uce.edu.ec/bitstream/25000/4156/3/T-UCE-0013-Ab-136.pdf. [Último acceso: 10 Marzo 2022].

[2] V. a. K. O. B. Ozdenizci, Professional NFC Application Development for Android, Primera ed., United Kingdom: Wrox Press, 2013.

[3] «atlasRFIDstore,» [En línea]. Available: https://www.atlasrfidstore.com/rfid-insider/rfid-vs-nfc/. [Último acceso: 25 enero 2022].

[4] P. Lathiya y W. Jing, Near-Field Communications (NFC) for Wireless Power Transfer (WPT): An Overview, 2021.

[5] «NFC Forum,» [En línea]. Available: https://nfc-forum.org/our-work/specification-releases/specifications/. [Último acceso: 19 enero 2022].

[6] «DAYDAYW,» [En línea]. Available: https://www.daydayw.com/index.php?main_page=product_info&products_id=331119. [Último acceso: 19 enero 2022].

[7] V. Reyes y S. Andrés, «Repositorio Digital EPN,» 27 Noviembre 2014. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/8844. [Último acceso: 17 Enero 2022].

[8] «GoToTags learn,» 20 agosto 2021. [En línea]. Available: https://learn.gototags.com/nfc/ndef. [Último acceso: 19 enero 2022].

[9] T. Igoe, D. Coleman y B. Jepson, Beginning NFC Near Field Communication with Arduino, Android, and PhoneGap, 2014.

[10] K. Rohde, Z. Kis y F. Beaufort, «W3C Community Group Draft Report,» 18 enero 2022. [En línea]. Available: https://w3c.github.io/web-nfc/#ndef-record-types. [Último acceso: 01 enero 2022].

[11] P. Uttarwar, R. Tidke, D. Dandwate y U. Tupe, «ResearchGate,» 14 septiembre 2021. [En línea]. Available: https://www.researchgate.net/publication/354576500_A_Literature_Review_on_Android_-A_Mobile_Operating_system. [Último acceso: 26 enero 2022].

[12] N. Ekanayake, «ResearchGate,» mayo 2018. [En línea]. Available: https://www.researchgate.net/publication/325257105_Android_Operating_System. [Último acceso: 25 enero 2022].

[13] R. Meier y I. Lake, Professional Android, Indianapolis: John Wiley & Sons, Inc., 2018.

[14] «Developers Android,» 7 mayo 2020. [En línea]. Available: https://developer.android.com/guide/platform?hl=es-419. [Último acceso: 31 enero 2022].

[15] «ReserchGate,» agosto 2019. [En línea]. Available: https://www.researchgate.net/publication/335419384_API_Features_Individualizing_of_Web_Services_REST_and_SOAP. [Último acceso: 31 enero 2022].

[16] «IBM,» [En línea]. Available: https://www.ibm.com/docs/en/odm/8.0.1?topic=api-http-methods. [Último acceso: 31 enero 2022].

[17] «IBM,» [En línea]. Available: https://www.ibm.com/docs/sr/zosconnect/3.0?topic=reference-http-response-code. [Último acceso: 31 enero 2022].

[18] «Universidad de Alicante,» [En línea]. Available: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html. [Último acceso: 30 enero 2022].

[19] «RapidApi,» [En línea]. Available: API vs Web Service: What's the Difference?. [Último acceso: 30 enero 2022].

[20] «Google Maps Plataform,» [En línea]. Available: https://developers.google.com/maps/documentation/places/web-service/overview. [Último acceso: 30 enero 2022].

[21] «Maplink,» [En línea]. Available: ¿Sabes qué es Google APIs? Descúbrelo ahora mismo. [Último acceso: 30 enero 2022].

[22] «ProgrammableWeb,» [En línea]. Available: https://www.programmableweb.com/api/google-firebase-cloud-messaging. [Último acceso: 30 enero 2022].

[23] «Firebase,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/send-message#rest. [Último acceso: 30 enero 2022].

