UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios

LOAYZA AGILA LUIS ENRIQUE
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

VILLAMAR CRUZ MARLON GEOVANNY
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

MACHALA
2023

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios

LOAYZA AGILA LUIS ENRIQUE
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

VILLAMAR CRUZ MARLON GEOVANNY
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

MACHALA
2023

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

PROPUESTAS TECNOLÓGICAS

Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios

LOAYZA AGILA LUIS ENRIQUE
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

VILLAMAR CRUZ MARLON GEOVANNY
INGENIERO EN TECNOLOGIAS DE LA INFORMACION

CARTUCHE CALVA JOFFRE JEORWIN

MACHALA
2023

TRABAJO DE INTEGRACIÓN CURRICULAR

by Marlon Geovanny Villamar Cruz

Submission date: 02-Oct-2023 02:46PM (UTC-0500)
Submission ID: 1991415063
File name: Para_Turniting.pdf (2.23M)
Word count: 14400
Character count: 80880

TRABAJO DE INTEGRACIÓN CURRICULAR

ORIGINALITY REPORT

3%	3%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universidad Técnica de Machala Student Paper	1%
2	documentop.com Internet Source	1%
3	www.dropbox.com Internet Source	<1%
4	moam.info Internet Source	<1%
5	hdl.handle.net Internet Source	<1%
6	repositorio.utmachala.edu.ec Internet Source	<1%
7	Submitted to UNIBA Student Paper	<1%
8	www.slideshare.net Internet Source	<1%
9	aws.amazon.com Internet Source	<1%

10	www.eluniversal.com Internet Source	<1%
11	www.coursehero.com Internet Source	<1%
12	Submitted to Universidad Alfonso X el Sabio Student Paper	<1%
13	Submitted to Universidad Nacional Abierta y a Distancia, UNAD,UNAD Student Paper	<1%
14	revistasinvestigacion.unmsm.edu.pe Internet Source	<1%

Exclude quotes	On	Exclude matches	< 20 words
Exclude bibliography	On		

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

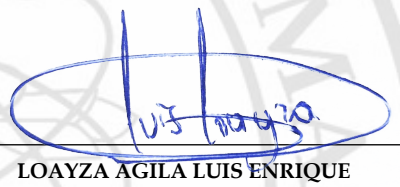
Los que suscriben, LOAYZA AGILA LUIS ENRIQUE y VILLAMAR CRUZ MARLON GEOVANNY, en calidad de autores del siguiente trabajo escrito titulado Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios, otorgan a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tienen potestad para otorgar los derechos contenidos en esta licencia.

Los autores declaran que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

Los autores como garantes de la autoría de la obra y en relación a la misma, declaran que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asumen la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.



LOAYZA AGILA LUIS ENRIQUE
0705832368

VILLAMAR CRUZ MARLON GEOVANNY
0751106485

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios

LOAYZA AGILA LUIS ENRIQUE
INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

VILLAMAR CRUZ MARLON GEOVANNY
INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

MACHALA

2023

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR.

PROPUESTA TECNOLÓGICAS

Creación de un sistema para la gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios

LOAYZA AGILA LUIS ENRIQUE

INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

VILLAMAR CRUZ MARLON GEOVANNY

INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

ING. CARTUCHE CALVA JOFFRE JEORWIN

MACHALA, 02 DE OCTUBRE 2023

MACHALA 2023

DECICATORIA

Este trabajo se lo dedico a mis queridos padres por el apoyo, cariño y sacrificio, han sido un pilar fundamental en mi proceso de formación personal y académica.

A mi hermana por estar presente en todo lo que he vivido y por su gran motivación para seguir adelante.

A mis familiares y mis amigos que siempre me brindaron su apoyo para continuar en esta etapa de vida.

Loayza Agila Luis Enrique

Dedico este trabajo en primer lugar a mis padres, quienes siempre me han apoyado incondicionalmente para seguir estudiando y trabajar duro por las cosas a las que aspiro lograr.

A mi hermana, quien ha sido una fuente constante de apoyo y compañía a lo largo de mi trayecto académico y personal.

A mis familiares y amigos que me han apoyado durante todo el proceso de mis estudios y me han ayudado en los momentos en que más los necesitaba.

Y dedico este trabajo de manera especial a mi querida abuelita Cupy (†), por su apoyo en la primera etapa de mi vida, brindándome no solo amor incondicional, sino también sabios consejos que han resonado a lo largo de los años y por inculcarme valores que se ha convertido en una guía en cada paso de mi trayectoria.

Villamar Cruz Marlon Geovanny

AGRADECIMIENTOS

Agradezco a mis queridos padres de todo corazón por el amor inquebrantable, apoyo incondicional, valores y sabiduría que me han brindado en mi vida, gracias a ellos he completado varios logros en lo personal y académico.

A mi hermana por el impulso motivacional, su cariño y darme la confianza necesaria para seguir adelante. Eres un regalo invaluable en mi vida.

A mi compañero de trabajo Marlon Geovanny Villamar Cruz por su dedicación y apoyo brindados en el desarrollo de este proyecto. Gracias por todo.

Al Ing. Cartuche Calva Joffre Jeorwin quien fue mi tutor. Por su experiencia, dedicación, orientación y apoyo brindados, su guía fue fundamental para lograr con éxito este proyecto.

Al Ing. Hernandez Dixys, mi cotutor, le agradezco por su contribución y orientación que fueron fundamentales en este proyecto.

Por último, a mis familiares y amigos los cuales me brindaron su apoyo y fortalezas tanto en lo personal como en lo académico.

Loayza Agila Luis Enrique

En primer lugar, quiero elevar mi agradecimiento a Dios por haberme brindado la sabiduría, la fuerza y la inspiración necesarias para llevar a cabo este trabajo.

A mi familia, quienes son el pilar de mi vida, expreso mi profundo agradecimiento. Su inquebrantable fe en mí y su apoyo constante que me han dado la confianza para enfrentar los desafíos y perseverar en este camino.

Al Ing. Cartuche Calva Joffre Jeorwin, mi tutor, le agradezco por su orientación y paciencia. su sabiduría ha sido una guía invaluable a lo largo de este proyecto.

Al Ing. Dixys Hernandez, mi cotutor, le agradezco su apoyo constante y su contribución a nuestras ideas y objetivos compartidos.

Un agradecimiento especial a mi compañero de trabajo, Luis Loayza, gracias a su colaboración constante y dedicación incansable. Juntos hemos logrado un gran éxito en este proyecto.

Villamar Cruz Marlon Geovanny

RESUMEN

La automatización de servicios en las organizaciones o empresas son aplicados con software los cuales tienen una arquitectura monolítica, la cual es muy usada para el desarrollo de sistemas de gestión; pero, al presentarse algún fallo interno obliga a quedar fuera de servicio en su totalidad. Debido a estos casos se opta como una solución implementar la arquitectura de microservicios. Actualmente, en el ámbito de la medicina veterinaria, la falta de herramientas tecnológicas dificulta la automatización y gestión eficiente de sus procesos en las clínicas, obligando a ejecutarlos de manera manual. El presente trabajo tiene el propósito de automatizar los procesos de una clínica veterinaria creando un aplicativo web y móvil utilizando una arquitectura basada en microservicios para un alto nivel de disponibilidad y escalabilidad. Para la elaboración de esta propuesta se eligió la metodología ágil DevOps. También se evaluó el sistema aplicando la norma ISO/IEC 25010 para asegurar un alto nivel de satisfacción en términos de funcionalidad, rendimiento, fiabilidad y portabilidad. Los resultados de la evaluación demostraron un alto nivel de satisfacción en la funcionalidad tanto del aplicativo web como móvil, respaldado por encuestas y un cumplimiento del 100% de funciones implementadas correctamente en los microservicios. Además, las pruebas de rendimiento y fiabilidad revelaron un sistema capaz de manejar cargas significativas con un alto rendimiento y una mínima tasa de error, mostrando una buena capacidad de recuperación incluso bajo cargas moderadas y pesadas. Mientras que el aplicativo web obtuvo un 90% en rendimiento, indicando su eficiencia en la interacción con usuarios. Por último, el sistema demostró buena portabilidad en diferentes navegadores y dispositivos. En conclusión, la creación de este sistema con microservicios ha demostrado que es eficiente en varios aspectos claves.

PALABRAS CLAVE: arquitecturas de software, microservicios, gestión de historial clínico, sistemas para clínicas veterinarias.

SUMMARY

The automation of services in organizations or companies are applied with software which have a monolithic architecture, which is widely used for the development of management systems; but, when an internal failure occurs, it is forced to be out of service in its entirety. Due to these cases, a solution is to implement the microservices architecture. Currently, in the field of veterinary medicine, the lack of technological tools hinders the automation and efficient management of processes in clinics, forcing them to be executed manually. The purpose of this work is to automate the processes of a veterinary clinic by creating a web and mobile application using an architecture based on microservices for an elevated level of availability and scalability. The agile DevOps methodology was chosen for the development of this proposal. The system was also assessed against ISO/IEC 25010 to ensure an elevated level of satisfaction in terms of functionality, performance, reliability, and portability. The evaluation results demonstrated an elevated level of satisfaction with the functionality of both the web and mobile application, supported by surveys and 100% compliance of correctly implemented functions in the microservices. In addition, performance and reliability tests revealed a system capable of handling significant loads with high throughput and minimal error rate, showing good resilience even under moderate and heavy loads. While the web application obtained 90% performance, indicating its efficiency in interacting with users. Finally, the system demonstrated good portability across different browsers and devices. In conclusion, the creation of this system with microservices has proven to be efficient in key aspects.

KEYWORDS: software architectures, microservices, medical record management, systems for veterinary clinics.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

D

DevOps: Considerada como una cultura y práctica de colaboración entre los equipos de desarrollo y operaciones de tecnología para crear, entregar y mantener software de forma más eficiente y confiable.

Disponibilidad: Se refiere a la medida en que un recurso o sistema está listo y accesible para su uso en un momento dado.

E

Escalabilidad: Capacidad de un sistema, aplicación o infraestructura para crecer y adaptarse a las necesidades cambiantes de una empresa o usuario sin perder rendimiento o eficiencia.

I

ISO/IEC 25010: Estándar internacional para la evaluación de calidad de productos software. Proporciona un marco para medir y evaluar diferentes atributos de calidad del software, como la funcionalidad, la confiabilidad, la usabilidad, la eficiencia, la mantenibilidad y otros.

M

Monolítico: Es aquel en el que todas sus partes y componentes están integrados en una sola unidad. Esto significa que todas las funciones y características de la aplicación se desarrollan, despliegan y mantienen como una única entidad.

Microservicios: son una arquitectura de software en la que una aplicación se divide en pequeños servicios independientes, cada uno de los cuales cumple una función específica y se ejecuta de manera autónoma.

R

Riguroso: Exacto, preciso, minucioso.

INTRODUCCIÓN

Con el avance tecnológico que se vive en la actualidad algunas empresas, instituciones u organizaciones buscan automatizar sus procesos con el objetivo de conseguir la satisfacción de sus clientes, la implementación de estos sistemas de gestión es una oportunidad para que dichas instituciones obtengan beneficios a corto plazo.

Las arquitecturas de software son la parte principal de todo proyecto software, estas permiten estructurar y relacionar las partes de un proyecto. Hace mucho tiempo en el desarrollo de software se ha usado una estructura tradicional para organizar proyectos, la arquitectura monolítica, la cual tiene todas sus funciones en un módulo. La arquitectura de microservicios es considerada como una solución a los problemas que causan la arquitectura monolítica.

El problema actual en el sector de medicina, específicamente en las veterinarias o clínicas veterinarias, no cuentan con las tecnologías para la gestión y control de información, realizando estos procesos de manera manual, lo que puede generar retrasos y errores en la gestión de pacientes y clientes.

En el presente trabajo se creará un sistema para la gestión de una clínica veterinaria implementando la arquitectura de microservicios para que dicho sistema sea factible, tenga disponibilidad y permita tener escalabilidad a futuro.

La siguiente propuesta contiene tres capítulos en el cual cada uno corresponde a una actividad para la elaboración de este proyecto de titulación y se los detallamos a continuación: **Capítulo I:** Presentación del marco teórico y los subtemas de interés como lo son los antecedentes de la investigación, antecedentes teóricos y antecedentes contextuales. **Capítulo II:** Detalle del desarrollo del prototipo, definiciones, metodología que se aplica para el desarrollo y la ejecución del prototipo. **Capítulo III:** En este último capítulo se indica la evaluación del prototipo, detallar el plan de evaluación y se presentan los resultados para luego pasar a las conclusiones y recomendaciones del proyecto.

i. Declaración y formulación del Problema

Declaración del problema

Actualmente, las necesidades de las mascotas se han vuelto muy considerable, desde la alimentación hasta su cuidado, por lo que surge la necesidad de llevar controles y registros de su salud. En el ámbito internacional, las clínicas veterinarias al igual que la tecnología han ido evolucionado con la implementación de sistemas para automatizar procesos de cuidado o atención de las necesidades, pero la mayoría de las clínicas aún manejan tradicionalmente sus operaciones [1].

Según [2], menciona que en el campo de la Medicina Veterinaria en Ecuador existe un problema en el cual la mayoría de las clínicas no eligen un sistema para la gestión de la información. Esto puede ser debido a que el país no implementa nuevas tecnologías o a que los sistemas disponibles en el mercado suelen tener un alto costo y no cumplen con las necesidades específicas de estas instituciones.

Por tal motivo, es necesario un sistema que permita gestionar y automatizar procesos administrativos y clínicos, teniendo como beneficiarios a estos negocios cerca de nuestra localidad. En la **Figura 1**, se presenta las causas, problema y efectos.

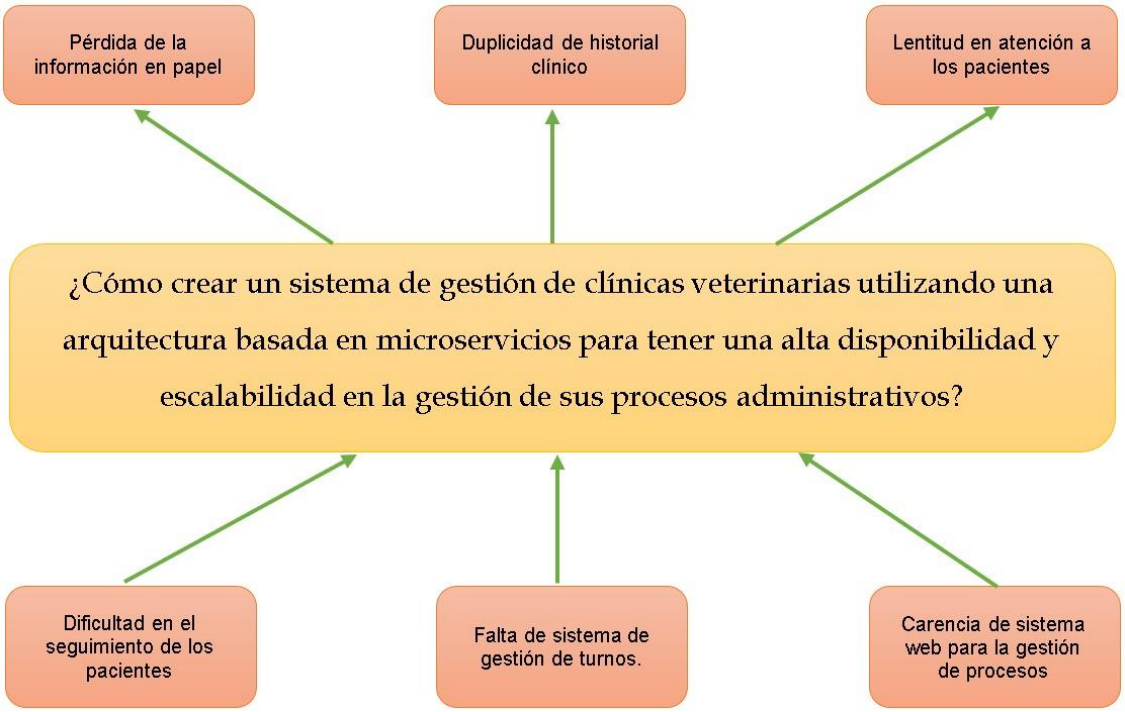


Figura 1: Árbol de problema

Formulación del problema

Problema principal:

- ¿Cómo crear un sistema de gestión de clínicas veterinarias utilizando una arquitectura basada en microservicios para tener una alta disponibilidad y escalabilidad en la gestión de sus procesos administrativos?

Problemas específicos:

- ¿Qué tendencias tecnológicas usar para la implementación de un sistema basado en una arquitectura de microservicios?,
- ¿Cómo obtener información de los procesos y la operabilidad que realizan las clínicas veterinarias?,
- ¿Qué técnicas usar para la implementación de una arquitectura de microservicios?
- ¿Cómo diseñar y desarrollar un sistema basado en microservicios?

ii. Objeto de estudio y Campo de acción

Objeto de estudio

- Procesos de una clínica veterinarias.

Campo de acción

- Desarrollo de un sistema para clínica veterinaria basada en microservicios.

iii. Objetivos

Objetivo general

- Crear un sistema web y móvil de gestión de clínicas veterinaria basado en una arquitectura de microservicios para la automatización, disponibilidad y escalabilidad de la información.

Objetivos específicos

- Realizar una revisión bibliográfica en artículos científicos sobre los procesos de clínicas veterinarias.
- Determinar los requisitos funcionales del sistema de clínica veterinaria.
- Desarrollar el prototipo de la aplicación web y móvil para la gestión de clínicas veterinarias en una arquitectura de microservicios.

- Evaluar el sistema de clínica veterinaria aplicando la norma y métricas de la ISO/IEC 25010.

iv. Hipótesis y variables o Preguntas de investigación

El desarrollo de un sistema para clínicas veterinarias utilizando una arquitectura de microservicios permitirá solventar la gestión de los procesos que estas realicen al dar atención a sus clientes.

Tabla 1: Definición conceptual de la hipótesis

Variables	Conceptos
Variable Independiente Desarrollo de un sistema utilizando una arquitectura de microservicios.	Un sistema se puede considerar como un conjunto de procesos que interactúan entre sí para optimizar la gestión de una empresa.
Variable Dependiente Gestión de los procesos que se realicen en una clínica veterinaria.	La gestión ayuda mejorar los procesos que se llevan a cabo dentro de una empresa.

Tabla 2: Operacionalización de variables

Variables	Categorías	Indicadores	Técnicas
Variable Independiente Desarrollo de un sistema web y móvil utilizando una arquitectura de microservicios.	1. Arquitectura de Microservicios 2. Sistema de Gestión.	1. Microservicios 2. Metodología 3. Requerimientos 4. Bases de datos 5. Aplicaciones móviles 6. Aplicación web	1. Arquitectura de microservicios 2. Metodología DevOps. 3. Recopilación de información mediante herramientas de recolección de datos. 4. Uso de distintas bases de datos como MySQL.
Variable Dependiente Gestión de los procesos que se realicen en una clínica veterinaria.	1. Gestión 2. Automatización 3. Eficacia	1. Gestión de turnos 2. Visualización de la plataforma 3. Agilidad en los procesos de la clínica veterinaria.	1. Correcto uso de la plataforma.

v. Justificación

En la actualidad, muchas clínicas veterinarias llevan el control y registro de información de las mascotas de manera tradicional, esto no garantiza que los datos proporcionados por los clientes estén seguros llegando a extraviarse o duplicarse, esto puede ocasionar que los servicios brindados por las clínicas se vean afectadas y a su vez creando una mala reputación de esta.

La creación de los sistemas de gestión de procesos ayuda a que los negocios mejoren sus servicios y estos sean eficaces logrando un adecuado manejo de la información, por ello surge la necesidad de crear un sistema para mejorar los procesos administrativos de las clínicas veterinarias y tener a

los clientes informados del historial de sus mascotas mediante un portal web o móvil que esté vinculado a la información respectiva.

Para el desarrollo de este sistema se considerará la gestión de turnos, inventario e historial clínico de las mascotas y a su vez permitir el acceso a la información de manera ágil y eficaz logrando también la mejora de búsqueda de registros.

CAPÍTULO I. MARCO TEÓRICO

1.1. Antecedentes de la Investigación

Para la elaboración de la revisión sistemática, se utilizó la Metodología de Revisión Sistemática de la Literatura (SRL: Systematic Review of the Literature). Una revisión sistemática es un tipo de investigación científica con el propósito de extraer la información relevante para escribir un trabajo investigativo. Este método es riguroso y explícito con el fin de identificar, evaluar e interpretar estudios anteriores sobre el problema de la investigación [3].

a) Preguntas de investigación

En la **Tabla 3**, se establece algunas preguntas de investigación como guía para la búsqueda de información sobre los microservicios y su aplicación en sistemas de gestión de información.

Tabla 3: Preguntas de Investigación

Preguntas de investigación	Descripción y motivación
RQ1. ¿Qué impacto tienen las arquitecturas de microservicios en los sistemas implementados?	Esta pregunta tiene el propósito de analizar el impacto de las distintas arquitecturas de microservicios para mejorar la integración de ellos al sistema.
RQ2. ¿Cuáles son los beneficios que proporcionan las arquitecturas basadas en microservicios?	El propósito de esta pregunta es de identificar los beneficios que aportan las arquitecturas de microservicios en los sistemas ya realizados.
RQ3. ¿Cuáles son las herramientas necesarias para la implementación de los microservicios?	El propósito de esta pregunta es de identificar y verificar que herramientas pueden ser necesarias para la implementación de microservicios.
RQ4. ¿Qué procesos internos realizan las clínicas veterinarias?	El objetivo de esta pregunta es de identificar que procesos internos y/o administrativos se realizan en una clínica veterinaria.
RQ5. ¿Qué desafíos se identifican al implementar los microservicios en sistemas de clínicas veterinarias?	Esta pregunta tiene como objetivo identificar los desafíos y/o problemas que se presentan en la implementación de los microservicios.

b) Palabras claves y Cadena(s) de búsqueda

Para la búsqueda se usaron palabras relevantes relacionadas a este trabajo y bases de datos para recolectar artículos de interés tales como:

- Scopus.
- Science Direct.
- IEEE Xplore.
- SpringerLink.
- ACM Digital Library.

En la elaboración de la cadena de búsqueda las palabras principales utilizadas permitirán que la búsqueda sea por títulos, resumen, palabras claves, y utilizando operadores AND y OR.

Cadena de búsqueda en inglés:

- (microservice* OR "microservices architecture" OR "microservices-architecture" OR "microservice-oriented architecture") AND (management AND software AND system OR "management software" OR "management system").

c) Criterios de inclusión y exclusión

En los criterios de inclusión y exclusión se tomaron en cuenta los siguientes:

Tabla 4: Criterios de inclusión y exclusión

#	Criterio de inclusión
1	Los trabajos deben cumplir un rango de 5 años (2018-2022)
2	Trabajos publicados hasta diciembre de 2022
3	Trabajos que relaciona Sistemas de Gestión y Microservicios
4	Trabajos que comparen sistemas de microservicios y monolíticos
#	**Criterio de exclusión**
1	Artículos cortos (≤ 3 páginas)
2	Trabajos duplicados (solo se incluyó una copia de cada estudio)
3	Conferencias y Actas de conferencias.
4	Trabajo redundante de la misma autoría
5	Artículos escritos que no sean en español e ingles
6	Trabajos irrelevantes para la investigación

d) Proceso y resultados de la búsqueda

Para el proceso y resultado de la metodología SRL, se desarrolló el filtrado de las bibliografías obtenidas a partir de la cadena de búsqueda y aplicando los criterios de inclusión y exclusión, como se ve en la figura**.**

En la **Figura 2,** se presencia el proceso de Revisión Sistemática de la Literatura utilizando la herramienta Parsifal, en la cual se pudo analizar cada uno de los artículos y generar los duplicados en cada base de datos.

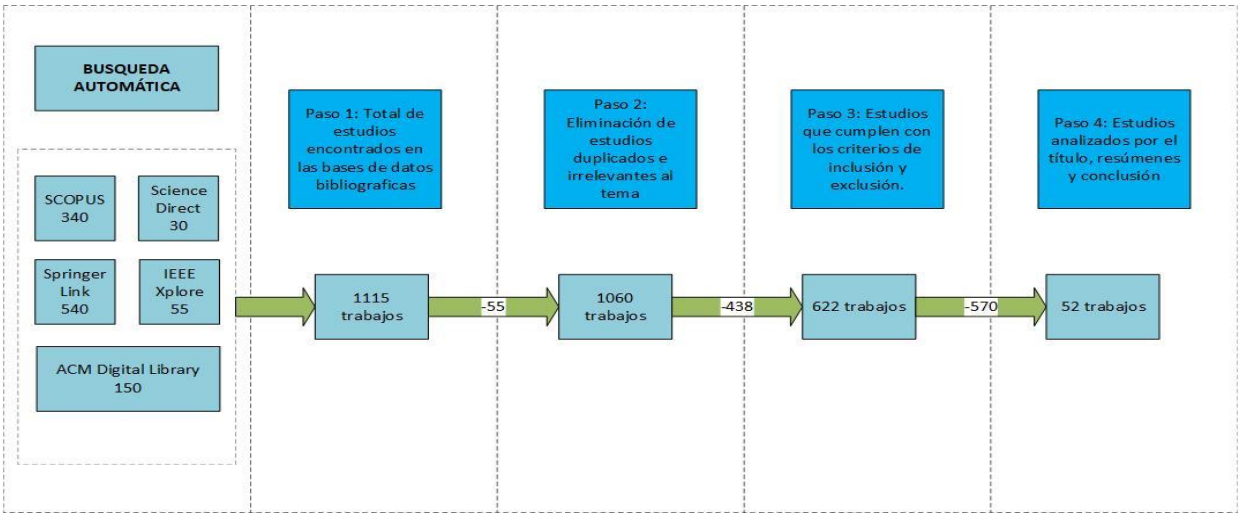


Figura 2: Proceso y resultados de la búsqueda

El proceso de Revisión Sistemática de la Literatura comienza desde la búsqueda automática utilizando la cadena de búsqueda que se propuso, estos resultados se importaron en la herramienta Parsifal para el análisis y realizar un filtrado de los trabajos a analizar.

Como primer paso, se descartaron los trabajos duplicados que estaban ubicados en distintas bases de datos bibliográficas, el siguiente paso se utilizó los criterios de inclusión y exclusión, y para finalizar se analizaron los títulos, resúmenes y conclusiones de los trabajos finales.

En la **Figura 3**, se presenta el porcentaje de los trabajos por las fuentes utilizadas en la revisión sistemática, con una gran cantidad de trabajos por parte de SpringerLink seguido de Scopus.

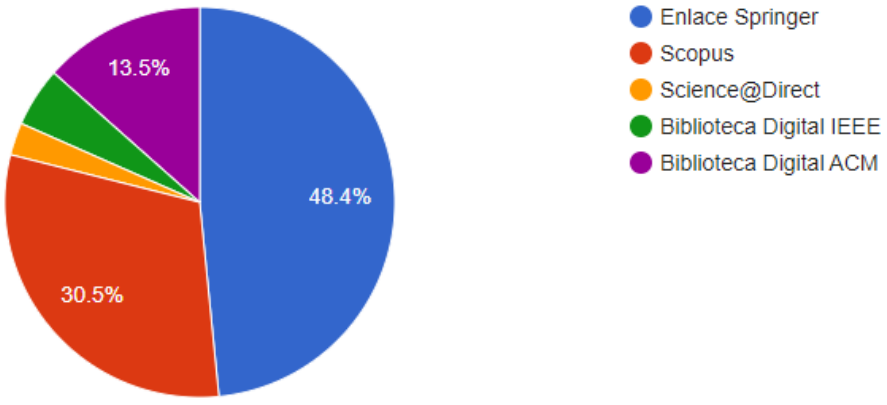


Figura 3: Artículos por fuente

Fuente: Parsifal

Utilizando Scopus, en la **Figura 4**, se evidencia la cantidad de documentos encontrados a partir de la cadena de búsqueda que se declaró.

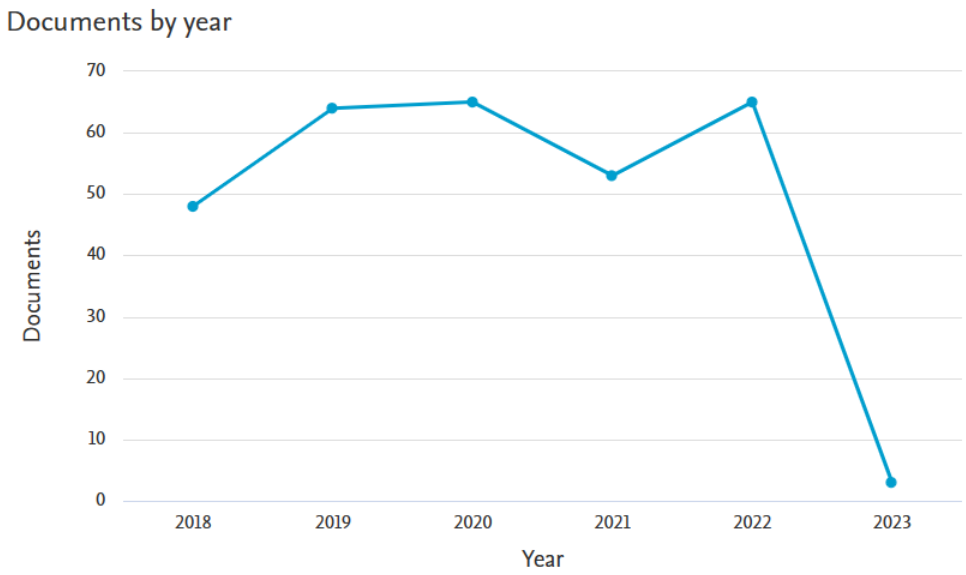


Figura 4: Cantidad de documentos por año

Fuente: Scopus

En la **Figura 4**, se realizó el análisis bibliométrico con el uso de palabras claves definidas en la cadena de búsqueda acerca de los sistemas de gestión y la aplicación de arquitecturas basada en microservicios.

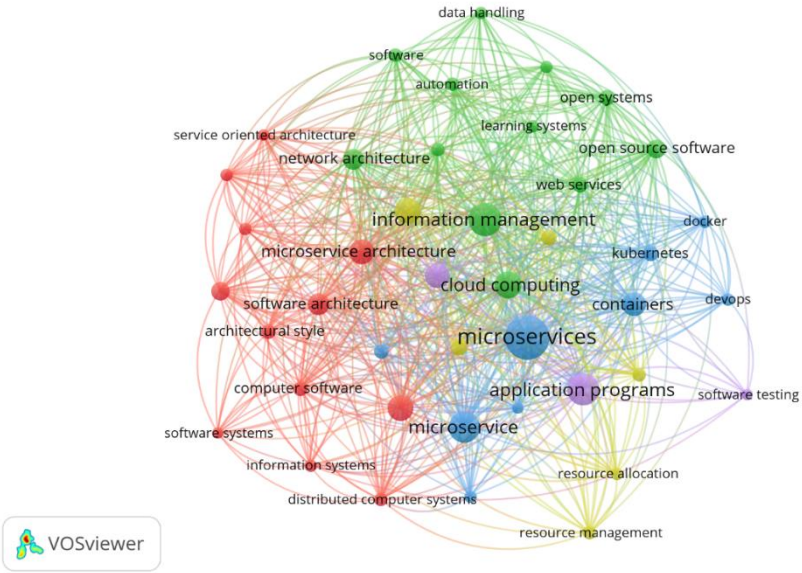


Figura 5: Cantidad de documentos por año

Fuente: VOSviewer

1.2. Antecedentes históricos

En 2020, Cedeño, Catuto y Rodas-Silva [2], desarrollaron una aplicación web para la administración de información en clínicas veterinarias, en la cual dio como resultado la reducción del tiempo de atención a las mascotas, mejora de la búsqueda de los registros clínicos y que sea más ágil el acceso a la información.

Como lo expresan De Gasperis, Della Penna y Facchini [4], rediseñar sistemas tradicionales con la arquitectura de microservicios permite que los sistemas sean escalables, permitiéndole adoptar progresivamente la asistencia en base al aprendizaje profundo en los procesos de gestión de eventos y también mejorar la protección de la sobrecarga de información.

Chen, *et al*. [5], resuelven un problema presente en los sistemas comunes, así como el retraso de flujo de información sobre datos materiales, baja eficiencia en la gestión y la incapacidad para la expansión de la arquitectura del sistema en la cadena de suministro verde de automóviles y proporcionando a la vez una efectiva plataforma de gestión y análisis de cadena de suministro verde vehiculares para los fabricantes basada en microservicios.

En el trabajo de Yu, Chang y Wang [6], se menciona que el uso de sistemas tradicionales provoca un bajo uso de recursos, valores altos en costo de mantenimiento, personalización extensa y una interfaz compleja así que proponen un sistema de control de supervisión integrado de tránsito ferroviario urbano basado en microservicios y en la nube para dar solución sistemática, inteligente y automática para este sistema.

