SISTEMA MULTIPLATAFORMA PARA REPORTAR Y RESGUARDAR MASCOTAS DESAPARECIDAS

MULTI-PLATFORM SYSTEM FOR REPORTING AND SAFEGUARD MISSING PETS

Irving Mejía Barrera
Universidad Autónoma Metropolitana, México
irvingslife@gmail.com

Ismael Ariel Robles Martínez
Universidad Autónoma Metropolitana, México
iarobles271@gmail.com

Sergio Zepeda Hernández
Universidad Autónoma Metropolitana, México
jzepeda@cua.uam.mx

Recepción: 29/noviembre/2022 **Aceptación:** 22/diciembre/2022

Resumen

En este artículo presentamos un sistema multiplataforma como aplicación móvil y web, que permite reportar mascotas extraviadas por medio del envío de datos, fotos y última ubicación vista. A su vez, el sistema permite reportar datos de avistamientos y mascotas encontradas por los usuarios, con el fin de que el propietario pueda visualizar esta información y pueda recuperar a su mascota. El sistema tiene el objetivo de reducir el índice de mascotas extraviadas que no logran regresar a sus hogares y pueden causar un impacto negativo tanto en humanos como en la fauna y el entorno donde tratan de sobrevivir.

Palabras Clave: Aplicaciones móviles, desarrollo de sistemas, sistemas de información.

Abstract

In this article we present a multi-platform system mobile and web application, that allows to report missing pets by sending data, photos and last known location. The system in turn allows users to report sightings and found pets, so that the owner can view this information and recover his pet. The main goal of this system is to reduce

the rate of lost pets that do not achieve to back home and may cause a negative impact on humans as well as on the fauna and the environment where they try to survive.

Keywords: *Mobile applications, software development, information systems.*

1. Introducción

El problema del extravío de mascotas se contempla en la legislación de la Ciudad de México, con la promulgación de la Ley de Protección Animal [ALDF, 2017]: la cual establece que todos los ciudadanos deben registrar a sus mascotas en el Registro Único de Animales de Compañía [RUAC, 2021]. A pesar de esto, hasta el término de 2018, estaban registradas menos del 1% del número estimado de mascotas que habitan en la Ciudad de México [Martínez, 2019]; con lo que se puede observar que la solución gubernamental no ha sido de interés para la población.

El uso de aplicaciones móviles se ha planteado en años recientes como una forma de dar aviso y localizar mascotas extraviadas. Por ejemplo, en [Aqraldo, 2021], se plantea una aplicación móvil que hace uso de un dispositivo de Posicionamiento Global para localizar mascotas desaparecidas; sin embargo, el funcionamiento de esta aplicación está limitado a regiones de habla inglesa. De manera similar, en [Tangsripairoj, 2018], también se propone una aplicación móvil que cuenta con un foro para dar aviso de mascotas extraviadas, pero este sistema sólo esta disponible en Tailandia.

Existen aplicaciones comerciales que se enfocan en este problema y que están disponibles en México, como lo son: Wizapet [Wizapet, 2021], la cual es una aplicación móvil que permite emitir alertas y avisos de mascotas perdidas; PeTrace [PeTrace, 2021], que además permite avisar sobre mascotas en adopción y por último; Huellitas Perdidas [Huellitas Perdidas, 2021], el cual es un sistema web (no es una aplicación móvil), que tiene un catálogo de mascotas desaparecidas. En este contexto, y con el fin de facilitar la localización de mascotas desparecidas, se desarrolló una aplicación híbrida multiplataforma, la cual es una aplicación que se puede exportar para ser una aplicación móvil nativa y una aplicación web [Minq, 2017]. La aplicación “BuscandoPet” que se presenta en este trabajo, permite

registrar y difundir información sobre mascotas extraviadas para contribuir a ubicarlas rápidamente, y así evitar que corran peligro o pongan en peligro a otros animales o personas.

2. Métodos

El desarrollo del proyecto se basó en la metodología Scrum, la cual permite generar prototipos funcionales en poco tiempo y va agregando mejoras incrementales a dicho prototipo para obtener la aplicación final deseada [Ockerman, 2021]. En Scrum, el desarrollo de una aplicación se hace mediante un enfoque iterativo, esto es: se plantean una serie de requerimientos para la aplicación y estos se desarrollan dentro de un intervalo de tiempo conocido como sprint, que puede durar hasta cuatro semanas. Concluido el sprint, se actualizan los requerimientos con el fin de mejorar la aplicación y se realiza de nuevo un sprint. Este proceso se repite varias veces hasta alcanzar un proyecto que cumpla la funcionalidad deseada. De forma paralela se utilizó el método Kanban [Anderson, 2010], el cual se enfoca en organizar las tareas pendientes, las cuales se escriben en tarjetas (*Kanban*, en japonés) y se van desplazando en tres grupos: tareas por hacer, tareas en progreso y tareas realizadas. El tiempo que toma cada tarea debe ser el menor posible, para evitar que haya bloqueos de otras tareas; en caso de haberlos, se debe identificar el motivo por el cual una tarea haya permanecido más tiempo del esperado. Las metodologías antes mencionadas se usaron en cuatro etapas: requerimientos, diseño de la base de datos, diseño de microservicios e implementación; las cuales se describen a continuación.

