

Universidad de La Habana  
Facultad de Matemática y Computación



# **HCVet: Aplicación móvil para historia clínica veterinaria**

Autor:

**David Campanería Cisneros  
Dayron Fernández Acosta  
Pablo Adrián Fuentes González**

Tutores:

**Alejandro Mesejo  
José L. Castañeda**

Trabajo de Diploma  
presentado en opción al título de  
Licenciado en Ciencia de la Computación)

17 de octubre, 2022

# Agradecimientos

A nuestros padres por todo su apoyo incondicional durante la carrera. A nuestros tutores Alejandro Mesejo y José L. Castañeda por haber dedicado parte de su tiempo en ayudar y servir de guía para el desarrollo de este trabajo. A nuestros compañeros de estudio y profesores de la carrera en general. Gracias a todos.

# Resumen

Este trabajo de diploma plantea el desarrollo de un programa que permite el almacenamiento y manipulación de historiales médicos de animales. Dicho programa cuenta con una interfaz gráfica y la habilidad de compartir esta información entre varios dispositivos y mantenerla sincronizada a través de un servidor central. Como una de las principales características en particular del problema, el programa debe funcionar con la menor cantidad posible de conexión a una red, así como la transferencia de datos entre dispositivos. Para el proceso de creación de la aplicación se observaron sistemas similares dentro del objeto de desarrollo, y se obtuvo información sobre las distintas características del campo de veterinaria para la creación de la base de datos. La propuesta de solución discutida muestra una implementación en Flutter como plataforma de desarrollo gracias a las facilidades dadas para la creación de aplicaciones móviles.

# Abstract

This diploma work proposes the development of a program that allows the storage and handling of medical records of animals. This program has a graphical interface and the ability to share this information between several devices and keep it synchronized through a central server. As one of the particular main features of the app, it must work with as little network connection as possible, as well as offline data transmission. For the process of creating the application, similar systems were observed within the object of investigation, and information was obtained on the different characteristics of the veterinary field for the creation of the database. The solution proposal discussed shows an implementation in Flutter as a development platform thanks to the facilities provided for the creation of mobile applications.

# Índice general

<b>Introducción</b>	<b>1</b>
<b>1. Estado del Arte</b>	<b>7</b>
1.1. La historia clínica en la medicina veterinaria . . . . .	8
1.1.1. Conclusiones . . . . .	10
1.2. Aplicaciones desarrolladas en Cuba en favor del Bienestar Animal . .	12
1.2.1. Web del Registro Cubano de Mascotas (RCM) . . . . .	12
1.2.2. BACuba . . . . .	12
1.3. Aplicaciones móviles que gestionan Historias Clínicas Veterinarias . .	14
1.3.1. Dog Health . . . . .	14
1.3.2. Pet Soft . . . . .	14
1.3.3. Petmeddata . . . . .	15
1.3.4. VitusVet . . . . .	16
1.3.5. Conclusiones de las aplicaciones presentadas . . . . .	17
1.4. Tecnologías para el desarrollo de aplicaciones móviles . . . . .	17
1.4.1. Programación Nativa de Android : Java y Kotlin . . . . .	17
1.4.2. React Native . . . . .	19
1.4.3. Xamarin . . . . .	21
1.4.4. Flutter . . . . .	23
1.4.5. Conclusiones sobre las plataformas mostradas . . . . .	25
1.5. Tecnologías para el desarrollo de base de datos . . . . .	25
1.5.1. Bases de Datos para Android . . . . .	25
1.5.2. Berkeley DB . . . . .	26
1.5.3. Couchbase Lite . . . . .	27
1.5.4. SQLite . . . . .	28
1.5.5. Conclusiones sobre las bibliotecas mostradas . . . . .	30
<b>2. Producto</b>	<b>31</b>

<b>3. Propuesta de Arquitectura y Modelo de Base de Datos</b>	<b>33</b>
3.0.1. Arquitectura propuesta . . . . .	33
3.0.2. Modelo de Datos . . . . .	34
<b>4. Detalles de Implementación</b>	<b>41</b>
4.1. Tecnologías utilizadas en HCVet . . . . .	41
4.1.1. Dart . . . . .	41
4.1.2. Kotlin . . . . .	41
4.1.3. Paquete http . . . . .	42
4.1.4. Paquete uuid . . . . .	42
4.1.5. Paquete encrypt . . . . .	42
4.1.6. Paquete sqfentity . . . . .	42
4.2. Detalles de la Implementación . . . . .	43
4.2.1. Estructura de Model . . . . .	43
4.2.2. Implementación de la transferencia de datos no sincronizada .	43
4.2.3. Implementación del componente Online . . . . .	44
4.2.4. Sincronización de la base de datos local con el servidor . . . .	44
<b>5. Pruebas de Funcionalidad y Experimentos</b>	<b>46</b>
5.1. Prueba 1: Creación de cuenta o autenticación de usuario . . . . .	47
5.2. Prueba 2: Creación de mascota . . . . .	48
5.3. Prueba 3: Eliminar mascota . . . . .	49
5.4. Prueba 4: Creación y mostrado de consultas . . . . .	50
<b>6. Conclusiones</b>	<b>51</b>
<b>Recomendaciones</b>	<b>53</b>
<b>Bibliografía</b>	<b>54</b>

# Índice de figuras

1.1. Diferencias entre la HCI y la HC tradicional. . . . .	11
3.1. Representación del modelo de la base de datos . . . . .	40
5.1. De izquierda a derecha: Página inicial de la aplicación, página de creación de usuario y página de autenticación . . . . .	47
5.2. Página de creación de mascotas . . . . .	48
5.3. Página de central de la aplicación antes y después de crear una mascota	48
5.4. Advertencia al eliminar una mascota . . . . .	49
5.5. Página de creación de consulta Visita Médica y página de mostrado de esta consulta . . . . .	50
5.6. De izquierda a derecha: Página de selección de consulta a agregar, historial de consultas y filtrado de consultas en historial . . . . .	50

# Introducción

El trabajo con información médica, tanto humana como animal, ha sido a lo largo de la historia un tema sumamente importante y sensible. Debido a la gran cantidad de descubrimientos obtenidos por las ciencias de la salud, con respecto al origen, síntomas y tratamientos de los distintos padecimientos, el manejo y la cantidad de datos que es necesario mantener a la hora de establecer un diagnóstico correcto es inmensa. Por esta razón han sido creados distintos métodos de preservación de documentos médicos.

El medio de almacenamiento que mantuvo la mayor popularidad en el anterior siglo fue, por supuesto, el analógico. Los historiales médicos<sup>1</sup> se creaban y actualizaban de forma manuscrita y en ocasiones se almacenaban en algún medio analógico. Este medio mantiene una serie de particularidades que hace difícil su recomendación. Las principales adversidades que contiene este sistema son su poca facilidad en maleabilidad, replicación y distribución. Es correcto mantener la opinión de que características como estas ayudan a garantizar la seguridad, pues en general la información médica tiene un carácter privado. Sin embargo, la seguridad no es un bien exclusivo ni absoluto, pues la lucha contra vulnerabilidades se encuentra presente en cualquier tipo de estructura.

Gracias a la informatización, varias aplicaciones y dispositivos se encuentran a disposición de pacientes, profesionales, investigadores sanitarios y el público general, relacionadas con la salud. Tienen como resultado una mejora de la calidad del tratamiento al respaldar la eficiencia de la asistencia sanitaria profesional, autogestión de pacientes y prevención de enfermedades. Como ejemplos existen medidores de frecuencia cardíaca y seguidores de niveles de insulina.

Una de las principales aplicaciones informáticas que surgen a partir de la segunda década del siglo XXI, relacionadas con la atención médica, son los historiales médicos digitales (o historiales clínicos electrónicos). Estos programas le permiten a los usuarios mantener la información correspondiente almacenada en su dispositivo o en alguna base de datos, permitiendo de esta manera que doctores, e incluso entidades de salud (hospitales, laboratorios, etc.) tengan acceso a ella en caso de ser necesario.

---

<sup>1</sup>colección de documentos actualizados por el personal de la salud con respecto a la historia y actualidad de los padecimientos de un individuo



Esta estandarización presentó un gran número de ventajas y permitió la evolución de muchos procesos sanitarios.

Existe una enorme diversidad cuando se trata de tipos de historiales médicos digitales existentes. Las características cambian según el principal problema que intentan resolver. Por ejemplo, algunas son solo utilizadas por profesionales, otras solo mantienen resultados de pruebas de laboratorios, varias son diseñadas para tratar con una única enfermedad, o solo tratan una rama específica de la medicina. Entrando en estas especializaciones, se encuentran aquellas que abordan la veterinaria. Estas se centran en la actualización y conservación de información del cuidado de la salud de distintos animales. Cumplen el mismo propósito que las utilizadas en salud humana pero cambiando el tipo de pacientes. La utilidad que presentan es increíble, tanto en el ámbito doméstico como pecuario. La posibilidad de llevar y compartir un registro de los animales ayuda a la prevención de enfermedades y a la facilidad del tratamiento.

Hoy en día el bienestar animal es prioridad de la sociedad a nivel mundial, siendo este definido como un concepto amplio que incluye diversos elementos contribuyentes a la calidad de vida de un animal, incluidos los referidos en las “cinco necesidades” (libertades) [14]:

- necesidad de no sufrir hambre, sed ni desnutrición,
- necesidad de no experimentar miedo ni angustia,
- necesidad de vivir libre de incomodidad física y térmica,
- necesidad de no sufrir de dolor, lesiones y enfermedad,
- y necesidad de expresar patrones normales de comportamiento.

El bienestar animal ha sido definido por la Organización Mundial de Sanidad Animal (OIE) como el término que describe la manera en que los animales se enfrentan con el medio ambiente y que incluye su sanidad, sus percepciones, su estado anímico y otros efectos positivos o negativos que influyen sobre los mecanismos físicos y psíquicos del animal [14].

Por otra parte, el biólogo y profesor de bienestar animal Donald Broom lo describe como: “*el equilibrio del estado físico y psicológico de un animal en su intento por adaptarse y sobrevivir en las condiciones de su entorno o medio ambiente*” [3]. Para Dawkins [7] el animal vive con un adecuado bienestar cuando “*está sano y tiene lo que quiere*”.

A continuación, se muestra una lista planteada por Blasco [2] que describe de manera general las principales formas de relación humano-animal a través de la historia:

1. Cría de animales en granja para consumir sus productos (leche, huevo, etc.).

2. Cría y matanza de animales para consumir su carne.
3. Cautiverio de animales fuera de sus ambientes naturales con fines de esparcimiento (zoológicos, circos, parques, etc.).
4. Deporte (caza, pesca).
5. Experimentación con animales.
6. Animales de compañía.
7. Animales usados para trabajo (guarda, transporte).
8. Espectáculos con animales amaestrados (acuarios, circos, etc.).
9. Espectáculos que involucran agresividad hacia los animales (tauromaquia, pelea de gallos, pelea de perros, etc.).
10. Tratamiento de plagas (ratas, conejos, insectos, etc.).

En la actualidad, desafortunadamente, la gran demanda de productos ha ocasionado el aumento de sistemas intensivos de producción animal que atentan contra el bienestar de los mismos. Ciertas especies animales son explotadas y consideradas como meras máquinas de producción, sin tener en cuenta que son seres que sufren de manera física y emocional. Es por ello que hoy en día se han desarrollado una serie de políticas y prácticas para protegerlos.

Según Giménez [9] la sociedad demanda, cada día más, no solo que los animales domésticos reciban un trato digno y que no proliferen los abandonos y maltratos, sino que se beneficien de una consideración cada vez mayor, que reciban un trato adecuado a su condición de seres vivos sensibles y que la concepción misma del animal como objeto del Derecho alcance una mayor coherencia jurídica. Este sentir colectivo se ha venido mostrando, fundamentalmente, en el desarrollo de legislaciones y marcos normativos para funcionar como vía de la protección estatal de los animales y sus posibles derechos, tanto en Europa y Estados Unidos, como en América Latina.

En el caso de Cuba, fue aprobado en febrero del año 2021 el Decreto-Ley No.31 de Bienestar Animal, lo cual constituye un paso de avance en el empeño de proteger a los animales. Esta ley regula los principios, deberes, reglas y fines respecto al cuidado, la salud y la utilización de estos individuos para garantizar su bienestar. El Ministerio de Salud Pública del país (MINSAP) dispone de programas de vigilancia, prevención y control de enfermedades zoonóticas, enfocados en aquellas que constituyen un riesgo para la salud. Para cumplir con los objetivos de estos programas se realizan diversas acciones como el monitoreo de la ocurrencia de casos, diagnóstico microbiológico,

control de focos, atención médica a las personas, vacunación de grupos de riesgo y poblaciones de animales, educación sanitaria y promoción de salud. [15]

La promulgación sistemática de leyes que protegen a estos seres vivos y prohíben prácticas violentas e innecesarias en contra de su vida y dignidad, evidencia una tendencia al reconocimiento y la protección normativa del derecho al bienestar de los animales [12]. Sin lugar a dudas, las personas se hallan cada vez más preocupadas por el cuidado de los mismos; lo cual ha conllevado a la generación de nuevas tecnologías para optimizar los servicios de atención a los animales. Es en este punto donde se inscribe el presente trabajo mediante el cual se intenta desarrollar una herramienta para conservar de forma segura y conveniente el historial médico de animales afectivos. Esta herramienta, en forma de aplicación móvil con respaldo de datos en un servidor, será de ayuda tanto a propietarios de mascotas como a médicos veterinarios en la consecución de un mayor bienestar de los animales bajo su cuidado.

## **Motivación**

Desde hace una década los historiales clínicos electrónicos forman parte de los sistemas de atención médica y veterinaria en gran parte de los sistemas de salud del mundo. La conveniencia y agilidad que brindan facilitan el proceso de atención al paciente, así como el trabajo que realiza el profesional. Las grandes capacidades que pueden ser explotadas de este campo es justificación suficiente para la realización de esta investigación. Durante todo este proceso hemos mantenido como motivación principal el avance que estos resultados podrían mostrar al sistema de salud animal cubano. Teniendo en cuenta los efectos que han mantenido otras investigaciones y la aplicación práctica de ellas, los veterinarios cubanos podrían llegar a aumentar en gran medida la índole y la calidad de los servicios que ofrecen.