Wang, Han y Nian [7], señalan que la arquitectura tradicional ya no logra satisfacer los requerimientos del desarrollo futuro, por lo que diseñan un sistema de gestión de tierra satelital basándose en la arquitectura de microservicios, con posibles resultados positivos en disponibilidad y confiabilidad, reduciendo grados de acoplamiento, cumplir con requisitos comerciales y buena escalabilidad.

En el trabajo de Fuentes Morales [8], se desarrolla un sistema de gestión para un centro veterinario que les permitió administrar los registros, control y seguimiento de las historias clínicas de los pacientes (mascotas) a través de la web. Este sistema brindó a los médicos veterinarios un mejor y eficiente servicio a sus clientes.

En [9], consideran construir una nube privada basada en microservicios con Docker y Kubernetes para los laboratorios de universidades para satisfacer los requisitos y demandas de un entorno de software complejo.

En 2022, Sadek, Craig y Trenell [10], proponen un sistema de búsqueda (ScanMedicine) utilizando funciones y microservicios con una técnica arquitectónica diferente de capas para proporcionar acceso a los profesionales a inteligencias que sustentan las innovaciones tecnológicas en el área de salud.

Como sostienen Aydemir y Başçiftçi [11], el desempeño del sistema es un factor por considerar, por lo que evalúan los tiempos de respuesta entre un sistema con arquitectura monolítica de sistemas bancarios centrales y un sistema con arquitectura de microservicios y demostrar el desempeño de la arquitectura de microservicios.

Batista, *et al*. [12], presentan una arquitectura de microservicios de múltiples inquilinos para atender varias aplicaciones de clientes. Analizan los beneficios, escalabilidad, administración e integración de sistemas de terceros y presentan los resultados empíricos de una evaluación para la adopción de multiusuarios y cuantificar su impacto en el rendimiento del sistema.

1.3. Antecedentes Teóricos

Para el desarrollo de los antecedentes teóricos se consideró los siguientes temas que se ven reflejados en la **Figura 6:**

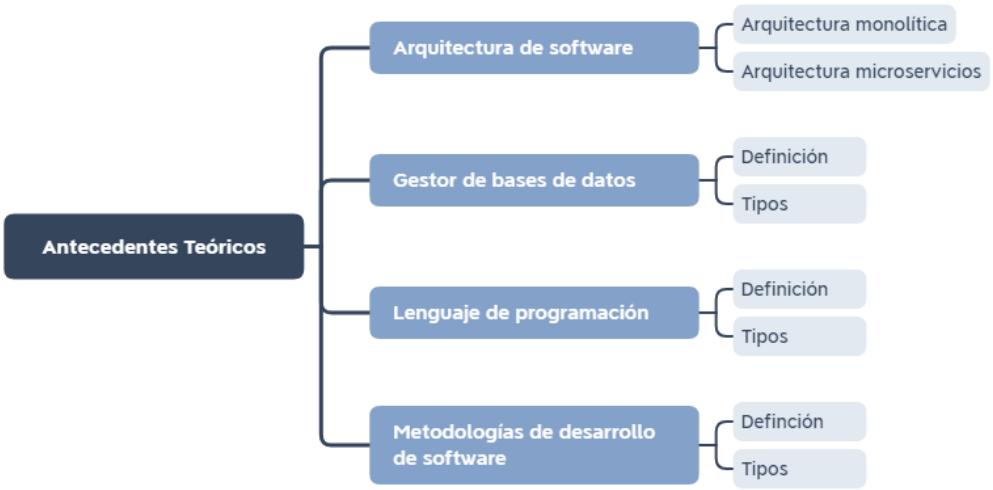


Figura 6: Antecedentes teóricos

1.3.1. Arquitectura de software

Las definiciones relacionadas a las arquitecturas de software en los sistemas informáticos son varias, y dentro de un conjunto de estructuras con elementos software incluidos, lo definen como el diseño de alto nivel, las relaciones y propiedades deben satisfacer ciertos requisitos de negocio. También se entiende como arquitectura software a un conjunto de patrones y abstracciones que sirven de referencia y guía para la construcción de los sistemas informáticos, es por eso que programadores, analistas e ingenieros les permita compartir una línea de trabajo en común [13].

Dentro de los conceptos sobre arquitectura de software se destacan los estilos y patrones arquitectónicos ya que son una colección de diseño de decisiones aplicables en contexto de desarrollo y problemas recurrentes. Dichas decisiones se denominan arquitectura de software [13].

La arquitectura de software es un conjunto de subsistemas y componentes interrelacionados que conforman un sistema de software. Estos elementos son descritos en distintas perspectivas y proveen funcionalidades y características no funcionales del sistema. Además, se busca definir una solución que satisfaga los requerimientos planteados, optimice el rendimiento, seguridad y facilidad de administración. La toma de decisiones es crucial en este proceso, ya que éstas pueden impactar significativamente la calidad, desempeño, mantenimiento y éxito del software involucrado [14].

Arquitectura monolítica

Las aplicaciones desarrolladas con una arquitectura monolítica abarcan algunas funciones y a su vez tienden a poseer una gran cantidad de código base. Para el desarrollo de aplicaciones monolíticas no requiere de una planificación avanzada de arquitectura y escala de manera horizontal mediante varias instancias de ejecución [15].

En el trabajo de Yang y Zhu [16], mencionan que la arquitectura monolítica presentan limitaciones en grandes proporciones, su estructura dirige a un acoplamiento de código serio y su escalabilidad es deficiente por lo que no llegan a satisfacer las necesidades de las empresas.

En [17], hacen mención que la aplicación del lado del servidor es monolítica por lo que es un ejecutable lógico, mientras que desde la vista de un sistema operativo, una aplicación monolítica es ejecutada como un proceso único en el servidor de aplicaciones y cuando se implementa una nueva versión esta debe ser reemplazada. La ventaja más significativa de esta arquitectura es su simplicidad y la agilidad en probar, implementar, depurar y monitorear.

Arquitectura de microservicios

Con el pasar del tiempo, este estilo ha cobrado impulso en el desarrollo e implementación de aplicaciones software, establecidos como un conjunto de pequeños servicios capaces de escalar de manera independiente y con la posibilidad de integrarse mediante mecanismos livianos de comunicación[18].

Esta arquitectura permite estructurar una aplicación en colecciones de servicios pequeños acoplados, los servicios tienen la posibilidad de ser implementados independientemente para ser probados y mantenerlos. Creada para superar las desventajas que presenta la arquitectura monolítica [15].

Hizo su aparición como un paradigma en la programación por componente pequeños, basados en los conceptos de SOA (Service Oriented Architecture) y OO (Orientación a Objetos) ayuda con la cohesión de las funciones de negocio por medio de la división de pequeños servicios que posea el sistema y facilitando su diseño para sistemas complejos. Los sistemas con esta arquitectura se destacan por su agilidad, resiliencia, escalabilidad, implementación simplificada, mantenimiento y despliegue [19].

En el trabajo de Auer, *et al*. [20], señalan que los microservicios escalan independientemente de otros servicios y a su vez desplegarse en hardware que más adapte las necesidades de sus características autónomas. Su tamaño con respecto a los grandes sistemas monolíticos favorece en el mantenimiento y tolerancia a fallos de los servicios y si llegase a presentar un fallo, esto no interrumpirá los demás procesos del sistema.

Balanceo de carga

Esta estrategia garantiza que el tráfico web no sea concentrado en un solo servicio, evita la saturación debido a la gran cantidad de solicitudes que proviene de algunos usuarios. Con esto obtenemos una optimización de recursos, mayor rendimiento, tiempo de respuestas cortos y reducción de sobrecarga de trabajo [21].

Proceso que ayuda a equilibrar o distribuir la carga receptada en un servidor o equipo mediante su interfaz con otros servicios, así logrando liberar la sobrecarga de lo procesado para que el tráfico sea atendido inmediatamente, evitando conflictos y malestar en los usuarios que requieran algún servicio [22].

Patrones arquitectónicos

Se define como una familia basados en esquemas de organización estructural que pueden ser vistos como bloques constructivos primitivos de las arquitecturas de software. Las arquitecturas de software en sistemas complejos suelen ser basados en más de un patrón de diseño, un ejemplo son las aplicaciones web, la construcción del diseño arquitectónico implica la vinculación de múltiples componentes y el uso de varios esquemas de organización [23].

Patrones de módulos (Module Patterns)

Permite descomponer el sistema en módulos o capas las cuales pueden ser jerárquicas y autónomas exponiendo su funcionalidad a través de una interfaz API. La relación que existe entre las capas es unidireccional permitiendo a las capas de más alto nivel hacer uso de los servicios de las capas adyacentes y no permitir una comunicación circular. Su diseño permite satisfacer los requisitos de modularidad, portabilidad, mantenibilidad y reusabilidad del código [13], [24].

Patrones C&C (Component-and-Connector Patterns)

Es una representación de conjuntos de decisiones asociadas a una estructura en la cual consideran los aspectos en tiempo de ejecución. Al definir componentes, estos serían las principales unidades funcionales y los conectores permiten la comunicación entre ellos [24].

MVC (Modelo-Vista-Controlador): Este enfoque presenta una clara separación entre los datos y la lógica de negocio en una aplicación, utilizando una interfaz de usuario y un módulo de gestión de eventos y comunicaciones. La arquitectura se compone de tres elementos fundamentales: el modelo, la vista y el controlador. El primero representa la información, mientras que los otros dos elementos permiten al usuario interpretarla de manera efectiva [25].

Patrones de asignación (Allocation Patterns)

Conjunto de decisiones de sistemas ligados a estructuras no software, como CPU, ficheros, redes, equipos de desarrollo, entre otros. Mediante la asignación de elementos software y los de computación físicos son considerados los entornos de ejecución [13].

1.3.2. Frameworks

1.1.1.1. Angular

Es un marco de aplicación web front-end, diseñado para facilitar la creación de aplicaciones web de alto rendimiento y funcionalidades. Basado en el lenguaje de programación TypeScript y en el

diseño Modelo Vista Controlador, agrega características estáticas y tipado fuerte, lo que proporciona robustez y escalabilidad al código[26].

1.1.1.2. Ionic

Es un SDK (Software Development Kit) front-end de código abierto para la creación de aplicaciones móviles hibridas ya sea para teléfonos Android, IOS y en aplicaciones web, usando HTML, CSS y JavaScript, ofreciendo componentes personalizados y acceso a APIs nativas a través de Apache Córdova con solo una base de código único [27], [28].

1.1.1.3. Spring

Este framework se basa en el modelo Vista Controlador, mediante componentes y librerías que facilita el desarrollo y despliegue de los servicios REST, eliminando una configuración para el uso de archivos XML. Dentro de sus principales características encontramos que nos permite la creación de aplicaciones independientes, ya que posee servidores embebidos y la integración con otros proyectos Spring es fácil[29].

Plecinski, *et al*. [30], mencionan sobre una ventaja que tiene Spring que es su propio contenedor IoC (Inversion of Control) esta práctica significa la transferencia de la responsabilidad para la creación de los objetos hacia el contenedor Spring, y se encarga de crear, gestionar y configurar objetos bean.

Spring cloud

En el trabajo de Wang, *et al*. [31], mencionan que Spring cloud pertenece al ecosistema de spring boot por lo que su uso y aprendizaje es intuitivo, y cuenta con varias herramientas para la construcción de los microservicios.

Servicio de registro y descubrimiento

Para los registros y descubrimientos de los microservicios existen varias herramientas que nos ayudan a realizar estas acciones, en este documento se hará un enfoque en Eureka ya que este servicio permite que los microservicios se registren y a la vez Feing permite que se comuniquen entre sí. Esto posibilita que se pueda volver a implementar un sistema determinado en varios entornos sin llegar a alterar las configuraciones de los servicios dependientes [31].

Comunicación entre servicios

En [32], señala que existen diferentes maneras de comunicar los servicios entre sí y lo considera como parte crucial de una arquitectura de microservicios, pero que no existe un método adecuado

y eficiente para esta comunicación. Resaltan la investigación de Christy Pachikkal en la cual menciona que la comunicación es sincrónica y la mensajería asincrónica.

1.3.3. Gestores de bases de datos

Una definición sobre las bases de datos seria como un conjunto de datos relacionados que esta agrupados o estructurados y almacenados. También consideradas como un banco de información en la cual están almacenadas varios datos con temáticas variadas pero con algún tipo de relación o vinculo para ser indexados, clasificados y ordenados [33].

Bases de datos relacionales

Las bases de datos de este tipo utilizan transacciones para garantizar la integridad de los datos. La propiedad ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) se aplica en estos casos debido al uso de transacciones, lo cual puede tener un efecto en el rendimiento en un entorno de datos concurrente. Se componen de conjuntos de tablas que, en las consultas de información, pueden estar relacionadas entre sí. Sin embargo, la operación JOIN entre tablas puede impactar negativamente en el rendimiento del sistema [34].

Bases de datos NoSQL o no relacionales

Estas bases de datos son una respuesta a la evolución en el almacenamiento de información y no son categorizadas como un tipo de base de datos, sino como un grupo de diferentes tipos de bases de datos, se distinguen de los tradicionales sistemas de gestión de bases de datos relacionales en varios aspectos, el principal es no poseer un lenguaje de consulta estructurada como lenguaje prioritario. No requieren de estructuras fijas y tabulación [34].

1.3.4. Lenguajes de programación

Es la base para la construcción de aplicaciones digitales que hoy en día usamos, ha permitido la creación y ejecución de hardware y software, sitios webs, apps, entre otras [35].

Son conjuntos de sintaxis y reglas semánticas que sirve como técnica de comunicación próximo al lenguaje humano para indicar instrucciones a un computador lo que específicamente quiere conseguir y se clasifican en tres grupos los cuales son: lenguaje máquina, lenguaje de bajo nivel y lenguaje de alto nivel [36].

JavaScript

Conocido por su uso en entornos y aplicaciones web. Es muy popular ya que está repleto de soluciones independientes en las cuales involucran bases de datos, aprendizaje automático y utilidades para red[37].

1.3.5. Metodologías de desarrollo de software

"Las metodologías son un conjunto de técnicas y métodos que mejoran la calidad de los productos controlando los procesos de desarrollo" [38]. Las metodologías se aplican en todas partes lo que permite que los procesos sean controlados y tener eficiencia y desempeño en el costo y el tiempo en el que desarrollan.

Las metodologías de desarrollo son un marco de trabajo en el que aplica para estructurar, planificar y controlar todos los procesos del desarrollo de los productos de software, mejorando la calidad de los productos [38], [39]. Las metodologías de software se clasifican en ágiles, tradicionales e hibridas.

Metodologías tradicionales

Las metodologías tradicionales son consideradas aquellas que cumplen con un software más eficiente y predecible. Estas se caracterizan por definir total rigidez en los requisitos al inicio de los proyectos de ingeniería de software lo cual exige una planificación de todo el trabajo antes de comenzar el desarrollo del software [38], [39].

En la **Tabla 5**, se presenta las definiciones de las metodologías tradicionales encontradas:

Tabla 5: Conceptos de metodologías tradicionales

Metodología	Concepto
Cascada	En [40], indica que es uno de los modelos más fáciles de administrar, realizando una planificación y desarrollo del software debido a que los requerimientos son definidos en la primera fase.
Espiral	Según [38], define como dimensión radial en el cual se presentan los costos acumulados y en el ángulo en el cual representa el progreso.
Incremental	Según [41], menciona que el modelo incremental o iterativo es semejante al modelo de cascada con una propuesta de iteraciones, esto beneficia al modelo debido a que en cada etapa podemos agregar nuevas funcionalidades al producto.
Proceso racional unificado (RUP)	Como sostienen Vera *et al.* [42], ordena y estructura el desarrollo de software, en la cual se realiza actividades necesarias para la obtención de los resultados respectos a los requerimientos del sistema.
Desarrollo de aplicaciones rápidas RAD	En el artículo [38], indica que la metodología RAD implica la creación de un prototipo rápido y su entrega al usuario para su evaluación. La retroalimentación obtenida se utiliza para mejorar el prototipo en función de las opiniones y necesidades del usuario.

Metodologías agiles

Las metodologías agiles presentan resultados en el menor tiempo posible sin afectar a la calidad de los productos de software, tomando en cuenta la satisfacción del cliente con las entregas tempranas y constantes [38].

En la **Tabla 6**, se presenta las definiciones de las metodologías ágiles encontradas:

Tabla 6: Conceptos de metodologías ágiles

Metodología	Concepto
Programación extrema	Es una metodología ágil que está formada por un conjunto de valores y principios para potenciar el trabajo en equipo y conseguir un buen clima de trabajo. Esta metodología se adecua a proyectos cambiantes el cual tiene un rápido desarrollo y un producto de software de alta calidad [43], [44].
Scrum	Como mencionan Núñez *et al.* [45], es un enfoque incremental de procesos que ayuda a tener organizado y mejorar la forma de entrega de los resultados.
Crystal	Según [46], es una metodología más ágil y flexible, debido a las entregas frecuentes del producto, pero es una de las metodologías más complicadas de entender porque depende de la experiencia del líder de equipo.
Mobile-D	Como sostiene Vásquez [47], Mobile-D es una metodología basada en otras como XP, Crystal y RUP, para producir software móvil utilizando las prácticas de desarrollo, escalabilidad de los métodos de Crystal y el ciclo de vida de RUP.

Metodologías híbridas

Las metodologías hibridas son la combinación de ambas metodologías en las cuales refuerzan las debilidades que se presentan en una metodología al realizar un desarrollo de software. Con las metodologías hibridas se puede tener un manejo de las actividades y su distribución.

1.4. Antecedentes Contextuales

Este estudio tiene el propósito del desarrollo de un sistema de gestión para los procesos que se realizan en una clínica veterinaria, desarrollando un sistema web y móvil en el cual se utilizó una arquitectura basada en microservicios para una alta disponibilidad y escalabilidad de estos. Se implementó diferentes microservicios para cada proceso identificado en una clínica veterinaria, en este caso la clínica veterinaria “Happy Pets”.

1.4.1. Ámbito de aplicación

Los sistemas para la gestión de información en la actualidad se han vuelto imprescindible para el desarrollo de actividades que realiza un negocio o una empresa, y mucho mejor si el sistema se maneja en la web o en el móvil.

Es evidente que estos sistemas web en una veterinaria o clínica veterinaria no están presentes o tienen fallos constantemente. En la clínica veterinaria “Happy Pets” de la Ciudad de Machala, no

tiene un sistema que gestione la información de los procesos que realizan, sus actividades de control las realiza de manera manual. Por tal motivo, se desarrolló un sistema web y móvil con la integración de diferentes microservicios en la que nos permita automatizar los procesos presentes en una clínica veterinaria y dar la oportunidad de llevar el control de las mascotas de manera digital.

1.4.2. Establecimiento de requerimientos

En la clínica veterinaria "Happy Pets", se identificaron y priorizaron los requerimientos más importantes para el desarrollo del sistema mediante una entrevista y guía de observaciones donde se indican que procesos se realizan.

Los procesos son los siguientes:

- Hospitalización.
- Consulta.
- Estética.
- Administrativos.

En los **procesos de hospitalización**, registra los datos de la historia clínica de las mascotas, incluyendo el motivo de la consulta, medicamentos administrados, datos del examen clínico, entre otros.

En el caso de los **procesos de consulta** se define dos opciones para requerir los servicios de la clínica, en este caso si es una consulta para revisión médica del veterinario o requerir algún proceso estético.

En los **procesos de estética**, comienza desde la programación de la cita para los servicios de Spa y belleza de las mascotas.

En los **procesos administrativos**, se puede destacar todos los procesos de contabilidad y compras, desde las transacciones de la empresa, hasta la revisión del inventario.

El formato de ambos instrumentos de recolección de datos se encuentra en Anexo 3 y Anexo 4, en estos se obtuvo información del manejo que lleva el veterinario para consultas, registros de productos, vacunas, desparasitación y el historial de las mascotas; todas estas actividades el veterinario las realiza de manera manual.

Requerimientos funcionales

La **Tabla 7** expone los requisitos funcionales que el sistema debe satisfacer.

Tabla 7: Requerimientos funcionales

COD.	Descripción	Descripción
RF01	Control de acceso	Se gestiona la autenticación para el sistema administrativo web y el aplicativo móvil para el agendamiento de citas.
RF02	Gestión de mascotas y dueños	Control de información de los propietarios y sus mascotas, en la cual se registrará toda la información.
RF03	Gestión de historias clínicas	Manejo de información clínica de las mascotas, registro de fichas clínicas, vacunación, desparasitación y exámenes.
RF04	Gestión de agendamiento de citas.	Agendamiento de citas para consulta clínica y/o procedimiento estético.
RF05	Gestión de inventario	Manejo de inventario de productos de la clínica veterinaria.
RF06	Gestión de mensajería.	Control de mensajería para notificaciones de próximas citas de las mascotas

Requerimientos no funcionales

La **Tabla 8** contiene los requisitos no funcionales que el sistema debe cumplir, así como la verificación de la arquitectura.

Tabla 8: Requerimientos no funcionales

COD.	Descripción
RNF01	El sistema debe presentar una disponibilidad del 97% cuando un usuario intente acceder al mismo.
RNF02	El sistema debe ser capaz de incorporar nuevas funcionalidades sin causar interrupciones en su operación.
RNF03	El sistema administrativo podrá ser utilizado a través de un navegador web.
RNF04	La aplicación móvil debe ser compatibles con dispositivos Android versión 6.0 y superiores
RNF05	Las peticiones que se realiza en el sistema deben ser inmediatas.

CAPÍTULO II. DESARROLLO DEL PROTOTIPO

2.1. Definición del prototipo

Una vez evaluados los requisitos del sistema, se estableció las componentes que conformarán la arquitectura de microservicios, junto con los patrones arquitectónicos y estrategias que se aplicarán.

- Se ha decidido utilizar una API Gateway para integrar los microservicios del sistema, la cual funcionará como el único punto de acceso al mismo. La API Gateway tambien funcionará como el control de acceso de los usuarios, definiendo una autenticación.
- Cada microservicio esta basado en un patrón de diseño de software, el cual es MVC (Modelo-Vista-Controlador) y como protocolo de comunicación utiliza REST.
- Se definió una base de datos en la cual integrará toda la información del sistema.

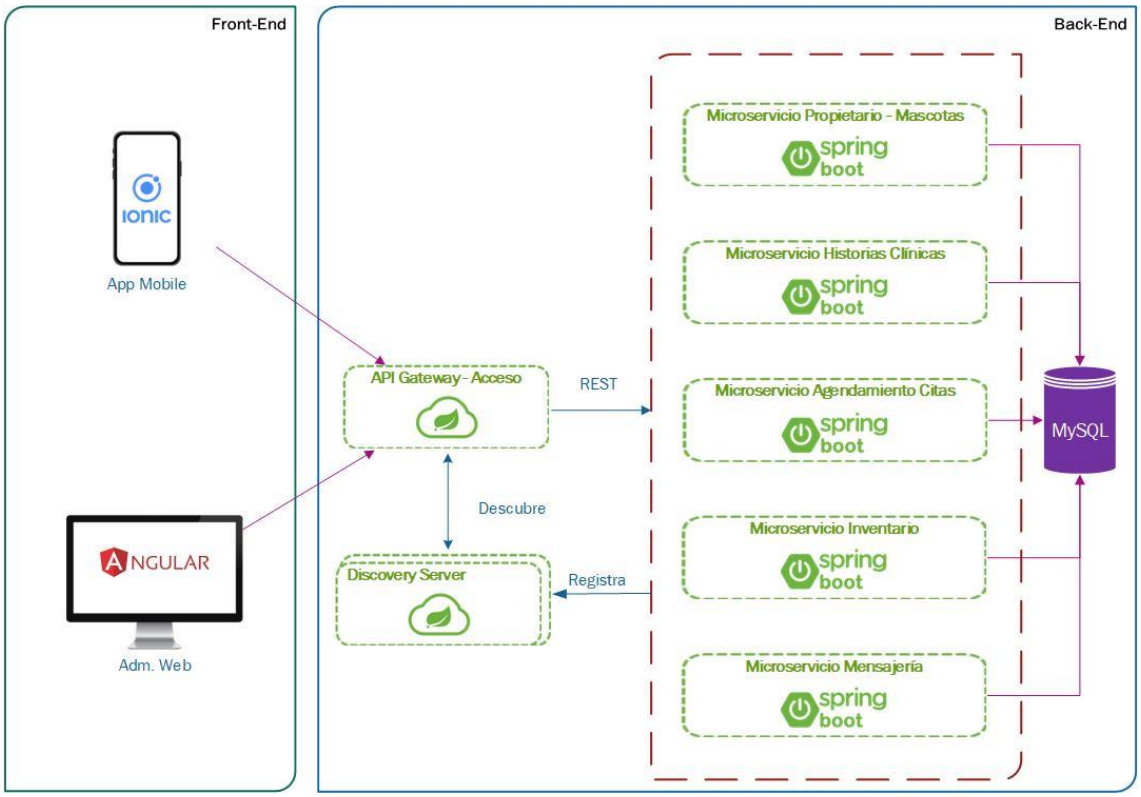


Figura 7: Arquitectura del Sistema

El modelo de prueba incluye una aplicación web destinada a la gestión del sistema por parte de los administradores, una aplicación móvil para los clientes (propietarios de las mascotas), una API Gateway, cinco microservicios específicos y una base de datos MySQL.

Estos microservicios presentados son: gestión de mascotas, historias clínicas, agendamiento de citas, inventario y mensajería.

2.2. Metodología de desarrollo del prototipo

2.2.1. Enfoque, alcance y diseño de investigación

Enfoque de investigación

El enforque de investigación de este trabajo es cuantitativo, ya que utiliza herramientas de recolección de datos con la finalidad de identificar y analizar los procesos que se realizan en las clínicas veterinarias, estos serán analizados a través de técnicas estadísticas.

Alcance de investigación

El alcance de investigación de este trabajo es descriptivo, en la cual nos permite describir las características del fenómeno a estudiar con el fin de determinar los procesos de una clínica veterinaria para la definición de los servicios a implementar en el sistema.

Diseño de investigación

El diseño de investigación es cuasi-experimental porque se analizó los procesos de una clínica veterinaria para el desarrollo del sistema de gestión, estos serán tomadas a partir de las técnicas de recolección de datos.

2.2.2. Unidades de análisis

Población

La población para la investigación será los procesos de una clínica veterinaria.

Muestra

Para determinar el muestreo será no probabilístico por conveniencia ya que no se conoce la cantidad exacta de los procesos de la clínica veterinaria.

Se evaluará otros parámetros del sistema como:

- Funcionalidad
- Fiabilidad
- Eficiencia
- Portabilidad

2.2.3. Técnicas e instrumentos de recopilación de datos

Para el desarrollo del sistema utiliza las siguientes técnicas:

- **Observación:** Guía de observaciones y lista de control de los procesos de las clínicas veterinarias.
- **Entrevista:** Documento formado por preguntas (máximo 5 preguntas), las preguntas relacionadas a la gestión de las clínicas veterinarias aplicada al jefe o propietario de una clínica.

2.2.4. Técnicas de procesamiento de datos para la obtención de resultados

Las técnicas para utilizar para el procesamiento de datos se definen a partir de las técnicas de recolección de datos en las cuales es necesario realizar la tabulación utilizando los siguientes:

- Tablas estadísticas.
- Cuadros estadísticos.
- Gráficos estadísticos.

2.2.5. Metodología o métodos específicos

La metodología utilizada para el presente trabajo de investigación es DevOps. Este término proviene de la combinación de "Development" (Desarrollo) y "Operations" (Operaciones), el cual tiene un enfoque colaborativo en base a principios ágiles que ayuda al desarrollo en el menor tiempo posible debido a su ciclo de vida y las entregas continuas [48].

En la **Figura. 8**, se presenta las fases del ciclo de vida DevOps:

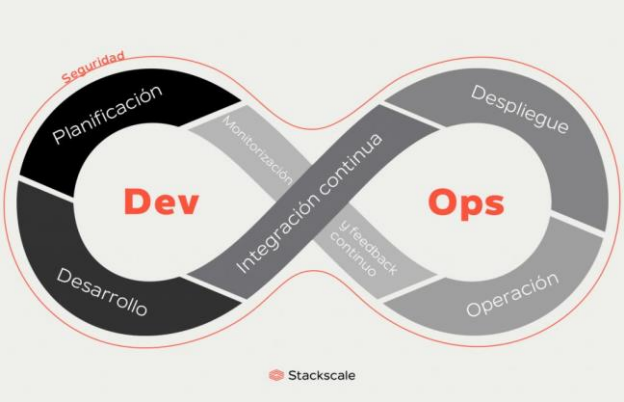


Figura 8: El ciclo DevOps *[49]*

Las fases de la metodología DevOps generalmente incluyen los siguientes pasos:

- **Planificación:** Establece los objetivos y se planifican los proyectos de software. Se definen las necesidades del negocio y se establecen las expectativas en términos de tiempo, costos y recursos.

- **Desarrollo:** Los desarrolladores escriben el código y construyen la aplicación. Utilizan herramientas de integración continua (CI) y entrega continua (CD) para automatizar la construcción y el despliegue del software.
- **Pruebas:** Los equipos realizan pruebas en entornos controlados para asegurarse de que el software funcione correctamente y cumpla con los estándares de calidad.
- **Despliegue:** Despliegue del software en producción. Utilizan técnicas de despliegue continuo para asegurarse de que el software se implemente de manera rápida y segura.
- **Monitorización:** Supervisión del software en producción para detectar y resolver problemas. Utilizan herramientas de monitorización en tiempo real para asegurarse de que el software funcione de manera eficiente y escalable.
- **Mejora continua:** Los equipos retroalimentan y mejoran el proceso de desarrollo y entrega. Utilizan los datos recopilados en la monitorización para identificar oportunidades de mejora y aplicar cambios.

La secuencia de acciones que conforman el ciclo de vida de DevOps resulta en una división entre el entorno de desarrollo y el de operaciones o producción. Esto establece una visión global de ambos entornos que implica la mejora continua del software, mediante la automatización de procesos y la implementación de entregas continuas, lo que se traduce en una mayor calidad del software. En la figura 9, se expone el diseño Arquitectura basada en Microservicios y DevOps.

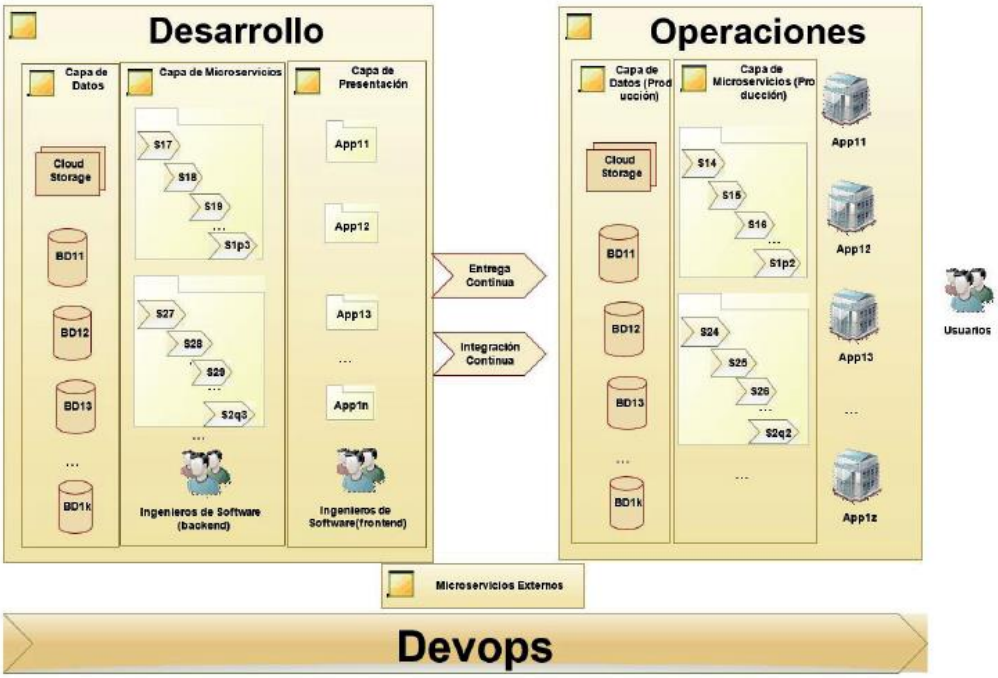


Figura 9: Arquitectura basada en Microservicios y DevOps.
Fuente: Rodríguez, *et al.* [50]

2.2.6. Herramientas y/o Materiales

El prototipo fue construido con las herramientas presentadas en la tabla 9.

Tabla 9: Herramientas y tecnologías utilizadas

Clasificación	Herramientas y/o materiales
Software	• IntelliJ IDEA • Visual Studio Code • Postman • Git • Github • Docker • Microsoft Visio
Frameworks	• **Front-end:** Angular, Ionic • Back-end: Spring Boot
Lenguajes de programación	• Java • JavaScript
Bases de datos	• MySQL
Proveedor de servicios	• DigitalOcean

2.3. Desarrollo del prototipo

En la sección anterior **2.2.5.** define la metodología DevOps para el desarrollo del prototipo, este permite acelerar el tiempo de entrega de software y mejorar el proceso de desarrollo.