[24] «Oracle,» [En línea]. Available: https://www.oracle.com/mx/database/what-is-database/. [Último acceso: 30 enero 2022].

[25] «Oracle Cloud Infrastructure (OCI),» [En línea]. Available: https://www.oracle.com/database/what-is-a-relational-database/. [Último acceso: 30 enero 2022].

[26] «XAMPP TUTORIAL,» [En línea]. Available: https://www.javatpoint.com/xampp. [Último acceso: 29 enero 2022].

[27] «Educative,» [En línea]. Available: https://www.educative.io/edpresso/what-is-git. [Último acceso: 29 enero 2022].

[28] «WebEmpresa,» [En línea]. Available: https://www.webempresa.com/hosting/que-es-github.html. [Último acceso: 29 enero 2022].

[29] «Digite,» [En línea]. Available: https://www.digite.com/kanban/what-is-kanban/. [Último acceso: 29 enero 2021].

[30] «Wizapet,» 22 Abril 2019. [En línea]. Available: https://www.wizapet.com/. [Último acceso: 22 Febrero 2022].

[31] Tractive, «Tractive GPS para perros y gatos,» 02 Enero 2012. [En línea]. Available: https://tractive.com/es/. [Último acceso: 22 Febrero 2022].

[32] «Welppy,» 11 Octubre 2019. [En línea]. Available: https://www.welppy.com/. [Último acceso: 22 Febrero 2022].

[33] J. Jonathan, «Repositorio Digital EPN,» 30 noviembre 2021. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/21942. [Último acceso: 3 febrero 2022].

[34] «Developers Android,» 30 Diciembre 2019. [En línea]. Available: https://developer.android.com/guide/topics/connectivity/nfc/advanced-nfc#java. [Último acceso: 14 Junio 2022].

6. ANEXOS

ANEXO A. Modelo de entrevista aplicado a propietarios de mascota

ANEXO B. *Sketches* realizados para las interfaces

ANEXO C. Modelo de entrevista para validar el funcionamiento del prototipo

ANEXO D. Resultados de la entrevista para validar el funcionamiento del prototipo

ANEXO E. Código fuente del prototipo desarrollado

ANEXO F. Manual de usuario del prototipo

ANEXO G. Manual de instalación del prototipo

ANEXO A

En este anexo se muestra el modelo de entrevista aplicado a propietarios de mascotas para la adquisición de requerimientos.

1. **¿A través de cuál de los siguientes mecanismos desea que se le informe cuando su mascota haya sido encontrada?**
   - Mensaje vía SMS
   - Notificación emergente en el teléfono inteligente
   - Llamada telefónica
   - Mensaje vía correo electrónico
   - Otro:_____

2. **¿Al informarse de que su mascota ha sido encontrada, cuál de las siguientes opciones desea conocer sobre el paradero de su mascota y la persona quién encontró a la mascota?**
   - Nombre de la persona
   - Correo electrónico de la persona
   - Numero de celular de la persona
   - Ubicación de la mascota
   - Otro:_____

3. **¿Desearía la opción de ponerse en contacto vía llamada telefónica con la persona quien encontró a su mascota?**
   - Si
   - No

4. **¿De las siguientes opciones seleccione que información de su mascota desea guardar en el sistema?**
   - Nombre
   - Raza
   - Género
   - Descripción
   - Otro:_____

5. **¿A través del mecanismo de notificaciones emergentes, cuál de las siguientes opciones desea dar a conocer a los usuarios del sistema cuando su mascota haya sido desaparecida?**
    - ☐ Nombre de la mascota
    - ☐ Raza de la mascota
    - ☐ Género de la mascota
    - ☐ Ubicación del lugar donde fue vista por última vez la mascota
    - ☐ Nombre del propietario de la mascota
    - ☐ Numero de celular del propietario de la mascota
    - ☐ Otro: _____

ANEXO B

En este anexo se incluye todos los *sketches* realizados para las interfaces de usuario.