Requerimientos de la aplicación

Los requerimientos para la aplicación, fueron elaborados en función de una extensa revisión de las limitaciones de aplicaciones similares, las cuales se mencionaron en el estado del arte. Los principales requerimientos del sistema y que se implementar en la aplicación final, son los siguientes:

- Registro de usuarios: permite dar de alta la cuenta de un usuario del sistema. Las credenciales de la cuenta se usan para dar acceso al sistema al usuario.

- Reportar mascotas desaparecidas: permite registrar información general de la mascota (nombre, especie, género, vacunas, descripción), publicar fotos y se tiene la posibilidad de indicar en un mapa, la última localización conocida de la mascota extraviada. Se tiene también la opción, si así lo desea el usuario, de hacer pública su información de contacto.
- Reportar mascotas encontradas: que permite registrar información de una mascota que se ha encontrado y se tiene en resguardo (no se conoce su dueño). Se tiene la posibilidad de indicar en un mapa, la región en la que fue encontrada la mascota, así como publicar fotos que permitan a su posible dueño identificarla.
- Búsqueda de mascotas: permite que un usuario pueda buscar en el sistema una mascota que se haya reportado como extraviada o encontrada, dentro de una región geográfica.
- Sistema de mensajería: permite que usuarios en el sistema puedan tener un primer contacto, afin de que esto les permita interactuar protegiendo la privacidad de sus datos. Posteriormente, si los usuarios por mutuo acuerdo así lo deciden, pueden revelar sus datos personales entre sí.
- Multiplataforma: la aplicación funciona como una aplicación nativa para dispositivos móviles en las plataformas comerciales principales (Android y iOS) y tambíen está disponible como aplicación web con diseño adaptativo [Frain, 2020], lo que le permite adaptarse a resoluciones de dispositivos diferentes.

La figura 1, muestra el diagrama general de casos de uso con los servicios ofrecidos por el sistema multiplataforma, se puede observar que para usar la aplicación el usuario se debe registrar con su correo electrónico y contraseña (credenciales). Posteriormente, para acceder debe iniciar sesión con sus credenciales, de esta manera al ingresar al sistema, el usuario tiene la posibilidad de generar un reporte de mascota extraviada o encontrada, y con ello puede registrar datos y fotos de la mascota. También puede buscar las mascotas que otros usuarios han reportado como extraviadas o encontradas, lo que le mostrará una lista con dichos resultados.

Finalmente, el usuario puede enviar y recibir mensajes de otros usuarios en el sistema.

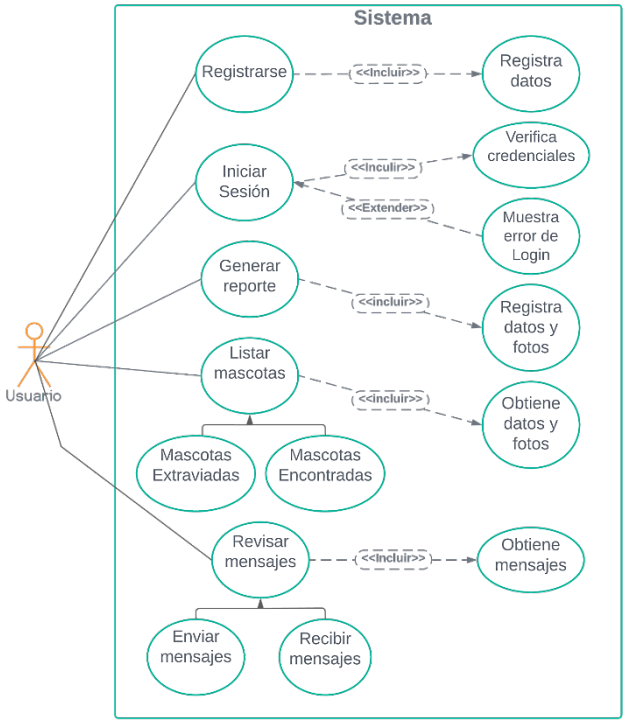


Figura 1 Diagrama general de casos de uso de la aplicación.