2.3.1. Fase I: Planificación

En este apartado define los usuarios, historias de usuarios, diagramas del prototipo, entre otros, los mismos que se detallan a continuación:

Definición de usuarios del sistema

Se considero los siguientes usuarios para el acceso del sistema:

- Usuario (Clientes o Dueños de Mascotas)
- Doctor (Usuario administrador)

Historias de usuarios

En la **Tabla 8** se presenta el resumen de las historias de usuarios analizadas para el desarrollo del prototipo y a partir de los requerimientos funcionales establecidos en el apartado **1.4.2**.

Tabla 10: Resumen de historias de usuarios

Nº		
1	Inicio de Sesión	Clientes – Doctor
2	Gestión de usuarios y roles (Gateway)	Administrador
3	Gestión de mascotas	Doctor -
4	Gestión de historias clínicas	Doctor
5	Gestión de citas	Cliente – Doctor
6	Gestión de inventario	Doctor
7	Gestión de comunicación	Doctor

Diagrama de colaboración

A través de los diagramas de colaboración se realizó el comportamiento dinámico del sistema administrativo en la cual se resume las iteraciones que realiza los usuarios al momento manejar el sistema con la implementación de microservicios.

En la **Figura 9** presenta el diagrama de colaboración propuesto para el sistema administrativo.

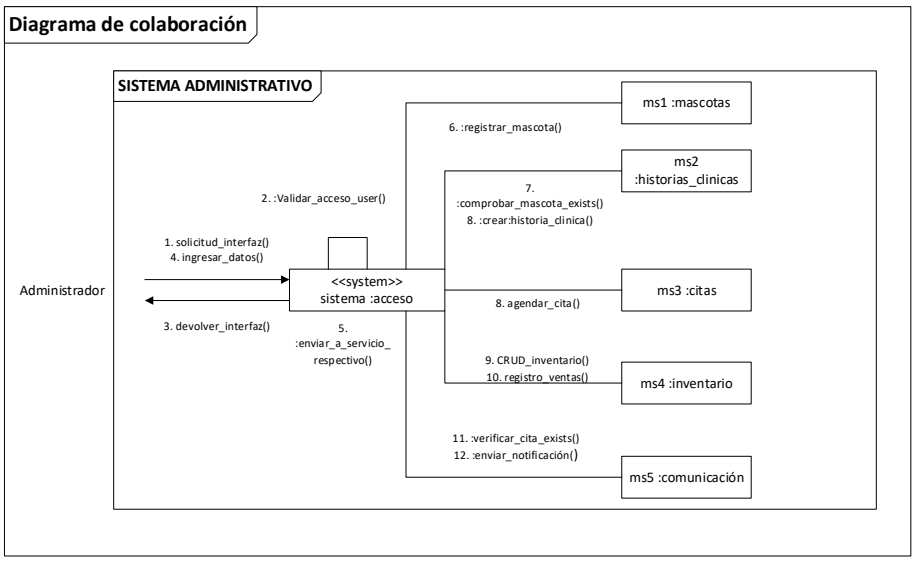


Figura 10: Diagrama de colaboración

Diagrama de carril

En el diagrama de carril presenta el comportamiento del flujo de datos de la aplicación para citas en la cual muestra el proceso de los datos a través de una arquitectura de microservicios. En la **Figura 10** propone el diagrama de carril para el flujo de datos en la aplicación móvil con arquitectura de microservicios.

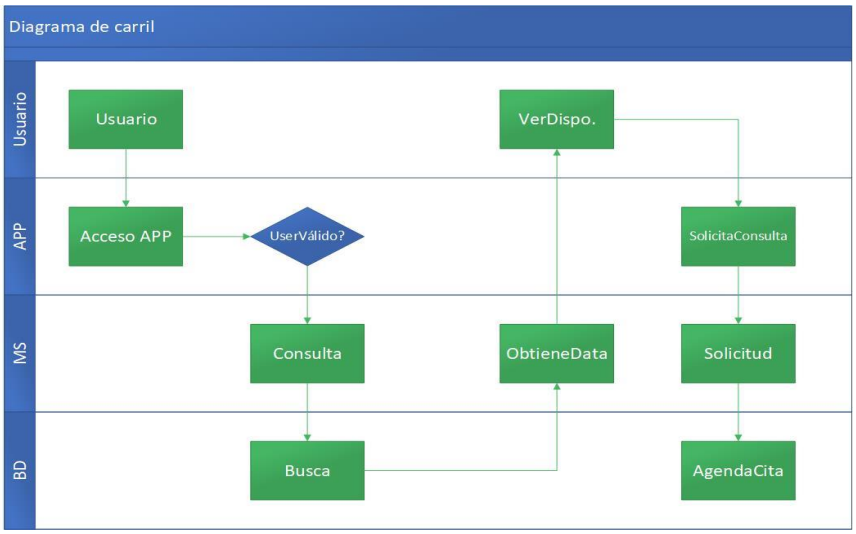


Figura 11: Diagrama de colaboración

Diseño de interfaces

Para el diseño de las interfaces del sistema se usó la herramienta balsamiq y a continuación se muestra los siguientes bosquejos:

Para entrar al sistema se debe ingresar el correo electrónico y contraseña del personal que tenga permiso de manejar el software, en la **Figura 12** se aprecia el diseño de Login.

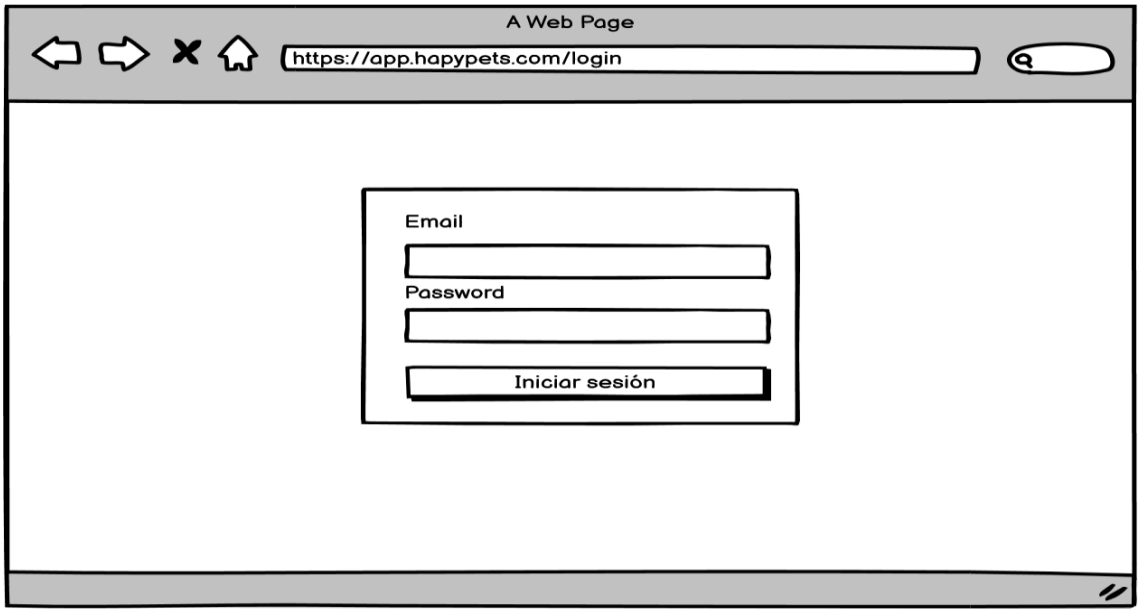


Figura 12: Diseño Login

A continuación, en la **Figura 13** se mostrará la pantalla principal del sistema, en la cual tiene varias opciones para seleccionar, ver información de los propietarios, agenda, inventario, entre otras.

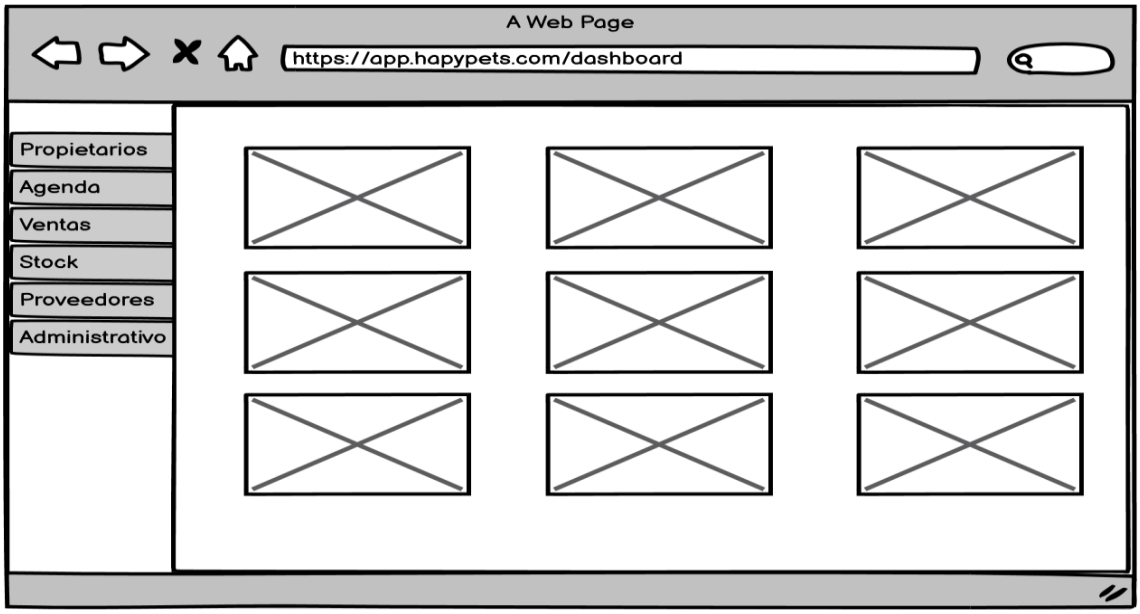


Figura 13: Diseño Dashboard

En la **Figura 14** se muestra la opción clientes, aquí se registrar los dueños de las mascotas y a su vez los datos del animalito.

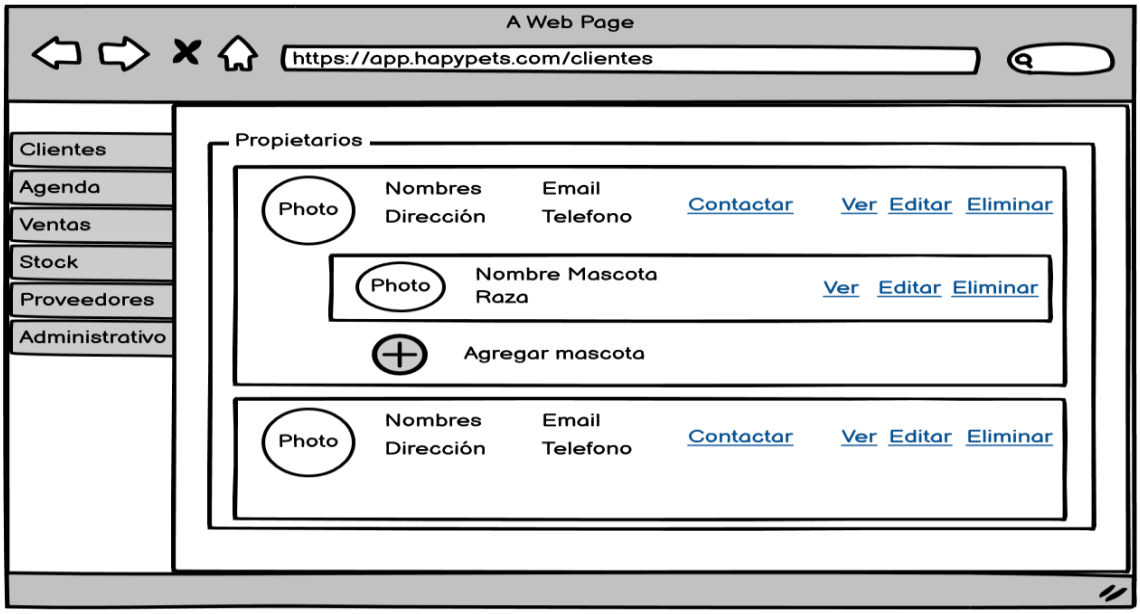


Figura 14: Diseño Propietarios - Mascotas

En la **Figura 15** se muestra la pestaña ficha, en la cual se permite registrar las fichas de diagnóstico de la mascota para las citas que tenga a futuro. Van desde los datos de las mascotas hasta los signos vitales.

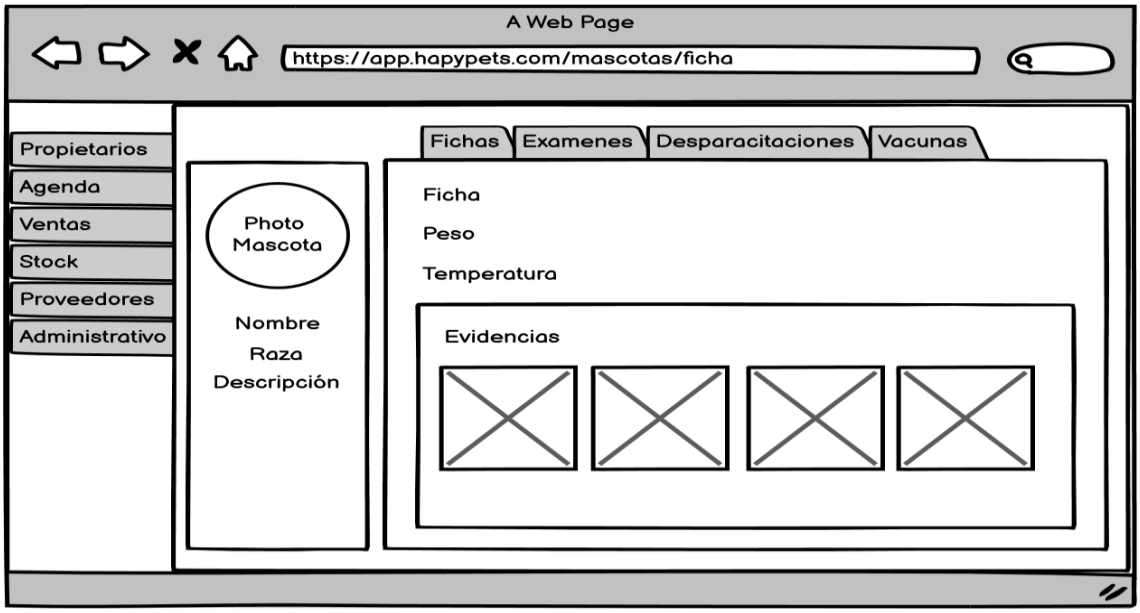


Figura 15: Diseño Historia Clínica-Ficha

En la **Figura 16** se muestra la pestaña exámenes donde está la información y evidencia de exámenes o evidencias, serán registrados después de obtener sus resultados para que el médico pueda visualizarlos sin ningún problema de extraviar la información.

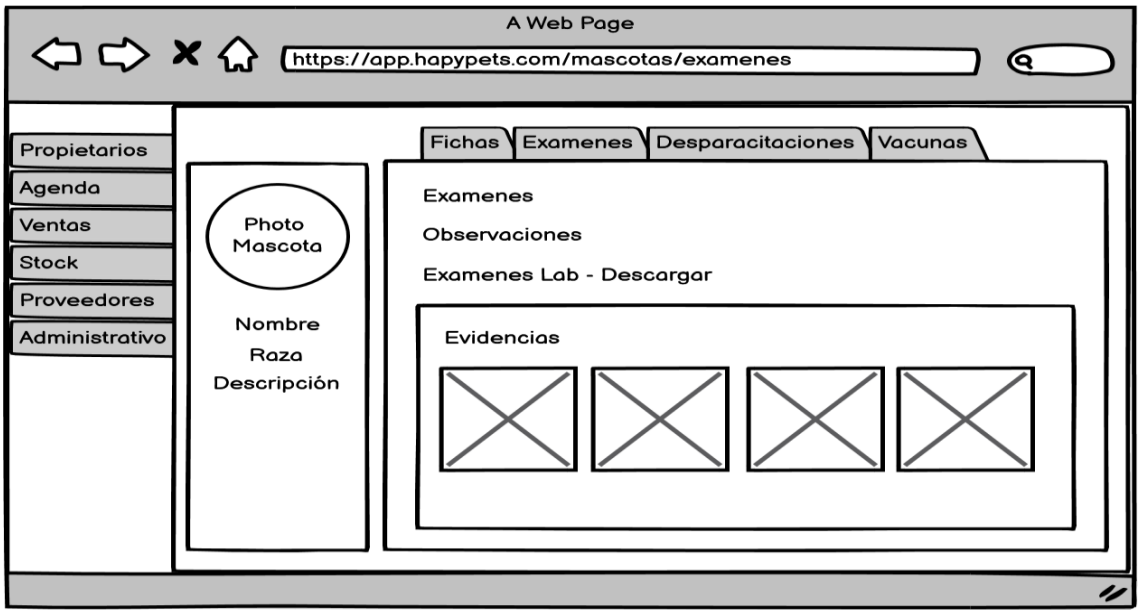


Figura 16: Diseño Historia Clínica-Exámenes

En la **Figura 17** está la pestaña de desparasitaciones aquí se muestra el historial de desparasitación de la mascota.

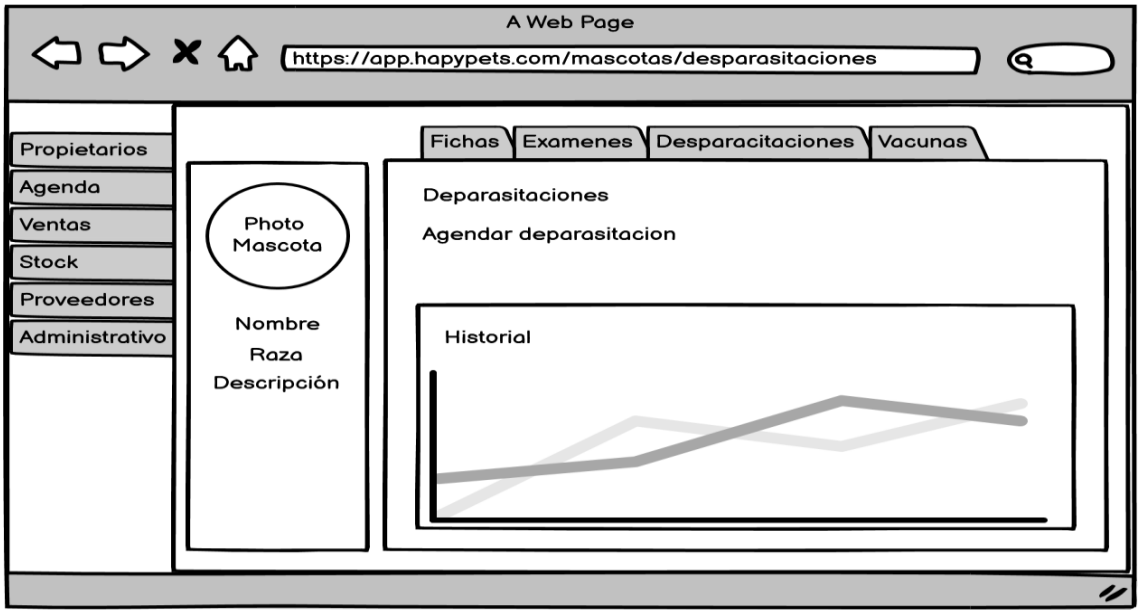


Figura 17: Diseño Historia Clínica-Desparasitaciones

En la **Figura 18** se revisa el historial de las vacunas de la mascota que se encuentre registrada dentro del sistemas.

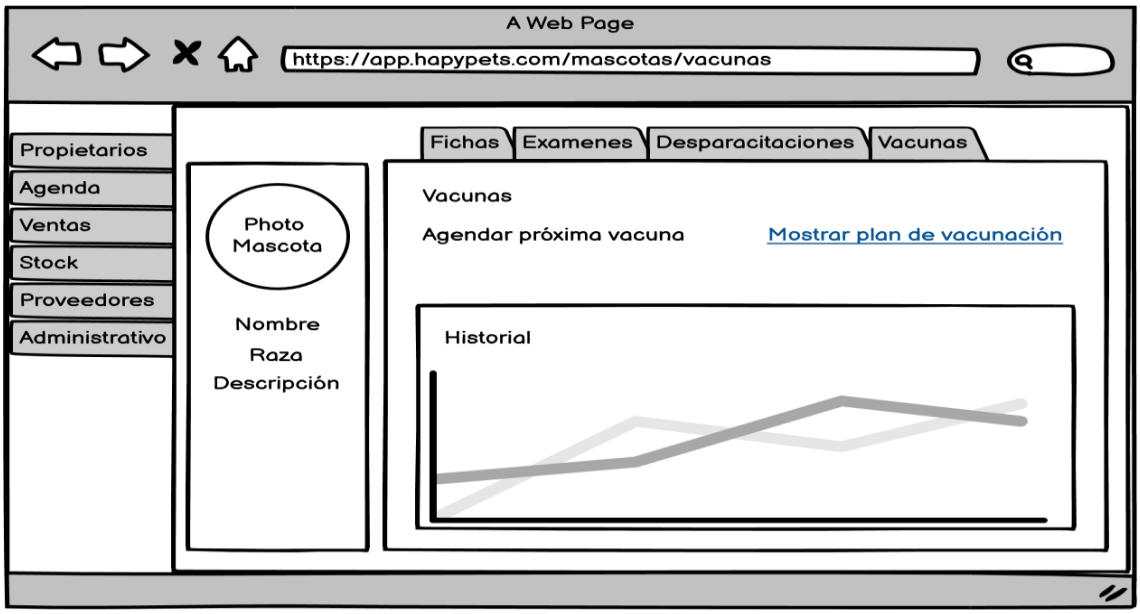


Figura 18: Diseño Historia Clínica-Vacunas

Para agendar una cita nueva, está disponible una pestaña de agenda para revisar citas programadas o agregar una nueva, en la **Figura 19** se muestra la interfaz de esta sección.

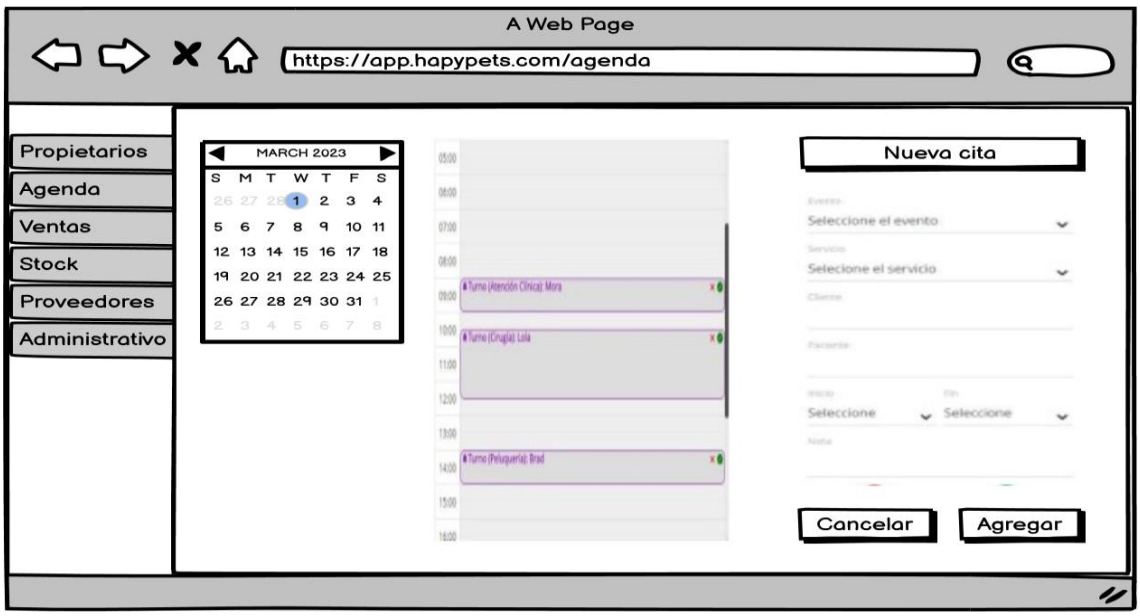


Figura 19: Diseño Agenda

En la **Figura 20**, se muestra el apartado productos donde se puede registrar los que la clínica adquiera también de eliminarlos

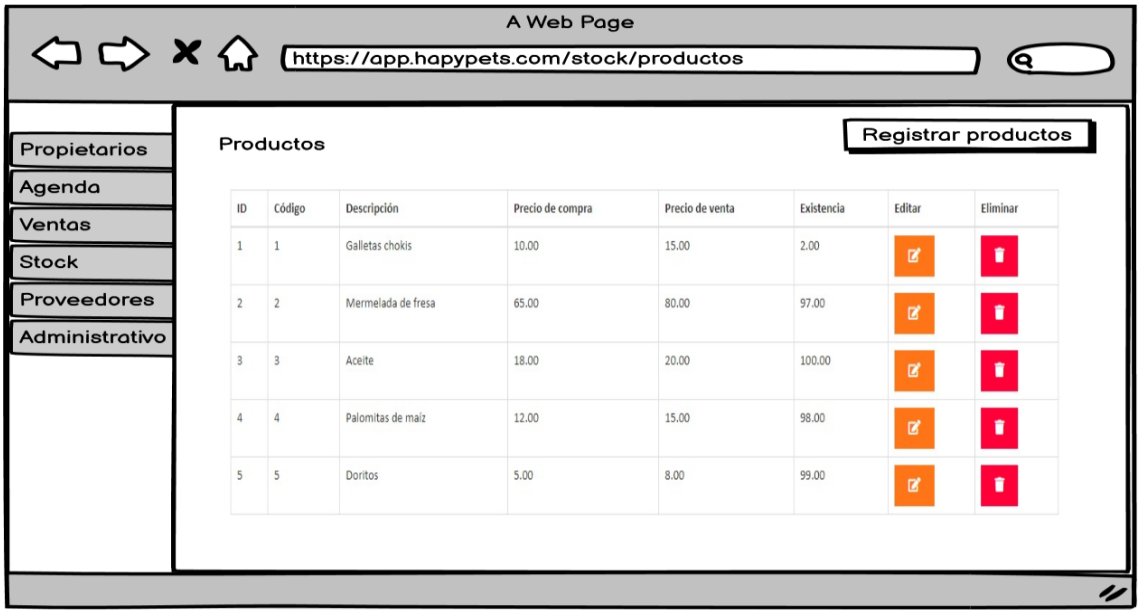


Figura 20: Diseño Productos

2.3.2. Fase II: Desarrollo

Desarrollo de Back-end

El desarrollo de cada uno de los microservicios se realizó de forma independiente, de manera similar a la construcción de una única aplicación. Cada microservicio tiene su propia estructura de carpetas y archivos, los cuales están organizados de acuerdo con lo que se muestra en la **Figura 22**.

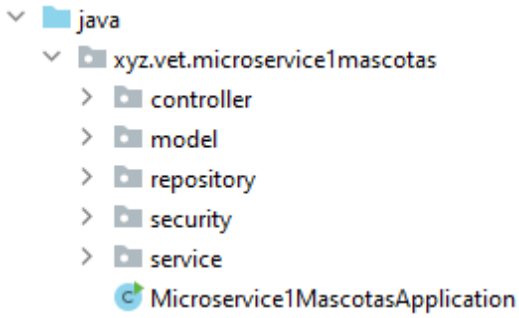


Figura 21: Directorio raíz del microservicio de mascotas--propietarios

En la Figura 21, se identifica el paquete *Models*, la cual alberga la lógica de negocio, es decir, las entidades que están relacionadas con la lógica de negocio del microservicio; el paquete *Repository* que contiene las interfaces de los modelos a implementar; el paquete *Service* que representa las operaciones que se realizó para cada modelo; el paquete *Controller* donde se encuentra la lógica

de las operaciones que se implementó y definir las rutas de la API; el paquete *Security* que contiene la seguridad del microservicio desarrollado.

Desarrollo de Front-end

El desarrollo de la parte del cliente, en este caso la aplicación web administrativa del software que interactúa con el usuario, se utilizó el framework Angular y Ionic. Son un framework de JavaScript que facilita el desarrollo de aplicaciones web del lado del cliente, proporcionando una estructura robusta y escalable.

La estructura que proporciona Angular junto con la lógica de negocio para cumplir con los requerimientos planteados se propone una estructura en la cual consta de los módulos que se va a utilizar según el diseño del prototipo, determinando los microservicios propuestos como un módulo en la estructura de angular.

Figura 22: Estructura de aplicación combinada con la lógica de negocio.

En la figura se muestra que a partir de la carpeta raíz existen subcarpetas, que estas son categorizadas como módulos y son los principales servicios de la aplicación, dentro están presente la lógica de angular que está estructurado por componentes, servicios, pipes y directivas.

A continuación, se presenta la estructura creada para el sistema:

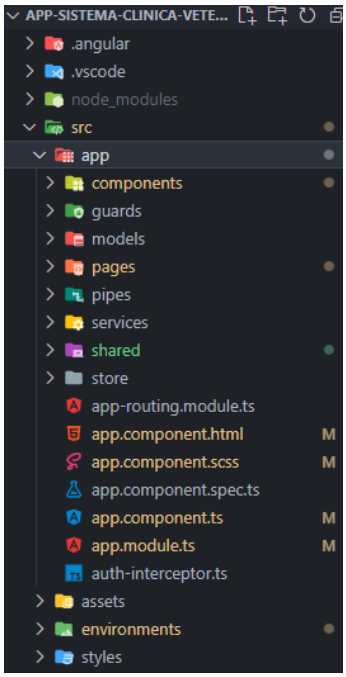


Figura 23: Estructura de aplicación.

2.4. Ejecución del prototipo

2.4.1. Fase III: Ejecución y pruebas

Para validar una arquitectura basada en microservicios deben centrarse en la funcionalidad, rendimiento y escalabilidad de cada microservicio y del sistema en su conjunto.

Según [51], un sistema basado en microservicios deben aplicarse las siguientes pruebas:

- Pruebas básicas: funcionalidad general.
- Pruebas de escala: pruebas de carga.
- Pruebas de resistencia: simular el comportamiento destructivo.

Estas pruebas realizadas se pueden observar en la sección 3.2., respecto a la evaluación del prototipo.

Ejecución de Microservicios

En la arquitectura de microservicio del sistema, cada microservicio debe ser sometido a pruebas para verificar su correcto funcionamiento. Estas pruebas se logran realizando las peticiones HTTP en los endpoints habilitados. En la figura 24, se muestra el registro de un propietario realizada para el microservicio propietario-mascota.

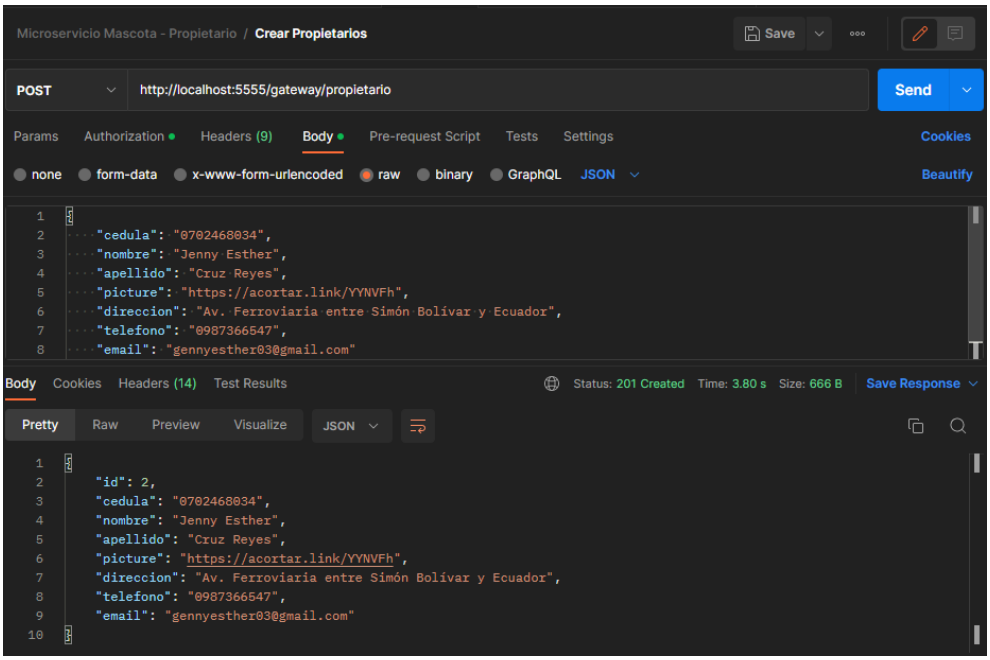


Figura 24: Creación de microservicio de propietarios-mascotas

Se llevó a cabo una solicitud GET a los registros de propietarios almacenados en la base de datos del microservicio, con el objetivo de confirmar que el registro sea preciso. En la figura 24 presenta la respuesta de la petición.