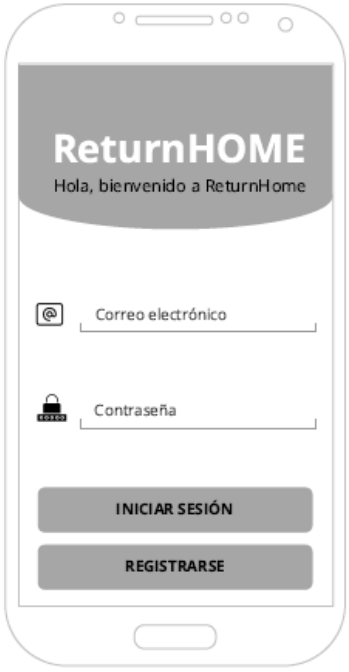

Figura B.1. *Sketch* de la pantalla para iniciar sesión

Figura B.2. *Sketch* de la pantalla para registrar un nuevo usuario

Figura B.3. *Sketch* de la pantalla Home con la lista de las mascotas registradas por el usuario

Figura B.4. *Sketch* de la pantalla con la lista de las mascotas desaparecidas

Figura B.5. *Sketch* del menú lateral principal

Figura B.6. *Sketch* de la vista para agregar o actualizar una mascota

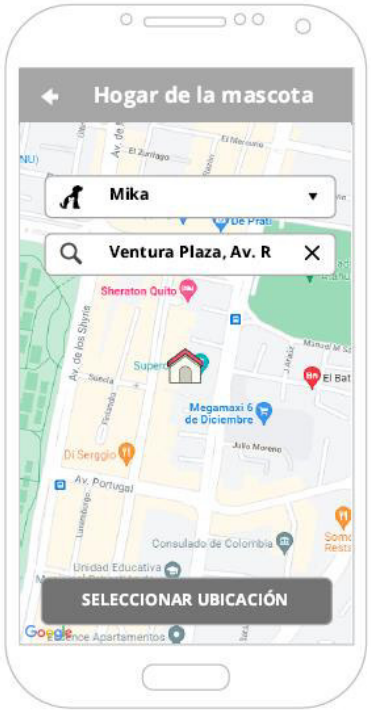

Figura B.7. *Sketch* de la pantalla para seleccionar la ubicación del hogar de una mascota

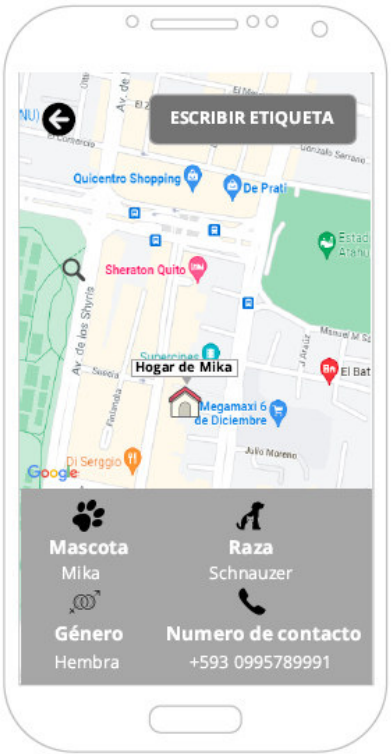

Figura B.8. *Sketch* de la pantalla con la información de una mascota a escribir en una etiqueta NFC

Figura B.9. *Sketch* de la pantalla para enviar datos hacia la etiqueta NFC

Figura B.10. *Sketch* de la pantalla para recibir datos desde la etiqueta NFC

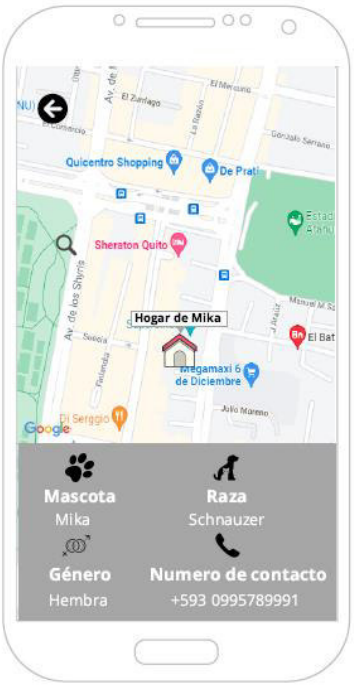

Figura B.11. *Sketch* de la pantalla para mostrar la información de una mascota almacenada en la etiqueta NFC