Diseño de la base de datos

Para almacenar la información se diseñó una base de datos relacional, se observa que se tienen seis tablas con los siguientes nombres: usuarios, mensajes, mascotas, mascotas_extraviadas, mascotas_encontradas, fotos_mascotas (ver figura 2. De éestas, la más relevante para el sistema es la tabla mascotas, que guarda información general de una mascota extraviada o encontrada.

La tabla mascotas tiene una columna denominada “tipo”, cuyos valores pueden ser: “extraviada” o “encontrada”. La columna “estatus” puede tener los valores: “activo” (para reportes activos), “cancelado” (para reportes cancelados) o “reunida” (cuando el usuario encontró o entregó a la mascota). En cuanto a la columna género, los valores esperados son “macho”, “hembra” o “sin identificar”. La información específica para una mascota encontrada o extraviada, se guarda en las tablas mascotas_encontradas y mascotas_extraviadas, respectivamente.

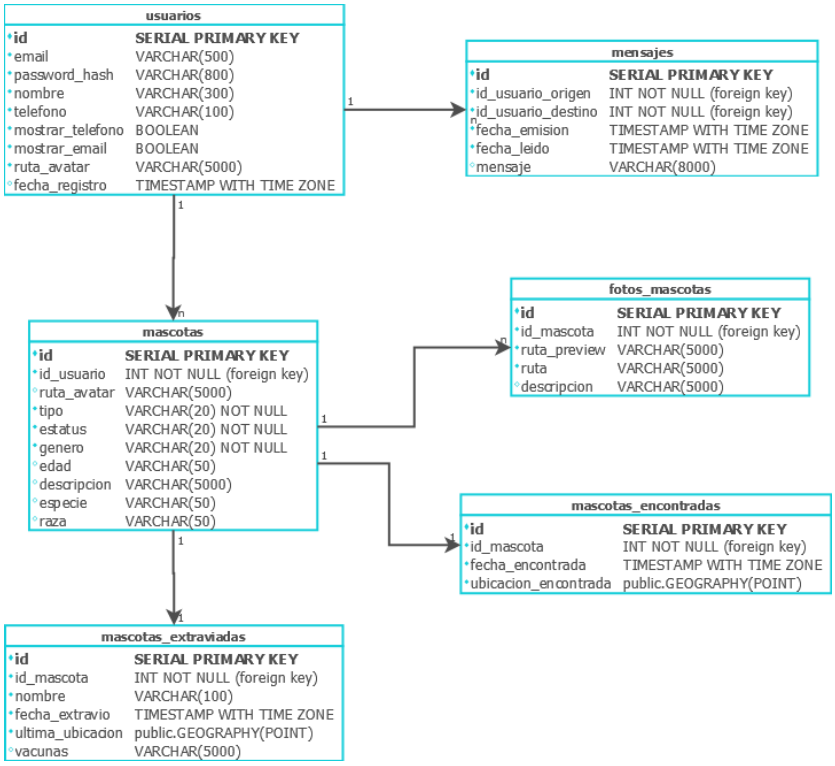


Figura 2 Base de datos.

La tabla fotos_mascotas, se usa para guardar la información de imágenes, sin embargo, el almacenamiento de los archivos de las imágenes no se hace directamente en la base de datos, sino en una carpeta interna del servidor. La tabla sólo almacena en la columna “ruta”, el nombre del archivo y la carpeta del sistema operativo donde se guarda la imagen. De manera similar, la columna “ruta_preview”, almacena la ruta de la misma imagen que se almacena en la columna “ruta”, pero en menor resolución para que se muestre como una miniatura en las interfaces.

Diseño de microservicios

El servidor se diseñó como una Interfaz de Programación de Aplicaciones (API), mediante Transferencia de Estado Representacional (REST, por sus siglas en inglés) [Fielding, 2000]. Este mecanismo consiste en que mediante un Identificador de Recursos Uniforme (URI), se pueden hacer operaciones en el servidor que permiten obtener, crear, actualizar y borrar datos del servidor. En particular, es común usar REST sobre el Protocolo de Transferencia de Hipertexto (HTTP)

[Gourley, 2002], en conjunto con el formato JSON (JavaScript Object Notation) [Bassett, 2015], para representar la información que responde el servidor. Por ejemplo, si se hace una petición HTTP al servidor solicitando la URL que termina en: /api/mascotas/1; se obtiene como respuesta en formato JSON, la información de la mascota que en la base de datos tiene el id=1, tal como se muestra en la columna "Mascota" de la tabla 1.

Tabla 1 Ejemplo de formato de objetos JSON usados por el sistema.