## **Formulación del problema**

A los dueños y cuidadores de animales domesticables les sería conveniente tener una herramienta que les permita gestionar los datos clínicos históricos de la condición de salud y los servicios que han recibido los animales a su cargo. También al sector de los profesionales que ofrecen servicios a estos animales le resultaría útil contar con un medio para el fácil acceso a los datos sobre los animales bajo su cuidado. Dicha herramienta estaría orientada a agilizar la prestación del servicio dado y a mejorar la calidad de este mediante el acceso a información actualizada y fiable sobre los animales atendidos. Las principales funcionalidades del producto propuesto son:

- Almacenar información de forma segura sobre las características básicas del animal bajo el cuidado del usuario.

- Proveer acceso a los profesionales a información histórica sobre el cuidado de los animales para no repetir errores cometidos anteriormente y agilizar la aplicación de sus servicios.
- Crear de forma rápida nueva información sobre los animales.
- Permitir el acceso a datos, controlado por el dueño del animal, a otros usuarios con el fin de delegar el cuidado del animal.
- Sincronizar nuevos datos creados por varios usuarios con un mismo animal bajo su cuidado.

## Hipótesis

Trabajando sobre la plataforma Flutter a través de Dart, es posible crear un interfaz de usuario funcional, así como la posibilidad de implementar las capacidades de transmisión de información mediante una red local.

Utilizando el lenguaje de programación *C#*, el framework *.NET* 6 y las herramientas construidas sobre este, se podrá implementar un sistema multiplataforma que funcione como soporte de las funcionalidades antes mencionadas. Se podrá también crear un mecanismo de interoperabilidad entre los sistemas de Dart y *.NET* orientado al cumplimiento de estas funcionalidades.

## Objetivos

En este trabajo se establece como objetivo el diseño y desarrollo de una aplicación móvil de fácil uso para la gestión y mantenimiento seguro de la Historia Clínica de animales domesticables. Se propone el empleo de una base de datos relacional y un sistema alojado en un servidor que permita el acceso a los datos y exponga una *API* de fácil uso para que servicios clientes accedan a ella mediante el protocolo *HTTPS*.

Dado el volumen de trabajo asociado al objetivo planteado este trabajo de diploma se ha dividido en dos. Uno de ellos será “HCVet: Servidor y API de soporte de aplicación móvil para historia clínica veterinaria” que abordará el diseño de una API de fácil uso que responda de manera eficiente a las demandas de manejo de datos desde y hacia la aplicación móvil. El otro trabajo es el presente en el cual abordará los aspectos relativos al desarrollo de la aplicación móvil.

## Tareas

- Investigar sobre el ciclo de desarrollo de software eficiente y sostenible y sobre arquitecturas de software que permitan implementar un sistema de fácil mantenimiento y extensión.

- Analizar y probar tecnologías de desarrollo de aplicaciones móviles y bases de datos que permitan el rápido desarrollo del producto deseado.
- Despliegue del software de forma pública y gratuita para una etapa de prueba del producto.

## Organización de la tesis

El Trabajo de Diploma consta de cinco capítulos, seguidos de conclusiones, recomendaciones y referencias bibliográficas.

- Capítulo 1: “Estado del Arte”, se expone un estudio de varias iniciativas similares a la presente propuesta. Se analiza el origen de los softwares usados actualmente, sus características y su impacto social.
- Capítulo 2: “Producto”, se expone brevemente las características y funcionalidades principales del producto a desarrollar.
- Capítulo 3: “Propuesta de Arquitectura y Modelo de Base de Datos”, se expone la arquitectura-patrones utilizados durante la creación del proyecto. Además, se explica el modelo de base de datos relacional propuesto.
- Capítulo 4: “Detalles de Implementación”, explicación del ciclo de desarrollo y análisis de las herramientas y tecnologías seleccionadas para la implementación de la solución.
- Capítulo 5: “Pruebas de Funcionalidad y Experimentos”, se revisará la etapa final del desarrollo del producto, realizando pruebas sobre la funcionalidad de este.

# Capítulo 1

## Estado del Arte

Actualmente existe una creciente preocupación por el cuidado animal donde la aparición de nuevas tendencias tecnológicas ha permitido el desarrollo de aplicaciones que facilitan no solo el cuidado de las mascotas por sus dueños, sino que hacen mucho más fácil el trabajo de las clínicas veterinarias como el de los profesionales de la salud.

Es así, como se puede observar un aumento de aplicaciones para Android dirigidas a establecimientos veterinarios que consiguen optimizar muchas de las tareas relacionadas con esta actividad permitiendo prestar un mejor servicio por parte de dichos clínicas veterinarios y a su vez ofrecerle suficiente información al dueño de la mascota de manera que pueda tomar la mejor decisión cuando se trata de vigilar la salud del animal y la elección del centro veterinario que mejor lo atienda.

En este capítulo se estudiará en que consisten las historias clínicas y que papel juegan dentro de la medicina veterinaria. Además, se analizarán algunas de las aplicaciones que en la actualidad tienen funcionalidades similares a las que se plantean en este documento para ser implementadas en el producto final. Así como las prestaciones de algunas de las aplicaciones encontradas que han sido desarrolladas en nuestro país en favor del bienestar animal. Luego de esto se estudian las posibilidades para la construcción del software en cuestión, decidiéndose una plataforma de desarrollo de aplicaciones móviles y un modelo de base de datos para almacenar los datos de manera local.

## 1.1. La historia clínica en la medicina veterinaria

La historia clínica (HC) es el documento que avala legalmente el trabajo del médico, pues en ella se expresan los resultados obtenidos en el diagnóstico clínico, y sirve de apoyo para el planeamiento, ejecución y control en cada caso, de las acciones destinadas a la recuperación y rehabilitación de la salud del paciente. Es un instrumento mediador a través del cual el médico elabora el diagnóstico, fundamenta el pronóstico, consigna el tratamiento y la evolución del paciente, siendo un documento único, integrado y acumulativo para cada paciente. La existencia de normas y leyes que hacen obligatorio su empleo, argumentan desde el punto de vista jurídico y normativo sus usos en lo asistencial, formativo y docente, científico e investigativo, evaluador de la calidad asistencial, administrativo y jurídico legal. De esta forma, la HC se convierte en el soporte escrito sobre el cual quedan las evidencias de la atención médica integrada que hacen todos los profesionales y técnicos de la salud partícipes en la misma [16].

El profesor cubano Llanio Navarro la considera como una guía metodológica para la identificación integral de los problemas de salud de cada paciente que establece todas sus necesidades. Es fundamental para analizar el proceso patológico del paciente y su evolución. A partir de lo expresado, toda la información que se obtiene con exactitud en la inspección médica debe ser registrada en un documento llamado HC [6].

Todos los datos registrados en este documento se obtienen realizando el método clínico, por las diferentes vías:

- **Anamnesis:** información que surge de la entrevista clínica proporcionada por el dueño del paciente. Es en esta sección donde se indica que ha ocurrido con el paciente, mencionado de forma ordenada los distintos síntomas y dolencias que la mascota ha presentado.
- **Exploración física:** pruebas o exámenes complementarios realizados por el médico; juicios de valor que el propio médico extrae de documentos que él elabora para fundamentar un diagnóstico, prescribir el tratamiento y finalmente dejar constancia del curso de la enfermedad y el tratamiento instaurado.

Entre la información contenida en la historia clínica veterinaria se encuentran:

- **Datos relativos al animal:** nombre y características físicas, fecha de nacimiento, sexo, etc. También en este punto se incluye la información relativa al propietario, datos de contacto, etc.
- **Datos proporcionados por el propietario de forma subjetiva:** a través de algunas preguntas, el personal veterinario anotará toda la información que

proporcione el propietario, cómo qué le ocurre, desde cuándo, comportamientos extraños en el animal, síntomas, etc.

- **Datos objetivos obtenidos de la exploración clínica:** el veterinario hará una exploración completa del animal para concretar la información recibida, y todo ello aparecerá también anotado en la historia clínica.
- **Diagnóstico, pronóstico y tratamiento:** una vez recabados todos los datos, el veterinario tendrá que tomar nota del diagnóstico del animal, del pronóstico y del tratamiento a seguir. Del mismo modo, será necesario realizar revisiones para comprobar la efectividad del tratamiento, lo que también tendrá que aparecer dentro de la historia clínica del animal.

La HC es única para cada paciente, este último es sujeto de su propia investigación, la cual comienza con el diagnóstico de su enfermedad. Cabe agregar que por razones económicas y gerenciales la HC debe estar siempre disponible y facilitarse en los casos legalmente contemplados, resguardando la confidencialidad de los datos reflejados en ella. El acceso al expediente clínico sin autorización, en detrimento de un tercero, está catalogado como delito.

Se advierte que el redactar datos y argumentaciones en la historia clínica es una responsabilidad de los médicos (veterinarios) de atención, sobre la cual podrían, luego de un tiempo, sustentar sus propias evidencias defensivas ante posibles denuncias concernientes a su responsabilidad profesional. En Cuba la historia clínica es propiedad del hospital con un plazo de conservación y custodia de cinco años. Por lo cual, es conveniente recordar que cada acto profesional que se realice debe ser datada con el tiempo y lugar donde se realiza, firmada, sellada con su cuño profesional y con las aclaraciones precisas sobre el acto ejercido [23].

La HC puede presentarse en diferentes soportes: Papel que tradicionalmente ha sido manuscrito, teniendo inconvenientes para la legibilidad de la caligrafía por el volumen de espacio que ocupa y el deterioro con el paso del tiempo, videos, fotografías, estudios radiológicos y soporte informático. En los nuevos centros de atención médica y veterinaria las historias clínicas están informatizadas, mediante complejos sistemas informáticos [8].

La sustitución de la HC tradicional (en soporte papel) por la HCI responde a varias necesidades [20]:

1. Dar cumplimiento a las características y objetivos del documento HC en cuanto a los requerimientos del equipo sanitario, manteniendo la confidencialidad.
2. Resolver los dos problemas clásicos de los archivos de HC el almacenamiento de grandes volúmenes documentales y la seguridad frente a los riesgos de pérdida y de deterioro.



3. Permitir la transferencia rápida de la información sanitaria existente de un paciente a puntos lejanos, garantizando que cada paciente solo tenga un único expediente y este pueda ser consultado simultáneamente en distintos lugares.
4. Soportar las decisiones médico-asistenciales, mediante la interacción con bases de datos, que permitan una rápida consulta de las mejores prácticas, los protocolos de manejo y las evidencias reconocidas.
5. Poner a disposición de los educadores, investigadores y de los planificadores sanitarios esta información, en forma eficiente.

La calidad en su confección está condicionada por muchos factores. Por un lado, está el nivel de exigencia en las instituciones; por otro, el nivel de aprendizaje de los que la confeccionan. Los problemas que puedan suscitar en su confección, pueden ser atribuidos al desconocimiento, beneficios o perjuicios derivados de un contenido incompleto [8].

En la figura 1.1 se resumen las diferentes características de la HC tradicional y la HCI a través de una comparación de sus fortalezas y debilidades.

### 1.1.1. Conclusiones

Las ciencias informáticas y bibliográficas han constituido un aporte significativo para el diseño y operación de los sistemas de información del sector sanitario, en especial de la HCI, la cual ha sido aplicada a la medicina veterinaria.

La motivación, y capacitación del personal sanitario, el uso de las herramientas informáticas y de comunicación, y la toma de decisiones basada en evidencias demostrables, se convierten en la piedra angular de los servicios de atención veterinaria.

Por ende, la historia clínica veterinaria constituye un documento único confidencial e individual, en el cual queda plasmado el pronóstico, tratamiento y la evolución del paciente, de ahí la importancia de su correcta confección.

Figura 1.1: Diferencias entre la HCI y la HC tradicional.

FORTALEZAS	
HCI	HC
Accesible en todo momento	Uso sencillo
Minimizar los errores de tratamiento y diagnóstico	Fácil portabilidad
Legible	No necesita electricidad
Evita pérdida de documentos clínicos	Flexibilidad de registro
Unificación de datos de manera estandarizada	Facilidad de búsqueda
Fácil y accesible	Garantiza la reserva de información privada del paciente por mecanismos de control de archivo.
Ahorro de optimización de costos y tiempo del personal	
Accesibilidad a información epidemiológica y estadística	
DEBILIDADES	
Necesita de electricidad para su empleo	Desorden y falta de uniformidad en documentos
Dificultad en unir la información con la perteneciente a la HC tradicional.	Información ilegible
Necesidad de aprender nuevas habilidades	Si se roba o pierde la HC es imposible de recuperar
Posibilidad de pérdida de la confidencialidad si no hay precauciones	Alto costo
Dificultad con el uso de dispositivos electrónicos.	Se deteriora con el tiempo
	Dificultad en el acceso y disponibilidad
	Información incompleta

## 1.2. Aplicaciones desarrolladas en Cuba en favor del Bienestar Animal

### 1.2.1. Web del Registro Cubano de Mascotas (RCM)

Luego de más de 30 años de lucha finalmente Cuba aprueba un Proyecto de Ley sobre protección animal que busca defender los derechos de animales silvestres, de granja y de compañía. Los animales de granja y silvestres tienen derechos y una connotación legal y ética, pero no social. En cambio, los de compañía si pueden y deben entrar en este grupo por su vinculación estrecha al dueño, su familia y el entorno comunitario. Desde esta perspectiva se hace obligatorio que la mascota sea identificada oficialmente y cambiar su estatus de objeto a sujeto. En una reunión efectuada en marzo de 2006 en el Consejo Científico Veterinario de la Habana algunos médicos, técnicos y estudiantes veterinarios abogaron por realizar un estudio para un registro de las mascotas a través de una intranet del Instituto de Medicina Veterinaria. Su objetivo era poder acceder a los datos de las mascotas desde cualquier lugar del país.

Con este enfoque se crea la **web del Registro Cubano de Mascotas**[11], un espacio digital que brinda identificación, respaldo en el cuidado, protección y tenencia responsable de los animales de compañía. Como una de las particularidades se destaca la posibilidad de obtener un documento de identidad, totalmente gratuito; que llevará en él todos los datos de los animales que se inscriben y las de su persona responsable. Este es un proceso sencillo e intuitivo en el que se deben realizar los pasos mostrados a continuación:

- Acceder a través del teléfono celular o una computadora al sitio [www.mascotascuba.com](http://www.mascotascuba.com)
- Rellenar los datos solicitados como nombre de la mascota, fecha de su nacimiento, sexo, raza, foto, dirección.
- Ingresar el correo electrónico o teléfono del tutor de la mascota.