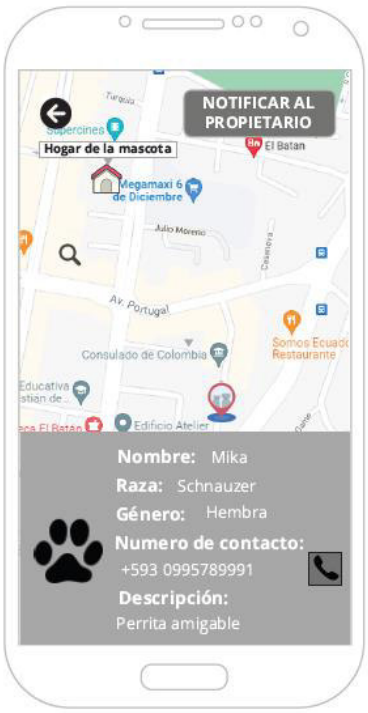

Figura B.12. *Sketch* de la pantalla para mostrar información de una mascota encontrada y notificar a su propietario

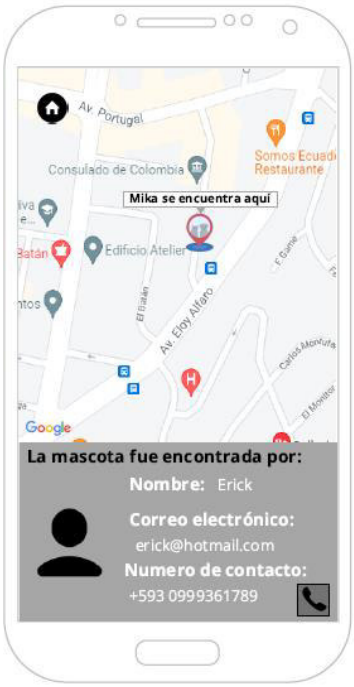

Figura B.13. *Sketch* de la pantalla para mostrar la información del usuario quien encontró una mascota

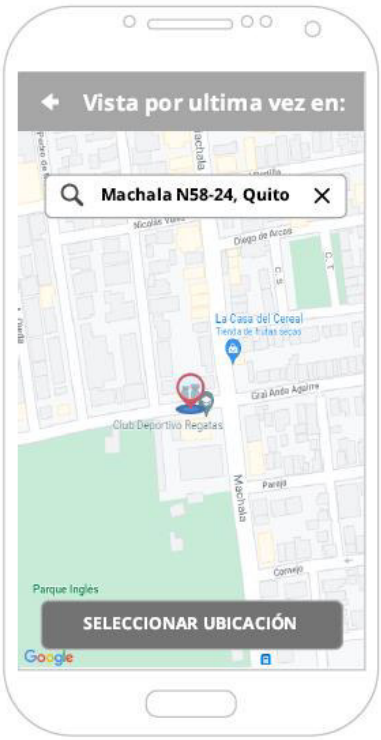

Figura B.14. *Sketch* de la pantalla para seleccionar la ubicación donde fue vista por última vez una mascota

Figura B.15. *Sketch* de la pantalla para mostrar información de una mascota reportada como desaparecida

Figura B.16. *Sketch* de la pantalla para seleccionar una opción de ajustes de la cuenta