Mascota	**Mascota Extraviada**

{
"id": 1,
"idUsuario": 1,
"avatar": "577af53c-4314-47db.jpg",
"tipo": "extraviada",
"estatus": "activo",
"genero": "Macho",
"edad": "2 años",
"descripcion": "Color blanco, manchas cafés",
"especie": "Perro",
"raza": "Mestizo"
}
	{
	"id": 1,
	"idMascota": 1,
	"nombre": "Milaneso",
	"fecha": "2022-05-01-T11:23:04.032",
	"vacunas": "Polivalente, rabia",
	"longitud": "-120.4079",
	"latitud": "37.9434",
	"ultimaUbicacion": "SRID=4326; POINT (-120.4079 37.9434)"
	}

De manera similar, si se hace la petición a la URL del servidor que termina en: /api/mascotas/extraviadas/1, se obtiene en formato JSON, la información que se muestra en la columna "MascotaExtraviada" de la tabla 1.

El protocolo HTTP cuenta con varios métodos [Gourley, 2002], pero lo usual en el contexto de REST, es usar los métodos: GET, PUT, POST Y DELETE. De tal forma que la operación que se realiza en el servidor al invocar una URL, depende del método HTTP que se utiliza. Por lo general, la operación que ocurre en el servidor en función del método HTTP, es la siguiente: GET para obtener datos, POST para crear datos, PUT para actualizar datos y DELETE para eliminar datos. El servidor de este sistema se basa en una arquitectura de microservicios [Newman, 2015], que permite desacoplar la implementación del servidor del cliente; lo cual, facilita el desarrollar interfaces de usuario para diferentes plataformas que consumen información del mismo servidor. Los componentes del servidor que conforman el microservicio para listar mascotas se muestran en la figura 3. Observe que, para

consultar información del servidor, el usuario invoca el API REST del servidor (una URL), por medio de un token que se le asigna a su cuenta. Este token se recibe una capa de seguridad que usa Spring Security [Spilca, 2020], la cual valida que el usuario esté registrado y tenga permisos para consultar o modificar la información solicitada, si el token es válido, la petición del usuario se pasa al componente MascotaREST, el cual regresa la información de la respuesta en formato JSON. Para obtener la información solicitada MascotaREST. a su vez invoca a MascotaDAO, el cual es un componente que implementa el patrón de diseño denominado Objeto de Acceso a Datos (DAO, por sus siglas en inglés) para obtener la información de la base de datos [Freeman, 2020].

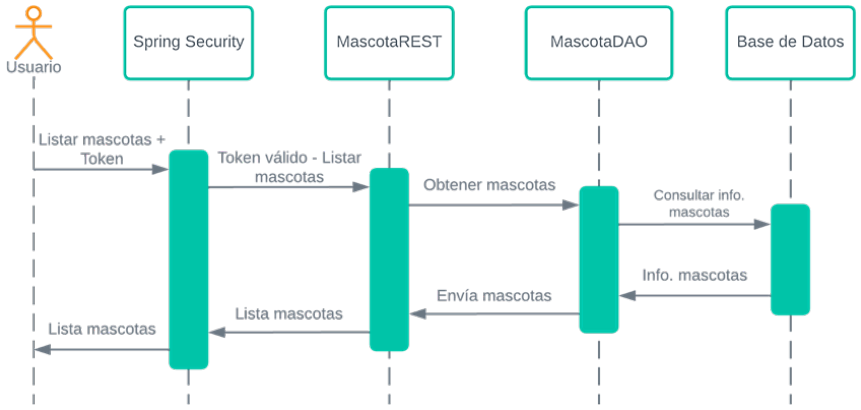


Figura 3 Diagrama de secuencia para listar mascotas.

Implementación

Las tecnologías utilizadas para la implementación del servidor fueron las siguientes: Java como lenguaje de programación; Spring como plataforma para desarrollar el servidor de la aplicación [Johnson, 2016], PostgreSQL para la base de datos y Heroku, como plataforma para almacenar y publicar el servidor en Internet. Del lado del cliente, la aplicación está diseñada con HTML, CSS y JavaScript. Por último, se utilizó el framework Ionic Capacitor que permite exportar el cliente a una aplicación móvil nativa (Android y iOS) y también como una aplicación web [Capacitor, 2021].

La implementación de los DAOs que se mencionaron en la figura 3, se hizo mediante la librería especializada de Spring, conocida como Spring Data JDBC [Schauder,

2022], la cual simplifica el acceso y modificación de sistemas de persistencia, como es el caso de una base de datos relacional. En esta librería, un DAO se crea mediante una interfaz que se extiende (en el sentido de herencia de la programación orientada a objetos) de otra interfaz llamada CrudRepository; la cual contiene métodos predefinidos para hacer consultas, de acuerdo con los parámetros que se le indiquen. Sin embargo, Spring Data JDBC también permite definir métodos propios a través de un mecanismo denominado como: consultas derivadas [Schauder, 2022].