Hasta agosto de 2021, estaban oficialmente registrados en la plataforma alrededor de 21 000 animales. De ellos, el 60 % eran perros, 30 % gatos, 7 % aves y un 3 % de roedores, reptiles, equinos, bovinos, porcinos, y otras especies.

### 1.2.2. BACuba

**BACuba**[1] [5] es una aplicación para teléfonos móviles que tiene como fin lograr el bienestar animal y sensibilizar a las personas para la protección de estos seres vivos. Dicha aplicación es una iniciativa de la organización **Bienestar Animal Cuba**[21],

la mayor red de voluntarios para la protección animal. Fue lanzada desde el pasado 19 de enero de 2022, y se basa en el principio que rige la organización: rescatar animales abandonados y brindarles todos los cuidados. También buscan promover campañas, actividades, ferias y spots para garantizar el bienestar animal. Esta novedosa aplicación posee varias utilidades separadas en varias secciones:

- **Adoptar:** En este panel se muestran todos los datos (nombre, edad, sexo, raza) y con una foto incluida de los animales que están en adopción, ya sea perros, gatos u otros.
- **Reportes:** Esta es una de las secciones más humanas, ya que los usuarios podrán reportar desapariciones de mascotas, acciones de maltrato animal, atropellos, abandonos, animales en estado de vulnerabilidad. Pueden hacerlo desde cualquier parte del país y con estas denuncias los voluntarios de las distintas redes de ayuda animal o cualquier persona sensibilizada tomarán cartas en el asunto.
- **Cuidados:** En este apartado se brindan informaciones de mucha ayuda para los dueños de mascotas y que son útiles para todas las provincias y municipios del país. Por ejemplo, una lista con direcciones y teléfonos de las principales clínicas veterinarias, tiendas de accesorios y medicinas para sus mascotas, y algunos artículos de interés para aumentar la cultura del cuidado animal.
- **Noticias:** Aquí te podrás enterar de eventos relevantes como ferias de adopciones a lo largo de toda Cuba, donaciones y otras actividades en las que quizás te interese participar.
- **Mis Mascotas:** En este apartado podrás registrar a tus animales con sus nombres, fecha de nacimiento, y todos los datos de su carnet de identidad, en el caso de poseer uno. En esta base de datos podrás incluir, además, un registro con los días de vacunación y desparasitación, a modo de recordatorio.

Información general de BACuba:

- Funciona con los datos móviles activados o por medio de servicio Wifi.
- Está diseñada únicamente para dispositivos móviles, con un tamaño de 12.60 MB y requiere una versión de Android 5.0 o versiones superiores.
- Es útil para cualquier lugar del país.
- Puede ser descargada, libre de costo, de la plataforma APKLIS.
- Tiene un diseño sencillo pero atractivo, y cumple con su función de informar, orientar, educar y ayudar en cuanto a materia de protección animal.

## 1.3. Aplicaciones móviles que gestionan Historias Clínicas Veterinarias

### 1.3.1. Dog Health

**Dog Health**[33] es una aplicación totalmente gratuita para móviles y tablets, fue creada por un comunicador italiano, por lo que está solo disponible en el idioma inglés. Creada para llevar el registro veterinario de nuestras mascotas en Android. Dentro de sus principales prestaciones se encuentran:

- Guardar los datos personales de las mascotas como su nombre, peso, fecha de nacimiento, número de chip, entre otros.
- Hacer un seguimiento de anteriores visitas al veterinario.
- Administrar a los veterinarios.
- Hacer un seguimiento de las vacunas.
- Recordatorios de citas y visitas.
- Memorizar todas las administraciones de medicamentos realizadas y por realizar.
- Buscar y contactar con el veterinario más cercano.
- Añadir múltiples mascotas.
- Realizar backups y restaurarlos.
- Guardar la evolución del peso y altura de las mascotas de forma continuada.

### 1.3.2. Pet Soft

**Pet Soft**[41] [4] es una plataforma para veterinarias y dueños de mascotas creada para almacenar y gestionar agendamientos con todo lo relacionado del paciente estableciendo citas, vacunas y todo tipo de información detallada con el fin de contar con un completo historial médico que le permita no solo al profesional veterinario, sino a los dueños de las mascotas acceder de manera segura a la información completa del animal. Esta plataforma es completamente web y la aplicación se puede descargar desde cualquier dispositivo con sistema operativo Android o IOS.

Como ventaja para el usuario sea médico o dueño de la mascota conseguirá descargar la aplicación de forma intuitiva donde podrá ingresar toda la información

requerida y vital para los pacientes. Su fortaleza radica en la fácil visualización y datos de carga que aseguran una experiencia agradable para sus usuarios. Con la implementación del software los usuarios pueden gestionar de forma segura y sencilla todo lo relacionado con las mascotas y acceder a la ubicación de la red más completa de veterinarias. La información de cada mascota, se alberga en la web, de modo que, en cada centro veterinario, se podrá descargar, sin necesidad de que se tengan que volver a registrar los datos en el sistema. Dentro de sus principales beneficios se encuentran:

Beneficios para veterinarias:

- Consultas y servicios: crear consultas y servicios; todo bajo control del software.
- Tarifas y bonos: manejar tus propias tarifas, generar bonos promocionales.
- Citas y alarmas: notificaciones para las citas programadas o crear una directamente.
- Veterinarias en red: consultar la historia clínica de la mascota si ha estado en otra veterinaria.
- Médicos veterinarios: crear usuarios especializados para tu veterinaria.

Beneficios para tu mascota:

- Datos básicos: registrar todas las mascotas.
- Historia clínica: llevar el historial de las consultas de tus mascotas directamente de las veterinarias.
- Citas: notificaciones para las citas programadas o crear una directamente.
- Veterinarias: un mapa para buscar la veterinaria más cercana.
- Carné: llevar en la app el carnet de vacunación de tus mascotas.

### 1.3.3. Petmeddata

**Petmeddata** [19][30] es un sistema electrónico seguro para almacenar el historial médico de todas las mascotas en un solo sitio. Creado en Finlandia a finales de 2018, facilita el tratamiento integral de los animales ya que muestra instrucciones de alta, medicamentos y valores de laboratorio en orden cronológico. Los problemas de seguridad se han considerado cuidadosamente. El dueño del animal decide quién puede ver la información de la mascota. Si el propietario comparte el perfil con una nueva clínica, esta puede ver las secciones más importantes del historial clínico en el historial

del paciente. El personal de la clínica también ve las entradas, textos e imágenes del propietario. Nadie puede ver el precio y la información de pago, ni las notas de la clínica de animales. El nombre del animal o el microchip nunca aparecerán en ninguna parte. Para dueños de animales y clínicas de animales, el sistema es gratuito. A los socios se les cobra una tarifa por prestar sus servicios en el sistema. “Al utilizar *Petmeddata*, el dueño siempre tiene consigo el historial médico de la mascota, incluso si va a una nueva clínica. Y el veterinario podrá leer en un formato fácil de leer lo que se le ha hecho al animal en el pasado. No hay necesidad de llevar papeles escritos a mano o impresos”, dice la Gerente de Proyecto Eva Kaisti.

Entre sus principales características se encuentran:

- Fácil acceso online a todos los datos sanitarios de las mascotas; crea el perfil médico para cada uno de tus animales y contrólalos desde un solo sitio
- Veterinarios y otros expertos envían los datos médicos oficiales al perfil del animal.
- Posibilidad de añadir tus propios documentos y notas.
- Funciona por conexión a internet desde un teléfono móvil, tablet u ordenador.
- Sitio seguro; no almacena información personal tuya, sólo la información médica de tus mascotas.
- Posibilidad de compartir el perfil de las mascotas con un profesional del cuidado de los animales, como un veterinario, un amigo, guardería para mascotas, etc.

#### 1.3.4. VitusVet

**VitusVet** [31] es un *software* de administración de prácticas veterinarias que permite a veterinarios y clínicas manejar el flujo de trabajo, comunicaciones con clientes, pagos, historias clínicas, horarios de consultas, entre otros. Permite el envío y recibimiento de texto, imágenes y actualizaciones en tiempo real.

Usando aplicaciones móviles de *VitusVet* múltiples veterinarios pueden compartir historias clínicas de mascotas con clientes y hacer recordatorios sobre encuentros futuros. La plataforma *VitusPay* permite a los dueños de mascota realizar pagos mensuales a través de tarjetas de crédito. Los clientes pueden pedir consultas con los veterinarios a través de mensajes de texto, emails, plataformas de redes sociales y aplicaciones móviles. El *software* permite a veterinarios configurar y automatizar respuestas basándose en el personal disponible, horario de trabajo, días feriados y enviar recordatorios a través de postales personalizadas.

### 1.3.5. Conclusiones de las aplicaciones presentadas

En esta sección se analizaron aplicaciones con similitudes a la que se plantea como finalidad en este documento. Todo esto con el objetivo de adoptar las ideas de la competencia o mejorarlas dependiendo del caso, sobre este tema se puede afirmar lo siguiente:

1. En general las herramientas en esta sección permiten la creación y el manejo de las historias clínicas digitales para nuestras mascotas, pero como se observó en cada caso estas no brindan la posibilidad para exportar de manera offline<sup>1</sup> [17] los datos de las mascotas que queramos compartir con otro usuario. Entre los objetivos de este trabajo está la mejora de este aspecto.
2. Pocas de estas aplicaciones permiten al usuario consultar los datos de sus mascotas sin la necesidad de una conexión a Internet, aspecto que goza de gran peso entre los principales objetivos de este trabajo.
3. Las herramientas para el manejo de las historias clínicas desde un dispositivo móvil, permiten a los usuarios ya sean médicos veterinarios o dueños y encargados de las mascotas acceder de manera sencilla y segura a toda la información referente al estado de salud, seguimiento, y cuidado del animal.
4. Luego de analizar las aplicaciones anteriores se concluye como necesario el diseño de una aplicación que si cumpla con los requerimientos expuestos en este documento, porque en general estas aplicaciones no permiten el trabajo con historias clínicas animales de manera offline, aspecto de vital importancia debido a las limitaciones de la conexión a Internet que presenta nuestro país.

## 1.4. Tecnologías para el desarrollo de aplicaciones móviles

### 1.4.1. Programación Nativa de Android : Java y Kotlin

El desarrollo de aplicaciones nativas es una opción increíble para ofrecer a los usuarios la experiencia más satisfactoria en términos de la sensación y la apariencia de su aplicación. Los lenguajes más populares para la programación nativa en Android son Java [27], creado por Sun Microsystems y Kotlin [28], desarrollado por JetBrains

---

<sup>1</sup>La transferencia de datos offline en *Android* refiere a una forma de enviar datos de forma inalámbrica sin necesidad de conexión a Internet, existen varias tecnologías que permiten este tipo de transferencia como *Wifi Direct*, *Bluetooth*, *SIP*, etc.



y otras colaboraciones de código abierto. Ambos lenguajes permiten construir aplicaciones móviles nativas para Android aunque al mismo tiempo también son aptos para el desarrollo multiplataforma

Ventajas de Java:

- Al usar Java, los desarrolladores no necesitan escribir cada nueva función desde cero. En cambio, Java proporciona un rico ecosistema de funciones y bibliotecas integradas para desarrollar una variedad de aplicaciones.
- Java ofrece varias herramientas para admitir la edición automatizada, la depuración, las pruebas, la implementación y la administración de cambios. Estas herramientas hacen que la programación de Java sea más rápida y rentable.
- El código de Java puede ejecutarse en cualquier plataforma subyacente, como Windows, Linux, iOS o Android, sin tener que volver a escribir. Esto lo hace especialmente poderoso en el entorno actual, donde queremos ejecutar aplicaciones en múltiples dispositivos.

Desventajas:

- En un sentido estricto, Java no es un lenguaje absolutamente orientado a objetos, a diferencia de, por ejemplo, Ruby o Smalltalk. Por motivos de eficiencia, Java ha relajado en cierta medida el paradigma de orientación a objetos, y así por ejemplo, no todos los valores son objetos. El código Java puede ser a veces redundante en comparación con otros lenguajes. Esto es en parte debido a las frecuentes declaraciones de tipos y conversiones de tipo manual (casting). También se debe a que no se dispone de operadores sobrecargados, y a una sintaxis relativamente simple. Sin embargo, J2SE 5.0 introduce elementos para tratar de reducir la redundancia, como una nueva construcción para los bucles “foreach”. A diferencia de C++, Java no dispone de operadores de sobrecarga definidos por el usuario. Los diseñadores de Java tomaron esta decisión puesto que consideraban que, bajo ciertas circunstancias, esta característica podía complicar la lectura y mantenimiento de los programas.
- Java fue diseñado para ofrecer seguridad y portabilidad, y no ofrece acceso directo al hardware de la arquitectura ni al espacio de direcciones. Java no soporta expansión de código ensamblador, aunque las aplicaciones pueden acceder a características de bajo nivel usando bibliotecas nativas (*JNI*, *Java Native Interfaces*).

Ventajas de Kotlin:

- Las construcciones son más concisas, por lo cual se escriben menos líneas del código.
- Kotlin cuenta con funciones de extensión. Se permite que los desarrolladores agreguen métodos a clases sin hacer cambios en el código.
- Las corrutinas se basan en conceptos de otros lenguajes, lo cual simplifica la gestión de las tareas.

Desventajas:

- Kotlin no es un lenguaje tan maduro como Java. Lo que significa que pueden existir varios errores y cambios considerables en cada actualización.
- Para obtener una referencia a una vista en Android, tienes que vincular manualmente los componentes, generando mucho más código.

### 1.4.2. React Native

**React Native**<sup>[40]</sup> es un framework que permite construir aplicaciones móviles nativas, para iOS y Android (así como también para Android TV, macOS, tvOS, Windows y UWP<sup>2</sup>, aunque nos enfocaremos en los dos gigantes de la telefonía móvil), utilizando solamente JavaScript y React.

Utiliza el mismo diseño que React.js, permitiendo usar elementos de interfaz de usuario móvil. A pesar de que React.js y React Native usen la misma estructura de código, no sirven para lo mismo, mientras React.js es utilizado para hacer páginas web y trabaja con elementos del virtual DOM, por el otro lado React Native utiliza elementos nativos de interfaz de usuario de Android y iOS (entre otras) para crear aplicaciones para ambas plataformas.