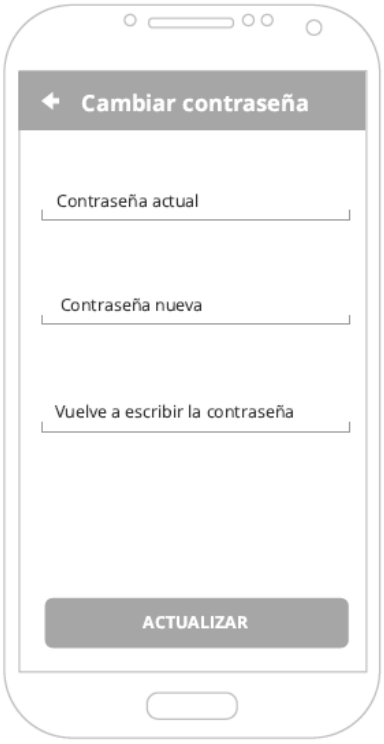

Figura B.17. *Sketch* de la pantalla para actualizar la contraseña

Figura B.18. *Sketch* de la pantalla para actualizar la información del usuario

ANEXO C

En este anexo se presenta el modelo de entrevista para validar el funcionamiento del prototipo.

1. **¿Mantuvo algún inconveniente al instalar y ejecutar la aplicación en su dispositivo móvil?**

- Si
- No

2. **¿Presentó alguna complicación al iniciar sesión con sus datos?**

- Si
- No

3. **¿Qué tan satisfecho está con el proceso de registro de nuevo usuario?**

- Nada satisfecho
- Poco satisfecho
- Moderadamente satisfecho
- Muy satisfecho

4. **¿Al momento de registrar una mascota, presentó algún tipo de inconveniente o confusión en los campos de ingreso?**

- Si
- No

5. **¿Cómo calificaría el proceso de editar la información de su mascota?**

- Proceso muy dificultoso
- Proceso difícil
- Proceso normar
- Proceso fácil

6. **¿Pudo visualizar con facilidad y de manera intuitiva la opción de eliminar una mascota?**

- Si
- No

7. **¿Qué tan complicado le resultó el proceso de escribir la información de su mascota en una etiqueta NFC?**

- ○ Bastante complicado
- ○ Moderadamente complicado
- ○ Casi nada complicado
- ○ Sencillo de utilizar

8. **Al momento de leer una etiqueta NFC, ¿la información se presentó sin ningún contratiempo?**

- ○ Si
- ○ No

9. **Durante el proceso de notificar una mascota como encontrada, ¿Qué tan complicado fueron los pasos a seguir?**

- ○ Los pasos fueron complicados de seguir
- ○ Los pasos fueron medianamente complicados
- ○ Los pasos no fueron complicados
- ○ Los pasos fueron sencillos de seguir

10. **Una vez en la aplicación ¿Pudo acceder con facilidad al módulo de actualizar su contraseña?**

- ○ Si
- ○ No

11. **¿Pudo actualizar sus datos personales dentro de la aplicación sin ningún tipo de problema?**

- ○ Si
- ○ No

12. **Cuando intentó eliminar su cuenta, ¿encontró algún tipo de dificultad?**

- ○ Si
- ○ No

13. **Indique la facilidad que tuvo al reportar una mascota como desaparecida**

- ○ Complicada de reportar

- Moderadamente complicada de reportar
- Casi nada complicada de reportar
- Sencilla de reportar

14. ¿Qué tanto esfuerzo le llevó el utilizar la función reportar una mascota como encontrada?

- Mucho esfuerzo
- Poco esfuerzo
- Casi nada de esfuerzo
- Ningún esfuerzo

15. Califique su experiencia general al utilizar la aplicación

- Mala experiencia, la aplicación tuvo diversos fallos.
- Experiencia regular, la aplicación cumple con lo básico
- Buena experiencia, la aplicación es muy intuitiva
- Experiencia sobresaliente, las funciones de la aplicación exceden las expectativas

16. Si encontró algún tipo de error en la aplicación o tiene una sugerencia. Por favor coméntela.

Tu respuesta

ANEXO D

En este anexo se presentan los resultados de la entrevista para validar el funcionamiento del prototipo.