En las figuras 4 y 5 se muestra el código de la interfaz MascotaExtraviadaDAO y de la interfaz MascotaEncontradaDAO, respectivamente. Ambas interfaces se implementaron a través de los mecanismos de consultas derivadas. El método *findByMascotaIn* que se muestra en las figuras 4 y 5, se implementó a través de una consulta derivada, la cual obtiene de la base de datos una lista de mascotas extraviadas o encontradas, dada una lista de enteros el idsMascota, es una lista con los ids de todas las mascotas. Observe que, en este caso, no es necesario especificar la consulta de SQL, sino que Spring JDBC es capaz de inferir la consulta que se quiere realizar, usando unicamente las palabras que aparecen en el nombre del método, en este caso findBy, IdMascota, In.

```
@Repository
public interface MascotaExtraviadaDAO extends CrudRepository<MascotaExtraviada,Integer>{

    public List<MascotaExtraviada> findByIdMascotaIn(Collection<Integer> idsMascota);

    @Query("SELECT * FROM mascotas_extraviadas WHERE id_mascota IN(:idsMascota)")
    public List<MascotaExtraviada> obtenMascotasExtraviadasPorIdMascota(@Param("idsMascota") List<Integer> idsMascota);

    @Modifying
    @Query("DELETE FROM mascotas_extraviadas WHERE id_mascota IN(:idsMascota)")
    @Transactional
    public void borrarPorIdsMascota(@Param("idsMascota") List<Integer> idsMascota);
}
```

Figura 4 Código de la interfaz MascotaExtraviadaDAO.

```
@Repository
public interface MascotaEncontradaDAO extends CrudRepository<MascotaEncontrada,Integer> {

    public List<MascotaEncontrada> findByIdMascotaIn(Collection<Integer> idsMascota);

    @Query("SELECT * FROM mascotas_encontradas WHERE id_mascota IN(:idsMascota)")
    public List<MascotaEncontrada> obtenMascotasEncontradasPorIdMascota(@Param("idsMascota") List<Integer> idsMascota);

    @Modifying
    @Query("DELETE FROM mascotas_encontradas WHERE id_mascota IN(:idsMascota)")
    @Transactional
    public void borrarPorIdsMascota(@Param("idsMascota") List<Integer> idsMascota);
}
```

Figura 5 Código de la interfaz MascotaEncontradaDAO.

Por otra parte, observe que también es posible especificar una consulta de SQL específica, por medio de la instrucción @Query, tal es el caso de los métodos (Figuras 4 y 5, respectivamente):

- ObtenMascotasExtraviadasPorIdMascota.
- *O*tenMascotasEncontradasPorIdMascota.

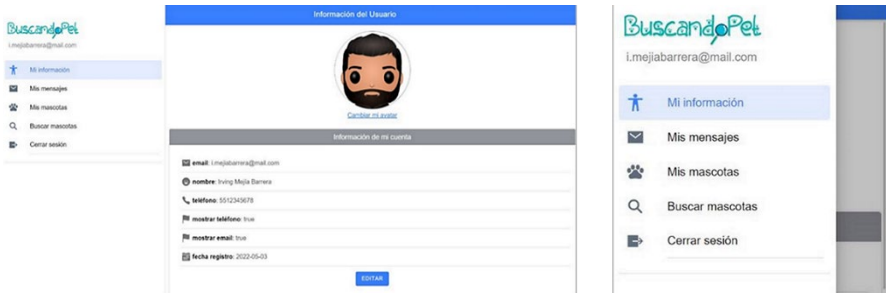
Note que estas consultas específicas permiten obtener los datos de todas aquellas mascotas (extraviadas o encontradas) de un usuario cuyos Id coincidan con el Id especificado. Como se observa en el código, se tiene que especificar qué consulta se va a hacer y, a continuación, se le tiene que dar como parámetro el Id de la mascota. Los métodos mencionados son los que utiliza el DAO del diagrama de secuencias de la figura 3.

Por último, el método borrarPorIdsMascota de ambas interfaces, permite borrar a todas las mascotas (extraviadas o encontradas) que coincidan con un Id de mascota especificado. Al igual que los métodos anteriores, se especifica la consulta que se hará a la base de datos y se le da como parámetro el Id de la mascota.