Habiendo mencionado la UI, React Native usa el mismo paradigma fundamental de construcción de bloques para la interfaz de usuario que las aplicaciones nativas puras de Android e iOS (a los que React Native denomina Componentes), pero gestiona la interacción entre los mismos utilizando las capacidades de JavaScript y React.

Similar a React para web, en este caso las aplicaciones están escritas usando una mezcla de JavaScript y marcado XML, conocido como JSX<sup>3</sup>. Luego, detrás del telón, el conocido **bridge** o puente de React Native invoca las APIs de renderizado nativas en Objective-C (para iOS) o Java (para Android). React Native también expone interfaces JavaScript para las API de la plataforma, por lo que sus aplicaciones pueden acceder a sensores y funciones de la plataforma como la cámara del teléfono o la ubicación del usuario.

---

<sup>2</sup>Universal Windows Platform

<sup>3</sup>JavaScript Syntax Extension y ocasionalmente mencionado como JavaScript XML

React Native surgió por la necesidad de una solución nativa para los problemas que presentaba el uso de HTML5 en la versión móvil de Facebook, que resultó en una aplicación inestable y con poca fluidez. La idea surge a partir de Jordan Walke, ingeniero de software de Facebook, que encontró una forma de generar elementos de interfaz de usuario para iOS a partir de un hilo de JavaScript en segundo plano, lo que sentó las bases de lo que sería React para la web. Esto unido al interés de la compañía llevó a que meses más tarde Facebook lanzara la primera versión de React.js, y Christopher Chedeau<sup>4</sup>, durante una charla técnica explicara, que Facebook ya estaba usando React Native en producción para su aplicación de grupo y su aplicación de administrador de anuncios.

#### Ventajas

1. Traduce su marcado a elementos de UI nativos reales, además, React funciona por separado del hilo principal de la interfaz de usuario, por lo que su aplicación puede mantener un alto rendimiento sin sacrificar la capacidad.
2. Para los desarrolladores acostumbrados a trabajar en la Web con React.js, significa que pueden desarrollar aplicaciones móviles con el rendimiento y la apariencia de una aplicación nativa, mientras usan herramientas familiares.
3. Una comunidad sólida, hay miles de colaboradores que actualizan constantemente la biblioteca.
4. Permite optimizar las aplicaciones nativas por separado escribiendo código específico de cada plataforma y usándolo desde JavaScript.
5. Mejora la experiencia de desarrollo gracias a su Hot Reload o recarga en caliente, que permite a los desarrolladores visualizar cambios efectuados en la aplicación de inmediato, sin necesidad de re-compilar todo el proyecto.
6. Alta fiabilidad al estar respaldada por Facebook, siendo usada como herramienta de desarrollo por grandes empresas como Instagram, Microsoft, Pinterest y Walmart.

#### Desventajas

1. La navegación integrada en React Native no es perfecta y no es comparable a la navegación nativa.
2. Tiene dificultades para crear transiciones y animaciones complejas.

---

<sup>4</sup>También conocido como Vjeux, es un ingeniero front-end en Facebook que se graduó de EPITA (la primera escuela de ingenieros especializada en informática de París)

3. Debido a que React Native es una abstracción sobre las API existentes de Android y iOS, usualmente cuando hay nuevas versiones puede ser que esa funcionalidad tarde un poco en llegar a la plataforma aunque siempre llega.
4. La documentación es deficiente para algunos casos, producto a la misma velocidad con que se realizan actualizaciones, muchos colaboradores no consideran necesaria su documentación sobre la marcha.

### 1.4.3. Xamarin

**Xamarin** [32] es un kit de herramientas de desarrollo de aplicaciones multi-plataforma que nos permite producir aplicaciones nativas de *Android*, *iOS*, *tvOS*, *watchOS*, *macOS* y *Windows* con *UI* unificadas, siendo parte en la actualidad de la plataforma de desarrollo *.NET*<sup>5</sup> [29]. Fue en sus inicios construida como herramienta por los desarrolladores de Mono<sup>6</sup> y siendo fundada como compañía homónima el 16 de mayo del 2011, para ser adquirida en el 2016 por Microsoft. Fue para cuando Microsoft adaptó el *Kit* de Desarrollo de Software (*SDK*)<sup>7</sup> de Xamarin a sus políticas de código abierto como hizo con *.NET*, que se convirtió en parte del entorno de desarrollo integrado de Xamarin Visual Studio lo cual le dio gran aceptación dentro del mundo de desarrolladores *.NET*.

Xamarin requiere de un solo lenguaje para desarrollar aplicaciones para las distintas plataformas, ese lenguaje es *C#*. Los proyectos con Xamarin compilan de forma nativa, lo que lo convierte en una opción para crear aplicaciones con un alto rendimiento y con un aspecto nativo. Estas aplicaciones a menudo se comparan con las nativas para las plataformas de desarrollo móvil *iOS* y *Android* en términos de rendimiento y experiencia del usuario. En las aplicaciones desarrolladas con Xamarin el código relacionado a la lógica de negocios, el manejo de la Base de datos y el acceso a la red puede ser compartido entre todas las plataformas y a su vez permitir desarrollar una capa de interfaz de usuario diferenciada para cada plataforma, permitiendo que las aplicaciones compiladas de esta manera, se vean de forma nativa. Con Xamarin se puede desarrollar en *Windows* o *Mac* y compilarse en paquetes de aplicaciones nativas, como un archivo *.apk* en Android o uno *.ipa* en *iOS*.

Ventajas:

---

<sup>5</sup>*.NET* es una plataforma de desarrollo compuesta por herramientas, lenguajes de programación y bibliotecas para crear muchos tipos diferentes de aplicaciones.

<sup>6</sup>Mono es una plataforma de desarrollo de código abierto basada en *.NET Framework*, dirigida por Miguel de Icaza y presentada por primera vez en 2001

<sup>7</sup>Un *kit* de desarrollo de software (*SDK*) es un conjunto de herramientas proporcionado usualmente por el fabricante de una plataforma de *hardware*, un sistema operativo o un lenguaje de programación.

- Se encuentra embebido en el ecosistema de *.NET Framework*, con todas las facilidades de Visual Studio concentrando todas las herramientas que eran accesibles desde Xamarin Studio junto con todo lo necesario para el trabajo con *C#* y más, de forma gratuita.
- Xamarin tiene una documentación bien estructurada que incluye casos, fragmentos y tutoriales paso a paso.
- El desarrollo multiplataforma de Xamarin requiere aproximadamente 1,5 veces menos tiempo (y dinero) que el desarrollo de un proyecto nativo independiente para cada plataforma.
- Las aplicaciones que se construyen en Xamarin nos darán un mejor rendimiento y mejorarán constantemente para que coincidan con los estándares de desarrollo nativo. A diferencia de las soluciones híbridas tradicionales, basadas en las tecnologías web, una aplicación multiplataforma desarrollada con Xamarin puede clasificarse como nativa en términos de rendimiento.
- Permite crear experiencias visuales completamente nativas. Incluso por herramientas asociadas como Xamarin.Forms convierte los componentes de la interfaz de usuario de la aplicación en elementos propios de la plataforma en tiempo de ejecución. Esta última, disminuyendo el tiempo de desarrollo sustancialmente pero con sacrificios leves en el rendimiento debido a la capa de abstracción adicional.
- Xamarin *SDK*, que incluye tiempo de ejecución, bibliotecas y herramientas de línea de comandos, pasa a ser de código abierto, estando disponible para todos bajo la licencia *MIT* como parte de Visual Studio a partir de febrero de 2016, lo que acelera el crecimiento de la plataforma.

#### Desventajas:

- Dependiendo del proyecto las aplicaciones desarrolladas por esta tecnología tienden a ocupar más espacio en memoria. Para ilustrar lo anterior una aplicación de Hola Mundo desarrollada con Xamarin para Android puede alcanzar los 16 MB, en su estado puro, sin paquetes adicionales, mientras que la desarrollada de forma nativa puede ser menor a los 3 MB.
- Puede haber trabas al integrar recursos de terceros con la aplicación en Xamarin. Aunque la mayoría de las herramientas y bibliotecas ofrecen soporte completo para tecnologías nativas, es posible que un proveedor no brinde soporte de Xamarin. Siempre existe la posibilidad de que necesite una capacidad o integración específica dentro de su aplicación que no sea proporcionada por la plataforma o su tienda de componentes.

- A pesar del tiempo en el mercado Xamarin.iOS, Xamarin.Android e incluso Xamarin.Forms se le presentan quejas sobre estabilidad, errores y fallas. Principalmente Xamarin.Forms que es más usado por su posibilidad de ahorrar tiempo creando prototipos y aprovechar al máximo el código compartido. Algunos desarrolladores se muestran inconformes afirmando que, en lugar de ahorrar tiempo con el uso compartido de código, puede terminar empleando más tiempo resolviendo problemas.

#### 1.4.4. Flutter

**Flutter** [26] es un conjunto de herramientas de interfaz de usuario multiplataforma que está diseñado para permitir la reutilización de código en sistemas operativos como iOS y Android, al mismo tiempo que permite que las aplicaciones interactúen directamente con los servicios de la plataforma subyacente. El objetivo es permitir que los desarrolladores entreguen aplicaciones de alto rendimiento que se sientan naturales en diferentes plataformas, adoptando las diferencias donde existen mientras comparten la mayor cantidad de códigos posibles.

Se trata de un conjunto de herramientas de Interfaz de usuario portátiles que fueron creados por Google. Este es un framework bastante nuevo, presentado por primera vez en 2015 [10], que pasó algunos años perfeccionándose en las versiones. La primera versión estable, Flutter 1.0, fue lanzado el 4 de diciembre de 2018. En el último año, ha experimentado un crecimiento muy grande en cuanto a su popularidad, esto debido a su velocidad de desarrollo, experiencia nativa y eficiente renderización de la interface. Sería para el 3 de marzo de 2021, en el evento virtual llamado Flutter Engage, que Google lanza Flutter 2, el que fuese su cambio oficial más grande, dejando de ser solo un SDK para aplicaciones móviles, convirtiéndose en un framework para el desarrollo multiplataforma incluyendo además de Android y iOS, a Windows, Linux, MacOS y Web. Como cambios en esta versión incluyendo null safety <sup>8</sup>, uno de los más emblemáticos, característica opcional al momento de crear un nuevo proyecto (siendo también posible migrar proyectos ya existentes a la nueva forma).

Flutter es un producto de código abierto, desarrollado en C, C++, Dart y Skia Graphics Engine, este último una biblioteca gráfica compacta, que también fue adquirida por Google. El lenguaje de programación estándar utilizado por Flutter es Dart, para quien no sabe, Dart es un lenguaje de propósito general un poco más antiguo que Flutter. Este fue creado en 2011 por Google para sustituir JavaScript, cuya tentativa no fue del todo exitosa, para convertirse en lo que conocemos hoy, con un lenguaje muy versátil pudiendo ser utilizado en varios entornos, con herramientas integradas

---

<sup>8</sup>Cuando se opta por la seguridad nula, los tipos de su código no admiten nulos de forma predeterminada, lo que significa que las variables no pueden contener nulos a menos que usted lo indique explícitamente

que facilitan el desarrollo y teniendo a Flutter como principal objetivo del lenguaje actualmente.

#### Ventajas

1. La representación en todas las plataformas es la misma, sin necesidad de implementar código específico para cada plataforma, independientemente de si se ejecuta en Android o diferentes versiones de iOS.
2. Con Flutter, los desarrolladores ya no tienen que lidiar con un largo tiempo en recargar su aplicación, gracias al ya mencionado Hot Reload. Pero al contrario de estos otros frameworks, la mayoría de los cambios en la interfaz de usuario se aplican mientras el desarrollador todavía está trabajando en el código, de forma instantánea.
3. Hace que sea más fácil no solo ver la interfaz de usuario, sino también crearla (diseñar un tema en una aplicación nativa de Android puede ser un desafío, con Flutter, crear un tema es mucho más sencillo). La compatibilidad con temas incorporada hace que el diseño de todos los aspectos de la interfaz de usuario sea más rápido y fácil para los desarrolladores.
4. La documentación oficial es excelente y completa, paquetes bien documentados, contando con un ritmo alto de actualizaciones y una comunidad numerosa, con muchas conferencias y eventos anualmente.
5. El lenguaje de programación Dart es muy similar a Java / Kotlin (Android), lo que facilita el aprendizaje para los desarrolladores que provienen del desarrollo móvil nativo.
6. Ha conseguido generar confianza y seguridad en los profesionales que trabajan con él, por el hecho de que una empresa como Google está detrás, además del auge que está experimentando Flutter en los últimos meses, las actualizaciones de las librerías y los widgets, así como el mantenimiento, de manera constante.

#### Desventajas

1. Al ser un framework bastante reciente con 3 años y medio (para el momento de este documento), implica que el desconocimiento y la experiencia que se tiene en su uso es mucho menor que con otros sistemas de desarrollo.
2. Presenta un mayor tamaño de las aplicaciones en el caso de iOS. En diciembre de 2018, se midió el tamaño de descarga de una aplicación Flutter mínima Hello World, arrojando como resultados, un peso de 4,06 MB para el caso de Android y 10,8 MB para el caso de iOS, el IPA es mayor que el APK principalmente

porque Apple encripta binarios dentro del IPA<sup>9</sup>, haciendo la compresión menos eficiente [24].

3. Pocas bibliotecas. A pesar de tener muchas bibliotecas nativas de Flutter que funcionan muy bien, cuando se trata de bibliotecas de terceros, las cosas no se ven tan bien. La cantidad de soluciones en Flutter es menor que la cantidad de paquetes disponibles para desarrollo nativo o React Native.

### 1.4.5. Conclusiones sobre las plataformas mostradas

La comparación está bastante equiparada, para el caso de React Native cuenta con una amplia comunidad, gran número de paquetes y posee un buen rendimiento, pero un motor gráfico aún deficiente y carece de fluidez para las animaciones más simples. Mientras Flutter a pesar de poseer una comunidad aún pequeña en comparación con React Native o Xamarin, es cooperativa y esta creciendo en una magnitud superior a la de ambas. La comunidad de Flutter en github es bastante grande con apoyo constante en comparación con las comunidades de Java y Kotlin.