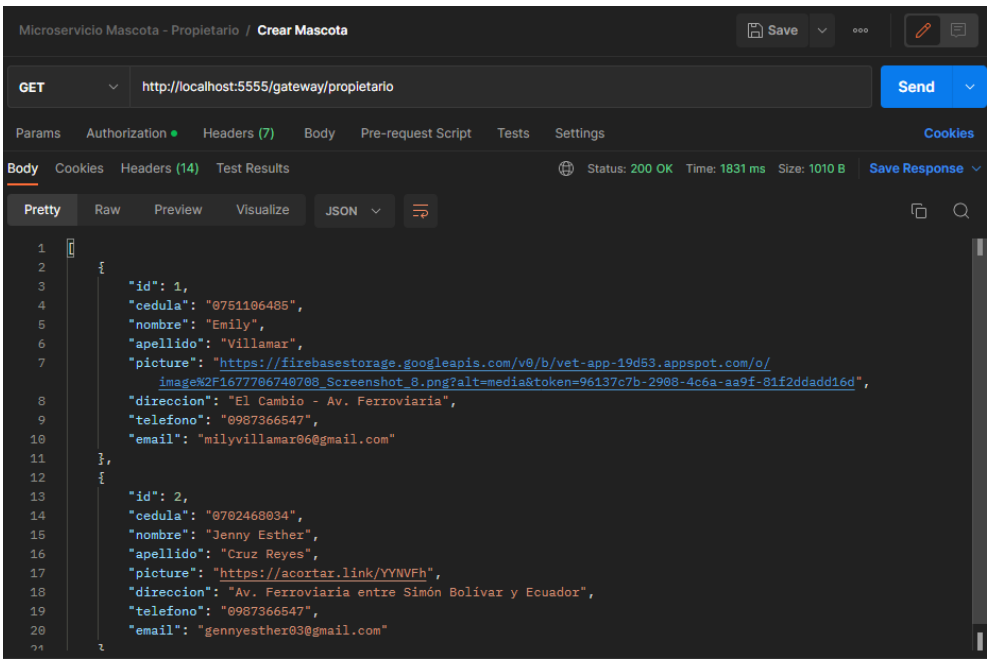


Figura 25: Respuesta de Petición GET para obtener los propietarios.

Ejecución de Front-End

La ejecución del sistema se llevó a cabo utilizando la estructura de Angular y se ajustó a los requerimientos funcionales específicos.

RQ1: Control de acceso

La ejecución del sistema se realizó de acuerdo con los parámetros necesarios para su presentación correspondiente.

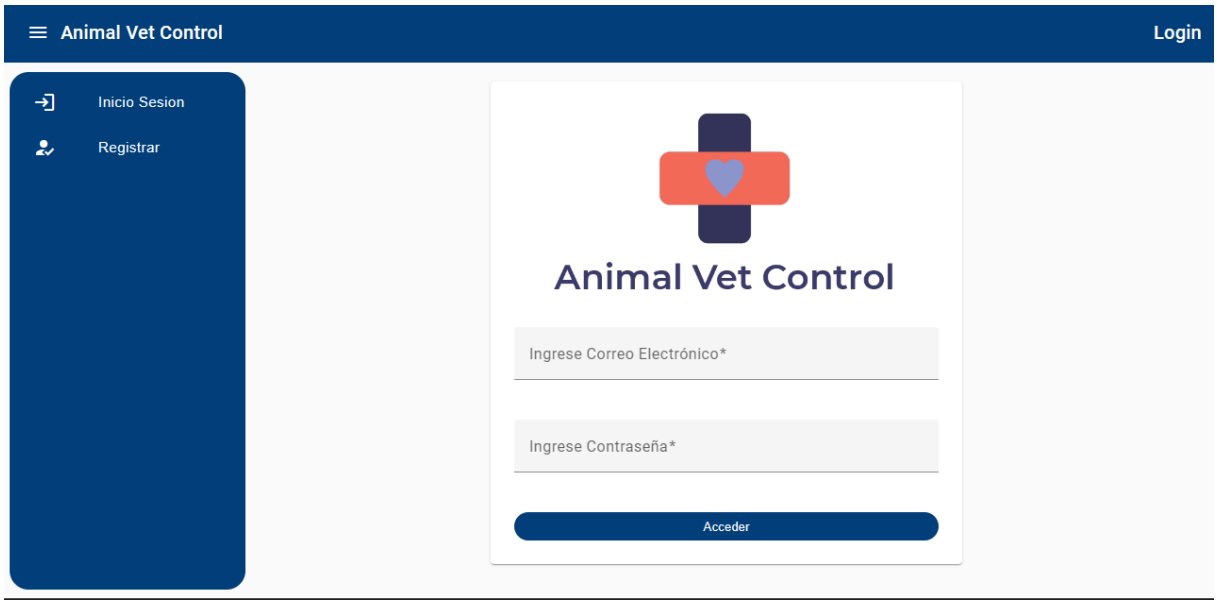


Figura 26: Ejecución de Sistema Front-End 01

RQ2: Mascotas y propietarios

En este se presenta la información principal de los propietarios y mascotas

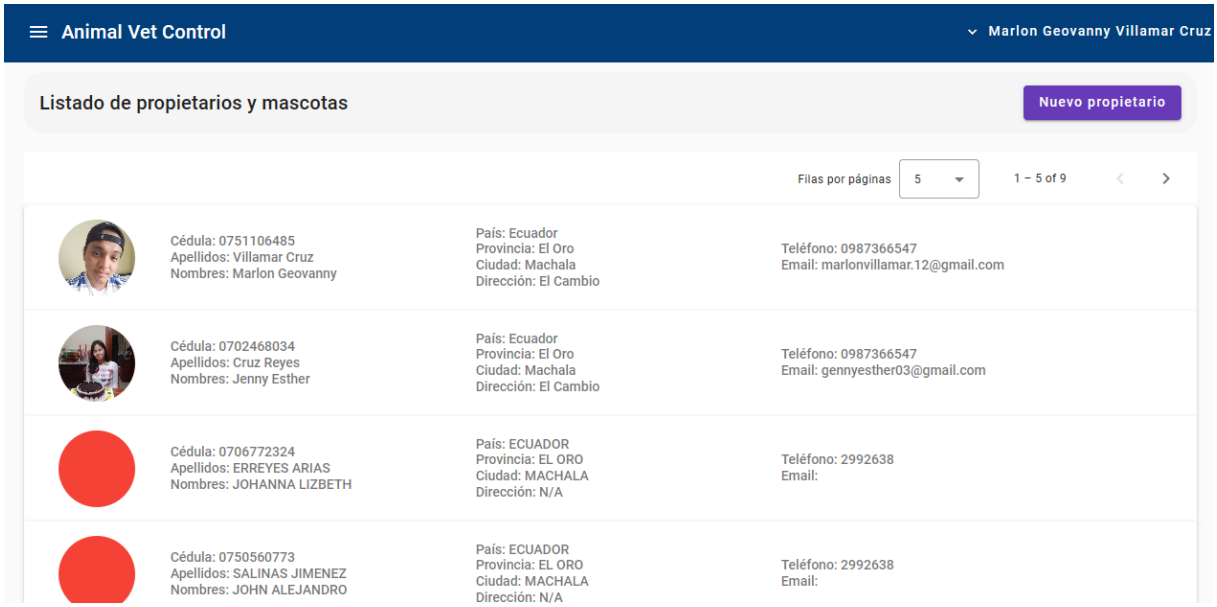


Figura 27: Ejecución de Sistema Front-End 02

2.4.2. Fase IV: Implementación

La arquitectura de microservicios se compone de varias aplicaciones. Al desplegar un microservicio, se crea una imagen de Docker mediante un archivo llamado "Dockerfile". Este archivo de configuración enumera los componentes que debe tener la imagen, junto con los comandos y herramientas necesarios para su correcta construcción. En la figura 28, presenta esta configuración para crear la imagen de Docker.

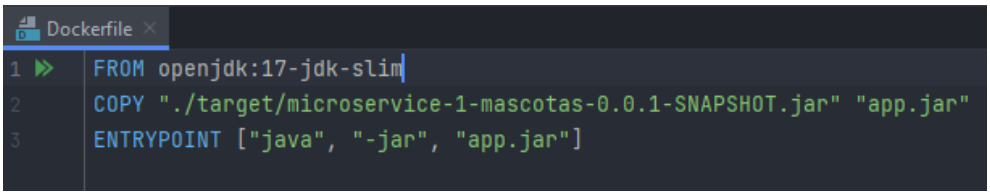


Figura 28: Dockerfile de microservicio mascotas

Para la ejecución de este archivo de configuración se utiliza el comando *docker build -t ms-1-mascotas:1.0.1.* en la terminal ubicada en la raíz del microservicio.

Una vez creada la imagen del microservicio en este caso *mascotas,* se subió a un repositorio de DigitalOcean cambiando el tag de la imagen con la dirección del repositorio y el nombre de la imagen, y se utilizó el comando *docker push registry.digitalocean.com/sysvet-registry/ ms-1-mascotas:1.0.1.*

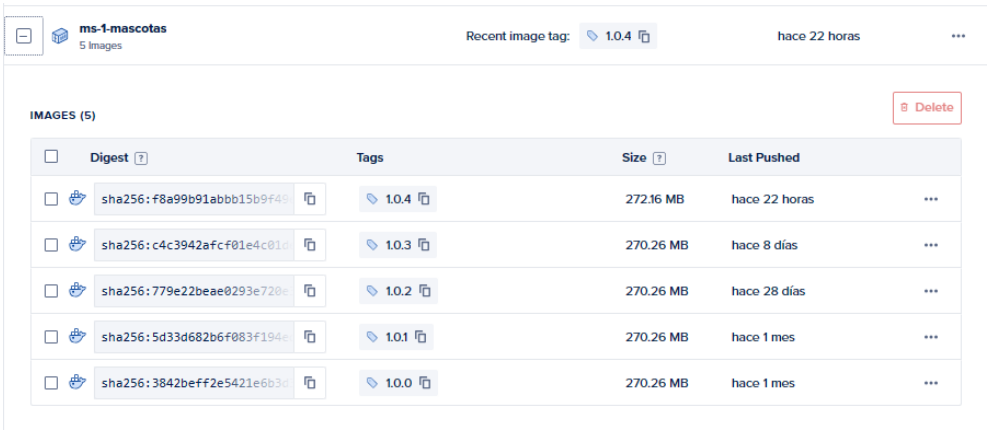


Figura 29: Repositorio sysvet-registry/ ms-1-mascotas de DigitalOcean

El siguiente paso consistió en desplegar la imagen del microservicio, en este caso se utilizó el servicio de DigitalOcean en el cual permite lanzar tu aplicación y que se ejecute a partir de la imagen creada. En la figura 30, se puede observar cada uno de los componentes que han sido desplegados para su ejecución.

Figura 30: App de *ms-control-vet* en DigitalOcean

Sin embargo, los microservicios desplegados aún no están conectados correctamente a la API Gateway. Para solucionar esto, se emplea el servicio de descubrimiento llamado *Eureka*. De manera similar a cómo funcionan los microservicios, se utiliza una imagen de Docker para ejecutar el despliegue de Eureka. Esto permite que Eureka establezca una comunicación adecuada entre los microservicios y API Gateway, lo que facilita la interacción y la ejecución de solicitudes entre los diversos componentes del sistema.

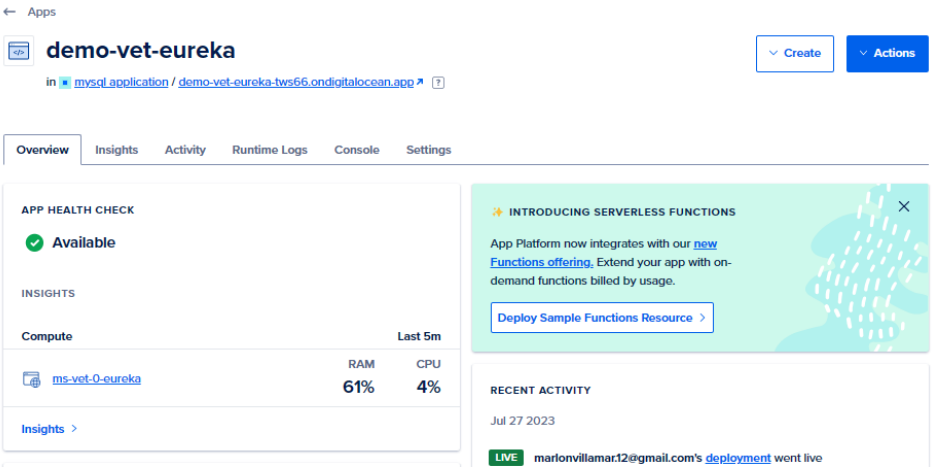


Figura 31: App de *demo-vet-eureka* en DigitalOcean

Para que los microservicios se registren en el servicio de Eureka, es necesario configurar las variables de entorno correspondientes en cada uno de ellos. Estas variables indican la dirección y el puerto donde se encuentra el servidor de Eureka, así como el nombre con el que se identificará cada microservicio. La figura 32 muestra un ejemplo de esta configuración.

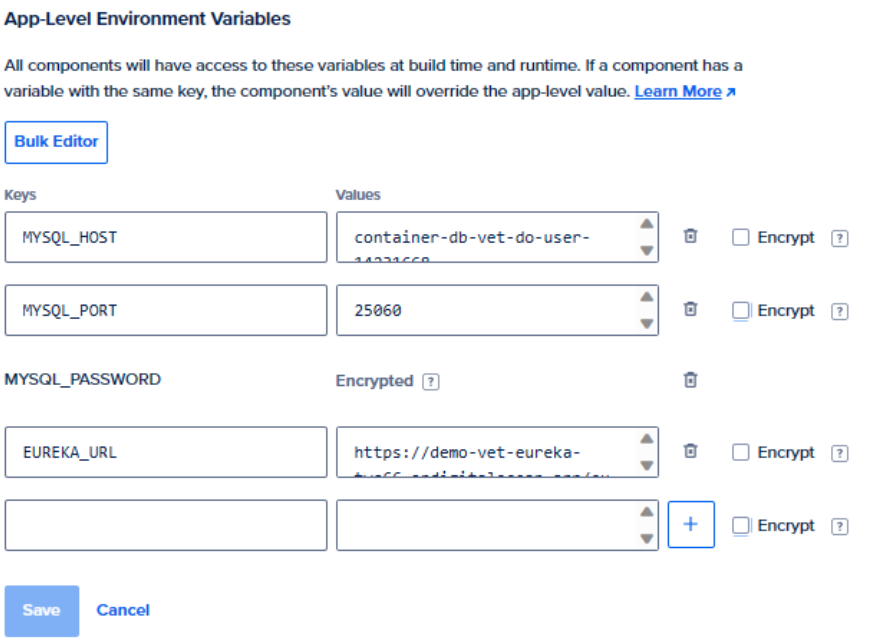


Figura 32: Variables de entorno.

Para realizar el despliegue de los demás servicios, se deben seguir los mismos pasos que se hicieron en el microservicio de mascotas, con la configuración del dockerfile, el despliegue en DigitalOcean y la definición de las variables de entorno. Estos pasos garantizan una correcta integración y funcionamiento de los microservicios en la nube. En la figura 33, se puede apreciar el funcionamiento de los microservicios en DigitalOcean.

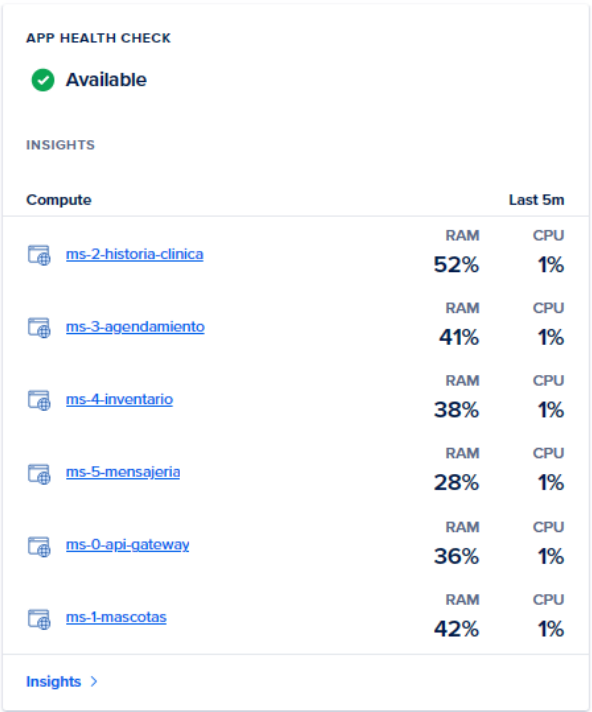


Figura 33: Comprobación del estado de los microservicios.

2.4.3. Fase V: Monitorización

El propósito de esta fase es monitorizar el rendimiento, disponibilidad y estado de los microservicios que han sido desplegados. Esto asegura que los microservicios estén en funcionamiento e identificar cualquier problema en su rendimiento.

Para ello, DigitalOcean facilita la tarea de supervisar estos microservicios a partir de su panel de administración en la cual permite ver la Salud, Actividad y los registros de estos. Esta información se encuentra ilustrada en la Figura 34.

Figura 34: Panel de control de *ms-control-vet* en DigitalOcean.

CAPÍTULO III. EVALUACIÓN DEL PROTOTIPO

3.1. Plan de evaluación

En este apartado se abordó la evaluación del software desarrollado, siguiendo la norma ISO 25010, la cual define un conjunto de características que permiten evaluar la calidad global del sistema. La evaluación se centró en aspectos como la funcionalidad, el rendimiento, entre otros, con el objetivo de garantizar que el sistema cumpla con los estándares de calidad y se adapte a las necesidades específicas de las clínicas veterinarias.

3.1.1. Objetivo de evaluación

Evaluar la calidad del producto de software del sistema y la aplicación móvil de clínica veterinaria diseñado con una arquitectura de microservicios, utilizando los criterios establecidos en la norma ISO 25010 para que cumplan con los estándares de desarrollo, permitiendo así un rendimiento eficiente y una experiencia de usuario óptima en la gestión de los procesos de la clínica veterinaria.

3.1.2. Características de calidad a evaluar

El modelo de calidad del producto establecido por la norma ISO/IEC 25010 consta de ocho características fundamentales que describen diversos aspectos de la calidad de un producto de software. Se enfocó en la evaluación las siguientes características de calidad:

a) Funcionalidad:

- Verificar el correcto funcionamiento del producto de software del sistema y la aplicación móvil en relación con las funcionalidades planificadas según los requerimientos.
- Evaluar la capacidad de los microservicios para ofrecer todas las funcionalidades requeridas, como la gestión de citas, historias médicas de mascotas.

b) Eficiencia en el rendimiento:

- Medir el tiempo de respuesta del aplicativo web.
- Medir el tiempo de respuesta y el rendimiento de los microservicios para asegurar una respuesta rápida y eficiente.

c) Fiabilidad:

- Realizar pruebas exhaustivas de funcionalidad en el producto de software del sistema para asegurar su correcto funcionamiento en condiciones normales.
- Realizar pruebas de estrés en los microservicios, sometiéndolos a cargas de trabajo intensivas para garantizar que mantengan su funcionalidad y rendimiento.

d) Portabilidad:

- Verificar la capacidad del software para ser desplegado y ejecutado en diferentes entornos y plataformas, asegurando una portabilidad adecuada.

3.1.3. Métricas y criterios de evaluación

a) Funcionalidad:

Métrica: Porcentaje de funcionalidades implementadas correctamente.

Criterio de evaluación: El sistema debe cumplir con todos los requisitos funcionales definidos, y las funcionalidades implementadas deben estar libres de errores y funcionar correctamente en todos los componentes, incluyendo microservicios y aplicativo móvil.

Herramientas:

- Encuestas de Evaluación de Usuarios.
- Para evaluar la funcionalidad de los microservicios, uso de la herramienta Postman.

b) Rendimiento:

Métrica: Tiempo de respuesta promedio de las acciones del usuario.

Criterio de evaluación: El sistema completo, incluyendo microservicios y aplicativo móvil, debe responder de manera rápida y eficiente a las acciones del usuario, sin retrasos significativos que puedan afectar negativamente la experiencia del usuario.

Herramientas:

- Apache JMeter para pruebas de carga y rendimiento en los microservicios.
- Website Grader para evaluación del rendimiento web.

c) Fiabilidad:

Métrica: Tiempo medio entre fallos (MTBF, por sus siglas en inglés).

Criterio de evaluación: El MTBF debe ser lo suficientemente alto para garantizar un funcionamiento estable y confiable.

Herramientas:

- Apache JMeter para pruebas de estrés.

d) Portabilidad:

Métrica: Número de plataformas compatibles.

Criterio de evaluación: El sistema debe poder ejecutarse en diferentes plataformas, como navegadores y dispositivos.

Herramientas: Pruebas manuales en diferentes plataformas y herramientas de emulación.

3.1.4. Cronograma

El cronograma de actividades para la planificación de la evaluación se llevó a cabo desde el 14 de agosto, hasta el 10 de septiembre del 2023.

Tabla 11: Cronogramas de actividades de evaluación

Actividades	Del 14 al 20 de Agosto	Del 21 al 27 de Agosto	Del 28 de Agosto al 03 de Septiembre	Del 04 al 10 de septiembre
Creación de cuestionarios para las encuestas de evaluación de usuarios.				
Ejecución de encuestas de evaluación de usuarios.				
Evaluación de la funcionalidad de los microservicios.				
Configuración de pruebas de carga y rendimiento en microservicios (Apache JMeter).				
Ejecución de pruebas de carga y rendimiento.				
Evaluación del tiempo de respuesta y rendimiento web (Website Grader).				
Ejecución de pruebas de estrés en microservicios.				
Pruebas manuales en diferentes plataformas (navegadores, dispositivos).				

3.1.5. Proceso de evaluación

3.1.9.1.Evaluación de funcionalidad de producto de software

Se realizó pruebas de funcionamiento del producto software (aplicativo web y móvil) con enfoque a la satisfacción del usuario al usar dicho producto y como principal objetivo el de verificar el correcto funcionamiento según sus requerimientos funcionales.

Método de evaluación

Estas pruebas se realizaron mediante encuestas tanto para usuarios como para especialistas (véase Anexo 5 y Anexo 6) donde se plantearon preguntas relacionadas con la satisfacción en el funcionamiento del aplicativo web y móvil con respecto a los requisitos funcionales establecidos.

Ejecución de pruebas de funcionalidad

Estas pruebas se las llevó a cabo con una cantidad especifica de especialistas y usuarios. Se les proporcionó acceso al aplicativo web y móvil, permitiéndoles utilizarlo antes de completar una encuesta de satisfacción.

Para el aplicativo web, se seleccionaron seis especialistas a quienes se les compartió el enlace del sitio y las credenciales necesarias para acceder y utilizar la plataforma. Después de su experiencia de uso, se les proporcionó una encuesta para que evaluaran el funcionamiento del sitio de acuerdo con su criterio.

En el caso del aplicativo móvil, se eligieron veinticinco usuarios a quienes se les proporcionó el archivo APK para que lo instalaran en sus smartphones. Tras utilizar la aplicación y experimentar su funcionalidad, se les facilitó un enlace para que completaran una encuesta de satisfacción, expresando su opinión sobre el funcionamiento del aplicativo móvil y respondiendo a las preguntas correspondientes.

3.1.9.2.Pruebas de funcionalidad en los microservicios.

Se realizó pruebas de funcionalidad en cada microservicio para asegurar su correcto funcionamiento. Estas pruebas tienen como objetivo principal garantizar que cada microservicio cumpla con sus respectivos requerimientos funcionales y que operara de manera adecuada.

Método de evaluación

Las pruebas de funcionalidad se realizaron a través de peticiones a los endpoints expuestos por cada uno de los microservicios. Para esto, se emplearon herramientas como Postman, que facilitaron la creación y ejecución de las peticiones HTTP necesarias.

Identificación de funcionalidades de cada uno de los microservicios según los requisitos funcionales:

Durante el proceso de evaluación, se analizaron las funcionalidades de cada uno de los microservicios. Estas funcionalidades fueron seleccionadas en base a los requerimientos funcionales previamente definidos.

A continuación, se presenta una lista de los microservicios y las funcionalidades asociadas que se identificaron:

Tabla 12: Funcionalidades identificadas según los requisitos funcionales

FUNCIONALIDADES IDENTIFICADAS		
RF01	Control de acceso	• Autenticación de usuarios • Creación de usuarios • Lista de usuarios • Obtener usuario autenticado • Cambiar roles • Eliminar usuarios
TOTAL		6 funciones
RF02	Gestión de mascotas y dueños	• Registro de nuevos propietarios • Actualizar data de propietarios

		• Lista de propietarios • Obtener propietario específico • Eliminar propietario • Registro de nuevas mascotas • Actualizar mascotas • Lista de mascotas • Búsqueda de mascota por propietario • Eliminar mascotas • Obtener mascota especifico
TOTAL		11 funciones
RF03	Gestión de historias clínicas	• Registrar, actualizar, eliminar y obtener historias. • Registrar, actualizar, eliminar y obtener vacunaciones • Registrar, actualizar, eliminar y obtener desparasitaciones • Registrar, actualizar, eliminar y obtener exámenes (evidencias de historias) • Búsquedas de todas las historias, vacunaciones, desparasitaciones y exámenes de las mascotas. • Búsquedas de exámenes por la historia. • Obtener certificado de vacunación en pdf.
TOTAL		22 funciones
RF04	Gestión de agendamiento de citas.	• Crear, actualizar, obtener y eliminar citas. • Búsqueda de citas por propietarios y mascotas • Obtener todas las citas
TOTAL		6 funciones
RF05	Gestión de inventario	• Crear, actualizar, obtener y eliminar categorías de productos. • Crear, actualizar, obtener y eliminar presentaciones de productos. • Crear, actualizar, obtener y eliminar proveedores. • Crear, actualizar, obtener y eliminar productos.
TOTAL		16 funciones
RF06	Gestión de mensajería.	• Enviar correos de avisos a propietarios con los detalles de la cita programada.
TOTAL		1 función

En la Tabla 12, se presentan las funciones que se esperan de los microservicios, las cuales se han identificado en base a los requerimientos funcionales. En total, se han identificado un conjunto de 62 funciones.

Ejecución de pruebas de funcionalidad

Basándonos en la información proporcionada en la Tabla 12, se procedió a llevar a cabo las pruebas de los microservicios utilizando las herramientas correspondientes (véase Anexo 7).

3.1.9.3.Pruebas de rendimiento en los microservicios

Se llevaron a cabo pruebas de rendimiento en los microservicios para evaluar su capacidad de respuesta bajo diferentes cargas de trabajo para asegurar que los microservicios funcionaran eficientemente.

Método de evaluación

Para simular diferentes niveles de carga en el sistema, se empleó la herramienta Apache JMeter. Se realizaron las siguientes configuraciones:

Tabla 13: Configuración de cargas para JMeter

Nombre de la carga	Números de Hilos (Usuarios)	Periodo de subida (en segundos)
Carga ligera	100	60
Carga moderada	500	30
Carga pesada	1000	15

En la Tabla 13, se muestra la configuración de las cargas, el primer grupo de hilos con la configuración de 100 usuarios con un período de subida de 60 segundos para simular una carga ligera. El segundo grupo de hilos con 500 usuarios con un período de subida de 30 segundos para simular una carga moderada. Y en el tercer grupo con 1000 usuarios y un período de subida de 15 segundos para simular una carga pesada.

Ejecución de pruebas de cargas

Las pruebas de carga se realizaron de acuerdo con las configuraciones previamente definidas. Se llevaron a cabo diversas acciones para simular situaciones de uso real.

Estas pruebas permitieron evaluar cómo los microservicios respondían bajo diferentes niveles de carga y analizar su rendimiento en términos de tiempo de respuesta y capacidad para manejar usuarios concurrentes.

3.1.9.4.Evaluación del tiempo de respuesta y rendimiento web.

El aplicativo web de la clínica veterinaria debe ser eficiente y responder de manera rápida y efectiva a las acciones realizadas por los usuarios. Con este fin, se llevó a cabo una evaluación del tiempo de respuesta y el rendimiento web, con el objetivo de medir y garantizar la eficiencia del aplicativo.

Método de evaluación

Para llevar a cabo esta evaluación, se utilizó la herramienta Website Grader, ampliamente reconocida por su capacidad para analizar diversos aspectos clave de las aplicaciones web, incluyendo el rendimiento, la seguridad y el diseño.

En este caso, la herramienta se aplicó al sistema web de la clínica veterinaria, con el propósito de identificar tanto sus fortalezas como sus debilidades, con un enfoque especial en el aspecto del rendimiento.

Durante el análisis se evaluó el tamaño de la página, solicitudes de páginas, velocidad de la página, caché web, redireccionamientos de páginas mínimos, tamaño de la imagen, javascript minimizado y css minimizado.

3.1.9.5.Pruebas de estrés en los microservicios

Este proceso de evaluación es para medir la capacidad de los microservicios para mantener un rendimiento estable y una operación confiable bajo diversas condiciones.

Método de evaluación

Las pruebas de estrés se realizaron utilizando la herramienta JMeter, en la cual simuló los mismos escenarios que se presentaron en las pruebas de carga. Esto permitió evaluar el tiempo medio entre fallos (MTBF) y la capacidad de recuperación ante fallos de los microservicios.

En la herramienta se realizó las siguientes configuraciones para realizar las pruebas:

Tabla 14: Configuración de pruebas de estrés para JMeter

Nombre de la carga	Números de Hilos (Usuarios)	Periodo de subida (en segundos)
Carga ligera	100	60
Carga moderada	500	30
Carga pesada	1000	15

Los escenarios presentados en la Tabla 13, simula pruebas de carga en la cual se verificó el tiempo de respuesta, porcentaje de error y los envíos de recursos.

3.1.9.6.Pruebas de portabilidad

Las pruebas de portabilidad tienen como objetivo verificar la capacidad del software para desplegarse y ejecutarse eficazmente en diversos entornos y plataformas. Esto es fundamental para asegurar que el software sea accesible y funcional para una amplia audiencia de usuarios, independientemente del sistema operativo o navegador que utilicen.

Método de evaluación

Para llevar a cabo la evaluación de portabilidad, se siguió el siguiente método:

- **Selección de Navegadores Web**
  Se eligieron los navegadores web más utilizados por la población objetivo, incluyendo Chrome, Edge, Brave y Firefox. Estos navegadores se consideran representativos de los diferentes entornos de navegación utilizados por los usuarios finales.

- **Selección de Dispositivos Móviles:**

  Para evaluar la portabilidad del aplicativo móvil, se probaron dispositivos móviles con diferentes versiones del sistema operativo Android. Estos dispositivos se seleccionaron para representar una variedad de configuraciones comunes utilizadas por los usuarios.

Ejecución de la evaluación

Se ejecutaron pruebas en cada uno de los navegadores y dispositivos seleccionados. Durante estas pruebas, se verificó la funcionalidad completa del sistema web y aplicativo móvil, así como la adaptabilidad del diseño a las características específicas de cada navegador o dispositivo móvil. Se marcó cada navegador y dispositivo móvil como "Funcional" si todas las pruebas se completaron con éxito.

3.2. Resultados de la evaluación

3.2.1. Resultados de evaluación de funcionalidad por encuestas

Este proceso se llevó a cabo en relación con los requerimientos planteados en la Tabla 7. Para cada ámbito (web y móvil), se formularon preguntas destinadas a conocer el nivel de satisfacción del usuario al experimentar con el sistema. De esta manera, se obtuvo una retroalimentación valiosa que ayudó a evaluar la funcionalidad del producto de software.

Aplicativo Web - Resultados de la Encuesta:

Para la encuesta del aplicativo web se obtuvieron los siguientes resultados por cada una de sus preguntas:

Pregunta 1: ¿Cuál es su grado de satisfacción con respecto al sistema web y su funcionalidad con los requisitos funcionales (Control de acceso, información de propietarios y mascotas, historias clínicas, inventario, agendamiento de citas y mensajería)?