3. Resultados

Una vez integrados los servicios y métodos, se diseñó la interfaz para escritorio del sistema y se exportó como una aplicación con entorno de ejecución nativo para móvil Android y iOS, por medio de la plataforma Ionic Capacitor. El sistema también está disponible como una aplicación web (escritorio). De esta manera, la aplicación es responsiva en su interfaz de aplicación web y también puede ser obtenida por una app para móvil. La figura 6 muestra ambas interfaces. La figura 6, muestra cómo la interfaz de la aplicación que se presenta en este trabajo denominada *BuscandoPet,* cambia de acuerdo con el tamaño de la pantalla del dispositivo en el que se utilice la aplicación. En la figura 6a, se muestra como se ve la aplicación en una pantalla de alta resolución como es el caso de una laptop; en este caso, el menú aparece completo y fijo del lado izquierdo de la ventana. Cuando el sistema es usado como una aplicación móvil nativa para Android o iOS, el menú y la interfaz reducen su tamaño y aparece el menú de hamburguesa como se muestra en la

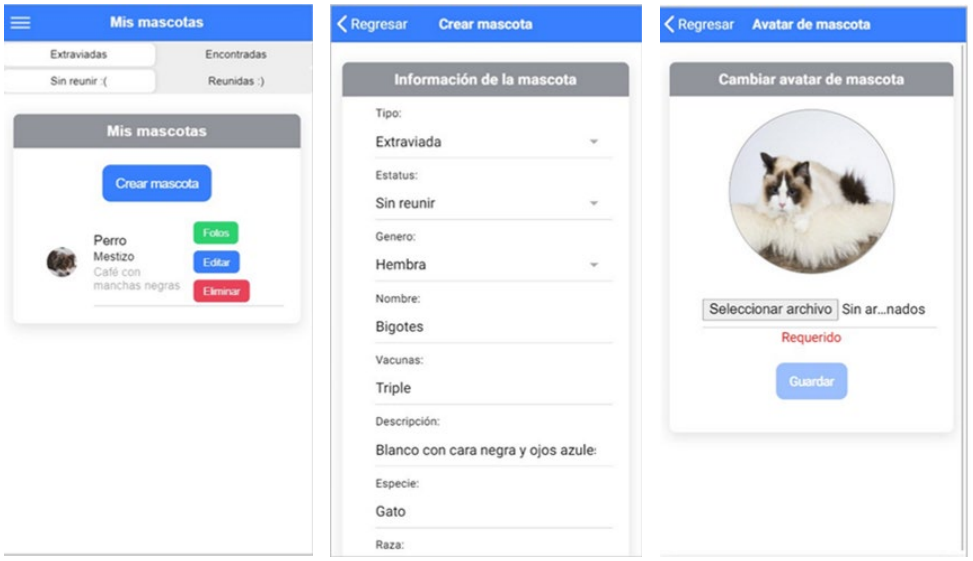
figura 6b, el menú en ambos casos permite al usuario navegar a las secciones principales de la aplicación: Mi información, Mis mensajes, Mis mascotas, Buscar mascotas y Cerrar sesión.



a) Interfaz para escritorio. *b) Interfaz móvil*

Figura 6 Interfaz para escritorio e interfaz móvil.

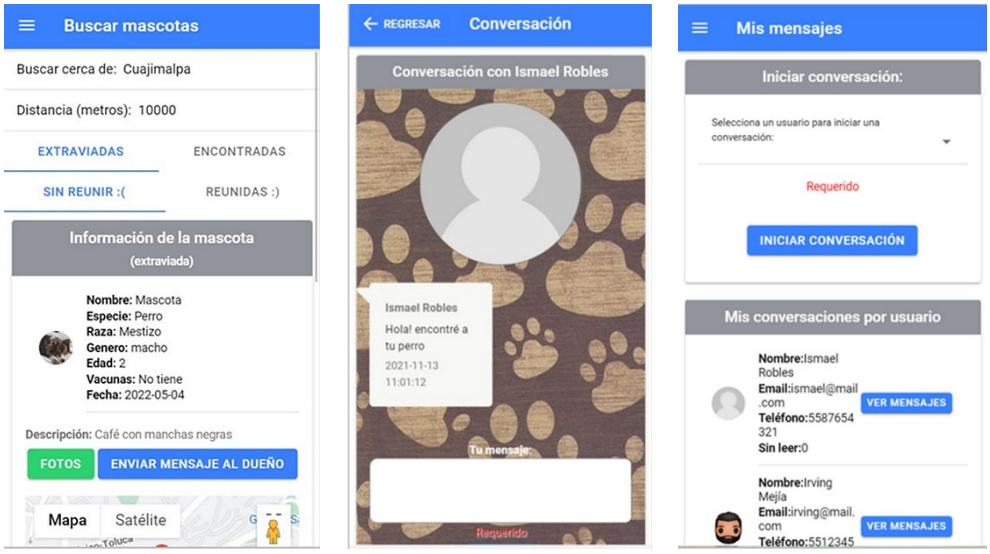
La figura 7a muestra la interfaz de Mis mascotas, en ella el usuario puede ver un listado de las mascotas que ha registrado en la aplicación, de acuerdo con cuatro filtros, los cuales son: extraviadas sin reunir, extraviadas reunidas, encontradas sin reunir y encontradas reunidas. Cada elemento de la lista es una ficha con la especie, raza y descripción de la mascota, desde este apartado es posible visualizar las fotografías de la mascota, se puede editar su información y se puede eliminar el reporte.



a) Mis mascotas *b) Crear mascotas* *c) Avatar de mascotas*

Figura 7 Interfaz de aplicación móvil nativa para Android o iOS para reportar mascotas.