Flutter llega para suplir, con creces, las deficiencias de React Native respecto al apartado gráfico y los problemas de estabilidad que padece Xamarin, esta superioridad en el apartado gráfico también se hace presente ante Java y Kotlin. Posee una excelente documentación y un lenguaje de programación que aunque no es tan conocido como JavaScript, es muy poderoso y cuenta con una sintaxis similar a la de Java y C#. Con aplicaciones optimizadas para ocupar menos espacio, donde a pesar no ser rivales para las desarrolladas de forma nativa, supera a la competencia en ese aspecto.

Luego de todo lo expuesto, sumado a la experiencia previa que se posee en el uso de **Flutter** para el desarrollo multiplataforma, **fue seleccionado para implementar la aplicación** que concluirá este trabajo.

## 1.5. Tecnologías para el desarrollo de base de datos

### 1.5.1. Bases de Datos para Android

Generalmente las bases de datos se manejan en el lado del servidor o en la nube y los dispositivos móviles solo se comunican con ellos a través de la red. Esto implica que la aplicación móvil necesite una conexión de red activa y bastante rápida para acceder a sus datos. Sin embargo, para hacer que las aplicaciones sean más receptivas y menos dependientes de la conectividad de la red, la tendencia del uso fuera de línea o la menor dependencia de la red está ganando popularidad.

---

<sup>9</sup>Es una extensión de archivo por sus siglas en inglés iPhone Application, referente al software que se puede instalar en la plataforma del sistema iOS y MacOS



Hoy en día, las aplicaciones mantienen la base de datos localmente o hacen una copia en la nube y en el dispositivo local y se sincronizan con ella una vez al día o cada vez que hay una conectividad de red. Esto ayuda a crear aplicaciones más rápidas y receptivas que son funcionales incluso cuando no hay conectividad a Internet o es limitada.

Las bases de datos para Android deben ser:

- Ligeras ya que el almacenamiento es limitado en dispositivos móviles.
- Sin requisito de servidor.
- En una forma de biblioteca con ninguna o muy limitada dependencia (incrustable<sup>10</sup>) para que se pueda usar cuando sea necesario.
- Rápidas y seguras.
- Fácil de manejar mediante código y opción para hacerlas privadas o compartidas con otras aplicaciones.
- Requerir poca memoria y consumo de energía.

Existen muchas bases de datos para Android y móviles en el mercado, pero no todas satisfacen todos los requisitos mencionados en este Trabajo. Vamos a discutir algunas de las bases de datos más populares para aplicaciones móviles y tratar de resaltar sus características, pros y contras.

### 1.5.2. Berkeley DB

**Berkeley DB (BDB)**[38] es una biblioteca de software destinada a proporcionar una base de datos integrada de alto rendimiento para datos clave/valor. Berkeley DB está escrito en C con enlaces API para C++ , C# , Java , Perl , PHP , Python , Ruby , Smalltalk , Tcl y muchos otros lenguajes de programación. BDB almacena pares de clave/datos arbitrarios como matrices de bytes y admite varios elementos de datos para una sola clave. Berkeley DB no es una base de datos relacional, aunque tiene funciones de base de datos avanzadas que incluyen transacciones de base de datos, control de concurrencia multiversión y registro de escritura anticipada.

Fue desarrollado y respaldado comercialmente por Sleepycat Software de 1996 a 2006. Sleepycat Software fue adquirido por Oracle Corporation en febrero de 2006, que continúa desarrollando y vendiendo la biblioteca C Berkeley DB. Desde su lanzamiento inicial, Berkeley DB ha pasado por varias versiones. Cada ciclo de lanzamiento

---

<sup>10</sup>Las bases de datos incrustadas son bibliotecas livianas y autónomas sin componentes de servidor, sin necesidad de administración.

importante ha introducido una sola característica principal nueva que generalmente se superpone a las características anteriores para agregar funcionalidad al producto. La evolución a veces ha llevado a cambios menores en la API o cambios en el formato de registro, pero muy rara vez han cambiado los formatos de la base de datos. Berkeley DB HA admite actualizaciones en línea de una versión a la siguiente al mantener la capacidad de leer y aplicar los registros de la versión anterior.

Entre sus principales características se encuentran:

- Los datos se almacenan en el formato nativo del lenguaje de programación.
- No tiene modo cliente-servidor.
- Caché configurable para modificar el rendimiento.
- Permite crear bloqueos de forma detallada. Esto es especialmente útil para trabajos concurrentes sobre la base de datos de forma que se bloquea una página de registros durante una transacción para evitar que se modifiquen hasta que termine, pero permitiendo actuar sobre el resto de páginas.
- Posibilidad de realizar copias de seguridad y replicación.
- Transacciones y recuperación ante errores ACID<sup>11</sup>. Esto es configurable de forma que se puede ir relajando en función de la aplicación.
- Es compatible con algunas interfaces históricas para bases de datos en UNIX.
- Permite utilizar la característica de snapshots para poder efectuar varias transacciones sobre los mismos registros de manera simultánea.
- No soporta SQL ni esquemas. A pesar de esto tiene un tamaño superior al de otras alternativas incrustadas (ocupa casi el doble que SQLite).

### 1.5.3. Couchbase Lite

**Couchbase Lite**[25] es un motor de base de datos sincronizable, liviano, orientado a documentos (*NoSQL*) integrado. Al estilo de Couchbase Server maneja documentos *JSON* y tiene el mismo mapa/reducción que este en una edición pequeña. Está escrito en *Objective-C* y *C++* y se requiere *Xcode 7* o posterior para construirlo. Couchbase Lite compila de forma nativa para *iOS*, *Android*, *MacOS* y *.NET*. Medio

---

<sup>11</sup>*ACID* es un conjunto de propiedades de las transacciones de base de datos que tienen como propósito garantizar la validez de los datos a pesar de posibles errores, fallos de energía u otros problemas que pueden surgir

megabyte optimizado, para un inicio rápido y una experiencia de usuario ágil en dispositivos conectados ocasionalmente cuando los datos importan.

Entre sus principales características se encuentran:

- Es integrado; el motor de la base de datos es una biblioteca vinculada a la aplicación, no un proceso de servidor independiente.
- Tamaño de código pequeño, actualmente menos de 600 kbytes.
- Bajo uso de memoria con conjuntos de datos móviles típicos. La expectativa es que la cantidad de documentos no sea enorme, aunque puede haber archivos adjuntos multimedia considerables.
- Al igual que Couchbase Server, almacena registros en formato *JSON* flexible en lugar de requerir esquemas predefinidos o normalización.
- Los registros/documentos pueden tener archivos adjuntos binarios de tamaño arbitrario, como contenido multimedia.
- El formato de datos de su aplicación puede evolucionar con el tiempo sin necesidad de migraciones explícitas.
- Mapear/reducir el indexado permite búsquedas rápidas sin necesidad de utilizar lenguajes de consulta especiales.
- El motor de sincronización admite conexiones de red intermitentes y poco fiables.
- Las *API* nativas son *Objective – C* (*iOS*, *tvOS*, *Mac*), *Java* (*Android*) y *C#* (*.NET*, *Xamarin*); pero un adaptador *API REST* interno opcional permite llamarlo desde otros lenguajes como JavaScript, para usar en aplicaciones creadas con PhoneGap/Cordova o Titanium.

#### 1.5.4. SQLite

**SQLite** [39] es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional autónomo, sin servidor y sin configuración. El código de SQLite es de dominio público y, por lo tanto, es de uso gratuito para cualquier propósito, comercial o privado. SQLite es la base de datos más implementada en el mundo con más aplicaciones de las que podemos contar, incluidos varios proyectos de alto perfil. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso de servidor separado. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con varias tablas, índices, *triggers* (disparadores) y

vistas está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 y 64 bits o entre arquitecturas big-endian y little-endian. Estas características hacen de SQLite una opción popular como formato de archivo de aplicación.

El código base de SQLite cuenta con el respaldo de un equipo internacional de desarrolladores que trabajan en SQLite a tiempo completo. Los desarrolladores continúan expandiendo las capacidades de SQLite y mejorando su confiabilidad y rendimiento mientras mantienen la compatibilidad con las especificaciones de interfaz publicadas, la sintaxis SQL y el formato de archivo de la base de datos. El código fuente es absolutamente gratuito para cualquiera que lo desee, pero también hay soporte profesional disponible. El proyecto SQLite se inició el 9 de mayo del 2000. El futuro siempre es difícil de predecir, pero la intención de los desarrolladores es admitir SQLite hasta el año 2050. Las decisiones de diseño se toman con ese objetivo en mente.

Entre sus principales características se encuentran:

- Las transacciones son atómicas, consistentes, aisladas y duraderas (ACID) incluso después de fallas del sistema y fallas de energía.
- No se necesita configuración ni administración.
- Implementación completa de SQL con capacidades avanzadas como índices parciales, índices en expresiones, JSON, expresiones de tablas comunes y funciones de ventana.
- Una base de datos completa se almacena en un único archivo de disco multiplataforma. Ideal para usar como formato de archivo de aplicación.
- Admite bases de datos del tamaño de un terabyte y cadenas y blobs del tamaño de un gigabyte.
- API simple y fácil de usar.
- Rápido, en algunos casos, SQLite es más rápido que la E/S directa del sistema de archivos.
- Código fuente bien comentado con una cobertura de prueba de rama del 100
- Disponible como un solo archivo de código fuente ANSI-C que es fácil de compilar y, por lo tanto, fácil de agregar a un proyecto más grande.
- Autónomo; sin dependencias externas.

- Multiplataforma: Android, BSD, iOS, Linux, Mac, Solaris, VxWorks y Windows (Win32, WinCE, WinRT) son compatibles desde el primer momento. Fácil de portar a otros sistemas.
- Las fuentes son de dominio público. Uso para cualquier propósito.

### 1.5.5. Conclusiones sobre las bibliotecas mostradas

Luego de analizar las características que presentan las bibliotecas antes mencionadas para la creación y el manejo de las bases de datos. Así como sus ventajas y desventajas propias de las bases de datos relacionales y no relacionales, y las características que deben tener las bases de datos para Android, **se decidió por SQLite**.

*SQLite* permite que el desarrollo sea más simple. Los paquetes `android.database` y `android.database.sqlite` ofrecen una alternativa de mayor rendimiento donde la compatibilidad de la fuente no representa mayor problema, aprovechando los recursos. Es ideal para consultar y almacenar datos de forma estructurada. La aplicación solo tiene que cargar tantos datos como necesite, en lugar de leer todo el archivo de la aplicación y mantener un análisis completo en la memoria, por ende, el tiempo de inicio y el consumo de memoria se reducen. El contenido se actualiza de forma continua y atómica, para que no se pierda el trabajo en caso de una falla de energía o algún bloqueo. Se puede acceder al contenido y actualizarlo mediante potentes consultas SQL, lo que reduce en gran medida la complejidad del código de la aplicación. Además, una gran cantidad de programas, escritos en diferentes lenguajes de programación, puede acceder al mismo archivo de aplicación sin problemas de compatibilidad.

# Capítulo 2

## Producto

Diseñada con el principal objetivo de crear y gestionar de manera sencilla e intuitiva historias clínicas veterinarias digitales de las mascotas en casa, la aplicación **HCVet** funciona tanto de forma local independiente (*offline*) como con conexión a un servidor utilizando internet. Teniendo como “único inconveniente” que se necesite una vez instalada la app en el móvil, conexión a internet para efectuar el registro en el servidor, la app fue pensada para que en todo momento el usuario tuviera el control total de los datos de sus mascotas. A través de diferentes vistas para distintos tipos de consulta el usuario puede introducir los datos y visualizarlos cuando desee, así, como eliminarlos cuando crea conveniente. Para esto, la app replica en una base de datos local los datos de sus mascotas que se sincronizan con los del servidor cuando exista una conexión a internet. La aplicación también permite el compartir mascotas vía wifi con otras personas que tengan instalada la aplicación a las que llamaremos “encargados”. Los encargados pueden almacenar en su base de datos local, y por ende visualizar y crear nuevas consultas, las mascotas a las cuales el dueño le ha compartido. Todo este proceso de sincronización se hará de forma automática o manual utilizando la conexión al servidor. Cabe señalar que una vez que el usuario se registre en el servidor podrá crear dos mascotas de forma gratuita, en caso de necesitar más deberá realizar una suscripción.

Entre las principales funcionalidades que ofrece la app se encuentran:

- **Creación de una mascota:** el usuario podrá crear una nueva mascota en la aplicación; para esto tendrá que llenar los campos de información básica de una mascota entre los que se encuentran: nombre de la mascota, fecha de nacimiento, raza, especie, tipo de sangre, entre otros.
- **Eliminar mascota:** el usuario podrá eliminar una mascota de su aplicación. Cabe señalar que este es un proceso peligroso puesto que una vez realizado no se podrán recuperar los datos de la mascota eliminada. Si la mascota es propia

se elimina tanto de su base de datos local como la que está almacenada en el servidor, en caso de ser una mascota que no es suya, solo se eliminará de su base de datos local y se notificará al servidor que ya no es encargado de dicha mascota.

- **Compartir/Recibir mascota:** el usuario podrá compartir/recibir los datos de una mascota (incluyendo todas las consultas de esta mascota) vía wifi. Una vez que exista conexión a internet se le notificará al servidor que el usuario que recibió la mascota ahora es un encargado de esta y por ende, el servidor se encargará de mantener sincronizados los datos de dicha mascota en ambos teléfonos (en el del dueño y en el del encargado).
- **Insertar nueva consulta:** el usuario podrá rellenar los campos de información para la creación de una nueva consulta. Entre las posibles consultas que ofrece la aplicación se encuentran las siguientes: Visitas Médicas, Pruebas de Laboratorio, Radiologías, Patologías, Cirugías y Prescripciones.
- **Insertar nueva Alergia/Condición/Vacuna:** el usuario podrá también llevar en la aplicación un registro de las vacunas, las alergias y las condiciones o enfermedades que se han realizado o diagnosticado a sus mascotas en dependencia del caso.
- **Insertar notas extras:** el usuario podrá insertar notas extras sobre las mascotas.
- **Visualización:** el usuario podrá visualizar de manera sencilla y organizada, ordenando por fecha, cada una de las consultas, así como las alergias, las vacunas, las condiciones, las notas extras, y los datos básicos de las mascotas. Teniendo la posibilidad de filtrar en cada caso.