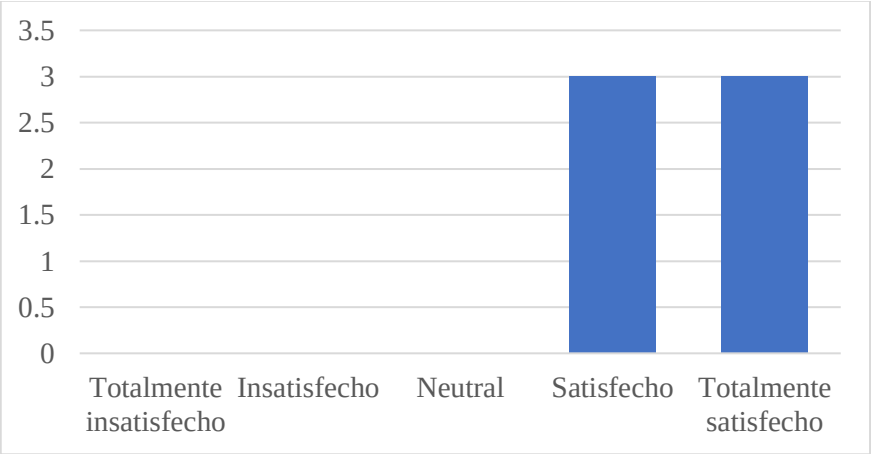


Figura 35: Gráfica de resultados de la pregunta 1 de encuesta del aplicativo web

Análisis

En la Figura 35 se puede evidenciar que existe un alto nivel de satisfacción respecto al control de acceso para el aplicativo web el cual contiene los datos para ingresar, la información del propietario y su respectiva mascota cargada en el sistema, así como sus respectivas historias clínicas, agendar citas e inventario de la clínica.

Pregunta 2: En cuanto al Control de acceso, ¿qué nivel de satisfacción tienes respecto a la autenticación para el control de los recursos del sistema?

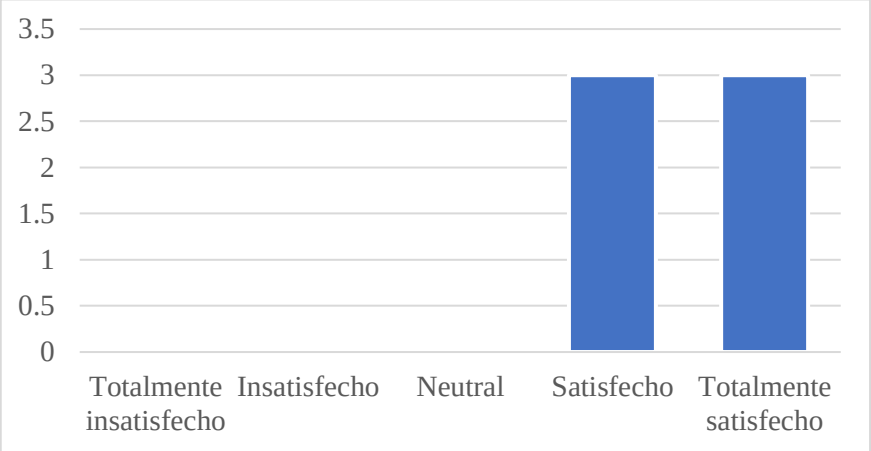


Figura 36: Gráfica de resultados de la pregunta 2 de encuesta del aplicativo web

Análisis

Para la autenticación de los datos ingresados en el aplicativo web y a su vez usarlos para la gestión de la clínica se obtuvo como resultado un alto nivel de satisfacción tal como se lo puede ver en la Figura 36.

Pregunta3: Indique el grado de satisfacción respecto al cumplimiento de la Gestión de propietarios y mascotas.

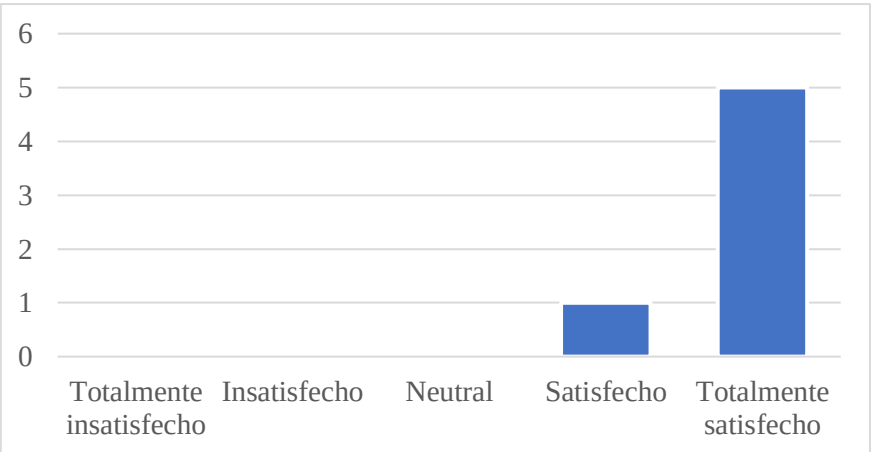


Figura 37: Gráfica de resultados de la pregunta 3 de encuesta del aplicativo web

Análisis

En la gestión de propietarios y mascotas que implica agregar y modificar los datos tanto para los propietarios y sus respectivas mascotas en la Figura 37 se puede notar un gran nivel de satisfacción por lo que sí existe conformidad en que se pueda realizar estas funciones dentro del sitio web.

Pregunta 4: ¿Encuentras que la información de las mascotas y sus propietarios está organizada de una manera clara y fácil de visualizar?

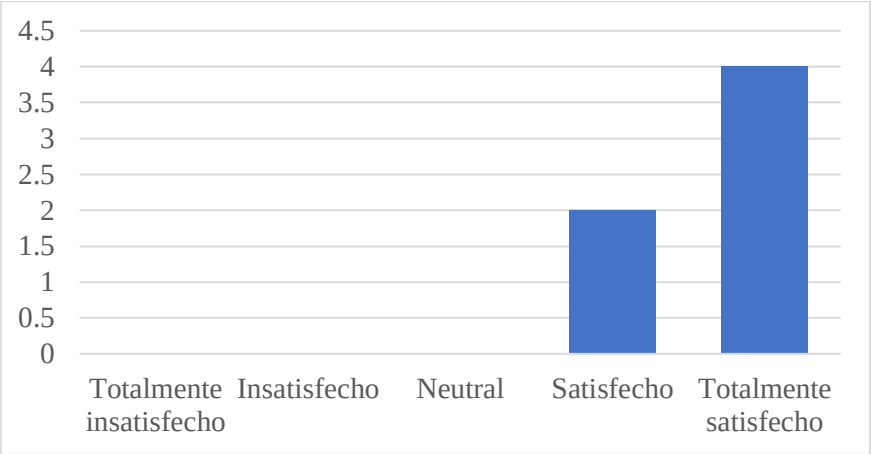


Figura 38: Gráfica de resultados de la pregunta 4 de encuesta del aplicativo web

Análisis

En la Figura 38 respecto al orden de cómo se encuentra organizada la información dentro del aplicativo web, existe un alto nivel de satisfacción a que dicha información sea visualizada de manera ordenada y clara.

Pregunta 5: En relación a la Gestión de historias clínicas, ¿qué nivel de satisfacción tienes respecto al software para permitirte visualizar las historias clínicas (consultas, vacunaciones y desparasitaciones) de las mascotas?

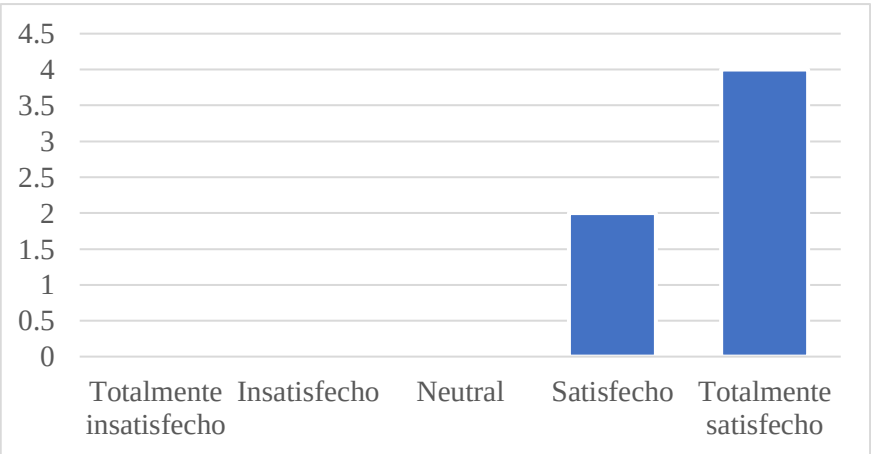


Figura 39: Gráfica de resultados de la pregunta 5 de encuesta del aplicativo web

Análisis

En la visualización de las historias clínicas dentro del aplicativo web, se obtuvo como resultado un alto nivel de satisfacción tal y como se muestra en la Figura 39 respecto a la visualización de las consultas, vacunaciones y desparasitaciones de la mascota registrada.

Pregunta 6: En lo que respecta a la Gestión de citas, ¿qué nivel de satisfacción tienes para realizar el agendamiento de citas y visualizarlas?

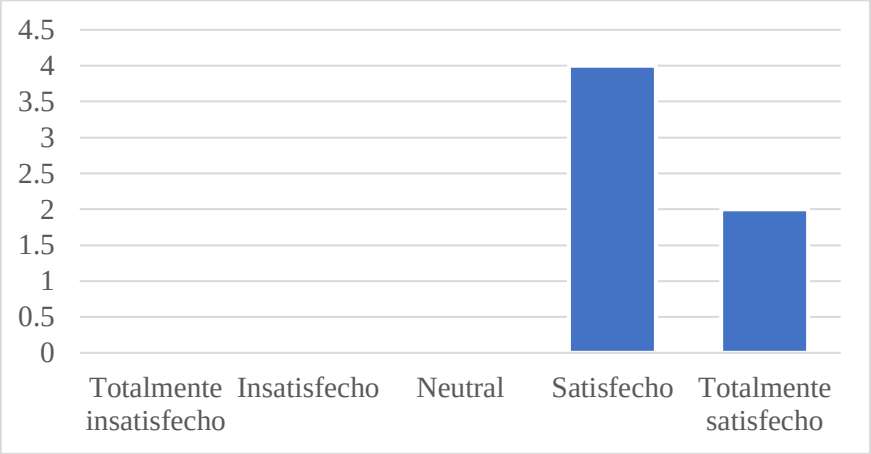


Figura 40: Gráfica de resultados de la pregunta 6 de encuesta del aplicativo web

Análisis

En la Figura 40 se demuestra que el grado de satisfacción es alto con respecto a la opción de agendamiento de citas y su respectiva visualización dentro del sistema web.

Pregunta 7: ¿Cuál es el nivel de satisfacción que tiene el software para registrar presentaciones, categorías, proveedores y productos en la gestión de inventario?

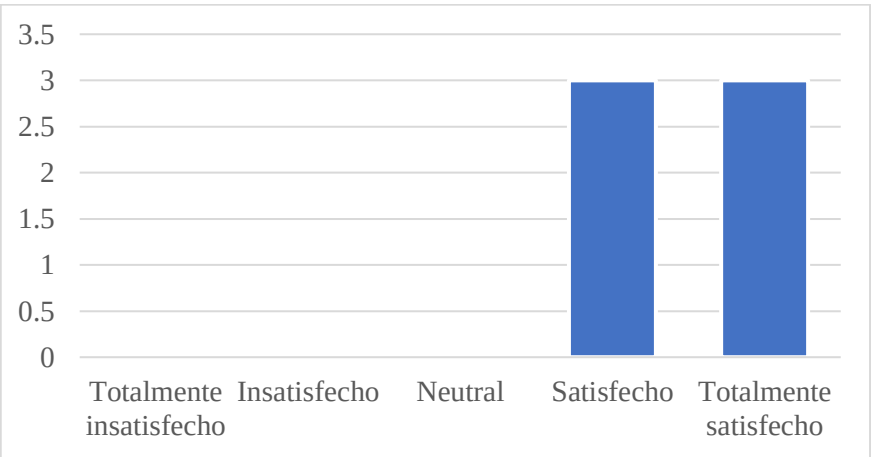


Figura 41: Gráfica de resultados de la pregunta 7 de encuesta del aplicativo web

Análisis

En lo que respecta al registro de presentaciones, categorías, proveedores y productos en la gestión de inventario, existe un grado de satisfacción positivo dentro del aplicativo web tal y como se lo muestra en la Figura 41.

Pregunta 8: En la gestión de mensajería, ¿qué nivel de satisfacción tienes con la acción del software de enviar notificaciones al correo de aviso de cita un día antes de la programada?

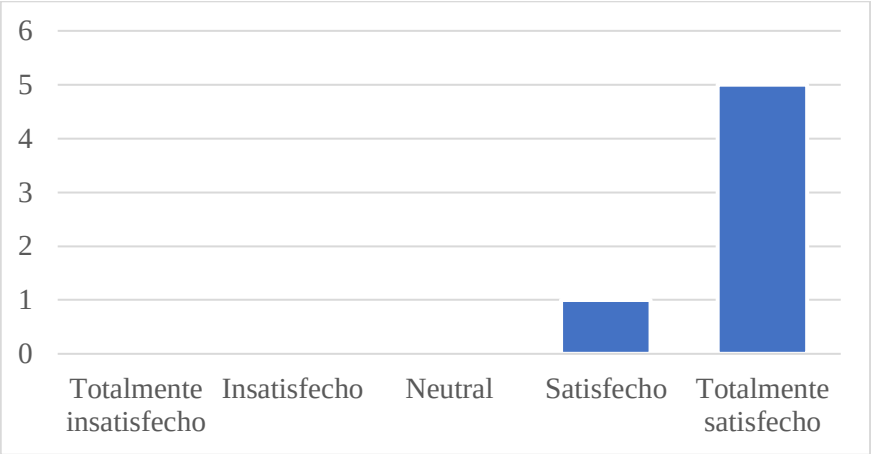


Figura 42: Gráfica de resultados de la pregunta 8 de encuesta del aplicativo web

Análisis

Existe un alto nivel de satisfacción en los resultados de la Figura 42 referente a la opción que permite enviar notificaciones al correo electrónico personal del propietario de la mascota un día antes de la cita médica en la clínica.

Aplicativo Móvil - Resultados de la Encuesta:

Pregunta 1: ¿Cuál es su grado de satisfacción respecto a los datos utilizados y su autenticación para el inicio de sesión de la aplicación móvil?

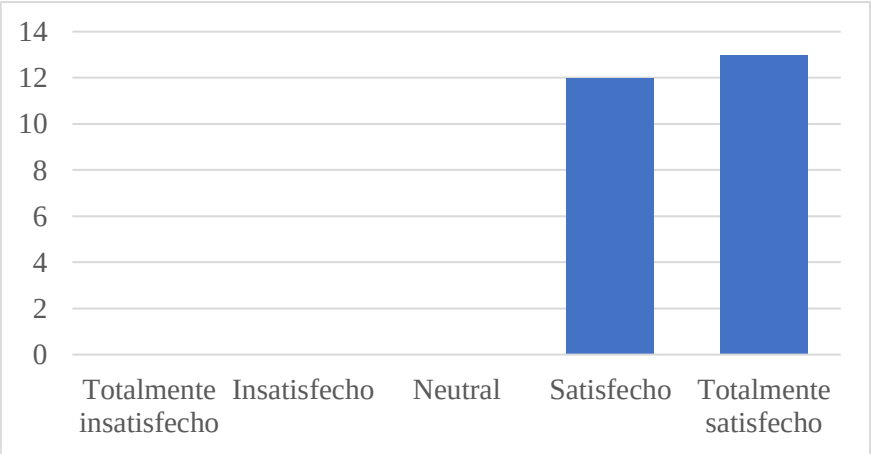


Figura 43: Gráfica de resultados de la pregunta 1 de encuesta del aplicativo móvil

Análisis

Para los datos utilizados y su autenticación en el inicio de sesión en el aplicativo móvil, existe un nivel de satisfacción alto respecto a esta acción tal y como se lo refleja en los resultados de la Figura 43.

Pregunta 2: Con respecto a la apariencia de la aplicación móvil, ¿Qué tan amigable es la aplicación respecto al diseño y operatividad en su uso?

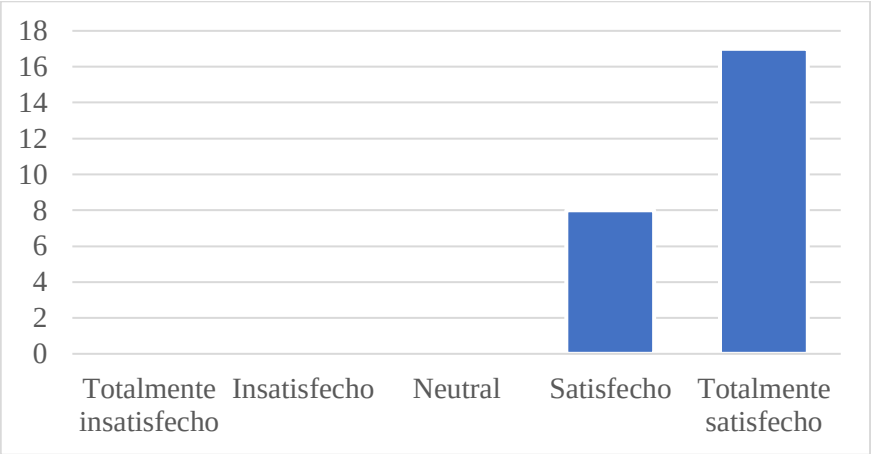


Figura 44: Gráfica de resultados de la pregunta 2 de encuesta del aplicativo móvil

Análisis

En la Figura 44 se refleja que existe un alto nivel de satisfacción de los usuarios respecto al diseño y uso del aplicativo móvil.

Pregunta 3: Indique su grado de satisfacción respecto al registro de un nuevo usuario desde la aplicación móvil.

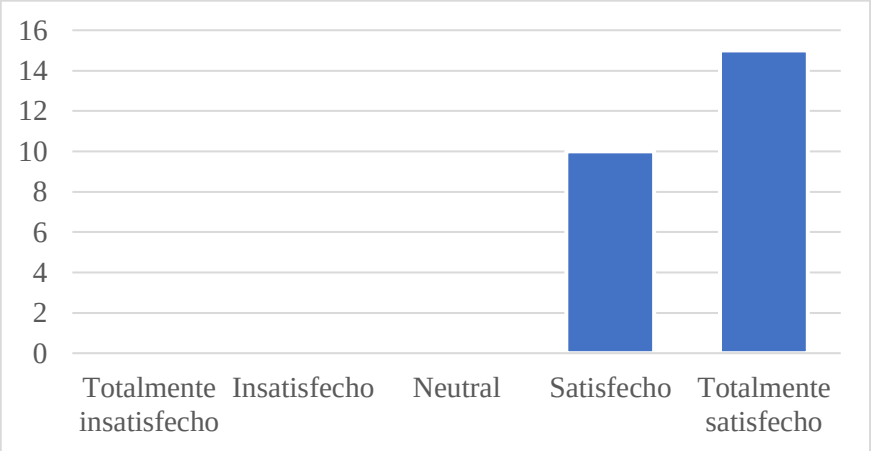


Figura 45: Gráfica de resultados de la pregunta 3 de encuesta del aplicativo móvil

Análisis

Para la opción que permite registrar nuevos usuarios desde el aplicativo móvil en la Figura 45 se demuestra que existe un nivel de satisfacción elevado con respecto a esta función.

Pregunta 4: La aplicación móvil permite editar y visualizar la información (vacunaciones y desparasitaciones) de la mascota, ¿Está usted conforme con esta opción?

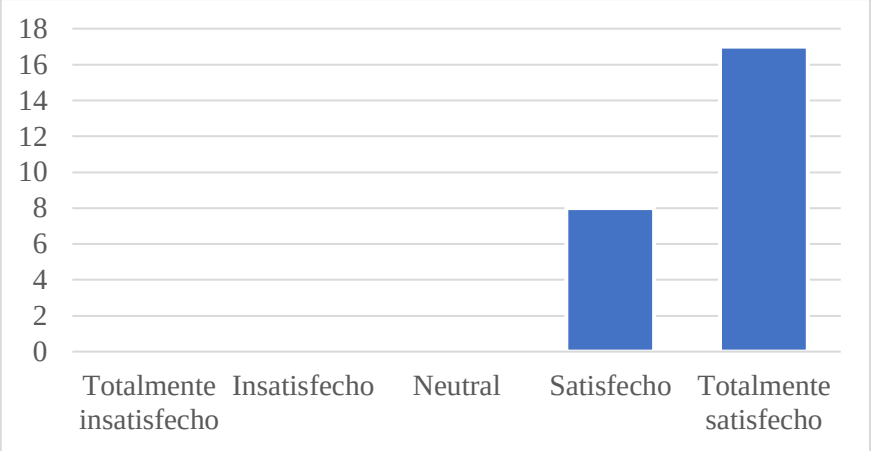


Figura 46: Gráfica de resultados de la pregunta 4 de encuesta del aplicativo móvil

Análisis

En la Figura 46 vemos que existe un gran nivel de satisfacción respecto a que el aplicativo móvil le permita editar y visualizar la información de la mascota registrada en el sistema.

Pregunta 5: Respecto a la opción de agendar citas médicas para su mascota desde la aplicación móvil, ¿Está usted de acuerdo que se permita esta opción?

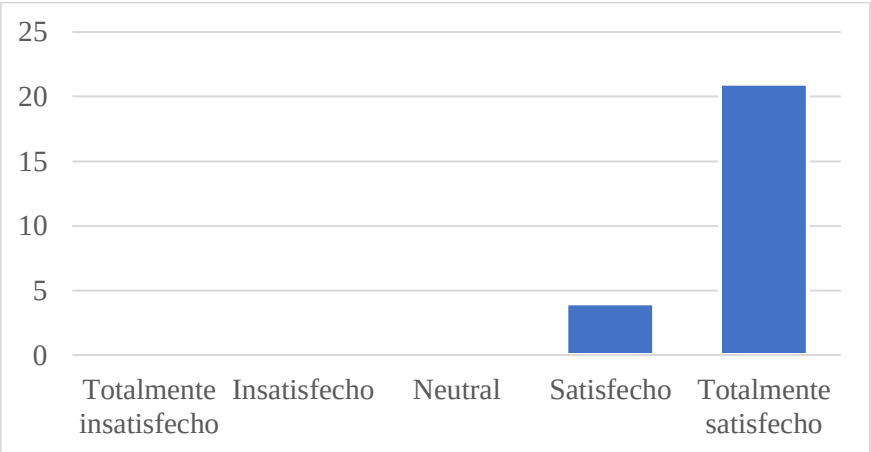


Figura 47: Gráfica de resultados de la pregunta 5 de encuesta del aplicativo móvil

Análisis

Se refleja un gran nivel de satisfacción respecto a que desde el aplicativo móvil el usuario tenga la posibilidad de agendar una cita en la clínica veterinaria tal y como lo refleja los resultados en la Figura 47.

3.2.2. Resultados de pruebas de funcionalidad de los microservicios

Durante el proceso de evaluación de funcionalidad de los microservicios, se identificaron un total de 62 funcionalidades distribuidas en los microservicios.

- Microservicio Control de acceso: 6 funciones
- Microservicio Propietario – Mascota: 11 funciones
- Microservicio Historia clínicas: 22 funciones
- Microservicio Agendamiento: 6 funciones
- Microservicio Inventario: 16 funciones
- Microservicio Mensajería: 1 función

Para calcular el porcentaje de funcionalidades implementadas correctamente, se utilizó la siguiente fórmula:

$$\left(\frac{Número\ de\ funcionalidades\ implementadas\ correctamente}{Total\ de\ funcionalidades\ identificadas}\right) \times 100$$

Se calcularon los porcentajes de funcionalidades implementadas correctamente en cada uno de los microservicios. A continuación, se presentan los resultados:

Microservicio: Control de acceso

- Número de funcionalidades implementadas correctamente: 6
- Total de funcionalidades Identificadas: 6

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{6}{6}\right) \times 100 = 100\%$

Microservicio: Gestión de propietarios y mascotas

- Número de funcionalidades implementadas correctamente: 11
- Total de funcionalidades Identificadas: 11

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{11}{11}\right) \times 100 = 100\%$

Microservicio: Gestión de historias clínicas

- Número de funcionalidades implementadas correctamente: 22
- Total de funcionalidades Identificadas: 22

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{22}{22}\right) \times 100 = 100\%$

Microservicio: Gestión de agendamiento de citas

- Número de funcionalidades implementadas correctamente: 6
- Total de funcionalidades Identificadas: 6

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{6}{6}\right) \times 100 = 100\%$

Microservicio: Gestión de inventario

- Número de funcionalidades implementadas correctamente: 16
- Total de funcionalidades Identificadas: 16

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{16}{16}\right) \times 100 = 100\%$

Microservicio: Gestión de mensajería

- Número de funcionalidades implementadas correctamente: 1
- Total de funcionalidades Identificadas: 1

Porcentaje de Funcionalidades Implementadas Correctamente = $\left(\frac{1}{1}\right) \times 100 = 100\%$

En todos los microservicios evaluados, se logró un porcentaje del 100% de funcionalidades implementadas correctamente. Esto indica que cada uno de los microservicios se implementaron correctamente y funcionan de acuerdo con los requisitos funcionales.

3.2.3. Resultados de pruebas de carga de los microservicios

Las pruebas se llevaron a cabo en tres configuraciones distintas: carga ligera, carga moderada y carga pesada, cada una con un número específico de usuarios concurrentes. Los resultados de las cargas se detallan en el Apéndice 1.

A continuación, se presentan los resultados detallados de estas pruebas, como el tiempo de respuesta y rendimiento.

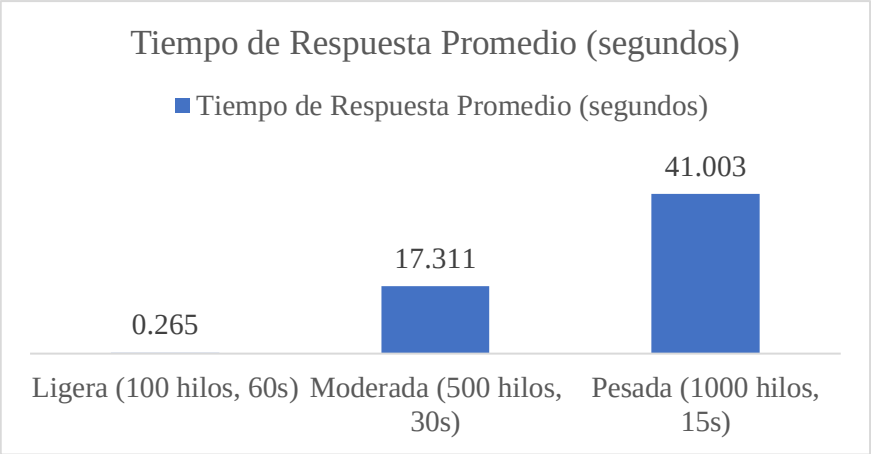


Figura 48: Gráfica de porcentaje de tiempo de respuesta

En la Figura 48, se observa que a medida que aumenta la carga, el incremento en el tiempo de respuesta es relativamente pequeño. Aunque un mayor número de carga puede resultar en tiempos de respuesta ligeramente mayores, estos aumentos no son significativos.

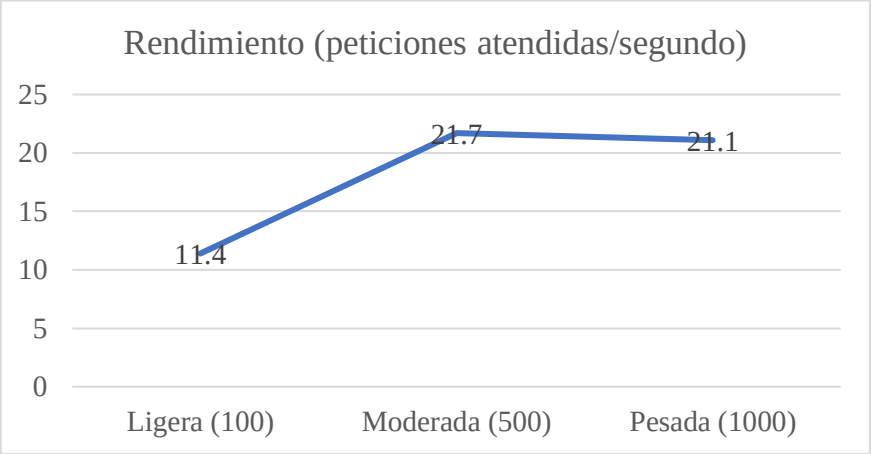


Figura 49: Gráfica de rendimiento

En la Figura 49, se observa que los microservicios son eficientes en cuanto a rendimiento. El rendimiento es relativamente mejor, lo que indica que un mayor número de peticiones son atendidas de manera inmediata.

Análisis de resultados de Carga Ligera:

Durante la configuración de carga ligera, con 100 usuarios concurrentes, se observaron tiempos de respuesta bajos en general. Las acciones presentaron tiempos de respuesta adecuados, y el porcentaje de error fue menor. Los resultados indican que el sistema es eficiente bajo cargas ligeras y proporciona una experiencia de usuario satisfactoria.

Análisis de resultados de Carga Moderada:

En la configuración de carga moderada, con 500 usuarios concurrentes, se registró un aumento en los tiempos de respuesta, dentro de límites aceptables. Las acciones del sistema, incluyendo "Listar propietarios" y "Crear mascota", mantuvieron un porcentaje de error mínimo. Estos resultados sugieren que el sistema puede manejar cargas moderadas de manera efectiva, proporcionando una experiencia de usuario razonable.

Análisis de resultados de Carga Pesada:

Con la configuración de carga pesada, que involucró a 1000 usuarios concurrentes, se observaron tiempos de respuesta significativamente mayores. A pesar del aumento en los tiempos de respuesta, el porcentaje de error se mantuvo en niveles bajos. El rendimiento global del sistema fue alto, lo que indica que es capaz de manejar una gran cantidad de solicitudes por segundo incluso bajo cargas pesadas.

En la medición de tiempo de respuesta para cada carga, se nota una elevación en el tiempo de respuesta

3.2.4. Resultados de rendimiento con Website Grader

En el proceso de evaluación del aplicativo web, se han obtenido resultados significativos en los aspectos evaluados para definir su rendimiento y eficiencia. Los resultados obtenidos son:

- **Tamaño de la página:** El tamaño de la página se encuentra en 794KB, lo que se considera eficiente y una carga rápida. Esta optimización mantiene la velocidad de carga del sitio web dentro de límites aceptables, proporcionando una experiencia de usuario ágil.
- **Solicitudes de páginas:** Con solo 24 solicitudes de HTTP, el sistema logra un equilibrio ideal en la cantidad de archivos que se cargan. Esta cifra refleja una gestión eficiente de recursos y favorece una navegación sin demoras innecesarias.
- **Velocidad de la página:** La velocidad de carga promedio de 5.3 segundos se encuentra dentro de los parámetros óptimos. Esto asegura que las páginas web sean interactivas de manera eficiente.
- **Caché web:** Se ha implementado de manera efectiva una caché web de alta calidad. Esto aumenta la velocidad del sitio web al almacenar contenido frecuentemente utilizado en una memoria local, lo que es fundamental para la eficiencia de carga.

- **Redireccionamientos de página mínimos**: El sistema presenta redireccionamientos mínimos, lo que garantiza una carga rápida y evita complicaciones en la navegación de los usuarios.
- **Tamaño de las imágenes:** Se ha logrado una óptima adaptación del tamaño de las imágenes, lo que disminuye los tiempos de carga y mejora la experiencia del usuario, especialmente en dispositivos móviles.
- **JavaScript minimizado:** El código JavaScript se encuentra correctamente comprimido, lo que contribuye significativamente a la velocidad de funcionamiento del sitio web.
- **CSS minimizado:** El CSS también ha sido comprimido de manera eficiente, lo que se traduce en un rendimiento web mejorado y una experiencia de usuario más ágil.

En estos parámetros de evaluación de rendimiento, se obtuvo un puntaje de 27/30, como se detalla en el Apéndice 2. Este puntaje refleja un rendimiento del 90% en el aplicativo web, lo que sugiere que el sistema web está optimizado para la interacción con usuarios y ofrece una experiencia eficiente.

3.2.5. Resultados de pruebas de estrés de microservicios

En el proceso de evaluación de las pruebas de estrés básicas, se obtuvieron resultados que incluyen el tiempo de respuesta, la tasa de error y la utilización de recursos. Los resultados detallados se encuentran en el Apéndice 1.