Para crear un reporte nuevo tanto de una mascota extraviada como encontrada, el usuario tiene que pulsar el botón Crear mascota, que se muestra en la figura 7a. Esto desplegará un formulario en el que deberá ingresar los datos de la mascota, su última ubicación y sus fotografías, tal como se observa en la figura 7b y 7c. Una vez creado el reporte, la ficha con la nueva mascota aparecerá en el listado de la figura 7a. La figura 8a muestra la interfaz de Buscar mascotas, en ella el usuario tiene que ingresar una ubicación para poder ver una lista con los reportes que se han hecho cerca de esa zona.



a) Buscador *b) Conversación* *c) Mensajes*

Figura 8 Interfaz de Buscar mascotas y Conversación para mascota encontrada.

También se puede ajustar la distancia de búsqueda en relación a la ubicación especificada, de tal forma que el sistema muestra los reportes de mascotas que estén en un radio de búsqueda dentro del valor especificado.

Al igual que en la sección Mis mascotas, al buscar se puede aplicar uno de los cuatro filtros: extraviadas sin reunir, extraviadas reunidas, encontradas sin reunir y encontradas reunidas. Cada ficha de la lista muestra los datos de una mascota, además de una fotografía y un mapa marcado con su última ubicación. Así mismo, al pulsar sobre el botón Fotos, se pueden ver más fotos de la mascota y al pulsar sobre el botón “Enviar Mensaje al Dueño”, se abrirá el sistema de mensajería de la aplicación como se puede ver en la figura 8b y 8c.

4. Discusión

Las mascotas típicamente representadas por perros y gatos son parte fundamental de la vida de muchas personas, ya que muchas ocasiones ayudan a cubrir necesidades emocionales y afectivas.

En un informe presentado por el Sistema de Información Legislativa (SIL) de la Secretaría de Gobernación, se estima que en México habitan alrededor de 23 millones de mascotas, donde solo un 30% son tutelados por un ser humano [Rodríguez, 2020].

En la Ciudad de México, cada semana se pierden más de 200 mascotas, de las cuales se calcula que el 80% no logran regresar a sus hogares [Delgado, 2016]. En el caso particular de los perros que terminan en situación de calle, estos suelen estar extremadamente desnutridos y sufren enfermedades como son: parvovirus, dirofilariasis canina, parásitos intestinales y sarna, situación que reduce su esperanza de vida a un año [Manabi, 2016].

Los riesgos a los que se enfrenta una mascota cuando no puede regresar a su hogar son en extremo preocupantes, por lo que resulta de vital importancia reportar su desaparición de inmediato para reducir su exposición a los peligros que podría enfrentar en la calle.

De igual forma, el reportar el avistamiento de una mascota perdida, así como su resguardo, contribuyen a que éstas puedan volver a un ambiente seguro. Las soluciones que se han revisado en la literatura y que existen como alternativa comercial, tienen el problema de que están disponibles sólo en ciertos países, no tienen una forma de delimitar la búsqueda de mascotas dentro de cierta región geográfica y carecen de un sistema de mensajería interno que facilite la comunicación entre usuarios.

El sistema que se propone en este trabajo denominado como BuscandoPet, es una aplicación que no tiene ninguna de estas restricciones y que, además, al ser una aplicación híbrida multiplataforma, puede funcionar como sistema web y como aplicación móvil, lo que la hace accesible a una gran variedad de dispositivos, por lo que puede ayudar a contribuir de manera importante en la búsqueda de mascotas extraviadas.

5. Conclusiones

En este trabajo se ha presentado una aplicación híbrida multiplataforma basada en microservicios, desarrollada para consumir información de un servidor con un API REST. Esta arquitectura permite que la información del servidor esté disponible en formato JSON, lo que facilita que cualquier cliente autorizado pueda consumir la información. Este diseño en combinación con las librerías de desarrollo: Ionic Capacitor y Spring framework; facilitaron la implementación del servidor y la creación de interfaces de usuario en múltiples plataformas (móvil y web). La aplicación permite a los usuarios reportar oportunamente el extravío de sus mascotas desde cualquier dispositivo y facilita generar reportes en el momento en que se extravía o es encontrada. Estos reportes incluyen información de la mascota, fotografías y vacunas, entre otros datos. La aplicación tiene varias características que la distinguen de otras similares, como son: reportes que incluyen información para geolocalización para reportar el último lugar en el que fue vista una mascota, un sistema de mensajería que permite una interacción inmediata entre los usuarios y protege su privacidad, así como la búsqueda de mascotas dentro de áreas geográficas específicas y que se puede delimitar a un rádio de búsqueda.