## Capítulo 3

# Propuesta de Arquitectura y Modelo de Base de Datos

### 3.0.1. Arquitectura propuesta

**Model-View-ViewModel (MVVM)** es un patrón de arquitectura que surgió como alternativa a los patrones *Model View Controller* y *Model View Presenter* [13] que tiene el objetivo para llevar a cabo la separación del apartado de la interfaz de usuario (*View*) de la parte lógica (*Model*). Esto lo hace con el objetivo de que el aspecto visual sea completamente independiente.

El recurso de *ViewModel*, por su lado, destaca como el componente que se encargará de servir como puente entre la interacción de la Vista (*View*) y el Modelo (*Model*).

- **View:** este componente representa la interfaz de usuario y se conforma de un grupo de *Pages*<sup>1</sup> que actúan como interfaz de los servicios que ofrece la aplicación. Cada *Page* está compuesta por *Widgets*<sup>2</sup> con los que el usuario interactúa.
- **Model:** este componente representa la lógica de la aplicación y se conforma por los componentes que refieren a los servicios de comunicación con el servidor, almacenamiento interno de datos y comunicación con otro dispositivo.
- **ViewModel:** este componente está conformado por un grupo de interfaces que establecen una comunicación entre la Vista y el Modelo.

Los autores consideraron que el patrón *MVVM* aportaba el beneficio de permitir crear pruebas unitarias para el *ViewModel* y el *Model* sin que fuera necesario el uso

---

<sup>1</sup>En *Flutter* un *Page* es una pantalla que es visible en un momento dado.

<sup>2</sup>Los *widgets* componentes utilizados para crear la interfaz de usuario de la aplicación .



del *View*. También aportaba el beneficio de poder trabajar en el diseño y desarrollo de la aplicación de manera independiente y simultánea.

### 3.0.2. Modelo de Datos

Para persistir los datos necesarios en el cumplimiento de los requerimientos de la aplicación, es fundamental el empleo de una base de datos. Con esto en mente, se consideró el uso de una *base de datos relacional*, la cual se elaboró teniendo en cuenta el buen diseño del modelo de datos para una ampliación del mismo de manera sencilla (ver Figura 3,1). El modelo diseñado consta de **quince entidades** las cuales son descritas a continuación:

**Allergies**, **Drug**, **Vaccines** y **Disease** son las tablas que contienen las definiciones por defecto de las alergias, los medicamentos, las vacunas y las enfermedades respectivamente:

- *id* ( $PK^3$ ): Valor entero que identifica de forma única cada una de las filas de estas tablas.
- *name*: Texto utilizado para nombrar una alergia, un medicamento, una vacuna o una enfermedad dependiendo del caso.

**Pets** es la tabla que contiene la información básica de las mascotas:

- *idPet* ( $PK$ ): Valor entero que identifica de forma única una fila de esta tabla y por ende a una mascota.
- *idPerson*: Texto utilizado para identificar de forma única al dueño de la mascota.
- *name*: Texto utilizado para almacenar el nombre de una mascota.
- *date*: Texto utilizado para almacenar la fecha de nacimiento de una mascota.
- *species*: Texto que contiene la especie a la que pertenece una mascota.
- *race*: Texto que contiene la raza a la que pertenece una mascota.
- *gender*: Texto que contiene el género de una mascota.
- *bloodType*: Texto que representa el tipo de sangre de una mascota.
- *own* : Valor entero utilizado para identificar si una mascota es propia o no.

---

<sup>3</sup>Llave primaria, *Primary Key* por sus siglas en inglés.

- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una mascota. Puede tener los siguientes valores:
  - *empty*: significa que una mascota fue creada pero sus datos aún no se han llenado.
  - *delete*: significa que la mascota ha sido eliminada pero aún no se ha notificado al servidor.
  - *waiting*: significa que se han llenado los campos de la mascota, pero aún no se ha notificado al servidor.
  - *synchro*: significa que ya se ha notificado al servidor los cambios referentes a la mascota.

**MedicalVisit** es la tabla que contiene la información de las visitas médicas:

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*<sup>4</sup>): Valor entero que identifica de forma única una fila de la tabla *Pets*.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una visita médica en la aplicación.
- *date*: Texto utilizado para almacenar la fecha de una visita médica.
- *place*: Texto que representa el lugar en el que se realizó una visita médica.
- *doctor*: Texto que contiene el nombre del médico o especialista que realizo una visita médica.
- *notes*: Texto utilizado para almacenar notas extras sobre una visita médica.
- *state*<sup>5</sup>: Texto utilizado por la base de datos para identificar en cual estado se encuentra una visita médica.

**LabTests** es la tabla que se utiliza para almacenar las pruebas de laboratorio:

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.

---

<sup>4</sup>Llave foránea, *Foreign Key* por sus siglas en inglés.

<sup>5</sup>Puede ser *waiting* o *synchro*.

- *idPerson*: Texto que identifica de forma única a la persona que insertó una prueba de laboratorio en la aplicación.
- *date*: Texto utilizado para almacenar la fecha en la que se realizó una prueba de laboratorio.
- *place*: Texto en el que se guarda el lugar en el cual se realizó una prueba de laboratorio.
- *doctor*: Texto que almacena el nombre del especialista que realizó una prueba de laboratorio.
- *test*: Texto que identifica el test realizado en una prueba de laboratorio.
- *result*: Texto que almacena el resultado de una prueba de laboratorio.
- *normal* : Valor entero que representa si una prueba de laboratorio fue normal o no.
- *notes*: Texto utilizado para almacenar notas extras sobre una prueba de laboratorio.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una prueba de laboratorio.

**Prescription** es la tabla que almacena las prescripciones:

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.
- *idMedicament* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Drug*.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una prescripción en la aplicación.
- *dose*: Texto que representa la dosis del medicamento recetado en una prescripción.
- *date*: Texto que almacena la fecha en la que se realizó una prescripción.
- *place*: Texto que representa el lugar donde se realizó una prescripción.

- *doctor*: Texto que contiene el nombre del especialista que realizó una prescripción.
- *notes*: Texto utilizado para almacenar notas extras sobre una prescripción.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una prescripción.

**Allergy** es la tabla que almacena las alergias de las mascotas.

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.
- *idAllergy* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Allergies*.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una alergia en la aplicación.
- *date*: Texto utilizado para almacenar la fecha en la que se detectó una alergia.
- *notes*: Texto utilizado para almacenar notas extras sobre la detección de una alergia.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una alergia.

**Condition** es la tabla para almacenar las condiciones o enfermedades que tienen las mascotas:

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.
- *idDisease* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Disease*.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una condición en la aplicación.
- *name*: Texto utilizado para guardar el nombre del diagnóstico de una condición.

- *date*: Texto utilizado para almacenar la fecha en la que se diagnosticó una condición.
- *place*: Texto en el que se guarda el lugar en el cual se diagnosticó una condición.
- *doctor*: Texto que almacena el nombre del especialista que diagnosticó una condición.
- *notes*: Texto utilizado para almacenar notas extras sobre el diagnóstico de una condición.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra el diagnóstico de una condición.

**Vaccine** es la tabla que almacena las vacunas de las mascotas.

- *uid* (*PK*): Texto que identifica de forma única una fila de esta tabla.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.
- *idVaccine* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Vaccines*.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una vacuna en la aplicación.
- *date*: Texto utilizado para almacenar la fecha en la que se realizó una vacuna.
- *place*: Texto que representa el lugar donde se realizó una vacuna.
- *doctor*: Texto que contiene el nombre del especialista que realizó una vacuna.
- *notes*: Texto utilizado para almacenar notas extras sobre la realización de una vacuna.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una vacuna.

**Radiology**, **Pathology** y **Surgery** son las tablas utilizadas para guardar las radiologías, las patologías y las cirugías respectivamente.

- *uid* (*PK*): Texto que identifica de forma única una fila de estas tablas.
- *idPet* (*FK*): Valor entero que identifica de forma única una fila de la tabla *Pets*.

- *idPerson*: Texto que identifica de forma única a la persona que insertó en la aplicación una radiología, una patología o una cirugía en dependencia del caso.
- *date*: Texto utilizado para almacenar la fecha de una radiología, una patología o una cirugía en dependencia del caso.
- *title*: Texto utilizado para guardar el título de una radiología, una patología o una cirugía en dependencia del caso.
- *result*: Texto que contiene el resultado de una radiología, una patología o una cirugía en dependencia del caso.
- *place*: Texto que representa el lugar en el que se realizó una radiología, una patología o una cirugía en dependencia del caso.
- *doctor*: Texto que contiene el nombre del médico o especialista que realizó una radiología, una patología o una cirugía en dependencia del caso.
- *notes*: Texto utilizado para almacenar notas extras sobre una radiología, una patología o una cirugía en dependencia del caso.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una radiología, una patología o una cirugía en dependencia del caso.

**Notes** es la tabla que contiene notas extras sobre las mascotas.

- *uid (PK)*: Texto que identifica de forma única una fila de esta tabla.
- *idPet (FK)*: Valor entero que identifica de forma única una fila de la tabla Pets.
- *idPerson*: Texto que identifica de forma única a la persona que insertó una nota extra en la aplicación.
- *title*: Texto que representa el título de una nota extra.
- *notes*: Texto utilizado para almacenar una nota extra.
- *date*: Texto utilizado para almacenar la fecha de una nota extra.
- *place*: Texto en el que se guarda el lugar en el cual se escribió una nota extra.
- *doctor*: Texto que almacena el nombre del especialista que escribió una nota extra.
- *state*: Texto utilizado por la base de datos para identificar en cual estado se encuentra una nota extra.

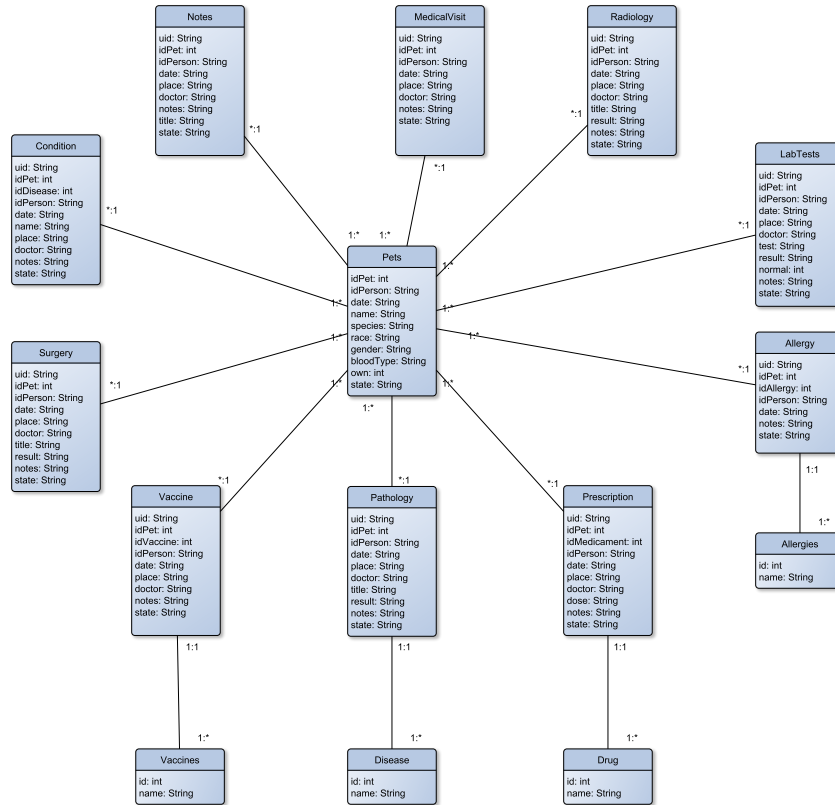


Figura 3.1: Representación del modelo de la base de datos

Para la creación y acceso a la clase `DatabaseHandler`; clase encargada de manejar las solicitudes a la base de datos en nuestra aplicación, se utilizó como patrón de diseño singleton.

**Singleton** [18] es un patrón de diseño creacional que consiste en tener durante toda la ejecución de un programa sólo una única instancia de una clase dada; proveyendo además un punto de acceso global a dicha instancia. Esto se logra entre otras cosas haciendo privado el método constructor de la clase y ofreciendo acceso a esta a través de un método especial de tipo `get`. Dicho método es el encargado de verificar si existe creada una instancia de esa clase y devolverla o crearla dependiendo del caso. De esta forma se asegura la propia clase de haber sido creada una única vez y por tanto tener una única instancia (singleton).

# Capítulo 4

## Detalles de Implementación

### 4.1. Tecnologías utilizadas en HCVet

#### 4.1.1. Dart

Para la implementación de la aplicación se empleó como núcleo principal *Dart* [10]. *Dart* es un lenguaje optimizado para el desarrollo rápido de aplicaciones en cualquier plataforma, su objetivo es ofrecer el lenguaje de programación más productivo para el desarrollo multiplataforma. *Dart* es un lenguaje orientado a objetos, basado en clases, con una sintaxis similar a *C* y con *GarbageCollector*<sup>1</sup>. Admite interfaces, genericidad<sup>2</sup>, mixins<sup>3</sup>, clases abstractas, e inferencia de tipos.

*Dart* también es la fundación de *Flutter* proveyendo el lenguaje y los tiempos de ejecución que caracterizan a *Flutter*.

#### 4.1.2. Kotlin

Para la implementación de algunas funcionalidades de la aplicación se utilizó *Kotlin* [28]. *Kotlin* es el lenguaje oficial para desarrollo de aplicaciones en *Android* declarado por *Google* en el 2019. *Kotlin* es un lenguaje de programación multiplataforma que remueve detalles superfluos de *java* como *nullpointerexceptions*<sup>4</sup> y es un lenguaje más simple y práctico en comparación con *java*.

---

<sup>1</sup>Garbage Collector o Recolector de basura es un mecanismo implícito de gestión de memoria implementado en algunos lenguajes de programación de tipo interpretado o semi-interpretado.

<sup>2</sup>Es una propiedad que permite definir una clase o función sin especificar el tipo de datos de uno o mas de sus parámetros

<sup>3</sup>Es una clase que contiene métodos para ser usados por otras clases sin tener que ser la clase padre de esas otras clases.