Figura D.1. Respuestas de la pregunta 1

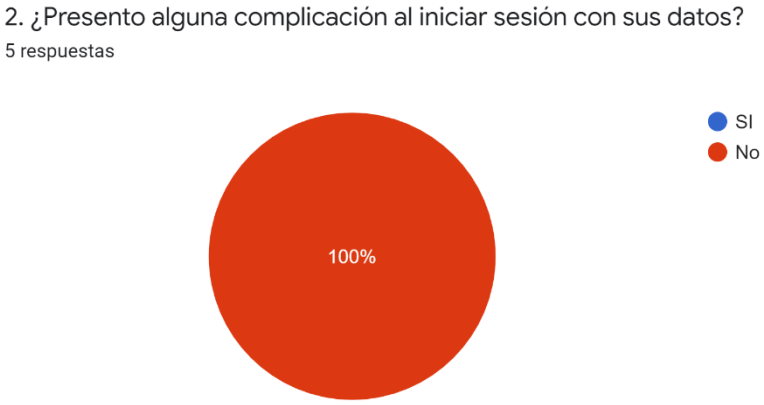

Figura D.2. Respuestas de la pregunta 2

Figura D.3. Respuestas de la pregunta 3

4. ¿Al momento de registrar una mascota, presento algún tipo de inconveniente o confusión en los campos de ingreso?

5 respuestas

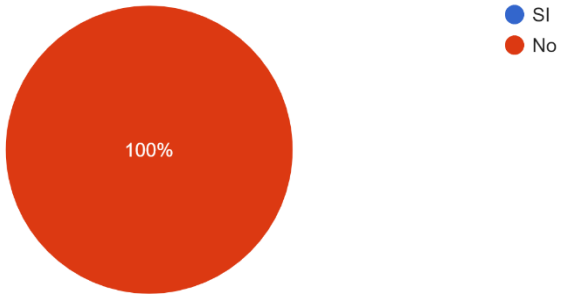

Figura D.4. Respuestas de la pregunta 4

5. ¿Cómo calificaría el proceso de editar la información de su mascota?

5 respuestas

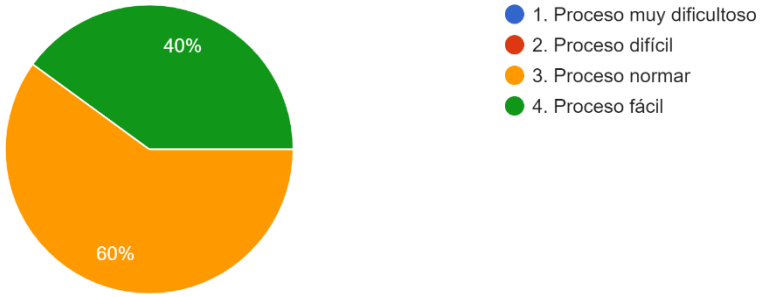

Figura D.5. Respuestas de la pregunta 5

6. ¿Pudo visualizar con facilidad y de manera intuitiva la opción de eliminar una mascota?

5 respuestas

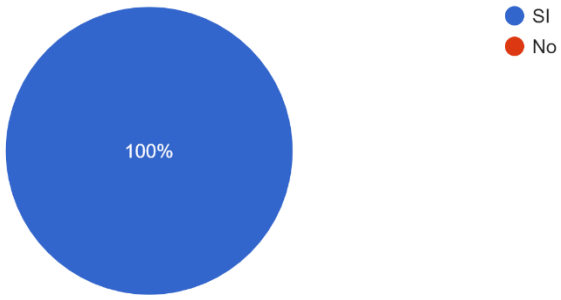

Figura D.6. Respuestas de la pregunta 6

7. ¿Qué tal complicado le resulto el proceso de escribir la información de su mascota en una etiqueta NFC?