A continuación, se presentan los resultados clave:

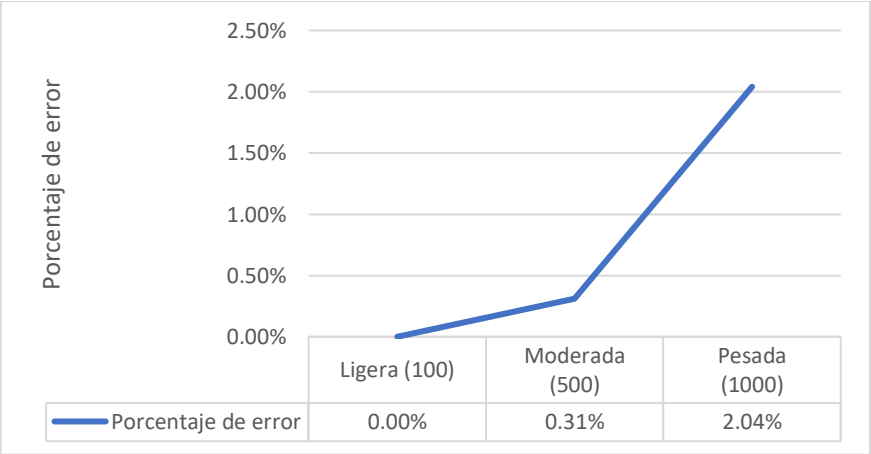


Figura 50: Gráfica de porcentaje de error de pruebas de estrés

En la Figura 50, se determina lo siguiente

- Durante la carga ligera, se observa tiempos de respuesta eficientes en los microservicios. El tiempo promedio de respuesta para operaciones como "GET - Listar propietarios" y

"PUT - Actualizar propietario" se mantuvo por debajo de los 300 ms, lo que indica un rendimiento satisfactorio, con un porcentaje de error del 0.000%. El rendimiento global alcanzó un máximo de 11.446 solicitudes por segundo, lo que refleja la capacidad de nuestros microservicios para manejar una carga inicial de usuarios virtuales.

- En cambio, en la carga moderada se registraron tiempos promedio de respuesta inferiores a 20 segundos. A pesar del aumento en la carga de usuarios virtuales, nuestro sistema mantuvo un rendimiento sólido con 21.706 solicitudes por segundo y un porcentaje de error del 0.31%.
- En la carga pesada, los tiempos de respuesta se extendieron debido a la mayor demanda. A pesar de este aumento en el tiempo de respuesta, el rendimiento global se mantuvo sólido con 21.092 solicitudes por segundo. El porcentaje de error se mantuvo bajo control en un 2.043%, lo que sugiere una capacidad de recuperación adecuada en situaciones de alta demanda.

En todas las ejecuciones de las pruebas de carga, se logró resultados en donde indican que los microservicios pueden mantener un rendimiento estable y una operación confiable incluso bajo condiciones de carga extrema. El bajo porcentaje de error en todas las cargas de prueba demuestra la estabilidad del sistema.

Con base en la información de los resultados y el análisis de los tiempos y errores, se calculó el Tiempo Medio Entre Fallos (MTBF) para cada carga realizada utilizando la siguiente fórmula:

$$MTBF = \frac{\text{Tiempo total de funcionamiento}}{\text{número de fallos}}$$

Carga ligera

Tiempo de Funcionamiento (en segundos) = 60.916 segundos

Número de Fallos = 0 (basado en el porcentaje de error)

$$MTBF = \frac{60{,}916}{0} = \text{Indefinido}$$

Carga moderada

Tiempo de Funcionamiento (en segundos) = 160.834 segundos

Número de Fallos = 11 (basado en el porcentaje de error)

$$MTBF = \frac{160{,}834}{11} = 14{,}621\ {}^{seg}/_{fallo}$$

Carga pesada

Tiempo de Funcionamiento (en segundos) = 300.443 segundos

Número de Fallos = 143 (basado en el porcentaje de error)

$$MTBF = \frac{300.443}{143} = 2{,}099\ {}^{seg}/_{fallo}$$

Los resultados de MTBF indican que los microservicios son confiables y tienen una buena capacidad de recuperación incluso bajo cargas moderadas y pesadas. Si bien se observaron algunos fallos en cargas más altas, estos no son suficientes para comprometer gravemente la estabilidad del sistema.

3.2.6. Resultados de pruebas de portabilidad

Para este tipo de pruebas se verificó la capacidad del software respecto despliegue y ejecución en diferentes entornos y plataformas para asegurar una portabilidad eficiente.

Navegadores Web:

Para evaluar la portabilidad del aplicativo web, se seleccionaron los navegadores más ampliamente utilizados. Los resultados se presentan en la Tabla 15, donde se indica el estado de funcionamiento de la aplicación en cada navegador:

Tabla 15: Navegadores probados para el sistema web

NAVEGADORES WEB	
Navegador	**Estado**
Chrome	Funcional
Edge	Funcional
Brave	Funcional
Firefox	Funcional

En esta tabla, se evidencia que el aplicativo web ha demostrado ser funcional y compatible con los navegadores mencionados. Se evaluó la funcionalidad, velocidad de respuesta de la página y la capacidad de adaptación del diseño a diferentes navegadores, cumpliendo con éxito los requisitos de portabilidad.

Dispositivos móviles

Se sometió el aplicativo móvil a pruebas de portabilidad en dispositivos móviles con diferentes versiones del sistema operativo Android. Los resultados se resumen en la Tabla 16:

Tabla 16: Dispositivos móviles probados para el sistema web

Dispositivos móviles Android	
Infinix (Android 11)	Funcional
Píxel (Android 10)	Funcional
Samsung A2 (Android 6)	Funcional (Diseño no conforme)
Xiaomi A2 Lite	Funcional

En esta tabla, se confirma que el aplicativo móvil es funcional en dispositivos móviles con distintas versiones de Android, incluyendo la versión 6 y superiores. Aunque se señala un diseño algo óptimo en el caso de Samsung A2 con Android 6, por lo que se cumple con el requerimiento no funcional RNF04 relacionado con la portabilidad.

Los resultados de las pruebas de portabilidad indican que el aplicativo móvil es altamente portable, se despliega y ejecuta de manera eficaz en una variedad de dispositivos móviles con diferentes versiones de Android. Esto asegura una experiencia de usuario consistente y satisfactoria, independientemente del dispositivo utilizado.

CONCLUSIONES

- Se logró con éxito la creación de un sistema web y móvil para una clínica veterinaria, utilizando una arquitectura de microservicios para la automatización de procesos de gestión de la clínica, se desarrolló de tal forma que garantiza la disponibilidad y escalabilidad de la información.
- Gracias a la metodología SRL se logró revisar varios artículos científicos sobre los procesos de clínicas veterinarias para una mejor comprensión de las actividades que se lleva a cabo dentro de las clínicas.
- Por la intervención del equipo de desarrollo y el cliente, se logró recolectar los requerimientos tanto funcionales como no funcionales para el sistema de clínica veterinaria.
- El desarrollo del prototipo de la aplicación web y móvil se completó usando la metodología DevOps y las herramientas que fueron seleccionadas para su creación.
- Al evaluar el producto de software según las métricas de funcionalidad, eficiencia, fiabilidad y portabilidad de la norma ISO/IEC 25010, se obtuvieron resultados satisfactorios tanto para la experiencia del usuario como para el administrador. Esto validó la hipótesis y demuestra que el sistema proporciona una solución efectiva para la gestión de procesos en clínicas veterinarias, mejorando la atención a los clientes.

RECOMENDACIONES

- Para la creación de un sistema web y móvil que trabajen juntos es recomendable recolectar requerimientos para que su funcionamiento sea el indicado y necesario del cliente que lo requiere.
- Se recomienda seleccionar una cadena de búsqueda que contenga las palabras claves del proyecto para que los resultados que arroje la base de datos sean más precisos.
- Para la evaluación del sistema se recomienda elegir una norma que sea adecuada para los requerimientos que se establecen para el sistema ya que cada norma posee un conjunto de características y métricas para dicha evaluación.
- Es recomendable implementar herramientas de monitoreo continuo para supervisar el rendimiento y la disponibilidad del sistema en tiempo real. Esto permitirá detectar problemas de manera proactiva y tomar medidas correctivas de inmediato.
- Diseña los microservicios en función de los dominios específicos de la aplicación. Cada microservicio debe tener una responsabilidad claramente definida y limitarse a una función o característica particular.
- Establece una comunicación efectiva entre microservicios utilizando protocolos y estándares de la industria, como RESTful API o gRPC. Asegúrate de que la comunicación sea segura y eficiente.
- Facilita el despliegue independiente de cada microservicio, en contenedores tales como Docker o Kubernetes Esto permite actualizar y escalar componentes individuales sin afectar a otros servicios, lo que mejora la flexibilidad y la disponibilidad.

REFERENCIAS BIBLIOGRÁFICAS

[1] Y. Y. Loor García, «Desarrollo de aplicación web para la gestión de consultas y agendamiento de citas de mascota de la clínica veterinaria burgos.», bachelorThesis, 2019. Accedido: 1 de enero de 2023. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/16991

[2] A. C. Ochoa, A. C. Murillo, y J. Rodas-Silva, «El uso de aplicaciones Web para la Gestión de clínicas veterinarias y su incidencia en la mejora de procesos administrativos», *Ecuadorian Science Journal*, vol. 5, n.º Esp.4, pp. 109-120, oct. 2020.

[3] G. Restrepo, L. Calvachi Galvez, I. Cano Álvarez, y A. Ruíz Marquez, «Las funciones ejecutivas y la lectura: Revisión sistemática de la literatura», *Informes Psicológicos*, vol. 19, n.º 2, pp. 81-94, 2019.

[4] G. de Gasperis, G. D. Penna, y S. D. Facchini, «A microservices architecture for machine learning assisted decision support in a real-time field sensors environment», presentado en CEUR Workshop Proceedings, 2021.

[5] C. Chen, C. Dong, J. Cai, y Q. Ding, «Design and Implementation of Software Architecture for Automotive Green Supply Chain Based on Microservices», presentado en Journal of Physics: Conference Series, 2019. doi: 10.1088/1742-6596/1314/1/012096.

[6] S. Yu, H. Chang, y H. Wang, «Design of Cloud Computing and Microservice-Based Urban Rail Transit Integrated Supervisory Control System Plus», *Urban Rail Transit*, vol. 6, n.º 4, pp. 187-204, dic. 2020, doi: 10.1007/s40864-020-00138-z.

[7] Y. Wang, W. Han, y Z. Nian, «Design of Satellite Ground Management System Based on Microservices», presentado en ACM International Conference Proceeding Series, 2020, pp. 119-123. doi: 10.1145/3403746.3403915.

[8] Y. A. F. Morales, «Healthy Pets, Sistema de Gestión de Información para uso Veterinario.»

[9] H. Liu, Z. Wang, L. Huang, y K. Wang, «Building a Private Cloud Based on Microservices for Computer Science Laboratory in Universities», presentado en Proceedings - 2020 7th International Conference on Information Science and Control Engineering, ICISCE 2020, 2020, pp. 379-384. doi: 10.1109/ICISCE50968.2020.00086.

[10] J. Sadek, D. Craig, y M. Trenell, «Design and Implementation of Medical Searching System Based on Microservices and Serverless Architectures», *Procedia Computer Science*, vol. 196, pp. 615-622, ene. 2022, doi: 10.1016/j.procs.2021.12.056.

[11] F. Aydemir y F. Başçiftçi, «Building a Performance Efficient Core Banking System Based on the Microservices Architecture», *J Grid Computing*, vol. 20, n.º 4, p. 37, nov. 2022, doi: 10.1007/s10723-022-09624-z.

[12] C. Batista, B. Proenca, E. Cavalcante, T. Batista, F. Morais, y H. Medeiros, «Towards a Multi-Tenant Microservice Architecture: An Industrial Experience», presentado en Proceedings - 2022 IEEE 46th Annual Computers, Software, and Applications Conference, COMPSAC 2022, 2022, pp. 553-562. doi: 10.1109/COMPSAC54236.2022.00100.

[13] E. Albertos Gómez, «Arquitecturas software para microservicios: una revisión sistemática de la literatura», masters, E.T.S.I de Sistemas Informáticos (UPM), 2018. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: https://oa.upm.es/51460/

[14] M. R. Huari Casas, «Revisión sistemática sobre generadores de código fuente y patrones de arquitectura.», Pontifica Universidad Católica del Perú, San Miguel, 2020.

[15] L. Matlekovic y P. Schneider-Kamp, «From Monolith to Microservices: Software Architecture for Autonomous UAV Infrastructure Inspection», en *Embedded Systems and Applications*, mar. 2022, pp. 253-272. doi: 10.5121/csit.2022.120622.

[16] J. Yang y X. Zhu, «Design and Implementation of Open Network Teaching System Based on Microservice Architecture», presentado en Proceedings - 2022 International Conference on Informatics, Networking and Computing, ICINC 2022, 2022, pp. 111-115. doi: 10.1109/ICINC58035.2022.00030.

[17] G. Blinowski, A. Ojdowska, y A. Przybylek, «Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation», *IEEE Access*, vol. 10, pp. 20357-20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[18] M. Waseem, P. Liang, y M. Shahin, «A Systematic Mapping Study on Microservices Architecture in DevOps», *Journal of Systems and Software*, vol. 170, p. 110798, dic. 2020, doi: 10.1016/j.jss.2020.110798.

[19] G. Muñoz y L. Fernanda, «Un framework de adaptación de micro-servicios y contenedores para procesos DevOps», oct. 2020, Accedido: 24 de julio de 2023. [En línea]. Disponible en: https://riunet.upv.es/handle/10251/151747

[20] F. Auer, V. Lenarduzzi, M. Felderer, y D. Taibi, «From monolithic systems to Microservices: An assessment framework», *Information and Software Technology*, vol. 137, p. 106600, sep. 2021, doi: 10.1016/j.infsof.2021.106600.

[21] J. W. Chonillo Bermúdez, «Diseño e implementación estructural de un sistema basado en balanceo de cargas para la optimización de tráfico de datos aplicando Failover con protocolo TCP/MSS en el laboratorio de Telecomunicaciones», bachelorThesis, La Libertad: Universidad Estatal Península de Santa Elena, 2022., 2022. Accedido: 24 de julio de 2023. [En línea]. Disponible en: https://repositorio.upse.edu.ec/handle/46000/8688

[22] D. F. Néjer Haro, «Desarrollo de un algoritmo de balanceo de carga dinámico de múltiples rutas para la redistribución de flujos en una red SDN que permita mejorar el rendimiento en el enrutamiento de paquetes TCP/IP», bachelorThesis, 2023. Accedido: 24 de julio de 2023. [En línea]. Disponible en: http://repositorio.utn.edu.ec/handle/123456789/13801

[23] M. J. Blas, H. P. Leone, y S. M. Gonnet, «Modelado y verificación de patrones de diseño de arquitectura de software para entornos de computación en la nube», *Revista Ibérica de Sistemas e Tecnologías de Información*, n.º 35, pp. 1-17, dic. 2019, doi: 10.17013/risti.35.1-17.

[24] J. M. Suárez Pedraza, «Tipificación de dominios de requerimientos para la aplicación de patrones arquitectónicos», 2015, Accedido: 29 de diciembre de 2022. [En línea]. Disponible en: https://repository.unab.edu.co/handle/20.500.12749/3366

[25] S. Estévez Gámez y B. A. Olivares Zepahua, «Desarrollo de un generador de Aplicaciones Enriquecidas de Internet modeladas bajo el patrón arquitectónico MVC usando UML e IFML», Thesis, 2018. Accedido: 29 de diciembre de 2022. [En línea]. Disponible en: http://repositorios.orizaba.tecnm.mx:8080/xmlui/handle/123456789/392

[26] K. Shah, H. Sinha, y P. Mishra, «Analysis of Cross-Platform Mobile App Development Tools», en *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, mar. 2019, pp. 1-7. doi: 10.1109/I2CT45611.2019.9033872.

[27] C. A. Muñoz Muñoz, «APLICACIÓN DE LA METODOLOGÍA MOBILE-D EN EL DESARROLLO DE UNA APP MÓVIL PARA GESTIONAR CITAS MÉDICAS DEL CENTRO JEL RIOBAMBA», bachelorThesis, Riobamba: Universidad Nacional de Chimborazo, 2020. Accedido: 24 de julio de 2023. [En línea]. Disponible en: http://dspace.unach.edu.ec/handle/51000/7073

[28] M. Shakir *et al.*, «Smart Mirror Based Home Automation Using Voice Command and Mobile Application», *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, n.º 35, 2022, doi: 10.4108/eai.12-11-2021.172102.

[29] «Desarrollo backend para aplicaciones web, Servicios Web Restful: Node.js vs Spring Boot - ProQuest». https://www.proquest.com/openview/a78cfaa62708fd24f38ac8d1025050eb/1?pq-origsite=gscholar&cbl=1006393 (accedido 24 de julio de 2023).

[30] P. Plecinski, N. Bokla, T. Klymkovych, M. Melnyk, y W. Zabierowski, «Comparison of Representative Microservices Technologies in Terms of Performance for Use for Projects Based on Sensor Networks», *Sensors (Basel, Switzerland)*, vol. 22, n.º 20, 2022, doi: 10.3390/s22207759.

[31] Y.-T. Wang, S.-P. Ma, Y.-J. Lai, y Y.-C. Liang, «Qualitative and quantitative comparison of Spring Cloud and Kubernetes in migrating from a monolithic to a microservice architecture», *Service Oriented Computing and Applications*, vol. 17, n.º 3, pp. 149-159, 2023, doi: 10.1007/s11761-023-00364-w.

[32] S. Weerasinghe y I. Perera, «Optimized Strategy for Inter-Service Communication in Microservices», *International Journal of Advanced Computer Science and Applications*, vol. 14, n.º 2, pp. 272-279, 2023, doi: 10.14569/IJACSA.2023.0140233.

[33] E. E. Espinoza Freire, «La búsqueda de información científica en las bases de datos académicas», *Revista Metropolitana de Ciencias Aplicadas*, vol. 3, n.º 1, Art. n.º 1, mar. 2020.

[34] L. Marrero *et al.*, «Un estudio comparativo de bases de datos relacionales y bases de datos NoSQL», presentado en XXV Congreso Argentino de Ciencias de la Computación (CACIC) (Universidad Nacional de Río Cuarto, Córdoba, 14 al 18 de octubre de 2019), 2019. Accedido: 29 de diciembre de 2022. [En línea]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/91403

[35] S. I. Logroño-Naranjo, C. R. López-Paredes, N. A. Estrada-Brito, y V. A. Vásconez-Nuñez, «Utilización de lenguajes de programación para generar plataformas informóticas en pro del desarrollo del sector turí-stico que ayude al Crecimiento Económico», *Domino de las Ciencias*, vol. 7, n.º 3, Art. n.º 3, jul. 2021, doi: 10.23857/dc.v7i3.2124.

[36] R. E. Roman Arenaza, «Lenguajes de programación Javascript Java y Javascript. Características. Norma de escritura. Variables y operadores lógicos. Mensajes. Ejercicios. Estructuras condicionales. Funciones y objetos. Aplicaciones», *Universidad Nacional de Educación Enrique Guzmán y Valle*, 2019, Accedido: 21 de febrero de 2023. [En línea]. Disponible en: http://repositorio.une.edu.pe/handle/20.500.14039/3026

[37] F. Gilbert R. y D. Dahl B., «jsr223: A Java Platform Integration for R with Programming Languages Groovy, JavaScript, JRuby, Jython, and Kotlin», *The R Journal*, vol. 10, n.º 2, p. 440, 2019, doi: 10.32614/RJ-2018-066.

[38] S. M. Velásquez, J. D. V. Montoya, M. E. G. Adasme, E. J. R. Zapata, A. A. Pino, y S. L. Marín, «Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software», *Revista CINTEX*, vol. 24, n.º 2, Art. n.º 2, dic. 2019, doi: 10.33131/24222208.334.

[39] J. Morales-Carrillo, L. Cedeño-Valarezo, J. S. C. Bravo, y J. G. O. Calderón, «Metodologías de desarrollo de software y su ámbito de aplicación: Una revisión sistemática».

[40] J. A. Barrera y S. A. Barrera, «Metodologías para el desarrollo de Proyectos», 2020.

[41] E. S. Fernández y D. P. Alfaro, «El modelo iterativo e incremental para el desarrollo de la aplicación de realidad aumentada Amón_RA», *Tecnología en Marcha*, vol. 33, n.º Extra 8, pp. 165-177, 2020.

[42] D. A. V. Paredes, L. C. C. Martínez, R. M. L. Bermúdez, y S. R. P. Mendoza, «Análisis de la metodología RUP en el desarrollo de software académico mediante la herramienta DJANGO», *RECIMUNDO*, vol. 3, n.º 2, Art. n.º 2, abr. 2019, doi: 10.26820/recimundo/3.(2).abril.2019.964-979.

[43] C. R. Vargas, «Metodología de desarrollo agil en programación extrema», *MoleQla: revista de Ciencias de la Universidad Pablo de Olavide*, n.º 18, pp. 3-3, 2015.

[44] M. T. Vivanco, Y. F. Rigondeaux, y Y. C. González, «Aplicación de la metodología scrum-programación extrema al sistema de control de consumo de combustibles», *Didasc@lia: Didáctica y Educación*, vol. 12, n.º 4, pp. 156-168, 2021.

[45] M. V. Estrada-Velasco, J. A. Núñez-Villacis, P. R. Saltos-Chávez, y W. C. Cunuhay-Cuchipe, «Revisión Sistemática de la Metodología Scrum para el Desarrollo de Software», *Domino de las Ciencias*, vol. 7, n.º 4, Art. n.º 4, dic. 2021, doi: 10.23857/dc.v7i4.2429.

[46] A. Escudero-Nahón y M. Ibarra Corona, «Metodologías de Ingeniería de Software para el Diseño y Desarrollo de Plataformas de Tecnología Educativa», jun. 2021.

[47] K. I. Flores Aguirre, «Diseño de un sistema guía de rutas basado en la metodología mobile-d para la movilidad de personas con discapacidad visual en el centro de la ciudad de Ibarra», bachelorThesis, 2021. Accedido: 28 de diciembre de 2022. [En línea]. Disponible en: http://repositorio.utn.edu.ec/handle/123456789/11639

[48] A. S. R. Molina, M. G. Aguirre, R. F. H. Alarcon, E. S. Carmona, y V. A. Hilario, «¿Qué es DevOps? Definición y características», *Revista Innova Ingeniería*, vol. 1, n.º 7, Art. n.º 7, 2022.

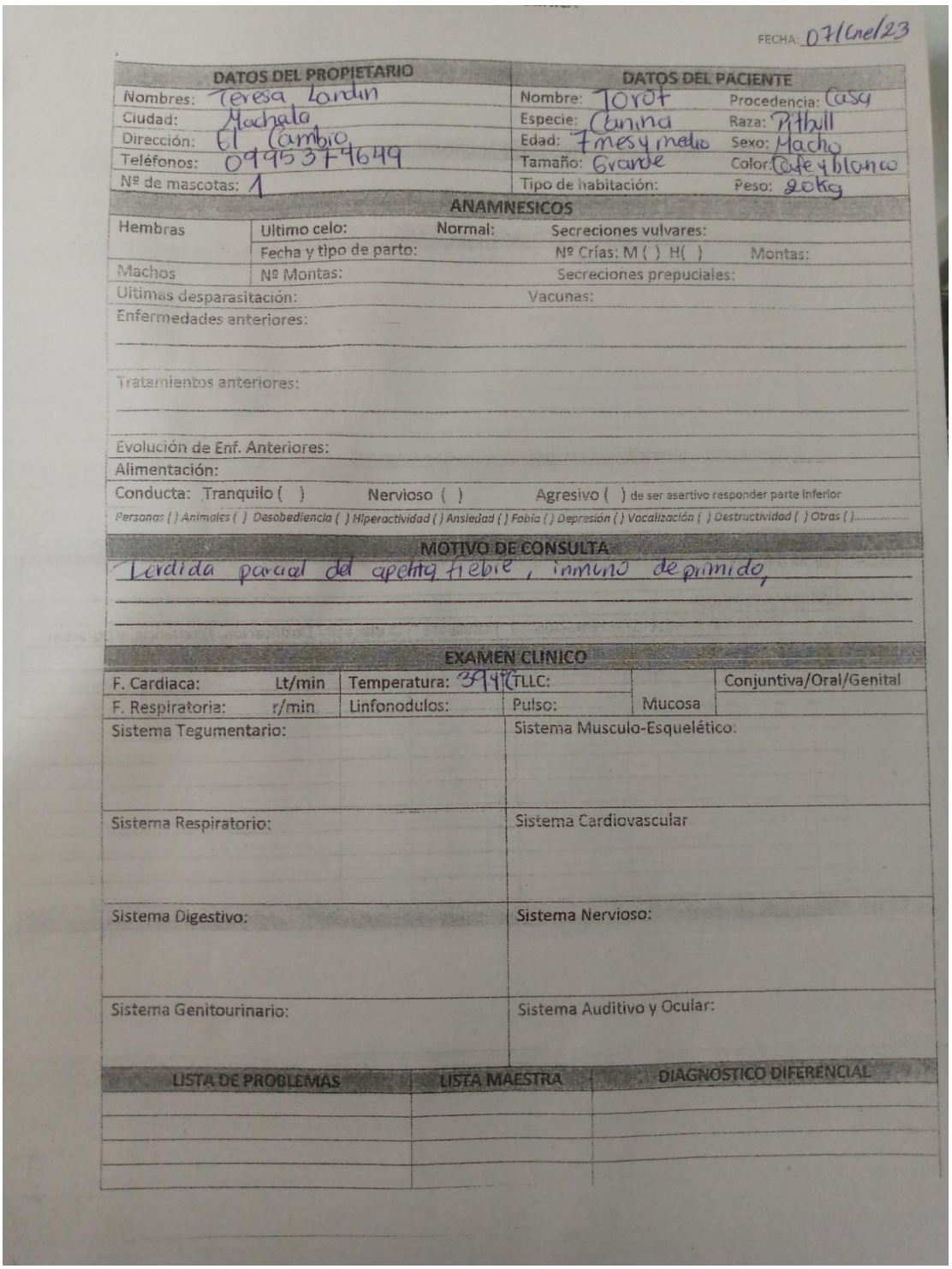
[49] A. M. F. Redondo y F. de J. N. Cárdenas, «DevOps: un vistazo rápido», *Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla*, vol. 10, n.º 19, Art. n.º 19, ene. 2022, doi: 10.29057/esh.v10i19.8121.

[50] Z. M. Rodríguez, L. D. P. Rodríguez, y J. C. G. Suarez, «Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua», *Industrial Data*, vol. 23, n.º 2, 2020, Accedido: 2 de marzo de 2023. [En línea]. Disponible en: https://www.redalyc.org/journal/816/81665362014/html/

[51] B. Golden, «3 types of tests for microservices and distributed application systems», *TechBeacon*. https://techbeacon.com/app-dev-testing/3-non-negotiable-tests-microservices-distributed-app-environments (accedido 3 de marzo de 2023).

ANEXOS

Anexo 1 – Historia clínica de Clínica Veterinaria HappyPets



FECHA: 07/Dnel/23

DATOS DEL PROPIETARIO		DATOS DEL PACIENTE			
Nombres:	Teresa Lardin	Nombre:	Torot	Procedencia:	Casa
Ciudad:	Machala	Especie:	Canina	Raza:	Pitbull
Dirección:	El Cambio	Edad:	7mes y medio	Sexo:	Macho
Teléfonos:	0995374649	Tamaño:	Grande	Color:	Cafe y blanco
Nº de mascotas:	1	Tipo de habitación:		Peso:	20kg

ANAMNESICOS

Hembras	Ultimo celo:	Normal:	Secreciones vulvares:
	Fecha y tipo de parto:		Nº Crías: M ( ) H( ) Montas:
Machos	Nº Montas:		Secreciones prepuciales:
Ultimas desparasitación:			Vacunas:
Enfermedades anteriores:			
Tratamientos anteriores:			
Evolución de Enf. Anteriores:			
Alimentación:			
Conducta: Tranquilo ( )	Nervioso ( )	Agresivo ( ) de ser asertivo responder parte inferior	
Personas ( ) Animales ( ) Desobediencia ( ) Hiperactividad ( ) Ansiedad ( ) Fobia ( ) Depresión ( ) Vocalización ( ) Destructividad ( ) Otras ( )			

MOTIVO DE CONSULTA

Perdida parcial del apetito fiebre, inmuno deprimido,

EXAMEN CLINICO

F. Cardiaca:	Lt/min	Temperatura: 39.9°C	TLLC:	Mucosa	Conjuntiva/Oral/Genital
F. Respiratoria:	r/min	Linfonodulos:	Pulso:		

Sistema Tegumentario:	Sistema Musculo-Esquelético:
Sistema Respiratorio:	Sistema Cardiovascular
Sistema Digestivo:	Sistema Nervioso:
Sistema Genitourinario:	Sistema Auditivo y Ocular:

LISTA DE PROBLEMAS	LISTA MAESTRA	DIAGNOSTICO DIFERENCIAL

Figura 51: Historia clínica de Clínica Veterinaria HappyPets

Anexo 2 – Historia de usuarios (detalle)

Tabla 17: Historia de usuario 1

Historia de usuario	
Número: 1	
Nombre de historia: Inicio de Sesión	
Prioridad en negocio: Alta	**Usuario:** Clientes – Empleados – Doctor
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El sistema permitirá al usuario iniciar sesión ingresando sus credenciales. Si el usuario no existe, el sistema mostrará una alerta indicando que no se encuentra registrado. Asimismo, si las credenciales ingresadas no coinciden con la información almacenada en la base de datos, se notificará al usuario mediante una alerta correspondiente.	

Tabla 18: Historia de usuario 2

Historia de usuario	
Número: 2	
Nombre de historia: Gestión de usuarios y roles (Gateway)	
Prioridad en negocio: Alta	**Usuario:** Administrador
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El Administrador tendrá los permisos en el control de acceso para asignar roles a los usuarios que van ingresando al sistema.	

Tabla 19: Historia de usuario 3

Historia de usuario	
Número: 3	
Nombre de historia: Gestión de propietarios-mascotas	
Prioridad en negocio: Alta	**Usuario:** Doctor - Empleado
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El usuario será capaz de realizar todas las operaciones acerca de la información de los propietarios y mascotas, pueden ver el listado de propietarios y mascotas que están registrados.	

Tabla 20: Historia de usuario 4

Historia de usuario	
Número: 4	
Nombre de historia: Gestión de historias clínicas	
Prioridad en negocio: Alta	**Usuario:** Doctor - Empleado
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El usuario será capaz de visualizar todas las historias clínicas que están registradas en el sistema, podrá crear, actualizar y eliminar las historias clínicas.	

Tabla 21: Historia de usuario 5

Historia de usuario	
Número: 5	
Nombre de historia: Gestión de citas	
Prioridad en negocio: Alta	**Usuario:** Cliente – Doctor – Empleado
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El usuario tendrá la capacidad de ver una lista de citas registradas en el sistema, así como filtrarlas por citas pendientes, atendidas o todas. Además, podrán agregar o eliminar citas según sea necesario.	

Tabla 22: Historia de usuario 6

Historia de usuario	
Número: 6	
Nombre de historia: Gestión de inventario y facturación	
Prioridad en negocio: Alta	**Usuario:** Doctor - Empleado
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El usuario podrá controlar el inventario y generar facturas, , permitiendo así mantener un registro claro y actualizado de las ventas y productos disponibles.	

Tabla 23: Historia de usuario 7

Historia de usuario	
Número: 7	
Nombre de historia: Gestión de comunicación	
Prioridad en negocio: Alta	**Usuario:** Doctor - Empleado
Programador responsable: Marlon Villamar Cruz, Luis Loayza Agila	
Descripción: El usuario podrá enviar y recibir mensajes, crear grupos de conversación, y tener una opción de búsqueda para encontrar conversaciones anteriores y mensajes específicos de forma rápida y sencilla.	

Anexo 3 – Herramientas de Recolección de Datos (Preguntas de entrevista)

Entrevista para trabajo de Proyecto de Integración Curricular

Entrevistador: Marlon Geovanny Villamar Cruz

Entrevistado: _____

1. ¿Qué actividades realizan en la clínica veterinarias

---

2. ¿Como denominaría los procesos en la gestión de clínica veterinaria?

---

3. ¿Cómo lleva el control de la información de las mascotas?

---

4. ¿Cómo lleva el control administrativo de la clínica veterinaria o veterinaria?

---

5. ¿Controla el agendamiento de citas para consultas de revisión médica de la mascota o requerir algún proceso estético?