<sup>4</sup>*nullpointerexceptions* es una excepción que ocurre cuando una variable es accedida en ejecución y esta no está apuntando a ningún objeto



### 4.1.3. Paquete http

**http** es una biblioteca de *Flutter* basada en **Future**<sup>5</sup> para hacer **requests** de *HTTP*.

Esta biblioteca es multiplataforma que permite su uso en dispositivos de escritorio, dispositivos móviles y en navegadores web. Contiene una variedad de funciones y clases de alto nivel que facilitan el consumo de recursos *HTTP*.

**http** [34] es uno de los paquetes más aclamados de *Flutter* que se encuentran disponible en *pub.dev*.

### 4.1.4. Paquete uuid

Este complemento para *Flutter* se encarga del análisis y la generación simple y rápida de **UUID**<sup>6</sup> RFC4122. Entre las posibles opciones de obtención de uuid la que se utilizó para las llaves de las tablas fue la basada en el tiempo [36].

### 4.1.5. Paquete encrypt

Esta biblioteca contiene un conjunto de APIs de alto nivel sobre *Pointycastle*<sup>7</sup> para criptografía simétrica. Facilita la generación de llaves aleatorias seguras y Vectores de Inicialización.

### 4.1.6. Paquete sqfentity

**SQLite**[39] es una biblioteca en lenguaje *C* que implementa un motor de base de datos *SQL* pequeño, rápido, autónomo, de alta confiabilidad y con todas las funciones. El formato de archivo que utiliza es estable, multiplataforma y compatible con versiones anteriores.

Entonces llegamos a **sqfentity**[35], un *ORM*<sup>8</sup> para *Flutter SQLite*. Se basa en el paquete *sqflite*, para la misma plataforma, permitiendo compilar y ejecutar comandos *SQL* de manera fácil y rápida con la ayuda de métodos fluidos similares a *Entity Framework* de *.Net*<sup>9</sup>.

---

<sup>5</sup>Una instancia de la clase **Future** representa el resultado de una operación asíncrona

<sup>6</sup>Identificador Único Universal

<sup>7</sup>*Pointycastle* es una biblioteca de *Dart* para encriptación y desencriptación.

<sup>8</sup>Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

<sup>9</sup>El Entity Framework es un conjunto de tecnologías de ADO.NET que admiten el desarrollo de aplicaciones de software orientadas a datos

## 4.2. Detalles de la Implementación

### 4.2.1. Estructura de Model

En el Modelo de la aplicación se buscó proporcionar todas funcionalidades deseadas del producto de la forma más legible y escalable a futuro que fuera posible. Con este objetivo los autores tomaron el patrón de arquitectura de micro-servicios [22], se dividió el modelo en 3 componentes de servicio:

- Componente de comunicación con el servidor (*Online*)
- Componente de transferencia de datos no sincronizada sin conexión a internet. (*Offline*)
- Componente de almacenamiento interno (*Database*)

Con los componentes de servicio definidos se emplean las interfaces del *ViewModel* para acceder a estos servicios. Consideramos que este diseño permitía crear un modelo escalable horizontalmente <sup>10</sup> además de facilitar el mantenimiento y actualización de dichos componentes.

### 4.2.2. Implementación de la transferencia de datos no sincronizada

Uno de los requerimientos fundamentales del producto era lograr la compatibilidad con la mayor cantidad de dispositivos móviles que fuera posible, intentando evitar la necesidad de adquirir un dispositivo móvil moderno para poder utilizar el producto.

Durante la investigación realizada para satisfacer este requisito los autores encontraron un problema: los paquetes de *Flutter* encontrados que permiten el manejo las conexiones entre dispositivos a través de *Wifi* restringían el uso de diferentes *APIs* como la creación de *Hotspots* locales para el *SDK* de *Android* mayor o igual que 26 <sup>11</sup> [37].

Para suplir esta necesidad se utilizó el *API WifiP2pManager* de *Android*. Esta *API* permite a la aplicación descubrir *peers*<sup>12</sup> disponibles y permite establecer una conexión directa con estos. Usando *Kotlin* se definió el comportamiento de los *peers* y se creó una interfaz para acceder a las funcionalidades de esta *API*.

---

<sup>10</sup>El escalado horizontal refiere a la adición de componentes adicionales para hacer frente a nuevas demandas

<sup>11</sup>*Android SDK* 26 refiere al sistema operativo *Android* 8,0

<sup>12</sup>*Peer* en un contexto de *network* es un nodo que cumple la misma funcionalidad que otro nodo en la red. Cualquier usuario que se conecta a la red de intercambio es un *peer*

### ¿Cómo se usa esta interfaz desde *Flutter*?

*Flutter* permite realizar llamados a *APIs* específicas de la plataforma disponibles en *Java* o *Kotlin* en *Android* y en *Objective C* o *Swift* en *iOS*

Desde la aplicación de *Flutter* se envía un mensaje a un *host* en *Android* o *iOS* a través de un canal entre plataformas. Tanto el mensaje como la respuesta se pasan de forma asíncrona.

#### 4.2.3. Implementación del componente Online

La comunicación con el servidor está basada en un protocolo de tipo *Hypertext transfer protocol secure (HTTPS)* de *request – response*. Se consideró la encriptación de los datos en ambas direcciones que ofrece el protocolo *https* para evitar la vulneración de datos sensibles transferidos como pueden ser el *id* del usuario o la mascota, correo y contraseña.

#### 4.2.4. Sincronización de la base de datos local con el servidor

Dentro de las problemáticas encontradas durante el proceso de sincronización de la base de datos local con la del servidor se pudieran mencionar las siguientes:

1. **Diferenciar los datos:** en cualquier momento dado en la base de datos local existirán datos que ya el servidor tendrá conocimiento de ellos y por tanto se puede decir que están sincronizados con el servidor y otros datos de nueva creación que hasta que no exista una conexión a internet, el servidor desconocerá por completo de su existencia y por tanto estarán pendientes a sincronizar. Diferenciar cuales de estos datos ya están sincronizados y cuales no, resulta un poco complejo de manejar si no se utiliza algún mecanismo de “control” sobre ellos. Todo esto para intentar reducir el número de envío de información al servidor por cuestiones de seguridad y rapidez, puesto que bien se podrían enviar todos los datos de la base de datos local al servidor cada vez que exista una conexión, pero esta opción además de no factible resulta lamentable. Para solucionar dicha problemática se optó por agregar en cada tabla de la base de datos un campo extra llamado “*state*”. Dicho campo tendrá el valor “*waiting*” cuando es un dato nuevo y el valor “*synchro*” cuando ya ha sido sincronizado con el servidor. Cuando los datos son creados desde el propio teléfono la función de insertar elementos en la base de datos declara el campo *state* de dicho objeto como *waiting*, sin embargo, cuando los datos provienen de otro móvil o incluso del propio servidor se declara el *state* como *synchro* puesto que no es necesario re sincronizar estos datos.

2. **Guardar llaves de futuras mascotas:** cuando el usuario se registra en el servidor y/o realiza algún tipo de suscripción, el servidor manda una lista de futuras llaves para la creación de nuevas mascotas. Para no tener que implementar una nueva tabla que tan solo guardase estas llaves o bien almacenarlas en un fichero apartado de la base de datos, se optó por almacenar en la tabla *Pets* una mascota “vacía”. Dicha mascota tendrá en sus campos solo esta llave como llave primaria y el *state* tendrá el valor “*empty*” indicando de este modo que la mascota aun no existe en la base de datos. Cuando se crea una nueva mascota que tiene alguna de estas llaves como llave primaria se actualizan los campos de la mascota ya “previamente” insertada en la base de datos y su *state* cambia a “*waiting*”.
3. **Guardar mascotas eliminadas:** aunque pareciese una contradicción hay momentos en la que nuestra base de datos necesita mantener guardada aun una mascota que ha sido eliminada por el usuario. Como nuestra base de datos debe ser sincronizada con la existente en el servidor no basta con eliminar una mascota directamente de la base de datos cuando el usuario requiera esta operación puesto que, si la mascota deja de existir, luego cuando exista una conexión a internet no habrá manera de indicar al servidor que debe eliminar dicha mascota de su base de datos también. Para resolver esta problemática se hace uso nuevamente del campo *state* en mascota. Cuando el usuario manda a eliminar una mascota de la aplicación, se actualiza el *state* de dicha mascota a valor “*delete*” indicando así al resto de los métodos de la base de datos que esa mascota, aunque está insertada no cuenta como una mascota válida (este funcionamiento se extiende también a todas las tablas que tienen como llave foránea un id de dicha mascota). Luego de que se indique al servidor que debe eliminar dicha mascota se eliminará también de la base de datos local.

## Capítulo 5

# Pruebas de Funcionalidad y Experimentos

En este capítulo se realizan un conjunto de pruebas para demostrar el funcionamiento de las herramientas implementadas en la aplicación.

Las pruebas fueron realizadas en dos teléfonos *android* con las siguientes características.

Primer teléfono:

- Procesador: Hisilicon Kirin 710*F*.
- 4 *GB* de *RAM*.
- Sistema Operativo Android 9

Segundo teléfono:

- Procesador: Octa-core Max 2,00*GHz*.
- 4 *GB* de *RAM*.
- Sistema Operativo Android 12

## 5.1. Prueba 1: Creación de cuenta o autenticación de usuario

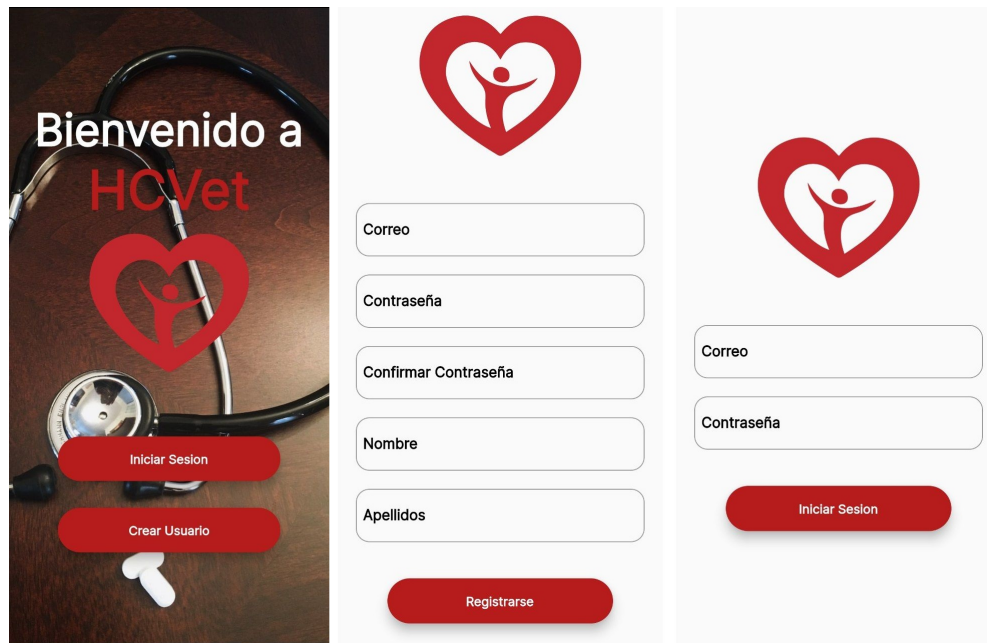


Figura 5.1: De izquierda a derecha: Página inicial de la aplicación, página de creación de usuario y página de autenticación

Se requiere que el usuario introduzca un nombre de usuario y contraseña para crear una cuenta en el servidor. La aplicación también utiliza el número de teléfono del dispositivo para identificar al usuario. En caso de que el número de teléfono o el correo introducido ya se encuentre registrado la aplicación muestra un error y espera una corrección por parte del usuario.

## 5.2. Prueba 2: Creación de mascota

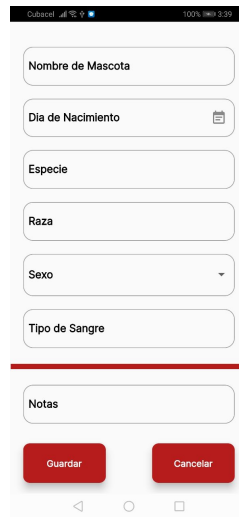
El formulario de creación de mascota se muestra en un dispositivo móvil. Incluye campos de texto para 'Nombre de Mascota', 'Especie', 'Raza' y 'Tipo de Sangre'. Un campo para 'Día de Nacimiento' tiene un icono de calendario. Un menú desplegable para 'Sexo' está actualmente cerrado. Debajo de estos campos, hay un campo de texto para 'Notas'. En la parte inferior del formulario, hay dos botones rojos: 'Guardar' y 'Cancelar'. El fondo del formulario es gris claro con una barra roja horizontal que separa los campos de entrada de las notas y los botones.

Figura 5.2: Página de creación de mascotas

La creación de una mascota requiere que el usuario introduzca todos los campos que se listan a excepción de *notas*. Solo se permite la creación de una mascota si el usuario tiene un *id* disponible en la base de datos local.

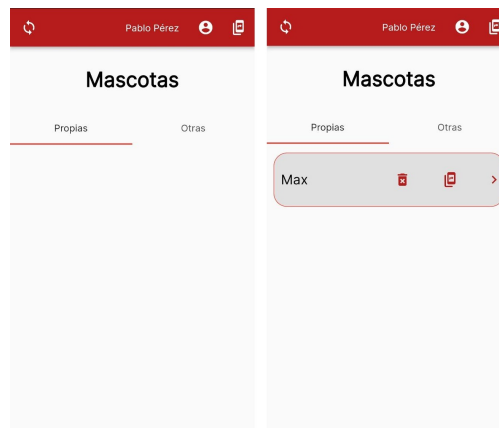


Figura 5.3: Página de central de la aplicación antes y después de crear una mascota

Página central de la aplicación antes y después de crear una mascota

### 5.3. Prueba 3: Eliminar mascota

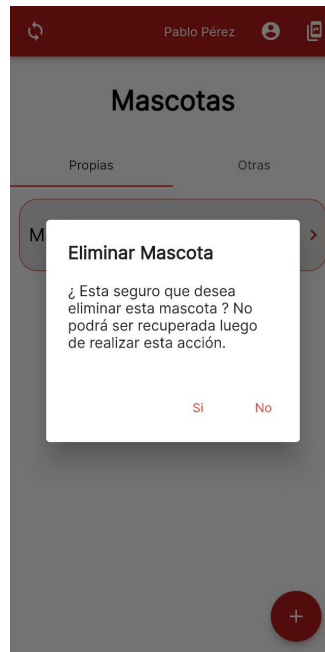


Figura 5.4: Advertencia al eliminar una mascota

Si el usuario decide eliminar una mascota los datos e historial de consultas de estas no podrán ser recuperados. Esta advertencia es mostrada ante el usuario, si escoge eliminar la mascota el servidor será notificado de esta decisión la próxima vez que se sincronice.