5 respuestas

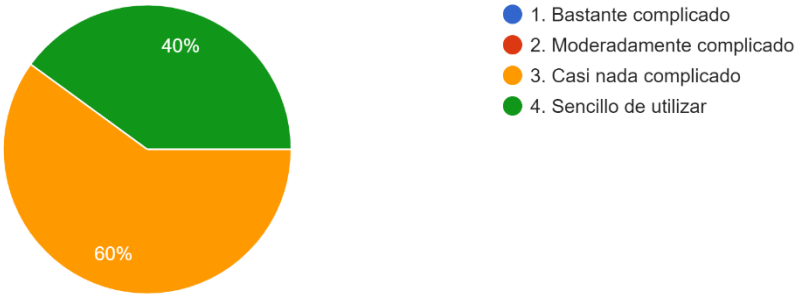

Figura D.7. Respuestas de la pregunta 7

8. Al momento de leer una etiqueta NFC, ¿la información se presentó sin ningún contratiempo?

5 respuestas

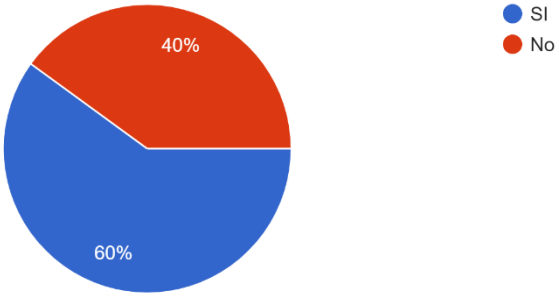

Figura D.8. Respuestas de la pregunta 8

9. Durante el proceso de notificar una mascota como encontrada, ¿Qué tan complicado fueron los pasos a seguir?

5 respuestas

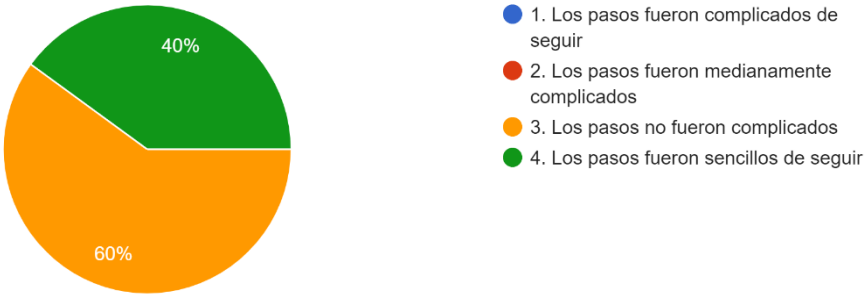

Figura D.9. Respuestas de la pregunta 9

10. Una vez en la aplicación ¿Pudo acceder con facilidad al módulo de actualizar su contraseña?
5 respuestas

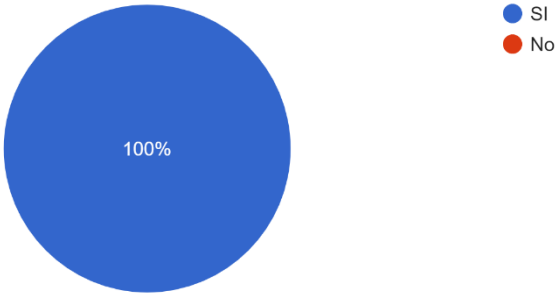

Figura D.10. Respuestas de la pregunta 10

11. ¿Pudo actualizar sus datos personales dentro de la aplicación sin ningún tipo de problema?
5 respuestas