---

6. Si tuviera un sistema para la gestión de la clínica veterinaria que le gustaría incluir

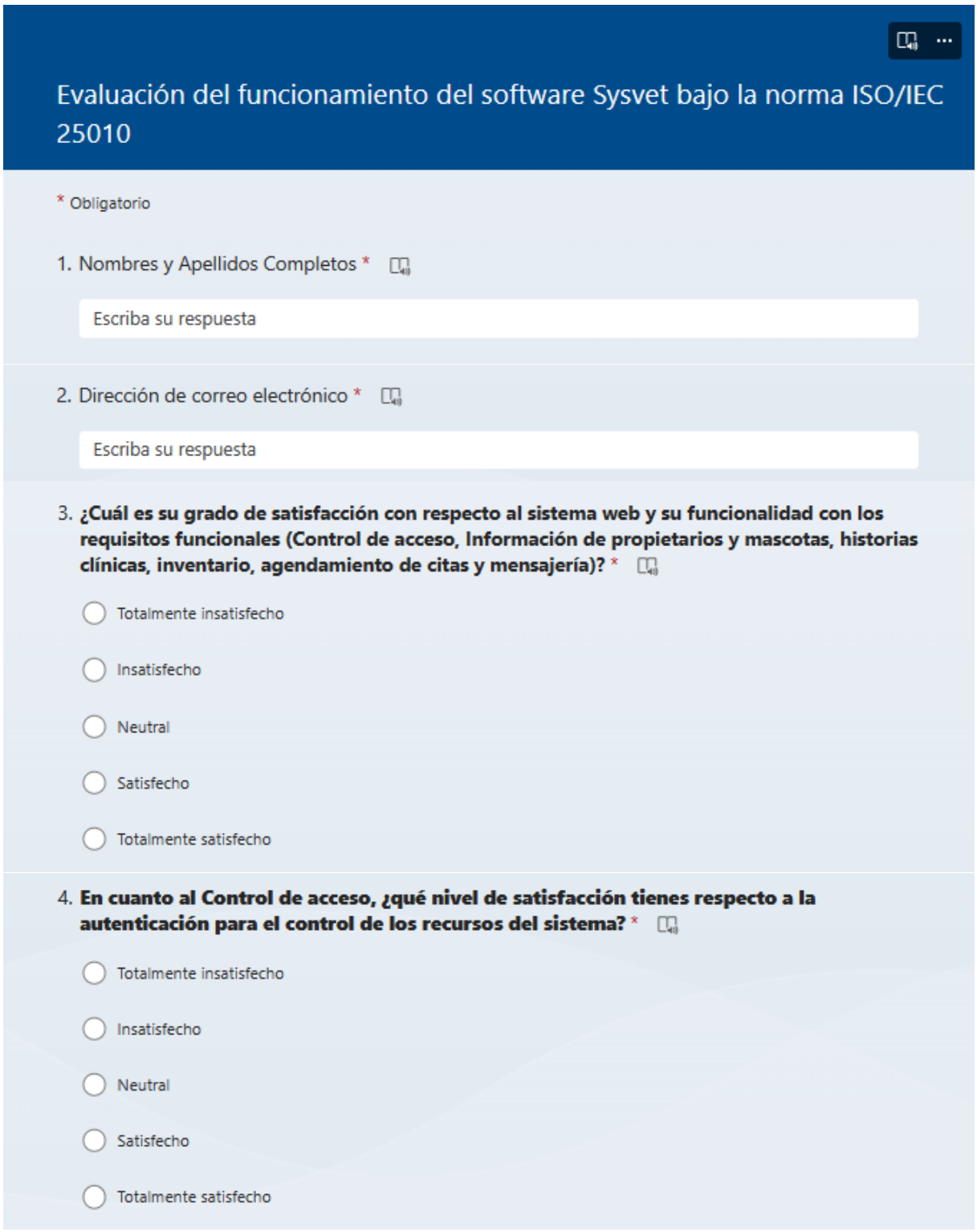
---

Anexo 4 – Herramientas de Recolección de Datos (Guía de Observación)

CONTROL DE PROCESOS Y ACTIVIDADES	
Nombre del establecimiento:	
PROCESO:	

ACTIVIDADES		
#	**Actividad**	**Descripción**

Anexo 5 – Herramientas de recolección de datos (Evaluación del funcionamiento del software Sysvet bajo la norma ISO/IEC 25010)



Evaluación del funcionamiento del software Sysvet bajo la norma ISO/IEC 25010

* Obligatorio

1. Nombres y Apellidos Completos *

Escriba su respuesta

2. Dirección de correo electrónico *

Escriba su respuesta

3. **¿Cuál es su grado de satisfacción con respecto al sistema web y su funcionalidad con los requisitos funcionales (Control de acceso, Información de propietarios y mascotas, historias clínicas, inventario, agendamiento de citas y mensajería)?** *

- Totalmente insatisfecho
- Insatisfecho
- Neutral
- Satisfecho
- Totalmente satisfecho

4. **En cuanto al Control de acceso, ¿qué nivel de satisfacción tienes respecto a la autenticación para el control de los recursos del sistema?** *

- Totalmente insatisfecho
- Insatisfecho
- Neutral
- Satisfecho
- Totalmente satisfecho

5. **Indique el grado de satisfacción respecto al cumplimiento de la Gestión de propietarios y mascotas.** *

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

6. **¿Encuentras que la información de las mascotas y sus propietarios está organizada de una manera clara y fácil de visualizar?** *

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

7. **En relación a la Gestión de historias clínicas, ¿qué nivel de satisfacción tienes respecto al software para permitirte visualizar las historias clínicas (consultas, vacunaciones y desparasitaciones) de las mascotas?** *

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

8. **En lo que respecta a la Gestión de citas, ¿qué nivel de satisfacción tienes para realizar el agendamiento de citas y visualizarlas?** *

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

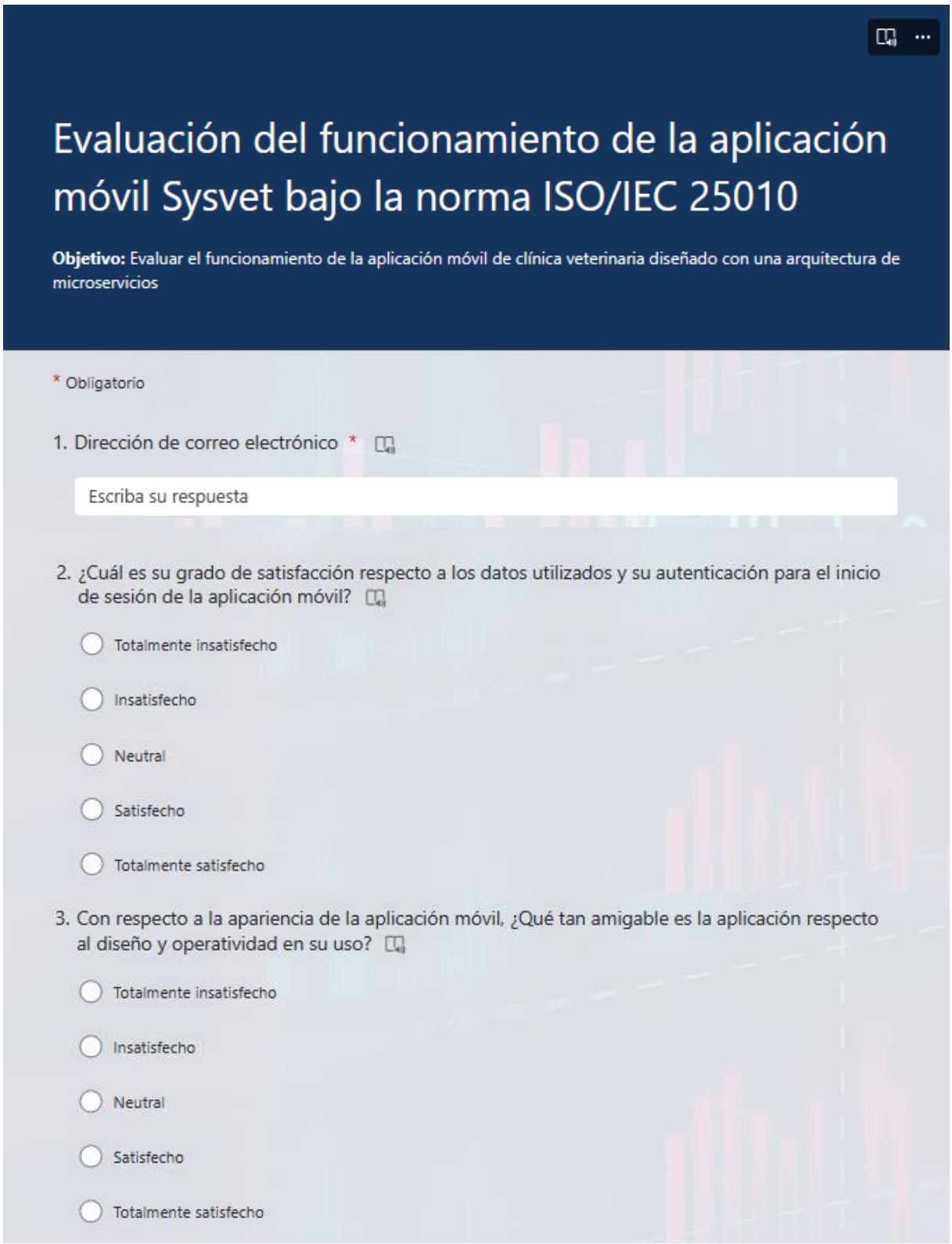
9. **¿Cuál es el nivel de satisfacción que tiene el software para registrar presentaciones, categorías, proveedores y productos en la gestión de inventario?**

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

10. **En la gestión de mensajería, ¿qué nivel de satisfacción tienes con la acción del software de enviar notificaciones al correo de aviso de cita un día antes de la programada?** *

- ○ Totalmente insatisfecho
- ○ Insatisfecho
- ○ Neutral
- ○ Satisfecho
- ○ Totalmente satisfecho

Anexo 6 – Herramientas de recolección de datos (Evaluación del funcionamiento de la aplicación móvil Sysvet bajo la norma ISO/IEC 25010)



Evaluación del funcionamiento de la aplicación móvil Sysvet bajo la norma ISO/IEC 25010

Objetivo: Evaluar el funcionamiento de la aplicación móvil de clínica veterinaria diseñado con una arquitectura de microservicios

* Obligatorio

1. Dirección de correo electrónico *

Escriba su respuesta

2. ¿Cuál es su grado de satisfacción respecto a los datos utilizados y su autenticación para el inicio de sesión de la aplicación móvil?

- Totalmente insatisfecho
- Insatisfecho
- Neutral
- Satisfecho
- Totalmente satisfecho

3. Con respecto a la apariencia de la aplicación móvil, ¿Qué tan amigable es la aplicación respecto al diseño y operatividad en su uso?

- Totalmente insatisfecho
- Insatisfecho
- Neutral
- Satisfecho
- Totalmente satisfecho

4. Indique su grado de satisfacción respecto al registro de un nuevo usuario desde la aplicación móvil

- ◯ Totalmente insatisfecho
- ◯ Insatisfecho
- ◯ Neutral
- ◯ Satisfecho
- ◯ Totalmente satisfecho

5. La aplicación móvil permite editar y visualizar la información (vacunaciones y desparasitaciones) de la mascota, ¿Está usted conforme con esta opción?

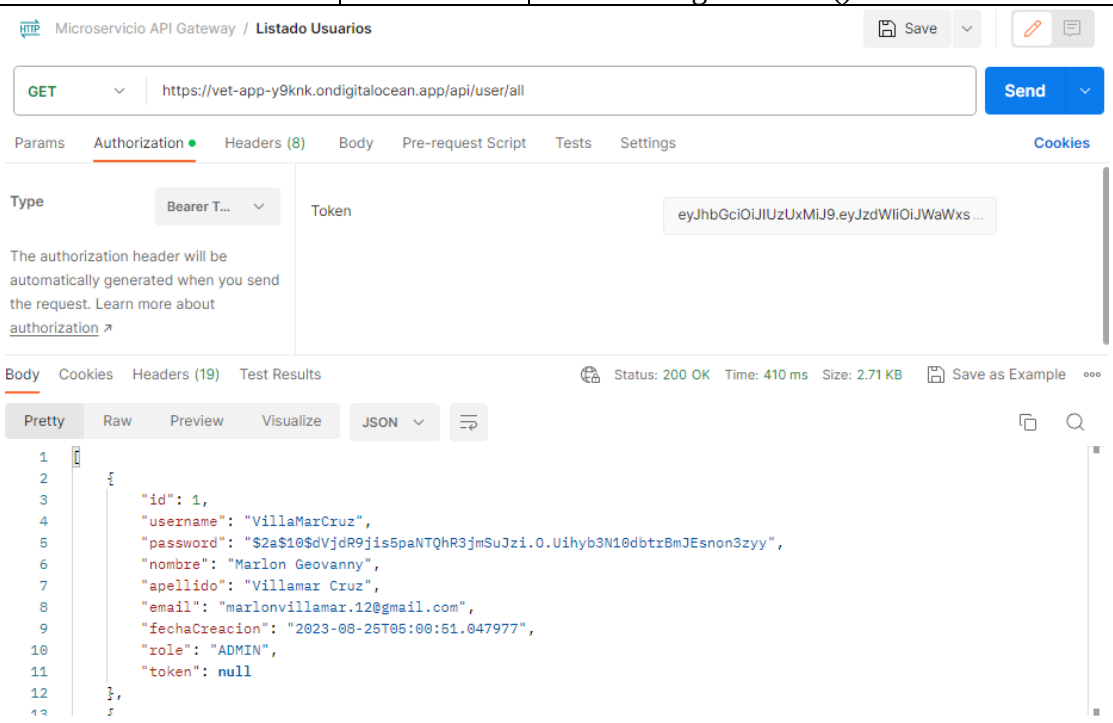
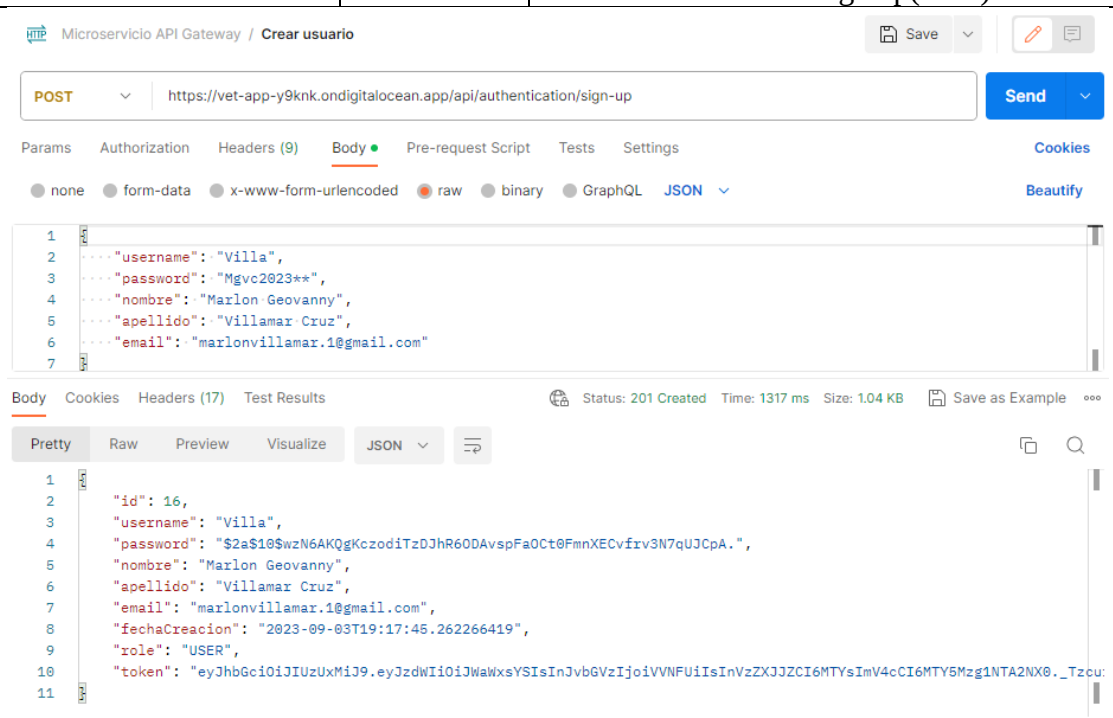
- ◯ Totalmente insatisfecho
- ◯ Insatisfecho
- ◯ Neutral
- ◯ Satisfecho
- ◯ Totalmente satisfecho

6. Respecto a la opción de agendar citas medicas para su mascota desde la aplicación móvil, ¿Está usted de acuerdo que se permita esta opción?

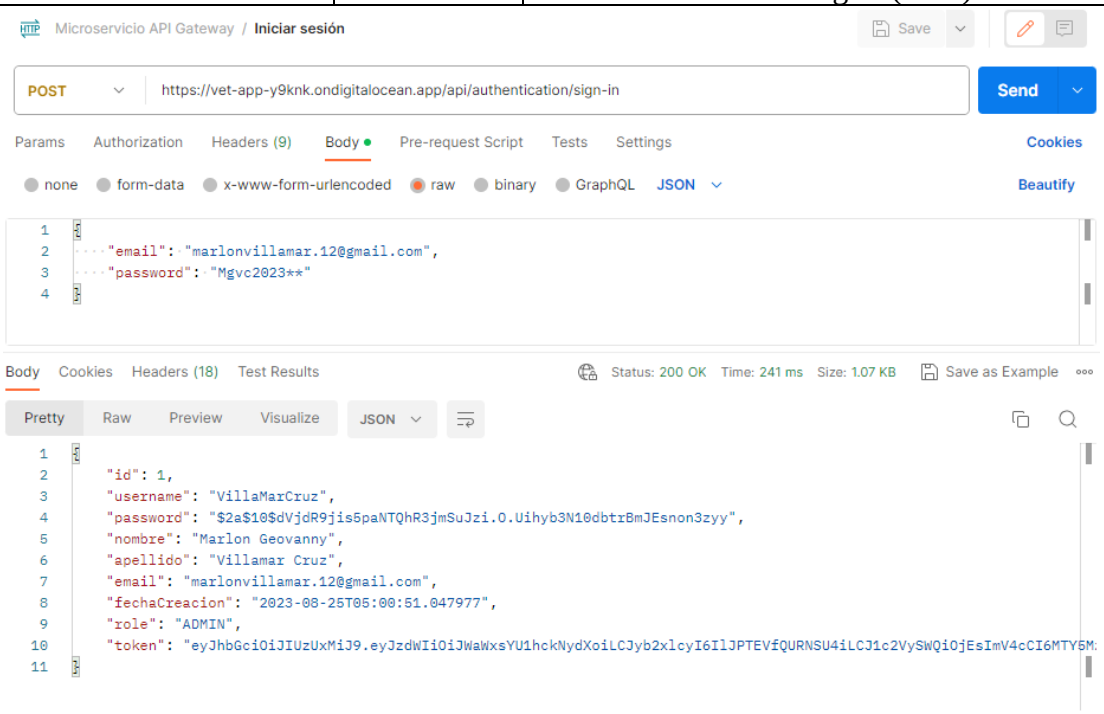
- ◯ Totalmente insatisfecho
- ◯ Insatisfecho
- ◯ Neutral
- ◯ Satisfecho
- ◯ Totalmente satisfecho

APÉNDICES

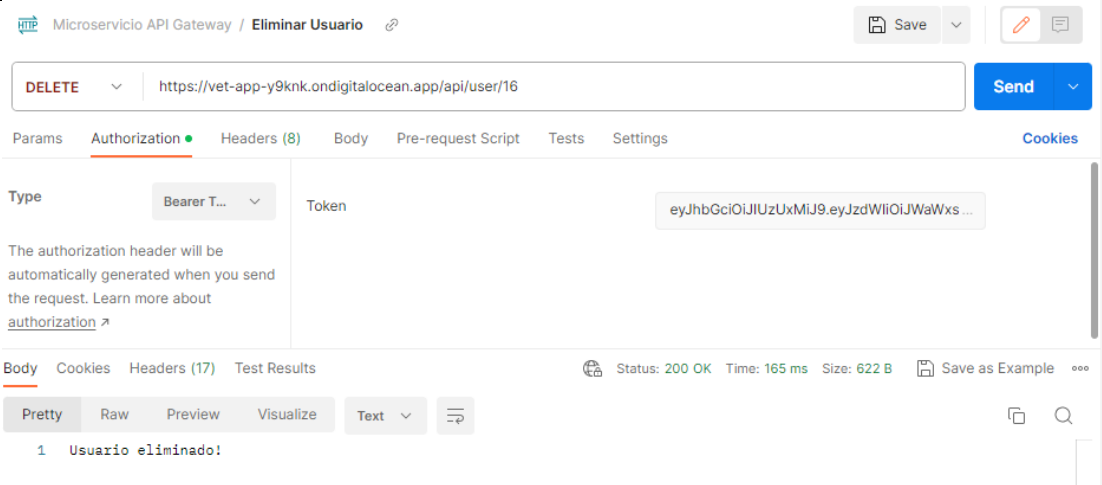
Apéndice 1 – Ejecución de pruebas de funcionamiento

<table>
<tr><th colspan="3">Microservicio Api Gateway</th></tr>
<tr><td rowspan="2">/api/user/all</td><td>Método</td><td>GET</td></tr>
<tr><td>Manejador</td><td>xyz.vet.microservice0apigateway.controller.UserController#getUserAll()</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td rowspan="2">/api/authentication/sign-up</td><td>Método</td><td>POST</td></tr>
<tr><td>Manejador</td><td>xyz.vet.microservice0apigateway.controller.AuthenticationController#signUp(User)</td></tr>
<tr><td colspan="3"> 
</td></tr>
</table>

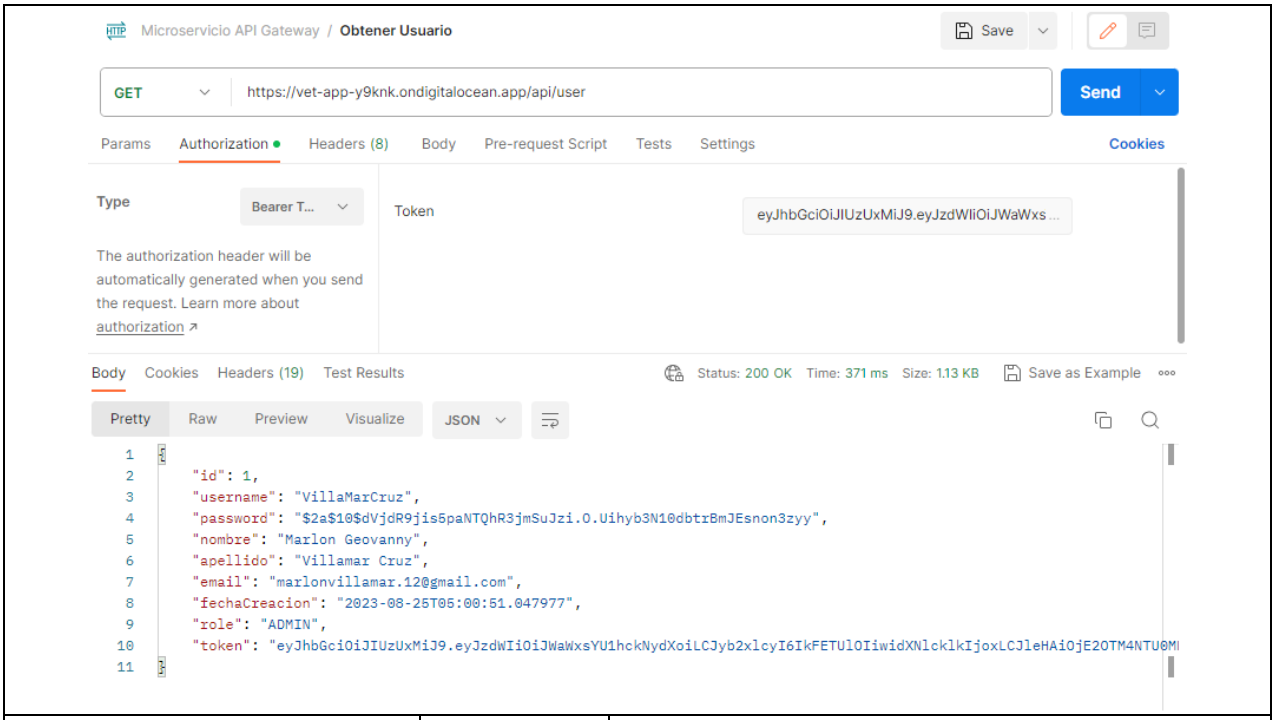
/api/authentication/sign-in	**Método**	POST
	Manejador	xyz.vet.microservice0apigateway.controller.AuthenticationController#signIn(User)



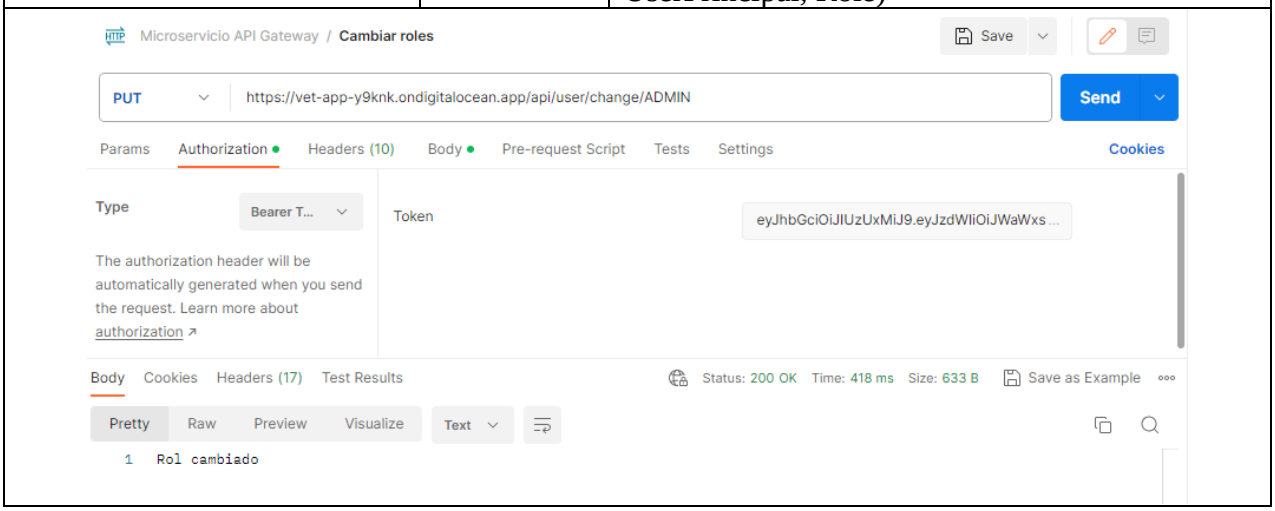
/api/user/{id}	**Método**	DELETE
	Manejador	xyz.vet.microservice0apigateway.controller.UserController#deleteUser(Long)



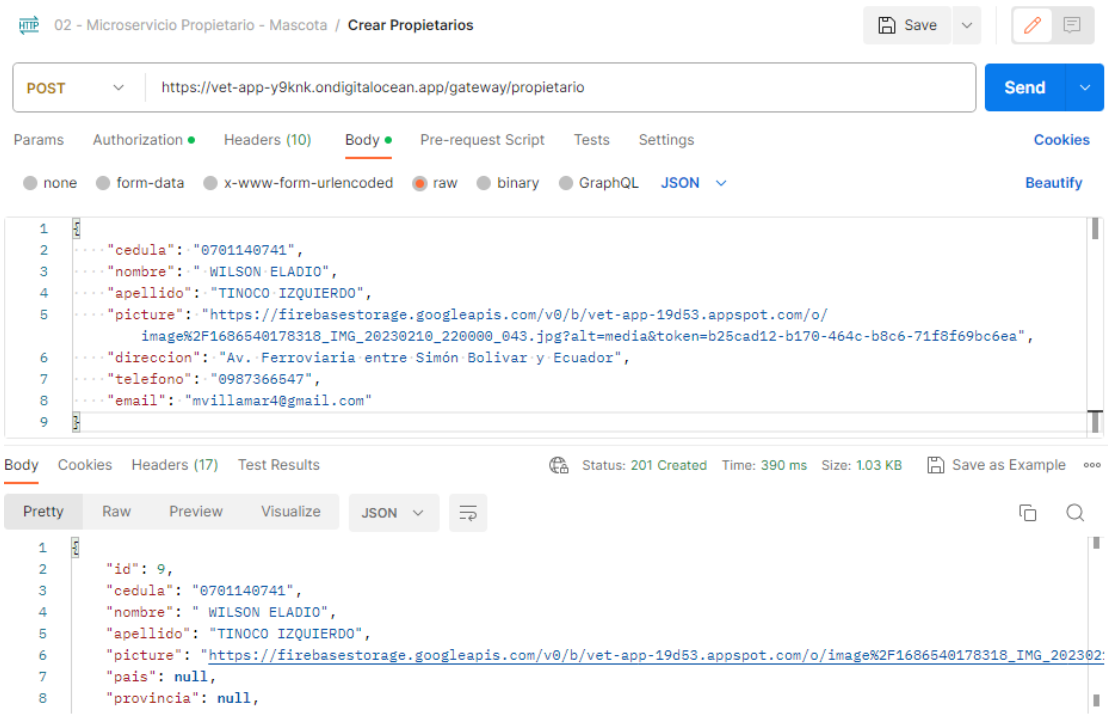
/api/user	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.UserController#getCurrentUser(UserPrincipal)



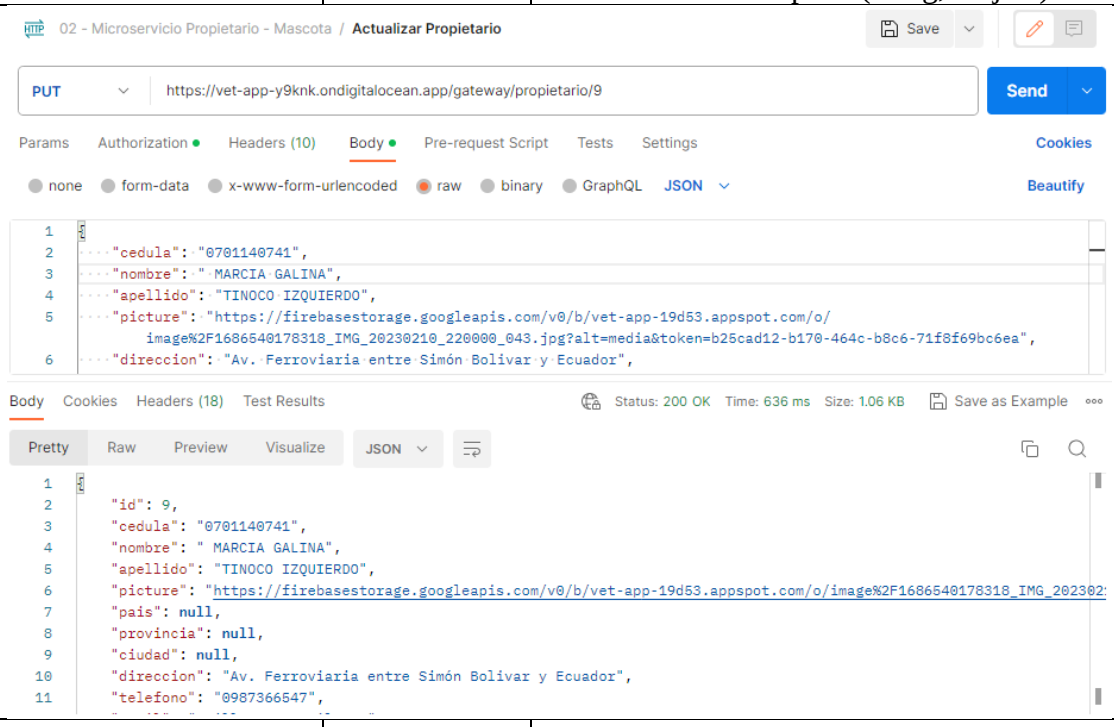
/api/user/change/{role}	**Método**	PUT
	Manejador	xyz.vet.microservice0apigateway.controller.UserController#changeRole(UserPrincipal, UserPrincipal, Role)