## 5.4. Prueba 4: Creación y mostrado de consultas

The figure displays two versions of the 'Visita Médica' (Medical Visit) form. The left form is for creating a new visit, featuring a dropdown for 'Tipo de Visita', text inputs for 'Hospital', 'Doctor', and 'Fecha y Hora' (with a calendar icon), a text area for 'Notas', and 'Guardar' (Save) and 'Cancelar' (Cancel) buttons. The right form shows a completed visit with pre-filled data: 'Hospital de Ejemplo', 'Doctor de Ejemplo', the date and time '11/17/2022 3:42 AM', and the note 'ejemplo'.

Figura 5.5: Página de creación de consulta Visita Médica y página de mostrado de esta consulta

La creación de una consulta solo requiere un dato fundamental, la fecha. Las consultas de Prescripción, Vacuna, Condición y Alergia requieren que se seleccione una de las llaves foráneas de la respectiva consulta. El usuario es notificado si uno de estos requerimientos no se cumple.

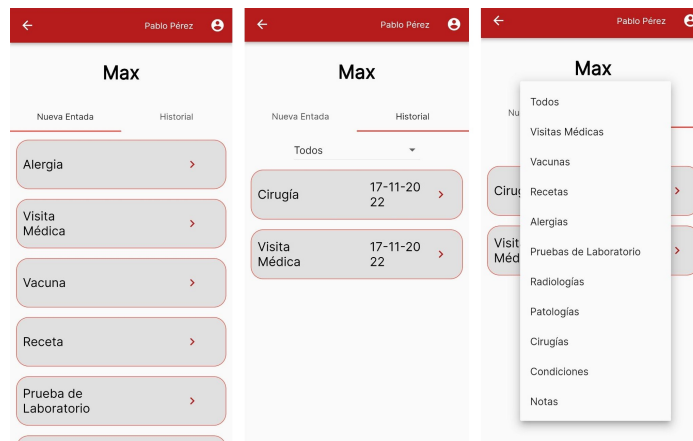


Figura 5.6: De izquierda a derecha: Página de selección de consulta a agregar, historial de consultas y filtrado de consultas en historial

Con una mascota seleccionada se dirige al usuario a la página de manejo de consultas, el usuario puede crear una consulta nueva o acceder a todas las consultas que se encuentran almacenadas en la base de datos local.

# Conclusiones

Controlar la información medica de una mascota es un tema de suma importancia, que es llevado a cabo en su mayoría de manera poca eficiente y raramente digital. Durante el proceso investigativo que los autores emplearon en la realización de este trabajo de diploma, encontraron clara la necesidad de la existencia de un programa que permitiera el almacenamiento, organización y distribución de datos en el sector veterinario. Con el desarrollo de HCVet, se le dio cumplimiento a los objetivos trazados en el capítulo 1. La aplicación construida es capaz de mostrar información contenida en la historia clínica veterinaria de las mascotas del usuario sin necesidad de una conexión a Internet, así como añadir nueva información por medio de formularios contruidos de forma dinámica a partir de distintas categorías. Siendo también posible compartir información entre usuarios utilizando la wifi. En este trabajo, se muestra el diseño de la base de datos utilizada, y algunos de los problemas surgidos durante la implementación, así como los mecanismos utilizados para dar solución a estas problemáticas. La aplicación desarrollada muestra un gran potencial que soluciona no solo los problemas planteados, sino la habilidad para expandirse a otros campos u objetivos.

# Opinión del tutor

Opiniones de los tutores

# Recomendaciones

Aunque consideramos que HCVet cumple con todas las características y funcionalidades propuestas no debe descartarse la opción de que en futuras versiones de la aplicación se incorporen nuevas funcionalidades que no contemplamos en esta versión actual. Entre algunas de las posibles acciones a realizar en un futuro con el objetivo de ampliar los casos de uso de la aplicación se proponen las siguientes ideas:

1. Investigar sobre otros tipos de consultas que pudieran ser añadidas al modelo de la aplicación.
2. Diferenciar entre distintos tipos de usuarios (dueño, veterinario, estilista, etc.) y establecer cuáles serían sus permisos sobre los datos de la aplicación.
3. Añadir la opción de modificar los datos de una mascota y/o los de una consulta previamente existente.
4. Implementar un sistema de citas controlado por el servidor para que los médicos veterinarios tengan un espacio para coordinar las consultas con los clientes.
5. Implementar un sistema de avisos y alarmas que indiquen al usuario sobre las visitas pre-programadas al veterinario y/o el horario de suministrarle una medicina a su mascota.

# Bibliografía

- [1] *Aplicación móvil promueve labor de bienestar animal en Cuba*. [Online; accessed 12. Sep. 2022]. Sep. de 2022. URL: <https://www.ipscuba.net/sociedad/aplicacion-movil-promueve-labor-de-bienestar-animal-en-cuba/> (vid. pág. 12).
- [2] Agustín Blasco y Agustín Blasco Mateu. *Ética y bienestar animal*. Ediciones Akal, 2011 (vid. pág. 2).
- [3] Donald M Broom y col. «Animal welfare in the European Union.» En: *Animal welfare in the European Union*. (2017) (vid. pág. 2).
- [4] *Conoce Pet Soft: la mejor app para tu clínica veterinaria (Sitio Oficial de Pet Soft)*. [Online; accessed 13. Sep. 2022]. Sep. de 2022. URL: <https://www.petsoft.com.co/blog/nwarticle/48/25/conoce-petsoft-la-mejor-app-para-tu-clinica-veterinaria> (vid. pág. 14).
- [5] *Cuba: disponible aplicación móvil sobre bienestar animal*. [Online; accessed 11. Sep. 2022]. Sep. de 2022. URL: <https://oncubanews.com/cuba/cuba-disponible-aplicacion-movil-sobre-bienestar-animal/> (vid. pág. 12).
- [6] Katiuska Cuenca Garcell y col. «La historia clínica estomatológica como herramienta en el método clínico y documento médico-legal». En: *Revista Cubana de Medicina Militar* 43.4 (2014), págs. 534-540 (vid. pág. 8).
- [7] Marian Stamp Dawkins. «Animal welfare and efficient farming: is conflict inevitable?» En: *Animal Production Science* 57.2 (2016), págs. 201-208 (vid. pág. 2).
- [8] Iván Peña García y Florangel Vidal Fernández. «Historia Clínica Veterinaria Informatizada (Software HisCliVet)». En: *REDVET. Revista Electrónica de Veterinaria* 7.10 (2006), págs. 1-19 (vid. págs. 9, 10).
- [9] Teresa Giménez-Candela. «Animales y Derecho en una sociedad global». En: *dA. Derecho Animal. Forum of Animal Law Studies*. Vol. 5. 3. 2014, págs. 1-3 (vid. pág. 3).

- [10] *Google's Dart language on Android aims for Java-free, 120 FPS apps*. [Online; accessed 5. Mar. 2022]. Mar. de 2022. URL: <https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/> (vid. págs. 23, 41).
- [11] *Inscripción y Registro de Mascotas en Cuba*. [Online; accessed 12. Sep. 2022]. Sep. de 2022. URL: <https://www.cubatramite.com/inscripcion-y-registro-de-mascotas-en-cuba/> (vid. pág. 12).
- [12] Mylene Jarrin Peña, Cindy Marcela Avendaño Peralta, Cesar Mauro Martínez Jaramillo y col. «Diseño de una aplicación móvil para el mejoramiento de los servicios médicos veterinarios en el municipio de Cereté». En: (2021) (vid. pág. 4).
- [13] John Kouraklis. «MVVM as Design Pattern». En: oct. de 2016. ISBN: 978-1-4842-2213-3. DOI: 10.1007/978-1-4842-2214-0\_1 (vid. pág. 33).
- [14] Xavier Manteca, D Mainau y D Temple. «Bienestar animal». En: *Manual de Buenas Prácticas de Producción Porcina. Lineamientos generales para el pequeño y mediano productor de cerdos. Red Porcina Iberoamericana 2012* (2012), págs. 97-111 (vid. pág. 2).
- [15] Redacción Minsap. «La voluntad de garantizar el bienestar animal en Cuba». En: *Sitio oficial de gobierno del Ministerio de Salud Pública en Cuba* (feb. de 2022). URL: <https://salud.msp.gob.cu/la-voluntad-de-garantizar-el-bienestar-animal-en-cuba> (vid. pág. 4).
- [16] Amado Antonio García Odio y Daymeris Álvarez Bolívar. «La historia clínica hospitalaria como instrumento médico, legal y administrativo en el nuevo contexto constitucional». En: *Mediciego 25.1* (2019), págs. 79-81 (vid. pág. 8).
- [17] *Offline data transfer in Android*. [Online; accessed 1. sep. 2022]. Oct. de 2022. URL: <https://www.tothenew.com/blog/offline-data-transfer-in-android> (vid. pág. 17).
- [18] *Patrón de Diseño Singleton*. [Online; accessed 4. sep. 2022]. Sep. de 2022. URL: <https://refactoring.guru/es/design-patterns/singleton> (vid. pág. 40).
- [19] *Petmeddata becomes the worlds most comprehensive pets health database*. [Online; accessed 2. sep. 2022]. Sep. de 2022. URL: <https://www.provet.cloud/blog/blog/petmeddata-becomes-the-worlds-most-comprehensive-pets-health-database> (vid. pág. 15).
- [20] Christian E Rueda Clausen Pinzón. «La historia clínica informatizada. Evaluación de los casos colombiano y español». En: *MedUNAB 9.1* (2006), págs. 63-71 (vid. pág. 9).

- [21] *Página Oficial de Bienestar Animal en Cuba*. [Online; accessed 11. Sep. 2022]. Sep. de 2022. URL: <https://bacuba.org/quienes-somos/> (vid. pág. 12).
- [22] Mark Richards. «Software Architecture Patterns». En: 2015 (vid. pág. 43).
- [23] Liyanis Santana Santana, Ana María Pereda Mirabal y Annia Teresa Acosta Cardoso. «A propósito de la responsabilidad jurídica del médico». En: *Revista de Ciencias Médicas de Pinar del Río* 20.1 (2016), págs. 0-0 (vid. pág. 9).
- [24] *Sitio oficial de Apple developers for App Store Specific Considerations*. [Online; accessed 5. Jun. 2022]. Jun. de 2022. URL: [https://developer.apple.com/documentation/xcode/reducing-your-app-s-size#//apple\\_ref/doc/uid/DTS40014195-CH1-APP\\_STORE\\_CONSIDERATIONS](https://developer.apple.com/documentation/xcode/reducing-your-app-s-size#//apple_ref/doc/uid/DTS40014195-CH1-APP_STORE_CONSIDERATIONS) (vid. pág. 25).
- [25] *Sitio oficial de Couchbase*. [Online; accessed 1. sep. 2022]. Sep. de 2022. URL: <https://developer.couchbase.com> (vid. pág. 27).
- [26] *Sitio oficial de Flutter*. [Online; accessed 2. Mar. 2022]. Mar. de 2022. URL: <https://flutter.dev> (vid. pág. 23).
- [27] *Sitio oficial de Java*. [Online; accessed 1. sep. 2022]. Oct. de 2022. URL: <https://www.java.com/es> (vid. pág. 17).
- [28] *Sitio Oficial de Kotlin*. [Online; accessed 4. oct. 2022]. Oct. de 2022. URL: <https://kotlinlang.org> (vid. págs. 17, 41).
- [29] *Sitio oficial de .Net*. [Online; accessed 1. sep. 2022]. Sep. de 2022. URL: <https://dotnet.microsoft.com> (vid. pág. 21).
- [30] *Sitio Oficial de Petmeddata*. [Online; accessed 2. sep. 2022]. Sep. de 2022. URL: <https://petmeddata.com/es/> (vid. pág. 15).
- [31] *Sitio Oficial de VitusPet*. [Online; accessed 2. sep. 2022]. Sep. de 2022. URL: <https://appadvice.com/app/vitusvet-pet-medical-records/955252538> (vid. pág. 16).
- [32] *Sitio oficial de Xamarin*. [Online; accessed 1. sep. 2022]. Sep. de 2022. URL: <https://dotnet.microsoft.com/apps/xamarin> (vid. pág. 21).
- [33] *Sitio Oficial: Dog Health*. [Online; accessed 12. Sep. 2022]. Sep. de 2022. URL: <http://doghealthapp.it/> (vid. pág. 14).
- [34] *Sitio Oficial el paquete http para Flutter*. [Online; accessed 4. sep. 2022]. Sep. de 2022. URL: <https://pub.dev/packages/http> (vid. pág. 42).
- [35] *Sitio Oficial el paquete sqfentity para Flutter*. [Online; accessed 4. sep. 2022]. Sep. de 2022. URL: <https://pub.dev/packages/sqfentity> (vid. pág. 42).
- [36] *Sitio Oficial el paquete uuid para Flutter*. [Online; accessed 4. sep. 2022]. Sep. de 2022. URL: <https://pub.dev/packages/uuid> (vid. pág. 42).

- [37] *Sitio Oficial el paquete `wifi_iot` para Flutter*. [Online; accessed 2. sep. 2022]. Sep. de 2022. URL: [https://pub.dev/packages/wifi\\_iot](https://pub.dev/packages/wifi_iot) (vid. pág. 43).
- [38] *Sitio Oficial: Oracle Berkeley DB*. [Online; accessed 5. Jun. 2022]. Jun. de 2022. URL: <http://www.oracle.com/us/products/database/berkeley-db/index.html> (vid. pág. 26).
- [39] *Sitio oficial SQLite*. [Online; accessed 5. Jun. 2022]. Jun. de 2022. URL: <https://developer.couchbase.com/documentation/mobile/current/installation/index.html> (vid. págs. 28, 42).
- [40] *¿Chapter 1. What Is React Native?* [Online; accessed 2. Mar. 2022]. Mar. de 2022. URL: <https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html> (vid. pág. 19).
- [41] *¿Qué es Pet Soft?* [Online; accessed 11. Sep. 2022]. Sep. de 2022. URL: <https://www.petsoft.com.co/es-CU/> (vid. pág. 14).