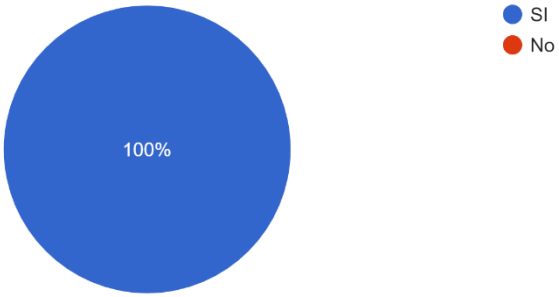

Figura D.11. Respuestas de la pregunta 11

12. Cuando intentó eliminar su cuenta, ¿encontró algún tipo de dificultad?
5 respuestas

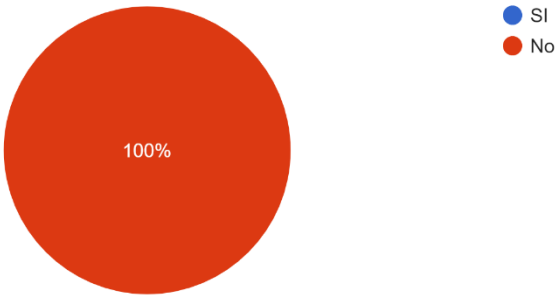

Figura D.12. Respuestas de la pregunta 12

13. Indique la facilidad que tuvo al reportar una mascota como desaparecida

5 respuestas

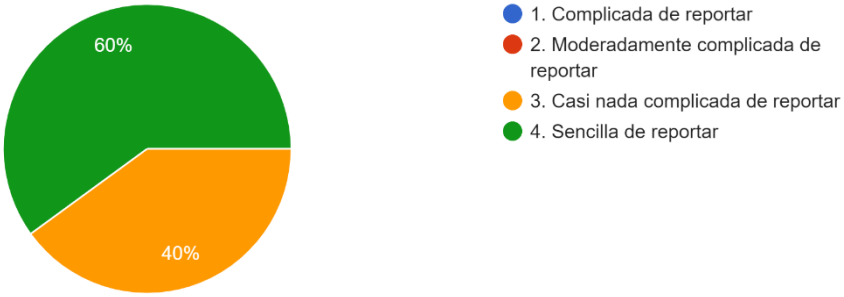

Figura D.13. Respuestas de la pregunta 13

14. ¿Qué tanto esfuerzo le llevo el utilizar la función al reportar una mascota como encontrada?

5 respuestas

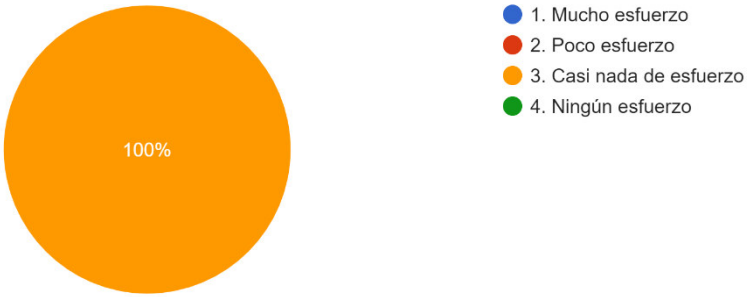

Figura D.14. Respuestas de la pregunta 14

15. Califique su experiencia general al utilizar la aplicación

5 respuestas

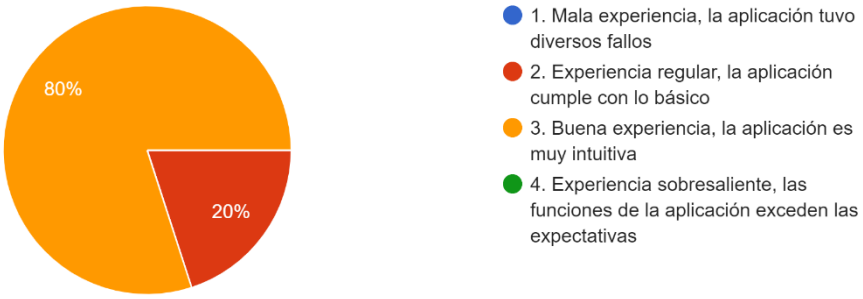

Figura D.15. Respuestas de la pregunta 15

ANEXO E

Este anexo se adjunta de forma digital e incluye el código fuente del prototipo desarrollado.

ANEXO F

Este anexo se adjunta de forma digital e incluye el manual de usuario del prototipo.

ANEXO G

Este anexo se adjunta de forma digital e incluye el manual de instalación del prototipo.

ORDEN DE EMPASTADO