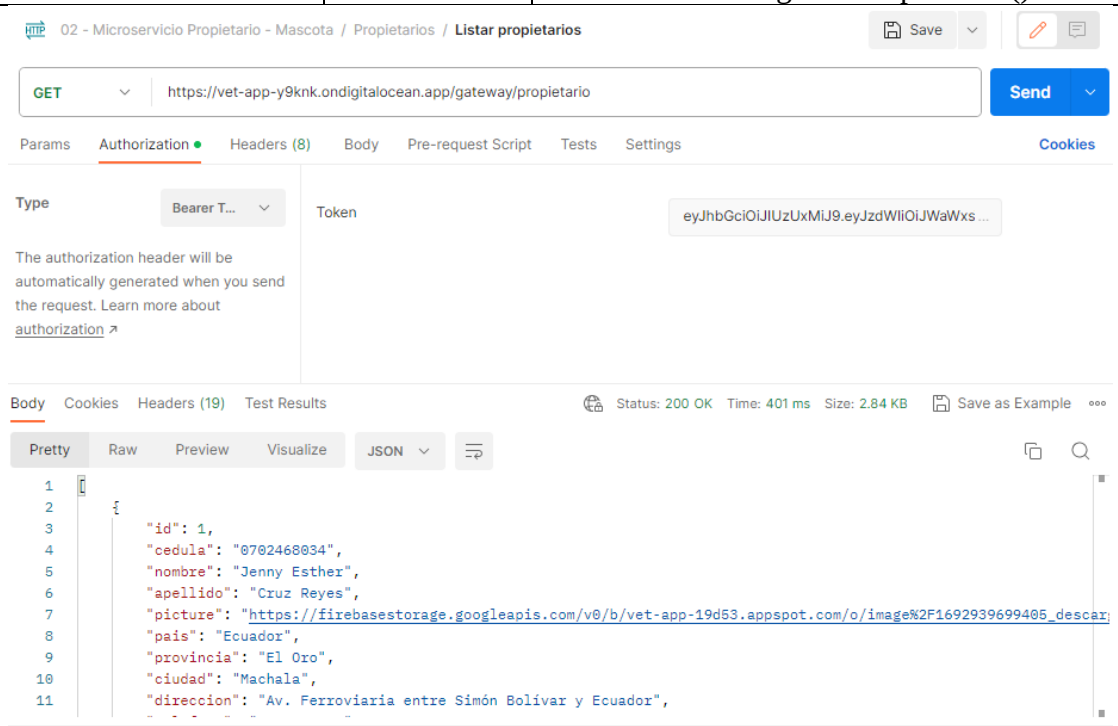
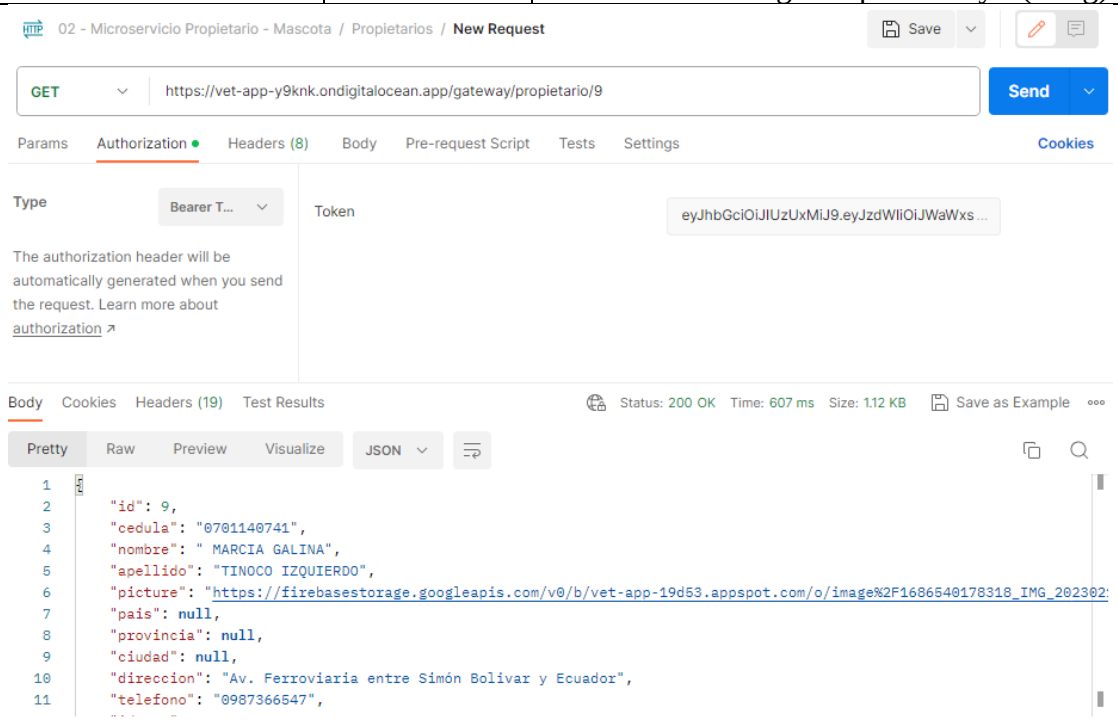
Microservicio Propietarios-Mascotas		
/gateway/propietario	**Método**	POST
	Manejador	xyz.vet.microservice0apigateway.controller.PropietarioController#create(PropietarioDto, BindingResult)

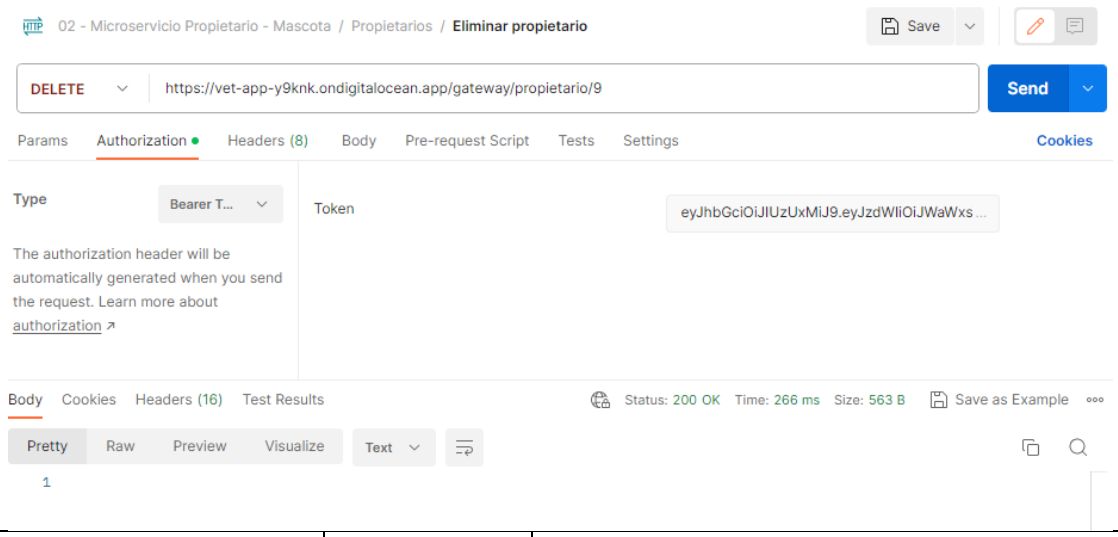


/gateway/propietario/{propietarioId}	**Método**	PUT
	Manejador	xyz.vet.microservice0apigateway.controller.PropietarioController#update(Long, Object)

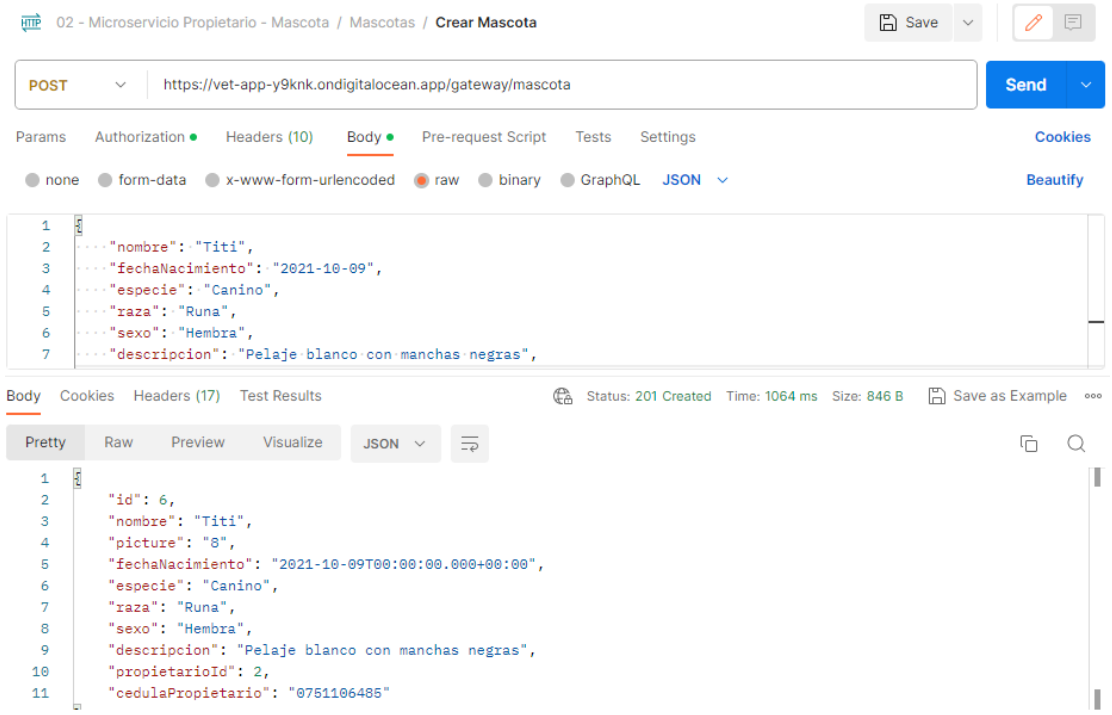


/gateway/propietario	**Método**	GET

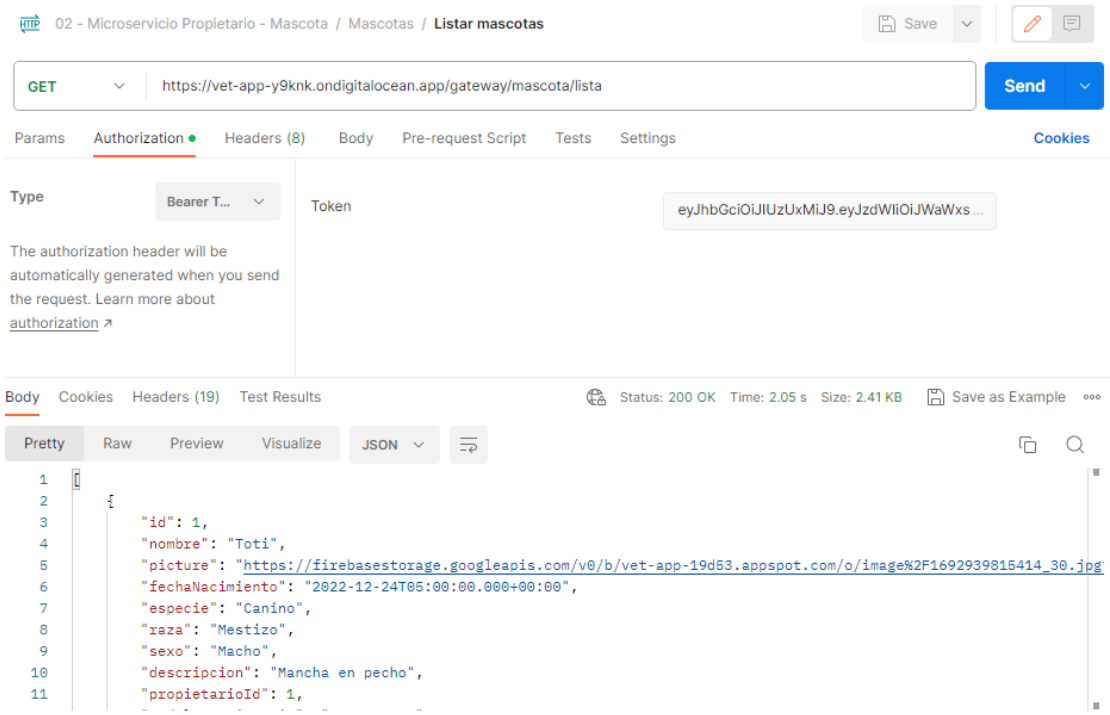
	Manejador	xyz.vet.microservice0apigateway.controller.PropietarioController#getAllPropietarios()
		
/gateway/propietario/{propietarioId}	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.PropietarioController#getPropietarioById(Long)
		
/gateway/propietario/{propietarioId}	**Método**	DELETE
	Manejador	xyz.vet.microservice0apigateway.controller.PropietarioController#delete(Long)



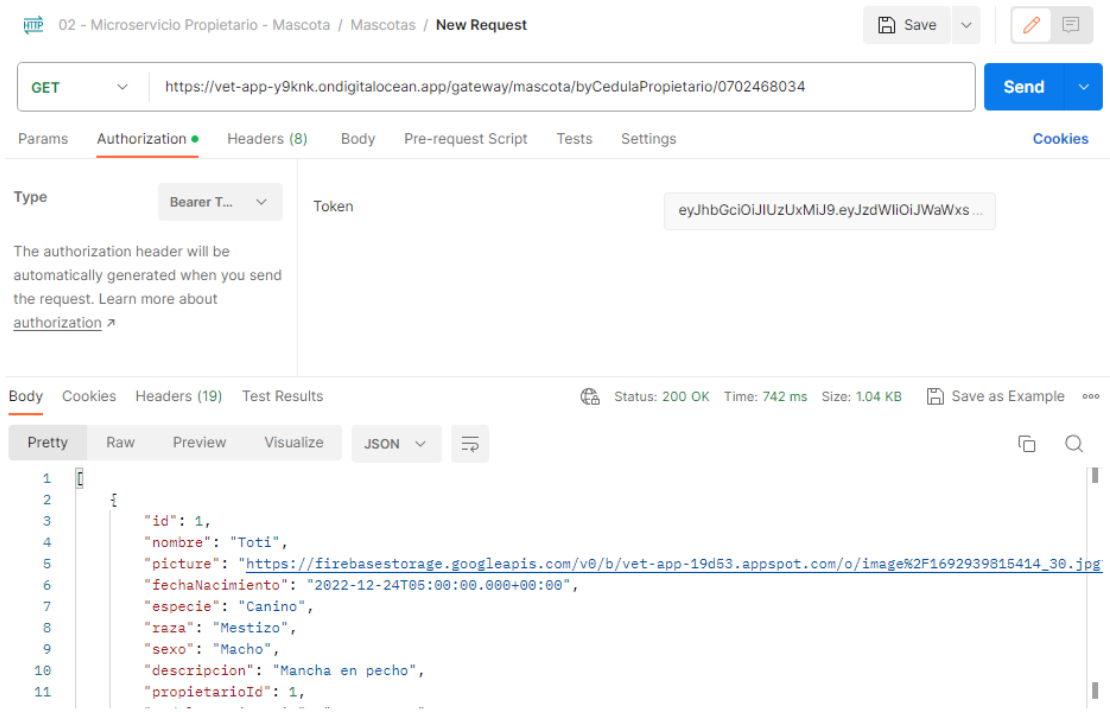
/gateway/mascota	**Método**	POST
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#create(Object)



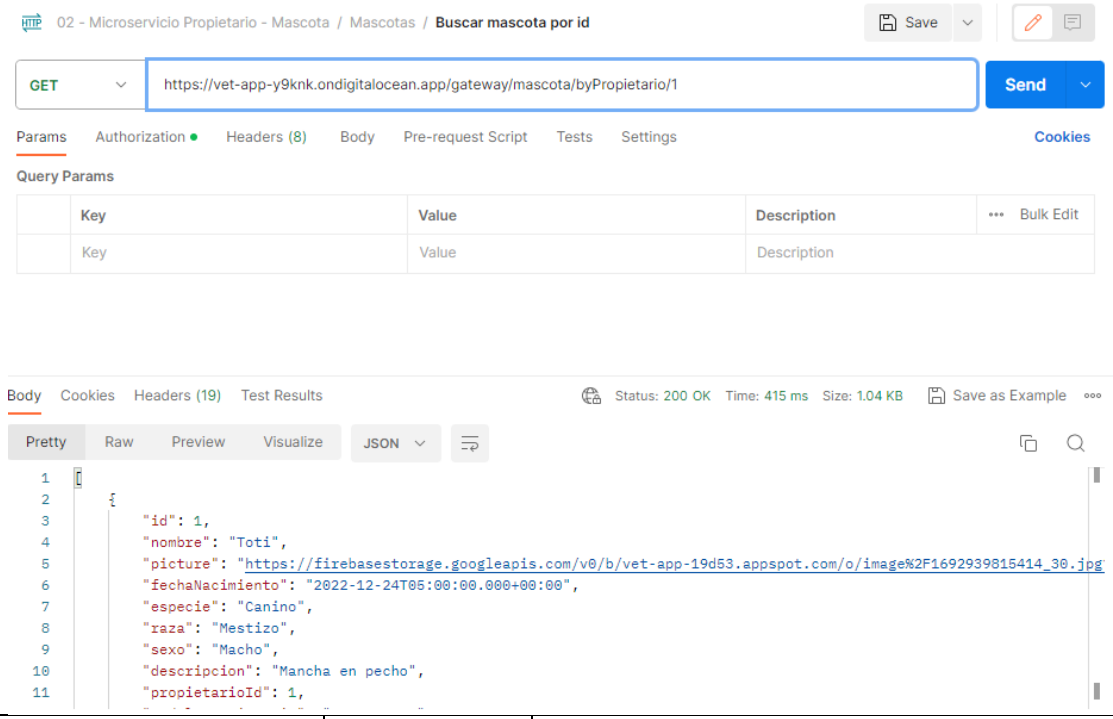
/gateway/mascota/lista	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#getAllMascotas()



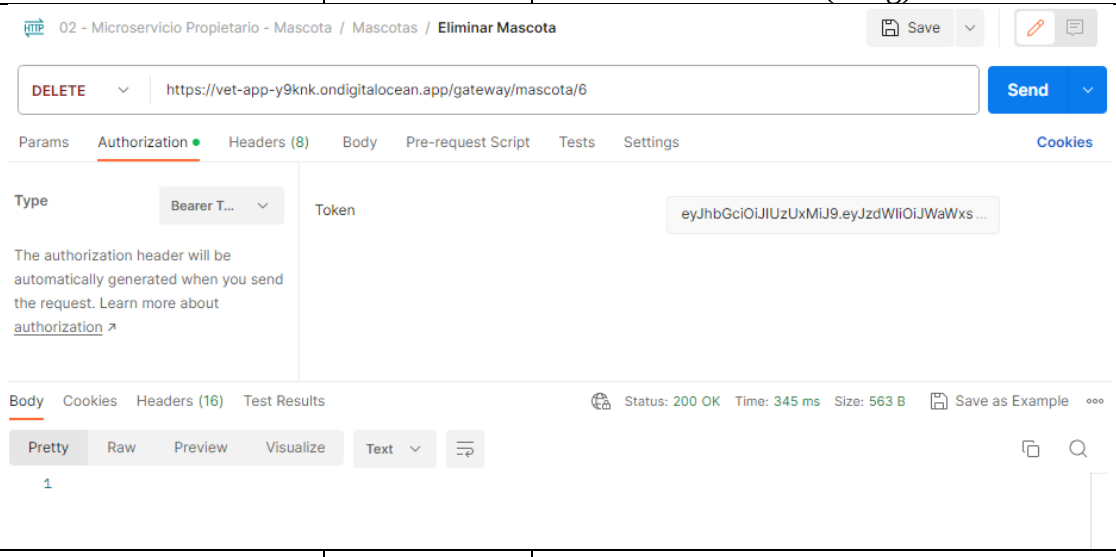
/gateway/mascota/byCedulaPropietario/{cedula}	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#getAllMascotasOfCedulaPropietario(String)



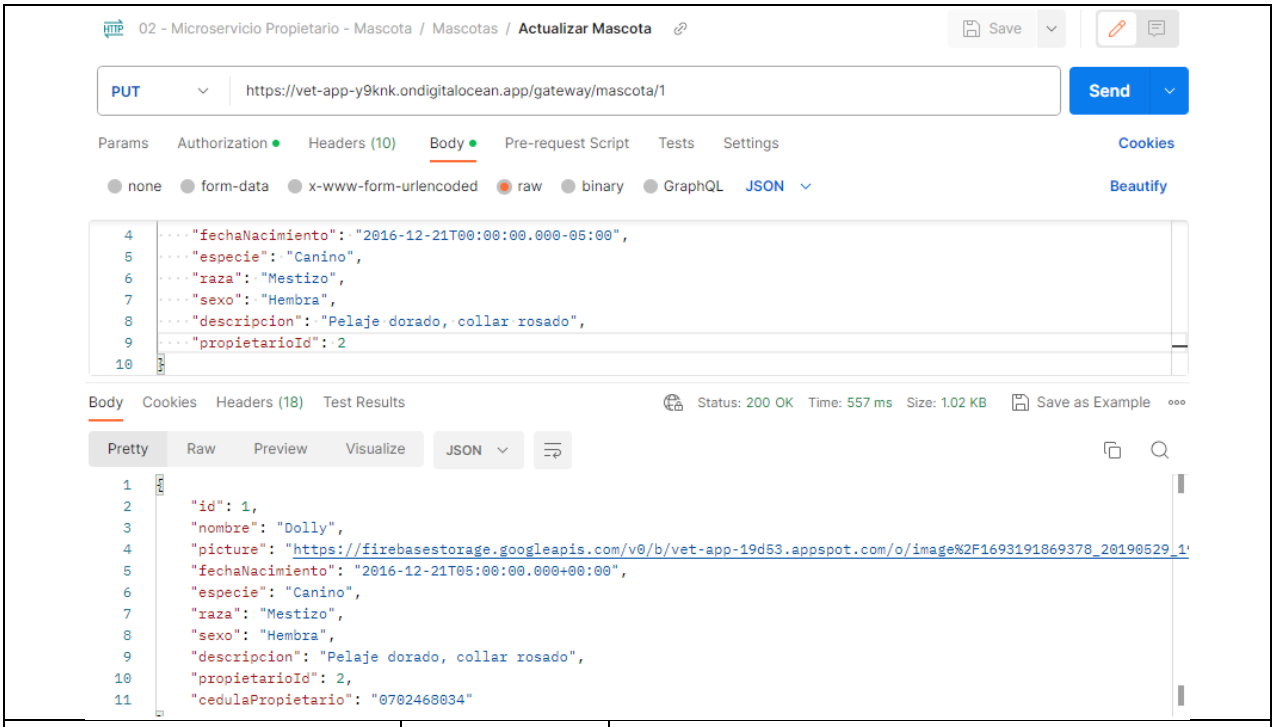
/gateway/mascota/byPropietario/{idPropietario}	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#getAllMascotasOfPropietario(Long)



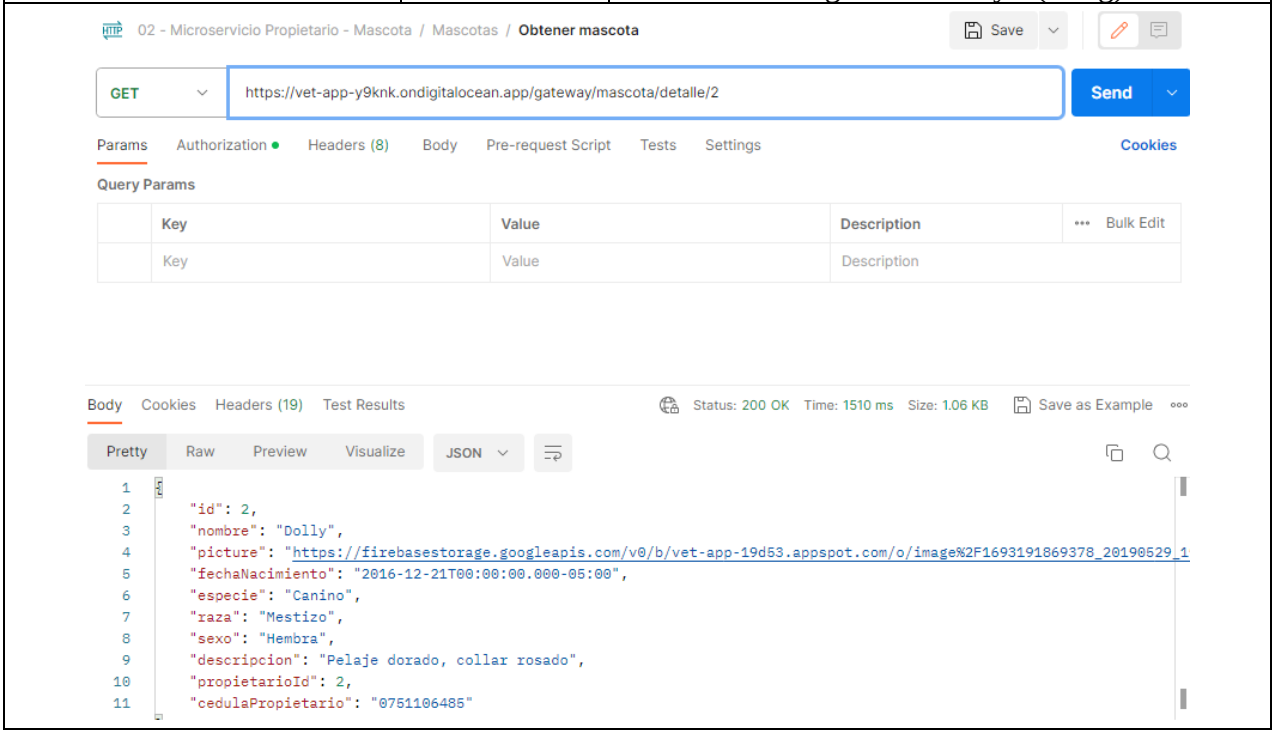
/gateway/mascota/{mascotaId}	**Método**	DELETE
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#delete(Long)



/gateway/mascota/{mascotaId}	**Método**	PUT
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#update(Long, Object)



/gateway/mascota/detalle/{idMascota}	**Método**	GET
	Manejador	xyz.vet.microservice0apigateway.controller.MascotaController#getMascotaById(Long)



Apéndice 2 – Resultados de pruebas de cargas de microservicios

Reporte resumen

Nombre: Reporte resumen 100 Hilos

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Grupo de Hilos 100:DELETE - Eliminar mascota	100	260	0	866	70,70	0,00%	1,7/sec	0,92	0,81	563,0
Grupo de Hilos 100:GET - Listar mascotas	100	260	0	429	19,76	0,00%	1,7/sec	1,99	0,81	1215,1
Grupo de Hilos 100:GET - Listar propietarios	100	265	0	787	59,37	0,00%	1,7/sec	2,48	0,72	1516,0
Grupo de Hilos 100:Login-74	100	280	0	576	37,83	0,00%	1,7/sec	1,67	0,57	1021,4
Grupo de Hilos 100:POST - Crear mascota	100	261	0	676	58,28	0,00%	1,7/sec	1,41	1,19	860,0
Grupo de Hilos 100:PUT - Actualizar mascota	100	276	0	792	63,67	0,00%	1,7/sec	1,58	1,49	963,3
Grupo de Hilos 100:PUT - Actualizar propietario	100	252	0	292	15,70	0,00%	1,7/sec	1,65	1,52	1003,8
Total	700	265	0	866	51,57	0,00%	11,4/sec	11,41	6,92	1020,4

Reporte resumen

Nombre: Reporte resumen 500 Hilos

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Grupo de Hilos 500:Login-74	500	6079	0	13083	3233,06	0,00%	12,2/sec	12,22	4,17	1022,3
Grupo de Hilos 500:GET - Listar propietarios	500	14673	0	30822	7824,33	0,00%	7,0/sec	10,39	3,00	1516,0
Grupo de Hilos 500:PUT - Actualizar propietario	500	19499	0	33603	8465,90	0,00%	5,0/sec	4,95	4,57	1003,8
Grupo de Hilos 500:GET - Listar mascotas	500	20965	0	37578	7069,01	0,20%	4,0/sec	5,78	1,95	1463,0
Grupo de Hilos 500:POST - Crear mascota	500	21686	0	34680	5591,54	0,60%	3,4/sec	2,87	2,41	859,7
Grupo de Hilos 500:PUT - Actualizar mascota	500	21141	0	33610	4722,60	0,60%	3,2/sec	3,00	2,85	961,4
Grupo de Hilos 500:DELETE - Eliminar mascota	500	17134	0	33084	6240,84	0,80%	3,2/sec	1,79	1,56	565,4
Total	3500	17311	0	37578	8202,10	0,31%	21,7/sec	22,38	13,12	1055,9

Reporte resumen

Nombre: Reporte resumen 1000 Hilos

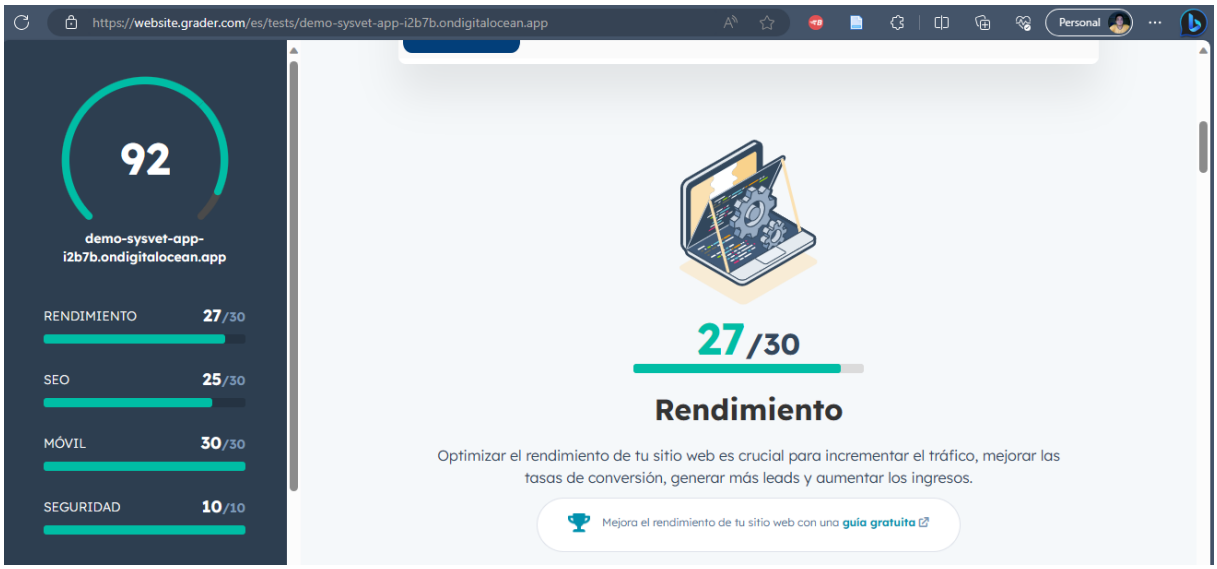
Comentarios

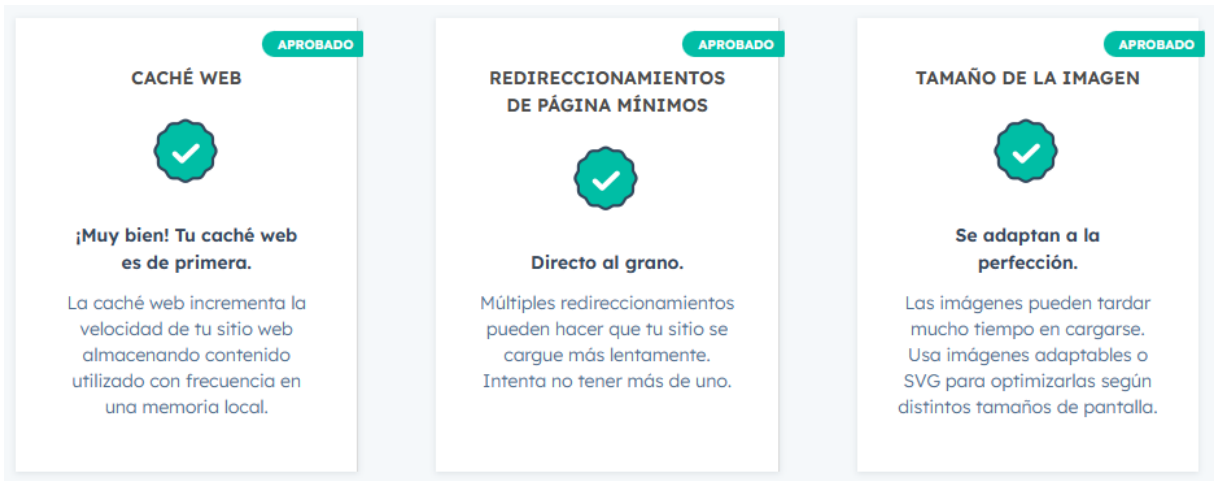
Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Grupo de Hilos 1000:Login-74	1000	22397	0	48575	12512,72	0,70%	16,3/sec	16,29	5,57	1020,3
Grupo de Hilos 1000:GET - Listar propietarios	1000	35692	0	69123	14418,81	1,00%	7,8/sec	11,43	3,30	1506,2
Grupo de Hilos 1000:PUT - Actualizar propietario	1000	45890	0	65900	13172,18	1,80%	5,7/sec	5,60	5,19	997,7
Grupo de Hilos 1000:GET - Listar mascotas	1000	47735	0	78207	10100,95	1,80%	4,4/sec	6,71	2,11	1557,1
Grupo de Hilos 1000:POST - Crear mascota	1000	47514	0	62479	8115,88	2,30%	3,5/sec	2,92	2,46	859,4
Grupo de Hilos 1000:PUT - Actualizar mascota	1000	47080	0	62429	6725,71	3,80%	3,1/sec	2,87	2,74	952,0
Grupo de Hilos 1000:DELETE - Eliminar mascota	1000	40710	0	58791	7724,40	2,90%	3,1/sec	1,69	1,47	564,6
Total	7000	41003	0	78207	13803,70	2,04%	21,1/sec	21,94	12,72	1065,3

Apéndice 3 – Resultados de pruebas de rendimiento de web






APROBADO
JAVASCRIPT MINIMIZADO
Todo en orden.
Cuando tu JavaScript se comprime correctamente, tu sitio web funciona con mucha más rapidez.
APROBADO
CSS MINIMIZADO
Breve, como debe ser.
Cuando tu CSS se comprime correctamente, tu sitio web funciona con mucha más rapidez.