Este sistema también sirve como un ejemplo de cómo el uso de tecnologías de información permite generar herramientas que puedan servir como una solución para problemáticas sociales, como es el caso de las mascotas que terminan en situación de calle.

6. Bibliografía y Referencias

[1] Anderson, D. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press. Washington, USA. 2010.

[2] Asamblea Legislativa del Distrito Federal, VI Legislatura. (2017). Ley de protección a los animales del Distrito Federal. Gaceta Oficial del Distrito Federal. Ciudad de México, México: http://www.aldf.gob.mx/archivo-54e6a63d674a3408db21ccd2c2414be8.pdf.

[3] Bassett, L. Introduction to JavaScript Object Notation. O'Reilly Media. Sebastopol, USA. 2015.

[4] Aqraldo, B., et al. Detepet Mobile Application for Pet Tracking. International Conference on Emerging Smart Computing and Informatics, pp. 48-52, 2021. doi: https://doi.org/10.1109/ESCI50559.2021.9397028.

[5] Capacitor: Cross-platform Native Runtime for Web Apps. Capacitor Docs, 2021. https://capacitorjs.com/docs.

[6] Gourley, D. Totty, B. Sayer, M. Aggard, A. Reddy, S. Http: The Definitive Guide, O'Reilly Media Inc, USA. 2002.

[7] Delgado, D. Redes y apps, la esperanza de volver a casa para mascotas perdidas en la CDMX. Animal Político. 2021: https://www.animalpolitico.com/2016/08/redes-apps-la-esperanza-volver-casa-mascotas-perdidas-la-cdmx/.

[8] Dirección General de Comunicación Social. Fauna feral en la Reserva Ecológica del Pedregal de San Ángel. Fundación UNAM. Ciudad de México, México. 2021.

[9] Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine. 2000.

[10] Rodríguez Ariel, Proposición de acuerdo sobre adopción y tenencia de perros y gatos. Sistema de Información Legislativa de la Secretaría de Gobernación. 6/octubre/2020: http://sil.gobernacion.gob.mx/Archivos/Documentos/2020/10/asun_4083538_20201006_1602003468.pdf.

[11] Frain, B. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition (3a ed.). Packt Publishing, 2020.

[12] Freeman, E. Robson, E. Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software, 2a ed., O'Reilly Media, 2021.

[13] Huellitas perdidas, 2021: https://huellitasperdidas.org/.

[14] Johnson, R., Hoeller, J., Colin, D., Harrop, R., Risberg, T. Spring Framework Reference Documentation, 2021. Spring: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/index.html.

[15] Minh, Q. Guimire, P. Truong, D. Hybrid App Approach: Could It Mark the End of Native App Domination? Issues in Informing Science and Information Technology. Vol. 14, pp. 049-065, 2017. https://doi.org/10.28945/3723.

[16] Martínez, D. Registro de animales de la CDMX en el olvido. Reporte índigo. Ciudad de México, México, 2019: https://www.reporteindigo.com/reporte/registro-de-animales-de-la-cdmx-en-el-olvido-tenencia-mascotas-bienestar-cuidado/.

[17] Newman, S. Building Microservices. Oreilly & Associates Inc. California, USA. 2015.

[18] Ockerman, S., Mastering professional scrum: A Practitioner's Guide to Overcoming Challenges and Maximizing the Benefits of Agility. Addison Wesley, 2021.

[19] Petrace. ¿Qué es Petrace?, 2021. Recuperado de: https://petrace.app/.

[20] RUAC, Registro Único Digital para Animales de Compañía de la Ciudad de México, 2022: https://www.ruac.cdmx.gob.mx/.

[21] Manabi, P., Sreejani, S, Majumder, S. Anjan, K. Anindita, B. High Early Life Mortality in Free-ranging Dogs is Largely Influenced by Humans. Scientific reports, Vol 6, Num. 9641, 2016. https://doi.org/10.1038/srep19641

[22] Schauder, J., Bryant, J. Spring Data JDBC – Reference Documentation, 2022. Recuperado en 22 de abril de 2022, de https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/#jdbc.query-methods.

[23] Spilca, L. Spring Security in Action. Manning Publications, 2020.

[24] Tangsripairoj, S. Kittirattanaviwat, P. Koophiran, K., Raksaithong, L. Bokk Meow: A Mobile Application for Finding and Tracking Pets, 15th International Joint Conference on Computer Science and Software Engineering, pp. 1-6, 2018. https://doi.org/10.1109/JCSSE.2018.8457351.

[25] Wizapet. (2021). Wizapet: https://play.google.com/store/apps/details?id=com.thequizproject.wizapet.